Unterschrift des Betreuers


D I P L O M A R B E I T

# Zuverlässige a posteriori Fehlerschätzung für datengestörte Randelementmethoden in 2D

Ausgeführt am Institut für
Analysis und Scientific Computing
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dipl.Math. Dr.techn. Dirk Praetorius

durch
Isabella Roth
Hermanngasse 22
2700 Wiener Neustadt

Datum

Unterschrift

## Kurzfassung

Die mathematische Beschreibung vieler physikalischer und technischer Prozesse führt zu partiellen Differentialgleichungen, die sich nur in wenigen Spezialfällen analytisch lösen lassen. Man ist also an numerischen Methoden interessiert, die die Lösung der Gleichung bis zu einer gewünschten Genauigkeit berechnen.
Zwei berühmte Beispiele für solche Verfahren sind die Finite Elemente Methode (FEM) und die Randelementmethode (BEM, engl. *boundary element method*). Diese Arbeit beschäftigt sich mit der Randelementmethode. Dabei führt die direkte Randintegralmethode zu elliptischen Integralgleichungen erster Art, auf die dann ein Galerkinverfahren angewendet wird. Wir betrachten im Speziellen die Integralgleichung zum Laplaceoperator mit gegebenen Dirichletdaten $g \in H^{1/2}(\Gamma)$, die so genannte Symmsche Integralgleichung $V\phi = \big(K + \frac{1}{2}\big)g$, die auf dem Rand $\Gamma$ eines Gebiets zu lösen ist.

Bei numerischen Verfahren ist man besonders an einer a posteriori Fehlerschätzung interessiert, das heißt, die Fehlerschätzung darf von der berechneten Näherungslösung abhängen, ist aber unabhängig von der exakten Lösung. Einerseits kann man somit den Approximationsfehler kontrollieren, andererseits ist mit solchen Fehlerschätzern auch eine Steuerung eines adaptiven Algorithmus möglich.

Von besonderem Interesse sind zuverlässige Fehlerschätzer, das heißt, es gibt eine Konstante $C > 0$, sodass für den Schätzer $\varrho_\ell$ gilt $|||\phi - \Phi_\ell||| \leq C\varrho_\ell$. Mit $\phi$ wird hierbei die unbekannte, kontinuierliche, exakte Lösung bezeichnet, $\Phi_\ell$ sei die zugehörige diskrete Galerkinlösung. Geht nämlich ein Fehlerschätzer mit dieser Eigenschaft gegen 0, so konvergiert das Verfahren.

Es ist im Allgemeinen nicht möglich, für beliebige Funktionen $g \in H^{1/2}(\Gamma)$ die Funktion $Kg \in H^{1/2}(\Gamma)$ numerisch zu berechnen. Daher ersetzen wir $g$ durch eine Approximation $g_\ell$, für die $Kg_\ell$ exakt berechnet werden kann. Die „gestörte“ Symmsche Integralgleichung lautet daher $V\widetilde{\phi}_\ell = \big(K + \frac{1}{2}\big)g_\ell$. In diesem Fall bezeichnet $\widetilde{\phi}_\ell$ die unbekannte, kontinuierliche, gestörte Lösung und $\widetilde{\Phi}_\ell$ die entsprechende diskrete Galerkinlösung. Aus diesem Grund steht die Entwicklung eines Fehlerschätzers, der außer dem Verfahrensfehler $|||\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|||$ auch den Datenfehler $|||\phi - \widetilde{\phi}_\ell|||$ berücksichtigt, im Mittelpunkt dieser Arbeit.

Für den Gesamtfehlerschätzer $\omega_\ell$, bestehend aus dem für $|||\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|||$ zuverlässigen Residualschätzer $\varrho_\ell := \big\|h_\ell^{1/2}\big(V\widetilde{\Phi}_\ell - (K + \frac{1}{2})g_\ell\big)'\big\|_{L^2(\Gamma)}$ und einem Datenfehlerschätzer $\zeta_\ell := \big\|h_\ell^{1/2}\big(g - g_\ell\big)'\big\|_{L^2(\Gamma)}$, der $|||\phi - \widetilde{\phi}_\ell||| \leq C_{\mathrm{data}}\zeta_\ell$ erfüllt, wird Zuverlässigkeit bewiesen.

In numerischen Beispielen wird die Zuverlässigkeit dieses Gesamtfehlerschätzers experimentell überprüft und bestätigt. Der Fehler $|||\phi - \Phi_\ell|||$ der „ungestörten“ Symmschen Integralgleichung wurde mit dem Fehler $|||\phi - \widetilde{\Phi}_\ell|||$ der Symmschen Integralgleichung mit approximierten Dirichletdaten verglichen und eine Annäherung der beiden festgestellt. Dazu war es nötig, den in der rechten Seite der Symmschen Integralgleichung vorkommenden Integraloperator $K$ zu implementieren. Die Berechnung der dafür benötigten Einfach- und Doppelintegrale ist in dieser Arbeit genau aufgeführt.

Außerdem wird als zentrales Resultat dieser Arbeit die Konvergenz der aus dem mit dem Gesamtfehlerschätzer $\omega_\ell = \big(\varrho_\ell^2 + \zeta_\ell^2\big)^{1/2}$ gesteuerten adaptiven Algorithmus entstehenden Galerkinlösung $\widetilde{\Phi}_\ell$ gegen die exakte Lösung $\phi$ bewiesen.

**Gewidmet meinem Opa**

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Die mathematische Beschreibung vieler physikalischer und technischer Prozesse führt auf elliptische, partielle Differentialgleichungen. Das analytische Lösen dieser Gleichungen ist nur in seltenen Spezialfällen möglich, und das macht die Entwicklung numerischer Verfahren zur näherungsweisen Lösung notwendig. Diese Verfahren sollen mit gewisser Genauigkeit die exakte Lösung der Differentialgleichung approximieren.

Die Finite Elemente Methode (FEM) und die Randelementmethode (BEM, engl. *boundary element method*) sind berühmte Beispiele solcher Verfahren zur Lösung elliptischer Randwertprobleme. Bei Kenntnis der Fundamentallösung des Differentialoperators entstehen durch Umformung von Randwertproblemen im $d$-dimensionalen Raum Randintegralgleichungen auf der $(d-1)$-dimensionalen Oberfläche, die dann mit Hilfe der BEM gelöst werden können. Dadurch wird der Diskretisierungsaufwand im Vergleich zur FEM stark gesenkt. Die Netzverwaltung und Netzverfeinerung wird vereinfacht und die Anzahl an Freiheitsgraden wesentlich verringert. Eine Diskretisierung führt bei beiden Methoden auf ein lineares Gleichungssystem, das bei der FEM schwach, bei der BEM hingegen voll besetzt ist. Allerdings konvergiert die Randelementmethode asymptotisch schneller als die Methode der Finiten Elemente.

Zwar können mittels BEM auch Außenraumprobleme, d.h. Probleme auf unbeschränkten Gebieten behandelt werden, doch im Gegensatz zur wesentlich verbreiteteren FEM, muss die Fundamentallösung des Differentialoperators explizit bekannt sein, was die Anwendung der Randelementmethode auf Differentialgleichungen mit stückweise konstanten Koeffizienten einschränkt. Doch auch für diese Klasse von Gleichungen gibt es breite Anwendungsmöglichkeiten, beispielsweise in der Elektrostatik, Festkörpermechanik, Elastostatik, Akustik oder Strömungsmechanik.

## 1.1 Modellproblem und Motivation

Sei $\Omega \subseteq \mathbb{R}^2$ ein Gebiet mit hinreichend glattem Rand $\Gamma := \partial\Omega$. Wir betrachten nun das Dirichletrandwertproblem

$$\begin{aligned} -\triangle u &= 0 \text{ in } \Omega \\ u &= g \text{ auf } \Gamma. \end{aligned} \tag{1.1}$$

Ziel ist es dieses Randwertproblem mittels BEM zu lösen.

## 1.2 Aufbau der Arbeit und Kernresultate

Zunächst werden in Kapitel 2 die benötigten Funktionenräume und ihre Eigenschaften vorgestellt, die analytischen Grundlagen gebracht und kurz zusammengefasst, wie man vom gestellten Dirichletproblem (1.1) zur dazu äquivalenten so genannten Symmschen Integralgleichung

$$V\phi = (K + \tfrac{1}{2})g \quad \text{auf } \Gamma \tag{1.2}$$

übergeht. Dabei gehen wir vor wie in Praetorius 2007 [17]. Die wesentlichen Resultate sind Sauter/Schwab 2004 [19] und Steinbach 2004 [21] entnommen.

Gesucht ist also die kontinuierliche Funktion $\phi$ als Lösung der Symmschen Integralgleichung.

Im Allgemeinen ist es nicht möglich, die Funktion $Kg \in H^{1/2}(\Gamma)$ für eine beliebige Funktion $g \in H^{1/2}(\Gamma)$ numerisch zu berechnen. Für glattes $g$ kann man $Kg$ in einen glatten und einen logarithmisch-singulären Teil aufspalten, die dann geeignet numerisch integriert werden. Diese Idee wird in Abschnitt 5 behandelt. Eine andere Möglichkeit ist, $g$ durch eine Approximation $g_\ell$ — beispielsweise die nodale Interpolation — zu ersetzen, für die man $Kg_\ell$ exakt berechnen kann. Das hat auch den Vorteil, dass man den Operator mit geeigneten Matrixkompressionstechniken effizient realisieren kann.

Die kontinuierliche Funktion $\widetilde{\phi}_\ell$ ist Lösung der „datengestörten“ Symmschen Integralgleichung $V\widetilde{\phi}_\ell = (K + \frac{1}{2})g_\ell$. Mit $\Phi_\ell$ bzw. $\widetilde{\Phi}_\ell$ bezeichnen wir die zu $\phi$ und $\widetilde{\phi}_\ell$ gehörigen diskreten Galerkinlösungen.

Ein zentraler Begriff ist die K-Netz-Eigenschaft einer Partition $\mathcal{T}_\ell$: Gibt es eine Konstante $\kappa(\mathcal{T}_\ell) \geq 1$, sodass für $\overline{T_1} \cap \overline{T_2} \neq \emptyset$ gilt $h_\ell(T_1)/h_\ell(T_2) \leq \kappa(\mathcal{T}_\ell)$, so sagt man die Partition $\mathcal{T}_\ell$ erfüllt die K-Netz-Eigenschaft.

Diese wichtige Größe fließt in mehrere Abschätzungen ein. So sind die $h - h/2$-Fehlerschätzer

$$\mu_\ell := \big\|h_\ell^{1/2}(\widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell)\big\|_{L^2(\Gamma)} \quad \text{und} \quad \eta_\ell := \big|\!\big|\!\big|\widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell\big|\!\big|\!\big|$$

nur dann äquivalent — nachzulesen in Ferraz-Leite/Praetorius 2007 [12] —, falls $\sup_{\ell\in\mathbb{N}} \kappa(\mathcal{T}_\ell) < \infty$ gilt. Die mit $\widehat{\cdot}$ gekennzeichneten Galerkinlösungen sind Lösungen über dem Netz $\widehat{\mathcal{T}}_\ell$, das aus $\mathcal{T}_\ell$ durch uniforme Verfeinerung hervorgeht.

Auch die Zuverlässigkeit des Residualschätzers aus Carstensen 1997 [6]

$$C_{\mathrm{rel}}^{-1}\big|\!\big|\!\big|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\big|\!\big|\!\big| \leq \varrho_\ell := \big\|h_\ell^{1/2}\big(V\widetilde{\Phi}_\ell - (K + \tfrac{1}{2})g_\ell\big)'\big\|_{L^2(\Gamma)}$$

ist abhängig von $\kappa(\mathcal{T}_\ell)$, in dem Sinn, dass $C_{\mathrm{rel}}$ mit $\kappa(\mathcal{T}_\ell)$ gegen unendlich strebt.

Die Entwicklung eines Datenfehlerschätzers

$$\zeta_\ell := \big\|h^{1/2}(1 - \Pi_\ell)g'\big\|$$

als obere Schranke für $\big|\!\big|\!\big|\phi - \widetilde{\phi}_\ell\big|\!\big|\!\big|$ stützt sich auf Carstensen/Praetorius 2007 [8]. Diese Bedingung für den Datenfehler und damit ebenso die Konvergenz des adaptiven Verfahrens, beruhen auf der Beschränktheit von $\kappa(\mathcal{T}_\ell)$.

Zusammenfassend sollte also eine geeignete Netzverfeinerungsstrategie zunächst $\sup_{\ell\in\mathbb{N}} \kappa(\mathcal{T}_\ell) < \infty$ erzwingen, damit alle Konstanten in den beschriebenen Abschätzungen beschränkt bleiben.

Zum Auffinden der diskreten Lösung $\widetilde{\Phi}_\ell$ benutzen wir folgenden adaptiven Algorithmus:

**Algorithmus 1.1.** *Input: Startnetz $\mathcal{T}_0$, Markierungsparameter $\theta \in (0,1)$, Abbruchparameter $\varepsilon > 0$.*

*(1) Berechne diskrete Lösung $\widetilde{\Phi}_\ell$.*

*(2) Berechne Fehlerschätzer $\varrho_\ell(T)$ und $\zeta_\ell(T)$ für alle $T \in \mathcal{T}_\ell$.*

*(3) Berechne Verfeinerungsindikator $\omega_\ell(T) = \big(\varrho_\ell(T)^2 + \zeta_\ell(T)^2\big)^{1/2}$ für alle $T \in \mathcal{T}_\ell$.*

*(4) Stoppe, falls $\omega_\ell := \big(\sum_{T\in\mathcal{T}_\ell} \omega_\ell(T)^2\big)^{1/2} \leq \varepsilon$.*

*(5) Bilde Menge $\mathcal{M}_\ell$, der zur Verfeinerung markierten Elemente, durch Markierung nach Dörfler für $\omega_\ell$.*

*(6) Markiere zusätzliche Elemente, damit $\kappa(\mathcal{T}_\ell) \leq 2\kappa(\mathcal{T}_0)$.*

*(7) Verfeinere mindestens die Elemente $T \in \mathcal{M}_\ell$ und erhalte $\mathcal{T}_{\ell+1}$.*

*(8) Erhöhe Zähler $\ell \to \ell + 1$ und gehe zu (1).*

*Output: Approximation $\widetilde{\Phi}_\ell$ von $\phi$.*

Um also $\kappa(\mathcal{T}_\ell) < \infty$ zu garantieren ist die Erweiterung der Markierungsstrategie um Punkt (6) notwendig. Ein entsprechender Beweis findet sich in Kapitel 3, das gemeinsam mit Abschnitt 4 den Mittelpunkt dieser Arbeit darstellt.

In den Kapitel 3 und 4 steht die Theorie der Fehlerschätzer und die Konvergenz der adaptiven Verfahren bei Verwendung des Residualschätzers im Zentrum. Der Beweis für die Zuverlässigkeit des Residualschätzers ist in Abschnitt 3 zu finden. Die Gültigkeit der Bedingung $\vertertertert \phi - \widetilde{\phi}_\ell \vertertertert \lesssim \zeta_\ell$ für den neu entwickelten Datenfehlerschätzer $\zeta_\ell$ wird dort ebenfalls gezeigt. Eine Folge dieser beiden Eigenschaften ist die Zuverlässigkeit des Gesamtfehlerschätzers $\omega_\ell := \big(\varrho_\ell^2 + \zeta_\ell^2\big)^{1/2}$. Das letzte wichtige und neue Resultat in diesem Abschnitt ist die Konvergenz des mit diesem Gesamtfehlerschätzer gesteuerten, adaptiven Verfahrens.

In Abschnitt 4 werden Effizienz und unter der Saturationsannahme auch Zuverlässigkeit der $h - h/2$-Fehlerschätzer gezeigt. Daraus folgt wiederum die Zuverlässigkeit des Gesamtfehlerschätzers $\omega_\ell = \big(\mu_\ell^2 + \zeta_\ell^2\big)^{1/2}$.

Kapitel 5 enthält eine detaillierte Beschreibung der Berechnung des Einfachschichtpotentials $V$, des Doppelschichtpotentials $K$ und der dabei auftretenden einfachen und doppelten Integrale. Auch eine Erklärung zur Berechnung der rechten Seite der Symmschen Integralgleichung sowie die Implementierung des Datenfehlerschätzers $\zeta_\ell$ und des residualen Fehlerschätzers $\varrho_\ell$ werden angegeben.

Die theoretischen Grundlagen und Berechnungen der auftretenden Operatoren sind hier nur für den zweidimensionalen Fall aufgeführt. In Kapitel 6 wird daher zusammengefasst, was sich für den Fall $d = 3$ ändert.

Zur Überprüfung der theoretischen Resultate wurden auch numerische Experimente durchgeführt. Anhand dreier Beispiele konnte die mathematisch bewiesenen Zuverlässigkeit des Gesamtfehlerschätzers $\omega_\ell := \big(\varrho_\ell^2 + \zeta_\ell^2\big)^{1/2}$ und die Beschränktheit von $\kappa(\mathcal{T}_\ell)$ bestätigt werden. Außerdem beobachten wir, dass aus numerischer Sicht auch der $h - h/2$-Fehlerschätzer geeignet ist zur Steuerung des adaptiven Algorithmus, was den Implementierungs- und Rechenaufwand erheblich senkt.

## 1.3 Danksagung

Zuerst möchte ich mich bei meinen Eltern Christa und Manfred Roth und meiner Schwester Florentina dafür bedanken, dass sie mich stets in jeder Hinsicht in der Verwirklichung meiner Träume unterstützt haben.
Meinem Lebensgefährten Ernst Moser gilt mein Dank dafür, dass er immer an meiner Seite steht und mir Rückhalt bietet.

Außerdem möchte ich meinem Betreuer Dirk Praetorius für seine fachliche Hilfe, sein großes Engagement und seine Ehrlichkeit danken. Ich habe viel von ihm gelernt und freue mich, dies auch weiterhin tun zu dürfen. Mein Studienkollege Samuel Ferraz-Leite stand mir stets mit Rat und Tat zur Seite. Ihm bin ich besonders für seine grenzenlose Geduld zu Dank verpflichtet.

Abschließend möchte ich mich noch bei meinem Opa bedanken, der mit über 70 Jahren noch mit mir Maturabeispiele gerechnet und mein Interesse an einem technischen Studium geweckt hat.

Wiener Neustadt, am 2. Januar 2009

# Kapitel 2

# Die Randelementmethode

In diesem Kapitel werden die analytischen Grundlagen für die Randelementmethode gesammelt.

Im Wesentlichen werden hier bereits bekannte Ergebnisse aus der Literatur zusammengetragen, vor allem aus Steinbach 2004 [21] und Sauter/Schwab 2004 [19].

Der inhaltliche Aufbau dieses Kapitels folgt dabei Praetorius 2007 [17], ist aber in einigen Bereichen ausführlicher.

Zunächst wird für das Modellproblem die so genannte Darstellungsformel aufgestellt. Als geeignete Lösungsräume der schwachen Formulierung der Poissongleichung werden Sobolevräume eingeführt und ihre wichtigsten Eigenschaften beschrieben. Die in der Darstellungsformel auftretenden Integraloperatoren und die zugehörigen, im Caldéron-System vorkommenden Randintegraloperatoren werden vorgestellt. Der nächste Abschnitt beschäftigt sich mit der Galerkin-Randelementmethode und deren Eigenschaften. Die a priori Analysis, die eine Aussage über die Konvergenzgeschwindigkeit der BEM trifft, wird am Ende dieses Kapitels behandelt.

## 2.1 Darstellungsformel

In diesem Abschnitt wird die so genannte Darstellungsformel (oder 3. Greensche Formel) vorgestellt, die besagt, dass die Lösung des Laplace-Problems $-\triangle u = f$ eindeutig durch die Cauchydaten $(u, \partial_n u)$ bestimmt ist.

Dazu definieren wir zunächst den so genannten Newton-Kern wie in Steinbach 2004 [21, Kapitel 5.1, Seite 94].

**Definition 2.1.** *Die Fundamentallösung des Laplaceoperators ist für $z \in \mathbb{R}^2$ durch*

$$G(z) := -\frac{1}{2\pi} \log |z| \tag{2.1}$$

*gegeben und wird Newton-Kern genannt.*

Wir definieren nun drei Integraloperatoren, die in der Darstellungsformel Verwendung finden.

**Definition 2.2.** *Zu einer Funktion $f : \Omega \to \mathbb{R}$ definiert $\widetilde{N}f$ mit*

$$\widetilde{N}f(x) := \int_\Omega G(x-y)f(y)\,dy \quad \text{für } x \in \Omega \tag{2.2}$$

*das Newtonpotential der Funktion $f$.*

*Zu einer Funktion $\phi : \Gamma \to \mathbb{R}$ definiert $\widetilde{V}\phi$ mit*

$$\widetilde{V}\phi(x) := \int_\Gamma G(x-y)\phi(y)\,ds_y \quad \text{für } x \in \Omega \tag{2.3}$$

*das Einfachschichtpotential der Funktion $\phi$.*

*Zu einer Funktion $g : \Gamma \to \mathbb{R}$ definiert $\widetilde{K}g$ mit*

$$\widetilde{K}g(x) := \int_\Gamma \partial_{n(y)}G(x-y)g(y)\,ds_y \quad \text{für } x \in \Omega \tag{2.4}$$

*das Doppelschichtpotential der Funktion $g$.*

Die Darstellungsformel für die Laplacegleichung lautet nun nach Steinbach 2004 [21, Kapitel 6., Seite 106ff] folgendermaßen:

**Satz 2.3.** *Sei $\Omega \subset \mathbb{R}^d$ ein beschränktes Gebiet mit Rand $\Gamma := \partial\Omega$ und $u \in \mathcal{C}^2(\overline{\Omega})$. Mit $-\triangle u = f \in \mathcal{C}(\overline{\Omega})$ gilt*

$$u(x) = \widetilde{N}(-\triangle u) + \widetilde{V}(\partial_n u) - \widetilde{K}(u) \quad \text{in } \Omega. \tag{2.5}$$

□

## 2.2 Sobolevräume und Variationsformulierung

In diesem Abschnitt fassen wir die funktionalanalytischen Grundlagen zusammen, um die Variationsformulierung der Laplacegleichung herzuleiten.

Sei $u \in \mathcal{C}^2(\overline{\Omega})$ die klassische Lösung von

$$-\triangle u = f. \tag{2.6}$$

Wir multiplizieren die Gleichung mit einer Testfunktion $v \in \mathcal{C}^1(\overline{\Omega})$ und integrieren über $\Omega$. Mittels partieller Integration erhalten wir dann für alle $v \in \mathcal{C}^1(\overline{\Omega})$

$$\langle f, v\rangle_\Omega = \langle \nabla u, \nabla v\rangle_\Omega - \langle \partial_n u, v\rangle_\Gamma,$$

bzw. nach Umformung

$$\langle \nabla u, \nabla v\rangle_\Omega = \langle f, v\rangle_\Omega + \langle \partial_n u, v\rangle_\Gamma. \tag{2.7}$$

Die Suche nach geeigneten Lösungsräumen führt nun zu den Sobolevräumen. Dazu benötigen wir zunächst einige Definitionen aus Steinbach 2004 [21, Kapitel 2.2, Seite 28ff].

**Definition 2.4.** *Eine Funktion $u \in L^1_{\text{loc}}(\Omega)$, die bezüglich jeder abgeschlossenen und beschränkten Teilmenge $M \subset \Omega$ integrierbar ist, heißt lokal integrierbar.*

**Definition 2.5.** *Eine Funktion $u \in L^1_{\text{loc}}(\Omega)$ heißt schwach differenzierbar mit schwacher Ableitung $\partial_j u \in L^1_{\text{loc}}(\Omega)$ für $j = 1, \ldots, d$, wenn die Formel der partiellen Integration gilt, d.h $(u, \partial_j u)$ erfüllen*

$$\int_\Omega u(\partial_j v)\, dx = -\int_\Omega (\partial_j u) v\, dx \quad \text{für alle } v \in \mathcal{C}_c^\infty(\Omega). \tag{2.8}$$

In diesem Fall ist $\partial_j u$ eindeutig bestimmt und stimmt für eine $\mathcal{C}^1$-Funktion mit der klassischen Ableitung überein. Nun können wir Sobolevräume auf Gebieten definieren.

**Definition 2.6.** *Der Sobolevraum $H^0(\Omega)$ wird als $L^2(\Omega)$ definiert.*

$$H^0(\Omega) := L^2(\Omega). \tag{2.9}$$

*Weiters definiert man*

$$H^1(\Omega) := \left\{u \in L^2(\Omega) | u \text{ ist schwach differenzierbar mit } \nabla u \in L^2(\Omega)^d\right\}. \tag{2.10}$$

*Die zugehörige Norm lautet $\|u\|^2_{H^1(\Omega)} = \|u\|^2_{L^2(\Omega)} + \|\nabla u\|^2_{L^2(\Omega)}$.*

*Die weiteren Sobolevräume ganzzahliger Ordnung werden induktiv definiert durch*

$$H^k(\Omega) := \left\{u \in L^2(\Omega) | u \text{ ist schwach differenzierbar mit } \nabla u \in H^{k-1}(\Omega)\right\} \tag{2.11}$$

*mit der Norm $\|u\|^2_{H^k(\Omega)} = \|u\|^2_{L^2(\Omega)} + \|\nabla u\|^2_{H^{k-1}(\Omega)}$. Auf $H^k(\Omega)$ ist*

$$|u|_{H^k(\Omega)} := \|D^k u\|_{L^2(\Omega)} \tag{2.12}$$

*eine Seminorm. $D^k u$ ist hierbei die $k$-te schwache Ableitung von $u$.*

**Definition 2.7.** *Für reelle Zahlen $0 < s < 1$ ist die so genannte Sobolev-Slobodeckij Seminorm gegeben durch*

$$|u|_{s,\Omega} := \left(\int_\Omega \int_\Omega \frac{|u(x) - u(y)|^2}{|x - y|^{d+2s}}\right)^{1/2}. \tag{2.13}$$

Mit Hilfe der Sobolev-Slobodeckij Seminorm können Sobolevräume mit nicht ganzzahliger Ordnung definiert werden.

**Definition 2.8.** *Sobolevräume mit nicht ganzzahliger Ordnung werden für $k \in \mathbb{N}_0$ und $0 < s < 1$ definiert durch*

$$H^{k+s}(\Omega) := \{u \in H^k(\Omega) \big| |D^k u|_{s,\Omega} < \infty\} \tag{2.14}$$

*mit der Norm $\|u\|^2_{H^{k+s}(\Omega)} := \|u\|^2_{H^k(\Omega)} + |D^k u|^2_{s,\Omega}$.*

**Satz 2.9. (Evans 1998 [10, Kapitel 5.2, Theorem 2]).** *Für $s \geq 0$ ist der Sobolevraum $H^s(\Omega)$ ein Hilbertraum.* □

**Satz 2.10. (Wloka 1982 [23, Kapitel 3.2, Satz 3.6]).** *Alle Räume $C^k(\overline{\Omega})$ mit $k \geq s$ liegen dicht in $H^s(\Omega)$.* □

Der erste Summand der rechten Seite von (2.7)

$$v \mapsto \langle f, v\rangle_\Omega$$

definiert ein Funktional in $H^1(\Omega)^*$.

Um den Dualraum $H^1(\Omega)^*$ von $H^1(\Omega)$ als Erweiterung von $L^2(\Omega)$ definieren zu können, benötigen wir zunächst noch folgendes Lemma:

**Lemma 2.11.** *Seien $X, Y$ zwei reelle Hilberträume mit stetiger Einbettung $X \subseteq Y$. Dann ist die Rieszabbildung*

$$J_Y : Y \hookrightarrow Y^*, \quad J_Y y := \langle y, \cdot\rangle_Y \tag{2.15}$$

*als Operator $J_Y \in L(Y, X^*)$ wohldefiniert und $J_Y(Y)$ ist ein dichter Unterraum von $X^*$.*

**Beweis:** Da $X$ in $Y$ stetig eingebettet ist gilt $\|x\|_Y \leq C\|x\|_X$ für beliebiges $x \in X$. Aus der Cauchyungleichung folgt

$$\langle y, x\rangle_Y \leq \|y\|_Y \|x\|_Y \leq C\|y\|_Y \|x\|_X.$$

Die Rieszabbildung ist also wohldefiniert.

Sei $J_X : X \hookrightarrow X^*$ die Rieszabbildung für $X$. Dann ist $J_Y(Y)$ genau dann dicht in $X^*$, wenn $V := J_X^{-1}(J_Y(Y))$ dicht in $X = \overline{V} \oplus \overline{V}^\perp$ ist. Man muss also nur noch $\overline{V}^\perp = \{0\}$ zeigen.

Sei $x \in \overline{V}^\perp$ und $y \in Y$, dann gilt

$$0 = \langle x, J_X^{-1}(J_y y)\rangle_X = (J_y y)(x) = \langle y, x\rangle_Y.$$

Wählt man $y = x \in \overline{V}^\perp$ folgt $x = 0$ in $Y \supseteq X$. □

Wendet man nun Lemma 2.11 an für $X = H^s(\Omega)$ und $Y = L^2(\Omega)$, erkennt man, dass $L^2(\Omega)$ ein dichter Teilraum von $H^s(\Omega)^*$ ist und die duale Klammer $\langle f, v\rangle_\Omega$ mit dem $L^2$-Skalarprodukt übereinstimmt, falls $f \in L^2(\Omega)$ gilt.

Das führt zur Definition von Sobolevräumen mit negativer Ordnung, wie folgt:

**Definition 2.12.** *Der bezüglich des erweiterten $L^2$-Skalarprodukts dargestellte Dualraum von $H^s(\Omega)$ wird mit $\widetilde{H}^{-s}(\Omega)$ bezeichnet. Man nennt diese Räume Sobolevräume der Ordnung $-s$.*

Für den zweiten Summanden von (2.7) müssen nun auch Sobolevräume am Rand von Gebieten definiert werden wie in Steinbach 2004 [21, Kapitel 2.5, Seite 42].

**Definition 2.13.** *Für $0 < s < 1$ definiere*

$$H^s(\Gamma) := \{u \in L^2(\Gamma) \big| \|u\|_{H^s(\Gamma)} < \infty\} \tag{2.16}$$

*mit der Norm $\|u\|^2_{H^s(\Gamma)} := \|u\|^2_{L^2(\Gamma)} + |u|_{s,\Gamma}$, wobei*

$$|u|_{s,\Gamma} := \left( \int_\Gamma \int_\Gamma \frac{|u(x) - u(y)|^2}{|x - y|^{d-1+2s}} \right)^{1/2}. \tag{2.17}$$

*die Sobolev-Slobodeckij Seminorm bezeichne.*

**Satz 2.14. (Sauter/Schwab 2004 [19, Kapitel 2.5, Seite 46ff]).** *$H^{1/2}(\Gamma)$ ist ein Hilbertraum mit stetiger Einbettung $H^{1/2}(\Gamma) \subset L^2(\Gamma)$.* □

**Definition 2.15.** *Der bezüglich des erweiterten $L^2$-Skalarprodukts dargestellte Dualraum von $H^s(\Gamma)$ wird mit $H^{-s}(\Gamma)$ bezeichnet.*

Im folgenden Satz soll erklärt werden, wie man vom Gebietsraum zum Randraum gelangt.

**Satz 2.16. (Evans 1998 [10, Kapitel 5.5, Theorem 1]).** *Der Spuroperator $\gamma_0 u = u|_\Gamma$ für $u \in \mathcal{C}^\infty(\overline{\Omega})$ lässt sich eindeutig von $\mathcal{C}^\infty(\overline{\Omega})$ zu einem Operator $\gamma_0 \in L\big(H^1(\Omega); H^{1/2}(\Gamma)\big)$ fortsetzen. Das bedeutet, dass $\gamma_0 u$ die Einschränkung einer Sobolevfunktion $u \in H^1(\Omega)$ auf $\Gamma$ darstellt.* □

Nun wollen wir noch näher auf die Konormalenableitung, die im zweiten Skalarprodukt der rechten Seite von (2.7) vorkommt, eingehen und die Variationsformulierung formal korrekt aufschreiben.

**Definition 2.17.** *Man sagt, eine Sobolev Funktion $u \in H^1(\Omega)$ löst für gegebenes $f \in \widetilde{H}^{-1}(\Omega)$*

$$-\triangle u = f \in \widetilde{H}^{-1}(\Omega), \tag{2.18}$$

*wenn es ein Funktional $\gamma_1 u \in H^{-1/2}(\Gamma)$ gibt, sodass die schwache Formulierung von* (2.6)

$$\langle \nabla u, \nabla v \rangle_\Omega = \langle f, v \rangle_\Omega + \langle \gamma_1 u, \gamma_0 u \rangle_\Gamma \quad \text{für alle } v \in H^1(\Omega) \tag{2.19}$$

*gilt.*

Wir wollen das Funktional $\gamma_1 u$ nun näher charakterisieren:

**Satz 2.18. (Praetorius 2007 [17, Kapitel 2.4, Lemma 7 und 8]).**

*(i) Für eine Funktion $u \in C^2(\overline{\Gamma})$ und $f := -\triangle u$ gilt* (2.19) *mit $\gamma_1 u = \partial_n u$.*

*(ii) Ist $u \in H^1(\Omega)$ und löst* (2.18)*, so ist die Konormalenableitung $\gamma_1 u \in H^{-1/2}(\Gamma)$ aus* (2.19) *eindeutig bestimmt.* □

Mit den eingeführten Notationen lässt sich folgendes Resultat beweisen:

**Korollar 2.19.** *Für gegebenes $f \in \widetilde{H}^{-1}(\Omega)$ und $g \in H^{1/2}(\Gamma)$, gibt es eine eindeutige Lösung $u \in H^1(\Omega)$ der Variationsformulierung* (2.19) *mit Dirichlet Randbedingungen $\gamma_0 u = g$.* □

## 2.3 Calderónsystem und Integraloperatoren

In diesem Kapitel wird das so genannte Calderónsystem hergeleitet und die wichtigsten Eigenschaften der verschiedenen Randintegraloperatoren beschrieben.
Für das betrachtete Gebiet $\Omega$ gelte im Folgenden $\operatorname{diam}(\Omega) < 1$. Das ist eine hinreichende Bedingung dafür, dass für die logarithmische Kapazität Cap gilt $\operatorname{Cap} < 1$ — siehe McLean 2000 [15, Kapitel 8, Theorem 8.16].

Um die Darstellungsformel

$$u = \widetilde{N} f + \widetilde{V}(\gamma_1 u) - \widetilde{K}(\gamma_0 u) \tag{2.20}$$

im schwachen Sinn für $u \in H^1(\Omega)$ zu verstehen, halten wir zunächst Abbildungseigenschaften der Integraloperatoren fest.

**Satz 2.20. (Steinbach 2004 [21, Kapitel 6.1]).** *$\widetilde{N}$ kann als Operator $\widetilde{N} \in L(\widetilde{H}^{-1}(\Omega); H^1(\Omega))$ aufgefasst werden, und es gilt*

$$-\Delta(\widetilde{N}f) = f \quad \text{für alle } f \in \widetilde{H}^{-1}(\Omega). \tag{2.21}$$

*Insbesondere sind Spur und Konormalenableitung als Operatoren*

$$\gamma_0\widetilde{N} \in L(\widetilde{H}^{-1}(\Omega); H^{1/2}(\Gamma)) \quad \text{und} \quad \gamma_1\widetilde{N} \in L(\widetilde{H}^{-1}(\Omega); H^{-1/2}(\Gamma)) \tag{2.22}$$

*wohldefiniert.* □

**Satz 2.21. (Steinbach 2004 [21, Kapitel 6.2]).** *$\widetilde{V}$ kann als Operator $\widetilde{V} \in L(H^{-1/2}(\Gamma); H^1(\Omega))$ aufgefasst werden, und es gilt*

$$-\Delta\widetilde{V}\phi = 0 \in \widetilde{H}^{-1}(\Omega) \quad \text{für alle } \phi \in H^{-1/2}(\Gamma). \tag{2.23}$$

*Insbesondere sind Spur und Konormalenableitung als Operatoren*

$$\gamma_0\widetilde{V} \in L(H^{-1/2}(\Gamma); H^{1/2}(\Gamma)) \quad \text{und} \quad \gamma_1\widetilde{V} \in L(H^{-1/2}(\Gamma); H^{-1/2}(\Gamma)) \tag{2.24}$$

*wohldefiniert.* □

**Satz 2.22. (Steinbach 2004 [21, Kapitel 6.4]).** *$\widetilde{K}$ kann als Operator $\widetilde{K} \in L(H^{1/2}(\Gamma); H^1(\Omega))$ aufgefasst werden, und es gilt*

$$-\Delta\widetilde{K}g = 0 \in \widetilde{H}^{-1}(\Omega) \quad \text{für alle } g \in H^{1/2}(\Gamma). \tag{2.25}$$

*Insbesondere sind Spur und Konormalenableitung als Operatoren*

$$\gamma_0\widetilde{K} \in L(H^{1/2}(\Gamma); H^{1/2}(\Gamma)) \quad \text{und} \quad \gamma_1\widetilde{K} \in L(H^{1/2}(\Gamma); H^{-1/2}(\Gamma)) \tag{2.26}$$

*wohldefiniert.* □

Wendet man nun auf die Darstellungsformel (2.20) den Spuroperator $\gamma_0$ bzw. die Konormalenableitung $\gamma_1$ an, so erhält man das Calderónsystem:

$$\begin{pmatrix}\gamma_0 u\\ \gamma_1 u\end{pmatrix} = \begin{pmatrix}-\gamma_0\widetilde{K} & \gamma_0\widetilde{V}\\ -\gamma_1\widetilde{K} & \gamma_1\widetilde{V}\end{pmatrix}\begin{pmatrix}\gamma_0 u\\ \gamma_1 u\end{pmatrix} + \begin{pmatrix}\gamma_0\widetilde{N}f\\ \gamma_1\widetilde{N}f\end{pmatrix}$$

Eine abkürzende Schreibweise führt zu den Randintegraloperatoren:

**Definition 2.23.** *Der Randintegraloperator $N_0 \in L\big(\widetilde{H}^{-1}(\Omega), H^{1/2}(\Gamma)\big)$ wird definiert als*

$$N_0 := \gamma_0\widetilde{N}. \tag{2.27}$$

*Der Randintegraloperator $N_1 \in L\big(\widetilde{H}^{-1}(\Omega), H^{-1/2}(\Gamma)\big)$ wird definiert als*

$$N_1 := \gamma_1\widetilde{N}. \tag{2.28}$$

*Der Randintegraloperator* $V \in L\big(H^{-1/2}(\Gamma), H^{1/2}(\Gamma)\big)$ *wird definiert als*

$$V := \gamma_0 \widetilde{V} \tag{2.29}$$

*und wird Einfachschichtpotential genannt.*

*Der Randintegraloperator* $K' \in L\big(H^{-1/2}(\Gamma), H^{-1/2}(\Gamma)\big)$ *wird definiert als*

$$K' := \gamma_1 \widetilde{V} - \frac{1}{2} \tag{2.30}$$

*und wird adjungiertes Doppelschichtpotential genannt.*

*Der Randintegraloperator* $K \in L\big(H^{1/2}(\Gamma), H^{1/2}(\Gamma)\big)$ *wird definiert als*

$$K := \gamma_0 \widetilde{K} + \frac{1}{2} \tag{2.31}$$

*und wird Doppelschichtpotential genannt.*

*Der Randintegraloperator* $W \in L\big(H^{1/2}(\Gamma), H^{-1/2}(\Gamma)\big)$ *wird definiert als*

$$W := -\gamma_1 \widetilde{K} \tag{2.32}$$

*und wird hypersingulärer Integraloperator genannt.*

Das Calderón System sieht dann in abgekürzter Schreibweise so aus:

$$\begin{pmatrix} \gamma_0 u \\ \gamma_1 u \end{pmatrix} = \begin{pmatrix} \frac{1}{2} - K & V \\ W & \frac{1}{2} + K' \end{pmatrix} \begin{pmatrix} \gamma_0 u \\ \gamma_1 u \end{pmatrix} + \begin{pmatrix} N_0 f \\ N_1 f \end{pmatrix} \tag{2.33}$$

Im Folgenden wollen wir einige wichtige Eigenschaften der obigen Integraloperatoren festhalten.

Das Einfachschichtpotential erfüllt folgende Bedingungen:

**Satz 2.24. (Steinbach 2004 [21, Kapitel 6.2, Seite 113ff und Kapitel 6.6.1, Satz 6.6]).** *Für das Einfachschichtpotential* $V : H^{-1/2+s}(\Gamma) \to H^{1/2+s}(\Gamma)$, $|s| \leq 1/2$, *gilt:*

*(i)* $V$ *ist wohldefiniert, linear und stetig.*

*(ii)* $V$ *ist symmetrisch, d.h*

$$\langle V\phi, \psi\rangle_\Gamma = \langle \phi, V\psi\rangle_\Gamma \quad \textit{für alle } \phi, \psi \in H^{-1/2}(\Gamma). \tag{2.34}$$

*(iii)* $V$ *ist* $H^{-1/2}(\Gamma)$*-elliptisch, d.h es gibt eine so genannte Elliptizitätskonstante* $c_V > 0$ *mit*

$$\langle V\phi, \phi\rangle_\Gamma \geq c_V \|\phi\|^2_{H^{-1/2}(\Gamma)} \quad \textit{für alle } \phi \in H^{-1/2}(\Gamma), \tag{2.35}$$

*insbesondere ist* $V$ *also ein Isomorphismus.* □

**Bemerkung 2.25.** *Die Generalvoraussetzung für dieses Kapitel, nämlich* $\operatorname{diam}(\Omega) < 1$*, geht in diesen Satz ein. Sie ist notwendig für die Elliptizität von* $V$ *und damit auch für die Invertierbarkeit von* $V$*.*

Der Operator $N_1$ lässt sich mit Hilfe der anderen Operatoren berechnen, siehe Formel (2.33) und Satz 2.24(iii).

**Korollar 2.26. (Steinbach 2004 [21, Kapitel 6.6, Lemma 6.13]).**
*Für das Newtonpotential $(N_1 f)(x)$, $x \in \Gamma$, gilt die Darstellung*

$$(N_1 f)(x) = \Big( -\frac{1}{2} + K' \Big) V^{-1} (N_0 f)(x). \tag{2.36}$$

□

Aus den Aussagen von Satz 2.24 lässt sich ein äquivalentes Skalarprodukt auf $H^{-1/2}(\Gamma)$ herleiten.

**Korollar 2.27.** *Für $\phi, \psi \in H^{-1/2}(\Gamma)$ definiert*

$$\langle\langle \phi, \psi \rangle\rangle := \langle V\phi, \psi \rangle_\Gamma \tag{2.37}$$

*ein äquivalentes Skalarprodukt auf $H^{-1/2}(\Gamma)$. Die induzierte Norm, die so genannte Energienorm,*

$$|||\phi||| := \langle\langle \phi, \phi \rangle\rangle^{1/2} \tag{2.38}$$

*ist eine äquivalente Norm auf $H^{-1/2}(\Gamma)$.*

**Beweis:** Man betrachte $|||\phi|||^2 = \langle V\phi, \phi \rangle_\Gamma$. Nach Definition der Dualität zwischen $H^{1/2}(\Gamma)$ und $H^{-1/2}(\Gamma)$ gilt

$$\langle V\phi, \phi \rangle_\Gamma \leq \|V\phi\|_{H^{1/2}(\Gamma)} \, \|\phi\|_{H^{-1/2}(\Gamma)}.$$

Aus der Stetigkeit folgt

$$\langle V\phi, \phi \rangle_\Gamma \leq \|V\| \, \|\phi\|^2_{H^{-1/2}(\Gamma)}.$$

Andererseits gibt es aufgrund der Elliptizität von $V$ eine Konstante $c_V > 0$, sodass

$$\langle V\phi, \phi \rangle_\Gamma \geq c_V \|\phi\|^2_{H^{-1/2}(\Gamma)}$$

erfüllt ist und die beiden Normen somit äquivalent sind. □

Das Doppelschichtpotential hat folgende Eigenschaften:

**Satz 2.28. (Steinbach 2004 [21, Kapitel 6.4, Seite 118ff und Kapitel 6.6.4, Seite 144]).** *Für das Doppelschichtpotential $K : H^{1/2+s}(\Gamma) \to H^{1/2+s}(\Gamma)$, $|s| \leq 1/2$, gilt:*

*(i) $K$ ist wohldefiniert, linear und stetig.*

*(ii) Ist $V$ invertierbar, so ist $K + \frac{1}{2}$ eine Kontraktion bezüglich der auf $H^{1/2}(\Gamma)$ äquivalenten Norm $\|g\|^2_{V^{-1}} := \langle V^{-1} g, g \rangle_\Gamma$, d.h es gilt*

$$\|(K + \tfrac{1}{2})g\|_{V^{-1}} \leq c_K \|g\|_{V^{-1}} \quad \text{für alle } g \in H^{1/2}(\Gamma) \tag{2.39}$$

*mit einer geeigneten Kontraktionskonstanten $c_K \in (0, 1)$.* □

Das adjungierte Doppelschichtpotential $K'$ ist im $L^2$-Sinn zum Doppelschichtpotential $K$ adjungiert.

**Satz 2.29. (McLean 2000 [15, Kapitel 6, Theorem 6.15]).**
*Für die Operatoren $K : H^{1/2}(\Gamma) \to H^{1/2}(\Gamma)$ und $K' : H^{-1/2}(\Gamma) \to H^{-1/2}(\Gamma)$ und Funktionen $g \in H^{1/2}(\Omega)$, $\psi \in H^{-1/2}(\Gamma)$ gilt*

$$\langle Kg, \psi \rangle_\Gamma = \langle g, K'\psi \rangle_\Gamma, \tag{2.40}$$

*d.h. $K$ und $K'$ sind adjungiert im Sinne der Funktionalanalysis.* □

## 2.4 Galerkin-Randelementmethode

In diesem Abschnitt wollen wir die Symmsche Integralgleichung und die so genannte Galerkin-Randelementmethode zum Lösen von Integralgleichungen vorstellen.

Zur besseren Übersicht geben wir hier nochmals das Modellproblem an:

Gesucht ist die schwache Lösung $u \in H^1(\Omega)$ von

$$\begin{aligned} -\triangle u &= f \in \widetilde{H}^{-1}(\Omega), \\ \gamma_0 u &= g \in H^{1/2}(\Gamma). \end{aligned} \tag{2.41}$$

Die schwache Formulierung des Problems lässt sich mittels der Darstellungsformel schreiben als

$$u = \widetilde{N} f + \widetilde{V}\phi - \widetilde{K} g$$

mit den unbekannten Neumanndaten $\phi = \gamma_1 u \in H^{-1/2}(\Gamma)$.

Die erste Zeile des Calderón Systems (2.33) ergibt

$$\gamma_0 u = \Big(\frac{1}{2} - K\Big)\gamma_0 u + V\gamma_1 u + N_0 f$$

und nach Umformung

$$V\gamma_1 u = \Big(K + \frac{1}{2}\Big)\gamma_0 u - N_0 f.$$

Einsetzen der Dirichlet- und gesuchten Neumanndaten, d.h $\gamma_0 u = g$ und $\gamma_1 u = \phi$, führt zur so genannten Symmschen Integralgleichung

$$V\phi = \Big(K + \frac{1}{2}\Big) g - N_0 f =: F. \tag{2.42}$$

Anwenden des Satzes von Riesz ergibt eine zur Symmschen Integralgleichung (2.42) äquivalente Variationsformulierung

$$\langle\!\langle \phi, \psi \rangle\!\rangle = \langle F, \psi \rangle_\Gamma \quad \text{für alle } \psi \in H^{-1/2}(\Gamma). \tag{2.43}$$

**Satz 2.30.** *Die Variationsformulierung* (2.43) *mit* $F \in H^{1/2}(\Gamma)$ *aus* (2.42) *hat eine eindeutige Lösung, und es gilt* $\phi = \gamma_1 u$ *für die Lösung* $u \in H^1(\Omega)$ *von* (2.41). □

Die Idee der Galerkin-Randelementmethode besteht nun darin, $H^{-1/2}(\Gamma)$ für $\phi$ und $\psi$ durch einen endlichdimensionalen Ansatzraum zu ersetzen.

Um geeignete diskrete Räume zu konstruieren, wollen wir zunächst das Konzept einer Unterteilung des Randes $\Gamma$ in endlich viele Teilstücke erklären.

Dazu definieren wir wie in Sauter/Schwab 2004 [19, Kapitel 4.1.2]).

**Definition 2.31.** *Eine Unterteilung des Randes* $\Gamma = \partial\Omega$, $\Omega \subseteq \mathbb{R}^2$, *in endlich viele, relativ offene, disjunkte Elemente* $T_1, \ldots, T_n \subseteq \Gamma$ *heißt Partition* $\mathcal{T}_\ell$, *falls die folgenden Bedingungen erfüllt sind:*

*(i)* $\mathcal{T}_\ell$ *ist eine endliche Überdeckung von* $\Gamma$*, d.h* $\Gamma = \overline{\bigcup_{T\in\mathcal{T}_\ell} T}$*.*

*(ii) Jedes Element* $T \in \mathcal{T}_\ell$ *ist Bild des Referenzelementes* $\widehat{T} = [-1,1]$ *unter einer regulären Abbildung* $\gamma_T$*, d.h es gibt Konstanten* $\lambda_{\min} > 0$ *und* $\lambda_{\max} > 0$ *mit*

$$0 < \lambda_{\min} \leq \|\dot{\gamma}_T(s)\| \leq \lambda_{\max} < \infty \quad \textit{für alle } s \in (-1;1). \tag{2.44}$$

**Definition 2.32.** *Für jedes Element* $T$ *der Partition* $\mathcal{T}_\ell$ *ist die lokale Netzweite* $h_\ell \in L^\infty(\Gamma)$ *durch* $h_\ell|_T := h_\ell(T) := \operatorname{diam}(T)$ *definiert.*
*Die globale Netzweite einer Partition* $\mathcal{T}_\ell$ *ist gegeben durch* $h_{\ell,\max} := \max_{T\in\mathcal{T}_\ell} h_\ell(T)$*.*

**Definition 2.33.** *Man sagt eine Partition* $\mathcal{T}_\ell$ *erfüllt die K-Netz-Eigenschaft, falls es eine Konstante* $\kappa(\mathcal{T}_\ell) \geq 1$ *gibt, sodass für* $\overline{T_1} \cap \overline{T_2} \neq \emptyset$ *gilt* $h_\ell(T_1)/h_\ell(T_2) \leq \kappa(\mathcal{T}_\ell)$*.*

**Definition 2.34.** *Man nennt eine Familie von Partitionen* $\{\mathcal{T}_\ell\}_{\ell\in\mathbb{N}}$ *lokal quasiuniform, falls Konstanten* $h_{\ell,\min}, h_{\ell,\max} > 0$ *existieren, sodass für alle* $T \in \mathcal{T}_\ell$ *gilt* $h_{\ell,\min} \leq h_\ell(T) \leq h_{\ell,\max}$ *und* $\sup_\ell \frac{h_{\ell,\max}}{h_{\ell,\min}} =: H < \infty$*.*

Für die Implementierung werden nur Partitionen, die die K-Netz Eigenschaft mit $\kappa(\mathcal{T}_\ell) \leq 2$ erfüllen, betrachtet und stets stückweise affine Randelemente verwendet, da sich die entstehenden Integrale dann analytisch berechnen lassen.

Unser Ziel ist es, einen geeigneten endlichdimensionalen Ansatz für die unbekannte Lösung $\phi$ zu finden. Bei der Galerkin-Randelementmethode erreicht man dies, indem man in der Variationsformulierung nur stückweise Polynome als Ansatzfunktionen zulässt. Das führt zu folgender Definition aus Sauter/Schwab 2004 [19, Kapitel 4.1.3, Seite 152]:

**Definition 2.35.** *Der Raum der stückweise konstanten Funktionen über einer Partition* $\mathcal{T}_\ell$ *wird durch*

$$\mathcal{P}^0(\mathcal{T}_\ell) := \{\psi \in L^\infty(\Gamma) \big| \forall T \in \mathcal{T}_\ell : \ \psi|_T \textit{ ist konstant}\} \tag{2.45}$$

*definiert. Jede Funktion* $\psi \in \mathcal{P}^0(\mathcal{T}_\ell)$ *ist durch ihre Werte* $\lambda_T$ *auf den Elementen* $T \in \mathcal{T}_\ell$ *bestimmt und kann in der Form*

$$\psi(x) := \sum_{T\in\mathcal{T}_\ell} \lambda_T \chi_T(x) \tag{2.46}$$

*geschrieben werden mit der charakteristischen Funktion* $\chi_T : \Gamma \to \mathbb{R}$ *von* $T \in \mathcal{T}_\ell$*. Die charakteristischen Funktionen bilden also eine Basis von* $\mathcal{P}_0(\mathcal{T}_\ell)$*.*

*Der Raum der Polynome vom Grad* $p$ *auf dem Referenzelement* $\widehat{T} = [-1,1]$ *ist gegeben durch*

$$\widehat{\mathcal{P}}^p = \operatorname{span}\{\widehat{x}^\mu \big| \mu \in \mathbb{N}_0 \textit{ und } \mu \leq p\}. \tag{2.47}$$

*In Folge wird der Raum der stückweisen Polynome vom Grad* $p$ *über einer Partition* $\mathcal{T}_\ell$ *definiert durch*

$$\mathcal{P}^p(\mathcal{T}_\ell) := \{\psi : \Gamma \to \mathbb{R} \big| \psi|_T \circ \gamma_T \in \widehat{\mathcal{P}}^p\}, \tag{2.48}$$

*wobei* $\gamma_T$ *die Parametrisierung aus Definition 2.31 bezeichne.*

Ersetzt man die Lösung $\phi \in H^{-1/2}(\Gamma)$ durch die diskrete Funktion $\Phi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)$, so muss die Gleichheit in (2.43) im Allgemeinen nicht erfüllt sein. Daher wird zusätzlich auch der Raum der Testfunktionen diskretisiert.

Der Satz von Riesz impliziert die Existenz einer eindeutig bestimmten Lösung $\Phi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell) \subseteq H^{-1/2}(\Gamma)$, sodass

$$\langle\langle \Phi_\ell, \Psi_\ell \rangle\rangle = \langle F, \Psi_\ell \rangle_\Gamma \quad \text{für alle } \Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell) \tag{2.49}$$

gilt.

Schreibt man die Lösung als Summe $\Phi_h = \sum_{j=1}^N x_j \chi_j$ mit einer fixen Basis, so löst der Koeffizientenvektor $x \in \mathbb{R}^N$ das lineare Gleichungssystem

$$Ax = b. \tag{2.50}$$

Die Galerkinmatrix $A \in \mathbb{R}^{N\times N}$ wird berechnet durch

$$A_{jk} = \langle\langle \chi_j, \chi_k \rangle\rangle = \langle V\chi_j, \chi_k\rangle_\Gamma = \int_\Gamma V\chi_j(x)\chi_k(x)\, ds_x = -\frac{1}{2\pi}\int_{T_j}\int_{T_k} \log|x-y|\, ds_y ds_x.$$

Die rechte Seite ist ein Vektor $b_k \in \mathbb{R}^N$ mit

$$\begin{aligned} b_k = \langle F, \chi_k\rangle_\Gamma = \big\langle \big(K+\tfrac{1}{2}\big)g - N_0 f, \chi_k\big\rangle_\Gamma &= \int_\Gamma \Big(\big(K+\tfrac{1}{2}\big)g - N_0 f\Big)\chi_k\, ds_x \\ &= -\frac{1}{2\pi}\int_{T_k}\int_{T_j} \frac{\langle y-x, \nu_y\rangle}{|x-y|^2}\, ds_y ds_x - \int_{T_k} N_0 f\, ds_x. \end{aligned}$$

$\nu_y$ ist hierbei der nach außen gerichtete Normalvektor von $\Omega$ in $y \in \Gamma$.

Der Zusammenhang zwischen exakter und diskreter Lösung führt zum Begriff der Galerkinprojektion.

**Definition 2.36.** *Das Galerkinverfahren definiert für einen diskreten Raum $X_\ell$ einen Lösungsoperator*

$$\mathbb{G}_\ell : H^{-1/2}(\Gamma) \to X_\ell \tag{2.51}$$

*durch $\langle\langle \mathbb{G}_\ell\phi, \Psi_\ell\rangle\rangle = \langle\langle \Phi_\ell, \Psi_\ell\rangle\rangle$ für alle $\Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)$. Der Operator $\mathbb{G}_\ell$ wird Galerkinprojektion genannt.*

Das diskrete Problem (2.50) besitzt eine eindeutige Lösung, die äquivalent ist zu (2.49).

Nun wollen wir uns mit dem Verfahrensfehler beschäftigen. Der folgende Satz zeigt, dass die Galerkinapproximation die Bestapproximationseigenschaft erfüllt:

**Satz 2.37. (Lemma von Céa. Sauter/Schwab 2004 [19, Kapitel 4.1.4, Seite 155ff]).** *Sei $\phi$ die exakte Lösung der Symmschen Integralgleichung* (2.42)*. Die Galerkinlösung $\Phi_\ell$ konvergiert optimal in der Energienorm, d.h*

$$|||\phi - \Phi_\ell||| = \min_{\Psi_\ell\in\mathcal{P}^0(\mathcal{T}_\ell)} |||\phi - \Psi_\ell|||. \tag{2.52}$$

*Insbesondere erfüllt $\Phi_\ell$ die so genannte Galerkinorthogonalität*

$$\langle\langle \phi - \Phi_\ell, \Psi_\ell\rangle\rangle = 0 \quad \textit{für alle } \Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell). \tag{2.53}$$

*In der $H^{-1/2}(\Gamma)$-Norm gilt mit $c_V$ der Elliptizitätskonstante des Einfachschichtpotentials $V$ die Quasioptimalität*

$$\|\phi - \Phi_\ell\|_{H^{-1/2}(\Gamma)} \le \left(\frac{\|V\|}{c_V}\right)^{1/2} \min_{\Psi_\ell\in\mathcal{P}^0(\mathcal{T}_\ell)} \|\phi - \Psi_\ell\|_{H^{-1/2}(\Gamma)} \tag{2.54}$$

**Beweis:** Als erstes wollen wir die Galerkinorthogonalität zeigen:
Sowohl die exakte als auch die Galerkinlösung erfüllen

$$\langle\!\langle \phi, \Psi_\ell \rangle\!\rangle = \langle F, \Psi_\ell \rangle_\Gamma = \langle\!\langle \Phi_\ell, \Psi_\ell \rangle\!\rangle \quad \text{für alle } \Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell).$$

Daher gilt

$$\langle\!\langle \phi - \Phi_\ell, \Psi_\ell \rangle\!\rangle = \langle\!\langle \phi, \Psi_\ell \rangle\!\rangle - \langle\!\langle \Phi_\ell, \Psi_\ell \rangle\!\rangle = \langle F, \Psi_\ell \rangle_\Gamma - \langle F, \Psi_\ell \rangle_\Gamma = 0.$$

Aufgrund der Galerkinorthogonalität gilt für den Verfahrensfehler $e_\ell := \phi - \Phi_\ell$

$$\begin{aligned} \left|\!\left|\!\left| e_\ell \right|\!\right|\!\right|^2 = \langle\!\langle e_\ell, \phi - \Phi_\ell \rangle\!\rangle = \langle\!\langle e_\ell, \phi \rangle\!\rangle - \langle\!\langle e_\ell, \Phi_\ell \rangle\!\rangle = \langle\!\langle e_\ell, \phi \rangle\!\rangle - \langle\!\langle e_\ell, \Psi_\ell \rangle\!\rangle = \langle\!\langle e_\ell, \phi - \Psi_\ell \rangle\!\rangle \\ \leq \left|\!\left|\!\left| e_\ell \right|\!\right|\!\right| \left|\!\left|\!\left| \phi - \Psi_\ell \right|\!\right|\!\right| \end{aligned}$$

für beliebiges $\Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)$. Insbesondere gilt also

$$\left|\!\left|\!\left| \phi - \Phi_\ell \right|\!\right|\!\right| \leq \min_{\Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)} \left|\!\left|\!\left| \phi - \Psi_\ell \right|\!\right|\!\right|.$$

Da $\Phi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)$ gilt, folgt die Gleichheit.

Die zweite Aussage erhält man sofort aufgrund der äquivalenten Normen

$$\begin{aligned} c_V \|e_\ell\|^2_{H^{-1/2}(\Gamma)} \leq \left|\!\left|\!\left| e_\ell \right|\!\right|\!\right|^2 = \min_{\Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)} \left|\!\left|\!\left| \phi - \Psi_\ell \right|\!\right|\!\right|^2 = \min_{\Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)} \langle V(\phi - \Psi_\ell), \phi - \Psi_\ell \rangle_\Gamma \\ \leq \min_{\Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)} \|V\| \|\phi - \Psi_\ell\|^2_{H^{-1/2}(\Gamma)}. \end{aligned}$$

Der Beweis ist vollständig nach Division durch $c_V$ und Wurzelziehen. □

**Korollar 2.38.** *Die Galerkinprojektion $\mathbb{G}_\ell : H^{-1/2}(\Gamma) \to X_\ell$ ist die Orthogonalprojektion auf $X_\ell$ in der Energienorm.* □

Nun bleibt nur noch die Konvergenz der Galerkinrandelementmethode zu zeigen.

**Satz 2.39. (Sauter/Schwab 2004 [19, Kapitel 4.1.5, Seite 159]).** *Sei $(\mathcal{T}_\ell)_{\ell \in \mathbb{N}}$ eine Folge von Partitionen mit globaler Netzweite $h_{\ell,\max} \to 0$ für $\ell \to \infty$, für die gilt, dass sich jedes Element aus $\mathcal{T}_\ell$ darstellen lässt als Vereinigung von Elementen aus $\mathcal{T}_{\ell-1}$. Dann konvergiert die Folge von Randelementlösungen $(\Phi_\ell)_{\ell \in \mathbb{N}}$ gegen die exakte Lösung $\phi$.*

**Beweis:** Der Raum $\mathcal{P}^0(\mathcal{T}_\ell)$ besteht aus Treppenfunktionen auf einem Gitter $\mathcal{T}_\ell$. Aufgrund der Definition der Lebesgue-Räume ist $\bigcup_{\ell \in \mathbb{N}} \mathcal{P}^0(\mathcal{T}_\ell)$ dicht in $L^2(\Gamma)$, welcher wiederum dicht in $H^{-1/2}(\Gamma)$ ist. Sei $\varepsilon > 0$ beliebig. Dann lassen sich $\widetilde{\phi} \in L^2(\Gamma)$, $\ell \in \mathbb{N}$ und $\widetilde{\Phi}_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)$ wählen, sodass

$$\|\phi - \widetilde{\phi}\|_{H^{-1/2}(\Gamma)} \leq \frac{\varepsilon}{2} \quad \text{und} \quad \|\widetilde{\phi} - \widetilde{\Phi}_\ell\|_{L^2(\Gamma)} \leq \frac{\varepsilon}{2}$$

gelten. Wegen $\|\cdot\|_{H^{1/2}(\Gamma)} \leq \|\cdot\|_{L^2(\Gamma)}$ erhält man insgesamt

$$\|\phi - \widetilde{\Phi}_\ell\|_{H^{-1/2}(\Gamma)} \leq \|\phi - \widetilde{\phi}\|_{H^{-1/2}(\Gamma)} + \|\widetilde{\phi} - \widetilde{\Phi}_\ell\|_{H^{-1/2}(\Gamma)} \leq \varepsilon$$

und somit die Konvergenz gegen die exakte Lösung. □

## 2.5 Einfluss von Störungen

Dieser Abschnitt beschäftigt sich mit der wichtigen Frage des Einflusses von Störungen. Es wird sich zeigen, dass die Galerkin-Randelementmethode stabil bezüglich hinreichend kleiner Störungen ist. Die Aussagen dieses Abschnitts werden hier nicht bewiesen, sind aber in ausführlicher Form in Ferraz-Leite 2007 [11] nachzulesen.

**Lemma 2.40. (Lemma von Strang. Braess 2003 [4, Kapitel III, Lemma 1.1]).** *Seien $H$ ein Hilbertraum, $a(\cdot,\cdot)$ eine stetige, elliptische Bilinearform auf $H$ und $u \in H$ mit $a(u,\cdot) = \ell \in H^*$. Für einen Diskretisierungsparameter $h$ seien $X_h \subseteq H$ ein abgeschlossener Unterraum von $H$, $a_h(\cdot,\cdot)$ eine stetige Bilinearform auf $X_h$ und $\ell_h \in X_h^*$ ein gegebenes lineares Funktional. Dann gelten folgende Aussagen:*

*(i) Falls $a_h(\cdot,\cdot)$ gegen $a(\cdot,\cdot)$ konvergiert, d.h*

$$\lim_{h\to 0} \alpha_h = 0 \quad \textit{mit} \quad \alpha_h := \sup_{v_h, w_h \in X_h \setminus \{0\}} \frac{|a(v_h,w_h) - a_h(v_h,w_h)|}{\|v_h\|_H \|w_h\|_H} \tag{2.55}$$

*gilt, so folgt*

$$\exists\, h_0 > 0\ \exists \alpha_0 > 0\ \forall h < h_0\, \forall v_h \in X_h :\ \alpha_0 \|v_h\|_H^2 \le a_h(v_h,v_h). \tag{2.56}$$

*Insbesondere ist die Bilinearform $a_h(\cdot,\cdot)$ elliptisch auf $X_h$ und es existiert somit ein eindeutiges $u_h \in X_h$, sodass $a_h(u_h,\cdot) = \ell_h \in X_h^*$ gilt für genügend kleines $h > 0$.*

*(ii) Gilt* (2.55)*, so folgt*

$$\begin{aligned} \|u - u_h\|_H &\le C \left( \inf_{v_h \in X_h} \left( \|u - v_h\|_H + \|a(v_h,\cdot) - a_h(v_h,\cdot)\|_{X_h^*} \right) + \|\ell - \ell_h\|_{X_h^*} \right) \\ &\le C \left( \inf_{v_h \in X_h} \left( \|u - v_h\|_H + \alpha_h \|v_h\| \right) + \|\ell - \ell_h\|_{X_h^*} \right). \end{aligned} \tag{2.57}$$

*Die Konstante $C$ hängt hierbei nur von der Bilinearform $a(\cdot,\cdot)$ ab.*

*(iii) Falls die Datenfehler gegen 0 konvergieren, d.h*

$$\lim_{h\to 0} \|\ell - \ell_h\|_{X^*} = 0 = \lim_{h\to 0} \alpha_h, \tag{2.58}$$

*und es eine dichte Teilmenge $D \subseteq H$ gibt, sodass*

$$\lim_{h\to 0} \min_{v_h \in X_h} \|v - v_h\|_H = 0, \tag{2.59}$$

*folgt insbesondere die Konvergenz des gestörten Galerkinverfahrens*

$$\lim_{h\to 0} \|u - u_h\|_H = 0. \tag{2.60}$$

*Ist die Konvergenzrate der Datenfehler also mindestens so groß wie die des Verfahrensfehlers, so konvergiert das gestörte Galerkinverfahren mit derselben Ordnung wie das ungestörte.* □

# Kapitel 3

# Adaptiver Algorithmus mit residualem Fehlerschätzer

Es nicht möglich, für beliebiges $g \in H^{1/2}(\Gamma)$ die Funktion $Kg \in H^{1/2}(\Gamma)$ numerisch zu berechnen. Man hat nun mehrere Möglichkeiten:

(a) Man approximiert $Kg(x)$ durch Quadratur.

(b) Für glattes $g$ kann man $Kg(x)$ in einen glatten und einen logarithmisch-singulären Anteil zerlegen, die dann geeignet numerisch integriert werden.

(c) Man ersetzt $g$ durch eine Approximation $g_\ell$, für die $Kg_\ell(x)$ exakt berechnet werden kann.

In diesem Abschnitt betrachten wir Möglichkeit (c), was auch den Vorteil hat, dass man den Operator mit geeigneten Matrixkompressionstechniken effizient realisieren kann.

Es erscheint nun sinnvoll, die gestörten Dirichletdaten $g_\ell$ bei Fehlerabschätzungen für das Verwenden der Galerkinrandelementmethode zum Lösen der Symmschen Integralgleichung miteinzubeziehen. Daher wird ein lokaler a posteriori Fehlerschätzer vorgestellt, der auch den Datenfehler $F_\ell$ der rechten Seite $F$ der Symmschen Integralgleichung $V\phi = F$ berücksichtigt.

Wir nehmen an, dass die Dirichletdaten $g \in H^1(\Gamma)$ erfüllen und definieren $g_\ell := \mathcal{I}_\ell g$ als deren nodale Interpolation:

**Definition 3.1.** *Sei $\mathcal{N}_\ell$ die Menge aller Knoten einer Triangulierung $\mathcal{T}_\ell$ und $\mathcal{S}^1(\mathcal{T}_\ell) := \mathcal{P}^1(\mathcal{T}_\ell) \cap \mathcal{C}(\Gamma)$ die Menge aller global stetigen Funktionen in $\mathcal{P}^1(\mathcal{T}_\ell)$. Für alle $z \in \mathcal{N}_\ell$ sei $\gamma_z \in \mathcal{S}^1(\mathcal{T}_\ell)$ die nodale Basisfunktion, d.h. $\gamma_z(z) = 1$ und $\gamma_z(\overline{z}) = 0$ für $\overline{z} \in \mathcal{N}_\ell \setminus \{z\}$.*

*Der nodale Interpolationsoperator $\mathcal{I}_\ell : \mathcal{C}(\Gamma) \to \mathcal{S}^1(\mathcal{T}_\ell)$ ist dann definiert durch*

$$\mathcal{I}_\ell v := \sum_{z \in \mathcal{N}_\ell} v(z)\gamma_z.$$

Wir erinnern daran, dass für 2D BEM nach Sobolevungleichung die Inklusion $H^1(\Gamma) \subseteq \mathcal{C}(\Gamma)$ gilt, nachzulesen in Sauter/Schwab 2004[6][Kapitel 2.5].

Seien im Folgenden

- $\phi \in H^{-1/2}(\Gamma)$ die kontinuierliche Lösung von $V\phi = (K + \frac{1}{2})g =: F \in H^{1/2}(\Gamma)$
- $\widetilde{\phi}_\ell \in H^{-1/2}(\Gamma)$ die kontinuierliche Lösung von $V\widetilde{\phi}_\ell = (K + \frac{1}{2})g_\ell =: F_\ell \in H^{1/2}(\Gamma)$
- $\Phi_\ell$ die Galerkinlösung zu $\phi$ in $X_\ell := \mathcal{P}^0(\mathcal{T}_\ell)$
- $\widetilde{\Phi}_\ell$ die Galerkinlösung zu $\widetilde{\phi}_\ell$ in $X_\ell$.

Zu lösen ist also $\langle V\widetilde{\Phi}_\ell, \Psi_\ell\rangle = \langle F_\ell, \Psi_\ell\rangle$ für alle $\Psi_\ell \in X_\ell$ mit dem Ziel einer Approximationsfehlerkontrolle von $|||\phi - \widetilde{\Phi}_\ell|||$.

Für $a \leq Cb$ mit einer Konstanten $C > 0$, die nicht von $a$ oder $b$ abhängt, verwenden wir die abkürzende Schreibweise $a \lesssim b$.

Gibt es eine Konstante $C_{\text{rel}} > 0$, sodass

$$|||\phi - \widetilde{\Phi}_\ell||| \leq C_{\text{rel}}\, \varrho_\ell$$

gilt, so nennt man den Fehlerschätzer $\varrho_\ell$ zuverlässig im Bezug auf $|||\phi - \widetilde{\Phi}_\ell|||$. Die Konstante $C_{\text{rel}}$ darf von der rechten Seite $F_\ell$ und $\kappa(\mathcal{T}_\ell)$ abhängen, nicht aber von den Lösungen $\phi$ und $\widetilde{\Phi}_\ell$.

Eine Norm $||\cdot||$ heißt lokal, wenn sie für eine Triangulierung $\mathcal{T}_\ell$ und einen Raum $X$

$$||\cdot||^2_{X(\Gamma)} \sim \sum_{T\in\mathcal{T}_\ell} ||\cdot||^2_{X(T)}$$

erfüllt. Die Energienorm ist im Gegensatz zur $L^2$-Norm oder zur $H^1$-Norm nicht lokal. Fehlerschätzer in lokalen Normen erlauben die Steuerung adaptiver Netzverfeinerungsstrategien.

Der a posteriori Fehlerschätzer $\varrho_\ell$ selbst darf nur von berechenbaren Größen abhängen, insbesondere also von der diskreten Lösung $\widetilde{\Phi}_\ell$, nicht aber von der exakten Lösung $\phi$. Der Fehlerschätzer kann also erst nach einer ersten Berechnung einer diskreten Lösung ermittelt werden — ganz im Gegensatz zur a priori Fehlerschätzung, die von der exakten Lösung $\phi$, nicht aber von $\widetilde{\Phi}_\ell$ abhängt.

Dieses Kapitel beschäftigt sich nun mit der Entwicklung eines zuverlässigen, lokalen Gesamtfehlerschätzers, bestehend aus dem Residualschätzer $\varrho_\ell$ — siehe Carstensen 1997 [6] — und einem Datenfehlerschätzer $\zeta_\ell$. Unter Verwendung eines speziellen, in diesem Abschnitt beschriebenen adaptiven Algorithmus, lässt sich Konvergenz der approximierten Lösung $\widetilde{\Phi}_\ell$ gegen die exakte Lösung $\phi$ beweisen.

## 3.1 Lokalisierungtechniken für $H^{1/2}(\Gamma)$

Dieses Kapitel beschreibt den Übergang von der $H^{1/2}$-Norm zu lokalen Normen um im Weiteren lokale Fehlerschätzer entwickeln zu können.

Für den nodalen Interpolationsoperator können wir eine Approximationseigenschaft formulieren, müssen dafür aber noch einen Satz bereitstellen, der Carstensen 1997 [6] entnommen wurde:

**Satz 3.2. (Carstensen 1997 [6, Theorem 1]).** *Hat eine Funktion $v \in H^1(\Gamma)$ zumindest eine Nullstelle in jedem Element einer Partition $\mathcal{T}_\ell$, so gilt*

$$\|v\|_{H^{1/2}(\Gamma)} \leq \widetilde{C}_{\mathrm{apx}} \|h_\ell^{1/2} v'\|_{L^2(\Gamma)}. \tag{3.1}$$

*Die Konstante $\widetilde{C}_{\mathrm{apx}} > 0$ hängt nur von $\Gamma$ und $\kappa(\mathcal{T}_\ell)$ ab.* □

**Korollar 3.3. (Carstensen/Praetorius 2007 [8, Corollary 3.4]).** *Ist $\mathcal{I}_\ell$ der nodale Interpolationsoperator und $v \in H^1(\Gamma)$, so gilt*

$$\|v - \mathcal{I}_\ell v\|_{H^{1/2}(\Gamma)} \leq \widetilde{C}_{\mathrm{apx}} \|h_\ell^{1/2} (v - \mathcal{I}_\ell v)'\|_{L^2(\Gamma)} \leq \widetilde{C}_{\mathrm{apx}} \|h_\ell^{1/2} v'\|_{L^2(\Gamma)}. \tag{3.2}$$

*Die Konstante $\widetilde{C}_{\mathrm{apx}} > 0$ hängt hierbei nur von $\Gamma$ und $\kappa(\mathcal{T}_\ell)$ ab.*

**Beweis:** Sei $w := v - \mathcal{I}_\ell v$ und $\mathcal{N}_\ell$ die Menge aller Knoten einer Partition $\mathcal{T}_\ell$, dann gilt $w(z) = 0$ für alle $z \in \mathcal{N}_\ell$, $w$ hat also auf jedem Element von $\mathcal{T}_\ell$ eine Nullstelle. Satz 3.2 kann daher angewendet werden, und es folgt

$$\|w\|_{H^{1/2}(\Gamma)} \leq \widetilde{C}_{\mathrm{apx}} \|h_\ell^{1/2} w'\|_{L^2(\Gamma)}$$

und damit die erste Ungleichung.
Die Bogenlängenableitung der nodalen Interpolation kann durch die $L^2$-Projektion der Ableitung auf $\mathcal{P}^0(\mathcal{T}_\ell)$ ersetzt werden. Für $T = [a, b]$ gilt

$$\big(\Pi_\ell\, v'\big)\big|_T = \frac{1}{|b-a|} \int_{[a,b]} v'(s)\, ds = \frac{v(b) - v(a)}{|b-a|} = (\mathcal{I}_\ell\, v)'|_T. \tag{3.3}$$

Für die $L^2$-Projektion als elementweise Bestapproximation bezüglich der $L^2$-Norm folgt

$$\|h_\ell^{1/2}(v - \mathcal{I}_\ell v)'\|_{L^2(\Gamma)}^2 = \|h_\ell^{1/2}(v' - \Pi_\ell v')\|_{L^2(\Gamma)}^2 = \|h_\ell^{1/2} v'\|_{L^2(\Gamma)}^2 + \|h_\ell^{1/2} \Pi_\ell v'\|_{L^2(\Gamma)}^2 \leq \|h_\ell^{1/2} v'\|_{L^2(\Gamma)}^2$$

und somit die zweite Ungleichung. □

Man kann auch eine inverse Abschätzung zu Satz 3.2 angeben.

**Satz 3.4. (Carstensen/Praetorius 2007 [8, Proposition 3.1]).** *Für eine diskrete Funktion $\Psi_\ell \in \mathcal{S}^p(\mathcal{T}_\ell) := \mathcal{P}^p(\mathcal{T}_\ell) \cap \mathcal{C}(\Gamma)$ gilt*

$$\|h_\ell^{1/2} \Psi_\ell'\|_{L^2(\Gamma)} \leq \widetilde{C}_{\mathrm{inv}} \|\Psi_\ell\|_{H^{1/2}(\Gamma)}. \tag{3.4}$$

*Die Konstante $\widetilde{C}_{\mathrm{inv}} > 0$ hängt nur von $p$ und $\Gamma$ ab.* □

## 3.2 Residualschätzer

Wir wenden uns nun einem zuverlässigen Fehlerschätzer — dem Residualfehlerschätzer — zu.

**Satz 3.5.** *Für $g_\ell \in H^1(\Gamma)$ gelten*

$$R_\ell := \big(K + \tfrac{1}{2}\big) g_\ell - V \widetilde{\Phi}_\ell \in H^1(\Gamma) \tag{3.5}$$

*und*

$$C_{\mathrm{rel}}^{-1} \big|\!\big|\!\big| \widetilde{\phi}_\ell - \widetilde{\Phi}_\ell \big|\!\big|\!\big| \leq \varrho_\ell := \big\| h_\ell^{1/2} R_\ell' \big\|_{L^2(\Gamma)}, \tag{3.6}$$

*d.h. $\varrho_\ell$ ist zuverlässig mit Bezug auf $\big|\!\big|\!\big| \widetilde{\phi}_\ell - \widetilde{\Phi}_\ell \big|\!\big|\!\big|$. Die Zuverlässigkeitskonstante $C_{\mathrm{rel}}$ hängt nur von $\Gamma$ und $\kappa(\mathcal{T}_\ell)$ ab und ist insbesondere unabhängig von den Dirichletdaten $g_\ell$.*

**Beweis:** Da laut Voraussetzung $g_\ell \in H^1(\Gamma)$ ist, folgt $\big(K + \frac{1}{2}\big)g_\ell \in H^1(\Gamma)$ laut Satz 2.28. Für $\widetilde{\Phi}_\ell \in \mathcal{P}^0(\mathcal{T}_\ell) \subseteq L^2(\Gamma)$ gilt außerdem $V\widetilde{\Phi}_\ell \in H^1(\Gamma)$ aufgrund der Abbildungsvorschrift aus Satz 2.24. Daraus folgt direkt $R_\ell \in H^1(\Gamma)$.
Wegen der Galerkinorthogonalität gilt mit der charakteristischen Funktion $\chi_T$

$$0 = \langle\!\langle \widetilde{\phi}_\ell - \widetilde{\Phi}_\ell, \chi_T \rangle\!\rangle = \langle V(\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell), \chi_T\rangle_\Gamma = \langle F_\ell - V\widetilde{\Phi}_\ell, \chi_T\rangle_\Gamma = \langle R_\ell, \chi_T\rangle_\Gamma.$$

Da $R_\ell \in H^1(\Gamma) \subseteq \mathcal{C}(\Gamma)$ gilt, hat $R_\ell$ auf jedem Element $T$ der Partition $\mathcal{T}_\ell$ zumindest eine Nullstelle, und Satz 3.2 kann angewendet werden

$$\begin{aligned}
\big|\!\big|\!\big| \widetilde{\phi}_\ell - \widetilde{\Phi}_\ell \big|\!\big|\!\big|^2 = \langle V(\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell), \widetilde{\phi}_\ell - \Phi_\ell\rangle_\Gamma = \langle R_\ell, \widetilde{\phi}_\ell - \Phi_\ell\rangle_\Gamma &\le \|R_\ell\|_{H^{1/2}(\Gamma)}\, \|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\|_{H^{-1/2}(\Gamma)} \\
&\lesssim \big\|h_\ell^{1/2} R_\ell'\big\|_{L^2(\Gamma)} \|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\|_{H^{-1/2}(\Gamma)} \\
&\lesssim \big\|h_\ell^{1/2} R_\ell'\big\|_{L^2(\Gamma)} \big|\!\big|\!\big| \widetilde{\phi}_\ell - \widetilde{\Phi}_\ell \big|\!\big|\!\big|.
\end{aligned}$$

Insgesamt erhält man also

$$\big|\!\big|\!\big| \widetilde{\phi}_\ell - \widetilde{\Phi}_\ell \big|\!\big|\!\big| \lesssim \big\|h_\ell^{1/2} R_\ell'\big\|_{L^2(\Gamma)}$$

und damit die Zuverlässigkeit des Fehlerschätzers. □

## 3.3 Datenfehlerschätzer

In diesem Kapitel wird der Datenfehlerschätzer $\zeta_\ell$ eingeführt und dessen Eigenschaften diskutiert.

**Satz 3.6.** *Der Datenfehlerschätzer*

$$\zeta_\ell = \big\|h_\ell^{1/2}(g - g_\ell)'\big\|_{L^2(\Gamma)} = \big\|h_\ell^{1/2}(1 - \Pi_\ell)g'\big\|_{L^2(\Gamma)} \tag{3.7}$$

*erfüllt* $\big|\!\big|\!\big| \phi - \widetilde{\phi}_\ell \big|\!\big|\!\big| \le C_{\mathrm{data}}\zeta_\ell$. *Die Konstante* $C_{\mathrm{data}} > 0$ *ist nur abhängig von* $\Gamma$ *und* $\kappa(\mathcal{T}_\ell)$.

**Beweis:** Zunächst wird die Äquivalenz der Energienorm $|\!|\!|\cdot|\!|\!|$ zur $H^{1/2}(\Gamma)$-Norm $\|V(\cdot)\|_{H^{1/2}(\Gamma)}$ gezeigt: Aus der Stetigkeit des Einfachschichtpotentials $V$ und der Normäquivalenz zwischen der Energienorm $|\!|\!|\cdot|\!|\!|$ und der $H^{-1/2}(\Gamma)$-Norm $\|\cdot\|_{H^{-1/2}(\Gamma)}$ folgt

$$|\!|\!|\phi|\!|\!|^2 = \langle V\phi, \phi\rangle_{H^{1/2}(\Gamma)\times H^{-1/2}(\Gamma)} \le \|V\phi\|_{H^{1/2}(\Gamma)}\|\phi\|_{H^{-1/2}(\Gamma)} \lesssim \|V\phi\|_{H^{1/2}(\Gamma)}|\!|\!|\phi|\!|\!|.$$

Durch einfache Umformung erhält man

$$|\!|\!|\phi|\!|\!| \lesssim \|V\phi\|_{H^{1/2}(\Gamma)}.$$

Die andere Richtung wird mit denselben Argumenten wie zuvor bewiesen

$$\|V\phi\|_{H^{1/2}(\Gamma)} \le \|V\|\|\phi\|_{H^{-1/2}(\Gamma)} \lesssim \|V\||\!|\!|\phi|\!|\!|.$$

Für den Datenfehler gilt nun also

$$\begin{aligned}
\big|\!\big|\!\big| \phi - \widetilde{\phi}_\ell \big|\!\big|\!\big| \lesssim \|V(\phi - \widetilde{\phi}_\ell)\|_{H^{1/2}(\Gamma)} = \|F - F_\ell\|_{H^{1/2}(\Gamma)} &= \|\big(K + \tfrac{1}{2}\big)g - \big(K + \tfrac{1}{2}\big)g_\ell\|_{H^{1/2}(\Gamma)} \\
&= \|\big(K + \tfrac{1}{2}\big)(g - g_\ell)\|_{H^{1/2}(\Gamma)}.
\end{aligned}$$

Aus der Stetigkeit des Doppelschichtpotentials $K$ folgt

$$|||\phi - \widetilde{\phi}_\ell||| \lesssim \|g - g_\ell\|_{H^{1/2}(\Gamma)}.$$

Die Datenapproximation $g_\ell = \mathcal{I}_\ell\, g$ erhält man durch Anwenden des nodalen Interpolationsoperators $\mathcal{I}_\ell$. Aus der Lokalisierungseigenschaft für die nodale Interpolation aus Satz 3.2 folgt

$$|||\phi - \widetilde{\phi}_\ell||| \lesssim \|h_\ell^{1/2}(g - g_\ell)'\|_{L^2(\Gamma)}.$$

Die Bogenlängenableitung der nodalen Interpolation kann durch die $L^2$-Projektion auf $\mathcal{P}^0(\mathcal{T}_\ell)$ der Ableitung ersetzt werden und man erhält durch Einsetzen von Formel (3.3)

$$|||\phi - \widetilde{\phi}_\ell||| \lesssim \|h_\ell^{1/2}(1 - \Pi_\ell)g'\|_{L^2(\Gamma)}$$

die Aussage des Satzes. □

## 3.4 Gesamtfehlerschätzer

Dieser Abschnitt beschäftigt sich mit einem Fehlerschätzer, der das Verhalten von $|||\phi - \widetilde{\Phi}_\ell|||$ beschränkt, also eine Kombination aus Daten- und Verfahrensfehlerschätzer ist.

**Satz 3.7.** *Der Gesamtfehlerschätzer* $\omega_\ell := \left(\varrho_\ell^2 + \zeta_\ell^2\right)^{1/2}$ *erfüllt*

$$|||\phi - \widetilde{\Phi}_\ell||| \leq \overline{C}_{\mathrm{rel}}\omega_\ell, \tag{3.8}$$

*d.h.* $\omega_\ell$ *ist zuverlässig in Bezug auf* $|||\phi - \widetilde{\Phi}_\ell|||$. *Die Zuverlässigkeitskonstante* $\overline{C}_{\mathrm{rel}}$ *hängt nur von* $\Gamma$ *und* $\kappa(\mathcal{T}_\ell)$ *ab.*

**Beweis:** Aus $(a+b)^2 \leq 2a^2 + 2b^2$ folgt

$$|||\phi - \widetilde{\Phi}_\ell|||^2 \leq 2|||\phi - \widetilde{\phi}_\ell|||^2 + 2|||\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|||^2. \tag{3.9}$$

Mit Satz 3.5 und 3.6 gilt also

$$|||\phi - \widetilde{\Phi}_\ell|||^2 \leq 2C_{\mathrm{data}}^2\zeta_\ell^2 + 2C_{\mathrm{rel}}^2\varrho_\ell^2 \leq 2\max\{C_{\mathrm{data}}^2, C_{\mathrm{rel}}^2\}\omega_\ell^2.$$

□

## 3.5 Adaptiver Algorithmus

Zunächst wird hier der verwendete adaptive Algorithmus angegeben und auf die verwendete Markierungsstrategie näher eingegangen.

Gegeben seien ein Abbruchparameter $\varepsilon > 0$ und eine Startpartition $\mathcal{T}_0$.

**Algorithmus 3.8.** *Input: Startnetz* $\mathcal{T}_0$, *Abbruchparameter* $\varepsilon > 0$.

*(1) Berechne diskrete Lösung* $\widetilde{\Phi}_\ell$.

*(2) Berechne Verfeinerungsindikatoren* $\omega_\ell(T) = \left(\varrho_\ell(T)^2 + \zeta_\ell(T)^2\right)^{1/2}$ *für alle* $T \in \mathcal{T}_\ell$.

*(3) Stoppe, falls* $\omega_\ell := \left(\sum_{T\in\mathcal{T}_\ell} \omega_\ell(T)^2\right)^{1/2} \leq \varepsilon$.

*(4) Bilde Menge* $\mathcal{M}_\ell$*, der zur Verfeinerung markierten Elemente.*

*(5) Verfeinere mindestens die Elemente* $T \in \mathcal{M}_\ell$ *und erhalte* $\mathcal{T}_{\ell+1}$.

*(6) Erhöhe Zähler* $\ell \to \ell + 1$ *und gehe zu (1).*

*Output: Approximation* $\widetilde{\Phi}_\ell$ *von* $\phi$.

Die in Schritt (4) angegebene Verfeinerungsstrategie besteht aus mehreren Teilen. Zunächst wird zur Markierung der Elemente mit „großem“ Fehler die so genannte Dörfler Markierung verwendet:

Für $\mathcal{M} \subseteq \mathcal{T}$, die Menge aller markierten Elemente, und ein fixes $\theta \in (0,1)$ gilt

$$\theta \sum_{T\in\mathcal{T}} \omega_\ell(T)^2 \leq \sum_{T\in\mathcal{M}} \omega_\ell(T)^2. \tag{3.10}$$

Die Zuverlässigkeit des Residualschätzers ist — wie man im Beweis zu Satz 3.5 erkennen kann — abhängig von der Gültigkeit von Satz 3.2 mit einer Konstanten $\widetilde{C}_{\mathrm{apx}} < \infty$. Dies Konstante ist abhängig von $\kappa(\mathcal{T}_\ell)$. Um einen zuverlässigen Fehlerschätzer zu erhalten, muss daher die Markierungsstrategie rekursiv um eine zusätzliche Bedingung erweitert werden, damit $\kappa(\mathcal{T}_\ell) \leq \max\{\kappa(\mathcal{T}_0), C_{\mathrm{mark}}\}$ gilt:

Sei $\mathcal{M}_\ell$ die Menge der markierten Elemente der Partition $\mathcal{T}_\ell$ und $T, T' \in \mathcal{T}_\ell$ zwei benachbarte Elemente, d.h. $\overline{T} \cap \overline{T}\,' \neq \emptyset$ und $T \neq T'$, mit $T' \in \mathcal{M}_\ell$ und $T \in \mathcal{T}_\ell \backslash \mathcal{M}_\ell$. $\widehat{T}, \widehat{T}' \in \mathcal{T}_{\ell+1}$ seien die benachbarten Söhne von $T$ und $T'$, d.h. es gelte $\widehat{T} \subseteq T$, $\widehat{T}' \subseteq T'$ und $\overline{\widehat{T}} \cap \overline{\widehat{T}'} \neq \emptyset$. Außerdem bezeichnen $h_{\ell+1}(\widehat{T}) := \mathrm{diam}(\widehat{T})$ und $h_{\ell+1}(\widehat{T}') := \mathrm{diam}(\widehat{T}')$ die beiden lokalen Schrittweiten. Sind nun $T' \in \mathcal{M}_\ell$ und $T \in \mathcal{T}_\ell \backslash \mathcal{M}_\ell$ und gilt

$$h_{\ell+1}(\widehat{T}) \geq C_{\mathrm{mark}} h_{\ell+1}(\widehat{T}') \tag{3.11}$$

so wird auch $T$ zur Verfeinerung markiert.

Die Markierungsstrategie wird zur Veranschaulichung als Algorithmus in Pseudocode angegeben.

**Algorithmus 3.9.** *Input: Partition* $\mathcal{T}_\ell$*, Verfeinerungsindikatoren* $\omega_\ell(T)$*, Markierungsparameter* $C_{\mathrm{mark}}$ *und* $\theta \in (0,1)$

*(4a) Bilde Menge* $\mathcal{M}_\ell$*, der nach Dörfler* (3.10) *markierten Elemente:*
**Bilde** *Menge* $\mathcal{S}_\ell$ *aus* $\mathcal{T}_\ell$*, der nach* $\omega_\ell^2(T)$ *absteigend sortierten Elemente.*
**Bilde** *Menge* $\mathcal{M}_\ell$*, der ersten* $k$ *Elemente aus* $\mathcal{S}_\ell$*, sodass*

$$\theta \sum_{T\in\mathcal{S}_\ell} \omega_\ell(T)^2 \leq \sum_{j=1}^{k} \omega_\ell(T_j)^2 = \sum_{T\in\mathcal{M}_\ell} \omega_\ell(T)^2$$

*erfüllt ist.*

*(4b) Erweitere Menge* $\mathcal{M}_\ell$*, sodass* $\kappa(\mathcal{T}_\ell)$ *beschränkt bleibt:*
**Solange** *Änderungen in der Menge* $\mathcal{M}_\ell$ *stattfinden.*
**Durchlaufe** *Menge* $\mathcal{M}_\ell$.

**Falls** *für den Sohn* $\widehat{T}'$ *eines markierten Elements* $T'$ *und den Sohn* $\widehat{T}$ *eines nicht markierten Nachbarn* $T$ *von* $T'$ *gilt* $h_{\ell+1}(\widehat{T}) \geq C_{\mathrm{mark}} h_{\ell+1}(\widehat{T}')$.
**Erweitere** $\mathcal{M}_\ell$ *um* $T$.
**ende**
**ende**
**ende**

*Output: Menge* $\mathcal{M}_\ell$.

Dass diese Zusatzbedingung ausreicht, um $\kappa(\mathcal{T}_\ell) < \infty$ zu garantieren, zeigt das folgende Lemma:

**Lemma 3.10.** *Ist die Markierungsstrategie für eine Konstante* $C_{\mathrm{mark}} > 0$ *um die Bedingung* (3.11) *erweitert, so gilt für alle* $\ell \in \mathbb{N}$

$$\kappa(\mathcal{T}_{\ell+1}) \leq \max\{\kappa(\mathcal{T}_\ell), C_{\mathrm{mark}}\}. \tag{3.12}$$

**Beweis:** Seien $\widehat{T}, \widehat{T}' \in \mathcal{T}_{\ell+1}$ mit $\widehat{T} \cap \widehat{T}' \neq \emptyset$, d.h $\widehat{T}$ und $\widehat{T}'$ sind Nachbarn in $\mathcal{T}_{\ell+1}$. O.B.d.A. gelte $\widehat{T} \neq \widehat{T}'$, da sonst $\frac{h_{\ell+1}(\widehat{T})}{h_{\ell+1}(\widehat{T}')} = 1$. Insbesondere folgt für die $T, T' \in \mathcal{T}_\ell$, die Väter von $\widehat{T}$ und $\widehat{T}'$, $T \cap T' \neq \emptyset$, d.h. $T$ und $T'$ sind Nachbarn in $\mathcal{T}_\ell$.
1. Fall: Für $T, T' \in \mathcal{T}_\ell \backslash \mathcal{M}_\ell$ gelten $h_\ell(T) = h_{\ell+1}(\widehat{T})$ und $h_\ell(T') = h_{\ell+1}(\widehat{T}')$ und somit

$$\frac{h_{\ell+1}(\widehat{T})}{h_{\ell+1}(\widehat{T}')} = \frac{h_\ell(T)}{h_\ell(T')} \leq \kappa(\mathcal{T}_\ell).$$

2. Fall: Für $T, T' \in \mathcal{M}_\ell$ gelten $\frac{1}{2} h_\ell(T) = h_{\ell+1}(\widehat{T})$ und $\frac{1}{2} h_\ell(T') = h_{\ell+1}(\widehat{T}')$ und somit

$$\frac{h_{\ell+1}(\widehat{T})}{h_{\ell+1}(\widehat{T}')} = \frac{\frac{1}{2} h_\ell(T)}{\frac{1}{2} h_\ell(T')} \leq \kappa(\mathcal{T}_\ell).$$

3. Fall: Für $T \in \mathcal{M}_\ell$, $T' \in \mathcal{T}_\ell \backslash \mathcal{M}_\ell$ gelten $\frac{1}{2} h_\ell(T) = h_{\ell+1}(\widehat{T})$ und $h_\ell(T') = h_{\ell+1}(\widehat{T}')$ und somit

$$\frac{h_{\ell+1}(\widehat{T})}{h_{\ell+1}(\widehat{T}')} = \frac{\frac{1}{2} h_\ell(T)}{h_\ell(T')} \leq \frac{1}{2} \kappa(\mathcal{T}_\ell).$$

4. Fall: Für $T \in \mathcal{T}_\ell \backslash \mathcal{M}_\ell$, $T' \in \mathcal{M}_\ell$ gelten $h_\ell(T) = h_{\ell+1}(\widehat{T})$ und $\frac{1}{2} h_\ell(T') = h_{\ell+1}(\widehat{T}')$.
In diesem Fall gilt stets $h_{\ell+1}(\widehat{T}) < C_{\mathrm{mark}}\, h_{\ell+1}(\widehat{T}')$, da für $h_{\ell+1}(\widehat{T}) \geq C_{\mathrm{mark}}\, h_{\ell+1}(\widehat{T}')$ die erweiterte Markierungsstrategie (3.11) greift, $T$ ebenfalls markiert wird und Fall 2. eintritt.
Also gilt

$$\frac{h_{\ell+1}(\widehat{T})}{h_{\ell+1}(\widehat{T}')} < C_{\mathrm{mark}}.$$

Insgesamt ist also für alle $T, T'$ Nachbarn in $\mathcal{T}_\ell$

$$\frac{h_{\ell+1}(\widehat{T})}{h_{\ell+1}(\widehat{T}')} \leq \max\{\kappa(\mathcal{T}_\ell), C_{\mathrm{mark}}\}$$

erfüllt. Insbesondere gilt also $\kappa(\mathcal{T}_{\ell+1}) \leq \max\{\kappa(\mathcal{T}_\ell), C_{\mathrm{mark}}\}$ und damit die Aussage des Lemmas. □

**Korollar 3.11.** *Mit den Voraussetzungen aus Lemma 3.10 gilt für alle* $\ell \in \mathbb{N}$

$$\kappa(\mathcal{T}_\ell) \leq \max\{\kappa(\mathcal{T}_0), C_{\mathrm{mark}}\}. \tag{3.13}$$

**Beweis:** Der Beweis erfolgt mittels Induktion:
Für $\ell = 0$ erhält man $\kappa(\mathcal{T}_1) \le \max\{\kappa(\mathcal{T}_0), C_{\mathrm{mark}}\}$.
Mit der Induktionsbehauptung $\kappa(\mathcal{T}_\ell) \le \max\{\kappa(\mathcal{T}_0), C_{\mathrm{mark}}\}$ folgt

$$\kappa(\mathcal{T}_{\ell+1}) \le \max\{\kappa(\mathcal{T}_\ell), C_{\mathrm{mark}}\} \le \max\{\kappa(\mathcal{T}_0), C_{\mathrm{mark}}\}$$

und damit die Aussage des Korollars. □

**Bemerkung 3.12.** *Für die Implementierung wird* $C_{\mathrm{mark}} = 2\kappa(\mathcal{T}_0)$ *gewählt.* $\kappa(\mathcal{T}_\ell)$ *ist somit für alle* $\ell \in \mathbb{N}$ *beschränkt durch* $2\kappa(\mathcal{T}_0)$.

## 3.6 Konvergenz des adaptiven Verfahrens

Die Partition $\mathcal{T}_\ell$ sei so, dass gelte

$$\begin{aligned} \big\|h_\ell^{1/2}\big(V\Psi_\ell\big)'\big\|_{L^2(\Gamma)} &\le C_V \|\!|\!|\Psi_\ell|\!|\!| \quad \text{für } \Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell) \\ \big\|h_\ell^{1/2}\big((K+\tfrac12)g_\ell\big)'\big\|_{L^2(\Gamma)} &\le C_K \big\|g_\ell\big\|_{H^{1/2}(\Gamma)} \quad \text{für } g_\ell \in \mathcal{S}^1(\mathcal{T}_\ell). \end{aligned} \tag{3.14}$$

Außerdem gelten in diesen Abschnitt folgende Bezeichnungen:

- $\zeta_\ell := \big\|h_\ell^{1/2}\big(g - g_\ell\big)'\big\|_{L^2(\Gamma)}$ ist der Datenfehlerschätzer aus Satz 3.6,
- $\varrho_\ell := \big\|h_\ell^{1/2} R_\ell'\big\|_{L^2(\Gamma)}$ ist der Verfahrensfehlerschätzer aus Satz 3.5,
- $\omega_\ell := \big(\varrho_\ell^2 + \zeta_\ell^2\big)^{1/2}$ ist der Gesamtfehlerschätzer aus Satz 3.7.

Aus theoretischen Gründen definieren wir zusätzlich den modifizierten Gesamtfehlerschätzer

- $\overline{\omega}_\ell := \big(\varrho_\ell^2 + \alpha\zeta_\ell^2\big)^{1/2}$ mit einer Konstante $\alpha > 0$.

**Lemma 3.13.** *Verwendet man die lokalen Beiträge* $\overline{\omega}_\ell(T) = \big(\varrho_\ell(T)^2 + \alpha\zeta_\ell(T)^2\big)^{1/2}$ *des modifizierten Gesamtfehlerschätzers* $\overline{\omega}_\ell$ *im Dörfler-Marking* (3.10)*, so gilt unter der Voraussetzung* (3.14) *die Schätzerreduktionsbedingung*

$$\begin{aligned} \overline{\omega}_{\ell+1}^2 \le \kappa\overline{\omega}_\ell^2 &- \alpha C_1 \big\|h_\ell^{1/2}\big(g - g_\ell\big)'\big\|_{L_2(\Gamma)}^2 \\ &+ (C_2 - \alpha)\big\|h_{\ell+1}^{1/2}\big(g_{\ell+1} - g_\ell\big)'\big\|_{L_2(\Gamma)}^2 + C_2 \|\!|\!|\widetilde{\Phi}_{\ell+1} - \widetilde{\Phi}_\ell|\!|\!|^2. \end{aligned} \tag{3.15}$$

*Die Kontraktionskonstante* $\kappa \in (0,1)$ *und die Konstante* $C_1 > 0$ *hängen nur vom Parameter* $\theta \in (0,1)$ *aus der Dörfler-Markierung* (3.10)*, die Konstante* $C_2 > 0$ *nur von* $\theta$, $\Gamma$ *und* $\kappa(\mathcal{T}_\ell)$ *ab. Die drei Konstanten sind insbesondere unabhängig von* $\alpha > 0$. *Die Konstanten* $C_V$ *und* $C_K$ *aus* (3.14) *fließen in die Konstante* $C_2$ *ein.*

**Beweis:** Wie bereits bei der Konstruktion des Datenfehlerschätzers gezeigt — siehe Satz 3.6 — gilt für die $L^2$-Projektion $\Pi_\ell$ und die nodale Interpolation $\mathcal{I}_\ell$ der Zusammenhang $\Pi_\ell g' = (\mathcal{I}_\ell g)' = g_\ell'$. Der Fehlerschätzer lässt sich daher im $(\ell+1)$-ten Rechenschritt in der Form

$$\overline{\omega}_{\ell+1}^2 = \varrho_{\ell+1}^2 + \alpha\zeta_{\ell+1}^2 = \big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_{\ell+1} - V\widetilde{\Phi}_{\ell+1}\big)'\big\|_{L^2(\Gamma)}^2 + \alpha\big\|h_{\ell+1}^{1/2}(g - g_{\ell+1})'\big\|_{L^2(\Gamma)}^2$$

schreiben.
Unter Verwendung der Ungleichung von Young $ab \leq \frac{a^2}{2} + \frac{b^2}{2}$, den binomischen Formeln und dem Ansatz $ab = a\sqrt{\varepsilon}\,\frac{b}{\sqrt{\varepsilon}}$ für ein beliebiges $\varepsilon > 0$ erhält man

$$(a+b)^2 = a^2 + 2ab + b^2 \leq a^2 + a^2\varepsilon + \tfrac{b^2}{\varepsilon} + b^2 = (1+\varepsilon)a^2 + (1+\varepsilon^{-1})b^2. \tag{3.16}$$

Für beliebiges $\varepsilon > 0$ folgt daraus für den Fehlerschätzer

$$\begin{aligned}\overline{\omega}_{\ell+1}^2 \leq{}& (1+\varepsilon)\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(\Gamma)}^2 \\ &+ (1+\varepsilon^{-1})\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)(g_{\ell+1}-g_\ell) - V(\widetilde{\Phi}_{\ell+1}-\widetilde{\Phi}_\ell)\big)'\big\|_{L^2(\Gamma)}^2 \\ &+ \alpha\big\|h_{\ell+1}^{1/2}(g-g_{\ell+1})'\big\|_{L^2(\Gamma)}^2.\end{aligned}$$

Für die $L^2$-Projektion auf $\mathcal{P}^0(\mathcal{T}_{\ell+1})$ als elementweise Bestapproximation bezüglich der $L^2$-Norm gilt

$$\begin{aligned}&\big\|h_{\ell+1}^{1/2}(g-g_{\ell+1})'\big\|_{L^2(\Gamma)}^2 + \big\|h_{\ell+1}^{1/2}(g_{\ell+1}-g_\ell)'\big\|_{L^2(\Gamma)}^2 \\ &\quad= \big\|h_{\ell+1}^{1/2}(1-\Pi_{\ell+1})g'\big\|_{L^2(\Gamma)}^2 + \big\|h_{\ell+1}^{1/2}(\Pi_{\ell+1}-\Pi_\ell)g'\big\|_{L^2(\Gamma)}^2 \\ &\quad= \big\|h_{\ell+1}^{1/2}(1-\Pi_\ell)g'\big\|_{L^2(\Gamma)}^2 \\ &\quad= \big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2\end{aligned} \tag{3.17}$$

und damit für den Fehlerschätzer

$$\begin{aligned}\overline{\omega}_{\ell+1}^2 \leq{}& (1+\varepsilon)\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(\Gamma)}^2 \\ &+ (1+\varepsilon^{-1})\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)(g_{\ell+1}-g_\ell) - V(\widetilde{\Phi}_{\ell+1}-\widetilde{\Phi}_\ell)\big)'\big\|_{L^2(\Gamma)}^2 \\ &+ \alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 - \alpha\big\|h_{\ell+1}^{1/2}(g_{\ell+1}-g_\ell)'\big\|_{L^2(\Gamma)}^2.\end{aligned}$$

Unter Verwendung der Voraussetzung (3.14) erhält man

$$\begin{aligned}\overline{\omega}_{\ell+1}^2 \leq{}& (1+\varepsilon)\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(\Gamma)}^2 \\ &+ 2(1+\varepsilon^{-1})C_K^2\big\|g_{\ell+1}-g_\ell\big\|_{H^{1/2}(\Gamma)}^2 + 2(1+\varepsilon^{-1})C_V^2\big|\!\big|\!\big|\widetilde{\Phi}_{\ell+1}-\widetilde{\Phi}_\ell\big|\!\big|\!\big|^2 \\ &+ \alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 - \alpha\big\|h_{\ell+1}^{1/2}(g_{\ell+1}-g_\ell)'\big\|_{L^2(\Gamma)}^2 \\ ={}& (1+\varepsilon)\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(\Gamma)}^2 + \alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 \\ &+ 2(1+\varepsilon^{-1})C_K^2\big\|g_{\ell+1}-g_\ell\big\|_{H^{1/2}(\Gamma)}^2 + 2(1+\varepsilon^{-1})C_V^2\big|\!\big|\!\big|\widetilde{\Phi}_{\ell+1}-\widetilde{\Phi}_\ell\big|\!\big|\!\big|^2 - \alpha\big\|h_{\ell+1}^{1/2}(g_{\ell+1}-g_\ell)'\big\|_{L^2(\Gamma)}^2 \\ ={}& (1+\varepsilon)\Big(\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(\Gamma)}^2 + \alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2\Big) - \varepsilon\alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 \\ &+ 2(1+\varepsilon^{-1})C_K^2\big\|g_{\ell+1}-g_\ell\big\|_{H^{1/2}(\Gamma)}^2 + 2(1+\varepsilon^{-1})C_V^2\big|\!\big|\!\big|\widetilde{\Phi}_{\ell+1}-\widetilde{\Phi}_\ell\big|\!\big|\!\big|^2 - \alpha\big\|h_{\ell+1}^{1/2}(g_{\ell+1}-g_\ell)'\big\|_{L^2(\Gamma)}^2.\end{aligned}$$

Wir betrachten nun zunächst die beiden ersten Terme

$$\begin{aligned}&\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(\Gamma)}^2 + \alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 \\ &\quad= \sum_{T\in\mathcal{T}_\ell\setminus\mathcal{M}_\ell}\Big(\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(T)}^2 + \alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(T)}^2\Big) \\ &\qquad+ \sum_{T\in\mathcal{M}_\ell}\Big(\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(T)}^2 + \alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(T)}^2\Big).\end{aligned}$$

Mit $h_{\ell+1}(T) = \frac{1}{2}h_\ell(T)$ für $T \in \mathcal{M}_\ell$ und $h_{\ell+1}(T) \le h_\ell(T)$ für $T \in \mathcal{T}_\ell \setminus \mathcal{M}_\ell$ gilt

$$\begin{aligned}
&\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(\Gamma)}^2 + \alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 \\
&\quad\le \sum_{T\in\mathcal{T}_\ell\setminus\mathcal{M}_\ell}\Big(\big\|h_\ell^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(T)}^2 + \alpha\big\|h_\ell^{1/2}(g-g_\ell)'\big\|_{L^2(T)}^2\Big) \\
&\qquad+\frac12\sum_{T\in\mathcal{M}_\ell}\Big(\big\|h_\ell^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(T)}^2 + \alpha\big\|h_\ell^{1/2}(g-g_\ell)'\big\|_{L^2(T)}^2\Big) \\
&\quad= \sum_{T\in\mathcal{T}_\ell}\Big(\big\|h_\ell^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(T)}^2 + \alpha\big\|h_\ell^{1/2}(g-g_\ell)'\big\|_{L^2(T)}^2\Big) \\
&\qquad-\frac12\sum_{T\in\mathcal{M}_\ell}\Big(\big\|h_\ell^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(T)}^2 + \alpha\big\|h_\ell^{1/2}(g-g_\ell)'\big\|_{L^2(T)}^2\Big) \\
&\quad= \sum_{T\in\mathcal{T}_\ell}\overline{\omega}_\ell(T)^2 - \frac12\sum_{T\in\mathcal{M}_\ell}\overline{\omega}_\ell(T)^2.
\end{aligned}$$

Mit der Dörfler-Markierung (3.10) mit $\overline{\omega}_\ell$ und $\theta \in (0,1)$ gilt:

$$\theta\sum_{T\in\mathcal{T}_\ell}\overline{\omega}_\ell^2(T) \le \sum_{T\in\mathcal{M}_\ell}\overline{\omega}_\ell^2(T).$$

Einsetzen ergibt also

$$\big\|h_{\ell+1}^{1/2}\big((K+\tfrac12)g_\ell - V\widetilde{\Phi}_\ell\big)'\big\|_{L^2(\Gamma)}^2 + \alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 \le \Big(1-\frac{\theta}{2}\Big)\sum_{T\in\mathcal{T}_\ell}\overline{\omega}_\ell(T)^2 = \Big(1-\frac{\theta}{2}\Big)\overline{\omega}_\ell^2.$$

Insgesamt gilt daher für den Fehlerschätzer

$$\begin{aligned}
\overline{\omega}_{\ell+1}^2 \le{}& (1+\varepsilon)\Big(1-\frac{\theta}{2}\Big)\overline{\omega}_\ell^2 - \varepsilon\alpha\big\|h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 + 2(1+\varepsilon^{-1})C_K^2\big\|g_{\ell+1}-g_\ell\big\|_{H^{1/2}(\Gamma)}^2 \\
&+ 2(1+\varepsilon^{-1})C_V^2\big|\!\big|\!\big|\widetilde{\Phi}_{\ell+1}-\widetilde{\Phi}_\ell\big|\!\big|\!\big|^2 - \alpha\big\|h_{\ell+1}^{1/2}(g_{\ell+1}-g_\ell)'\big\|_{L^2(\Gamma)}^2.
\end{aligned}$$

Durch Anwenden von Korollar 3.3 um $\big\|g_{\ell+1}-g_\ell\big\|_{H^{1/2}(\Gamma)}^2$ abzuschätzen und der Bedingung $g_\ell = \mathcal{I}_\ell g_{\ell+1}$ erhält man

$$\begin{aligned}
\overline{\omega}_{\ell+1}^2 \le{}& (1+\varepsilon)\Big(1-\frac{\theta}{2}\Big)\overline{\omega}_\ell^2 - \varepsilon\alpha\big\|h_\ell^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 + 2(1+\varepsilon^{-1})C_K^2\widetilde{C}_{\mathrm{apx}}^2\big\|h_{\ell+1}^{1/2}\big(g_{\ell+1}-g_\ell\big)'\big\|_{L^2(\Gamma)}^2 \\
&+ 2(1+\varepsilon^{-1})C_V^2\big|\!\big|\!\big|\widetilde{\Phi}_{\ell+1}-\widetilde{\Phi}_\ell\big|\!\big|\!\big|^2 - \alpha\big\|h_{\ell+1}^{1/2}(g_{\ell+1}-g_\ell)'\big\|_{L^2(\Gamma)}^2 \\
\le{}& (1+\varepsilon)\Big(1-\frac{\theta}{2}\Big)\overline{\omega}_\ell^2 - \varepsilon\alpha\big\|h_\ell^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)}^2 \\
&+ 2(1+\varepsilon^{-1})\max\big\{C_K^2\widetilde{C}_{\mathrm{apx}}^2, C_V^2\big\}\Big(\big\|h_{\ell+1}^{1/2}\big(g_{\ell+1}-g_\ell\big)'\big\|_{L^2(\Gamma)}^2 + \big|\!\big|\!\big|\widetilde{\Phi}_{\ell+1}-\widetilde{\Phi}_\ell\big|\!\big|\!\big|^2\Big) \\
&- \alpha\big\|h_{\ell+1}^{1/2}(g_{\ell+1}-g_\ell)'\big\|_{L^2(\Gamma)}^2.
\end{aligned}$$

Wählt man nun $\varepsilon \in (0,1)$, sodass $\kappa := (1+\varepsilon)(1-\frac{\theta}{2}) < 1$ erfüllt ist, so folgt mit $C_1 := \varepsilon$ und $C_2 := 2(1+\varepsilon^{-1})\max\big\{C_K^2\widetilde{C}_{\mathrm{apx}}^2, C_V^2\big\}$ die Schätzerreduktionsaussage (3.15). □

**Satz 3.14.** *Dörflermarkierung* (3.10) *für* $\omega_\ell$ *liefert*

$$\lim_{\ell\to\infty}\big|\!\big|\!\big|\phi-\widetilde{\Phi}_\ell\big|\!\big|\!\big| = 0 = \lim_{\ell\to\infty}\omega_\ell. \tag{3.18}$$

**Beweis:** 1. Schritt: Die Fehlerschätzer $\omega_\ell$ und $\overline{\omega}_\ell$ sind äquivalent:
Es gelten

$$\overline{\omega}_\ell^2(T) = \varrho_\ell^2(T) + \alpha\zeta_\ell^2(T) \leq \max\{1,\alpha\}\big(\varrho_\ell^2(T) + \zeta_\ell^2(T)\big) = \max\{1,\alpha\}\omega_\ell^2(T)$$

und

$$\min\{1,\alpha\}\omega_\ell^2(T) = \min\{1,\alpha\}\big(\varrho_\ell^2(T) + \zeta_\ell^2(T)\big) \leq \varrho_\ell^2(T) + \alpha\zeta_\ell^2(T) = \overline{\omega}_\ell^2(T).$$

Insgesamt erhält man also

$$\min\{1,\alpha\}\omega_\ell^2(T) \leq \overline{\omega}_\ell^2(T) \leq \max\{1,\alpha\}\omega_\ell^2(T). \tag{3.19}$$

2. Schritt: Dörflermarkierung für $\omega_\ell$ mit Parameter $\theta$ impliziert Dörflermarkierung für $\overline{\omega}_\ell$ mit Parameter $\overline{\theta} := \theta\max\{1,\alpha\}^{-1}\min\{1,\alpha\} \in (0,1)$:
Aus der Äquivalenz der Fehlerschätzer folgt

$$\begin{aligned}\overline{\theta}\sum_{T\in\mathcal{T}_\ell}\overline{\omega}_\ell^2(T) = \theta\max\{1,\alpha\}^{-1}\min\{1,\alpha\}\sum_{T\in\mathcal{T}_\ell}\overline{\omega}_\ell^2(T) &\leq \theta\min\{1,\alpha\}\sum_{T\in\mathcal{T}_\ell}\omega_\ell^2(T)\\ &\leq \min\{1,\alpha\}\sum_{T\in\mathcal{M}_\ell}\omega_\ell^2(T)\\ &\leq \sum_{T\in\mathcal{M}_\ell}\overline{\omega}_\ell^2(T).\end{aligned} \tag{3.20}$$

Es sei an dieser Stelle angemerkt, dass wir den Parameter $\alpha > 0$ weiter unten im Beweis fixieren werden. 3. Schritt: Für ein geeignetes $\alpha > 0$ impliziert Dörflermarkierung für $\overline{\omega}_\ell$ eine Kontraktionsbedingung für $\overline{W}_\ell := \gamma\overline{\omega}_\ell^2 + |||\phi - \Phi_\ell|||^2$ mit einem geeigneten $\gamma > 0$, also die Existenz einer Kontraktionskonstanten $\overline{\kappa} \in (0,1)$, sodass $\overline{W}_{\ell+1} \leq \overline{\kappa}\overline{W}_\ell$ gilt:
Unter Verwendung von Lemma 3.13 und Ausnützung der Galerkinorthogonalität von $\Phi_{\ell+1}$ erhält man

$$\begin{aligned}\overline{W}_{\ell+1} &= \gamma\overline{\omega}_{\ell+1}^2 + |||\phi - \Phi_{\ell+1}|||^2\\ &\leq \gamma\Big[\kappa\overline{\omega}_\ell^2 - \alpha C_1\big\|h_\ell^{1/2}\big(g-g_\ell\big)'\big\|_{L_2(\Gamma)}^2 + (C_2-\alpha)\big\|h_{\ell+1}^{1/2}\big(g_{\ell+1}-g_\ell\big)'\big\|_{L_2(\Gamma)}^2\\ &\quad + C_2|||\widetilde{\Phi}_{\ell+1} - \widetilde{\Phi}_\ell|||^2\Big] + |||\phi - \Phi_\ell|||^2 - |||\Phi_{\ell+1} - \Phi_\ell|||^2.\end{aligned}$$

Der Term $|||\widetilde{\Phi}_{\ell+1} - \widetilde{\Phi}_\ell|||$ kann aufgrund der Stetigkeit der Galerkinprojektionen $\mathbb{G}_\ell$ und $\mathbb{G}_{\ell+1}$ nach oben abgeschätzt werden durch

$$\begin{aligned}|||\widetilde{\Phi}_{\ell+1} - \widetilde{\Phi}_\ell|||^2 &\leq 2|||\widetilde{\Phi}_{\ell+1} - \widetilde{\Phi}_\ell - \Phi_{\ell+1} + \Phi_\ell|||^2 + 2|||\Phi_{\ell+1} - \Phi_\ell|||^2\\ &= 2|||(\widetilde{\Phi}_{\ell+1} - \Phi_{\ell+1}) - (\widetilde{\Phi}_\ell - \Phi_\ell)|||^2 + 2|||\Phi_{\ell+1} - \Phi_\ell|||^2\\ &\leq 4|||\widetilde{\Phi}_{\ell+1} - \Phi_{\ell+1}|||^2 + 4|||\widetilde{\Phi}_\ell - \Phi_\ell|||^2 + 2|||\Phi_{\ell+1} - \Phi_\ell|||^2\\ &\leq 4|||\widetilde{\phi}_{\ell+1} - \phi|||^2 + 4|||\widetilde{\phi}_\ell - \phi|||^2 + 2|||\Phi_{\ell+1} - \Phi_\ell|||^2.\end{aligned}$$

Nach Satz 3.6, der eine Abschätzung für Datenfehler garantiert, gilt damit

$$|||\widetilde{\Phi}_{\ell+1} - \widetilde{\Phi}_\ell|||^2 \leq C_{\mathrm{data}}\big\|h_{\ell+1}^{1/2}(g-g_{\ell+1})'\big\|_{L_2(\Gamma)}^2 + C_{\mathrm{data}}\big\|h_\ell^{1/2}(g-g_\ell)'\big\|_{L_2(\Gamma)}^2 + 2|||\Phi_{\ell+1} - \Phi_\ell|||^2.$$

Die Konstante $C_{\mathrm{data}}$ ist nur abhängig von $\Gamma$ und $\kappa(\mathcal{T}_\ell)$.
Durch Einsetzen erhält man

$$\begin{aligned}\overline{W}_{\ell+1} \leq{}& \gamma\kappa\overline{\omega}_\ell^2 - \gamma\alpha C_1 \big\| h_\ell^{1/2}\big(g-g_\ell\big)'\big\|_{L_2(\Gamma)}^2 + \gamma(C_2-\alpha)\big\| h_{\ell+1}^{1/2}\big(g_{\ell+1}-g_\ell\big)'\big\|_{L_2(\Gamma)}^2 \\ &+ \gamma C_2 C_{\mathrm{data}} \big\| h_{\ell+1}^{1/2}(g-g_{\ell+1})'\big\|_{L_2(\Gamma)}^2 + \gamma C_2 C_{\mathrm{data}} \big\| h_\ell^{1/2}(g-g_\ell)'\big\|_{L_2(\Gamma)}^2 + 2\gamma C_2 \big|\!\big|\!\big| \Phi_{\ell+1}-\Phi_\ell \big|\!\big|\!\big|^2 \\ &+ \big|\!\big|\!\big| \phi-\Phi_\ell \big|\!\big|\!\big|^2 - \big|\!\big|\!\big| \Phi_{\ell+1}-\Phi_\ell \big|\!\big|\!\big|^2.\end{aligned}$$

Durch Ausnutzen der elementweisen Bestapproximationseigenschaft der $L^2$-Projektion und der Gültigkeit von $h_{\ell+1} \leq h_\ell$ ergibt sich

$$\big\| h_{\ell+1}^{1/2}(g-g_{\ell+1})'\big\|_{L_2(\Gamma)}^2 \leq \big\| h_{\ell+1}^{1/2}(g-g_\ell)'\big\|_{L_2(\Gamma)}^2 \leq \big\| h_\ell^{1/2}(g-g_\ell)'\big\|_{L_2(\Gamma)}^2.$$

Zusammenfassen und die obige Abschätzung führen zu

$$\begin{aligned}\overline{W}_{\ell+1} \leq{}& \gamma\kappa\overline{\omega}_\ell^2 + \gamma(C_2-\alpha)\big\| h_{\ell+1}^{1/2}\big(g_{\ell+1}-g_\ell\big)'\big\|_{L_2(\Gamma)}^2 + \gamma(2C_2C_{\mathrm{data}}-\alpha C_1)\big\| h_\ell^{1/2}\big(g-g_\ell\big)'\big\|_{L_2(\Gamma)}^2 \\ &+ (2\gamma C_2-1)\big|\!\big|\!\big| \Phi_{\ell+1}-\Phi_\ell \big|\!\big|\!\big|^2 + \big|\!\big|\!\big| \phi-\Phi_\ell \big|\!\big|\!\big|^2.\end{aligned}$$

Wir fixieren nun die Konstanten $\alpha > 0$ und $\gamma > 0$:

- Wir wählen $\alpha > 0$ groß genug, sodass $C_2 \leq \alpha$ und $2C_2C_{\mathrm{data}} \leq \alpha C_1$.
- Ferner wählen wir $\gamma > 0$ klein genug, sodass $2\gamma C_2 \leq 1$.

Dann folgt für alle $\beta > 0$

$$\overline{W}_{\ell+1} \leq \gamma\kappa\overline{\omega}_\ell^2 + \big|\!\big|\!\big| \phi-\Phi_\ell \big|\!\big|\!\big|^2 = \gamma(\kappa+\beta)\overline{\omega}_\ell^2 + \big|\!\big|\!\big| \phi-\Phi_\ell \big|\!\big|\!\big|^2 - \gamma\beta\overline{\omega}_\ell^2.$$

Der Gesamtfehlerschätzer $\overline{\omega}_\ell$ ist zuverlässig für $\big|\!\big|\!\big| \phi-\Phi_\ell \big|\!\big|\!\big|$ mit Zuverlässigkeitskonstante $\overline{C}_{\mathrm{rel}} := (C_{\mathrm{rel}} + C_{\mathrm{data}})\big(\min\{1,\alpha\}\big)^{-1/2}$.
Aus der Zuverlässigkeit von $\omega_\ell$ für $\big|\!\big|\!\big| \phi-\widetilde{\Phi}_\ell \big|\!\big|\!\big|$, der Abschätzung für den Datenfehler aus Satz 3.6 und der Stetigkeit der Galerkinprojektion folgt nämlich

$$\begin{aligned}\big|\!\big|\!\big| \phi-\Phi_\ell \big|\!\big|\!\big| \leq \big|\!\big|\!\big| \phi-\widetilde{\Phi}_\ell \big|\!\big|\!\big| + \big|\!\big|\!\big| \widetilde{\Phi}_\ell-\Phi_\ell \big|\!\big|\!\big| \leq C_{\mathrm{rel}}\omega_\ell + \big|\!\big|\!\big| \widetilde{\phi}_\ell-\phi \big|\!\big|\!\big| &\leq C_{\mathrm{rel}}\omega_\ell + C_{\mathrm{data}}\zeta_\ell \\ &\leq C_{\mathrm{rel}}\omega_\ell + C_{\mathrm{data}}\omega_\ell \\ &= (C_{\mathrm{rel}}+C_{\mathrm{data}})\omega_\ell \\ &\leq (C_{\mathrm{rel}}+C_{\mathrm{data}})(\min\{1,\alpha\})^{-1/2}\overline{\omega}_\ell.\end{aligned} \tag{3.21}$$

Deshalb folgt

$$\overline{W}_{\ell+1} \leq \gamma(\kappa+\beta)\overline{\omega}_\ell^2 + \big(1-\overline{C}_{\mathrm{rel}}^{-2}\gamma\beta\big)\big|\!\big|\!\big| \phi-\Phi_\ell \big|\!\big|\!\big|^2 \leq \max\{\kappa+\beta, 1-\overline{C}_{\mathrm{rel}}^{-2}\gamma\beta\}\big(\gamma\overline{\omega}_\ell^2 + \big|\!\big|\!\big| \phi-\Phi_\ell \big|\!\big|\!\big|^2\big).$$

Wählt man nun $\beta$ klein genug, sodass $\kappa+\beta < 1$ und $\gamma\beta\overline{C}_{\mathrm{rel}}^{-2} < 1$, so erhält man die gewünschte Reduktionsbedingung $\overline{W}_{\ell+1} \leq \overline{\kappa}\overline{W}_\ell$ mit $\overline{\kappa} := \max\{\kappa+\beta, 1-\overline{C}_{\mathrm{rel}}^{-2}\gamma\beta\} \in (0,1)$.

4.Schritt: Die Kontraktionsbedingung $\overline{W}_{\ell+1} \leq \overline{\kappa}\overline{W}_\ell$ impliziert die Konvergenz des adaptiven Verfahrens:
Aufgrund der Reduktionsbedingung für $\overline{W}_\ell$ gilt $\lim_{\ell\to\infty}\overline{W}_\ell = 0$. Daher muss auch jeder der beiden positiven Summanden von $\overline{W}_\ell$ gegen 0 gehen. Insbesondere gilt also $\lim_{\ell\to\infty}\overline{\omega}_\ell = 0$. Wegen der Äquivalenz der Fehlerschätzer erfüllt auch $\omega_\ell$ die Bedingung $\lim_{\ell\to\infty}\omega_\ell = 0$. Aus der Zuverlässigkeit des Gesamtfehlerschätzers aus Satz 3.7 folgt schließlich die Konvergenz des adaptiven Verfahrens. □

**Bemerkung 3.15.** *Die Voraussetzungen* (3.14) *führen auf das Lösen eines Eigenwertproblems. Dies wird hier exemplarisch für die erste Gleichung durchgeführt. Ausgehend von*

$$\big\|h_\ell^{1/2}\big(V\Psi_\ell\big)'\big\|_{L^2(\Gamma)}^2 \le C_V^2 \,\big|\!\big|\!\big|\Psi_\ell\big|\!\big|\!\big|^2$$

*erhält man*

$$\lambda := C_V^2 = \max_{\Psi_\ell\in\mathcal{P}^0(\mathcal{T}_\ell)} \frac{\big\|h_\ell^{1/2}\big(V\Psi_\ell\big)'\big\|_{L^2(\Gamma)}^2}{\big|\!\big|\!\big|\Psi_\ell\big|\!\big|\!\big|^2}.$$

*Wegen $\Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)$ existiert eine Darstellung $\Psi_\ell = \sum_{k=1}^N x_k\chi_k$, wobei $\mathcal{T}_\ell = \{T_1,\dots T_N\}$ gilt und $\chi_k$ die charakteristische Funktion des Elements $T_k$ bezeichnet. Mit $A_{ij} := \langle V\chi_i,\chi_j\rangle$ und $B_{ij} := \langle h_\ell^{1/2}(V\chi_i)', h_\ell^{1/2}(V\chi_j)'\rangle$ ist*

$$\big|\!\big|\!\big|\Psi_\ell\big|\!\big|\!\big|^2 = x\cdot Ax \qquad \text{und} \qquad \big\|h_\ell^{1/2}\big(V\Psi_\ell\big)'\big\|_{L^2(\Gamma)}^2 = x\cdot Bx.$$

*Zu Lösen ist also das verallgemeinerte Eigenwertproblem*

$$\lambda = \max_{x\in\mathbb{R}^N\setminus\{0\}} \frac{x\cdot Bx}{x\cdot Ax},$$

*das sich wie folgt auf ein gewöhnliches Eigenwertproblem zurückführen lässt. Da $A$ eine symmetrische und positiv definite Matrix ist, existiert ihre Cholesky-Zerlegung und $Ax$ lässt sich in der Form $Ax = (LL^T)x$ schreiben. Setzt man $y := L^Tx$ und $M := L^{-1}BL^{-T}$, so folgt*

$$\lambda = \max_{y\in\mathbb{R}^N\setminus\{0\}} \frac{(L^{-T}y)\cdot B(L^{-T}y)}{\|y\|_2^2} = \max_{y\in\mathbb{R}^N\setminus\{0\}} \frac{y\cdot(L^{-1}BL^{-T})y}{\|y\|_2^2} = \max_{y\in\mathbb{R}^N\setminus\{0\}} \frac{y\cdot My}{\|y\|_2^2}.$$

*Die Matrix $M$ ist symmetrisch und positiv semidefinit. Aus der Symmetrie von $M$ folgt, dass der maximale Eigenwert $\lambda = \lambda_{\max}$ von $M$ gesucht wird.*

**Bemerkung 3.16.** *Wir betrachten nun die Voraussetzungen* (3.14) *für quasiuniforme Gitter, d.h für alle $\ell\in\mathbb{N}$ existieren Konstanten $h_{\ell,\min}, h_{\ell,\max} > 0$, sodass für alle $T\in\mathcal{T}_\ell$ gilt $h_{\ell,\min} \le h_\ell(T) \le h_{\ell,\max}$ und $\sup_\ell \frac{h_{\ell,\max}}{h_{\ell,\min}} =: H < \infty$. Für die erste Ungleichung aus* (3.14) *folgt aufgrund der Stetigkeit des Einfachschichtpotentials $V \in L\big(L^2(\Gamma), H^1(\Gamma)\big)$ — siehe Satz 2.24 —*

$$\begin{aligned}\big\|h_\ell^{1/2}\big(V\Psi_\ell\big)'\big\|_{L^2(\Gamma)}^2 \le h_{\ell,\max}\big\|\big(V\Psi_\ell\big)'\big\|_{L^2(\Gamma)}^2 \le h_{\ell,\max}\big\|V\Psi_\ell\big\|_{H^1(\Gamma)}^2 &\le C_3 h_{\ell,\max}\big\|\Psi_\ell\big\|_{L^2(\Gamma)}^2\\ &\le C_3 H h_{\ell,\min}\big\|\Psi_\ell\big\|_{L^2(\Gamma)}^2.\end{aligned}$$

*Mit der inversen Abschätzung aus Satz 4.2 erhält man also*

$$h_{\ell,\min}\big\|\Psi_\ell\big\|_{L^2(\Gamma)}^2 \le \big\|h_\ell^{1/2}\big(V\Psi_\ell\big)'\big\|_{L^2(\Gamma)}^2 \le C_{\mathrm{inv}}^2 C_3 \,\big|\!\big|\!\big|\Psi_\ell\big|\!\big|\!\big|^2.$$

*Die Konstanten $C_3 := \|V : L^2(\Gamma)\to H^1(\Gamma)\|^2$ und $C_{\mathrm{inv}}$ sind nur abhängig von $\Gamma$ und $\kappa(\mathcal{T}_\ell)$.*

*Wir betrachten nun die zweite Ungleichung von* (3.14). *Aus der Stetigkeit von $K+\frac12 \in L(H^1(\Gamma), H^1(\Gamma))$ mit einer Konstanten $C_4$ folgt*

$$\begin{aligned}\big\|h_\ell^{1/2}\big((K+\tfrac12)g_\ell\big)'\big\|_{L^2(\Gamma)}^2 \le h_{\ell,\max}\big\|\big((K+\tfrac12)g_\ell\big)'\big\|_{L^2(\Gamma)}^2 \le h_{\ell,\max}\big\|(K+\tfrac12)g_\ell\big\|_{H^1(\Gamma)}^2 &\\ \le C_4^2 h_{\ell,\max}\big\|g_\ell\big\|_{H^1(\Gamma)}^2.&\end{aligned}$$

*Aufgrund der inversen Abschätzung in Satz 3.4 und der Stetigkeit der Einbettung* $H^{1/2}(\Gamma) \subseteq L^2(\Gamma)$ *mit einer Konstanten* $C_5$ *gilt*

$$\begin{aligned}\|g_\ell\|_{H^1(\Gamma)}^2 \le \|g_\ell\|_{L^2(\Gamma)}^2 + \|g_\ell'\|_{L^2(\Gamma)}^2 &\le \|g_\ell\|_{L^2(\Gamma)}^2 + h_{\ell,\min}^{-1/2}\|h_\ell^{1/2} g_\ell'\|_{L^2(\Gamma)}^2 \\ &\le C_5^2\|g_\ell\|_{H^{1/2}(\Gamma)}^2 + \widetilde{C}_{\rm inv}^2 h_{\ell,\min}^{-1/2}\|g_\ell\|_{H^{1/2}(\Gamma)}^2 \\ &\le \big(C_5^2 + \widetilde{C}_{\rm inv}^2 h_{\ell,\min}^{-1}\big)\|g_\ell\|_{H^{1/2}(\Gamma)}^2.\end{aligned}$$

*Einsetzen ergibt*

$$\|h_\ell^{1/2}\big((K+\tfrac12)g_\ell\big)'\|_{L^2(\Gamma)}^2 \le C_4^2\big(h_{\ell,\max}C_5^2 + H\widetilde{C}_{\rm inv}^2\big)\|g_\ell\|_{H^{1/2}(\Gamma)}^2 \le C_4^2\big(|\Gamma|C_3 + H\widetilde{C}_{\rm inv}^2\big)\|g_\ell\|_{H^{1/2}(\Gamma)}^2.$$

*Die Konstanten* $C_4 := \|(K+\frac12) : H^1(\Gamma) \to H^1(\Gamma)\|$, $C_5$ *und* $\widetilde{C}_{\rm inv}$ *hängen nur von* $\Gamma$ *und* $\kappa(\mathcal{T}_\ell)$ *ab.*

**Bemerkung 3.17.** *Im Fall von nichtgestörter rechter Seite* $g = g_\ell$ *folgt aus Satz 3.14 insbesondere die Konvergenz des adaptiven Algorithmus. Der Beweis zeigt dann, dass* $\overline{W}_\ell := \gamma\varrho_\ell^2 + |||\phi - \Phi_\ell|||^2$ *für geeignetes* $\gamma > 0$ *kontrahiert, d.h es existiert eine Konstante* $\overline{\kappa} \in (0,1)$ *mit* $\overline{W}_{\ell+1} \le \overline{\kappa}\overline{W}_\ell$*, und insbesondere folgt* $\lim_{\ell\to\infty}\varrho_\ell = 0 = \lim_{\ell\to\infty}|||\phi - \Phi_\ell|||$.

# Kapitel 4

# Adaptiver Algorithmus mit $h - h/2$-Fehlerschätzer

In diesem Abschnitt werden zunächst der a posteriori Fehlerschätzer $\eta_\ell$, der den Fehler in der Energienorm $|||\cdot|||$ misst, und dessen Eigenschaften angeführt. $\eta_\ell$ ist ein so genannter $h - h/2$-Fehlerschätzer, verwendet also zwei Partitionen $\mathcal{T}_\ell$ und $\widehat{\mathcal{T}}_\ell$, wobei $\widehat{\mathcal{T}}_\ell$ aus $\mathcal{T}_\ell$ durch uniforme Verfeinerung entsteht. Mittels Lokalisierungstechniken lassen sich auch Fehlerschätzer in der $L^2$-Norm entwickeln, die äquivalent zu $\eta_\ell$ sind.

Ein adaptiver Algorithmus, der mit einem aus einem $h - h/2$-Schätzer und dem Datenfehlerschätzer aus Satz 3.6 bestehenden Gesamtfehlerschätzer gesteuert wird, wird angegeben.

Seien im Folgenden

- $\phi \in H^{-1/2}(\Gamma)$ die kontinuierliche Lösung von $V\phi = (K + \frac{1}{2})g =: F \in H^{1/2}(\Gamma)$
- $\widetilde{\phi}_\ell \in H^{-1/2}(\Gamma)$ die kontinuierliche Lösung von $V\widetilde{\phi}_\ell = (K + \frac{1}{2})g_\ell =: F_\ell \in H^{1/2}(\Gamma)$
- $\Phi_\ell$ die Galerkinlösung zu $\phi$ in $X_\ell := \mathcal{P}^0(\mathcal{T}_\ell)$
- $\widetilde{\Phi}_\ell$ die Galerkinlösung zu $\widetilde{\phi}_\ell$ in $X_\ell$
- $\widehat{\Phi}_\ell$ die Galerkinlösung zu $\phi$ in $\widehat{X}_\ell := \mathcal{P}^0(\widehat{\mathcal{T}}_\ell)$
- $\widehat{\widetilde{\Phi}}_\ell$ die Galerkinlösung zu $\widetilde{\phi}_\ell$ in $\widehat{X}_\ell$.

Man nennt einen Fehlerschätzer $\eta$ effizient im Bezug auf $|||\phi - \Phi_\ell|||$, wenn für eine Konstante $C_{\text{eff}} > 0$ gilt

$$\eta \leq C_{\text{eff}} |||\phi - \Phi_\ell|||.$$

Die Zuverlässigkeit der $h - h/2$-Fehlerschätzer für das ungestörte Problem ist nur unter der Saturationsannahme $|||\phi - \widehat{\Phi}_\ell||| \leq C_{\text{sat}} |||\phi - \Phi_\ell|||$ für $C_{\text{sat}} \in (0, 1)$ gegeben. Auch auf die Saturationsannahme für das gestörte Problem wird näher eingegangen.

## 4.1 Lokalisierungtechniken für $H^{-1/2}(\Gamma)$

Dieses Kapitel beschreibt den Übergang von der Energienorm zu lokalen Normen um im Weiteren lokale Fehlerschätzer entwickeln zu können.

Der folgende Satz beschreibt die Approximationseigenschaft der $L^2$-Projektion. Eine Formulierung des Satzes für beliebige Normen $\|\cdot\|_{H^{-\alpha}}$ und ausführlichem Beweis lässt sich in Carstensen/Praetorius 2006 [9] finden. Für uns ist die Aussage für $\alpha = 1/2$ ausreichend:

**Satz 4.1. (Carstensen/Praetorius 2006 [9, Lemma 4.3]).** *Ist $\Pi_\ell$ die $L^2$-Projektion auf $\mathcal{P}^0(\mathcal{T}_\ell)$ und $v \in L^2(\Gamma)$, so gilt*

$$|||v - \Pi_\ell v||| \leq C_{\text{apx}} \|h_\ell^{1/2}(v - \Pi_\ell v)\|_{L^2(\Gamma)} \leq C_{\text{apx}} \|h_\ell^{1/2} v\|_{L^2(\Gamma)}. \tag{4.1}$$

*Die Konstante $C_{\text{apx}} > 0$ hängt hierbei nur von $\Gamma$, nicht aber von der Partition $\mathcal{T}_\ell$ ab.* □

Für diskrete Funktionen ist es auch möglich die $L^2$-Norm durch die Energienorm abzuschätzen. Der folgende Satz stammt aus Graham/Hackbusch/Sauter 2005, wurde aber an die Notation dieser Arbeit angepasst und hier nur für einen speziellen Fall formuliert.

**Satz 4.2. (Graham/Hackbusch/Sauter 2005 [13, Theorem 3.6]).** *Für eine diskrete Funktion $\Psi_\ell \in \mathcal{P}^0(\mathcal{T}_\ell)$ gilt die inverse Abschätzung*

$$\|h_\ell^{1/2} \Psi_\ell\|_{L^2(\Gamma)} \leq C_{\text{inv}} |||\Psi_\ell|||. \tag{4.2}$$

*Die Konstante $C_{\text{inv}} > 0$ hängt nur von $\Gamma$ und $\kappa(\mathcal{T}_\ell)$ ab.* □

## 4.2 $h - h/2$-Fehlerschätzer für ungestörtes Galerkinverfahren

Wir betrachten zunächst den Fehler $|||\phi - \Phi_\ell|||$ zwischen der diskreten und kontinuierlichen Lösung der ungestörten Symmschen Integralgleichung und entsprechende $h - h/2$-Fehlerschätzer.

**Satz 4.3. (Ferraz-Leite /Praetorius 2007 [12, Proposition 1.1]).** *Der Fehlerschätzer*

$$\eta_\ell := |||\Phi_\ell - \widehat{\Phi}_\ell||| \tag{4.3}$$

*ist effizient für $|||\phi - \Phi_\ell|||$ mit $C_{\text{eff}} = 1$. Unter der Saturationsannahme mit $C_{\text{sat}} \in (0, 1)$*

$$|||\phi - \widehat{\Phi}_\ell||| \leq C_{\text{sat}} |||\phi - \Phi_\ell||| \tag{4.4}$$

*ist $\eta_\ell$ auch zuverlässig in Bezug auf $|||\phi - \Phi_\ell|||$ mit $C_{\text{rel}} = 1/\sqrt{1 - C_{\text{sat}}^2}$.*

**Beweis:** Für die endlichdimensionalen Räume $X_\ell$ und $\widehat{X}_\ell$ gilt $X_\ell \subseteq \widehat{X}_\ell$. Daher ist $\Phi_\ell - \widehat{\Phi}_\ell \in \widehat{X}_\ell$. Durch Anwenden des Satzes von Pythagoras, was aufgrund der Galerkinorthogonalität für $\phi - \widehat{\Phi}_\ell$ erlaubt ist, erhält man

$$|||\phi - \Phi_\ell|||^2 = |||\phi - \widehat{\Phi}_\ell|||^2 + |||\widehat{\Phi}_\ell - \Phi_\ell|||^2 = |||\phi - \widehat{\Phi}_\ell|||^2 + \eta_\ell^2$$

und somit die Effizienz von $\eta_\ell$ mit $C_{\text{eff}} = 1$:

$$\eta_\ell \leq |||\phi - \Phi_\ell|||.$$

Mit derselben Argumentation wie zuvor und dem Anwenden der Saturationsannahme gilt

$$|||\phi - \Phi_\ell|||^2 = |||\phi - \widehat{\Phi}_\ell|||^2 + |||\widehat{\Phi}_\ell - \Phi_\ell|||^2 \leq C_{\text{sat}}^2 |||\phi - \Phi_\ell|||^2 + \eta_\ell^2.$$

Durch Umformen erhält man die Zuverlässigkeit von $\eta_\ell$ mit $C_{\text{rel}} = 1/\sqrt{1 - C_{\text{sat}}^2}$

$$|||\phi - \Phi_\ell||| \leq \frac{1}{\sqrt{1 - C_{\text{sat}}^2}} \eta_\ell$$

und damit die Aussage des Satzes. □

Wie bereits zuvor erwähnt, ist die Energienorm nicht lokal und man versucht daher einen Fehlerschätzer in der $L^2$-Norm mit denselben Eigenschaften wie $\eta_\ell$ zu finden.

**Definition 4.4.** *Sei $\Pi_\ell : L^2(\Gamma) \to \mathcal{P}^0(\mathcal{T}_\ell)$ die $L^2$-Projektion auf $\mathcal{P}^0(\mathcal{T}_\ell)$. Wir definieren den Fehlerschätzer*

$$\mu_\ell := \|h_\ell^{1/2}(\widehat{\Phi}_\ell - \Phi_\ell)\|_{L^2(\Gamma)}. \tag{4.5}$$

**Satz 4.5. (Ferraz-Leite/Praetorius 2007 [12, Theorem 3.2 und Theorem 3.4]).**
*Die Fehlerschätzer $\mu_\ell$ und $\eta_\ell$ sind äquivalent, d.h*

$$(\sqrt{2}C_{\text{inv}})^{-1}\mu_\ell \leq \eta_\ell \leq C_{\text{apx}}\mu_\ell. \tag{4.6}$$

**Beweis:** $\mathbb{G}_\ell$ bezeichne die Galerkinprojektion auf $\mathcal{P}^0(\mathcal{T}_\ell)$.
$\widehat{\mathcal{T}}_\ell$ entsteht aus $\mathcal{T}_\ell$ durch uniforme Verfeinerung, d.h $\mathcal{P}^0(\mathcal{T}_\ell) \subset \mathcal{P}^0(\widehat{\mathcal{T}}_\ell)$ und somit ist

$$\mathbb{G}_\ell \widehat{\Phi}_\ell = \Phi_\ell. \tag{4.7}$$

Gleichung (4.7) und die Bestapproximationseigenschaft der Galerkinprojektion liefern

$$\eta_\ell = |||\widehat{\Phi}_\ell - \Phi_\ell||| = |||\widehat{\Phi}_\ell - \mathbb{G}_\ell\widehat{\Phi}_\ell||| \leq |||\widehat{\Phi}_\ell - \Pi_\ell\widehat{\Phi}_\ell|||.$$

Mittels der Approximationseigenschaft für die $L^2$-Projektion aus Satz 4.1 folgt:

$$\eta_\ell \leq C_{\text{apx}}\|h_\ell^{1/2}(\widehat{\Phi}_\ell - \Pi_\ell\widehat{\Phi}_\ell)\|_{L^2(\Gamma)}.$$

Die $L^2$-Projektion $\Pi_\ell$ erfüllt elementweise die Bestapproximationseigenschaft

$$\|\widehat{\Phi}_\ell - \Pi_\ell\widehat{\Phi}_\ell\|_{L^2(T)} \leq C_{\text{apx}}\|\widehat{\Phi}_\ell - \mathbb{G}_\ell\widehat{\Phi}_\ell\|_{L^2(T)},$$

und damit folgt durch Summation über alle Elemente $T \in \mathcal{T}_\ell$

$$\eta_\ell \leq C_{\text{apx}}\|h_\ell^{1/2}(\widehat{\Phi}_\ell - \mathbb{G}_\ell\widehat{\Phi}_\ell)\|_{L^2(\Gamma)} = C_{\text{apx}}\|h_\ell^{1/2}(\widehat{\Phi}_\ell - \Phi_\ell)\|_{L^2(\Gamma)} = C_{\text{apx}}\mu_\ell.$$

Nun fehlt noch zu zeigen, dass $\mu_\ell \lesssim \eta_\ell$ gilt. Direkt aus der inversen Abschätzung in Satz 4.2 folgt

$$\begin{aligned}\mu_\ell = \|h_\ell^{1/2}(\widehat{\Phi}_\ell - \Phi_\ell)\|_{L^2(\Gamma)} = \sqrt{2}\|(h_\ell/2)^{1/2}(\widehat{\Phi}_\ell - \mathbb{G}_\ell\widehat{\Phi}_\ell)\|_{L^2(\Gamma)} &\leq \sqrt{2}C_{\text{inv}}|||\widehat{\Phi}_\ell - \Phi_\ell||| \\ &= \sqrt{2}C_{\text{inv}}\eta_\ell\end{aligned}$$

und somit die Äquivalenz der Fehlerschätzer. □

**Korollar 4.6.** *Der $h - h/2$-Fehlerschätzer $\mu_\ell$ ist effizient und unter der Saturationsannahme $|||\phi - \widehat{\Phi}_\ell||| \leq C_{\text{sat}}|||\phi - \Phi_\ell|||$ zuverlässig mit Bezug auf $|||\phi - \Phi_\ell|||$.* □

## 4.3 Saturationsannahme

**Satz 4.7.** *Aus der Saturationsannahme für das ungestörte Problem* $\|\!|\phi - \widehat{\Phi}_\ell|\!\| \le C_{\mathrm{sat}} \|\!|\phi - \Phi_\ell|\!\|$ *mit* $C_{\mathrm{sat}} \in (0,1)$ *und der Abschätzung* $\|\!|\phi - \widetilde{\phi}_\ell|\!\| \le \lambda \|\!|\phi - \Phi_\ell|\!\|$ *für ein geeignetes* $\lambda > 0$*, folgt*

$$\|\!|\widetilde{\phi}_\ell - \widehat{\widetilde{\Phi}}_\ell|\!\| \le \widetilde{C}_{\mathrm{sat}} \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\| \tag{4.8}$$

*mit* $\widetilde{C}_{\mathrm{sat}} := C_{\mathrm{sat}} + 2\frac{\lambda}{1-\lambda}(1 + C_{\mathrm{sat}}) < 1$.

**Beweis:** Durch Anwenden der Dreiecksungleichung und der Saturationsannahme erhält man

$$\begin{aligned} \|\!|\widetilde{\phi}_\ell - \widehat{\widetilde{\Phi}}_\ell|\!\| &\le \|\!|\widetilde{\phi}_\ell - \phi|\!\| + \|\!|\phi - \widehat{\Phi}_\ell|\!\| + \|\!|\widehat{\Phi}_\ell - \widehat{\widetilde{\Phi}}_\ell|\!\| \\ &\le \|\!|\phi - \widetilde{\phi}_\ell|\!\| + C_{\mathrm{sat}} \|\!|\phi - \Phi_\ell|\!\| + \|\!|\widehat{\Phi}_\ell - \widehat{\widetilde{\Phi}}_\ell|\!\|. \end{aligned}$$

Wegen der Stetigkeit der Galerkinprojektion und der Dreiecksungleichung gilt

$$\begin{aligned} \|\!|\widetilde{\phi}_\ell - \widehat{\widetilde{\Phi}}_\ell|\!\| &\le \|\!|\phi - \widetilde{\phi}_\ell|\!\| + C_{\mathrm{sat}}\big(\|\!|\phi - \widetilde{\phi}_\ell|\!\| + \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\| + \|\!|\widetilde{\Phi}_\ell - \Phi_\ell|\!\|\big) + \|\!|\widehat{\Phi}_\ell - \widehat{\widetilde{\Phi}}_\ell|\!\| \\ &\le \|\!|\phi - \widetilde{\phi}_\ell|\!\| + C_{\mathrm{sat}}\big(\|\!|\phi - \widetilde{\phi}_\ell|\!\| + \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\| + \|\!|\widetilde{\phi}_\ell - \phi|\!\|\big) + \|\!|\phi - \widetilde{\phi}_\ell|\!\| \\ &= C_{\mathrm{sat}} \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\| + 2(1 + C_{\mathrm{sat}}) \|\!|\phi - \widetilde{\phi}_\ell|\!\|. \end{aligned} \tag{4.9}$$

Durch Anwenden der Dreiecksungleichung, der Voraussetzung an den Datenfehler und der Bestapproximationseigenschaft der Galerkinprojektion folgt für den Datenfehler

$$\|\!|\phi - \widetilde{\phi}_\ell|\!\| \le \lambda \|\!|\phi - \Phi_\ell|\!\| \le \lambda \|\!|\phi - \widetilde{\Phi}_\ell|\!\| \le \lambda \|\!|\phi - \widetilde{\phi}_\ell|\!\| + \lambda \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\|.$$

Umformung führt zu

$$\|\!|\phi - \widetilde{\phi}_\ell|\!\| \le \frac{\lambda}{1-\lambda} \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\|.$$

Einsetzen dieser Bedingung in Formel (4.9) liefert

$$\begin{aligned} \|\!|\widetilde{\phi}_\ell - \widehat{\widetilde{\Phi}}_\ell|\!\| \le C_{\mathrm{sat}} \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\| + 2\frac{\lambda}{1-\lambda}(1 + C_{\mathrm{sat}}) \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\| &= \big(C_{\mathrm{sat}} + 2\frac{\lambda}{1-\lambda}(1 + C_{\mathrm{sat}})\big) \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\| \\ &= \widetilde{C}_{\mathrm{sat}} \|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\| \end{aligned}$$

und damit die Behauptung. □

## 4.4 $h - h/2$-Fehlerschätzer für gestörtes Galerkinverfahren

**Satz 4.8.** *Der Fehlerschätzer*

$$\widetilde{\eta}_\ell := \|\!|\widetilde{\Phi}_\ell - \widehat{\widetilde{\Phi}}_\ell|\!\| \tag{4.10}$$

*ist effizient für* $\|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\|$ *mit* $C_{\mathrm{eff}} = 1$*. Unter der Saturationsannahme mit* $C_{\mathrm{sat}} \in (0,1)$

$$\|\!|\phi - \widehat{\Phi}_\ell|\!\| \le C_{\mathrm{sat}} \|\!|\phi - \Phi_\ell|\!\|, \tag{4.11}$$

*und der Beschränktheit des Datenfehler für ein* $\lambda > 0$ *durch*

$$\|\!|\phi - \widetilde{\phi}_\ell|\!\| \le \lambda \|\!|\phi - \Phi_\ell|\!\|, \tag{4.12}$$

*ist* $\widetilde{\eta}_\ell$ *auch zuverlässig in Bezug auf* $\|\!|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell|\!\|$ *mit* $C_{\mathrm{rel}} = 1/\sqrt{1 - \widetilde{C}_{\mathrm{sat}}^2}$*. Die Konstante* $\widetilde{C}_{\mathrm{sat}}$ *stammt aus Satz 4.7.*

**Beweis:** Aufgrund der Galerkinorthogonalität für $\widetilde{\phi}_\ell - \widehat{\widetilde{\Phi}}_\ell$ darf der Satz von Pythagoras angewendet werden und man erhält

$$\left|\!\left|\!\left|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\right|\!\right|\!\right|^2 = \left|\!\left|\!\left|\widetilde{\phi}_\ell - \widehat{\widetilde{\Phi}}_\ell\right|\!\right|\!\right|^2 + \left|\!\left|\!\left|\widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell\right|\!\right|\!\right|^2 = \left|\!\left|\!\left|\widetilde{\phi}_\ell - \widehat{\widetilde{\Phi}}_\ell\right|\!\right|\!\right|^2 + \widetilde{\eta}_\ell^2$$

und somit die Effizienz von $\widetilde{\eta}_\ell$ mit $C_{\text{eff}} = 1$:

$$\widetilde{\eta}_\ell \leq \left|\!\left|\!\left|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\right|\!\right|\!\right|.$$

Wegen Satz 4.7 gilt $\left|\!\left|\!\left|\widetilde{\phi}_\ell - \widehat{\widetilde{\Phi}}_\ell\right|\!\right|\!\right| \leq \widetilde{C}_{\text{sat}}\left|\!\left|\!\left|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\right|\!\right|\!\right|$. Mit derselben Argumentation wie zuvor und dem Anwenden dieser Saturationsannahme folgt

$$\left|\!\left|\!\left|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\right|\!\right|\!\right|^2 = \left|\!\left|\!\left|\widetilde{\phi}_\ell - \widehat{\widetilde{\Phi}}_\ell\right|\!\right|\!\right|^2 + \left|\!\left|\!\left|\widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell\right|\!\right|\!\right|^2 \leq \widetilde{C}_{\text{sat}}^2\left|\!\left|\!\left|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\right|\!\right|\!\right|^2 + \widetilde{\eta}_\ell^2.$$

Durch Umformen erhält man die Zuverlässigkeit von $\widetilde{\eta}_\ell$ mit $C_{\text{rel}} = 1/\sqrt{1 - \widetilde{C}_{\text{sat}}^2}$

$$\left|\!\left|\!\left|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\right|\!\right|\!\right| \leq C_{\text{rel}}\widetilde{\eta}_\ell$$

und somit die Aussage des Satzes. □

Nun definieren wir einen lokalen Fehlerschätzer in der $L^2$-Norm mit denselben Eigenschaften wie $\widetilde{\eta}_\ell$.

**Definition 4.9.** *Sei $\Pi_\ell : L^2(\Gamma) \to \mathcal{P}^0(\mathcal{T}_\ell)$ die $L^2$-Projektion auf $\mathcal{P}^0(\mathcal{T}_\ell)$. Wir definieren den Fehlerschätzer*

$$\widetilde{\mu}_\ell := \|h_\ell^{1/2}(\widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell)\|_{L^2(\Gamma)}. \tag{4.13}$$

**Satz 4.10.** *Die Fehlerschätzer $\widetilde{\mu}_\ell$ und $\widetilde{\eta}_\ell$ sind äquivalent, d.h*

$$(\sqrt{2}C_{\text{inv}})^{-1}\widetilde{\mu}_\ell \leq \widetilde{\eta}_\ell \leq C_{\text{apx}}\widetilde{\mu}_\ell. \tag{4.14}$$

**Beweis:** Der Beweis läuft vollkommen analog zum Beweis von Satz 4.5. Alle Aussagen gelten statt für $\Phi_\ell$ und $\widehat{\Phi}_\ell$ ebenso für $\widetilde{\Phi}_\ell$ und $\widehat{\widetilde{\Phi}}_\ell$. □

**Korollar 4.11.** *Der Fehlerschätzer $\widetilde{\mu}_\ell$ ist effizient und unter der Bedingung $\left|\!\left|\!\left|\phi - \widetilde{\phi}_\ell\right|\!\right|\!\right| \leq \lambda\left|\!\left|\!\left|\widetilde{\phi}_\ell - \Phi_\ell\right|\!\right|\!\right|$ für ein $\lambda > 0$ und der Saturationsannahme $\left|\!\left|\!\left|\phi - \widehat{\Phi}_\ell\right|\!\right|\!\right| \leq C_{\text{sat}}\left|\!\left|\!\left|\phi - \Phi_\ell\right|\!\right|\!\right|$ zuverlässig mit Bezug auf $\left|\!\left|\!\left|\widetilde{\phi}_\ell - \widetilde{\Phi}_\ell\right|\!\right|\!\right|$.* □

## 4.5 Gesamtfehlerschätzer

Analog zu Satz 3.7 wird in diesem Abschnitt die Zuverlässigkeit des $h - h/2$-basierten Gesamtfehlerschätzers gezeigt.

**Satz 4.12.** *Der Gesamtfehlerschätzer $\omega_\ell := \left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ erfüllt unter der Saturationsannahme $\left|\!\left|\!\left|\phi - \widehat{\Phi}_\ell\right|\!\right|\!\right| \leq C_{\text{sat}}\left|\!\left|\!\left|\phi - \Phi_\ell\right|\!\right|\!\right|$ mit $C_{\text{sat}} \in (0,1)$ und der Voraussetzung $\left|\!\left|\!\left|\phi - \widetilde{\phi}_\ell\right|\!\right|\!\right| \leq \lambda\left|\!\left|\!\left|\phi - \Phi_\ell\right|\!\right|\!\right|$ für ein $\lambda > 0$ die Bedingung*

$$\left|\!\left|\!\left|\phi - \widetilde{\Phi}_\ell\right|\!\right|\!\right| \leq \overline{C}_{\text{rel}}\omega_\ell. \tag{4.15}$$

*D.h. $\omega_\ell$ ist zuverlässig in Bezug auf $\left|\!\left|\!\left|\phi - \widetilde{\Phi}_\ell\right|\!\right|\!\right|$. Die Zuverlässigkeitskonstante $\overline{C}_{\text{rel}}$ hängt nur von $\Gamma$ und $\kappa(\mathcal{T}_\ell)$ ab.*

**Beweis:** Aus $(a+b)^2 \leq 2a^2 + 2b^2$ folgt

$$\left|\!\left|\!\left| \phi - \widetilde{\Phi}_\ell \right|\!\right|\!\right|^2 \leq 2\left|\!\left|\!\left| \phi - \widetilde{\phi}_\ell \right|\!\right|\!\right|^2 + 2\left|\!\left|\!\left| \widetilde{\phi}_\ell - \widetilde{\Phi}_\ell \right|\!\right|\!\right|^2.$$

Mit Satz 3.6 und Korollar 4.11 gilt also

$$\left|\!\left|\!\left| \phi - \widetilde{\Phi}_\ell \right|\!\right|\!\right|^2 \leq 2C_{\mathrm{data}}^2 \zeta_\ell^2 + 2C_{\mathrm{rel}}^2 \widetilde{\mu}_\ell^{\,2} \leq 2\max\{C_{\mathrm{data}}^2, C_{\mathrm{rel}}^2\}\omega_\ell^2 = \overline{C}_{\mathrm{rel}}\omega_\ell^2.$$

□

## 4.6 Adaptiver Algorithmus

Zur Berechnung der $h - h/2$-Fehlerschätzer wird auch die Galerkinlösung $\widehat{\widetilde{\Phi}}_\ell$ benötigt. Deshalb muss Algorithmus 3.5 etwas verändert werden:

**Algorithmus 4.13.** *Input: Startnetz $\mathcal{T}_0$, Abbruchparameter $\varepsilon > 0$.*

*(1) Verfeinere das Netz $\mathcal{T}_\ell$ uniform und erhalte $\widehat{\mathcal{T}}_\ell$.*

*(2) Berechne diskrete Lösung $\widehat{\widetilde{\Phi}}_\ell$.*

*(3) Berechne diskrete Lösung $\widetilde{\Phi}_\ell$ durch Einführung von Nebenbedingungen.*

*(2) Berechne Verfeinerungsindikatoren $\omega_\ell(T) = \big(\widetilde{\mu}_\ell(T)^2 + \zeta_\ell(T)^2\big)^{1/2}$ für alle $T \in \mathcal{T}_\ell$.*

*(3) Stoppe, falls $\widetilde{\Phi}_\ell$ ausreichend genau, das heißt $\omega_\ell = \big(\sum_{T\in\mathcal{T}_\ell} \omega_\ell(T)^2\big)^{1/2} \leq \varepsilon$.*

*(4) Bilde Menge $\mathcal{M}_\ell$ der zur Verfeinerung markierten Elemente.*

*(5) Verfeinere mindestens die Elemente $T \in \mathcal{M}_\ell$ und erhalte $\mathcal{T}_{\ell+1}$.*

*(6) Erhöhe Zähler $\ell \to \ell + 1$ und gehe zu (1).*

*Output: Approximation $\widetilde{\Phi}_\ell$ von $\phi$.*

# Kapitel 5

# Die Symm'sche Integralgleichung in 2D

In diesem Kapitel werden die Galerkin-Randelementmethode auf die Symm'sche Integralgleichung in 2D angewendet, die vorkommenden Einfach und Doppelintegrale für beliebigen Polynomgrad der Ansatzfunktionen explizit ausgerechnet und die zugehörigen MATLAB-Codes angeführt.

Außerdem wird eine bis auf Quadratur exakte Berechnung der rechten Seite der Symm'schen Integralgleichung angegeben. Die in den Kapiteln 3 und 4 vorgestellten Fehlerschätzer werden berechnet und die MATLAB-Implementierung angeführt.

## 5.1 Markierungsstrategie

In Abschnitt 3 wurde die Dörflermarkierung (3.10) vorgestellt. Hier wird der zugehörige MATLAB-Code angegeben.

Um die Zuverlässigkeit des Residualschätzer zu zeigen, wurde Satz 3.2 mit einer Konstanten $\widetilde{C}_{\text{apx}}$ angewendet. In den Beweis der Äquivalenz der $h - h/2$-Fehlerschätzer — siehe Sätze 4.5 und 4.10 — geht Satz 4.2 mit einer Konstanten $C_{\text{inv}} < \infty$ ein. Die beiden Konstanten $\widetilde{C}_{\text{apx}}$ und $C_{\text{inv}}$ sind abhängig von $\kappa(\mathcal{T}_\ell)$.

Will man also einen zuverlässigen residualen Fehlerschätzer erhalten bzw. die Äquivalenz der $h - h/2$-Schätzer gewährleisten, muss die Markierungsstrategie rekursiv um Bedingung (3.11) erweitert werden, damit $\kappa(\mathcal{T}_\ell) \leq 2\kappa(\mathcal{T}_0)$ gilt. Auch für diese Erweiterung wird hier der MATLAB-Code angeführt.

### 5.1.1 Dörflermarkierung

```
nE = size(elements,1)
if theta > 0
   indicator = rho_T.^2 + zeta_T.^2;
   [sort_indicator,buch] = sort(indicator,'descend');
   i=1;
   while theta*sum(indicator) > sum(sort_indicator(1:i))
      i=i+1;
```

```
      end
      marked = buch(1:i);
   else
      marked = 1:nE;
   end
```

### 5.1.2 Markierung für $\kappa(\mathcal{T}_\ell)$

```
function marked = kappa_marking(elements,coordinates,marked,kappa)

   K = size(coordinates,1);
   N = size(elements,1);

   % find elements belonging to nodes
   elements4node = zeros(K,2);
   for j = 1:N
      elements4node(elements(j,1),2) = j;
      elements4node(elements(j,2),1) = j;
   end

   % find neighbors of each elements
   neighbors = zeros(N,2);
   for j = 1:N
      neighbors(j,:) = [elements4node(elements(j,1),1) ...
                        elements4node(elements(j,2),2)];
   end

   % compute size of each element
   for j = 1:N
      h(j) = norm( [1 -1] * coordinates(elements(j,:),:) );
   end

   % for T,T' neighbors do some extra markings
   % if h_{\ell+1}(T) >= 2*kappa*h_{\ell+1}(T')
   % for an element T' which is already marked
   % and the neighbor T who is not marked
   flag = zeros(1,N);
   flag(marked) = 1;
   changing = 1;

   while changing
      changing = 0;
      marked_new = find(flag);
      for k = 1:length(marked_new)
         j = marked_new(k);
         nleft = neighbors(j,1);
         nright = neighbors(j,2);
```

```
        if h(nleft)>=2*kappa*0.5*h(j) && flag(nleft) ~= 1
           flag(nleft)=1;
           changing = 1;
        end
        if h(nright)>=2*kappa*0.5*h(j) && flag(nright) ~= 1
           flag(nright) = 1;
           changing = 1;
        end
     end
  end

end
```

Es ist durch geschicktes Vorsortieren der Elemente nach ihrer Größe möglich, die Bedingungsschleife zu umgehen. Dies zeigt der folgende Algorithmus:

```
function marked = better_kappa_marking(elements,coordinates,marked,kappa)

  K = size(coordinates,1);
  N = size(elements,1);

  % find elements belonging to nodes
  elements4node = zeros(K,2);
  for j = 1:N
     elements4node(elements(j,1),2) = j;
     elements4node(elements(j,2),1) = j;
  end

  % find neighbors of each elements
  neighbors = zeros(N,2);
  for j = 1:N
     neighbors(j,:) = [elements4node(elements(j,1),1) ...
                       elements4node(elements(j,2),2)];
  end

  % compute size of each element
  for j = 1:N
     h(j) = norm( [1 -1] * coordinates(elements(j,:),:) );
  end

  % for T,T' neighbors do some extra markings
  % if h_{\ell+1}(T) >= 2*kappa*h_{\ell+1}(T')
  % for an element T' which is already marked
  % and the neighbor T who is not marked
  flag = zeros(1,N);
  flag(marked) = 1;
  changing = 1;
```

```
    [tmp,idx] = sort(h);

    for j = idx
       if flag(j)
          n = neighbors(j,:);
          n = n(find(n));
          n = n(find( h(n)/h(j) >= kappa ));
          flag(n) = 1;
       end
    end

end
```

## 5.2 einfache Integrale

Im Zuge der Berechnung des Einfach- und Doppelschichtpotentiales treten eindimensionale Integrale auf, für die folgende Rekursionsformeln gelten:

**Lemma 5.1.** *Seien $p, q \in \mathbb{R}^2$ mit $p \neq 0$, $p \neq \pm q$ und $\Delta := 4(|p|^2|q|^2 - \langle p, q\rangle^2)$. Für $k \in \mathbb{N}_0$ definieren wir*

$$G(k,p,q) := \int_{-1}^{1} t^k \, \frac{1}{|p|^2 t^2 + 2\langle p,q\rangle t + |q|^2} \, dt. \tag{5.1}$$

*Für $k \geq 2$ gilt dann folgende Rekursionsformel*

$$G(k,p,q) = \frac{1}{|p|^2}\Big(\frac{1-(-1)^{k-1}}{k-1} - 2\langle p,q\rangle G(k-1,p,q) - |q|^2 G(k-2,p,q)\Big) \tag{5.2}$$

*mit Abbruchbedingungen*

$$G(1,p,q) = \frac{1}{|p|^2}\Big(\log|p+q| - \log|p-q| - \langle p,q\rangle G(0,p,q)\Big) \tag{5.3}$$

*und*

$$\begin{aligned} &\textit{für } \Delta = 0: \\ &\quad G(0,p,q) = \frac{2}{|q|^2 - |p|^2} \\ &\textit{für } \Delta > 0: \\ &\quad G(0,p,q) = \frac{2}{\sqrt{\Delta}} \begin{cases} \arctan \frac{\sqrt{\Delta}}{|q|^2-|p|^2} & \textit{für } |p| < |q| \\ \frac{\pi}{2} & \textit{für } |p| = |q| \\ \arctan \frac{\sqrt{\Delta}}{|q|^2-|p|^2} + \pi & \textit{für } |p| > |q|. \end{cases} \end{aligned} \tag{5.4}$$

**Beweis:** Dieses Integral ist ein Spezialfall des Integrals in Maischak [14, Definition 2.1], für dessen Fall $n = -1$. Setzt man nun $a = |p|^2$, $b = 2\langle p,q\rangle$ und $c = |q|^2$, so erhält man die Rekursionsformel aus Maischak [14, Lemma 2.1]. □

**Lemma 5.2.** *Seien $p, q \in \mathbb{R}^2$ mit $p \neq 0$, $p \neq \pm q$ und $\Delta := 4(|p|^2|q|^2 - \langle p, q\rangle^2)$. Für $k \in \mathbb{N}_0$ definieren wir*

$$L(k,p,q) := \int_{-1}^{1} t^k \log\Big(|p|^2t^2 + 2\langle p,q\rangle t + |q|^2\Big)\,dt. \tag{5.5}$$

*Für $k \geq 2$ gilt dann folgende Rekursionsformel*

$$\begin{aligned} L(k,p,q) = \frac{1}{|p|^2(k+1)}\Big(&|p+q|^2\log|p+q|^2 + (-1)^k|p-q|^2\log|p-q|^2 - 2|p|^2\frac{1-(-1)^{k+1}}{k+1} \\ &- 2\langle p,q\rangle\frac{1-(-1)^k}{k} - 2\langle p,q\rangle kL(k-1,p,q) - |q|^2(k-1)L(k-2,p,q)\Big) \end{aligned} \tag{5.6}$$

*mit Abbruchbedingungen*

$$L(1,p,q) = \frac{1}{|p|^2}\Big(|p+q|^2\log|p+q| - |p-q|^2\log|p-q| - 2\langle p,q\rangle - \langle p,q\rangle L(0,p,q)\Big) \tag{5.7}$$

*und*

$$\begin{aligned} &\textit{für } \Delta = 0: \\ &L(0,p,q) = 2\Big[\Big(1+\frac{\langle p,q\rangle}{|p|^2}\Big)\log\Big(\frac{\langle p,q\rangle}{|p|^2}+|p|\Big) + \Big(1-\frac{\langle p,q\rangle}{|p|^2}\Big)\log\Big(\frac{\langle p,q\rangle}{|p|^2}-|p|\Big) - 2\Big] \\ &\textit{für } \Delta > 0: \\ &L(0,p,q) = 2\Big[\Big(1+\frac{\langle p,q\rangle}{|p|^2}\Big)\log|p+q| + \Big(1-\frac{\langle p,q\rangle}{|p|^2}\Big)\log|p-q| - 2 + \Big(|q|^2 - \frac{\langle p,q\rangle^2}{|p|^2}\Big)G(0,p,q)\Big]. \end{aligned} \tag{5.8}$$

**Beweis:** Dieses Integral ist ein Spezialfall des Integrals in Maischak [14, Definition 2.2], für dessen Fall $n = 0$. Setzt man nun $a = |p|^2$, $b = 2\langle p,q\rangle$ und $c = |q|^2$, so erhält man die Rekursionsformel aus Maischak [14, Lemma 2.2]. □

## 5.3 Doppelintegrale

In diesem Abschnitt wollen wir Rekursionsformeln für die im Einfach- und Doppelschichtpotential auftretenden zweidimensionalen Integrale angeben:

**Lemma 5.3.** *Seien $u, v, w \in \mathbb{R}^2$ mit $u \neq 0, v \neq 0$ und $u \parallel v$, d.h. $u = \mu v$ für ein $\mu \in \mathbb{R}\backslash\{0\}$. Für eine integrierbare Funktion $f : \mathbb{R}^2 \to \mathbb{R}$ definieren wir für $k, \ell \in \mathbb{N}_0$*

$$\mathcal{I}(k,\ell,u,v,w) := \int_{-1}^{1}\int_{-1}^{1} s^k t^\ell\, f(su+tv+w)\,dt\,ds. \tag{5.9}$$

*Für $k, \ell \geq 0$ gilt dann folgende Rekursionsformel*

$$\begin{aligned} (\ell+1)\mathcal{I}(k,\ell,u,v,w) = &\int_{-1}^{1} s^k\, f(su+w+v)\,ds - (-1)^{\ell+1}\int_{-1}^{1} s^k\, f(su+w-v)\,ds \\ &- \mu\int_{-1}^{1} t^{\ell+1}\, f(tv+w+u)\,dt + (-1)^k\mu\int_{-1}^{1} t^{\ell+1}\, f(tv+w-u)\,dt \\ &+ k\mu\mathcal{I}(k-1,\ell+1,u,v,w), \end{aligned} \tag{5.10}$$

*mit Abbruchbedingung $k = 0$, wobei der letzte Summand verschwindet.*

**Beweis:** Diese Formel erhält man sofort aus MAISCHAK [14, Lemma 3.1] durch Einsetzen der Integrationsgrenzen $-1$ und 1. □

**Lemma 5.4.** *Seien $u, v, w \in \mathbb{R}^2$ mit $u \neq 0, v \neq 0$ und $u \nparallel v$, d.h. $w = \mu_1 u + \mu_2 v$, $\mu_1, \mu_2 \in \mathbb{R}$. Für eine integrierbare Funktion $f : \mathbb{R}^2 \to \mathbb{R}$, $p > -2$ und eine Funktion $h : \mathbb{R}^2 \to \mathbb{R}$ gelte*

$$\big\langle x, \nabla f(x) \big\rangle = h(x) + p f(x). \tag{5.11}$$

*Wir definieren für $k, \ell \in \mathbb{N}_0$*

$$\mathcal{I}(k, \ell, u, v, w) := \int_{-1}^{1} \int_{-1}^{1} s^k t^\ell \, f(su + tv + w) \, dt \, ds. \tag{5.12}$$

*Für $k, \ell \geq 0$ gilt dann folgende Rekursionsformel*

$$\begin{aligned}
&(k + \ell + p + 2)\, \mathcal{I}(k, \ell, u, v, w) \\
&\quad = -\int_{-1}^{1} \int_{-1}^{1} s^k t^\ell \, h(su + tv + w) \, dt \, ds + (\mu_1 + 1) \int_{-1}^{1} t^\ell \, f(tv + w + u) \, dt \\
&\quad - (-1)^k (\mu_1 - 1) \int_{-1}^{1} t^\ell \, f(tv + w - u) \, dt + (\mu_2 + 1) \int_{-1}^{1} s^k \, f(su + w + v) \, ds \\
&\quad - (-1)^\ell (\mu_2 - 1) \int_{-1}^{1} s^k \, f(su + w - v) \, ds - k\mu_1 \mathcal{I}(k - 1, \ell, u, v, w) - \ell \mu_2 \mathcal{I}(k, \ell - 1, u, v, w),
\end{aligned} \tag{5.13}$$

*mit Abbruchbedingung $k = 0$, $\ell = 0$, da die letzten beiden Summanden verschwinden.*

**Beweis:** Durch Einsetzen der Integrationsgrenzen $-1$ und 1 in MAISCHAK [14, Lemma 3.1] ergibt sich die Rekursionsformel. □

**Lemma 5.5.** *Sei $k, \ell \in \mathbb{N}_0$, $k, \ell \geq 0$. Dann gilt*

$$\widetilde{\mathcal{I}}(k, \ell) := \int_{-1}^{1} \int_{-1}^{1} s^k t^\ell \, dt \, ds = \frac{1 + (-1)^\ell}{\ell + 1} \cdot \frac{1 + (-1)^k}{k + 1}. \tag{5.14}$$

**Beweis:** Dieses Integral kann leicht mittels der Ableitungsregel für Potenzen berechnet werden:

$$\begin{aligned}
\int_{-1}^{1} \int_{-1}^{1} s^k t^\ell \, dt \, ds = \int_{-1}^{1} s^k \left[ \frac{t^{\ell+1}}{\ell + 1} \right]_{t=-1}^{t=1} ds = \frac{1 - (-1)^{\ell+1}}{\ell + 1} \cdot \int_{-1}^{1} s^k \, ds = \frac{1 + (-1)^\ell}{\ell + 1} \cdot \left[ \frac{s^{k+1}}{k + 1} \right]_{s=-1}^{s=1} \\
= \frac{1 + (-1)^\ell}{\ell + 1} \cdot \frac{1 + (-1)^k}{k + 1}
\end{aligned}$$

□

## 5.4 Steifigkeitsmatrix und rechte Seite

Sei $\Gamma := \partial\Omega$ der Rand eines beschränkten Lipschitz-Gebietes $\Omega \subset \mathbb{R}^2$. Die Randelementmethode führt, wie in Kapitel 2. beschrieben, zu Integraloperatoren, nämlich dem Einfachschichtpotential

$$V\phi(x) := -\frac{1}{2\pi} \int_\Gamma \phi(y) \log |x - y| \, ds_y$$

und dem Doppelschichtpotential

$$K\phi(x) := -\frac{1}{2\pi}\int_\Gamma \phi(y)\frac{\big\langle (y-x),\nu(y)\big\rangle}{|x-y|^2}\,ds_y.$$

$\nu(y)$ bezeichnet hierbei den äußeren Normalenvektor von $y \in \Omega$.
Im Folgenden nehmen wir an, dass $\Gamma$ ein Polygonzug ist, d.h. $\Gamma$ ist die Vereinigung endlich vieler affiner Teilstücke $[a,b] := \mathrm{conv}\{a,b\} \subset \mathbb{R}^2$ mit $a,b \in \mathbb{R}^2$. Die Berechnung von $V\phi$ und $K\phi$ reduziert sich also auf die Berechnung von

$$V_{[a,b]}\phi(x) := -\frac{1}{2\pi}\int_{[a,b]} \phi(y)\log|x-y|\,ds_y \tag{5.15}$$

und

$$K_{[a,b]}\phi(x) := -\frac{1}{2\pi}\int_{[a,b]} \phi(y)\frac{\big\langle (y-x),\nu(y)\big\rangle}{|x-y|^2}\,ds_y. \tag{5.16}$$

Man beachte, dass der Normalenvektor $\nu(y) = \nu_{[a,b]}$ konstant ist auf $[a,b]$. Angenommen die mathematische Orientierung von $\Gamma$ folgt dem Vektor $b-a$, so berechnet sich der Normalenvektor als

$$\nu_{[a,b]} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\frac{b-a}{|b-a|} = \frac{1}{|b-a|}\begin{pmatrix} b_2-a_2 \\ a_1-b_1 \end{pmatrix}.$$

Wir betrachten für $T=[a,b]$ die Parametrisierung

$$\gamma_T : [-1,1] \to [a,b], \quad t \mapsto x = \tfrac{1}{2}\big(a+b+t(b-a)\big). \tag{5.17}$$

Für das Oberflächenintegral gilt dann

$$\int_{[a,b]} \dots\, ds_x = \frac{|b-a|}{2}\int_{-1}^{1} \dots\, dt.$$

Üblicherweise wird die Funktion $\phi$ durch ein stückweises Polynom $\phi_h$ in der Bogenlänge vom Grad N approximiert. Für jedes Randstück ist demnach $\phi_h \circ \gamma_{[a,b]}$ ein Polynom auf $[-1,1]$,

$$\phi_h \circ \gamma(t) = \sum_{\ell=0}^{N} \lambda_\ell t^\ell \quad \text{für } t \in [-1,1].$$

$V\phi$ und $K\phi$ lassen sich also darstellen in der Form

$$V_{[a,b]}\phi_h(x) := -\frac{1}{2\pi}\frac{|b-a|}{2}\sum_{\ell=0}^{N}\lambda_\ell\int_{-1}^{1} t^\ell \log|x-\gamma_{[a,b]}(t)|\,dt,$$

$$K_{[a,b]}\phi_h(x) := -\frac{1}{2\pi}\frac{|b-a|}{2}\sum_{\ell=0}^{N}\lambda_\ell\int_{-1}^{1} t^\ell \frac{\big\langle \gamma_{[a,b]}(t)-x,\nu_{[a,b]}\big\rangle}{|x-\gamma_{[a,b]}(t)|^2}\,dt.$$

Bei der Galerkin-Randelementmethode zur Symm'schen Integralgleichung müssen $L^2$-Skalarprodukte berechnet werden

$$\big\langle V\phi_h,\psi_h\big\rangle_\Gamma = \int_\Gamma V\phi_h(x)\psi_h(x)\,ds_x,$$

$$\big\langle (K+\frac{1}{2})\phi_h,\psi_h\big\rangle_\Gamma = \big\langle K\phi_h,\psi_h\big\rangle_\Gamma + \frac{1}{2}\big\langle \phi_h,\psi_h\big\rangle_\Gamma = \int_\Gamma K\phi_h(x)\psi_h(x)\,ds_x + \frac{1}{2}\int_\Gamma \phi_h(x)\psi_h(x)\,ds_x,$$

wobei $\phi_h$ und $\psi_h$ stückweise polynomial sind.
Insgesamt muss also für alle Intervalle $[a,b] \subseteq \Gamma$, $[\mathfrak{a},\mathfrak{b}] \subseteq \Gamma$ für das Einfachschichtpotential

$$\langle V\phi_h, \psi_h\rangle_\Gamma = -\frac{1}{2\pi}\frac{|b-a|}{2}\frac{|\mathfrak{b}-\mathfrak{a}|}{2}\sum_{\ell=0}^{N}\sum_{k=0}^{M}\lambda_\ell\lambda_k\int_{-1}^{1}\int_{-1}^{1} s^k t^\ell \log|\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)-\gamma_{[a,b]}(t)|\,dt\,ds$$

und für das Doppelschichtpotential

$$\begin{aligned}\Big\langle\Big(K+\frac{1}{2}\Big)\phi_h,\psi_h\Big\rangle_\Gamma &= \frac{|\mathfrak{b}-\mathfrak{a}|}{2}\sum_{\ell=0}^{N}\sum_{k=0}^{M}\lambda_\ell\lambda_k\Bigg[\frac{1}{2}\int_{-1}^{1}s^{k+\ell}\,ds\\ &\quad -\frac{1}{2\pi}\frac{|b-a|}{2}\int_{-1}^{1}\int_{-1}^{1}s^k t^\ell\frac{\big\langle\gamma_{[a,b]}(t)-\gamma_{[\mathfrak{a},\mathfrak{b}]}(s),\nu_{[a,b]}\big\rangle}{|\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)-\gamma_{[a,b]}(t)|^2}\,dt\,ds\Bigg]\end{aligned}$$

berechnet werden, wobei $\operatorname{supp}\phi_h \subseteq [a,b]$ und $\operatorname{supp}\psi_h \subseteq [\mathfrak{a},\mathfrak{b}]$ gilt.
Im Folgenden wollen wir Formeln zur Bestimmung der Einträge der entstehenden Einfach- und Doppelschichtpotentialmatrizen angeben.

### 5.4.1 Einfachschichtpotential

Für alle Intervalle $[a,b] \subseteq \Gamma$, $[\mathfrak{a},\mathfrak{b}] \subseteq \Gamma$ ist

$$\langle V\phi_h, \psi_h\rangle_\Gamma = -\frac{1}{2\pi}\frac{|b-a|}{2}\frac{|\mathfrak{b}-\mathfrak{a}|}{2}\sum_{\ell=0}^{N}\sum_{k=0}^{M}\lambda_\ell\lambda_k\int_{-1}^{1}\int_{-1}^{1} s^k t^\ell \log\Big|\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)-\gamma_{[a,b]}(t)\Big|\,dt\,ds$$

mit $\operatorname{supp}\phi_h \subseteq [a,b]$ und $\operatorname{supp}\psi_h \subseteq [\mathfrak{a},\mathfrak{b}]$ zu berechnen. Wir betrachten zunächst einen Teil des Integranden und setzen die Parametrisierung aus (5.17) ein:

$$\begin{aligned}\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)-\gamma_{[a,b]}(t) &= \frac{1}{2}\Big(\mathfrak{a}+\mathfrak{b}+s(\mathfrak{b}-\mathfrak{a})-a-b-t(b-a)\Big)\\ &= \frac{1}{2}\Big((\mathfrak{b}-\mathfrak{a})\cdot s+(a-b)\cdot t+(\mathfrak{a}+\mathfrak{b}-a-b)\Big).\end{aligned}$$

Wir setzen nun

$$\begin{aligned}u &:= \mathfrak{b}-\mathfrak{a}\\ v &:= a-b\\ w &:= \mathfrak{a}+\mathfrak{b}-a-b\end{aligned}$$

und erhalten so für das Integral

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1} s^k t^\ell \log\Big|\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)-\gamma_{[a,b]}(t)\Big|\,dt\,ds\\ &\quad= \int_{-1}^{1}\int_{-1}^{1} s^k t^\ell \log\Big|\frac{1}{2}\Big(su+tv+w\Big)\Big|\,dt\,ds\\ &\quad= \log(1/2)\int_{-1}^{1}\int_{-1}^{1} s^k t^\ell\,dt\,ds+\int_{-1}^{1}\int_{-1}^{1} s^k t^\ell\log|su+tv+w|\,dt\,ds.\end{aligned}$$

Wendet man nun Lemma 5.3 und 5.1 auf das zweite Integral an, so erhält man:

**Lemma 5.6.** *Seien $u, v, w \in \mathbb{R}^2$, $u \neq 0$ und $u \parallel v$, d.h. $u = \mu v$ für ein $\mu \in \mathbb{R}\backslash\{0\}$. Wir definieren für $k, \ell \in \mathbb{N}_0$*

$$J(k, \ell, u, v, w) := \int_{-1}^{1} \int_{-1}^{1} s^k t^\ell \log |su + tv + w| \, dt \, ds. \tag{5.18}$$

*Für $k, \ell \geq 0$ gilt dann folgende Rekursionsformel*

$$\begin{aligned}(\ell + 1) J(k, \ell, u, v, w) = \frac{1}{2} L(k, u, w + v) + \frac{1}{2}(-1)^\ell L(k, u, w - v) - \frac{\mu}{2} L(\ell + 1, v, w + u) \\ + \frac{\mu}{2}(-1)^k L(\ell + 1, v, w - u) + k\mu J(k - 1, \ell + 1, u, v, w)\end{aligned} \tag{5.19}$$

*mit Abbruchbedingung $k = 0$, da der letzte Summand verschwindet:*

$$\begin{aligned}J(0, \ell, u, v, w) = \frac{1}{2(\ell + 1)} \Big[ L(0, u, w + v) + (-1)^\ell L(0, u, w - v) \\ - \mu L(\ell + 1, v, w + u) + \mu L(\ell + 1, v, w - u) \Big]\end{aligned} \tag{5.20}$$

□

Die Rekursionsformel aus Lemma 5.6 lässt sich auch induktiv umschreiben, man kann $J(k, \ell, u, v, w)$ also ohne zusätzliche Funktionsaufrufe berechnen:
Zuerst bestimmt man $J(0, \ell + k, u, v, w)$ und dann $J(j, \ell + k - j, u, v, w)$ für $j = 1, \ldots, k$. Insgesamt erhält man daher das gewünschte Integral in $k + 1$ Schritten:

$$J_{0,\ell+k} \to J_{1,\ell+k-1} \to \ldots \to J_{k,\ell}$$

Auch die benötigten einfachen Integrale, d.h die Funktionen $L(j, p, q)$, kann man induktiv berechnen. Um $J(k, \ell, u, v, w)$ zu erhalten, müssen die Funktionswerte von $L(j, u, w + v)$ und $L(j, u, w - v)$ für $j = 0, \ldots, k$ sowie $L(j, v, w + u)$ und $L(j, v, w - u)$ für $j = \ell + 1, \ldots, \ell + k + 1$ bekannt sein.
Man bestimmt also in $k+1$ Schritten $L(j, u, w+v)$ und $L(j, u, w-v)$. In $\ell+k+2$ Schritten errechnet man $L(j, v, w + u)$ und $L(j, v, w - u)$.

**Algorithmus 5.7.** *u,v parallel*

```
function output = J1(k,ell,u,v,w)

  compute mu;

  L_u_wpv = zeros(k+1,1);
  L_u_wmv = zeros(k+1,1);
  L_v_wpu = zeros(k+1,1);
  L_v_wmu = zeros(k+1,1);
  for j = 0:k
      L_u_wpv(j+1) = computation of L(j,u,w+v);
      L_u_wmv(j+1) = computation of L(j,u,w-v);
  end
  for j = 0:ell
      L_v_wpu(1) = computation of L(j,v,w+u) by updating L_v_wpu(1);
```

```
    L_v_wmu(1) = computation of L(j,v,w-u) by updating L_v_wmu(1);
end
for j = (ell+1):(ell+k+1)
    L_v_wpu(j-ell) = computation of L(j,v,w+u);
    L_v_wmu(j-ell) = computation of L(j,v,w-u);
end

output = ( L_u_wpv(1) + (-1)^(ell+k)*L_u_wmv(1) - mu*L_v_wpu(k+1) ...
           + mu*L_v_wmu(k+1) ) *0.5/(ell+k+1);
for j = 1:k
   output = ( 0.5*L_u_wmv(j) + 0.5*(-1)^(ell+k-j)*L_u_wmv(j) ...
              - 0.5*mu*L_v_wpu(k+1-j) + 0.5*mu*(-1)^j*L_v_wmu(k+1-j) ...
              + j*mu*output ) / (ell+k+1-j);
end
```

**Lemma 5.8.** *Seien $u, v, w \in \mathbb{R}^2$ mit $u \neq 0, v \neq 0$ und $u \nparallel v$, d.h. $w = \mu_1 u + \mu_2 v$ mit $\mu_1, \mu_2 \in \mathbb{R}$. Wir definieren für $k, \ell \in \mathbb{N}_0$*

$$J(k,\ell,u,v,w) := \int_{-1}^{1}\int_{-1}^{1} s^k t^\ell \log|su+tv+w|\, dt\, ds. \tag{5.21}$$

*Für $k, \ell \geq 0$ gilt dann folgende Rekursionsformel*

$$\begin{aligned}(k+\ell+2)J(k,\ell,u,v,w) = &-\frac{1+(-1)^\ell}{\ell+1}\cdot\frac{1+(-1)^k}{k+1}\\ &+\frac{\mu_1+1}{2}L(\ell,v,w+u)-(-1)^k\frac{\mu_1-1}{2}L(\ell,v,w-u)\\ &+\frac{\mu_2+1}{2}L(k,u,w+v)-(-1)^\ell\frac{\mu_2-1}{2}L(k,u,w-v)\\ &-k\mu_1 J(k-1,\ell,u,v,w)-\ell\mu_2 J(k,\ell-1,u,v,w),\end{aligned} \tag{5.22}$$

*wobei für $k = 0$ der vorletzte und für $\ell = 0$ der letzte Summand entfällt. Die Rekursion bricht ab, sobald $k = \ell = 0$ gilt:*

$$\begin{aligned}J(0,0,u,v,w) = \frac{1}{4}\Big[&-8+(\mu_1+1)L(0,v,w+u)-(\mu_1-1)L(0,v,w-u)\\ &+(\mu_2+1)L(0,u,w+v)-(\mu_2-1)L(0,u,w-v)\Big]\end{aligned} \tag{5.23}$$

**Beweis:** Die Funktion $f : \mathbb{R}^2 \to \mathbb{R}$,

$$f(x) = \log|x|$$

erfüllt die Bedingung $\langle x, \nabla f(x)\rangle = h(x) + pf(x)$ aus Lemma 5.4 mit $h(x) \equiv 1$ und $p = 0$:
Nach Kettenregel gilt

$$\begin{aligned}\frac{\partial f(x)}{\partial x_1} &= \frac{1}{2|x|}\cdot 2x_1 = \frac{x_1}{|x|},\\ \frac{\partial f(x)}{\partial x_2} &= \frac{1}{2|x|}\cdot 2x_2 = \frac{x_2}{|x|}.\end{aligned}$$

Es folgt

$$\langle x, \nabla f(x)\rangle = \frac{x_1^2+x_2^2}{|x|} = 1 = h(x) + pf(x)$$

Es gilt also $h(x) \equiv 1$ und $p = 0$. Wendet man nun Lemma 5.4 und 5.1 auf das Integral an, so erhält man die obige Formel. □

Auch die Rekursionsformel aus Lemma 5.8 lässt sich induktiv umschreiben. Man muss hierzu allerdings mehr Werte als im Fall $u \parallel v$ speichern - das Berechnungsschema entspricht hierbei einer Matrix und nicht mehr einem Vektor.
Man bestimmt zuerst $J(0,0,u,v,w)$ und dann $J(0,j,u,v,w)$ für $j = 1,\ldots,\ell$ - das entspricht der ersten Zeile des Schemas. Aus den $\ell+1$ nun erhaltenen Werten errechnet man $J(i,j,u,v,w)$ für $j = 1,\ldots,k$ und $i = 1,\ldots,\ell$,man geht also zeilenweise vor.
Insgesamt erhält man das gewünschte Integral in $(k+1)(\ell+1)$ Schritten:

$$\begin{array}{ccccccc}
J_{0,0} & \to & J_{0,1} & \to & J_{0,2} & \cdots & J_{0,\ell} \\
\downarrow & & \downarrow & & \downarrow & & \downarrow \\
J_{1,0} & \to & J_{1,1} & \to & J_{1,2} & \cdots & J_{1,\ell} \\
\vdots & & \vdots & & \vdots & & \vdots \\
J_{k-1,0} & \to & J_{k-1,1} & \to & J_{k-1,2} & \cdots & J_{k-1,\ell} \\
\downarrow & & \downarrow & & \downarrow & & \downarrow \\
J_{k,0} & \to & J_{k,1} & \to & J_{k,2} & \cdots & J_{k,\ell}
\end{array}$$

Auch die benötigten einfachen Integrale, d.h die Funktionen $L(i,p,q)$, kann man wiederum induktiv berechnen. Um $J(k,\ell,u,v,w)$ zu erhalten, müssen die Funktionswerte von $L(i,u,w+v)$ und $L(i,u,w-v)$ für $i = 0,\ldots,k$ und $L(i,v,w+u)$ und $L(i,v,w-u)$ für $i = 0,\ldots,\ell$ bekannt sein.
Man bestimmt also in $k+1$ Schritten $L(i,u,w+v)$ und $L(i,u,w-v)$ und in $\ell+1$ Schritten $L(i,v,w+u)$ und $L(i,v,w-u)$.

**Algorithmus 5.9.** *u,v nicht parallel*

```
function output = J2(k,ell,u,v,w)

  compute mu1,mu2;

  L_u_wpv = zeros(k+1,1);
  L_u_wmv = zeros(k+1,1);
  L_v_wpu = zeros(ell+1,1);
  L_v_wmu = zeros(ell+1,1);
  for j = 0:k
      L_u_wpv(j+1) = computation of L(j,u,w+v);
      L_u_wmv(j+1) = computation of L(j,u,w-v);
  end
  for j = 0:ell
      L_v_wpu(j+1) = computation of L(j,v,w+u);
      L_v_wmu(j+1) = computation of L(j,v,w-u);
  end

  output = zeros(ell+1,1);
  output(1) =  ( -8 + (mu1+1)*L_v_wpu(1) - (mu1-1)*L_v_wmu(1) ...
                + (mu2+1)*L_u_wpv(1) - (mu2-1)*L_u_wmv(1) ) *0.25;
  for j = 1:ell
     output(j+1) = ( 2*(1+(-1)^j)/(j+1) + 0.5*(mu1+1)*L_v_wpu(j+1) ...
```

```
                    - 0.5*(mu1-1)*L_v_wmu(j+1) + 0.5*(mu2+1)*L_u_wpv(1) ...
                    - 0.5*(-1)^j*(mu2-1)*L_u_wmv(1) - j*mu2*output(j) ) / (ell+2);
end
for i = 1:k
   output(1) = ( 2*(1+(-1)^k)/(k+1) + 0.5*(mu1+1)*L_v_wpu(0) ...
                 - 0.5*(-1)^k*(mu1-1)*L_v_wmu(0) + 0.5*(mu2+1)*L_u_wpv(i) ...
                 - 0.5*(mu2-1)*L_u_wmv(i) - k*mu1*output(j) ) / (k+2);
   for j = 1:ell
      output(j+1) = ( (1+(-1)^ell)*(1+(-1)^k)/((ell+1)*(k+1)) ...
                    + 0.5*(mu1+1)*L_v_wpu(j) - 0.5*(-1)^k*(mu1-1)*L_v_wmu(j) ...
                    + 0.5*(mu2+1)*L_u_wpv(i) - 0.5*(-1)^ell*(mu2-1)*L_u_wmv(i) ...
                    - k*mu1*output(j) - ell*mu2*output(j+1) ) / (k+ell+2);
   end
end
output = output(ell+1);
```

Unter den Voraussetzungen von Lemma 5.8, d.h. $u \nparallel v$, und induktiver Berechnung aller Funktionen ist besondere Vorsicht geboten, falls sich die beiden Intervalle $[a,b]$ und $[\mathfrak{a},\mathfrak{b}]$ berühren, d.h entweder $a = \mathfrak{b}$ oder $b = \mathfrak{a}$ gilt. Unter diesen Gegebenheiten treten Probleme bei der numerischen Berechnung von $L(k,p,q)$ auf, da $p+q=0$ oder $p-q=0$ gilt und $\log|p+q|$ bzw. $\log|p-q|$ errechnet werden sollte. Betrachtet man jedoch die Formel für $J(k,ell,u,v,w)$ aus Lemma 5.8 so erkennt man, dass die kritischen Terme wegfallen:

**Lemma 5.10.** *Seien $u,v,w \in \mathbb{R}^2$ mit $u \neq 0, v \neq 0$, für die gilt $u \nparallel v$.*

*(i) Ist $b = \mathfrak{a}$, so gilt $w = u - v$, d.h. $\mu_1 = 1$ und $\mu_2 = -1$, und mit $k,\ell \geq 0$ für das in Lemma 5.8 definierte Integral die Rekursionsformel*

$$\begin{aligned}(k+\ell+2)J(k,\ell,u,v,w) = &-\frac{1-(-1)^{\ell+1}}{\ell+1}\cdot\frac{1-(-1)^{k+1}}{k+1} + L(\ell,v,w+u) + (-1)^{\ell}L(k,u,w-v)\\ &- kJ(k-1,\ell,u,v,w) + \ell J(k,\ell-1,u,v,w),\end{aligned} \tag{5.24}$$

*wobei für $k=0$ der vorletzte und für $\ell = 0$ der letzte Summand entfällt. Die Rekursion bricht ab, sobald $k = \ell = 0$ gilt:*

$$J(0,0,u,v,w) = \frac{1}{2}\Big[-4 + L(0,v,w+u) + L(0,u,w-v)\Big] \tag{5.25}$$

*(ii) Ist $a = \mathfrak{b}$, so gilt $w = v - u$, d.h. $\mu_1 = -1$ und $\mu_2 = 1$, und mit $k,\ell \geq 0$ für das in Lemma 5.8 definierte Integral die Rekursionsformel*

$$\begin{aligned}(k+\ell+2)J(k,\ell,u,v,w) = &-\frac{1-(-1)^{\ell+1}}{\ell+1}\cdot\frac{1-(-1)^{k+1}}{k+1} + (-1)^{k}L(\ell,v,w-u) + L(k,u,w+v)\\ &+ kJ(k-1,\ell,u,v,w) - \ell J(k,\ell-1,u,v,w),\end{aligned} \tag{5.26}$$

*wobei für $k=0$ der vorletzte und für $\ell = 0$ der letzte Summand entfällt. Die Rekursion bricht ab, sobald $k = \ell = 0$ gilt:*

$$J(0,0,u,v,w) = \frac{1}{2}\Big[-4 + L(0,v,w-u) + L(0,u,w+v)\Big] \tag{5.27}$$

□

**Algorithmus 5.11.** *u,v nicht parallel; a=bb bzw. b=aa*

```
function output = J2_touch_b_aa(k,ell,u,v,w)

  L_u_wmv = zeros(k+1,1);
  L_v_wpu = zeros(k+1,1);
  for j = 0:k
      L_u_wmv(j+1) = computation of L(j,u,w-v);
  end
  for j = 0:ell
      L_v_wpu(j+1) = computation of L(j,v,w+u);
  end

  output = zeros(ell+1,1);
  output(1) =  ( -4 + L_v_wpu(1) +L_u_wmv(1) ) *0.5;
  for j = 1:ell
     output(j+1) = ( 2*(1+(-1)^j)/(j+1) + L_v_wpu(j+1) ...
                     + (-1)^j*L_u_wmv(1) + j*output(j) ) / (ell+2);
  end
  for i = 1:k
     output(1) = ( 2*(1+(-1)^k)/(k+1) + L_v_wpu(0) ...
                   - L_u_wmv(i) - k*output(j) ) / (k+2);
     for j = 1:ell
        output(j+1) = ( (1+(-1)^ell)*(1+(-1)^k)/((ell+1)*(k+1)) ...
                        + L_v_wpu(j) + (-1)^ell*L_u_wmv(i) ...
                        - k*output(j) + ell*output(j+1) ) / (k+ell+2);
     end
  end
  output = output(ell+1);


function output = J2_touch_a_bb(k,ell,u,v,w)

  L_u_wpv = zeros(k+1,1);
  L_v_wmu = zeros(k+1,1);
  for j = 0:k
      L_u_wpv(j+1) = computation of L(j,u,w+v);
  end
  for j = 0:ell
      L_v_wmu(j+1) = computation of L(j,v,w-u);
  end

  output = zeros(ell+1,1);
  output(1) =  ( -4 + L_v_wmu(1) + L_u_wpv(1) ) *0.5;
  for j = 1:ell
     output(j+1) = ( 2*(1+(-1)^j)/(j+1) + L_v_wmu(j+1) + L_u_wpv(1) ...
                     - j*output(j) ) / (ell+2);
  end
  for i = 1:k
```

```
    output(1) = ( 2*(1+(-1)^k)/(k+1) + (-1)^k*L_v_wmu(0) + L_u_wpv(i) ...
                  + k*output(j) ) / (k+2);
    for j = 1:ell
       output(j+1) = ( (1+(-1)^ell)*(1+(-1)^k)/((ell+1)*(k+1)) ...
                       + (-1)^k*L_v_wmu(j) + L_u_wpv(i) ...
                       + k*output(j) - ell*output(j+1) ) / (k+ell+2);
    end
end
output = output(ell+1);
```

**Zusammenfassung:**
Insgesamt muss also für alle Intervalle $[a,b] \subseteq \Gamma$, $[\mathfrak{a},\mathfrak{b}] \subseteq \Gamma$

$$\big\langle V\phi_h, \psi_h \big\rangle_\Gamma = -\frac{1}{2\pi}\frac{|b-a|}{2}\frac{|\mathfrak{b}-\mathfrak{a}|}{2}\sum_{\ell=0}^{N}\sum_{k=0}^{M}\lambda_\ell\lambda_k\Big[J(k,\ell,\mathfrak{b}-\mathfrak{a},a-b,\mathfrak{a}+\mathfrak{b}-a-b)+\log\frac{1}{2}\cdot\widetilde{\mathcal{I}}(k,\ell)\Big]$$

mit $\operatorname{supp}\phi_h \subseteq [a,b]$ und $\operatorname{supp}\psi_h \subseteq [\mathfrak{a},\mathfrak{b}]$ berechnet werden, um die Galerkinmatrix des Einfachschichtpotentials zu erhalten.

### 5.4.2 Doppelschichtpotential

Für alle Intervalle $[a,b] \subseteq \Gamma$, $[\mathfrak{a},\mathfrak{b}] \subseteq \Gamma$ ist

$$\begin{aligned}\Big\langle\Big(K+\frac{1}{2}\Big)\phi_h,\psi_h\Big\rangle_\Gamma &= \frac{|\mathfrak{b}-\mathfrak{a}|}{2}\sum_{\ell=0}^{N}\sum_{k=0}^{M}\lambda_\ell\lambda_k\left[\frac{1}{2}\int_{-1}^{1}s^{k+\ell}\,ds\right.\\ &\qquad\left.-\frac{1}{2\pi}\frac{|b-a|}{2}\int_{-1}^{1}\int_{-1}^{1}s^k t^\ell\frac{\big\langle\gamma_{[a,b]}(t)-\gamma_{[\mathfrak{a},\mathfrak{b}]}(s),\nu_{[a,b]}\big\rangle}{|\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)-\gamma_{[a,b]}(t)|^2}\,dt\,ds\right]\end{aligned}$$

zu berechnen, wobei $\operatorname{supp}\phi_h \subseteq [a,b]$ und $\operatorname{supp}\psi_h \subseteq [\mathfrak{a},\mathfrak{b}]$ gilt.
Wir betrachten zunächst einen Teil des Integranden des zweiten Integrals:

$$\begin{aligned}\frac{\gamma_{[a,b]}(t)-\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)}{|\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)-\gamma_{[a,b]}(t)|^2} &= \frac{\frac{1}{2}\big(a+b+t(b-a)-\mathfrak{a}-\mathfrak{b}-s(\mathfrak{b}-\mathfrak{a})\big)}{\frac{1}{4}\Big|\mathfrak{a}+\mathfrak{b}+s(\mathfrak{b}-\mathfrak{a})-a-b-t(b-a)\Big|^2}\\ &= 2\cdot\frac{(b-a)\cdot t+(\mathfrak{a}-\mathfrak{b})\cdot s+(a+b-\mathfrak{a}-\mathfrak{b})}{|(\mathfrak{b}-\mathfrak{a})\cdot s+(a-b)\cdot t+(\mathfrak{a}+\mathfrak{b}-a-b)|^2}.\end{aligned}$$

Wir setzen wieder

$$\begin{aligned}u &:= \mathfrak{a}-\mathfrak{b}\\ v &:= b-a\\ w &:= a+b-\mathfrak{a}-\mathfrak{b}\end{aligned}$$

und erhalten so für das zweite Integral

$$\int_{-1}^{1}\int_{-1}^{1}s^k t^\ell\frac{\big\langle(\gamma_{[a,b]}(t)-\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)),\nu_{[a,b]}\big\rangle}{|\gamma_{[\mathfrak{a},\mathfrak{b}]}(s)-\gamma_{[a,b]}(t)|^2}\,dt\,ds = 2\int_{-1}^{1}\int_{-1}^{1}s^k t^\ell\frac{\big\langle su+tv+w,\nu\big\rangle}{|su+tv+w|^2}\,dt\,ds.$$

Liegen die beiden Intervalle $[a,b]$ und $[\mathfrak{a},\mathfrak{b}]$ auf derselben Geraden, so haben sie denselben Normalvektor und das im Integranden vorkommende Skalarprodukt ist 0. Also erhält man auch für das Doppelschichtpotential den Wert 0:

**Lemma 5.12.** *Seien $[a,b]$ und $[\mathfrak{a},\mathfrak{b}]$ zwei Teilintervalle von $\Gamma$, $c = a+b-2x$ und $d := b-a$. Liegen $[a,b]$ und $[\mathfrak{a},\mathfrak{b}]$ auf einer Geraden, d.h.*

$$|c_1 d_2 - c_2 d_1| = 0 \quad \text{für } x \in [\mathfrak{a},\mathfrak{b}] \quad \text{und} \quad \langle \mathfrak{b}-\mathfrak{a}, \nu\rangle = 0, \tag{5.28}$$

*so gilt für den zugehörigen Eintrag in der Doppelschichtpotentialmatrix $\left\langle K_{[a,b]}\phi_h, \psi_{h,[\mathfrak{a},\mathfrak{b}]}\right\rangle = 0$.*

**Beweis:** $[a,b]$ und $[\mathfrak{a},\mathfrak{b}]$ liegen auf einer Geraden, wenn gilt

$$x \in \overline{ab} \quad \text{für } x \in [\mathfrak{a},\mathfrak{b}] \quad \text{und} \quad \langle \mathfrak{b}-\mathfrak{a}, \nu\rangle = 0$$

$$\begin{aligned} x \in \overline{ab} &\Leftrightarrow \exists t \in \mathbb{R} :\ x = \frac{1}{2}\{a+b+t(b-a)\} \\ &\Leftrightarrow \exists t \in \mathbb{R} :\ 2x-a-b = t(b-a) \\ &\Leftrightarrow \exists t \in \mathbb{R} :\ c = td \\ &\Leftrightarrow c, d \text{ sind linear abhängig} \\ &\Leftrightarrow |c_1 d_2 - c_2 d_1| = 0 \end{aligned}$$

□

**Lemma 5.13.** *Seien $u, v, w \in \mathbb{R}^2$, $u \neq 0$ und $u \parallel v$, d.h. $u = \mu v$ für ein $\mu \in \mathbb{R}\backslash\{0\}$. Wir definieren für $k, \ell \in \mathbb{N}_0$*

$$I(k,\ell,u,v,w) := \int_{-1}^{1}\int_{-1}^{1} s^k t^\ell \frac{\langle su+tv+w, \nu\rangle}{|su+tv+w|^2}\, dt\, ds. \tag{5.29}$$

*Für $k, \ell \geq 0$ gilt dann folgende Rekursionsformel*

$$\begin{aligned} (\ell+1)I(k,\ell,u,v,w) =& \langle w,\nu\rangle G(k,u,w+v) + (-1)^\ell \langle w,\nu\rangle G(k,u,w-v) \\ &- \mu\langle w,\nu\rangle G(\ell+1,v,w+u) + (-1)^k \mu\langle w,\nu\rangle G(\ell+1,v,w-u) \\ &+ k\mu I(k-1,l+1,u,v,w) \end{aligned} \tag{5.30}$$

*mit Abbruchbedingung $k = 0$, da der letzte Summand verschwindet:*

$$\begin{aligned} (\ell+1)I(0,\ell,u,v,w) =& \langle w,\nu\rangle G(0,u,w+v) + (-1)^\ell \langle w,\nu\rangle G(0,u,w-v) \\ &- \mu\langle w,\nu\rangle G(\ell+1,v,w+u) + \mu\langle w,\nu\rangle G(\ell+1,v,w-u) \end{aligned} \tag{5.31}$$

**Beweis:** Wir verwenden Lemma 5.3 mit

$$f(x) = \frac{\langle x,\nu\rangle}{|x|^2}$$

und erhalten

$$\begin{aligned} (\ell+1)I(k,\ell,u,v,w) =& \int_{-1}^{1} s^k \frac{\langle su+w+v,\nu\rangle}{|su+w+v|^2}\, ds - (-1)^{\ell+1}\int_{-1}^{1} s^k \frac{\langle su+w-v,\nu\rangle}{|su+w-v|^2}\, ds \\ &- \mu\int_{-1}^{1} t^{\ell+1}\frac{\langle tv+w+u,\nu\rangle}{|tv+w+u|^2}\, dt + (-1)^k\mu\int_{-1}^{1} t^{\ell+1}\frac{\langle tv+w-u,\nu\rangle}{|tv+w-u|^2}\, dt \\ &+ k\mu I(k-1,\ell+1,u,v,w). \end{aligned}$$

Wegen $\langle pt+q,\nu\rangle = t\langle p,\nu\rangle + \langle q,\nu\rangle$ kann man die Summenregel für Integrale anwenden. Der Vektor $\nu$ steht aufgrund seiner speziellen Wahl normal auf $v$ und, da $u \parallel v$ gilt, auch normal auf $u$.
Das heißt $\langle v,\nu\rangle = \langle u,\nu\rangle = 0$ und $\langle w \pm v,\nu\rangle = \langle w \pm u,\nu\rangle = \langle w,\nu\rangle$. Durch Streichen der Terme mit diesen Vorfaktoren ergibt sich für das Integral

$$\begin{aligned}(\ell+1)I(k,\ell,u,v,w) =& \langle w,\nu\rangle \int_{-1}^{1} \frac{s^k}{|su+w+v|^2}\,ds + (-1)^\ell \langle w,\nu\rangle \int_{-1}^{1} \frac{s^k}{|su+w-v|^2}\,ds \\ &- \mu\langle w,\nu\rangle \int_{-1}^{1} \frac{t^{\ell+1}}{|tv+w+u|^2}\,dt + (-1)^k \mu\langle w,\nu\rangle \int_{-1}^{1} \frac{t^{\ell+1}}{|tv+w-u|^2}\,dt \\ &+ k\mu I(k-1,\ell+1,u,v,w).\end{aligned}$$

Verwendet man nun die Funktion aus Lemma 5.1, so erhält man

$$\begin{aligned}(\ell+1)I(k,\ell,u,v,w) =& \langle w,\nu\rangle G(k,u,w+v) + (-1)^\ell \langle w,\nu\rangle G(k,u,w-v) \\ &- \mu\langle w,\nu\rangle G(\ell+1,v,w+u) + (-1)^k \mu\langle w,\nu\rangle G(\ell+1,v,w-u) \\ &+ k\mu I(k-1,\ell+1,u,v,w).\end{aligned}$$

□

Um das Integral $I(k,\ell,u,v,w)$ aus Lemma 5.13 zu bestimmen, muss eine Rekursion ausgeführt werden. Dies lässt sich auch leicht induktiv bewerkstelligen:

$$I_{0,\ell+k} \to I_{1,\ell+k-1} \to \ldots \to I_{k-1,\ell+1} \to I_{k,\ell}$$

Man bestimmt zuerst $I(0,\ell+k,u,v,w)$ und dann $I(i,\ell+k-i,u,v,w)$ für $i = 1,\ldots,k$. Zur Berechnung dieses Integrals benötigt man daher $k+1$ Schleifendurchläufe.

Auch die benötigten einfachen Integrale, d.h die Funktionen $G(i,p,q)$, kann man induktiv berechnen. Um $I(k,\ell,u,v,w)$ zu erhalten, müssen die Funktionswerte von $G(i,u,w+v)$ und $G(i,u,w-v)$ für $i = 0,\ldots,k$ und $G(i,v,w+u)$ und $G(i,v,w-u)$ für $i = \ell+1,\ldots,\ell+k+1$ bekannt sein.
Man bestimmt also in $k+1$ Schritten $G(i,u,w+v)$ und $G(i,u,w-v)$. In $\ell+k+2$ Schritten ermittelt man $G(i,v,w+u)$ und $G(i,v,w-u)$.

**Algorithmus 5.14.** *u,v parallel*

```
function output = I1(k,ell,u,v,w)

  compute mu;

  n = [u(2), -u(1)];
  w_n = w(1)*n(1)+w(2)*n(2);

  G_u_wpv = zeros(k+1,1);
  G_u_wmv = zeros(k+1,1);
  G_v_wpu = zeros(ell+1,1);
  G_v_wmu = zeros(ell+1,1);
  for j = 0:k
      G_u_wpv(j+1) = computation of G(j,u,w+v);
      G_u_wmv(j+1) = computation of G(j,u,w-v);
  end
```

```
for j = 0:ell
    G_v_wpu(1) = computation of G(j,v,w+u) by updating G_v_wpu(1);
    G_v_wmu(1) = computation of G(j,v,w-u) by updating G_v_wmu(1);
end
for j = (ell+1):(ell+k+1)
    G_v_wpu(j-ell) = computation of G(j,v,w+u);
    G_v_wmu(j-ell) = computation of G(j,v,w-u);
end

output = ( G_u_wpv(1) + (-1)^(ell+k)*G_u_wmv(1) - mu*G_v_wpu(k+1) ...
           + mu*G_v_wmu(k+1) ) *w_n/(ell+k+1);
for j = 1:k
   output = ( w_n*G_u_wpv(j) + (-1)^(ell+k-j)*w_n*G_u_wmv(j) ...
              - mu*w_n*G_v_wpu(k+1-j) + mu*w_n*(-1)^j*G_v_wmu(k+1-j) ...
              + j*mu*output ) / (ell+k+1-j);
end
```

**Lemma 5.15.** *Seien $u, v, w \in \mathbb{R}^2$ mit $u \neq 0, v \neq 0$ und $u \nparallel v$, d.h. $w = \mu_1 u + \mu_2 v$ mit $\mu_1, \mu_2 \in \mathbb{R}$. Wir definieren für $k, \ell \in \mathbb{N}_0$*

$$I(k,\ell,u,v,w) := \int_{-1}^{1}\int_{-1}^{1} s^k t^\ell \, \frac{\langle su+tv+w, \nu\rangle}{|su+tv+w|^2}\, dt\, ds. \tag{5.32}$$

*Für $k, \ell \geq 0$ gilt dann folgende Rekursionsformel*

$$\begin{aligned}
&(k+\ell+1)I(k,\ell,u,v,w)\\
&\quad= (\mu_1+1)\langle w+u,\nu\rangle G(\ell,v,w+u) - (-1)^k(\mu_1-1)\langle w-u,\nu\rangle G(\ell,v,w-u)\\
&\qquad+(\mu_2+1)\langle u,\nu\rangle G(k+1,u,w+v) + (\mu_2+1)\langle w,\nu\rangle G(k,u,w+v)\\
&\qquad-(-1)^\ell(\mu_2-1)\langle u,\nu\rangle G(k+1,u,w-v) - (-1)^\ell(\mu_2-1)\langle w,\nu\rangle G(k,u,w-v)\\
&\qquad-k\mu_1 I(k-1,\ell,u,v,w) - \ell\mu_2 I(k,\ell-1,u,v,w),
\end{aligned} \tag{5.33}$$

*wobei für $k = 0$ der vorletzte und für $\ell = 0$ der letzte Summand entfällt. Die Rekursion bricht ab, sobald $k = \ell = 0$ gilt:*

$$\begin{aligned}
I(0,0,u,v,w) = {}&(\mu_1+1)\langle w+u,\nu\rangle G(0,v,w+u) - (\mu_1-1)\langle w-u,\nu\rangle G(0,v,w-u)\\
&+(\mu_2+1)\langle u,\nu\rangle G(1,u,w+v) + (\mu_2+1)\langle w,\nu\rangle G(0,u,w+v)\\
&-(\mu_2-1)\langle u,\nu\rangle G(1,u,w-v) - (\mu_2-1)\langle w,\nu\rangle G(0,u,w-v)
\end{aligned} \tag{5.34}$$

**Beweis:** Die Funktion $f : \mathbb{R}^2 \to \mathbb{R}$,

$$f(x) = \frac{\langle x,\nu\rangle}{|x|^2}$$

erfüllt die Bedingung $\langle x, \nabla f(x)\rangle = h(x) + pf(x)$ von Lemma 5.4 mit $h(x) \equiv 0$ und $p = -1$: Ausrechnen des Skalarprodukts liefert

$$f(x) = \frac{x_1}{|x|^2}\cdot\nu_1 + \frac{x_2}{|x|^2}\cdot\nu_2$$

Nach Kettenregel gilt

$$\begin{aligned}
\partial_k \frac{x_j}{|x|^2} &= \frac{\delta_{jk}|x|^2 - 2x_j x_k}{|x|^4} \\
\nabla \frac{x_1}{|x|^2} &= \left( \frac{x_2^2 - x_1^2}{|x|^4} \; ; \; -\frac{2x_1x_2}{|x|^4} \right) \\
\nabla \frac{x_2}{|x|^2} &= \left( -\frac{2x_1x_2}{|x|^4} \; ; \; \frac{x_1^2 - x_2^2}{|x|^4} \right)
\end{aligned}$$

$$\begin{aligned}
\big\langle x, \nabla f(x) \big\rangle &= \left\langle \left( \frac{x_1x_2^2 - x_1^3 - 2x_1x_2^2}{|x|^4} \; ; \; \frac{-2x_1^2x_2 + x_1^2x_2 - x_2^3}{|x|^4} \right), \nu \right\rangle \\
&= \left\langle \left( \frac{-x_1\big(x_1^2 + x_2^2\big)}{|x|^4} \; ; \; \frac{-x_2\big(x_1^2 + x_2^2\big)}{|x|^4} \right), \nu \right\rangle \\
&= \left\langle \left( -\frac{x_1}{|x|^2} \; ; \; -\frac{x_2}{|x|^2} \right), \nu \right\rangle \\
&= -f(x) = h(x) + pf(x)
\end{aligned}$$

Es gilt also $h(x) \equiv 0$ und $p = -1$. Wendet man nun Lemma 5.4 an, erhält man

$$\begin{aligned}
&(k+\ell+1)I(k,\ell,u,v,w) \\
&\quad = (\mu_1+1)\int_{-1}^{1} t^\ell \frac{\big\langle tv+w+u, \nu \big\rangle}{|tv+w+u|^2}\, dt - (-1)^k(\mu_1-1)\int_{-1}^{1} t^\ell \frac{\big\langle tv+w-u, \nu \big\rangle}{|tv+w-u|^2}\, dt \\
&\qquad + (\mu_2+1)\int_{-1}^{1} s^k \frac{\big\langle su+w+v, \nu \big\rangle}{|su+w+v|^2}\, ds - (-1)^\ell(\mu_2-1)\int_{-1}^{1} s^k \frac{\big\langle su+w-v, \nu \big\rangle}{|su+w-v|^2}\, ds \\
&\qquad - k\mu_1 I(k-1,\ell,u,v,w) - \ell\mu_2 I(k,\ell-1,u,v,w).
\end{aligned}$$

Wegen $\langle pt+q, \nu\rangle = t\langle p, \nu\rangle + \langle q, \nu\rangle$ kann man wiederum die Summenregel für Integrale anwenden. Der Vektor $\nu$ steht aufgrund seiner speziellen Wahl normal auf $v$.
Das heißt $\langle v, \nu\rangle = 0$ und $\langle w \pm v, \nu\rangle = \langle w, \nu\rangle$. Durch Streichen der Terme mit diesen Vorfaktoren ergibt sich für das Integral:

$$\begin{aligned}
&(k+\ell+1)I(k,\ell,u,v,w) \\
&\quad = (\mu_1+1)\langle w+u, \nu\rangle \int_{-1}^{1} \frac{t^\ell}{|tv+w+u|^2}\, dt - (-1)^k(\mu_1-1)\langle w-u, \nu\rangle \int_{-1}^{1} \frac{t^\ell}{|tv+w-u|^2}\, dt \\
&\qquad + (\mu_2+1)\langle u, \nu\rangle \int_{-1}^{1} \frac{s^{k+1}}{|su+w+v|^2}\, ds + (\mu_2+1)\langle w, \nu\rangle \int_{-1}^{1} \frac{s^k}{|su+w+v|^2}\, ds \\
&\qquad - (-1)^\ell(\mu_2-1)\langle u, \nu\rangle \int_{-1}^{1} \frac{s^{k+1}}{|su+w-v|^2}\, ds - (-1)^\ell(\mu_2-1)\langle w, \nu\rangle \int_{-1}^{1} \frac{s^k}{|su+w-v|^2}\, ds \\
&\qquad - k\mu_1 I(k-1,\ell,u,v,w) - \ell\mu_2 I(k,\ell-1,u,v,w).
\end{aligned}$$

Verwendet man nun die Funktion aus Lemma 5.1, so erhält man

$$\begin{aligned}
&(k+\ell+1)I(k,\ell,u,v,w) \\
&\quad = (\mu_1+1)\langle w+u, \nu\rangle G(\ell, v, w+u) - (-1)^k(\mu_1-1)\langle w-u, \nu\rangle G(\ell, v, w-u) \\
&\qquad + (\mu_2+1)\langle u, \nu\rangle G(k+1, u, w+v) + (\mu_2+1)\langle w, \nu\rangle G(k, u, w+v) \\
&\qquad - (-1)^\ell(\mu_2-1)\langle u, \nu\rangle G(k+1, u, w-v) - (-1)^\ell(\mu_2-1)\langle w, \nu\rangle G(k, u, w-v) \\
&\qquad - k\mu_1 I(k-1,\ell,u,v,w) - \ell\mu_2 I(k,\ell-1,u,v,w)
\end{aligned}$$

□

Auch diese Rekursionsformel lässt sich induktiv umschreiben - das Berechnungsschema entspricht hierbei einer Matrix.

$$\begin{array}{ccccccc}
I_{0,0} & \to & I_{0,1} & \to & I_{0,2} & \cdots & I_{0,\ell} \\
\downarrow & & \downarrow & & \downarrow & & \downarrow \\
I_{1,0} & \to & I_{1,1} & \to & I_{1,2} & \cdots & I_{1,\ell} \\
\vdots & & \vdots & & \vdots & & \vdots \\
I_{k-1,0} & \to & I_{k-1,1} & \to & I_{k-1,2} & \cdots & I_{k-1,\ell} \\
\downarrow & & \downarrow & & \downarrow & & \downarrow \\
I_{k,0} & \to & I_{k,1} & \to & I_{k,2} & \cdots & I_{k,\ell}
\end{array}$$

Zuerst ermittelt man den Wert von $I(0,0,u,v,w)$ und dann von $I(0,j,u,v,w)$ für $j = 1,\ldots,\ell$ - das entspricht der ersten Zeile des Schemas. Aus den $\ell+1$ nun errechneten Werten bestimmt man nun zeilenweise von links nach rechts $I(i,j,u,v,w)$ für $j = 1,\ldots,k+1$ und $i = 1,\ldots,\ell+1$. Insgesamt erhält das gewünschte Integral also in $(k+1)\cdot(\ell+1)$ Schritten.

Auch die benötigten einfachen Integrale, d.h die Funktionen $G(i,p,q)$, kann man induktiv berechnen. Um $I(k,\ell,u,v,w)$ zu erhalten, müssen die Funktionswerte von $G(i,u,w+v)$ und $G(i,u,w-v)$ für $i = 0,\ldots,k+1$ und $G(i,v,w+u)$ und $G(i,v,w-u)$ für $i = 0,\ldots,\ell$ bekannt sein.

**Algorithmus 5.16.** *u,v nicht parallel*

```
function output = I2(k,ell,u,v,w)

  compute mu1,mu2;

  wpu = w+u;
  wmu = w-u;
  n = [v(2), -v(1)];
  u_n = u(1)*n(1)+u(2)*n(2);
  w_n = w(1)*n(1)+w(2)*n(2);
  wpu_n = wpu(1)*n(1)+wpu(2)*n(2);
  wmu_n = wmu(1)*n(1)+wmu(2)*n(2);

  G_u_wpv = zeros(k+2,1);
  G_u_wmv = zeros(k+2,1);
  G_v_wpu = zeros(ell+1,1);
  G_v_wmu = zeros(ell+1,1);
  for j = 0:k+1
      G_u_wpv(j+1) = computation of G(j,u,w+v);
      G_u_wmv(j+1) = computation of G(j,u,w-v);
  end
  for j = 0:ell
      G_v_wpu(j+1) = computation of G(j,v,w+u);
      G_v_wmu(j+1) = computation of G(j,v,w-u);
  end

  output = zeros(ell+1,1);
```

```
output(1) = (mu1+1)*wpu_n*G_v_wpu(1) - (mu1-1)*wmu_n*G_v_wmu(1) ...
            + (mu2+1)*u_n*G_u_wpv(2) + (mu2+1)*w_n*G_u_wpv(1) ...
            - (mu2-1)*u_n*G_u_wmv(2) - (mu2-1)*w_n*G_u_wmv(1);
for j = 1:ell
   output(j+1) = ( (mu1+1)*wpu_n*G_v_wpu(j+1) - (mu1-1)*wmu_n*G_v_wmu(j+1) ...
                 + (mu2+1)*u_n*G_u_wpv(2) + (mu2+1)*w_n*G_u_wpv(1) ...
                 - (-1)^j*(mu2-1)*u_n*G_u_wmv(2) ...
                 - (-1)^j*(mu2-1)*w_n*G_u_wmv(1) ...
                 - j*mu2*output(j) ) / (j+1);
end
for i = 1:k
   output(1) = ( (mu1+1)*wpu_n*G_v_wpu(1) - (-1)^i*(mu1-1)*wmu_n*G_v_wmu(1) ...
               + (mu2+1)*u_n*G_u_wpv(i+2) + (mu2+1)*w_n*G_u_wpv(i+1) ...
               - (mu2-1)*u_n*G_u_wmv(i+2) - (mu2-1)*w_n*G_u_wmv(i+1) ...
               - i*mu1*output(1) ) / (i+1);
   for j = 1:ell
      output(j+1) = ( (mu1+1)*wpu_n*G_v_wpu(j+1) ...
                    - (-1)^i*(mu1-1)*wmu_n*G_v_wmu(j+1) ...
                    + (mu2+1)*u_n*G_u_wpv(i+2) + (mu2+1)*w_n*G_u_wpv(i+1) ...
                    - (-1)^j*(mu2-1)*u_n*G_u_wmv(i+2) ...
                    - (-1)^j*(mu2-1)*w_n*G_u_wmv(i+1) ...
                    - i*mu1*output(j+1) - j*mu2*output(j) ) / (i+j+1);
   end
end
output = output(ell+1);
```

Im Fall von Lemma 5.15, d.h. $u \nparallel v$, und induktiver Berechnung aller Funktionen ist besondere Vorsicht geboten, falls sich die beiden Intervalle $[a,b]$ und $[\mathfrak{a},\mathfrak{b}]$ berühren, d.h entweder $a = \mathfrak{b}$ oder $b = \mathfrak{a}$ gilt. Unter diesen Gegebenheiten treten Probleme bei der Berechnung von $G(1,p,q)$ auf, da $p+q = 0$ oder $p-q = 0$ gilt und $\log|p+q|$ bzw. $\log|p-q|$ errechnet werden sollte. Betrachtet man jedoch die Formel für $I(k,l,u,v,w)$ aus Lemma 5.15, so erkennt man, dass die kritischen Terme wegfallen:

**Lemma 5.17.** *Seien $u,v,w \in \mathbb{R}^2$ mit $u \neq 0, v \neq 0$, für die gilt $u \nparallel v$.*

*(i) Ist $b = \mathfrak{a}$, so gilt $w = u - v$, d.h. $\mu_1 = 1$ und $\mu_2 = -1$, und mit $k,\ell \geq 0$ für das in Lemma 5.15 definierte Integral die Rekursionsformel*

$$\begin{aligned}(k+\ell+1)I(k,\ell,u,v,w) = {} & 2\langle w+u,\nu\rangle G(\ell,v,w+u) + 2(-1)^\ell\langle u,\nu\rangle G(k+1,u,w-v) \\ & + 2(-1)^\ell\langle w,\nu\rangle G(k,u,w-v) - kI(k-1,\ell,u,v,w) + \ell I(k,\ell-1,u,v,w),\end{aligned} \tag{5.35}$$

*wobei für $k = 0$ der vorletzte und für $\ell = 0$ der letzte Summand entfällt. Die Rekursion bricht ab, sobald $k = \ell = 0$ gilt:*

$$I(0,0,u,v,w) = 2\Big(\langle w+u,\nu\rangle G(0,v,w+u) + \langle u,\nu\rangle G(1,u,w-v) + \langle w,\nu\rangle G(0,u,w-v)\Big) \tag{5.36}$$

*(ii) Ist $a = \mathfrak{b}$ so gilt $w = v - u$, d.h. $\mu_1 = -1$ und $\mu_2 = 1$, und mit $k, \ell \geq 0$ für das in Lemma 5.15 definierte Integral die Rekursionsformel*

$$(k+\ell+1)I(k,\ell,u,v,w) = 2(-1)^k\langle w-u,\nu\rangle G(\ell,v,w-u) + 2\langle u,\nu\rangle G(k+1,u,w+v) \\ + 2\langle w,\nu\rangle G(k,u,w+v) + kI(k-1,\ell,u,v,w) - \ell I(k,\ell-1,u,v,w), \tag{5.37}$$

*wobei für $k = 0$ der vorletzte und für $\ell = 0$ der letzte Summand entfällt. Die Rekursion bricht ab, sobald $k = \ell = 0$ gilt:*

$$I(0,0,u,v,w) = 2\Big(\langle w-u,\nu\rangle G(0,v,w-u) + \langle u,\nu\rangle G(1,u,w+v) + \langle w,\nu\rangle G(0,u,w+v)\Big) \tag{5.38}$$

□

**Algorithmus 5.18.** *u,v nicht parallel; a=bb bzw. b=aa*

```
function output = I2_touch_b_aa(k,ell,u,v,w)

  wpu = w+u;
  n = [v(2), -v(1)];
  u_n = u(1)*n(1)+u(2)*n(2);
  w_n = w(1)*n(1)+w(2)*n(2);
  wpu_n = wpu(1)*n(1)+wpu(2)*n(2);

  G_u_wmv = zeros(k+2,1);
  G_v_wpu = zeros(ell+1,1);
  for j = 0:k+1
      G_u_wpv(j+1) = computation of G(j,u,w-v);
  end
  for j = 0:ell
      G_v_wmu(j+1) = computation of G(j,v,w+u);
  end

  output = zeros(ell+1,1);
  output(1) = 2*wpu_n*G_v_wpu(1) + 2*u_n*G_u_wmv(2) + 2*w_n*G_u_wmv(1);
  for j = 1:ell
     output(j+1) = ( 2*wpu_n*G_v_wpu(j+1) + 2*(-1)^j*u_n*G_u_wmv(2) ...
                   + 2*(-1)^j*w_n*G_u_wmv(1) + j*output(j) ) / (j+1);
  end
  for i = 1:k
     output(1) = ( + 2*wpu_n*G_v_wpu(1) + 2*u_n*G_u_wmv(i+2) ...
                 + 2*w_n*G_u_wmv(i+1) - i*output(1) ) / (i+1);
     for j = 1:ell
        output(j+1) = ( 2*wpu_n*G_v_wpu(j+1) + 2*(-1)^j*u_n*G_u_wmv(i+2) ...
                      + 2*(-1)^j*w_n*G_u_wmv(i+1) ...
                      - i*output(j+1) - j*output(j) ) / (i+j+1);
     end
  end
```

```
  output = output(ell+1);


function output = I2_touch_a_bb(k,ell,u,v,w)

  wmu = wmu;
  n = [v(2), -v(1)];
  u_n = u(1)*n(1)+u(2)*n(2);
  w_n = w(1)*n(1)+w(2)*n(2);
  wmu_n = wmu(1)*n(1)+wmu(2)*n(2);

  G_u_wpv = zeros(k+2,1);
  G_v_wmu = zeros(ell+1,1);

  for j = 0:k+1
      G_u_wmv(j+1) = computation of G(j,u,w-v);
  end
  for j = 0:ell
      G_v_wpu(j+1) = computation of G(j,v,w+u);
  end

  output = zeros(ell+1,1);
  output(1) = 2*wmu_n*G_v_wpu(1) + 2*u_n*G_u_wpv(2) + 2*w_n*G_u_wpv(1);
  for j = 1:ell
     output(j+1) = ( 2*wmu_n*G_v_wmu(j+1) + 2*u_n*G_u_wpv(2)...
                   + 2*w_n*G_u_wpv(1) + j*output(j) ) / (j+1);;
  end
  for i = 1:k
     output(1) = ( 2*(-1)^i*wmu_n*G_v_wmu(1) + 2*u_n*G_u_wpv(i+2)...
                 + 2*w_n*G_u_wpv(i+1) + i*output(1) ) / (i+1);
     for j = 1:ell
        output(j+1) = ( 2*(-1)^i*wmu_n*G_v_wmu(j+1) + 2*u_n*G_u_wpv(i+2)...
                      + 2*w_n*G_u_wpv(i+1) + i*output(j+1) ...
                      + j*output(j+1) ) / (i+j+1);
     end
  end
  output = output(ell+1);
```

Das erste Integral lässt sich elementar wie in Lemma 5.5 berechnen.

**Zusammenfassung:**
Insgesamt muss für alle Intervalle $[a,b] \subseteq \Gamma$, $[\mathfrak{a},\mathfrak{b}] \subseteq \Gamma$

$$\left\langle \left(K+\frac{1}{2}\right)\phi_h, \psi_h \right\rangle_\Gamma = \frac{|\mathfrak{b}-\mathfrak{a}|}{2} \sum_{\ell=0}^{N} \sum_{k=0}^{M} \lambda_\ell \lambda_k \left[ \frac{1-(-1)^{k+\ell+1}}{k+\ell+1} - \frac{1}{\pi} \frac{|b-a|}{2} I(k,\ell,b-a,\mathfrak{a}-\mathfrak{b},a+b-\mathfrak{a}-\mathfrak{b}) \right]$$

mit $\operatorname{supp}\phi_h \subseteq [a,b]$ und $\operatorname{supp}\psi_h \subseteq [\mathfrak{a},\mathfrak{b}]$ berechnet werden, um die Galerkinmatrix des Doppelschichtpotentials zu erhalten.

### 5.4.3 exakte rechte Seite

Unter Verwendung der Tatsache, dass die Integraloperatoren $K$ und $K'$ adjungiert sind, kann man die rechte Seite der Symm'schen Integralgleichung exakt berechnen.

Wir betrachten die rechte Seite

$$F := \langle (K + \tfrac{1}{2})g, \psi_h \rangle_\Gamma.$$

$\psi_h$ sei — wie schon zu Beginn dieses Abschnitts — ein stückweises Polynom in der Bogenlänge vom Grad N

$$\psi_h|_{T_j} = \psi_h \circ \gamma(s) = \sum_{\ell=0}^{M} \lambda_\ell s^\ell \quad \text{für } s \in [-1;1],$$

wobei $\gamma : [-1;1] \to T_j$ die Parametrisierung aus Gleichung (5.17) ist.

Wendet man Satz 2.29 an, so kann man vom Doppelschichtpotential zu dessen Adjungierter übergehen

$$F = \langle (K + \tfrac{1}{2})g, \psi_h \rangle_\Gamma = \langle g, (K' + \tfrac{1}{2})\psi_h \rangle_\Gamma,$$

wobei sich das adjungierte Doppelschichtpotential darstellen lässt als

$$(K'\psi_h)(x) := -\frac{1}{2\pi} \int_\Gamma \psi_h(y) \frac{\langle x-y, \nu_x \rangle}{|x-y|^2} ds_y.$$

Für jedes Element $T_j = [a,b]$ der Partition $\mathcal{T}_h$ muss also

$$\begin{aligned} F_j &= \frac{1}{2} \int_{T_j} g(x)\psi_h(x)\, ds_x + \int_\Gamma g(x) \big(K'\psi_h|_{T_j}\big)(x)\, ds_x \\ &= \frac{1}{2} \int_{T_j} g(x)\psi_h(x)\, ds_x + \sum_{\substack{T_k = [\mathfrak{a},\mathfrak{b}] \\ T_k \in \mathcal{T}_h}} \int_{T_k} g(x) \big(K'\psi_h|_{T_j}\big)(x)\, ds_x \end{aligned}$$

berechnet werden.

Wir betrachten zunächst das erste Integral

$$\frac{1}{2} \int_{T_j} g(x)\psi_h(x)\, ds_x = \frac{1}{2} \int_{[a,b]} g(x)\psi_h(x)\, ds_x = \frac{1}{2} \frac{|b-a|}{2} \sum_{\ell=0}^{M} \lambda_\ell \int_{-1}^{1} s^\ell g\big(\gamma_{[a,b]}(s)\big)\, ds.$$

Hierfür kann man die übliche Gauss-Quadraturformel benutzen, d.h

$$\frac{1}{2} \int_{T_j} g(x)\psi_h(x)\, ds_x = \frac{1}{2} \frac{|b-a|}{2} \sum_{\ell=0}^{M} \lambda_\ell \sum_{i=1}^{p} \omega_i\, s_i^\ell\, g\big(\gamma_{[a,b]}(s_i)\big)\, ds.$$

Für das zweite Integral berechnen wir zunächst das Doppelschichtpotential für ein Element $T_j = [a,b]$

$$(K'\psi|_{[a,b]})(x) = -\frac{1}{2\pi}\int_{[a,b]} \psi(y)\frac{\langle x-y,\nu_x\rangle}{|x-y|^2}\,ds_y = -\frac{1}{2\pi}\frac{|b-a|}{2}\sum_{\ell=0}^{M}\lambda_\ell\int_{-1}^{1} s^\ell \frac{\langle x-\gamma(s),\nu_x\rangle}{|x-\gamma(s)|^2}\,ds.$$

Mit $d := b-a$ und $c := a+b-2x$ gilt

$$\begin{aligned}|x-y|^2 = \Big|x-\frac{1}{2}\big(a+b+s(b-a)\big)\Big|^2 = \frac{1}{4}\Big|2x-a-b-s(b-a)\Big|^2 &= \frac{1}{4}|ds-c|^2\\ &= \frac{1}{4}\Big(|d|^2s^2 - 2s\langle c,d\rangle + |c|^2\Big).\end{aligned}$$

Die Diskriminante lautet $\Delta := 4\langle c,d\rangle^2 - 4|c|^2|d|^2$. Wegen der Cauchy-Schwarz-Ungleichung gilt stets $\Delta \le 0$ und es müssen daher nur zwei Fälle unterschieden werden:

$$\begin{aligned}&\Delta = 0 \Leftrightarrow D := \sqrt{|d|^2|c|^2 - \langle c,d\rangle^2} = 0\\ &\Delta < 0 \Leftrightarrow D > 0\end{aligned}$$

**1. Fall:** $D > 0$. Für den Integranden gilt

$$\begin{aligned}s^\ell\frac{\langle x-\gamma(s),\nu_x\rangle}{|x-\gamma(s)|^2} &= 4s^\ell\frac{\frac{1}{2}\langle ds-c,\nu_x\rangle}{|d|^2s^2 - 2\langle c,d\rangle s + |c|^2}\\ &= 2s^\ell\frac{\langle d,\nu_x\rangle s - \langle c,\nu_x\rangle}{|d|^2s^2 - 2\langle c,d\rangle s + |c|^2}\\ &= 2\langle d,\nu_x\rangle\frac{s^{\ell+1}}{|d|^2s^2 - 2\langle c,d\rangle s + |c|^2} - 2\langle c,\nu_x\rangle\frac{s^\ell}{|d|^2s^2 - 2\langle c,d\rangle s + |c|^2},\end{aligned}$$

also insgesamt mit der Funktion $G$ aus Lemma 5.1

$$\begin{aligned}(K'\psi|_{[a,b]})(x) &= -\frac{1}{2\pi}\frac{|b-a|}{2}2\sum_{\ell=0}^{M}\lambda_\ell\bigg(\langle d,\nu_x\rangle\frac{s^{\ell+1}}{|d|^2s^2 - 2\langle c,d\rangle s + |c|^2} - \langle c,\nu_x\rangle\frac{s^\ell}{|d|^2s^2 - 2\langle c,d\rangle s + |c|^2}\bigg)\\ &= -\frac{1}{2\pi}|b-a|\sum_{\ell=0}^{M}\lambda_\ell\Big(\langle d,\nu_x\rangle G(\ell+1,d,-c) - \langle c,\nu_x\rangle G(\ell,d,-c)\Big).\end{aligned}$$

**2. Fall:** $D = 0$.
**2.1 Fall:** $|c| \le |d|$. In diesem Fall gibt es ein $s \in [-1;1]$, sodass $|c| = |s|\ |d|$. Dies ist gleichbedeutend mit der Aussage, dass es ein $s \in [-1;1]$ gibt, dass $2x-a-b = s(b-a)$ gilt. Durch eine Äquivalenzumformung erhält man die Existenz eines $s \in [-1;1]$ mit $x = \frac{1}{2}\big(a+b+s(b-a)\big)$. Das ist wiederum äquivalent zur Aussage, dass $x$ im Intervall $[a,b]$ liegen muss. Für $y \in [a,b]$, $x \ne y$ ist dann der Vektor $x-y$ parallel zu $b-a$, also orthogonal zur Normalen $\nu_x$. Damit verschwindet der Zähler des Integranden, d.h.

$$(K'\psi_h|_{[a,b]})(x) = 0.$$

**2.2 Fall:** $|c| > |d|$.
Die Idee für diesen Fall ist es, $\nu_x$ in einen tangentialen und einen normalen Anteil bezüglich $x-y$ aufzuteilen, d.h.

$$\nu_x = T + N \qquad \text{mit} \qquad N = \tilde{\alpha}\nu_y = \alpha\begin{pmatrix} b_2-a_2 \\ a_1-b_1\end{pmatrix},\ \ T = \tilde{\beta}(x-y) = \beta(b-a).$$

Nach Voraussetzung existiert für jedes $y \in [a,b]$ ein $\mu \neq 0$ mit $\mu(x-y) = d$, da $x \in \overline{ab}\backslash[a,b]$. Aufgrund der Orthogonalität (insb. der linearen Unabhängigkeit) von $(b_2-a_2, a_1-b_1)$ und $d = (b_1-a_1, b_2-a_2)$ existieren $\alpha, \beta \in \mathbb{R}$ mit

$$\frac{(\mathfrak{d}_2, -\mathfrak{d}_1)}{|\mathfrak{d}|} = \nu_x = \alpha\begin{pmatrix} d_2 \\ -d_1 \end{pmatrix} + \beta\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} \beta d_1 + \alpha d_2 \\ \beta d_2 - \alpha d_1 \end{pmatrix}.$$

Im Fall $d_2 = 0$ erhalten wir $\frac{\mathfrak{d}_2}{|\mathfrak{d}|} = \beta d_1$, also

$$\beta = \frac{\mathfrak{d}_2}{|\mathfrak{d}|d_1} = \frac{\mathfrak{d}_2 d_1}{|\mathfrak{d}|d_1^2} = \frac{\mathfrak{d}_2 d_1}{|\mathfrak{d}||d|^2},$$

denn aus $d \neq 0$ und $d_2 = 0$ folgt $d_1 \neq 0$.
Im Fall $d_2 \neq 0$ erhalten wir zunächst aus der ersten Zeile

$$\frac{\mathfrak{d}_2}{|\mathfrak{d}|} = \beta d_1 + \alpha d_2, \quad \text{also} \quad \alpha = \frac{1}{d_2}\Big(\frac{\mathfrak{d}_2}{|\mathfrak{d}|} - \beta d_1\Big).$$

Nun lautet die zweite Zeile

$$\begin{aligned}\frac{-\mathfrak{d}_1}{|\mathfrak{d}|} = \beta d_2 - \alpha d_1 = \beta d_2 - \frac{d_1}{d_2}\Big(\frac{\mathfrak{d}_2}{|\mathfrak{d}|} - \beta d_1\Big) = \beta d_2 - \frac{d_1\mathfrak{d}_2}{d_2|\mathfrak{d}|} + \beta\frac{d_1^2}{d_2} = \beta\,\frac{d_1^2+d_2^2}{d_2} - \frac{d_1\mathfrak{d}_2}{d_2|\mathfrak{d}|} \\ = \beta\,\frac{|d|^2}{d_2} - \frac{d_1\mathfrak{d}_2}{d_2|\mathfrak{d}|},\end{aligned}$$

und wir erhalten für $\beta$ die Gleichung

$$\beta = \frac{d_2}{|d|^2}\Big(\frac{d_1\mathfrak{d}_2}{d_2|\mathfrak{d}|} - \frac{\mathfrak{d}_1}{|\mathfrak{d}|}\Big) = \frac{d_2}{|d|^2}\Big(\frac{d_1\mathfrak{d}_2 - \mathfrak{d}_1 d_2}{d_2|\mathfrak{d}|}\Big) = \frac{d_1\mathfrak{d}_2 - d_2\mathfrak{d}_1}{|d|^2|\mathfrak{d}|} = \frac{\langle d, (\mathfrak{d}_2, -\mathfrak{d}_1)\rangle}{|d|^2|\mathfrak{d}|} = \frac{\langle d, \nu_x\rangle}{|d|^2},$$

die sich auch mit dem Fall $d_2 = 0$ verträgt. Für $y \in [a,b]$ gilt nun

$$\begin{aligned}\langle x-y, \nu_x\rangle = \langle x-y, (T+N)\rangle = \langle x-y, T\rangle = \beta\,\langle x-y, d\rangle = \beta\mu\,\langle x-y, x-y\rangle \\ = \beta\mu\,|x-y|^2,\end{aligned}$$

was das Problem darauf eingrenzt, für jedes $y$ den Skalar $\mu = \mu(y)$ zu bestimmen und die so erhaltene Funktion $\mu$ über $[a,b]$ zu integrieren. Es gilt $\mu(x-y) = d$, also auch komponentenweise aufaddiert

$$\begin{aligned}d_1 + d_2 &= \mu\big\{x_1 - y_1 + x_2 - y_2\big\} \\ &= \mu\frac{1}{2}\big\{2x_1 - \big(a_1 + b_1 + s(b_1-a_1)\big) + 2x_2 - \big(a_2 + b_2 + s(b_2-a_2)\big)\big\} \\ &= \mu\frac{1}{2}\big\{ - c_1 - sd_1 - c_2 - sd_2\big\} = -\frac{1}{2}\big\{c_1 + c_2 + s(d_1+d_2)\big\}\mu,\end{aligned}$$

also

$$\mu\big(\gamma(s)\big) = -2\,\frac{d_1+d_2}{c_1+c_2+s(d_1+d_2)}.$$

Wir setzen nun die neu gewonnenen Erkenntnisse in die Formel für das adjungierte Doppelschichtpotential ein:

$$\begin{aligned}(K'\psi_h|_{[a,b]})(x) &= -\frac{1}{2\pi}\frac{|b-a|}{2}\sum_{\ell=0}^{M}\lambda_\ell\int_{-1}^{1} s^\ell\,\frac{\langle x-\gamma(s), \nu_x\rangle}{|x-\gamma(s)|^2}\,ds \\ &= -\frac{1}{2\pi}\frac{|d|}{2}\sum_{\ell=0}^{M}\lambda_\ell\int_{-1}^{1} s^\ell\Big(-2\frac{\langle d, \nu_x\rangle}{|d|^2}\frac{d_1+d_2}{c_1+c_2+s(d_1+d_2)}\Big)\,ds \\ &= \frac{\langle d, \nu_x\rangle}{2\pi|d|}\sum_{\ell=0}^{M}\lambda_\ell\int_{-1}^{1} s^\ell\,\frac{d_1+d_2}{c_1+c_2+s(d_1+d_2)}\,ds.\end{aligned}$$

Wir betrachten nun folgendes Integral

$$J_n = \int_{-1}^{1} \frac{bs^n}{a+bs}\,ds,$$

das sich mit $t = a + bs$ wie folgt schreibt

$$\begin{aligned} J_n &= \frac{1}{b}\int_{a-b}^{a+b} \frac{b\,(\frac{(t-a)}{b})^n}{t}\,dt \\ &= \frac{1}{b^n}\int_{a-b}^{a+b} \frac{(t-a)^n}{t}\,dt. \end{aligned}$$

Unter Verwendung des binomischen Lehrsatzes ergibt das

$$\begin{aligned} J_n &= \frac{1}{b^n}\int_{a-b}^{a+b} t^{-1}\sum_{i=0}^{n}\binom{n}{i}t^{n-i}(-a)^i\,dt \\ &= \frac{1}{b^n}\sum_{i=0}^{n}\binom{n}{i}(-a)^i\int_{a-b}^{a+b} t^{n-i-1}\,dt \\ &= \Big(-\frac{a}{b}\Big)^n\int_{a-b}^{a+b} t^{-1}\,dt + \frac{1}{b^n}\sum_{i=0}^{n-1}\binom{n}{i}(-a)^i\int_{a-b}^{a+b} t^{n-i-1}\,dt \\ &= \Big(-\frac{a}{b}\Big)^n\Big[\log|t|\Big]_{t=a-b}^{t=a+b} + \frac{1}{b^n}\sum_{i=0}^{n-1}\binom{n}{i}(-a)^i\Big[\frac{t^{n-i}}{n-i}\Big]_{t=a-b}^{t=a+b} \\ &= \Big(-\frac{a}{b}\Big)^n\log\frac{|a+b|}{|a-b|} + \frac{1}{b^n}\sum_{i=0}^{n-1}\binom{n}{i}(-a)^i\frac{(a+b)^{n-i}-(a-b)^{n-i}}{n-i}. \end{aligned} \tag{5.39}$$

Wir wenden nun Formel (5.39) an und erhalten für das adjungierte Doppelschichtpotential

$$\begin{aligned} &(K'\psi_h|_{[a,b]})(x) \\ &\quad= \frac{\langle d,\nu_x\rangle}{2\pi|d|}\sum_{\ell=0}^{M}\lambda_\ell\int_{-1}^{1} s^\ell\frac{d_1+d_2}{c_1+c_2+s(d_1+d_2)}\,ds \\ &\quad= \frac{\langle d,\nu_x\rangle}{2\pi|d|}\sum_{\ell=0}^{M}\lambda_\ell\Bigg[\Big(-\frac{c_1+c_2}{d_1+d_2}\Big)^\ell\log\frac{(c_1+d_1+c_2+d_2)}{(c_1-d_1+c_2-d_2)} \\ &\qquad+\frac{1}{(d_1+d_2)^\ell}\sum_{i=0}^{\ell-1}\binom{\ell}{i}(-c_1-c_2)^i\frac{(c_1+d_1+c_2+d_2)^{\ell-i}-(c_1-d_1+c_2-d_2)^{\ell-i}}{\ell-i}\Bigg] \\ &\quad\overset{!}{=} \frac{\langle d,\nu_x\rangle}{2\pi|d|}\sum_{\ell=0}^{M}\lambda_\ell\Bigg[\Big(-\frac{c_1+c_2}{d_1+d_2}\Big)^\ell\log\frac{(b_1+b_2)-(x_1+x_2)}{(a_1+a_2)-(x_1+x_2)} \\ &\qquad+\frac{1}{(d_1+d_2)^\ell}\sum_{i=0}^{\ell-1}\binom{\ell}{i}(-c_1-c_2)^i\cdot \\ &\qquad\frac{(2(b_1+b_2)-2(x_1+x_2))^{\ell-i}-(2(a_1+a_2)-2(x_1+x_2))^{\ell-i}}{\ell-i}\Bigg] \end{aligned}$$

wegen

$$c_j \pm d_j = a_j + b_j - 2x_j \pm (b_j - a_j) = \begin{cases} 2b_j - 2x_j & = 2(b_j - x_j) \\ 2a_j - 2x_j & = 2(a_j - x_j). \end{cases}$$

Nun wenden wir uns wieder dem Integral

$$\int_{[\mathfrak{a},\mathfrak{b}]} g(x)\big(K'\psi_h|_{[a,b]}\big)(x)\; ds_x$$

zu. Hier könnten wegen des logarithmischen Terms im adjungierten Doppelschichtpotential Probleme auftreten. Daher unterscheiden wir folgende Fälle:

**1. Fall:** $\mathfrak{b} \neq a$ und $b \neq \mathfrak{a}$.
Das bedeutet, dass sich die Intervalle $[a,b]$ und $[\mathfrak{a},\mathfrak{b}]$ nicht berühren. In diesem Fall treten keine Probleme auf, da die Integrationsvariable von $\mathfrak{a}$ und $\mathfrak{b}$ weg ist. Man kann also eine Gauss-Quadratur verwenden.

**2. Fall:** $\mathfrak{b} = a$ oder $b = \mathfrak{a}$.
**2.1. Fall:** $D = 0$.
Auch hier treten keine Probleme auf, da der Integrand 0 ist.

**2.2. Fall:** $D > 0$.
Zu berechnen ist in diesem Fall

$$\int_{[\mathfrak{a},\mathfrak{b}]} \big(K'\psi_h|_{[a,b]}\big)(x)\; ds_x = -\frac{|d|}{2\pi}\sum_{\ell=0}^{M}\lambda_\ell \int_{[\mathfrak{a},\mathfrak{b}]} g(x)\Big(\langle d,\nu_x\rangle G(\ell+1,d,-c) - \langle c,\nu_x\rangle G(\ell,d,-c)\Big)\; ds_x.$$

Für $k = 0$ berechnet man $\int g(x)G(k,\cdot,\cdot)\; ds_x$ ohne Problem. Für $k \geq 1$ treten allerdings Terme der Form

$$\int_{[\mathfrak{a},\mathfrak{b}]} g(x)|c+d|\; ds_x = \int_{[\mathfrak{a},\mathfrak{b}]} g(x)\log|2(b-x)|ds_x$$

und

$$\int_{[\mathfrak{a},\mathfrak{b}]} g(x)|c-d|\; ds_x = \int_{[\mathfrak{a},\mathfrak{b}]} g(x)\log|2(a-x)|ds_x$$

auf.

**2.2.1. Fall:** $a = \mathfrak{b}$.
Der problematische Term ist hier

$$\int_{[\mathfrak{a},\mathfrak{b}]} g(x)\log|x-a|ds_x.$$

Wir betrachten die Parametrisierung $\hat\gamma : [0,1] \to [\mathfrak{a},\mathfrak{b}], s \mapsto \mathfrak{a} + s(\mathfrak{b}-\mathfrak{a})$ und erhalten

$$\begin{aligned}
\int_{[\mathfrak{a},\mathfrak{b}]} g(x)\log|x-a|\, ds_x &= |\mathfrak{b}-\mathfrak{a}|\int_0^1 g\big(\hat\gamma(s)\big)\log|\mathfrak{a}+s(\mathfrak{b}-\mathfrak{a})-a|\, ds \\
&= |\mathfrak{b}-\mathfrak{a}|\int_0^1 g\big(\hat\gamma(s)\big)\log|(s-1)(\mathfrak{b}-\mathfrak{a})|\, ds \\
&= |\mathfrak{b}-\mathfrak{a}|\int_0^1 g\big(\hat\gamma(s)\big)\big(\log(1-s)+\log|\mathfrak{b}-\mathfrak{a}|\big)\, ds \\
&= |\mathfrak{b}-\mathfrak{a}|\int_0^1 g\big(\hat\gamma(s)\big)\log(1-s)\, ds + |\mathfrak{b}-\mathfrak{a}|\int_0^1 g\big(\hat\gamma(s)\big)\log|\mathfrak{b}-\mathfrak{a}|\, ds \\
&= |\mathfrak{b}-\mathfrak{a}|\int_0^1 g\big(\hat\gamma(1-t)\big)\log t\, dt + \int_{[\mathfrak{a},\mathfrak{b}]} g(x)\log|\mathfrak{b}-\mathfrak{a}|\, ds_x.
\end{aligned}$$

Das Integral

$$\int_0^1 g\big(\hat{\gamma}(1-t)\big)\log t\,dt$$

kann mit einer Gauss-Quadratur zur Gewichtsfunktion $\omega(t)=\log(t)$ berechnet werden. Solche Quadraturformeln sind beispielsweise in Büchern wie Secrest-Stroud oder Abramowitz tabelliert. Alle übrigen Integrale können mittels der üblichen Gauss-Quadratur approximiert werden.

**2.2.2. Fall:** $b=\mathfrak{a}$.
Analog zum Fall 2.2.1. kann man das Probleme machende Integral umformen

$$\int_{[\mathfrak{a},\mathfrak{b}]} g(x)\log|x-b|\,ds_x = |\mathfrak{b}-\mathfrak{a}|\int_0^1 g\big(\hat{\gamma}(s)\big)\log s\,ds + \int_{[\mathfrak{a},\mathfrak{b}]} g(x)\log|\mathfrak{b}-\mathfrak{a}|\,ds_x$$

und alle auftretenden Integrale wie oben beschrieben berechnen.

## 5.5 Fehlerschätzer

Für die in den Kapiteln 3 und 4 vorgestellten Fehlerschätzer werden hier mögliche Berechnungen gezeigt und die zugehörigen Matlab-Codes angeführt.

### 5.5.1 $h-h/2$-Schätzer

Wie in Algorithmus 4.6 vorgestellt benötigt man für die Berechnung der $h-h/2$-Fehlerschätzer $\mu_\ell$ und $\eta_\ell$ aus Abschnitt 4.2 auch die Lösung des Gleichungssystems der feineren Partition $\widehat{\mathcal{T}}_\ell$. Die Steifigkeitsmatrix $\widehat{A}\in\mathbb{R}^{NxN}$ und rechte Seite $\widehat{b}\in\mathbb{R}^N$ werden berechnet durch

$$\widehat{A}_{jk}=\langle V\chi_{\widehat{T}_j},\chi_{\widehat{T}_k}\rangle_\Gamma \qquad \widehat{b}_j=\langle(K+\tfrac{1}{2}),\chi_{\widehat{T}_j}\rangle_\Gamma \quad \text{für alle } \widehat{T}_j,\widehat{T}_k\in\mathcal{T}_\ell.$$

Durch Lösen des linearen Gleichungssystems $\widehat{A}\widehat{x}=\widehat{b}$ erhält man den Lösungsvektor $\widehat{x}\in\mathbb{R}^N$. Bei uniformer Verfeinerung des Netzes $\mathcal{T}_\ell$ entstehen aus einem Element $T$ der groben Partition zwei Elemente $T_1$ und $T_2$ in $\widehat{\mathcal{T}}_\ell$, beim Übergang von $\widehat{\mathcal{T}}_\ell$ zu $\mathcal{T}_\ell$ wird durch die Nebenbedingungen eine Gleichheit der Lösungen auf den Elementen $T_1$ und $T_2$ erzwungen, und $x$ und $\widehat{x}$ haben diesselbe Länge.
Daher kann $\eta_\ell$ berechnet werden durch

$$\eta_\ell^2=\big(\widehat{x}-x\big)\cdot\widehat{A}\big(\widehat{x}-x\big).$$

Dieser Fehlerschätzer kann als Indikator für den Abbruch des Algorithmus herangezogen werden, doch für eine Steuerung des Verfeinerungsalgorithmus ist er nicht geeignet, da er nicht lokal ist. Für diesen Zweck verwendet man daher $\mu_\ell$, da er den Fehler in der $L^2$-Norm schätzt. Für $\mu_\ell$ berechne man

$$\mu_\ell^2=\sum_{T\in\mathcal{T}_\ell}\mu_\ell(T)^2 \quad \text{mit} \quad \mu_\ell(T)^2=h_\ell\big(\widehat{x}-x\big)^2.$$

Auf analoge Weise kann man auch $\widetilde{\mu}_\ell$ und $\widetilde{\eta}_\ell$ aus Kapitel 4.4 ermitteln.

### 5.5.2 Residualschätzer

Ein Nachteil des residualen Fehlerschätzers ist seine im Vergleich zu den $h-h/2$-Fehlerschätzern sehr aufwändige Implementierung, auf die wir nun näher eingehen wollen:

Wir betrachten für $T=[a,b]$ die Parametrisierung aus Gleichung (5.17).

Der Residualschätzer lässt sich damit darstellen als

$$\varrho_\ell^2 = \left\| h_\ell^{1/2} R_\ell' \right\|^2_{L^2(\Gamma)} = \sum_{T\in\mathcal{T}_\ell} h_T \left\| R_\ell' \right\|^2_{L^2(T)} = \sum_{T\in\mathcal{T}_\ell} h_T \int_T \big(R_\ell'(x)\big)^2 \, dx.$$

Für ein Element $T=[\mathfrak{a},\mathfrak{b}]$ erhält man also den lokalen Beitrag

$$\varrho_\ell(T)^2 = |\mathfrak{b}-\mathfrak{a}| \frac{|\mathfrak{b}-\mathfrak{a}|}{2} \int_{-1}^1 \Big( R_\ell'\big(\gamma_T(t)\big)\Big)^2 dt.$$

Approximation des Integrals mittels Gaussquadratur der Ordnung $p$ liefert

$$\varrho_\ell(T)^2 \approx \frac{|\mathfrak{b}-\mathfrak{a}|^2}{2} \sum_{m=1}^p w_m \Big( R_\ell'\big(\gamma_T(s_m)\big)\Big)^2.$$

Man möchte also $R_\ell'$ für gewisse, fixe $\overline{s}_m = \gamma_T(s_m)$ berechnen. Dazu wird zunächst $R_\ell(x)$ an der Stelle $\overline{s}_m$ ausgewertet:

$$R_\ell(x) = \big(K+\tfrac12\big) g_\ell(x) - V\widetilde{\Phi}_\ell(x)$$

Für das Doppelschichtpotential erhält man mit $T=[a,b]$ Folgendes:

$$\begin{aligned}
&\big(K+\tfrac12\big) g_\ell(x) \\
&\quad = -\frac{1}{2\pi} \int_\Gamma \frac{\langle y-x, n_y\rangle}{|x-y|^2} g_\ell(y)\, ds_y + \frac12 g_\ell(x) \\
&\quad = \sum_{T\in\mathcal{T}_\ell} \bigg( -\frac{1}{2\pi} \int_T \frac{\langle y-x, n_y\rangle}{|x-y|^2} g_\ell(y)\, ds_y \bigg) + \frac12 g_\ell(x) \\
&\quad = \sum_{T\in\mathcal{T}_\ell} -\frac{|b-a|}{4\pi} \int_{-1}^1 \frac{\langle \frac12\big(a+b+t(b-a)\big)-x, n_y\rangle}{|x-\frac12\big(a+b+t(b-a)\big)|^2} g_\ell\Big(\tfrac12\big(a+b+t(b-a)\big)\Big)\, dt + \frac12 g_\ell(x).
\end{aligned}$$

$g_\ell(x) = I_\ell g(x)$ ist die nodale Interpolation von $g(x)$, also stückweise affin. Deshalb gilt

$$\frac{g(\mathfrak{b})-g(\mathfrak{a})}{|\mathfrak{b}-\mathfrak{a}|} = \frac{g_\ell(x)-g(\mathfrak{a})}{|x-\mathfrak{a}|}$$

und nach Umformen

$$g_\ell(x) = \frac{|x-\mathfrak{a}|}{|\mathfrak{b}-\mathfrak{a}|}\Big(g(\mathfrak{b})-g(\mathfrak{a})\Big) + g(\mathfrak{a})$$

und

$$g_\ell\Big(\tfrac12\big(a+b+t(b-a)\big)\Big) = \frac12\Big(g(a)+g(b)+t\big(g(b)-g(a)\big)\Big).$$

Wir setzen nun

$$\begin{aligned} p &:= \tfrac{1}{2}(b-a) \\ q &:= \tfrac{1}{2}(a+b) - x \\ \hat{p} &:= g(b) - g(a) \\ \hat{q} &:= g(a) + g(b) \end{aligned}$$

und erhalten so mit $n_y = n$ auf $[a,b]$.

$$\begin{aligned} &\big(K + \tfrac{1}{2}\big) g_\ell(x) \\ &\quad = \sum_{T \in \mathcal{T}_\ell} -\frac{|b-a|}{4\pi} \frac{1}{2} \int_{-1}^{1} \frac{\langle p, n\rangle t + \langle q, n\rangle}{|p|^2 t^2 + 2\langle p, q\rangle t + |q|^2} \cdot \big(\hat{p}t + \hat{q}\big)\, dt + \frac{1}{2}\frac{|x-\mathfrak{a}|}{|\mathfrak{b}-\mathfrak{a}|}\big(g(\mathfrak{b}) - g(\mathfrak{a})\big) + g(\mathfrak{a}). \end{aligned}$$

Zum Lösen des Integrals wird die Funktion $G$ aus Kapitel 5.2. verwendet

$$\begin{aligned} &\big(K + \tfrac{1}{2}\big) g_\ell(x) \\ &\quad = \sum_{T \in \mathcal{T}_\ell} -\frac{|b-a|}{8\pi}\Big(\hat{p}\langle p, n\rangle G(2,p,q) + \big(\hat{q}\langle p, n\rangle + \hat{p}\langle q, n\rangle\big) G(1,p,q) + \hat{q}\langle q, n\rangle G(0,p,q)\Big) \\ &\qquad + \frac{1}{2}\frac{|x-\mathfrak{a}|}{|\mathfrak{b}-\mathfrak{a}|}\big(g(\mathfrak{b}) - g(\mathfrak{a})\big) + g(\mathfrak{a}). \end{aligned}$$

$n$ ist der äußere Normalenvektoren in $y$-Richtung, d.h. es gilt $n \perp (b-a)$ und somit $n \perp p$. Damit erhält man den vereinfachten Ausdruck

$$\big(K + \tfrac{1}{2}\big) g_\ell(x) = \sum_{T \in \mathcal{T}_\ell} -\frac{|b-a|}{8\pi}\langle q, n\rangle\big(\hat{p}\, G(1,p,q) + \hat{q}\, G(0,p,q)\big) + \frac{1}{2}\frac{|x-\mathfrak{a}|}{|\mathfrak{b}-\mathfrak{a}|}\big(g(\mathfrak{b}) - g(\mathfrak{a})\big) + g(\mathfrak{a}).$$

Für das Einfachschichtpotential berechnet man für $T = [a,b]$ mit der Parametrisierung $\gamma$ aus Gleichung (5.17)

$$\begin{aligned} V\widetilde{\Phi}_\ell(x) &= -\frac{1}{2\pi}\int_\Gamma \widetilde{\Phi}_\ell(y) \log|x-y|\, ds_y \\ &= -\frac{1}{2\pi}\sum_{T \in \mathcal{T}_\ell} \frac{1}{2}\int_T \widetilde{\Phi}_\ell(y) \log|x-y|^2\, ds_y \\ &= -\frac{1}{2\pi}\sum_{T \in \mathcal{T}_\ell} \frac{|b-a|}{4}\int_{-1}^{1} \widetilde{\Phi}_\ell\Big(\tfrac{1}{2}\big(a+b+t(b-a)\big)\Big) \log\Big|x - \Big(\tfrac{1}{2}\big(a+b+t(b-a)\big)\Big)\Big|^2\, dt. \end{aligned}$$

Es sei hier wiederum

$$\begin{aligned} p &:= \tfrac{1}{2}(b-a) \\ q &:= \tfrac{1}{2}(a+b) - x \\ \widetilde{\Phi}_\ell &:= \sum_{T \in \mathcal{T}_\ell} x_T \chi_T. \end{aligned}$$

Daraus folgt mit der Funktion $L$ aus Kapitel 5.2

$$\begin{aligned} V\widetilde{\Phi}_\ell(x) &= -\frac{1}{2\pi}\sum_{T \in \mathcal{T}_\ell} \frac{|b-a|}{4} x_T \int_{-1}^{1} \log\left(t^2|p|^2 + 2t\langle p, q\rangle + |q|^2\right) dt \\ &= -\frac{1}{8\pi}\sum_{T \in \mathcal{T}_\ell} |b-a|\, x_T\, L(0,p,q). \end{aligned}$$

Man hat also einen Vektor $\bar{s}$ der Länge $p$ und einen Vektor $R_\ell(\bar{s})$ derselben Länge, der das Residuum an den Stellen $\bar{s}_m$ enthält.
Um das Verfahren der dividierten Differenzen anwenden zu können, das ein durch die vorher bestimmten Punkte $\big(\bar{s}_m, R_\ell(\bar{s}_m)\big)$ gehendes Polynom berechnet, müssen die zu den auf einer Geraden liegenden Punkten $\bar{s}_m$ zugehörigen Parameter $t_m$ gefunden werden. Sei also $\gamma_T^{-1} : [a,b] \to [-1,1]$, $x \mapsto \gamma_T^{-1}(x) = t$ die Umkehrabbildung von $\gamma_T$, so gilt

$$2\bar{s_\ell} - a - b = t_\ell(b-a).$$

Die polynomiale Approximation $R_\ell \circ \gamma_T \approx r_\ell \in \mathcal{P}^p([-1,1])$ hat dann folgende Darstellung in der Newton-Basis

$$r_\ell(s) = \sum_{j=1}^{p} \lambda_j \prod_{k=1}^{j-1}(s - t_k),$$

wobei die Koeffizienten $\lambda_j$ mittels des Verfahrens der dividierten Differenzen bestimmt werden. Nun kann auch die Ableitung dieses Polynoms berechnet werden

$$\begin{aligned} r'_\ell(s) &= \sum_{j=2}^{p} \lambda_j \Big( \sum_{i=1}^{j-1} \prod_{\substack{k=1 \\ k\neq i}}^{j-1} (s - t_k) \Big) \\ &= \sum_{j=1}^{p-1} \lambda_{j+1} \Big( \sum_{i=1}^{j} \prod_{\substack{k=1 \\ k\neq i}}^{j} (s - t_k) \Big). \end{aligned}$$

Zusammenfassend wird also zuerst $R_\ell(x)$ an den Stellen $\bar{s}_m = \gamma(s_m)$ berechnet und ein durch diese Punkte gehendes Interpolationspolynom bestimmt, dessen Ableitung errechnet werden kann. Insgesamt erhält man so mit $\big(R_\ell \circ \gamma_T(s)\big)' \approx r'_\ell(s)$

$$R'_\ell \approx \frac{2}{|\mathfrak{b} - \mathfrak{a}|} r'_\ell(s)$$

und damit für den Fehlerschätzer

$$\varrho_\ell^2 = \sum_{T \in \mathcal{T}_\ell} \varrho_{\ell,T}^2$$

mit

$$\varrho_{\ell,T}^2 = \frac{|\mathfrak{b} - \mathfrak{a}|^2}{2} \sum_{m=1}^{p} w_m \Big( \frac{2}{|\mathfrak{b} - \mathfrak{a}|} r'_\ell(s_m) \Big)^2 = 2 \sum_{m=1}^{p} w_m \big( r'_\ell(s_m) \big)^2.$$

Dies ist der Code zur Berechnung des Residualfehlerschätzers $\varrho_T^2$:

```
function val = residual_error_estimator(elements,coordinates,x_tilde,p)

  % input: 'elements', 'coordinates', discrete solution 'x_tilde',
  %        order for gauss quadrature 'p'

  N = size(elements,1);
```

```
  [nodes, weights] = gauss(p);

  % Rh is a vector of length p containing the residual
  % Rh(x) = (K+1/2)g_\ell(x)-V\Phi_tilde(x)
  Rh = zeros(p,1);

  % returnvalue, the error estimator is calculated for every element
  val = zeros(N,1);

  for m=1:N   % loop runs through the elements
    tmp = 0;
    aa = coordinates(elements(m,1),:);
    bb = coordinates(elements(m,2),:);
    bbmaa = bb-aa;
    trafo_nodes = 0.5*(ones(p,1)*(aa+bb)+nodes*(bbmaa));

    % the residual is evaluated at the parameterized gauss nodes
    for ell=1:p %loop runs through the gauss nodes
        x = trafo_nodes(ell,:);
        Rh(ell) = residual(elements,coordinates,x_tilde,x,aa,bb);
    end

    lambda = div_diff(nodes,Rh);

    for ell = 1:p
       rh_prime = 0;
       for j = 1:p-1
          sum = 0;
          for i=1:j
             product = 1;
             for k=[1:i-1,i+1:j]
                product = product * (nodes(ell)-nodes(k));
             end
             sum = sum + product;
          end
          rh_prime = rh_prime+lambda(j+1)*sum;
        end
        tmp = tmp+weights(ell)*rh_prime^2;
    end

    val(m) = val(m) + 2*tmp;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Rh = residual(elements,coordinates,x_tilde,x,aa,bb)

  % input: 'elements', 'coordinates', discrete solution 'x_tilde',
```

```
  %        evaluation point for residual 'x', coordinates of the
  %        element, for which residual error is calculated '[aa,bb]'

  N = size(elements,1);
  Rh = 0;

  for i=1:N
      a = coordinates(elements(i,1),:);
      b = coordinates(elements(i,2),:);

      % Kg_\ell
      Rh = Rh + dlp(a,b,x);

      % V\phi
      Rh = Rh - x_tilde(i)*slp(a,b,x);
  end

  % 1/2 g_\ell
  Rh = Rh+0.5*( norm(x-aa)/norm(bb-aa)*(g(bb)-g(aa) )+g(aa));

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function K_a_b = dlp(a,b,x)

  q = 0.5*(a+b)-x;
  p = 0.5*(b-a);
  gp = g(b)-g(a);
  gq = g(a)+g(b);
  bma = b-a;
  norm_bma = norm(bma);
  n = [bma(2), -bma(1)]/norm_bma;
  sc_q_n = q(1)*n(1)+q(2)*n(2);

  K_a_b = -norm_bma*sc_q_n* ( gq*G(0,p,q) + gp*G(1,p,q) ) /(8*pi);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function V_a_b = slp(a,b,x)

  q = 0.5*(a+b)-x;
  p = 0.5*(b-a);

  V_a_b = -norm(b-a)*L(0,p,q) / (8*pi);

end
```

### 5.5.3 Datenfehlerschätzer

Da die Ableitung von $g$ im Allgemeinen nicht analytisch berechnet werden kann, wird der Datenfehlerschätzer $\zeta_\ell$ approximativ ermittelt. Dazu wird die Funktion $g$ durch eine polynomiale Funktion $\widetilde{g}$ ersetzt, um die inverse Abschätzung aus Satz 3.4 anwenden zu können:

$$\left\|h_\ell^{1/2}(1-\Pi_\ell)g'\right\|^2_{L^2(\Gamma)} \approx \left\|h_\ell^{1/2}\big(\widetilde{g}\,' - \Pi_\ell g'\big)\right\|^2_{L^2(\Gamma)} = \left\|h_\ell^{1/2}\big(\widetilde{g}\,' - g_\ell'\big)\right\|^2_{L^2(\Gamma)} \lesssim \left\|h_\ell^{-1/2}\big(\widetilde{g} - g_\ell\big)\right\|^2_{L^2(\Gamma)}.$$

Mit $T = [a,b]$ erhält man dann

$$\begin{aligned}\left\|h_\ell^{1/2}(1-\Pi_\ell)g'\right\|^2_{L^2(\Gamma)} &= \sum_{T\in\mathcal{T}_\ell} h_T^{-1}\left\|\widetilde{g} - g_\ell\right\|^2_{L^2(T)} \\ &= \sum_{T\in\mathcal{T}_\ell} h_T^{-1}\int_T \big(\widetilde{g} - g_\ell\big)^2(t)\,dt \\ &= \sum_{T\in\mathcal{T}_\ell} \frac{1}{|b-a|}\frac{|b-a|}{2}\int_{-1}^1 \big(\widetilde{g} - g_\ell\big)^2\Big(\tfrac{1}{2}\big(a+b+t(b-a)\big)\Big)\,dt.\end{aligned}$$

Das Integral wird mittels Gaussquadratur berechnet

$$\left\|h_\ell^{1/2}(1-\Pi_\ell)g'\right\|^2_{L^2(\Gamma)} = \frac{1}{2}\sum_{T\in\mathcal{T}_\ell}\sum_{i=1}^p w_i\left[\widetilde{g}\Big(\tfrac{1}{2}\big(a+b+t_i(b-a)\big)\Big) - g_\ell\Big(\tfrac{1}{2}\big(a+b+t_i(b-a)\big)\Big)\right]^2.$$

Die Gaussquadratur ist eine Interpolationsquadratur also gilt mit $t\in[-1;1]$

$$\widetilde{g}(a,b,t) = \widetilde{g}\Big(\tfrac{1}{2}\big(a+b+t(b-a)\big)\Big) = g\Big(\tfrac{1}{2}\big(a+b+t(b-a)\big)\Big).$$

Außerdem ist $g_\ell = I_\ell g$ die nodale Interpolation von $g$, also ist $g_\ell|_T$ linear und es gilt

$$g_\ell(a,b,t) = g_\ell\Big(\tfrac{1}{2}\big(a+b+t(b-a)\big)\Big) = \tfrac{1}{2}\Big(g(a)+g(b)+t\big(g(b)-g(a)\big)\Big).$$

Insgesamt muss also für alle $T\in\mathcal{T}_\ell$

$$\begin{aligned}\zeta_\ell(T)^2 &= \left\|h_\ell^{1/2}(1-\Pi_\ell)g'\right\|^2_{L^2(T)} \\ &= \frac{1}{2}\sum_{j=1}^p w_j\left[g\Big(\tfrac{1}{2}\big(a+b+t_j(b-a)\big)\Big) - \tfrac{1}{2}\Big(g(a)+g(b)+t_j\big(g(b)-g(a)\big)\Big)\right]^2\end{aligned}$$

berechnet werden. Durch Aufsummieren erhält man schlussendlich

$$\zeta_\ell^2 = \sum_{T\in\mathcal{T}_\ell}\zeta_\ell(T)^2.$$

Dies ist der Code zur Berechnung des Datenfehlerschätzers $\zeta_\ell^2$:

```
function val = zeta_ell(coordinates,elements)

  % Number of quadrature points
```

```
NOQP = 2;
[gauss_weights,gauss_nodes] = gauss(NOQP);

for i = 1:length(elements)
    a = coordinates(elements(i,1),:);
    b = coordinates(elements(i,2),:);
    ga = g(a);
    gb = g(b);
    g_ell= zeros(NOQP,1);
    g_tilde = zeros(NOQP,1);
    for j=1:NOQP
        g_ell(j) = 0.5* (ga+gb+gauss_nodes(j)*(gb-ga) );
        g_tilde(j) = g( 0.5*(a+b+gauss_nodes(j)*(b-a)) );
        val(i) = val(i) + gauss_weights(j)* ( g_tilde(j)-g_ell(j) )^2;
    end
    val(i) = val(i)*0.5;
 end
```

# Kapitel 6

# Änderungen im Fall $d = 3$

In diesem Abschnitt wird beschrieben, was sich für den Fall $d = 3$ verändert. Die übrigen Resultate können wie in den vorigen Kapiteln beschrieben übernommen werden.

## 6.1 Die Randelementmethode

**Definition 2.1** *Die Fundamentallösung des Laplaceoperators ist für $z \in \mathbb{R}^3$ durch*

$$G(z) := \frac{1}{4\pi}\frac{1}{|z|}$$

*gegeben.*

Das Einfachschichtpotential hat für $d = 3$ andere Eigenschaften:

Die Elliptizität von $V$ erhält man auch ohne einer Voraussetzung an die logarithmische Kapazität bzw. an das Gebiet $\Omega$:

**Satz 2.24** *Für das Einfachschichtpotential $V : H^{-1/2+s}(\Gamma) \to H^{1/2+s}(\Gamma)$, $|s| \leq \frac{1}{2}$ gilt: $V$ ist $H^{-1/2}(\Gamma)$-elliptisch, d.h es gibt eine so genannte Elliptizitätskonstante $\gamma_V > 0$ mit*

$$\langle V\phi, \phi\rangle \geq \gamma_V \|\phi\|^2_{H^{-1/2}(\Gamma)} \quad \text{für alle } \phi \in H^{-1/2}(\Gamma).$$

*Somit ist $V$ stets invertierbar.* □

Die beiden folgenden Sätze gelten daher ohne zusätzliche Voraussetzungen:

**Satz 2.26** *Für das Newtonpotential $(N_1 f)(x)$, $x \in \Gamma$, gilt stets die Darstellung*

$$(N_1 f)(x) = \Big(-\frac{1}{2} + K'\Big)V^{-1}(N_0 f)(x).$$

□

**Satz 2.28** *Für das Doppelschichtpotential $K : H^{1/2+s}(\Gamma) \to H^{1/2+s}(\Gamma)$, $|s| \leq \frac{1}{2}$ gilt: $K + \frac{1}{2}$ ist eine Kontraktion bezüglich der auf $H^{1/2}(\Gamma)$ äquivalenten Norm $\|g\|^2_{V^{-1}} := \langle V^{-1}g, g\rangle$.* □

Um eine Triangulierung $\mathcal{T}_\ell$ definieren zu können, wurde vorausgesetzt, dass jedes Element $T \in \mathcal{T}_\ell$ das Bild eines Referenzelements $\widehat{T}$ unter einer regulären Abbildung ist. Die Definition einer regulären Abbildung ließ sich aber für den zweidimensionalen Fall vereinfachen. Allgemein gilt:

**Definition 2.31** *Unter einer regulären Abbildung $\gamma_T$ versteht man eine Abbildung, für die die zugehörige Jacobimatrix $J_T$ die Bedingung*

$$0 < \lambda_{\min} \leq \inf_{\widehat{x}\in\widehat{T}} \inf_{\substack{v\in\mathbb{R}^2\\ \|v\|=1}} \langle J_T(\widehat{x})v, J_T(\widehat{x})v\rangle \leq \sup_{\widehat{x}\in\widehat{T}} \sup_{\substack{v\in\mathbb{R}^2\\ \|v\|=1}} \langle J_T(\widehat{x})v, J_T(\widehat{x})v\rangle \leq \lambda_{\max} < \infty$$

*erfüllt.*

Die Regularität einer Triangulierung bedarf ebenfalls einer allgemeineren Beschreibung, da bei Randelementen im $\mathbb{R}^3$ — Dreiecke oder Vierecke — mehrere Möglichkeiten einer Berührung zweier Nachbarn bestehen.

**Definition 2.33** *Man nennt eine Partition $\mathcal{T}_\ell$ regulär, falls:*

*(i) Der Schnittpunkt des Abschlusses zweier verschiedener Elemente $T, T' \in \mathcal{T}_\ell$ entweder leer, ein gemeinsamer Punkt oder eine gemeinsame Kante ist.*

*(ii) Liegt im Schnitt eine gemeinsame Kante $e = \overline{T} \cap \overline{T'}$ vor, so stimmen die Parametrisierungen entlang der Kante überein. Das heißt*

$$\gamma_T|_{\widehat{e}} = \gamma_{T'} \circ \beta_{T,T'}|_{\widehat{e}}$$

*ist mit einer geeigneten affinen Bijektion $\beta_{T,T'} : \widehat{T} \to \widehat{T}$ erfüllt.*

## 6.2 Adaptiver Algorithmus mit residualem Fehlerschätzer

Für den Beweis der Zuverlässigkeit des Residualschätzers werden Satz 3.2, Korollar 3.3 und Satz 3.4 verwendet. Da diese nur für $d = 2$ gültig sind, ist der Beweis in dieser Form nur im $\mathbb{R}^2$ möglich. Trotzdem lässt sich die Zuverlässigkeit auch für den dreidimensionalen Fall zeigen — siehe Carstensen, Maischak, Stephan 2001 [7].

Auch die Konstruktion des Datenfehlerschätzers stützt sich auf Satz 3.2 und ist daher speziell auf den Fall $d = 2$ ausgerichtet. Es bleibt an dieser Stelle offen, wie eine geeignete Approximation $g_\ell$ zu wählen ist.

Abschnitt 3.6 zur Konvergenz des adaptiven Algorithmus gilt für das ungestörte Verfahren, d.h $g = g_\ell$, und isotrope Netzverfeinerung unverändert.

## 6.3 Adaptiver Algorithmus mit $h - h/2$- Fehlerschätzer

Sowohl die $h - h/2$-Fehlerschätzer für das ungestörte als auch das gestörte Verfahren werden für $d = 3$ identisch definiert und haben dieselben Eigenschaften (Effizienz, Zuverlässigkeit unter der Saturationsannahme und Äquivalenz). Vergleiche dazu Ferraz-Leite/Praetorius 2007 [12].

# Kapitel 7

# Numerische Experimente

In diesem Abschnitt wird das Verhalten der in Kapitel 3 und 4 eingeführten Fehlerschätzer an Hand von drei Testbeispielen untersucht. Wir gehen dazu vom Laplaceproblem

$$\begin{aligned} \triangle u &= 0 \quad \text{in } \Omega \\ u &= g \quad \text{auf } \Gamma \end{aligned} \tag{7.1}$$

mit gegebenen Dirichletdaten $g$ aus. Dieses Problem ist äquivalent zur Symmschen Integralgleichung $V\phi = \big(K + \frac{1}{2}\big)g$. Die exakte Lösung $\phi$ ist dann die Normalenableitung $\phi = \frac{\partial u}{\partial n}$ von $u$ am Rand $\Gamma$.

Der Fehler in der Energienorm zwischen der kontinuierlichen Lösung $\phi$ des ungestörten Problems und der Approximation $\widetilde{\Phi}_\ell$ erfüllt

$$\left|\!\left|\!\left| \phi - \widetilde{\Phi}_\ell \right|\!\right|\!\right|^2 = \left|\!\left|\!\left| \phi - \Phi_\ell \right|\!\right|\!\right|^2 + \left|\!\left|\!\left| \Phi_\ell - \widetilde{\Phi}_\ell \right|\!\right|\!\right|^2$$

und kann unter Einsatz der Galerkinorthogonalität dargestellt werden als

$$\left|\!\left|\!\left| \phi - \widetilde{\Phi}_\ell \right|\!\right|\!\right|^2 = \left|\!\left|\!\left| \phi \right|\!\right|\!\right|^2 - \left|\!\left|\!\left| \Phi_\ell \right|\!\right|\!\right|^2 + \left|\!\left|\!\left| \Phi_\ell - \widetilde{\Phi}_\ell \right|\!\right|\!\right|^2 = \left|\!\left|\!\left| \phi \right|\!\right|\!\right|^2 - 2\langle\!\langle \Phi_\ell, \widetilde{\Phi}_\ell \rangle\!\rangle + \left|\!\left|\!\left| \widetilde{\Phi}_\ell \right|\!\right|\!\right|^2.$$

Die unbekannte Energienorm $\left|\!\left|\!\left| \phi \right|\!\right|\!\right|$ wird geschätzt. Dazu wird eine Folge von Energien $\left|\!\left|\!\left| \Phi_\ell \right|\!\right|\!\right|^2$ bei uniformer Netzverfeinerung erzeugt und mittels Aitkenschem $\Delta^2$-Verfahren die exakte Energie $\left|\!\left|\!\left| \phi \right|\!\right|\!\right|^2$ extrapoliert.

In Korollar 3.11 wurde die Beschränktheit von $\kappa(\mathcal{T}_\ell)$ bewiesen, unter der Voraussetzung, dass eine erweiterte Markierungsstrategie zur Steuerung des adaptiven Algorithmus verwendet wird. Das soll mit Hilfe von numerischen Experimenten bestätigt werden.

Aufgrund von Rechenfehlern wächst die Konditionszahl der Steifigkeitsmatrix $A$ im Laufe des Verfahrens stark an, sodass ein sinnvolles Lösen des linearen Gleichungssystems $Ax = b$ nicht mehr möglich ist. Aus diesem Grund wird $A$ vorkonditioniert. Eine Verringerung der Konditionszahl wird numerisch untersucht.

## 7.1 Laplaceproblem am Quadrat

Im ersten numerischen Beispiel betrachten wir das Laplaceproblem (7.1) auf dem Quadrat $\Omega = (0, 0.5)^2$ mit der exakten Lösung $u(x, y) = \sinh(2\pi x)\cos(2\pi y)$ und lösen die äquivalente Symmsche

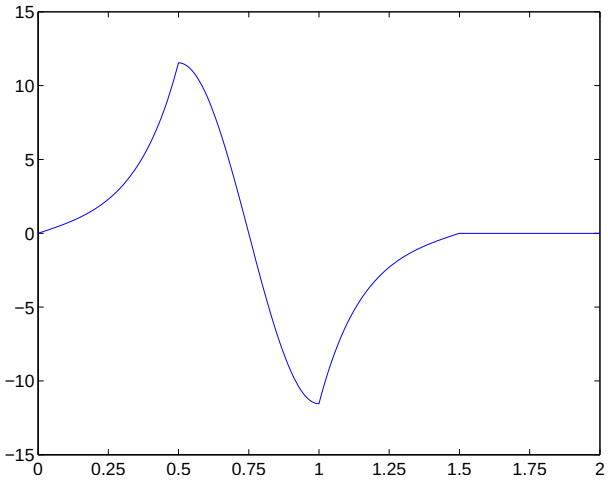


Abbildung 7.1: Exakte Dirichletdaten des Laplaceproblems 7.1 über der Bogenlänge $s \in [0, 2]$, wobei $s \in \{0, 2\}$ gerade dem Nullpunkt $(0, 0) \in \mathbb{R}^2$ entspricht.

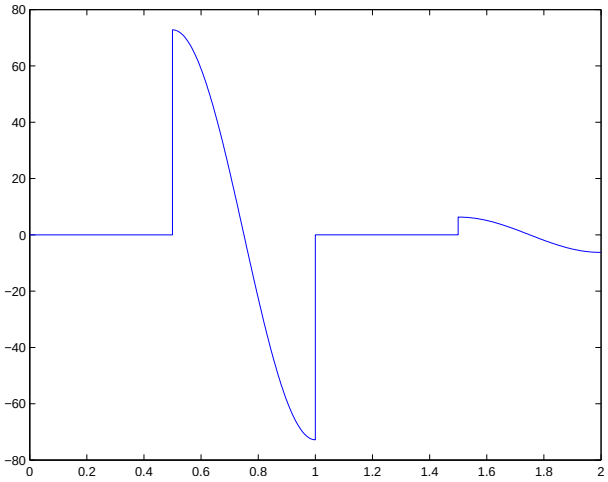


Abbildung 7.2: Exakte Lösung des Laplaceproblems 7.1 über der Bogenlänge $s \in [0, 2]$, wobei $s \in \{0, 2\}$ gerade dem Nullpunkt $(0, 0) \in \mathbb{R}^2$ entspricht.

Integralgleichung $V\phi = \left(K + \frac{1}{2}\right)g$ mit $g = u|_\Gamma$. Die Dirichletdaten $g \in H^{1/2}(\Gamma)$ sind in Abbildung 7.1 über der Bogenlänge $0 \leq s < 2$ dargestellt, wobei $s \in \{0, 2\}$ gerade dem Nullpunkt $(0, 0) \in \mathbb{R}^2$ entspricht.

Die exakte Lösung der Symmschen Integralgleichung $\phi = \frac{\partial u}{\partial n}$ ist dann auf jedem affinen Randstück glatt und lässt sich schreiben in der Form

$$\phi(x, y) = 2\pi \begin{pmatrix} \cosh(2\pi x)\cos(2\pi y) \\ -\sinh(2\pi x)\sin(2\pi y) \end{pmatrix} \cdot n(x, y)$$

mit dem äußeren Normalenvektor $n(x, y)$. Zur Veranschaulichung ist $\phi$ dargestellt in Abbildung 7.2.

Es ist zu beachten, dass es ist nicht möglich ist, das Einheitsquadrat für die Berechnungen heranzuziehen, da für den Durchmesser des Gebiets $\operatorname{diam}(\Omega) < 1$ gelten muss, um die Elliptizität des Einfachschichtpotentials zu garantieren. Das skalierte Quadrat $\Omega$ erfüllt gerade $\operatorname{diam}(\Omega) = \sqrt{2}/2 \approx 0.7071 < 1$ Das verwendete Startnetz $\mathcal{T}_0$ mit $N_0 = 4$ Elementen ist in Abbildung 7.3 zu sehen.

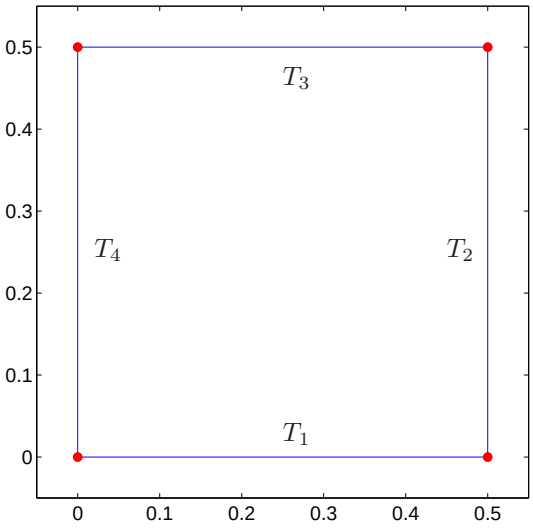


Abbildung 7.3: Startnetz $\mathcal{T}_0$ mit $N_0 = 4$ Elementen des Laplaceproblems 7.1.

Die approximative Berechnung der Energienorm der exakten, kontinuierlichen Lösung mittels Aitkenscher $\triangle^2$-Methode liefert $\left|\!\left|\!\left|\phi\right|\!\right|\!\right|^2 = 81.072404842406$.

Die experimentelle Konvergenzrate $\alpha_\ell$ wird aus dem exakten Fehler $e_\ell$ in der Energienorm und der Anzahl der Elemente $N_\ell$ einer Partition $\mathcal{T}_\ell$ mit dem Ansatz $e_\ell = CN_\ell^{-\alpha}$ und $N_\ell = \mathrm{card}(\mathcal{T}_\ell)$ bestimmt, d.h.

$$\alpha_\ell := -\frac{\log(\frac{e_{\ell-1}}{e_\ell})}{\log(\frac{N_{\ell-1}}{N_\ell})}.$$

Die Energienorm des Fehlers $e_\ell$ ist bekannt.

Zunächst wird die Rechnung mit exakter rechter Seite durchgeführt. Tabelle 7.1 zeigt die Anzahl der Elemente $N_\ell$, den Fehler $e_\ell$ die experimentelle Konvergenzrate $\alpha_\ell$ und die Fehlerschätzer

$$\varrho_\ell = \left\|h_\ell^{1/2}\big((K+\tfrac{1}{2})g - V\Phi_\ell\big)'\right\|_{L^2(\Gamma)}, \qquad \mu_\ell = \left\|h_\ell^{1/2}\big(\widehat{\Phi}_\ell - \Phi_\ell\big)\right\|_{L^2(\Gamma)}, \qquad \eta_\ell = \left|\!\left|\!\left|\widehat{\Phi}_\ell - \Phi_\ell\right|\!\right|\!\right|$$

bei uniformer Verfeinerung. In Tabelle 7.2 sind dieselben Größen für einen $\varrho_\ell$-adaptiven Algorithmus aufgelistet.
Der exakte Fehler wird mittels

$$e_\ell^2 = \left|\!\left|\!\left|\phi - \Phi_\ell\right|\!\right|\!\right|^2 = \left|\!\left|\!\left|\phi\right|\!\right|\!\right|^2 - \left|\!\left|\!\left|\Phi_\ell\right|\!\right|\!\right|^2$$

bestimmt. Da die Lösung $\phi$ eine glatte Funktion ist, ergibt die experimentelle Konvergenzrate sowohl bei uniformer als auch bei adaptiver Verfeinerung $\alpha = \frac{3}{2}$. Dies wird nochmals verdeutlicht durch die Abbildung 7.4, die ebenfalls den exakten Fehler und die Fehlerschätzer für uniforme bzw. adaptive Verfeinerung enthält.

Nun lösen wir das Laplaceproblem 7.1 mit approximierter rechter Seite, das heißt approximierten Dirichletdaten. Der Fehler des gestörten Verfahrens wird mittels

$$e_\ell^2 = \left|\!\left|\!\left|\phi - \widetilde{\Phi}_\ell\right|\!\right|\!\right|^2 = \left|\!\left|\!\left|\phi\right|\!\right|\!\right|^2 - 2\langle\!\langle \Phi_\ell, \widetilde{\Phi}_\ell \rangle\!\rangle + \left|\!\left|\!\left|\widetilde{\Phi}_\ell\right|\!\right|\!\right|^2$$

berechnet.

| $N_\ell$ | $\|\|\|\phi - \Phi_\ell\|\|\|$ | $\alpha_\ell$ | $\eta_\ell$ | $\mu_\ell$ | $\varrho_\ell$ |
|---|---|---|---|---|---|
| 4 | $8.336_0$ | – | $8.064_0$ | $1.855_1$ | $9.121_0$ |
| 8 | $2.109_0$ | 1.98 | $1.979_0$ | $5.049_0$ | $2.274_0$ |
| 16 | $7.266_{-1}$ | 1.54 | $6.815_{-1}$ | $1.807_0$ | $7.816_{-1}$ |
| 32 | $2.519_{-1}$ | 1.53 | $2.359_{-1}$ | $6.363_{-1}$ | $2.851_{-1}$ |
| 64 | $8.853_{-2}$ | 1.51 | $8.283_{-2}$ | $2.245_{-1}$ | $1.045_{-1}$ |
| 128 | $3.125_{-2}$ | 1.50 | $2.923_{-2}$ | $7.932_{-2}$ | $3.778_{-2}$ |
| 256 | $1.104_{-2}$ | 1.50 | $1.033_{-2}$ | $2.804_{-2}$ | $1.352_{-2}$ |
| 512 | $3.903_{-3}$ | 1.50 | $3.651_{-3}$ | $9.912_{-3}$ | $4.811_{-3}$ |
| 1024 | $1.380_{-3}$ | 1.50 | $1.291_{-3}$ | $3.504_{-3}$ | $1.706_{-3}$ |
| 2048 | $4.879_{-4}$ | 1.50 | $4.879_{-4}$ | $1.239_{-3}$ | $6.043_{-4}$ |

Tabelle 7.1: Fehler $\|\|\|\phi - \Phi_\ell\|\|\|$ und Fehlerschätzer $\mu_\ell$, $\varrho_\ell$, $\eta_\ell$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine uniforme Netzfolge im Laplaceproblem 7.1 mit exakter rechter Seite.

| $N_\ell$ | $\|\|\|\phi - \Phi_\ell\|\|\|$ | $\alpha_\ell$ | $\eta_\ell$ | $\mu_\ell$ | $\varrho_\ell$ |
|---|---|---|---|---|---|
| 4 | $8.336_0$ | – | $8.064_0$ | $1.855_1$ | $9.121_0$ |
| 7 | $2.199_0$ | 2.38 | $2.069_0$ | $5.286_0$ | $2.396_0$ |
| 9 | $9.842_{-1}$ | 3.20 | $9.350_{-1}$ | $2.403_0$ | $1.285_0$ |
| 12 | $7.463_{-1}$ | 0.96 | $7.000_{-1}$ | $1.851_0$ | $8.337_{-1}$ |
| 16 | $4.189_{-1}$ | 2.01 | $3.902_{-1}$ | $1.011_0$ | $5.229_{-1}$ |
| 20 | $2.587_{-1}$ | 2.16 | $2.422_{-1}$ | $6.528_{-1}$ | $3.286_{-1}$ |
| 26 | $1.527_{-1}$ | 2.01 | $1.432_{-1}$ | $3.737_{-1}$ | $1.888_{-1}$ |
| 34 | $1.009_{-1}$ | 1.54 | $9.415_{-2}$ | $2.514_{-1}$ | $1.254_{-1}$ |
| 44 | $6.702_{-2}$ | 1.59 | $6.272_{-2}$ | $1.655_{-1}$ | $8.482_{-3}$ |
| 56 | $4.044_{-2}$ | 2.09 | $3.786_{-2}$ | $1.017_{-1}$ | $5.115_{-2}$ |
| 70 | $3.337_{-2}$ | 0.86 | $3.123_{-2}$ | $8.429_{-2}$ | $4.059_{-2}$ |
| 88 | $2.193_{-2}$ | 1.83 | $2.053_{-2}$ | $5.518_{-2}$ | $2.695_{-2}$ |
| 110 | $1.412_{-2}$ | 1.97 | $1.320_{-2}$ | $3.557_{-2}$ | $1.742_{-2}$ |
| 138 | $1.176_{-2}$ | 0.81 | $1.100_{-2}$ | $2.980_{-2}$ | $1.451_{-2}$ |
| 173 | $7.775_{-3}$ | 1.83 | $7.278_{-3}$ | $1.962_{-2}$ | $9.744_{-3}$ |
| 217 | $5.069_{-3}$ | 1.89 | $4.744_{-3}$ | $1.281_{-2}$ | $6.263_{-3}$ |
| 272 | $4.223_{-3}$ | 0.81 | $3.951_{-3}$ | $1.070_{-2}$ | $5.217_{-3}$ |
| 340 | $2.866_{-3}$ | 1.74 | $2.681_{-3}$ | $7.240_{-3}$ | $3.554_{-3}$ |
| 426 | $1.835_{-3}$ | 1.98 | $1.717_{-3}$ | $4.649_{-3}$ | $2.280_{-3}$ |
| 533 | $1.503_{-3}$ | 0.89 | $1.406_{-3}$ | $3.812_{-3}$ | $1.870_{-3}$ |
| 667 | $1.043_{-3}$ | 1.63 | $9.754_{-4}$ | $2.643_{-3}$ | $1.292_{-3}$ |
| 834 | $6.602_{-4}$ | 2.04 | $6.176_{-4}$ | $1.674_{-3}$ | $8.184_{-4}$ |
| 1043 | $5.425_{-4}$ | 0.87 | $5.074_{-4}$ | $1.375_{-3}$ | $6.726_{-4}$ |
| 1304 | $3.857_{-4}$ | 1.52 | $3.608_{-4}$ | $9.785_{-4}$ | $4.789_{-4}$ |
| 1630 | $2.409_{-4}$ | 2.07 | $2.254_{-4}$ | $6.112_{-4}$ | $2.997_{-4}$ |
| 2038 | $1.948_{-4}$ | 0.92 | $1.823_{-4}$ | $4.946_{-4}$ | $2.416_{-4}$ |
| 2552 | $1.426_{-4}$ | 1.31 | $1.334_{-4}$ | $3.618_{-4}$ | $1.768_{-4}$ |
| 3190 | $8.874_{-5}$ | 1.88 | $8.301_{-5}$ | $2.251_{-4}$ | $1.101_{-4}$ |

Tabelle 7.2: Fehler $\|\|\|\phi - \Phi_\ell\|\|\|$ und Fehlerschätzer $\mu_\ell$, $\varrho_\ell$, $\eta_\ell$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine $\varrho_\ell$-adaptive Netzfolge im Laplaceproblem 7.1 mit exakter rechter Seite.

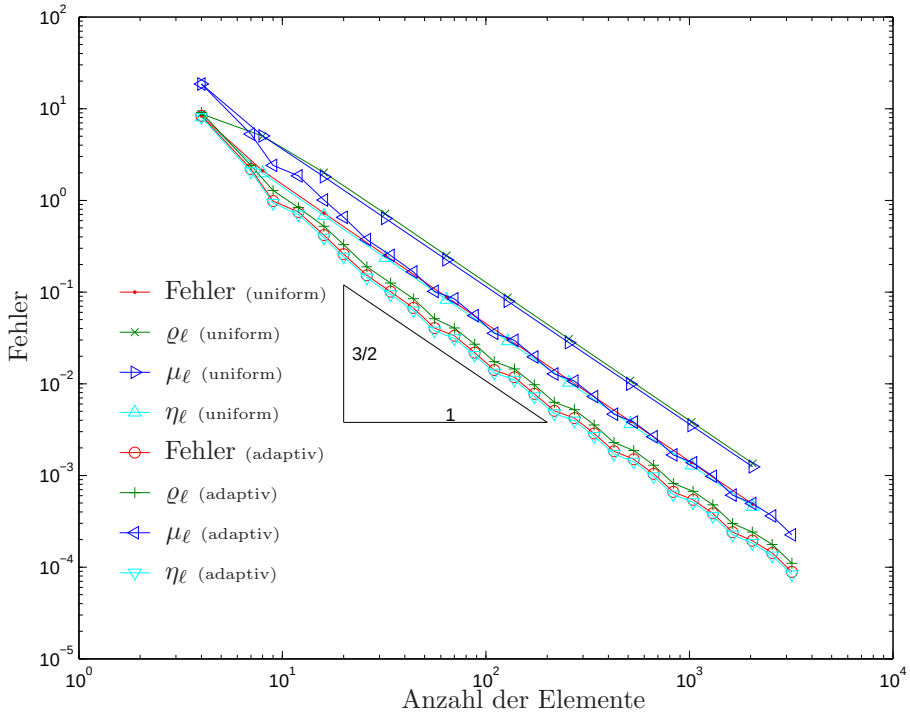


Abbildung 7.4: Fehler $\|\!|\!|\phi - \Phi_\ell|\!|\!|$ und Fehlerschätzer $\mu_\ell$, $\varrho_\ell$, $\eta_\ell$ für eine uniforme und eine $\varrho_\ell$-adaptive Netzfolge im Laplaceproblem 7.1 mit exakter rechter Seite.

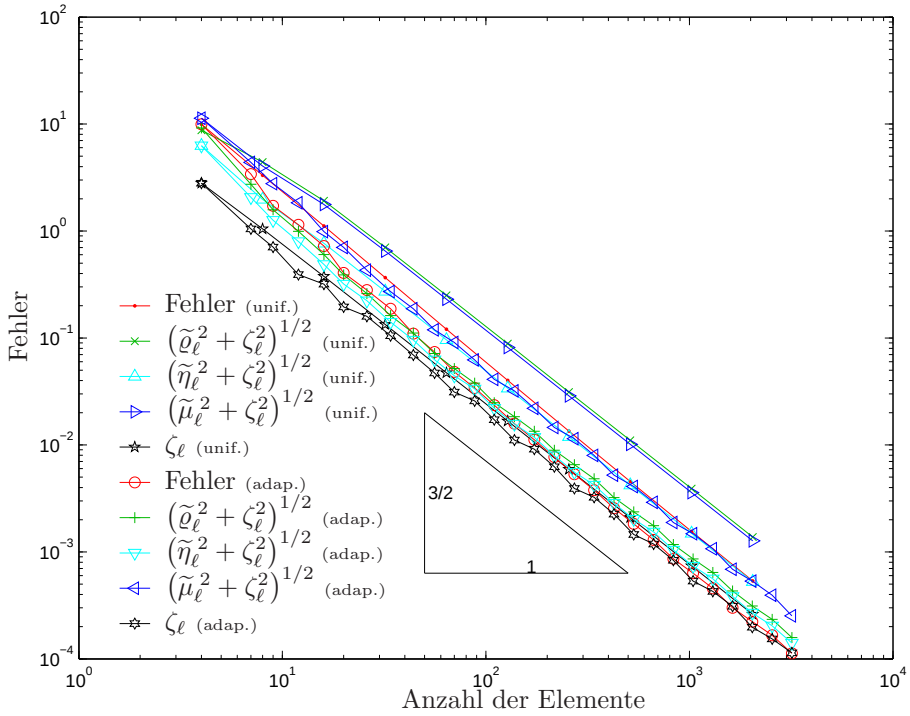


Abbildung 7.5: Fehler $\|\!|\!|\phi - \widetilde{\Phi}_\ell|\!|\!|$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ für eine uniforme und eine $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.1 mit gestörter rechter Seite.

| $N_\ell$ | $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ | $\alpha_\ell$ | $\zeta_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|---|
| 4 | $9.936_{0}$ | – | $2.795_{0}$ | $9.198_{0}$ | $1.132_{1}$ | $6.276_{0}$ |
| 8 | $3.316_{0}$ | 1.98 | $1.048_{0}$ | $2.632_{0}$ | $4.051_{0}$ | $1.971_{0}$ |
| 16 | $1.113_{0}$ | 1.54 | $3.763_{-1}$ | $9.432_{-1}$ | $1.772_{0}$ | $7.674_{1}$ |
| 32 | $3.660_{-1}$ | 1.53 | $1.336_{-1}$ | $3.312_{-1}$ | $6.500_{-1}$ | $2.723_{1}$ |
| 64 | $1.208_{-1}$ | 1.51 | $4.727_{-2}$ | $1.178_{-1}$ | $2.307_{-1}$ | $9.592_{-2}$ |
| 128 | $4.015_{-2}$ | 1.50 | $1.672_{-2}$ | $4.188_{-2}$ | $8.142_{-2}$ | $3.380_{-2}$ |
| 256 | $1.346_{-2}$ | 1.50 | $5.910_{-3}$ | $1.486_{-2}$ | $2.873_{-2}$ | $1.193_{-2}$ |
| 512 | $4.552_{-3}$ | 1.50 | $2.090_{-3}$ | $5.263_{-3}$ | $1.015_{-2}$ | $4.212_{-3}$ |
| 1024 | $1.553_{-3}$ | 1.50 | $7.388_{-4}$ | $1.863_{-3}$ | $3.584_{-3}$ | $1.488_{-3}$ |
| 2048 | $5.352_{-4}$ | 1.50 | $2.612_{-4}$ | $6.589_{-4}$ | $1.267_{-3}$ | $5.260_{-4}$ |

Tabelle 7.3: Fehler $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine uniforme Netzfolge im Laplaceproblem 7.1 mit gestörter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

| $N_\ell$ | $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ | $\alpha_\ell$ | $\zeta_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|---|
| 4 | $9.936_{0}$ | – | $2.795_{0}$ | $9.198_{0}$ | $1.132_{1}$ | $6.276_{0}$ |
| 7 | $3.407_{0}$ | 2.38 | $1.048_{0}$ | $2.735_{0}$ | $4.382_{0}$ | $2.064_{0}$ |
| 9 | $1.717_{0}$ | 3.20 | $7.074_{-1}$ | $1.577_{0}$ | $2.773_{0}$ | $1.261_{0}$ |
| 12 | $1.141_{0}$ | 0.96 | $3.908_{-1}$ | $9.943_{-1}$ | $1.832_{0}$ | $7.970_{-1}$ |
| 16 | $7.230_{-1}$ | 2.01 | $3.184_{-1}$ | $6.028_{-1}$ | $9.875_{-1}$ | $4.872_{-1}$ |
| 20 | $4.062_{-1}$ | 2.16 | $1.953_{-1}$ | $3.888_{-1}$ | $7.026_{-1}$ | $3.193_{-1}$ |
| 26 | $2.805_{-1}$ | 2.01 | $1.591_{-1}$ | $2.597_{-1}$ | $4.308_{-1}$ | $2.229_{-1}$ |
| 44 | $1.099_{-1}$ | 1.59 | $6.935_{-2}$ | $1.112_{-1}$ | $1.876_{-1}$ | $9.583_{-2}$ |
| 56 | $7.397_{-2}$ | 2.09 | $4.709_{-2}$ | $7.156_{-2}$ | $1.193_{-1}$ | $6.267_{-2}$ |
| 70 | $4.820_{-2}$ | 0.86 | $3.120_{-2}$ | $5.172_{-2}$ | $9.081_{-2}$ | $4.443_{-2}$ |
| 88 | $3.413_{-2}$ | 1.83 | $2.583_{-2}$ | $3.779_{-2}$ | $6.233_{-2}$ | $3.338_{-2}$ |
| 110 | $2.356_{-2}$ | 1.97 | $1.720_{-2}$ | $2.484_{-2}$ | $4.101_{-2}$ | $2.209_{-2}$ |
| 138 | $1.583_{-2}$ | 0.81 | $1.112_{-2}$ | $1.837_{-2}$ | $3.202_{-2}$ | $1.571_{-2}$ |
| 174 | $1.102_{-2}$ | 1.87 | $9.131_{-3}$ | $1.329_{-2}$ | $2.165_{-2}$ | $1.169_{-2}$ |
| 218 | $7.698_{-3}$ | 1.81 | $6.166_{-3}$ | $8.862_{-3}$ | $1.451_{-2}$ | $7.858_{-3}$ |
| 274 | $5.302_{-3}$ | 0.80 | $3.826_{-3}$ | $6.491_{-3}$ | $1.141_{-2}$ | $5.512_{-3}$ |
| 344 | $3.677_{-3}$ | 1.86 | $3.231_{-3}$ | $4.730_{-3}$ | $7.759_{-3}$ | $4.156_{-3}$ |
| 430 | $2.534_{-3}$ | 1.88 | $2.210_{-3}$ | $3.172_{-3}$ | $5.166_{-3}$ | $2.804_{-3}$ |
| 538 | $1.821_{-3}$ | 0.85 | $1.410_{-3}$ | $2.344_{-3}$ | $4.077_{-3}$ | $1.995_{-3}$ |
| 674 | $1.265_{-3}$ | 1.72 | $1.171_{-3}$ | $1.726_{-3}$ | $2.847_{-3}$ | $1.513_{-3}$ |
| 843 | $8.513_{-4}$ | 2.00 | $8.175_{-4}$ | $1.152_{-3}$ | $1.853_{-3}$ | $1.022_{-3}$ |
| 1054 | $6.266_{-4}$ | 0.86 | $5.235_{-4}$ | $8.479_{-4}$ | $1.461_{-3}$ | $7.263_{-4}$ |
| 1318 | $4.458_{-4}$ | 1.59 | $4.243_{-4}$ | $6.311_{-4}$ | $1.045_{-3}$ | $5.516_{-4}$ |
| 1648 | $2.961_{-4}$ | 2.02 | $3.028_{-4}$ | $4.231_{-4}$ | $6.759_{-4}$ | $3.760_{-4}$ |
| 2060 | $2.208_{-4}$ | 0.89 | $1.932_{-4}$ | $3.079_{-4}$ | $5.274_{-4}$ | $2.647_{-4}$ |
| 2576 | $1.637_{-6}$ | 1.36 | $1.523_{-4}$ | $2.310_{-4}$ | $3.867_{-4}$ | $2.009_{-4}$ |

Tabelle 7.4: Fehler $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.1 mit gestörter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

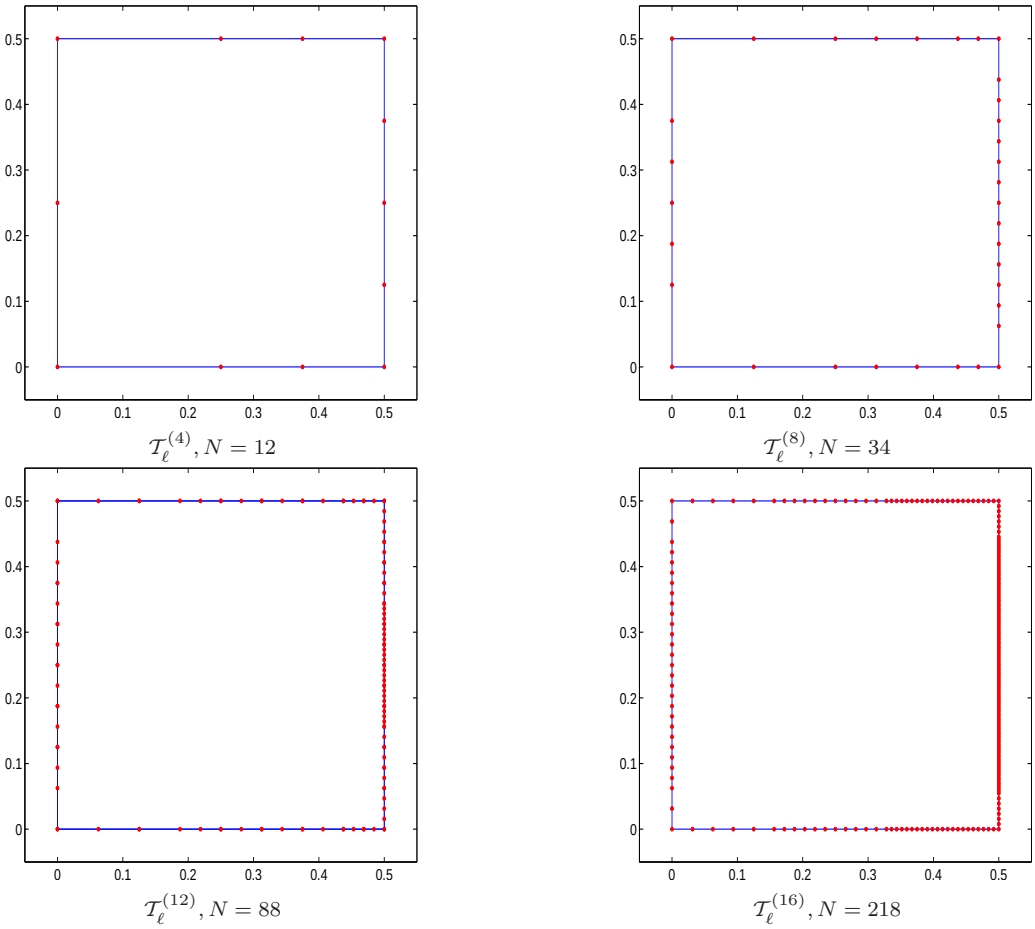


Abbildung 7.6: Ausschnitt der Folge von Partitionen, die durch $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Verfeinerungen im Laplaceproblem 7.1 hervorgeht.

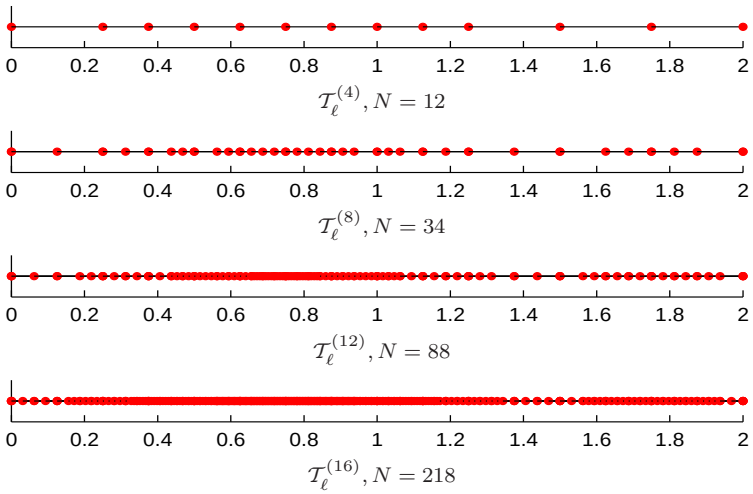


Abbildung 7.7: Ausschnitt der Folge von Partitionen, die durch $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Verfeinerungen im Laplaceproblem 7.1 hervorgeht.

| $N_\ell$ | $\vert\!\vert\!\vert \phi - \Phi_\ell \vert\!\vert\!\vert$ | $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ | $\zeta_\ell$ | $\varrho_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\eta_\ell$ | $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|---|---|
| 4 | $8.336_{0}$ | $9.936_{0}$ | $2.795_{0}$ | $9.121_{0}$ | $9.198_{0}$ | $8.064_{0}$ | $6.276_{0}$ |
| 7 | $2.199_{0}$ | $3.407_{0}$ | $1.048_{0}$ | $2.396_{0}$ | $2.735_{0}$ | $2.069_{0}$ | $2.064_{0}$ |
| 9 | $9.842_{-1}$ | $1.717_{0}$ | $7.074_{-1}$ | $1.285_{0}$ | $1.577_{0}$ | $9.350_{-1}$ | $1.261_{0}$ |
| 12 | $7.463_{-1}$ | $1.141_{0}$ | $3.908_{-1}$ | $8.337_{-1}$ | $9.943_{-1}$ | $7.000_{-1}$ | $7.970_{-1}$ |
| 16 | $4.189_{-1}$ | $7.230_{-1}$ | $3.184_{-1}$ | $5.229_{-1}$ | $6.028_{-1}$ | $3.902_{-1}$ | $4.872_{-1}$ |
| 20 | $2.587_{-1}$ | $4.062_{-1}$ | $1.953_{-1}$ | $3.286_{-1}$ | $3.888_{-1}$ | $2.422_{-1}$ | $3.193_{-1}$ |
| 26 | $1.527_{-1}$ | $2.805_{-1}$ | $1.591_{-1}$ | $1.888_{-1}$ | $2.597_{-1}$ | $1.432_{-1}$ | $2.229_{-1}$ |
| 34 | $1.009_{-1}$ | $1.876_{-1}$ | $1.048_{-1}$ | $1.254_{-1}$ | $1.621_{-1}$ | $9.415_{-2}$ | $1.409_{-1}$ |
| 44 | $6.702_{-2}$ | $1.099_{-1}$ | $6.935_{-2}$ | $8.482_{-2}$ | $1.112_{-1}$ | $6.272_{-2}$ | $9.583_{-2}$ |
| 56 | $4.044_{-2}$ | $7.397_{-2}$ | $4.709_{-2}$ | $5.115_{-2}$ | $7.156_{-2}$ | $3.786_{-2}$ | $6.267_{-2}$ |
| 70 | $3.337_{-2}$ | $4.820_{-2}$ | $3.120_{-2}$ | $4.059_{-2}$ | $5.172_{-2}$ | $3.123_{-2}$ | $4.443_{-2}$ |
| 88 | $2.193_{-2}$ | $3.413_{-2}$ | $2.583_{-2}$ | $2.695_{-2}$ | $3.779_{-2}$ | $2.053_{-2}$ | $3.338_{-2}$ |
| 110 | $1.412_{-2}$ | $2.356_{-2}$ | $1.720_{-2}$ | $1.742_{-2}$ | $2.484_{-2}$ | $1.320_{-2}$ | $2.209_{-2}$ |
| 138 | $1.176_{-2}$ | $1.583_{-2}$ | $1.112_{-2}$ | $1.451_{-2}$ | $1.837_{-2}$ | $1.100_{-2}$ | $1.571_{-2}$ |
| 174 | $7.630_{-3}$ | $1.102_{-2}$ | $9.131_{-3}$ | $9.570_{-3}$ | $1.329_{-2}$ | $7.143_{-3}$ | $1.169_{-2}$ |
| 218 | $5.069_{-3}$ | $7.698_{-3}$ | $6.166_{-3}$ | $6.263_{-3}$ | $8.862_{-3}$ | $4.744_{-3}$ | $7.858_{-3}$ |
| 274 | $4.223_{-3}$ | $5.301_{-3}$ | $3.826_{-3}$ | $5.219_{-3}$ | $6.491_{-3}$ | $3.951_{-3}$ | $5.512_{-3}$ |
| 344 | $2.767_{-3}$ | $3.677_{-3}$ | $3.231_{-3}$ | $3.433_{-3}$ | $4.730_{-3}$ | $2.589_{-3}$ | $4.156_{-3}$ |
| 430 | $1.817_{-3}$ | $2.534_{-3}$ | $2.210_{-3}$ | $2.258_{-3}$ | $3.172_{-3}$ | $1.701_{-3}$ | $2.804_{-3}$ |
| 538 | $1.503_{-3}$ | $1.820_{-3}$ | $1.410_{-3}$ | $1.871_{-3}$ | $2.344_{-3}$ | $1.406_{-3}$ | $1.995_{-3}$ |
| 674 | $1.020_{-3}$ | $1.265_{-3}$ | $1.171_{-3}$ | $1.264_{-3}$ | $1.726_{-3}$ | $9.540_{-4}$ | $1.513_{-3}$ |
| 843 | $6.513_{-4}$ | $8.501_{-4}$ | $8.175_{-4}$ | $8.074_{-4}$ | $1.152_{-3}$ | $6.093_{-4}$ | $1.022_{-3}$ |
| 1054 | $5.372_{-4}$ | $6.251_{-4}$ | $5.235_{-4}$ | $6.661_{-4}$ | $8.479_{-4}$ | $5.025_{-4}$ | $7.263_{-4}$ |
| 1318 | $3.758_{-4}$ | $4.436_{-4}$ | $4.243_{-4}$ | $4.666_{-4}$ | $6.311_{-4}$ | $3.516_{-4}$ | $5.516_{-4}$ |
| 1648 | $2.371_{-4}$ | $2.927_{-4}$ | $3.028_{-4}$ | $2.950_{-4}$ | $4.231_{-4}$ | $2.218_{-4}$ | $3.760_{-4}$ |
| 2060 | $1.932_{-4}$ | $2.167_{-4}$ | $1.932_{-4}$ | $2.396_{-4}$ | $3.079_{-4}$ | $1.807_{-4}$ | $2.647_{-4}$ |
| 2576 | $1.399_{-4}$ | $1.580_{-4}$ | $1.523_{-4}$ | $1.735_{-4}$ | $2.310_{-4}$ | $1.309_{-4}$ | $2.009_{-4}$ |

Tabelle 7.5: Fehler $\vert\!\vert\!\vert \phi - \Phi_\ell \vert\!\vert\!\vert$, $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ sowie Fehlerschätzer $\zeta_\ell$, $\widetilde{\varrho}_\ell$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\widetilde{\eta}_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ für eine $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.1 mit gestörter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

Die Steuerung des adaptiven Algorithmus erfolgte mit $\omega_\ell = \left(\varrho_\ell^2 + \zeta_\ell^2\right)^{1/2}$ und $\theta = \frac{1}{4}$.

Die in den Tabellen 7.3 und 7.4 angegebenen Größen, das sind der Fehler $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$, die experimentelle Konvergenzrate $\alpha_\ell$ und die Fehlerschätzer

$$
\begin{aligned}
&\zeta_\ell = \left\| h_\ell^{1/2} (g - g_\ell)' \right\|_{L^2(\Gamma)}, \\
&\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\varrho}_\ell = \left\| h_\ell^{1/2} \big( (K + \tfrac{1}{2}) g_\ell - V \widetilde{\Phi}_\ell \big)' \right\|, \\
&\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\mu}_\ell = \left\| h_\ell^{1/2} \big( \widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell \big) \right\|_{L^2(\Gamma)}, \\
&\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\eta}_\ell = \vert\!\vert\!\vert \widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell \vert\!\vert\!\vert,
\end{aligned}
$$

sind zur Veranschaulichung in Abbildung 7.5 in einem doppelt logarithmischen Diagramm dargestellt. Der parallele Verlauf der Kurven des Fehlers $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ und des Gesamtfehlerschätzers $\omega_\ell = \left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ lassen auf die Zuverlässigkeit und Effizienz des Fehlerschätzers schließen.

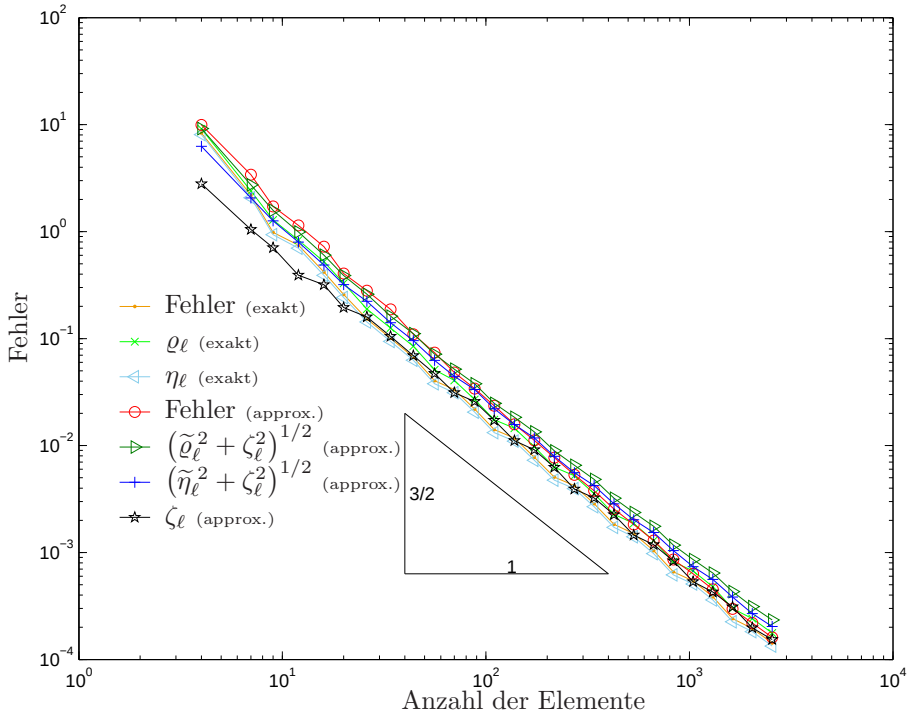


Abbildung 7.8: Fehler $\left|\!\left|\!\left|\phi-\Phi_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\eta_\ell$, $\varrho_\ell$ für eine Berechnung mit exakter rechter Seite und Fehler $\left|\!\left|\!\left|\phi-\widetilde{\Phi}_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\eta}_\ell^2+\zeta_\ell^2\right)^{1/2}$ für eine Berechnung mit approximativer rechter Seite bei einem $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptiven Algorithmus des Laplaceproblems 7.1.

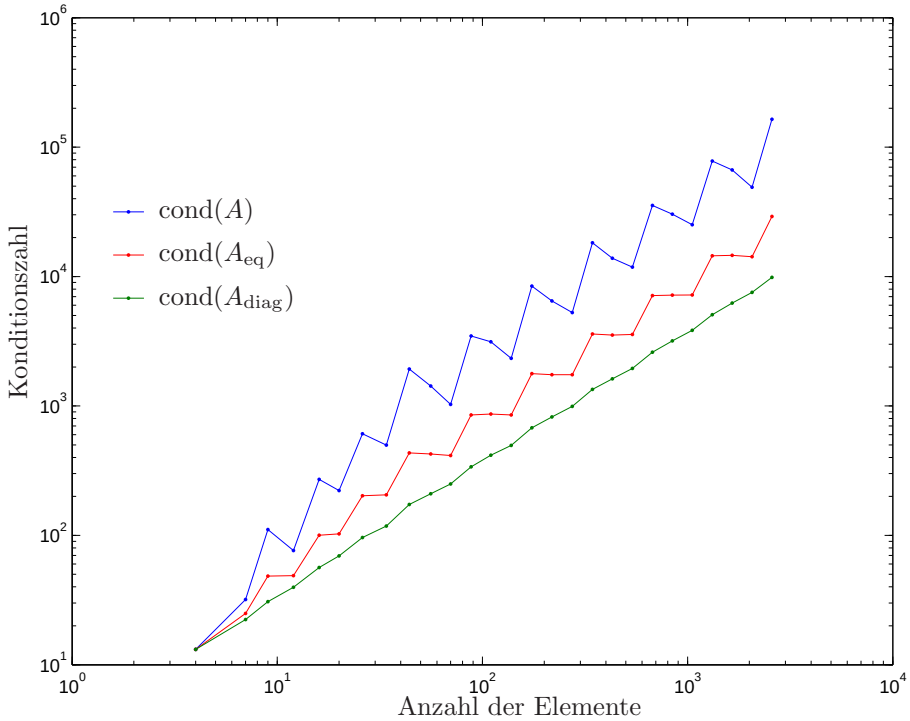


Abbildung 7.9: Konditionszahlen der Steifigkeitsmatrix ohne Vorkonditionierung, mit zeilenäquilibrierter Vorkonditionierung und mit diagonaler Vorkonditionierung für eine $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.1 über der Anzahl der Elemente.

| $N_\ell$ | $\mathrm{cond}(A)$ | $\mathrm{cond}(A_{\mathrm{eq}})$ | $\mathrm{cond}(A_{\mathrm{diag}})$ |
|---|---|---|---|
| 4 | $1.3_1$ | $1.3_{\ 1}$ | $1.3_{\ 1}$ |
| 7 | $3.2_1$ | $2.5_{\ 1}$ | $2.2_{\ 1}$ |
| 9 | $1.1_2$ | $4.8_{\ 1}$ | $3.1_{\ 1}$ |
| 12 | $7.6_2$ | $4.9_{\ 1}$ | $4.0_{\ 1}$ |
| 16 | $2.7_2$ | $1.0_{\ 2}$ | $5.6_{\ 1}$ |
| 20 | $2.2_2$ | $1.0_{\ 2}$ | $6.9_{\ 1}$ |
| 26 | $6.1_2$ | $2.0_{\ 2}$ | $9.6_{\ 1}$ |
| 34 | $5.0_2$ | $2.1_{\ 2}$ | $1.2_{\ 2}$ |
| 44 | $1.9_3$ | $4.3_{\ 2}$ | $1.7_{\ 2}$ |
| 56 | $1.4_3$ | $4.3_{\ 2}$ | $2.1_{\ 2}$ |
| 70 | $1.0_3$ | $4.1_{\ 2}$ | $2.5_{\ 2}$ |
| 88 | $3.5_3$ | $8.5_{\ 2}$ | $3.4_{\ 2}$ |
| 110 | $3.1_3$ | $8.7_{\ 2}$ | $4.2_{\ 2}$ |
| 138 | $2.3_3$ | $8.5_{\ 2}$ | $5.0_{\ 2}$ |
| 174 | $8.4_3$ | $1.8_{\ 3}$ | $6.8_{\ 2}$ |
| 218 | $6.5_3$ | $1.7_{\ 3}$ | $8.2_{\ 2}$ |
| 274 | $5.3_3$ | $1.7_{\ 3}$ | $9.9_{\ 2}$ |
| 344 | $1.8_4$ | $3.6_{\ 3}$ | $1.3_{\ 3}$ |
| 430 | $1.4_4$ | $3.5_{\ 3}$ | $1.6_{\ 3}$ |
| 538 | $1.2_4$ | $3.6_{\ 3}$ | $2.0_{\ 3}$ |
| 674 | $3.5_4$ | $7.1_{\ 3}$ | $2.6_{\ 3}$ |
| 843 | $3.0_4$ | $7.2_{\ 3}$ | $3.3_{\ 3}$ |
| 1054 | $2.5_4$ | $7.2_{\ 3}$ | $3.8_{\ 3}$ |
| 1318 | $7.8_4$ | $1.4_{\ 4}$ | $5.1_{\ 3}$ |
| 1648 | $6.7_4$ | $1.5_{\ 4}$ | $6.2_{\ 3}$ |
| 2060 | $4.9_4$ | $1.4_{\ 4}$ | $7.5_{\ 3}$ |
| 2576 | $1.6_5$ | $2.9_{\ 4}$ | $9.9_{\ 3}$ |

Tabelle 7.6: Konditionszahlen der Steifigkeitsmatrix ohne Vorkonditionierung, mit zeilenäquilibrierter Vorkonditionierung und mit diagonaler Vorkonditionierung für eine $\big(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\big)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.1.

| $N_\ell$ | $\kappa(\mathcal{T}_\ell)$ | $\frac{h_{\max}}{h_{\min}}$ |
|---|---|---|
| 4 | 1 | 1 |
| 7 | 2 | 2 |
| 9 | 2 | 4 |
| 12 | 2 | 2 |
| 16 | 2 | 4 |
| 20 | 2 | 4 |
| 26 | 2 | 4 |
| 34 | 2 | 4 |
| 44 | 2 | 8 |
| 56 | 2 | 8 |
| 70 | 2 | 4 |
| 88 | 2 | 8 |
| 110 | 2 | 8 |
| 138 | 2 | 8 |
| 174 | 2 | 1 |
| 218 | 2 | 8 |
| 274 | 2 | 8 |
| 344 | 2 | 16 |
| 430 | 2 | 16 |
| 538 | 2 | 16 |
| 674 | 2 | 16 |
| 843 | 2 | 16 |
| 1054 | 2 | 16 |
| 1318 | 2 | 32 |
| 1648 | 2 | 32 |
| 2060 | 2 | 16 |
| 2576 | 2 | 32 |

Tabelle 7.7: Anzahl der Elemente $N_\ell$ in $\mathcal{T}_\ell$, $\kappa(\mathcal{T}_\ell)$ und $\frac{h_{\max}}{h_{\min}}$ bei Rechnung wie in Tabelle 7.6.

In Abbildung 7.6 ist ein Ausschnitt einer Folge von adaptiven Verfeinerungen über der Zahl der Elemente der Partition $\mathcal{T}_\ell$ zu sehen. Man kann erkennen, dass, da die Funktion glatt ist, zu keiner Stelle speziell verfeinert wird. Abbildung 7.7 stellt dieselbe Netzfolge über der Bogenlänge dar.

Zum besseren Vergleich der exakten und der approximativen Berechnung zeigt Abbildung 7.8 sowohl den exakten Fehler $\big|\!\big|\!\big|\phi-\widetilde{\Phi}_\ell\big|\!\big|\!\big|$ und die Fehlerschätzer

$$\begin{aligned}
&\zeta_\ell=\big\|h_\ell^{1/2}(g-g_\ell)'\big\|_{L^2(\Gamma)},\\
&\big(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\big)^{1/2}\quad\text{mit }\widetilde{\varrho}_\ell=\big\|h_\ell^{1/2}\big((K+\tfrac12)g_\ell-V\widetilde{\Phi}_\ell\big)'\big\|,\\
&\big(\widetilde{\eta}_\ell^2+\zeta_\ell^2\big)^{1/2}\quad\text{mit }\widetilde{\eta}_\ell=\big|\!\big|\!\big|\widehat{\widetilde{\Phi}}_\ell-\widetilde{\Phi}_\ell\big|\!\big|\!\big|
\end{aligned}$$

für die approximierte rechte Seite als auch den exakten Fehler $\big|\!\big|\!\big|\phi-\Phi_\ell\big|\!\big|\!\big|$ und die Fehlerschätzer

$$\varrho_\ell=\big\|h_\ell^{1/2}\big((K+\tfrac12)g-V\Phi_\ell\big)'\big\|_{L^2(\Gamma)},\qquad \eta_\ell=\big|\!\big|\!\big|\widehat{\Phi}_\ell-\Phi_\ell\big|\!\big|\!\big|$$

für die exakte rechte Seite.

Die Steuerung des adaptiven Algorithmus erfolgte mittels $\omega_\ell = \left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ und $\theta = \frac{1}{4}$. Man sieht, dass sich die Fehler und auch Fehlerschätzer kaum unterscheiden. Aus diesem Grund enthält Tabelle 7.5 die genauen Größen der Fehler und der Fehlerschätzer.

Bei adaptiver Netzverfeinerung kann der Durchmesser der Elemente einer Partition sehr stark variieren und die Steifigkeitsmatrix daher schlecht konditioniert sein. Aus diesem Grund wird vorkonditioniert. Zu Lösen ist das lineare Gleichungssystem $Ax = b$ mit einer symmetrischen Matrix $A$.

Um gleichzeitig die Symmetrie zu erhalten und die Konditionszahl zu vermindern, wird stattdessen das folgende äquivalente Gleichungssystem

$$D^{1/2}AD^{1/2}D^{-1/2}x = D^{1/2}b$$

betrachtet. Setzt man $y := D^{-1/2}x$, muss also $D^{1/2}AD^{1/2}y = D^{1/2}b$ gelöst und danach $x = D^{1/2}y$ berechnet werden. Die Matrix $D$ ist eine Diagonalmatrix, deren Einträge die Reziprokwerten der Diagonaleinträge von $A$ sind, d.h. $d_{jj} = a_{jj}^{-1}$. Zu Testzwecken wird außerdem auch Zeilenäquilibrierung als Vorkonditionierung verwendet. Die Diagonalmatrix $D$ enthält in diesem Fall die Reziprokwerte der Zeilensummen von $A$ als Diagonaleinträge, d.h. $d_{jj} = \left(\sum_{i=1}^n |a_{ij}|\right)^{-1}$.

Die zeilenäquilibrierte Vorkonditionierung liefert eine Verbesserung der Konditionszahl, gemessen in der Zeilensummennorm sogar die optimale Verbesserung, die durch Multiplikation mit einer Diagonalmatrix erreicht werden kann.

**Satz 7.1. (Plato 2000 [16, Seite 79f]).**
*Sei $A_{\mathrm{eq}} \in \mathbb{R}^{N\times N}$ eine reguläre, zeilenäquilibirierte Matrix, d.h. $\sum_{j=1}^N |a_{ij}|$, $i = 1,\dots,N$. Für jede reguläre Diagonalmatrix $D \in \mathbb{R}^{N\times N}$ gilt*

$$\mathrm{cond}_\infty(A_{\mathrm{eq}}) \le \mathrm{cond}_\infty(DA_{\mathrm{eq}}). \tag{7.2}$$

□

Die Arbeit von Ainsworth/McLean/Tran 1999 [1] zeigt, dass es möglich ist mittels diagonaler Vorkonditionierung beinahe dieselbe Konditionszahl für den adaptiven Algorithmus wie für uniforme Verfeinerung zu erhalten.

**Satz 7.2. (Ainsworth/McLean/Tran 1999 [1, Theorem 2.1]).**
*Es bezeichnen* $\mathrm{cond}(A)$ *und* $\mathrm{cond}(A_{\mathrm{diag}})$ *die Konditionszahlen der Steifigkeitsmatrix $A \in \mathbb{R}^{N\times N}$ und der diagonal vorkonditionierten Matrix $A_{\mathrm{diag}} \in \mathbb{R}^{N\times N}$ gemessen in der Spektralnorm $\|\cdot\|_2$. Für $h_{\max}$ genügend klein gelten mit Konstanten $C_{\mathrm{stiff}}, C_{\mathrm{diag}} > 0$*

$$\mathrm{cond}_2(A) \le C_{\mathrm{stiff}} N\big(1 + \big|\log(Nh_{\max})\big|\big)\Big(\frac{h_{\max}}{h_{\min}}\Big)^2 \tag{7.3}$$

*und*

$$\mathrm{cond}_2(A_{\mathrm{diag}}) \le C_{\mathrm{diag}} N\big(1 + \big|\log(Nh_{\min})\big|\big)\frac{1 + |\log h_{\min}|}{1 + |\log h_{\max}|}. \tag{7.4}$$

*Die Konstanten $C_{\mathrm{stiff}}$ und $C_{\mathrm{diag}}$ sind nur abhängig von $\Gamma$.* □

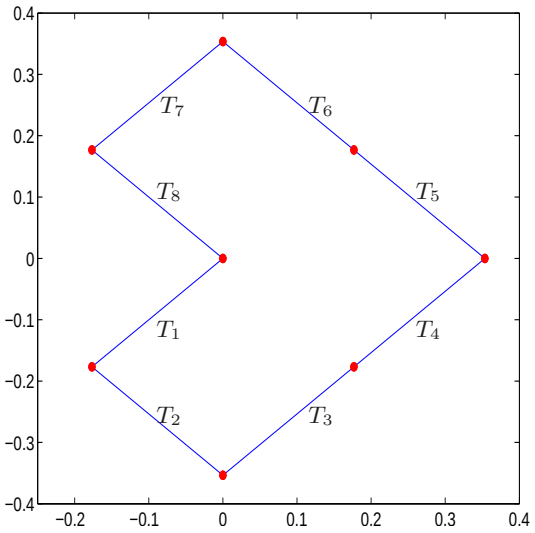


Abbildung 7.10: Startnetz $\mathcal{T}_0$ mit $N_0 = 8$ Elementen des Laplaceproblems 7.2.

In Tabelle 7.6 und Abbildung 7.9 sieht man die Anzahl der Elemente $N_\ell$ und die Konditionszahlen der unveränderten Matrix $A$, der mittels Zeilenäquilibrierung vorkonditionierten Matrix $A_{\text{eq}}$ und der

diagonal vorkonditionierten Matrix $A_{\text{diag}}$ einer adaptiven Netzfolge. Die Konditionszahlen wurden mit dem MATLAB-Befehl `cond` berechnet, der $\text{cond}(A) = \|A\|_2\|A^{-1}\|_2$ mittels Singulärwertzerlegung bestimmt.

In Korollar 3.11 wurde bewiesen, dass $\kappa(\mathcal{T}_\ell)$, das ist der maximale Quotient der Netzweiten zweier benachbarter Elemente, bei Anwendung einer erweiterten Markierungsstrategie (3.11) beschränkt bleibt. Das untermauern die Daten in Tabelle 7.7. Zum Vergleich ist in jedem Iterationsschritt auch der maximale Quotient der lokalen Netzweiten aller, also nicht unbedingt benachbarter Elemente, d.h $\frac{h_{\max}}{h_{\min}}$, angegeben.

## 7.2 Laplaceproblem am L-förmigen Gebiet

Im diesem Abschnitt wird das Laplaceproblem 7.1 für ein L-förmiges Gebiet mit der exakten Lösung $u(x, y)$ betrachtet und die dazu äquivalente Symmsche Integralgleichung $V\phi = (K + \frac{1}{2})g$ mit $g = u|_\Gamma$ gelöst.

Das verwendete Startnetz $\mathcal{T}_0$ mit $N_0 = 8$ Elementen ist in Abbildung 7.10 zu sehen. Man kann erkennen, dass für den Durchmesser des Gebiets $\text{diam}(\Omega) = \frac{1}{\sqrt{2}} \approx 0.7071 < 1$ gilt, um die Elliptizität des Einfachschichtpotentials zu garantieren.

### 7.2.1 Erstes Experiment

Die exakte Lösung von 7.1 lautet $u(x, y) = u(r\cos\psi, r\sin\psi) = r^{2/3}\cos\frac{2}{3}\psi$. Die Dirichletdaten $g = u|_\Gamma \in H^{1/2}(\Gamma)$ sind in Abbildung 7.11 über der Bogenlänge $0 \le s < 2$ dargestellt, wobei $s \in \{0, 2\}$ gerade dem gemeinsamen Eckpunkt $(0.35355, 0)$ der Elemente $T_4$ und $T_5$ der Partition $\mathcal{T}_0$ entspricht.

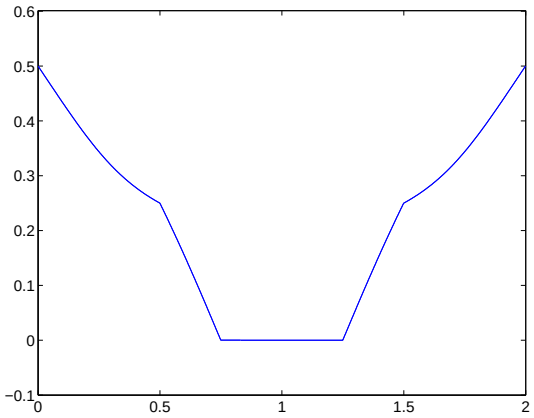


Abbildung 7.11: Exakte Dirichletdaten des Laplaceproblems 7.2.1 über der Bogenlänge $s \in [0,2]$, wobei $s \in \{0,2\}$ gerade dem Eckpunkt $(0.35355, 0)$ entspricht.

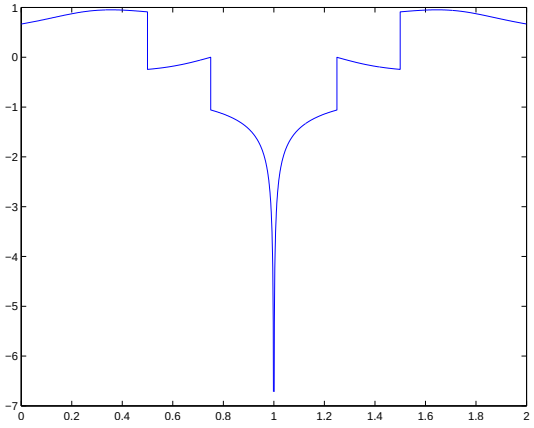


Abbildung 7.12: Exakte Lösung des Laplaceproblems 7.2.1 über der Bogenlänge $s \in [0,2]$, wobei $s \in \{0,2\}$ gerade dem Eckpunkt $(0.35355, 0)$ entspricht.

Die Lösung $\phi = \frac{\partial u}{\partial n}$ der Symmschen Integralgleichung hat an der einspringenden Ecke $(0,0)$ eine Singularität. Um einen Eindruck über das Verhalten der kontinuierlichen Lösung

$$\phi(x,y) = \phi(r\cos\psi, r\sin\psi) = \frac{2}{3} r^{-1/3} \begin{pmatrix} \cos\psi\cos(\frac{2}{3}\psi) + \sin\psi\sin(\frac{2}{3}\psi) \\ \sin\psi\cos(\frac{2}{3}\psi) - \cos\psi\sin(\frac{2}{3}\psi) \end{pmatrix} \cdot n(x,y),$$

wobei $n(x,y)$ den äußeren Normalenvektor bezeichnet, zu erhalten, ist diese in Abbildung 7.12 dargestellt.

Extrapolation mit der Aitkenschen $\triangle^2$-Methode führt zu einer geschätzten Energienorm $\left|\!\left|\!\left|\phi\right|\!\right|\!\right|^2 = 0.202058098880$ der unbekannten Lösung $\phi$.

Wiederum wird zunächst die Symmsche Integralgleichung mit exakter rechter Seite, das heißt ungestörten Dirichletdaten, gelöst. Die Anzahl der Elemente $N_\ell$, die experimentelle Konvergenzrate $\alpha_\ell$, der exakte Fehler $\left|\!\left|\!\left|\phi - \Phi_\ell\right|\!\right|\!\right|$ und die Fehlerschätzer

$$\varrho_\ell = \left\| h_\ell^{1/2} \big( (K + \tfrac{1}{2})g - V\Phi_\ell \big)' \right\|_{L^2(\Gamma)}, \qquad \mu_\ell = \left\| h_\ell^{1/2} \big( \widehat{\Phi}_\ell - \Phi_\ell \big) \right\|_{L^2(\Gamma)}, \qquad \eta_\ell = \left|\!\left|\!\left| \widehat{\Phi}_\ell - \Phi_\ell \right|\!\right|\!\right|$$

sind für eine uniforme Verfeinerung in Tabelle 7.8 und für eine adaptive Verfeinerung in Tabelle 7.9 zusammengefasst. Abbildung 7.13 zeigt den exakten Fehler und die Fehlerschätzer in einem doppelt

| $N_\ell$ | $\|\|\|\phi - \Phi_\ell\|\|\|$ | $\alpha_\ell$ | $\eta_\ell$ | $\mu_\ell$ | $\varrho_\ell$ |
|---|---|---|---|---|---|
| 8 | $8.355_{-2}$ | – | $6.632_{-2}$ | $1.393_{-1}$ | $1.290_{-1}$ |
| 16 | $5.083_{-2}$ | 0.72 | $3.944_{-2}$ | $8.597_{-2}$ | $7.576_{-2}$ |
| 32 | $3.206_{-2}$ | 0.66 | $2.490_{-2}$ | $5.389_{-2}$ | $4.768_{-2}$ |
| 64 | $2.020_{-2}$ | 0.67 | $1.569_{-2}$ | $3.394_{-2}$ | $3.003_{-2}$ |
| 128 | $1.273_{-2}$ | 0.67 | $9.844_{-3}$ | $2.138_{-2}$ | $1.892_{-2}$ |
| 256 | $8.018_{-3}$ | 0.67 | $6.227_{-3}$ | $1.347_{-2}$ | $1.192_{-2}$ |
| 512 | $5.051_{-3}$ | 0.67 | $3.923_{-3}$ | $8.487_{-3}$ | $7.508_{-3}$ |
| 1024 | $3.182_{-3}$ | 0.67 | $2.471_{-3}$ | $5.346_{-3}$ | $4.730_{-3}$ |
| 2048 | $2.005_{-3}$ | 0.67 | $1.557_{-3}$ | $3.368_{-3}$ | $2.980_{-3}$ |

Tabelle 7.8: Fehler $\|\|\|\phi - \Phi_\ell\|\|\|$ und Fehlerschätzer $\mu_\ell$, $\varrho_\ell$, $\eta_\ell$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine uniforme Netzfolge im Laplaceproblem 7.2.1 mit exakter rechter Seite.

| $N_\ell$ | $\|\|\|\phi - \Phi_\ell\|\|\|$ | $\alpha_\ell$ | $\eta_\ell$ | $\mu_\ell$ | $\varrho_\ell$ |
|---|---|---|---|---|---|
| 8 | $8.355_{-2}$ | – | $6.632_{-2}$ | $1.393_{-1}$ | $1.290_{-1}$ |
| 10 | $5.209_{-2}$ | 2.12 | $4.089_{-2}$ | $8.984_{-2}$ | $8.017_{-2}$ |
| 14 | $2.458_{-2}$ | 2.28 | $2.062_{-2}$ | $4.417_{-2}$ | $4.240_{-2}$ |
| 18 | $1.851_{-2}$ | 1.13 | $1.617_{-2}$ | $3.603_{-2}$ | $3.079_{-2}$ |
| 26 | $8.187_{-3}$ | 2.75 | $7.235_{-3}$ | $1.693_{-2}$ | $1.292_{-2}$ |
| 32 | $5.461_{-3}$ | 1.95 | $4.846_{-3}$ | $1.177_{-2}$ | $9.601_{-3}$ |
| 44 | $4.133_{-3}$ | 0.97 | $3.822_{-3}$ | $9.492_{-3}$ | $5.476_{-3}$ |
| 48 | $2.959_{-3}$ | 3.84 | $2.687_{-3}$ | $6.848_{-3}$ | $4.137_{-3}$ |
| 57 | $2.176_{-3}$ | 1.79 | $1.995_{-3}$ | $5.094_{-3}$ | $3.313_{-3}$ |
| 69 | $1.816_{-3}$ | 0.95 | $1.638_{-3}$ | $4.322_{-3}$ | $2.536_{-3}$ |
| 80 | $1.451_{-3}$ | 1.52 | $1.348_{-3}$ | $3.499_{-3}$ | $1.959_{-3}$ |
| 105 | $8.243_{-4}$ | 2.24 | $7.653_{-4}$ | $1.998_{-3}$ | $1.171_{-3}$ |
| 127 | $6.994_{-4}$ | 0.86 | $6.520_{-4}$ | $1.716_{-3}$ | $9.121_{-4}$ |
| 149 | $5.493_{-4}$ | 1.51 | $5.127_{-4}$ | $1.355_{-3}$ | $7.079_{-4}$ |
| 169 | $4.497_{-4}$ | 1.59 | $4.191_{-4}$ | $1.109_{-3}$ | $5.398_{-4}$ |
| 197 | $3.156_{-4}$ | 2.31 | $2.944_{-4}$ | $7.816_{-4}$ | $4.202_{-4}$ |
| 240 | $2.699_{-4}$ | 0.79 | $2.521_{-4}$ | $6.728_{-4}$ | $3.260_{-4}$ |
| 286 | $2.153_{-4}$ | 1.29 | $2.014_{-4}$ | $5.363_{-4}$ | $2.526_{-4}$ |
| 327 | $1.688_{-4}$ | 1.82 | $1.577_{-4}$ | $4.203_{-4}$ | $1.945_{-4}$ |
| 381 | $1.126_{-4}$ | 2.65 | $1.051_{-4}$ | $2.817_{-4}$ | $1.490_{-4}$ |
| 468 | $9.606_{-5}$ | 0.77 | $8.970_{-5}$ | $2.410_{-4}$ | $1.144_{-4}$ |
| 562 | $7.847_{-5}$ | 1.11 | $7.326_{-5}$ | $1.969_{-4}$ | $8.753_{-5}$ |
| 649 | $5.928_{-5}$ | 1.96 | $5.523_{-5}$ | $1.483_{-4}$ | $6.688_{-5}$ |
| 773 | $3.929_{-5}$ | 2.38 | $3.643_{-5}$ | $9.813_{-5}$ | $5.079_{-5}$ |
| 950 | $3.303_{-5}$ | 0.86 | $3.053_{-5}$ | $8.238_{-5}$ | $3.861_{-5}$ |
| 1142 | $2.759_{-5}$ | 1.00 | $2.537_{-5}$ | $6.846_{-5}$ | $2.935_{-5}$ |
| 1324 | $2.019_{-5}$ | 2.21 | $1.828_{-5}$ | $4.939_{-5}$ | $2.222_{-5}$ |
| 1619 | $1.436_{-5}$ | 1.85 | $1.257_{-5}$ | $3.398_{-5}$ | $1.677_{-5}$ |
| 1987 | $1.197_{-5}$ | 1.04 | $1.014_{-5}$ | $2.746_{-5}$ | $1.267_{-5}$ |
| 2370 | $1.025_{-5}$ | 1.10 | $8.347_{-6}$ | $2.259_{-5}$ | $9.567_{-6}$ |

Tabelle 7.9: Fehler $\|\|\|\phi - \Phi_\ell\|\|\|$ und Fehlerschätzer $\mu_\ell$, $\varrho_\ell$, $\eta_\ell$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine $\varrho_\ell$-adaptive Netzfolge im Laplaceproblem 7.2.1 mit exakter rechter Seite.

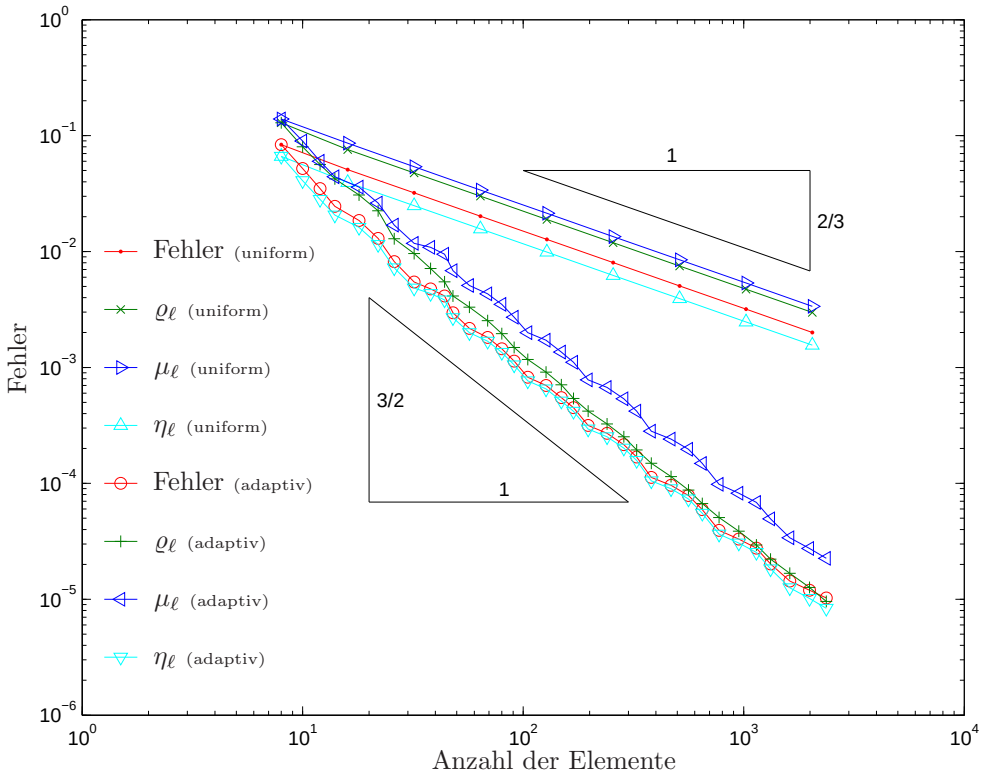


Abbildung 7.13: Fehler $\|\!|\!|\phi - \Phi_\ell|\!|\!|$ und Fehlerschätzer $\mu_\ell$, $\varrho_\ell$, $\eta_\ell$ für eine uniforme und eine $\varrho_\ell$-adaptive Netzfolge im Laplaceproblem 7.2.1 mit exakter rechter Seite.

logarithmischen Diagramm. Bei uniformer Verfeinerung wird entsprechend der Ordnung der Singularität der Lösung eine Konvergenzordnung von $\alpha = \frac{2}{3}$ erreicht. Mittels eines adaptiven Algorithmus — gesteuert mit $\varrho_\ell$ und $\theta = \frac{1}{4}$— lässt sich die Singularität besser auflösen und man erreicht die optimale Konvergenzordnung von $\alpha = \frac{3}{2}$. Die Parallelität der Fehlerschätzer und des Fehlers bestätigen numerisch die Zuverlässigkeit und Effizienz der Fehlerschätzer.

Auch die Symmsche Integralgleichung mit gestörter rechter Seite, das heißt nodal interpolierten Dirichletdaten, wird gelöst. Sowohl für uniforme als auch adaptive Verfeinerung und Steuerung mittels $\omega_\ell = \left(\widetilde{\varrho}_\ell^{\,2} + \zeta_\ell^2\right)^{1/2}$ werden die Anzahl der Elemente $N_\ell$, der exakte Fehler $e_\ell = \|\!|\!|\phi - \widetilde{\Phi}_\ell|\!|\!|$, die experimentelle Konvergenzrate $\alpha_\ell$ und die Fehlerschätzer

$$
\begin{aligned}
&\zeta_\ell = \left\| h_\ell^{1/2}(g - g_\ell)' \right\|_{L^2(\Gamma)}, \\
&\left(\widetilde{\varrho}_\ell^{\,2} + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\varrho}_\ell = \left\| h_\ell^{1/2}\left((K + \tfrac{1}{2})g_\ell - V\widetilde{\Phi}_\ell\right)' \right\|, \\
&\left(\widetilde{\mu}_\ell^{\,2} + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\mu}_\ell = \left\| h_\ell^{1/2}\left(\widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell\right) \right\|_{L^2(\Gamma)}, \\
&\left(\widetilde{\eta}_\ell^{\,2} + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\eta}_\ell = \|\!|\!|\widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell|\!|\!|
\end{aligned}
$$

in Tabelle 7.10 bzw. 7.11 angegeben. Zur Illustration der Ergebnisse sind der exakte Fehler und die Fehlerschätzer in Abbildung 7.16 in einem doppelt logarithmischen Diagramm dargestellt.
Für die uniforme Verfeinerung ergibt sich eine Konvergenzrate von $\alpha = \frac{2}{3}$, was genau dem Reziprokwert des Innenwinkels an der Singularität entspricht. Adaptive Verfeinerung führt zu optimaler Konvergenzordnung $\alpha = \frac{3}{2}$. Alle Fehlerschätzer verlaufen parallel, was die Zuverlässigkeit und Effizienz bestätigt.

| $N_\ell$ | $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ | $\alpha_\ell$ | $\zeta_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|---|
| 8 | $8.715_{-2}$ | – | $3.822_{-3}$ | $1.292_{-1}$ | $1.414_{-1}$ | $6.759_{-2}$ |
| 16 | $5.185_{-2}$ | 0.75 | $1.358_{-3}$ | $7.579_{-2}$ | $8.611_{-2}$ | $3.951_{-2}$ |
| 32 | $3.234_{-2}$ | 0.68 | $4.809_{-4}$ | $4.768_{-2}$ | $5.391_{-2}$ | $2.491_{-2}$ |
| 64 | $2.027_{-2}$ | 0.67 | $1.701_{-4}$ | $3.003_{-2}$ | $3.395_{-2}$ | $1.569_{-2}$ |
| 128 | $1.275_{-2}$ | 0.67 | $6.014_{-5}$ | $1.892_{-2}$ | $2.138_{-2}$ | $9.884_{-3}$ |
| 256 | $8.023_{-3}$ | 0.67 | $2.126_{-5}$ | $1.192_{-2}$ | $1.347_{-2}$ | $6.227_{-3}$ |
| 512 | $5.053_{-3}$ | 0.67 | $7.518_{-6}$ | $7.508_{-3}$ | $8.487_{-3}$ | $3.923_{-3}$ |
| 1024 | $3.182_{-3}$ | 0.67 | $2.658_{-6}$ | $4.730_{-3}$ | $5.346_{-3}$ | $2.471_{-3}$ |
| 2048 | $2.005_{-3}$ | 0.67 | $9.397_{-7}$ | $2.980_{-3}$ | $3.368_{-3}$ | $1.557_{-3}$ |

Tabelle 7.10: Fehler $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine uniforme Netzfolge im Laplaceproblem 7.2.1 mit gestörter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

| $N_\ell$ | $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ | $\alpha_\ell$ | $\zeta_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|---|
| 8 | $8.715_{-2}$ | – | $3.822_{-3}$ | $1.292_{-1}$ | $1.414_{-1}$ | $6.759_{-2}$ |
| 10 | $5.581_{-2}$ | 2.00 | $3.822_{-3}$ | $7.996_{-2}$ | $9.137_{-2}$ | $4.153_{-2}$ |
| 14 | $3.686_{-2}$ | 1.23 | $2.578_{-4}$ | $5.330_{-2}$ | $6.204_{-2}$ | $2.867_{-2}$ |
| 18 | $2.469_{-2}$ | 1.60 | $2.043_{-4}$ | $3.525_{-2}$ | $4.163_{-2}$ | $1.888_{-2}$ |
| 24 | $1.464_{-2}$ | 1.82 | $1.358_{-5}$ | $2.078_{-2}$ | $2.431_{-2}$ | $1.100_{-2}$ |
| 30 | $1.002_{-2}$ | 1.70 | $9.250_{-3}$ | $1.378_{-2}$ | $1.787_{-2}$ | $7.823_{-3}$ |
| 38 | $6.743_{-3}$ | 1.68 | $6.743_{-3}$ | $9.053_{-3}$ | $1.210_{-2}$ | $5.190_{-3}$ |
| 48 | $4.186_{-3}$ | 2.04 | $4.620_{-4}$ | $5.815_{-3}$ | $7.809_{-3}$ | $3.347_{-3}$ |
| 60 | $2.956_{-3}$ | 1.56 | $3.668_{-4}$ | $3.939_{-3}$ | $5.700_{-3}$ | $2.369_{-3}$ |
| 75 | $2.035_{-3}$ | 1.67 | $2.305_{-5}$ | $2.624_{-3}$ | $4.036_{-3}$ | $1.658_{-3}$ |
| 94 | $1.261_{-3}$ | 2.12 | $1.672_{-5}$ | $1.698_{-3}$ | $2.583_{-3}$ | $1.056_{-3}$ |
| 118 | $9.439_{-4}$ | 1.27 | $1.199_{-6}$ | $1.188_{-3}$ | $2.002_{-3}$ | $7.973_{-4}$ |
| 148 | $6.611_{-4}$ | 1.57 | $8.020_{-6}$ | $8.042_{-4}$ | $1.429_{-3}$ | $5.636_{-4}$ |
| 185 | $4.006_{-4}$ | 2.25 | $5.990_{-6}$ | $5.243_{-4}$ | $8.812_{-4}$ | $3.478_{-4}$ |
| 232 | $3.130_{-4}$ | 1.09 | $4.179_{-6}$ | $3.768_{-4}$ | $7.047_{-4}$ | $2.727_{-4}$ |
| 290 | $2.340_{-4}$ | 1.30 | $2.562_{-6}$ | $2.643_{-4}$ | $5.404_{-4}$ | $2.075_{-4}$ |
| 363 | $1.369_{-4}$ | 2.39 | $2.127_{-6}$ | $1.733_{-4}$ | $3.172_{-4}$ | $1.224_{-4}$ |
| 454 | $1.068_{-4}$ | 1.11 | $1.508_{-6}$ | $1.261_{-4}$ | $2.501_{-4}$ | $9.517_{-5}$ |
| 568 | $8.196_{-5}$ | 1.19 | $9.111_{-6}$ | $8.936_{-5}$ | $1.961_{-4}$ | $7.401_{-5}$ |
| 710 | $4.885_{-5}$ | 2.33 | $7.521_{-6}$ | $5.946_{-5}$ | $1.170_{-4}$ | $4.439_{-5}$ |
| 888 | $3.778_{-5}$ | 1.16 | $5.447_{-6}$ | $4.383_{-5}$ | $9.023_{-5}$ | $3.400_{-5}$ |
| 1110 | $2.970_{-5}$ | 1.10 | $3.371_{-7}$ | $3.150_{-5}$ | $7.128_{-5}$ | $2.670_{-5}$ |
| 1388 | $1.842_{-5}$ | 2.24 | $2.663_{-7}$ | $2.102_{-5}$ | $4.353_{-5}$ | $1.636_{-5}$ |
| 1735 | $1.416_{-5}$ | 1.30 | $2.010_{-7}$ | $1.559_{-5}$ | $3.237_{-5}$ | $1.215_{-5}$ |
| 2169 | $1.161_{-5}$ | 1.05 | $1.281_{-7}$ | $1.132_{-5}$ | $2.565_{-5}$ | $9.569_{-6}$ |
| 2712 | $8.281_{-6}$ | 2.04 | $9.444_{-7}$ | $7.570_{-6}$ | $1.627_{-5}$ | $6.086_{-6}$ |
| 3390 | $6.968_{-6}$ | 1.35 | $7.371_{-7}$ | $5.621_{-6}$ | $1.172_{-5}$ | $4.383_{-6}$ |

Tabelle 7.11: Fehler $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.2.1 mit gestörter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

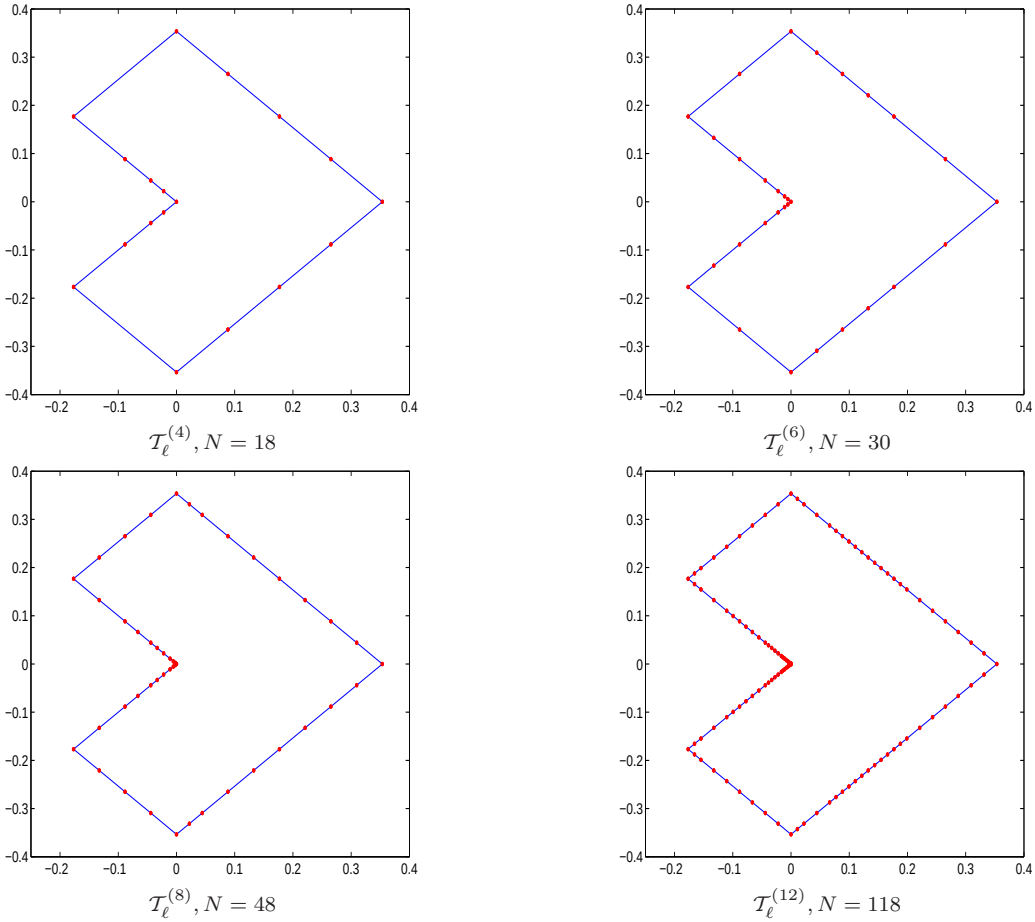


Abbildung 7.14: Ausschnitt der Folge von Partitionen, die durch $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Verfeinerungen im Laplaceproblem 7.2.1 hervorgeht.

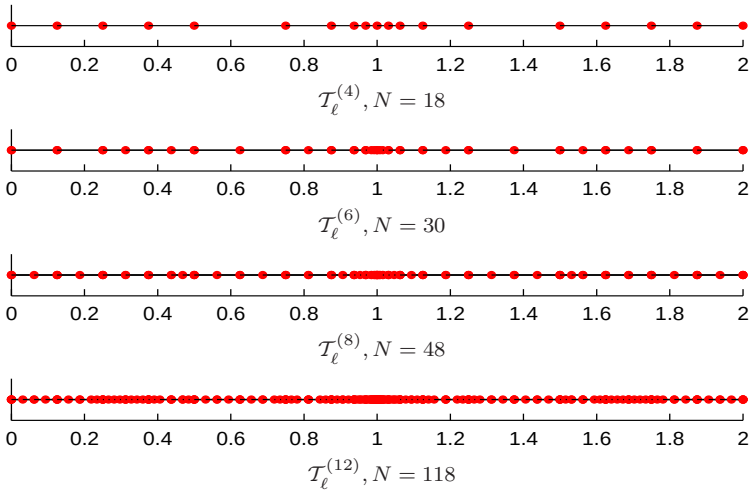


Abbildung 7.15: Ausschnitt der Folge von Partitionen, die durch $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Verfeinerungen im Laplaceproblem 7.2.1 hervorgeht, über der Bogenlänge$s \in [0,2]$, wobei $s \in \{0,2\}$ gerade dem Eckpunkt $(0.35355, 0)$ entspricht.

| $N_\ell$ | $\vert\!\vert\!\vert \phi - \Phi_\ell \vert\!\vert\!\vert$ | $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ | $\zeta_\ell$ | $\varrho_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\eta_\ell$ | $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|---|---|
| 8 | $8.355_{-2}$ | $8.715_{-2}$ | $3.822_{-3}$ | $1.290_{-1}$ | $1.292_{-1}$ | $6.632_{-2}$ | $6.759_{-2}$ |
| 10 | $5.209_{-2}$ | $5.581_{-2}$ | $3.822_{-3}$ | $8.017_{-2}$ | $7.996_{-2}$ | $4.089_{-2}$ | $4.153_{-2}$ |
| 14 | $3.466_{-2}$ | $3.686_{-2}$ | $2.578_{-3}$ | $5.341_{-2}$ | $5.330_{-2}$ | $2.789_{-2}$ | $2.867_{-2}$ |
| 18 | $2.257_{-2}$ | $2.468_{-2}$ | $2.043_{-3}$ | $3.554_{-2}$ | $3.525_{-2}$ | $1.836_{-2}$ | $1.888_{-2}$ |
| 24 | $1.361_{-2}$ | $1.464_{-2}$ | $1.358_{-3}$ | $2.084_{-2}$ | $2.078_{-2}$ | $1.088_{-2}$ | $1.100_{-2}$ |
| 30 | $9.352_{-3}$ | $1.002_{-2}$ | $9.250_{-4}$ | $1.384_{-2}$ | $1.378_{-2}$ | $7.699_{-3}$ | $7.824_{-3}$ |
| 38 | $6.159_{-3}$ | $6.743_{-3}$ | $6.743_{-4}$ | $9.118_{-3}$ | $9.053_{-3}$ | $5.120_{-3}$ | $5.190_{-3}$ |
| 48 | $3.954_{-3}$ | $4.186_{-3}$ | $4.619_{-4}$ | $5.813_{-3}$ | $5.815_{-3}$ | $3.309_{-3}$ | $3.347_{-3}$ |
| 60 | $2.733_{-3}$ | $2.956_{-3}$ | $3.668_{-4}$ | $3.953_{-3}$ | $3.939_{-3}$ | $2.332_{-3}$ | $2.369_{-3}$ |
| 75 | $1.882_{-3}$ | $2.035_{-3}$ | $2.305_{-4}$ | $2.663_{-3}$ | $2.624_{-3}$ | $1.634_{-3}$ | $1.658_{-3}$ |
| 94 | $1.198_{-3}$ | $1.261_{-3}$ | $1.672_{-4}$ | $1.693_{-3}$ | $1.698_{-3}$ | $1.042_{-3}$ | $1.056_{-3}$ |
| 118 | $8.856_{-4}$ | $9.439_{-4}$ | $1.199_{-4}$ | $1.187_{-3}$ | $1.188_{-3}$ | $7.866_{-4}$ | $7.973_{-4}$ |
| 148 | $6.192_{-4}$ | $6.610_{-4}$ | $8.020_{-5}$ | $8.057_{-4}$ | $8.042_{-4}$ | $5.560_{-4}$ | $5.636_{-4}$ |
| 185 | $3.822_{-4}$ | $4.006_{-4}$ | $5.990_{-5}$ | $5.217_{-4}$ | $5.243_{-4}$ | $3.425_{-4}$ | $3.478_{-4}$ |
| 232 | $2.959_{-4}$ | $3.129_{-4}$ | $4.179_{-5}$ | $3.756_{-4}$ | $3.768_{-4}$ | $2.692_{-4}$ | $2.727_{-4}$ |
| 290 | $2.238_{-4}$ | $2.340_{-4}$ | $2.562_{-5}$ | $2.640_{-4}$ | $2.642_{-4}$ | $2.055_{-4}$ | $2.075_{-4}$ |
| 363 | $1.317_{-4}$ | $1.368_{-4}$ | $2.127_{-5}$ | $1.722_{-4}$ | $1.733_{-4}$ | $1.205_{-4}$ | $1.224_{-4}$ |
| 454 | $1.019_{-4}$ | $1.067_{-4}$ | $1.508_{-5}$ | $1.255_{-4}$ | $1.261_{-4}$ | $9.392_{-5}$ | $9.517_{-5}$ |
| 568 | $7.920_{-5}$ | $8.184_{-5}$ | $9.111_{-6}$ | $8.907_{-5}$ | $8.936_{-5}$ | $7.388_{-5}$ | $7.401_{-5}$ |
| 710 | $4.731_{-5}$ | $4.865_{-5}$ | $7.521_{-6}$ | $5.902_{-5}$ | $5.946_{-5}$ | $4.374_{-5}$ | $4.439_{-5}$ |
| 888 | $3.622_{-5}$ | $3.752_{-5}$ | $5.447_{-6}$ | $4.354_{-5}$ | $4.383_{-5}$ | $3.356_{-5}$ | $3.400_{-5}$ |
| 1110 | $2.853_{-5}$ | $2.937_{-5}$ | $3.371_{-6}$ | $3.138_{-5}$ | $3.150_{-5}$ | $2.646_{-5}$ | $2.669_{-5}$ |
| 1388 | $1.753_{-5}$ | $1.788_{-5}$ | $2.662_{-6}$ | $2.085_{-5}$ | $2.102_{-5}$ | $1.614_{-5}$ | $1.636_{-5}$ |
| 1735 | $1.311_{-5}$ | $1.346_{-5}$ | $2.010_{-6}$ | $1.547_{-5}$ | $1.559_{-5}$ | $1.198_{-5}$ | $1.215_{-5}$ |
| 2169 | $1.049_{-5}$ | $1.074_{-5}$ | $1.281_{-6}$ | $1.126_{-5}$ | $1.132_{-5}$ | $9.481_{-6}$ | $9.569_{-6}$ |
| 2712 | $6.935_{-6}$ | $7.027_{-6}$ | $9.444_{-7}$ | $7.512_{-5}$ | $7.570_{-6}$ | $6.012_{-6}$ | $6{,}086_{-6}$ |
| 3390 | $5.320_{-6}$ | $5.413_{-6}$ | $7.371_{-7}$ | $5.547_{-6}$ | $5.621_{-6}$ | $4.320_{-6}$ | $4.383_{-6}$ |

Tabelle 7.12: Fehler $\vert\!\vert\!\vert \phi - \Phi_\ell \vert\!\vert\!\vert$, $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ sowie Fehlerschätzer $\zeta_\ell$, $\widetilde{\varrho}_\ell$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\widetilde{\eta}_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ für eine $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.2.1 mit gestörter rechter Seite.

Wie bereits in Kapitel 5.5.2 erklärt, ist die Berechnung des Residualschätzers im Vergleich zur Berechnung der $h-h/2$-Fehlerschätzer sehr aufwändig. Die Kurven der Fehlerschätzer verlaufen aber — nach einer anfänglichen präasymptotischen Phase — parallel, was ein Ersetzen von $\widetilde{\varrho}_\ell$ durch $\widetilde{\mu}_\ell$ aus numerischer Sicht nahe legt.
In Abbildung 7.14 ist ein Ausschnitt einer Folge von adaptiven Verfeinerungen über der Zahl der Elemente der Partition $\mathcal{T}_\ell$ zu sehen. Man kann erkennen, dass besonders zur einspringenden Ecke — bei der sich ja die Singularität befindet — verfeinert wird. Abbildung 7.15 stellt dieselbe Netzfolge über der Bogenlänge dar.

Um die Berechnung mit exakter rechter Seite und gestörter rechter Seite besser vergleichen zu können, sind in Abbildung 7.17 sowohl der exakte Fehler $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ und die Fehlerschätzer

$$\begin{aligned}
&\zeta_\ell = \left\| h_\ell^{1/2} (g - g_\ell)' \right\|_{L^2(\Gamma)}, \\
&\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\varrho}_\ell = \left\| h_\ell^{1/2} \big( (K + \tfrac{1}{2}) g_\ell - V \widetilde{\Phi}_\ell \big)' \right\|, \\
&\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\eta}_\ell = \vert\!\vert\!\vert \widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell \vert\!\vert\!\vert
\end{aligned}$$

| $N_\ell$ | $\mathrm{cond}(A)$ | $\mathrm{cond}(A_{\mathrm{eq}})$ | $\mathrm{cond}(A_{\mathrm{diag}})$ |
|---|---|---|---|
| 8 | $2.8_1$ | $2.9_1$ | $2.8_1$ |
| 10 | $9.0_1$ | $6.0_1$ | $3.8_1$ |
| 14 | $3.0_2$ | $1.3_2$ | $5.9_1$ |
| 18 | $9.6_2$ | $2.6_2$ | $8.3_1$ |
| 24 | $2.8_3$ | $5.2_2$ | $1.2_2$ |
| 30 | $9.6_3$ | $1.1_3$ | $1.6_2$ |
| 38 | $2.8_4$ | $2.1_3$ | $2.1_2$ |
| 48 | $9.3_4$ | $4.3_3$ | $2.8_2$ |
| 60 | $3.1_5$ | $8.6_3$ | $3.6_2$ |
| 75 | $9.8_5$ | $1.7_4$ | $4.7_2$ |
| 94 | $3.0_6$ | $3.4_4$ | $6.2_2$ |
| 118 | $1.0_7$ | $7.0_4$ | $7.9_2$ |
| 148 | $3.3_7$ | $1.4_5$ | $1.0_3$ |
| 185 | $9.9_7$ | $2.7_5$ | $1.3_3$ |
| 232 | $3.3_8$ | $5.6_5$ | $1.7_3$ |
| 290 | $1.0_9$ | $1.1_6$ | $2.1_3$ |
| 363 | $3.2_9$ | $2.2_6$ | $2.8_3$ |
| 454 | $1.1_{10}$ | $4.4_6$ | $3.5_3$ |
| 568 | $3.5_{10}$ | $8.8_6$ | $4.4_3$ |
| 710 | $1.0_{11}$ | $1.7_7$ | $5.7_3$ |
| 888 | $3.5_{11}$ | $3.5_7$ | $7.2_3$ |
| 1110 | $1.1_{12}$ | $7.1_7$ | $9.0_3$ |
| 1388 | $3.4_{12}$ | $1.4_8$ | $1.2_4$ |
| 1735 | $1.2_{13}$ | $2.8_8$ | $1.5_4$ |
| 2169 | $3.7_{13}$ | $5.7_8$ | $1.8_4$ |
| 2712 | $1.1_{14}$ | $1.1_9$ | $2.4_4$ |
| 3390 | $3.7_{14}$ | $2.3_9$ | $3.0_4$ |

Tabelle 7.13: Konditionszahlen der Steifigkeitsmatrix ohne Vorkonditionierung, mit zeilenäquilibrierter Vorkonditionierung und mit diagonaler Vorkonditionierung für eine $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.2.1.

| $N_\ell$ | $\kappa(\mathcal{T}_\ell)$ | $\frac{h_{\max}}{h_{\min}}$ |
|---|---|---|
| 8 | 1 | $1.0_0$ |
| 10 | 2 | $2.0_0$ |
| 14 | 2 | $4.0_0$ |
| 18 | 2 | $8.0_0$ |
| 24 | 2 | $8.0_0$ |
| 30 | 2 | $1.6_1$ |
| 38 | 2 | $3.2_1$ |
| 48 | 2 | $3.2_1$ |
| 60 | 2 | $6.4_1$ |
| 75 | 2 | $1.3_2$ |
| 94 | 2 | $1.3_2$ |
| 118 | 2 | $2.6_2$ |
| 148 | 2 | $5.1_2$ |
| 185 | 2 | $5.1_2$ |
| 232 | 2 | $1.0_3$ |
| 290 | 2 | $2.0_3$ |
| 363 | 2 | $4.1_3$ |
| 454 | 2 | $4.1_3$ |
| 568 | 2 | $8.2_3$ |
| 710 | 2 | $1.6_4$ |
| 888 | 2 | $1.6_4$ |
| 1110 | 2 | $3.3_4$ |
| 1388 | 2 | $6.6_4$ |
| 1735 | 2 | $6.6_4$ |
| 2169 | 2 | $1.3_5$ |
| 2712 | 2 | $2.6_5$ |
| 3390 | 2 | $2.6_5$ |

Tabelle 7.14: Anzahl der Elemente $N_\ell$ in $\mathcal{T}_\ell$, $\kappa(\mathcal{T}_\ell)$ und $\frac{h_{\max}}{h_{\min}}$ für das Laplaceproblem 7.2.1 bei Rechnung wie in Tabelle 7.13.

für die approximierte rechte Seite als auch der exakte Fehler $\left|\!\left|\!\left|\phi-\Phi_\ell\right|\!\right|\!\right|$ und die Fehlerschätzer

$$\varrho_\ell=\left\|h_\ell^{1/2}\big((K+\tfrac{1}{2})g-V\Phi_\ell\big)'\right\|_{L^2(\Gamma)},\qquad \eta_\ell=\left|\!\left|\!\left|\widehat{\Phi}_\ell-\Phi_\ell\right|\!\right|\!\right|$$

für die exakte rechte Seite angegeben. Die Steuerung des adaptiven Algorithmus erfolgte mittels $\omega_\ell=\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$ und $\theta=\frac{1}{4}$. Man sieht, dass sich die Fehler und auch Fehlerschätzer kaum unterscheiden. Aus diesem Grund enthält Tabelle 7.12 die genauen Größen des Fehlers und der Fehlerschätzer.

Aufgrund von Rechenfehlern verschlechtert sich mit steigender Anzahl von Iterationen die Kondition des Gleichungssystem. Aus diesem Grund wird, wie bereits in Abschnitt 7.1 beschrieben, diagonal oder mittels Zeilenäquilibrierung vorkonditioniert.

Tabelle 7.13 und Abbildung 7.18 enthalten die Anzahl der Elemente $N_\ell$ und die Konditionszahlen der unveränderten Matrix $A$, der mittels Zeilenäquilibrierung vorkonditionierten Matrix $A_{\mathrm{eq}}$ und der diagonal vorkonditionierten Matrix $A_{\mathrm{diag}}$.

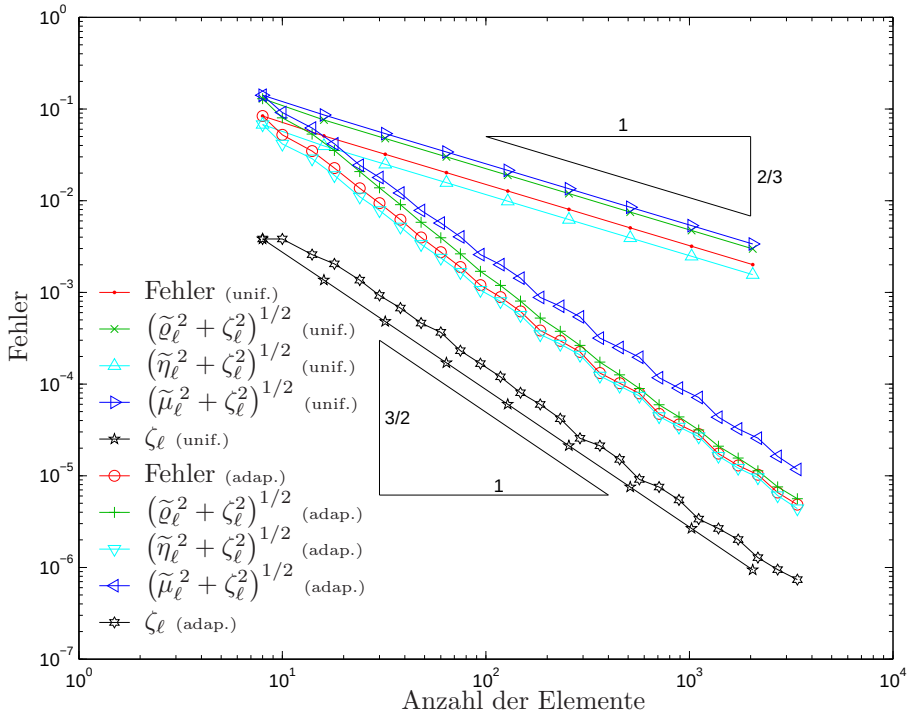


Abbildung 7.16: Fehler $\left|\!\left|\!\left|\phi-\widetilde{\Phi}_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\eta}_\ell^2+\zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\mu}_\ell^2+\zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$ für eine uniforme und eine $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.2.1 mit gestörter rechter Seite.

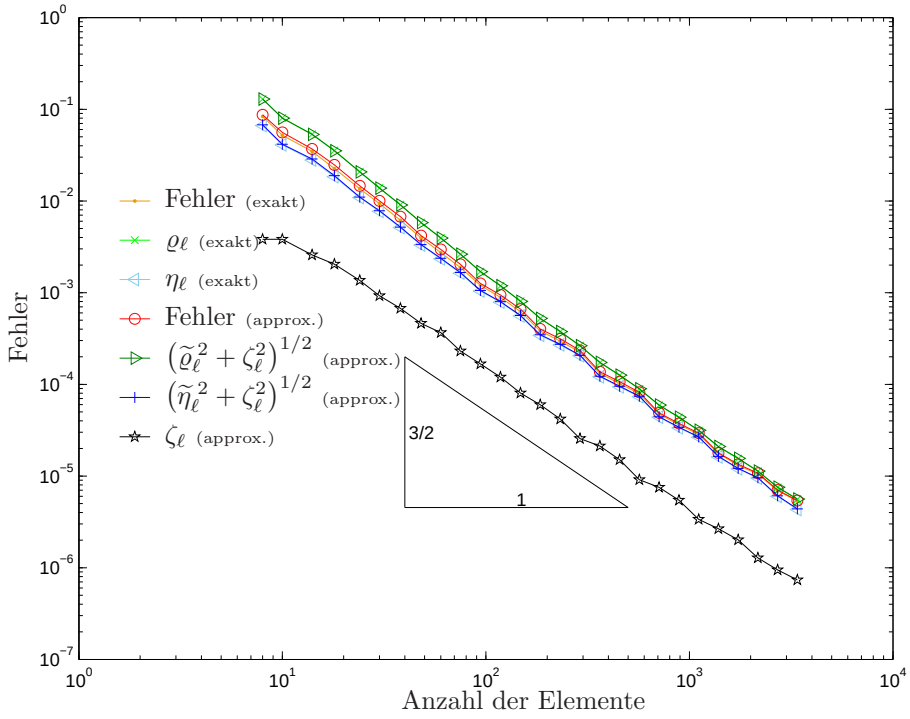


Abbildung 7.17: Fehler $\left|\!\left|\!\left|\phi-\Phi_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\eta_\ell$, $\varrho_\ell$ für eine Berechnung mit exakter rechter Seite und Fehler $\left|\!\left|\!\left|\phi-\widetilde{\Phi}_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\eta}_\ell^2+\zeta_\ell^2\right)^{1/2}$ für eine Berechnung mit approximativer rechter Seite des Laplaceproblems 7.2.1 und $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptiver Verfeinerung.

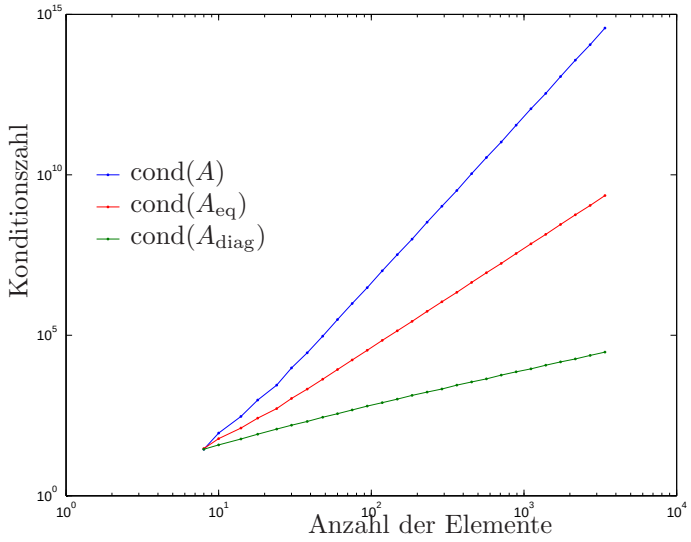


Abbildung 7.18: Konditionszahlen der Steifigkeitsmatrix ohne Vorkonditionierung, mit zeilenäquilibrierter Vorkonditionierung und mit diagonaler Vorkonditionierung für eine $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.2.1 über der Anzahl der Elemente.

Auch für dieses Beispiel wollen wir wieder die Beschränktheit von $\kappa(\mathcal{T}_\ell)$ numerisch überprüfen. Tabelle 7.14 bestätigt die Aussage von Korollar 3.11. Zum Vergleich ist wiederum der maximale Quotient der lokalen Netzweiten von beliebigen, also nicht notwendigerweise benachbarten Elementen, d.h $\frac{h_{\max}}{h_{\min}}$, angegeben.

### 7.2.2 Zweites Experiment

Wie man an der Abbildung 7.17 erkennen kann, ist der Datenfehler von Anfang an wesentlich kleiner als der Verfahrensfehler. Aus diesem Grund wird auf dem L-förmigen Gebiet eine weitere Rechnung durchgeführt, aber mit veränderten Dirichletdaten

$$g(x,y)=g(r\cos\psi,r\sin\psi)=r^{2/3}\cos\tfrac{2}{3}\psi+\begin{cases}\sin\left(8\sqrt{2}\pi x\right), & \text{für } x>0,|y|\leq\frac{1}{4\sqrt{2}}\\ 0 & \text{sonst}\end{cases}$$

die in Abbildung 7.19 zu sehen sind. Man kann erkennen, dass die Datenoszillationen im Vergleich zu Abbildung 7.11 im Bereich um die Ecke $(0.35355,0)$ groß sind, um einen großen Datenfehler zu erzeugen.

Das Aitkensche $\Delta^2$-Verfahren liefert eine exakte Energienorm von $\left|\!\left|\!\left|\phi\right|\!\right|\!\right|^2=2.692689154904$.

Wie schon zuvor wird die Symmsche Integralgleichung mit approximierter und exakter rechter Seite gelöst. Die Steuerung des Algorithmus erfolgt mittels des Gesamtfehlerschätzers $\omega_\ell=\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$ und $\theta=\frac{1}{4}$.

Die Anzahl der Elemente $N_\ell$, die experimentelle Konvergenzrate sowie sowohl der exakte Fehler $\left|\!\left|\!\left|\phi-\Phi_\ell\right|\!\right|\!\right|$ und der Fehlerschätzer

$$\varrho_\ell=\left\|h_\ell^{1/2}\left((K+\tfrac{1}{2})g-V\Phi_\ell\right)'\right\|_{L^2(\Gamma)}$$

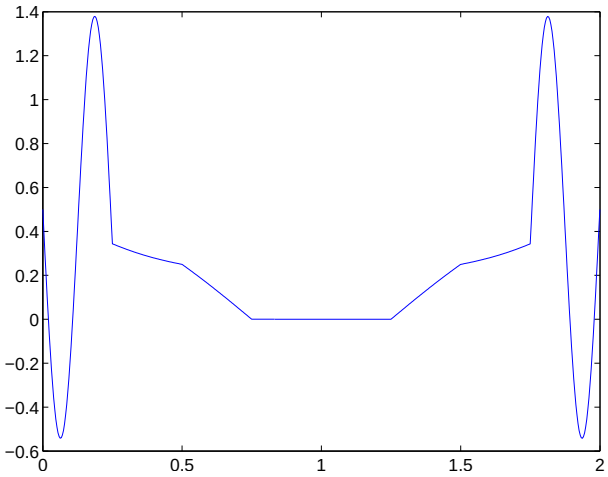


Abbildung 7.19: Exakte Dirichletdaten des Laplaceproblems 7.2.2 über der Bogenlänge $s \in [0,2]$, wobei $s \in \{0,2\}$ gerade dem Eckpunkt $(0.35355, 0)$ entspricht.

| $N_\ell$ | $\lvert\!\lvert\!\lvert \phi - \Phi_\ell \rvert\!\rvert\!\rvert$ | $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ | $\zeta_\ell$ | $\varrho_\ell$ | $\big(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\big)^{1/2}$ |
|---|---|---|---|---|---|
| 8 | $1.552_{0}$ | $1.556_{0}$ | $1.232_{0}$ | $1.706_{-1}$ | $1.238_{0}$ |
| 16 | $8.739_{-1}$ | $9.150_{-1}$ | $3.880_{-1}$ | $1.071_{0}$ | $9.296_{1}$ |
| 32 | $6.686_{-1}$ | $6.725_{-1}$ | $1.432_{-1}$ | $1.041_{0}$ | $1.046_{0}$ |
| 64 | $2.476_{-1}$ | $2.484_{-1}$ | $5.119_{-2}$ | $4.103_{-1}$ | $4.149_{-1}$ |
| 128 | $1.025_{-1}$ | $1.026_{-1}$ | $1.815_{-2}$ | $1.643_{-1}$ | $1.657_{-1}$ |
| 256 | $4.669_{-2}$ | $4.670_{-2}$ | $6.420_{-3}$ | $7.178_{-2}$ | $7.215_{-2}$ |
| 512 | $2.260_{-2}$ | $2.260_{-2}$ | $2.270_{-3}$ | $3.379_{-2}$ | $3.388_{-2}$ |
| 1024 | $1.135_{-2}$ | $1.135_{-2}$ | $8.026_{-4}$ | $1.675_{-2}$ | $1.677_{-2}$ |
| 2048 | $5.840_{-3}$ | $5.840_{-3}$ | $2.838_{-4}$ | $8.616_{-3}$ | $8.621_{-3}$ |

Tabelle 7.15: Fehler $\lvert\!\lvert\!\lvert \phi - \Phi_\ell \rvert\!\rvert\!\rvert$, $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ sowie Fehlerschätzer $\zeta_\ell$, $\widetilde{\varrho}_\ell$, $\big(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\big)^{1/2}$ für eine uniforme Netzfolge im Laplaceproblem 7.2.2 mit gestörter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

für die exakte rechte Seite als auch der exakte Fehler $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ und die Fehlerschätzer

$$\zeta_\ell = \big\| h_\ell^{1/2} (g - g_\ell)' \big\|_{L^2(\Gamma)},$$
$$\big(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\big)^{1/2} \quad \text{mit } \widetilde{\varrho}_\ell = \big\| h_\ell^{1/2} \big( (K + \tfrac{1}{2}) g_\ell - V \widetilde{\Phi}_\ell \big)' \big\|,$$

für die approximierte rechte Seite sind in der Tabelle 7.15 für eine uniforme und in Tabelle 7.16 für eine $\big(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\big)^{1/2}$-adaptive Netzfolge angegeben.

Abbildung 7.20 zeigt den exakten Fehler und die Fehlerschätzer in einem doppelt logarithmischen Diagramm. Bei uniformer Verfeinerung wird entsprechend der Ordnung der Singularität der Lösung eine Konvergenzordnung von $\alpha = \frac{2}{3}$ erreicht. Die präasymptotische Phase dauert hier besonders lange. Man kann erkennen, dass der exakte Fehler, solange der Datenfehler noch verhältnismäßig groß ist, mit der Ordnung $\alpha = \frac{3}{2}$ des Datenfehlerschätzers fällt, und erst später flacher wird.

Mittels eines adaptiven Algorithmus — zu sehen in Abbildung 7.21 — lässt sich die Singularität und der Datenfehler besser auflösen und man erreicht die optimale Konvergenzordnung von $\alpha = \frac{3}{2}$.

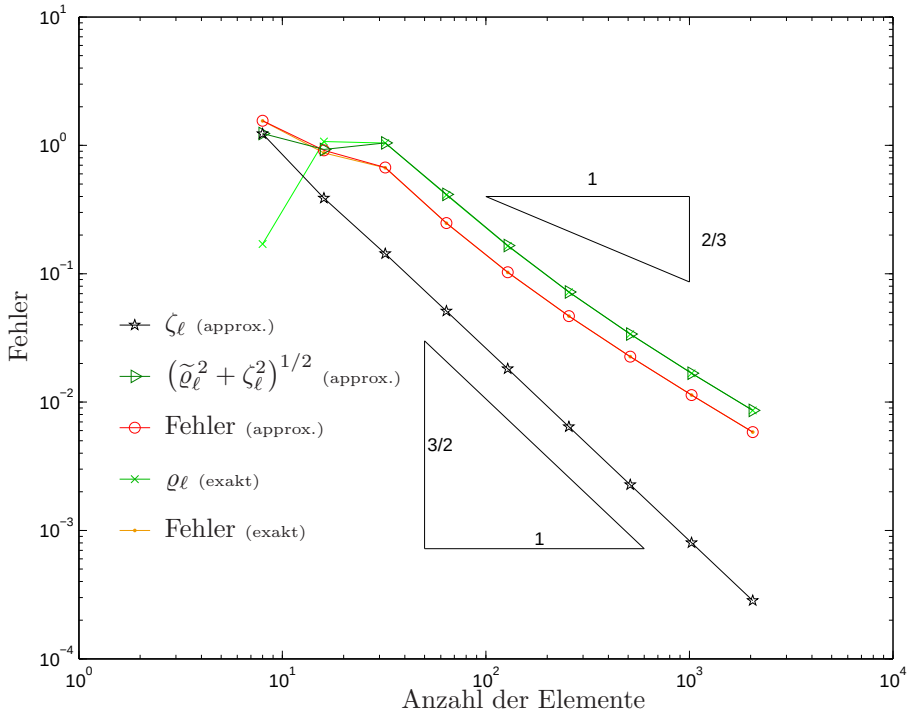


Abbildung 7.20: Fehler $\left|\!\left|\!\left|\phi-\Phi_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\varrho_\ell$ für eine Berechnung mit exakter rechte Seite und Fehler $\left|\!\left|\!\left|\phi-\widetilde{\Phi}_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$ für eine Berechnung mit approximativer rechter Seite des Laplaceproblems 7.2.2 bei uniformer Verfeinerung.

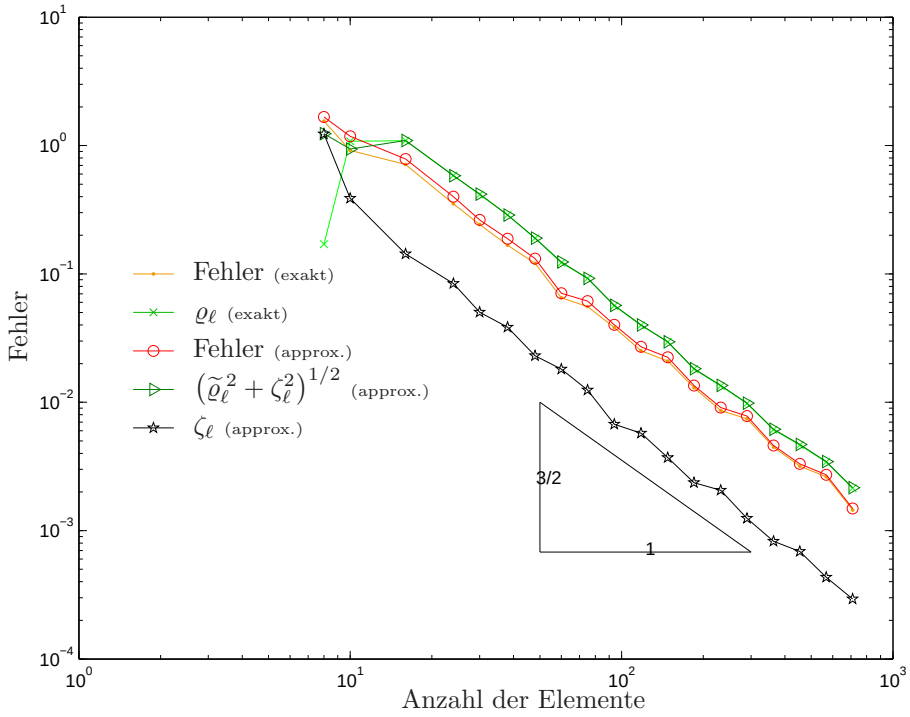


Abbildung 7.21: Fehler $\left|\!\left|\!\left|\phi-\Phi_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\varrho_\ell$ für eine Berechnung mit exakter rechte Seite und Fehler $\left|\!\left|\!\left|\phi-\widetilde{\Phi}_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$ für eine Berechnung mit approximativer rechter Seite des Laplaceproblems 7.2.2 bei $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptiver Verfeinerung.

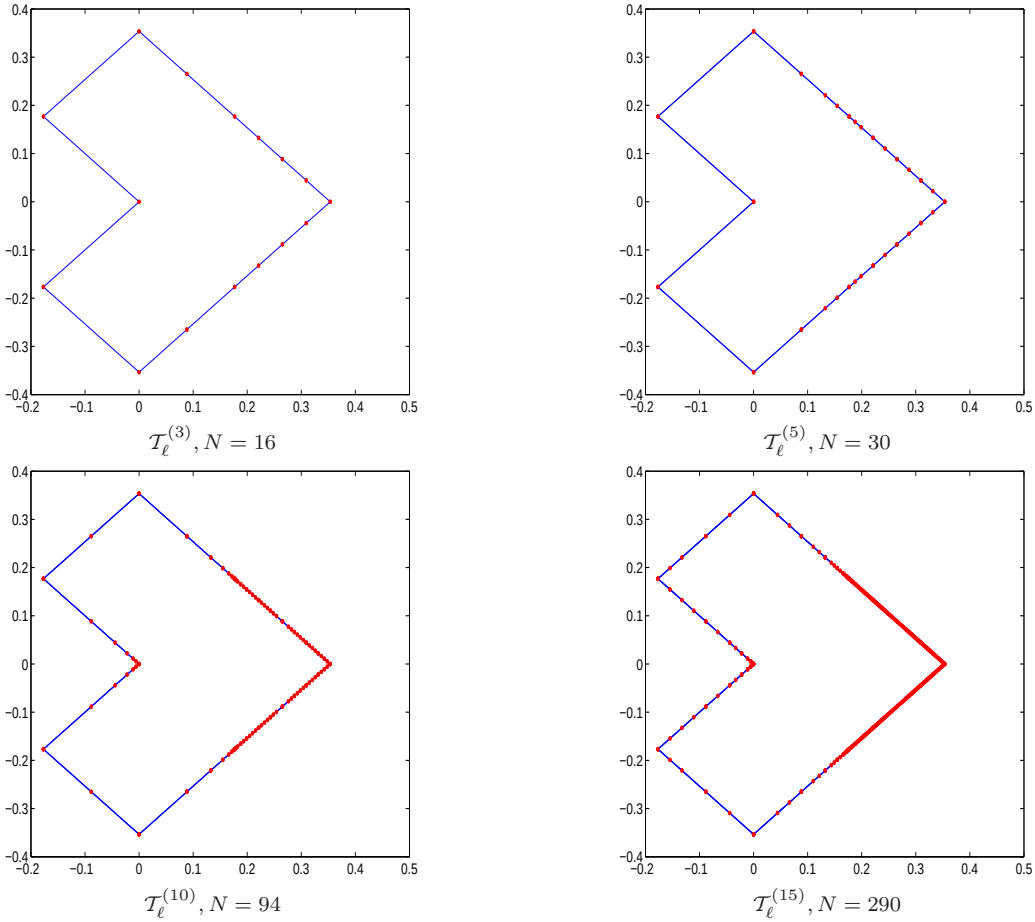


Abbildung 7.22: Ausschnitt der Folge von Partitionen, die durch $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Verfeinerungen im Laplaceproblem 7.2.2 hervorgeht.

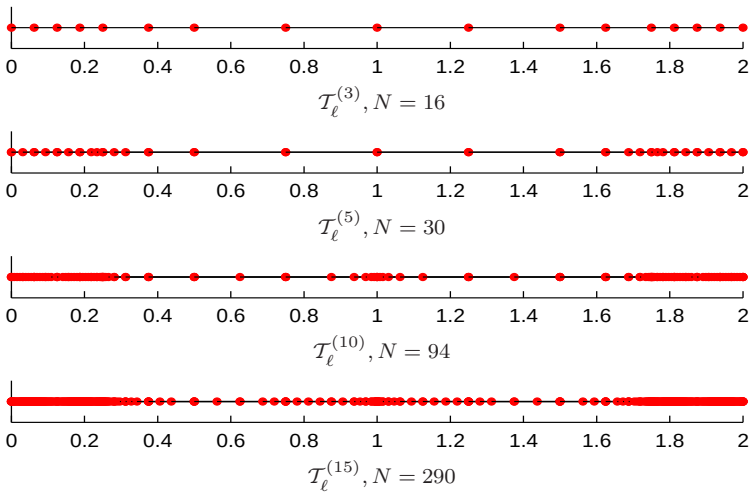


Abbildung 7.23: Ausschnitt der Folge von Partitionen, die durch $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Verfeinerungen im Laplaceproblem 7.2.2 hervorgeht, über der Bogenlänge$s \in [0,2]$, wobei $s \in \{0,2\}$ gerade dem Eckpunkt $(0.35355, 0)$ entspricht.

| $N_\ell$ | $\lvert\!\lvert\!\lvert \phi - \Phi_\ell \rvert\!\rvert\!\rvert$ | $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ | $\zeta_\ell$ | $\varrho_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|
| 8 | $1.552_{0}$ | $1.667_{0}$ | $1.232_{0}$ | $1.706_{1}$ | $1.238_{0}$ |
| 10 | $9.157_{-1}$ | $1.182_{0}$ | $3.880_{-1}$ | $1.080_{0}$ | $9.395_{-1}$ |
| 16 | $7.124_{-1}$ | $7.840_{-1}$ | $1.432_{-1}$ | $1.093_{0}$ | $1.095_{0}$ |
| 24 | $3.513_{-1}$ | $3.995_{-1}$ | $8.420_{-2}$ | $5.846_{-1}$ | $5.821_{-1}$ |
| 30 | $2.426_{-1}$ | $2.632_{-1}$ | $5.024_{-2}$ | $4.146_{-1}$ | $4.180_{-1}$ |
| 38 | $1.669_{-1}$ | $1.879_{-1}$ | $3.844_{-2}$ | $2.879_{-1}$ | $2.873_{-1}$ |
| 48 | $1.209_{-1}$ | $1.317_{-1}$ | $2.301_{-2}$ | $1.895_{-1}$ | $1.902_{-1}$ |
| 60 | $6.520_{-2}$ | $7.076_{-2}$ | $1.814_{-2}$ | $1.223_{-1}$ | $1.239_{-1}$ |
| 75 | $5.571_{-2}$ | $6.130_{-2}$ | $1.243_{-2}$ | $9.224_{-2}$ | $9.243_{-2}$ |
| 94 | $3.820_{-2}$ | $4.015_{-2}$ | $6.736_{-3}$ | $5.639_{-2}$ | $5.684_{-2}$ |
| 118 | $2.522_{-2}$ | $2.707_{-2}$ | $5.743_{-3}$ | $3.976_{-2}$ | $4.007_{-2}$ |
| 148 | $2.093_{-2}$ | $2.242_{-2}$ | $3.709_{-3}$ | $2.952_{-2}$ | $2.958_{-2}$ |
| 185 | $1.299_{-2}$ | $1.394_{-2}$ | $2.367_{-3}$ | $1.810_{-2}$ | $1.827_{-2}$ |
| 232 | $8.607_{-3}$ | $9.092_{-3}$ | $2.061_{-3}$ | $1.340_{-2}$ | $1.354_{-2}$ |
| 290 | $7.440_{-3}$ | $7.810_{-3}$ | $1.284_{-3}$ | $9.798_{-3}$ | $9.859_{-3}$ |
| 363 | $4.464_{-3}$ | $4.606_{-3}$ | $8.256_{-4}$ | $6.111_{-3}$ | $6.166_{-3}$ |
| 454 | $3.183_{-3}$ | $3.311_{-3}$ | $6.884_{-4}$ | $4.641_{-3}$ | $4.689_{-3}$ |
| 568 | $2.627_{-3}$ | $2.725_{-3}$ | $4.336_{-4}$ | $3.426_{-3}$ | $3.451_{-3}$ |
| 710 | $1.448_{-3}$ | $1.488_{-3}$ | $2.935_{-4}$ | $2.144_{-3}$ | $2.163_{-3}$ |

Tabelle 7.16: Fehler $\lvert\!\lvert\!\lvert \phi - \Phi_\ell \rvert\!\rvert\!\rvert$, $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ sowie Fehlerschätzer $\zeta_\ell$, $\widetilde{\varrho}_\ell$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ für eine $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.2.2 mit gestörter rechter Seite.

| $N_\ell$ | $h_{\ell,\min}$ um $(0,0)$ | $h_{\ell,\min}$ um $(0.35355, 0)$ |
|---|---|---|
| 8 | $2.5_{-1}$ | $2.5_{-1}$ |
| 10 | $2.5_{-1}$ | $1.2_{-1}$ |
| 16 | $2.5_{-1}$ | $6.2_{-2}$ |
| 24 | $2.5_{-1}$ | $3.1_{-2}$ |
| 30 | $2.5_{-1}$ | $1.6_{-2}$ |
| 38 | $1.2_{-1}$ | $1.6_{-2}$ |
| 48 | $6.2_{-2}$ | $7.8_{-3}$ |
| 60 | $3.1_{-2}$ | $3.9_{-3}$ |
| 75 | $1.6_{-2}$ | $3.9_{-3}$ |
| 94 | $7.8_{-3}$ | $2.0_{-3}$ |
| 118 | $3.9_{-3}$ | $9.8_{-4}$ |
| 148 | $2.0_{-3}$ | $9.8_{-4}$ |
| 185 | $9.8_{-4}$ | $4.9_{-4}$ |
| 232 | $4.9_{-4}$ | $2.4_{-4}$ |
| 290 | $2.4_{-4}$ | $2.4_{-4}$ |
| 363 | $1.2_{-4}$ | $1.2_{-4}$ |
| 454 | $6.1_{-5}$ | $6.1_{-5}$ |
| 568 | $3.1_{-5}$ | $6.1_{-5}$ |
| 710 | $1.5_{-5}$ | $3.1_{-5}$ |

Tabelle 7.17: Kleinstes Element im Bereich um die Ecke $(0,0)$ im Vergleich mit dem kleinsten Element um die Ecke $(0.35355, 0)$ für eine adaptive Netzfolge im Laplaceproblem 7.2.2 mit gestörter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

Zur Veranschaulichung des adaptiven Algorithmus werden auch für dieses Beispiel Netzfolgen in den Abbildungen 7.22 und 7.23 dargestellt. Man erkennt deutlich, dass zunächst eine Verfeinerung im Bereich der Ecke $(0.35355, 0)$, wo der Datenfehler groß ist, stattfindet und erst später auch um die Singularität bei $(0, 0)$ adaptiv verfeinert wird. Um dies noch deutlicher zu machen, sind in Tabelle 7.17, die Größe des kleinsten Elements bei der Singularität und zum Vergleich die Größe des kleinsten Elements im Bereich der Ecke mit dem großen Datenfehler aufgelistet.

## 7.3 Laplaceproblem am Z-förmigen Gebiet

Im letzten numerischen Beispiel betrachten wir das Laplaceproblem (7.1) auf einem Z-förmigen Gebiet mit $\operatorname{diam}(\Omega) = \sqrt{2}/2 \approx 0.7071 < 1$ mit der exakten Lösung $u(x, y) = u(r\cos\psi, r\sin\psi) = r^{4/7}\cos\frac{4}{7}\psi$ und lösen die äquivalente Symmsche Integralgleichung $V\phi = \big(K + \frac{1}{2}\big)g$ mit $g = u|_\Gamma$.

Die Dirichletdaten $g \in H^{1/2}(\Gamma)$ sind in Abbildung 7.24 über der Bogenlänge $0 \le s < 2$ dargestellt, wobei $s \in \{0, 2\}$ gerade dem Eckpunkt $(0.35355, 0) \in \mathbb{R}^2$ entspricht.

Die exakte Lösung der Symmschen Integralgleichung $\phi = \frac{\partial u}{\partial n}$ hat dann eine Singularität in der einspringenden Ecke $(0, 0)$ und lässt sich schreiben in der Form

$$\phi(x, y) = \phi(r\cos\psi, r\sin\psi) = \frac{4}{7} r^{-3/7} \begin{pmatrix} \cos\psi\cos(\frac{4}{7}\psi) + \sin\psi\sin(\frac{4}{7}\psi) \\ \sin\psi\cos(\frac{4}{7}\psi) - \cos\psi\sin(\frac{4}{7}\psi) \end{pmatrix} \cdot n(x, y),$$

mit dem äußeren Normalenvektor $n(x, y)$. Zur Veranschaulichung ist $\phi$ dargestellt in Abbildung 7.25.

Das verwendete Startnetz $\mathcal{T}_0$ mit $N_0 = 9$ Elementen ist in Abbildung 7.26 zu sehen.

Extrapolation mit der Aitkenschen $\triangle^2$-Methode führt zu einer geschätzten Energienorm $\big|\!\big|\!\big|\phi\big|\!\big|\!\big|^2 = 0.2606507012499$ der unbekannten Lösung $\phi$.

Zuerst werden wieder die Daten bei Lösung der Symmschen Integralgleichung mit exakter rechter Seite, das heißt exakten Dirichletdaten, angegeben.

Abbildung 7.13 zeigt den exakten Fehler $\big|\!\big|\!\big|\phi - \Phi_\ell\big|\!\big|\!\big|$ und die Fehlerschätzer

$$\varrho_\ell = \big\|h_\ell^{1/2}\big((K + \tfrac{1}{2})g - V\Phi_\ell\big)'\big\|_{L^2(\Gamma)}, \qquad \mu_\ell = \big\|h_\ell^{1/2}\big(\widehat{\Phi}_\ell - \Phi_\ell\big)\big\|_{L^2(\Gamma)}, \qquad \eta_\ell = \big|\!\big|\!\big|\widehat{\Phi}_\ell - \Phi_\ell\big|\!\big|\!\big|$$

bei uniformer bzw. $\varrho_\ell$-adaptiver Verfeinerung mit $\theta = \frac{1}{4}$. Die gezeichneten Größen, sowie die Anzahl der Elemente $N_\ell$ und die experimentell bestimmte Konvergenzrate $\alpha_\ell$ sind in den Tabellen 7.18 und 7.19 genau aufgelistet.

Man erkennt, dass uniforme Verfeinerung Konvergenzordnung $\alpha = \frac{4}{7} \approx 0.57$ liefert und nur bei adaptiver Verfeinerung die optimale Konvergenzordnung von $\alpha = \frac{3}{2}$ erreicht wird. Da alle Fehlerschätzer zum exakten Fehler parallel sind, kann davon ausgegangen werden, dass die Fehlerschätzer zuverlässig und effizient sind.

Nun wollen wir die Symmsche Integralgleichung mit gestörter rechter Seite, also nodal interpolierten Dirichletdaten, betrachten.

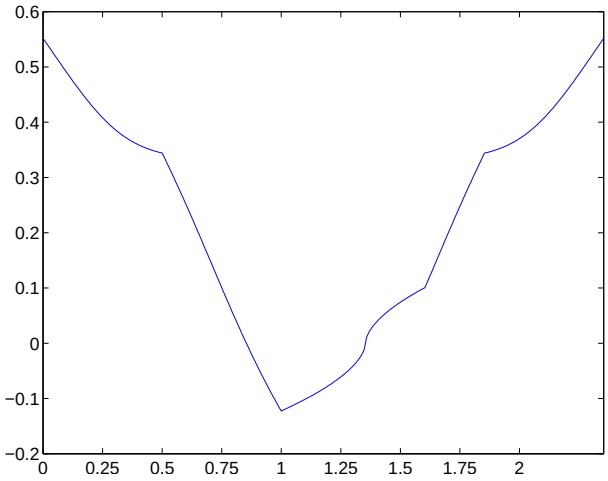


Abbildung 7.24: Exakte Dirichletdaten des Laplaceproblems 7.3 über der Bogenlänge $s \in [0, 2]$, wobei $s \in \{0, 2\}$ gerade dem Eckpunkt $(0.35355, 0)$ entspricht.

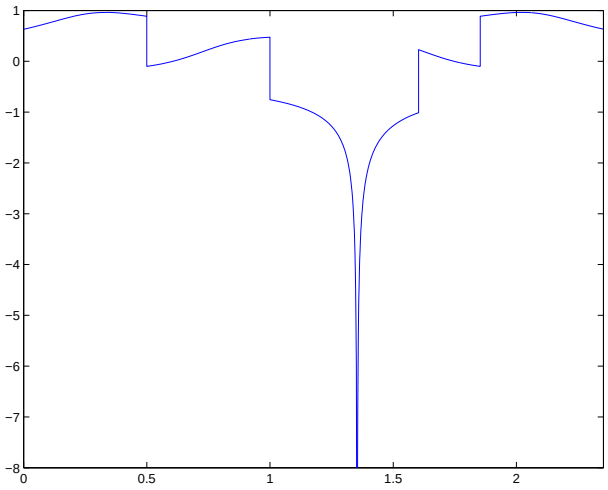


Abbildung 7.25: Exakte Lösung des Laplaceproblems 7.3 über der Bogenlänge $s \in [0, 2]$, wobei $s \in \{0, 2\}$ gerade dem Eckpunkt $(0.35355, 0)$ entspricht.

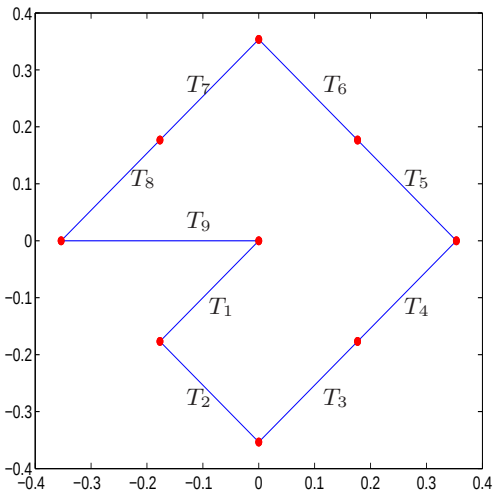


Abbildung 7.26: Startnetz $\mathcal{T}_0$ mit $N_0 = 9$ Elementen des Laplaceproblems 7.3.

| $N_\ell$ | $\vert\!\vert\!\vert \phi - \Phi_\ell \vert\!\vert\!\vert$ | $\alpha_\ell$ | $\eta_\ell$ | $\mu_\ell$ | $\varrho_\ell$ |
|---|---|---|---|---|---|
| 9 | $1.449_{-1}$ | – | $1.075_{-1}$ | $2.326_{-1}$ | $2.285_{-1}$ |
| 18 | $9.723_{-2}$ | 0.58 | $7.165_{-2}$ | $1.521_{-1}$ | $1.535_{-1}$ |
| 35 | $6.572_{-2}$ | 0.56 | $4.855_{-2}$ | $1.024_{-1}$ | $1.040_{-1}$ |
| 72 | $4.430_{-2}$ | 0.57 | $3.275_{-2}$ | $6.901_{-2}$ | $7.018_{-2}$ |
| 144 | $2.984_{-2}$ | 0.57 | $2.206_{-2}$ | $4.648_{-2}$ | $4.730_{-2}$ |
| 288 | $2.009_{-2}$ | 0.57 | $1.486_{-2}$ | $3.129_{-2}$ | $3.185_{-2}$ |
| 576 | $1.352_{-2}$ | 0.57 | $9.999_{-3}$ | $2.106_{-2}$ | $2.144_{-2}$ |
| 1152 | $9.098_{-3}$ | 0.57 | $6.730_{-3}$ | $1.417_{-2}$ | $1.443_{-2}$ |
| 2304 | $6.123_{-3}$ | 0.57 | $4.529_{-3}$ | $9.539_{-3}$ | $9.712_{-3}$ |

Tabelle 7.18: Fehler $\vert\!\vert\!\vert \phi - \Phi_\ell \vert\!\vert\!\vert$ und Fehlerschätzer $\mu_\ell$, $\varrho_\ell$, $\eta_\ell$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine uniforme Netzfolge im Laplaceproblem 7.3 mit exakter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

| $N_\ell$ | $\vert\!\vert\!\vert \phi - \Phi_\ell \vert\!\vert\!\vert$ | $\alpha_\ell$ | $\eta_\ell$ | $\mu_\ell$ | $\varrho_\ell$ |
|---|---|---|---|---|---|
| 9 | $1.449_{-1}$ | – | $1.075_{-1}$ | $2.326_{-1}$ | $2.285_{-1}$ |
| 12 | $9.797_{-2}$ | 1.36 | $7.255_{-2}$ | $1.547_{-1}$ | $1.562_{-1}$ |
| 15 | $6.753_{-2}$ | 1.67 | $5.075_{-2}$ | $1.071_{-1}$ | $1.086_{-1}$ |
| 19 | $4.689_{-2}$ | 1.54 | $3.586_{-2}$ | $7.667_{-2}$ | $7.485_{-2}$ |
| 24 | $3.156_{-2}$ | 1.70 | $2.409_{-2}$ | $5.212_{-2}$ | $5.006_{-2}$ |
| 32 | $2.107_{-2}$ | 1.40 | $1.602_{-2}$ | $3.477_{-2}$ | $3.368_{-2}$ |
| 40 | $1.471_{-2}$ | 1.61 | $1.139_{-2}$ | $2.516_{-2}$ | $2.310_{-2}$ |
| 50 | $1.001_{-2}$ | 1.73 | $7.784_{-3}$ | $1.735_{-2}$ | $1.569_{-2}$ |
| 63 | $6.650_{-3}$ | 1.77 | $5.141_{-3}$ | $1.144_{-2}$ | $1.052_{-2}$ |
| 79 | $4.670_{-3}$ | 1.56 | $3.679_{-3}$ | $8.375_{-3}$ | $7.164_{-3}$ |
| 99 | $3.204_{-3}$ | 1.67 | $2.545_{-3}$ | $5.825_{-3}$ | $4.873_{-3}$ |
| 124 | $2.142_{-3}$ | 1.79 | $1.696_{-3}$ | $3.904_{-3}$ | $3.302_{-3}$ |
| 155 | $1.507_{-3}$ | 1.58 | $1.213_{-3}$ | $2.849_{-3}$ | $2.265_{-3}$ |
| 194 | $1.052_{-3}$ | 1.60 | $8.580_{-4}$ | $2.041_{-3}$ | $1.540_{-3}$ |
| 243 | $6.866_{-4}$ | 1.90 | $5.538_{-4}$ | $1.308_{-3}$ | $1.043_{-3}$ |
| 304 | $4.929_{-4}$ | 1.48 | $4.059_{-4}$ | $9.792_{-4}$ | $7.218_{-4}$ |
| 380 | $3.455_{-4}$ | 1.60 | $2.876_{-4}$ | $7.030_{-4}$ | $4.909_{-4}$ |
| 475 | $2.227_{-4}$ | 1.98 | $1.830_{-4}$ | $4.429_{-4}$ | $3.344_{-4}$ |
| 594 | $1.628_{-4}$ | 1.42 | $1.366_{-4}$ | $3.380_{-4}$ | $2.327_{-4}$ |
| 743 | $1.161_{-4}$ | 1.54 | $9.852_{-5}$ | $2.468_{-4}$ | $1.594_{-4}$ |
| 929 | $7.344_{-5}$ | 2.15 | $6.104_{-5}$ | $1.510_{-4}$ | $1.086_{-4}$ |
| 1162 | $5.504_{-5}$ | 1.42 | $4.645_{-5}$ | $1.173_{-4}$ | $7.619_{-5}$ |
| 1453 | $4.071_{-5}$ | 1.62 | $3.439_{-5}$ | $8.797_{-5}$ | $5.256_{-5}$ |
| 1817 | $2.631_{-5}$ | 2.10 | $2.098_{-5}$ | $5.303_{-5}$ | $3.851_{-5}$ |
| 2272 | $2.068_{-5}$ | 1.89 | $1.596_{-5}$ | $4.099_{-5}$ | $2.537_{-5}$ |

Tabelle 7.19: Fehler $\vert\!\vert\!\vert \phi - \widetilde{\Phi}_\ell \vert\!\vert\!\vert$ und Fehlerschätzer $\mu_\ell$, $\varrho_\ell$, $\eta_\ell$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine $\varrho_\ell$-adaptive Netzfolge im Laplaceproblem 7.3 mit exakter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

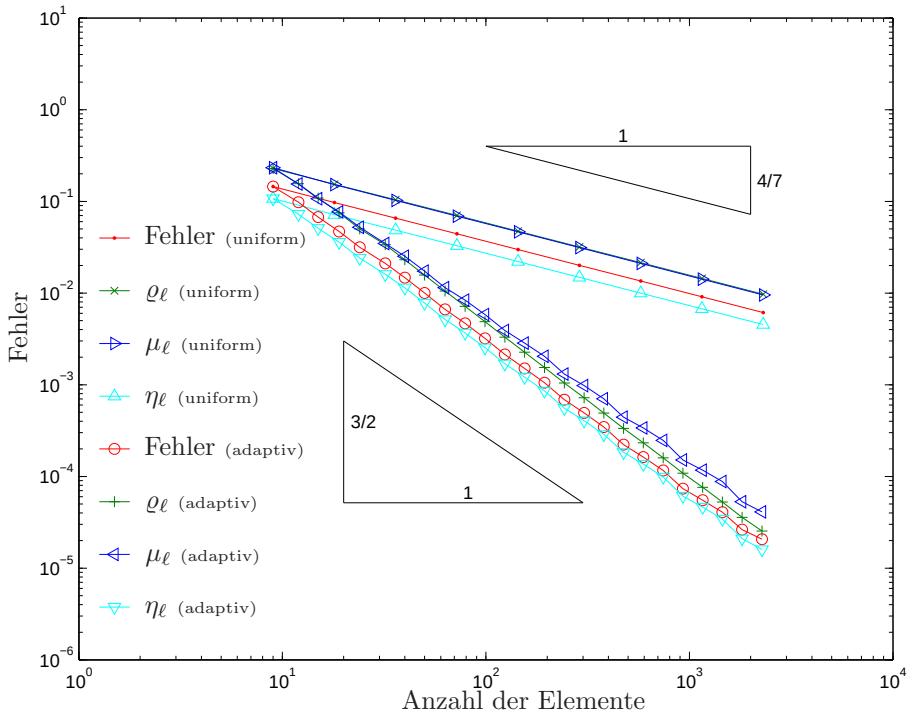


Abbildung 7.27: Fehler $\|\!|\!|\phi - \Phi_\ell|\!|\!|$ und Fehlerschätzer $\mu_\ell$, $\varrho_\ell$, $\eta_\ell$ für eine uniforme und eine $\varrho_\ell$-adaptive Netzfolge im Laplaceproblem 7.3 mit exakter rechter Seite.

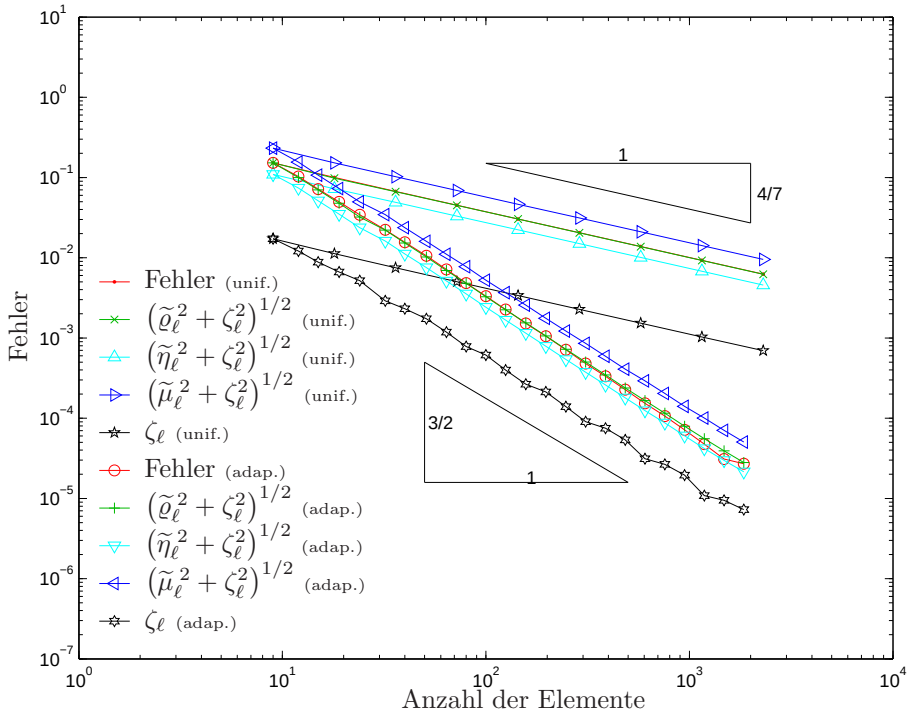


Abbildung 7.28: Fehler $\|\!|\!|\phi - \widetilde{\Phi}_\ell|\!|\!|$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ für eine uniforme und eine $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.2.1 mit gestörter rechter Seite.

| $N_\ell$ | $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ | $\alpha_\ell$ | $\zeta_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|---|
| 9 | $1.521_{-1}$ | – | $1.721_{-2}$ | $2.289_{-1}$ | $2.319_{-1}$ | $1.093_{-1}$ |
| 18 | $1.001_{-1}$ | 0.58 | $1.123_{-2}$ | $1.535_{-1}$ | $1.520_{-1}$ | $7.237_{-2}$ |
| 36 | $6.717_{-2}$ | 0.56 | $7.493_{-3}$ | $1.040_{-1}$ | $1.023_{-1}$ | $4.899_{-2}$ |
| 72 | $4.520_{-2}$ | 0.57 | $5.030_{-3}$ | $7.021_{-2}$ | $6.895_{-2}$ | $3.305_{-2}$ |
| 144 | $3.043_{-2}$ | 0.57 | $3.383_{-3}$ | $4.732_{-2}$ | $4.644_{-2}$ | $2.226_{-2}$ |
| 288 | $2.048_{-2}$ | 0.57 | $2.276_{-3}$ | $3.187_{-2}$ | $3.127_{-2}$ | $1.499_{-2}$ |
| 576 | $1.378_{-2}$ | 0.57 | $1.532_{-3}$ | $2.145_{-2}$ | $2.105_{-2}$ | $1.009_{-2}$ |
| 1152 | $9.276_{-3}$ | 0.57 | $1.031_{-3}$ | $1.444_{-2}$ | $1.416_{-2}$ | $6.791_{-3}$ |
| 2304 | $6.242_{-3}$ | 0.57 | $6.936_{-4}$ | $9.718_{-3}$ | $9.532_{-3}$ | $4.570_{-3}$ |

Tabelle 7.20: Fehler $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine uniforme Netzfolge im Laplaceproblem 7.3 mit gestörter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

| $N_\ell$ | $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ | $\alpha_\ell$ | $\zeta_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|---|
| 9 | $1.521_{-1}$ | – | $1.721_{-2}$ | $2.289_{-1}$ | $2.319_{-1}$ | $1.093_{-1}$ |
| 12 | $1.029_{-1}$ | 1.36 | $1.212_{-2}$ | $1.562_{-1}$ | $1.557_{-1}$ | $7.378_{-2}$ |
| 15 | $7.204_{-2}$ | 1.67 | $8.926_{-3}$ | $1.085_{-1}$ | $1.083_{-1}$ | $5.182_{-2}$ |
| 19 | $5.000_{-2}$ | 1.61 | $6.408_{-3}$ | $7.425_{-2}$ | $7.611_{-2}$ | $3.585_{-2}$ |
| 24 | $3.332_{-2}$ | 1.63 | $3.987_{-3}$ | $5.000_{-2}$ | $5.283_{-2}$ | $2.460_{-2}$ |
| 32 | $2.232_{-2}$ | 1.40 | $2.865_{-3}$ | $3.366_{-2}$ | $3.491_{-2}$ | $1.627_{-2}$ |
| 40 | $1.565_{-2}$ | 1.61 | $1.953_{-3}$ | $2.306_{-2}$ | $2.541_{-2}$ | $1.161_{-2}$ |
| 50 | $1.067_{-2}$ | 1.73 | $1.336_{-3}$ | $1.566_{-2}$ | $1.749_{-2}$ | $7.923_{-3}$ |
| 63 | $6.963_{-3}$ | 1.77 | $9.420_{-4}$ | $1.053_{-2}$ | $1.144_{-2}$ | $5.213_{-3}$ |
| 79 | $4.926_{-3}$ | 1.56 | $6.182_{-4}$ | $7.160_{-3}$ | $8.419_{-3}$ | $3.736_{-3}$ |
| 99 | $3.342_{-3}$ | 1.64 | $4.120_{-4}$ | $4.895_{-3}$ | $5.896_{-3}$ | $2.602_{-3}$ |
| 124 | $2.241_{-3}$ | 1.82 | $3.012_{-4}$ | $3.305_{-3}$ | $3.907_{-3}$ | $1.719_{-3}$ |
| 155 | $1.586_{-3}$ | 1.58 | $2.064_{-4}$ | $2.265_{-3}$ | $2.859_{-3}$ | $1.231_{-3}$ |
| 194 | $1.085_{-3}$ | 1.60 | $1.351_{-4}$ | $1.542_{-3}$ | $2.047_{-3}$ | $8.690_{-4}$ |
| 243 | $7.152_{-4}$ | 1.89 | $1.001_{-4}$ | $1.046_{-3}$ | $1.313_{-3}$ | $5.632_{-4}$ |
| 304 | $5.151_{-4}$ | 1.48 | $6.812_{-5}$ | $7.225_{-4}$ | $9.847_{-4}$ | $4.124_{-4}$ |
| 380 | $3.542_{-4}$ | 1.61 | $4.500_{-5}$ | $4.921_{-4}$ | $7.041_{-4}$ | $2.910_{-4}$ |
| 475 | $2.305_{-4}$ | 1.96 | $3.435_{-5}$ | $3.355_{-4}$ | $4.464_{-4}$ | $1.869_{-4}$ |
| 594 | $1.686_{-4}$ | 1.42 | $2.328_{-5}$ | $2.333_{-4}$ | $3.405_{-4}$ | $1.392_{-4}$ |
| 743 | $1.180_{-4}$ | 1.56 | $1.537_{-5}$ | $1.598_{-4}$ | $2.473_{-4}$ | $9.970_{-5}$ |
| 929 | $7.321_{-5}$ | 2.13 | $1.208_{-5}$ | $1.092_{-4}$ | $1.522_{-4}$ | $6.246_{-5}$ |
| 1162 | $5.350_{-5}$ | 1.41 | $8.155_{-6}$ | $7.651_{-5}$ | $1.183_{-4}$ | $4.740_{-5}$ |
| 1453 | $3.682_{-5}$ | 1.65 | $5.299_{-6}$ | $5.275_{-5}$ | $8.812_{-5}$ | $3.479_{-5}$ |
| 1817 | $2.909_{-5}$ | 2.11 | $4.222_{-6}$ | $3.825_{-5}$ | $4.454_{-5}$ | $2.212_{-5}$ |

Tabelle 7.21: Fehler $\lvert\!\lvert\!\lvert \phi - \widetilde{\Phi}_\ell \rvert\!\rvert\!\rvert$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\mu}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ sowie experimentelle Konvergenzrate $\alpha_\ell$ des Fehlers für eine $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.3 mit gestörter rechter Seite. Es bezeichnet $N_\ell$ die Anzahl der Elemente in $\mathcal{T}_\ell$.

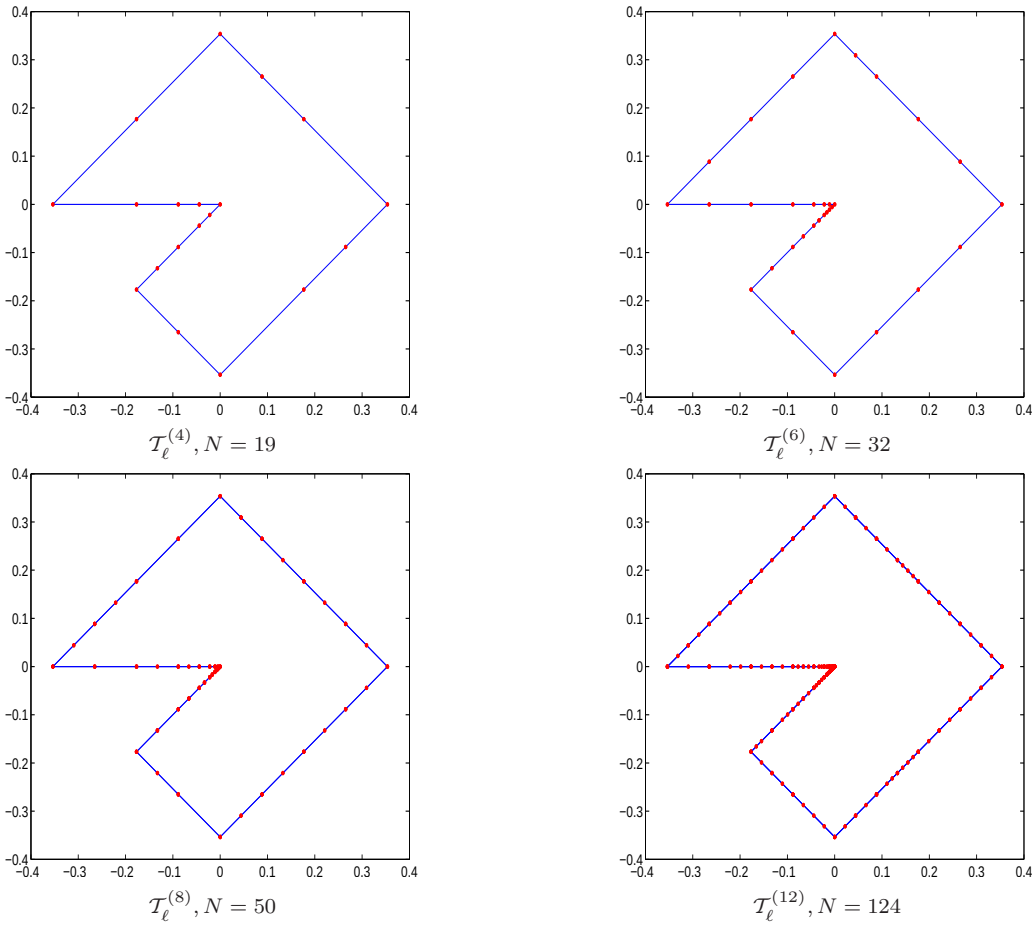


Abbildung 7.29: Ausschnitt der Folge von Partitionen, die durch $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Verfeinerungen im Laplaceproblem 7.3 hervorgeht.

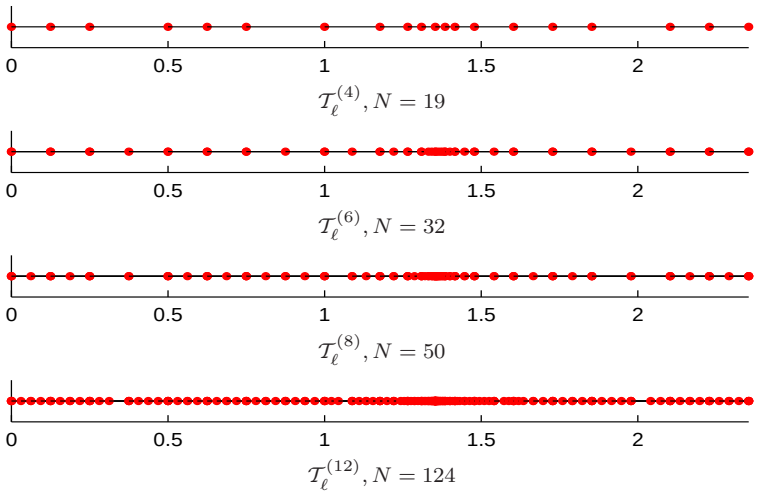


Abbildung 7.30: Ausschnitt der Folge von Partitionen über der Bogenlänge, die durch $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Verfeinerungen im Laplaceproblem 7.3 hervorgeht.

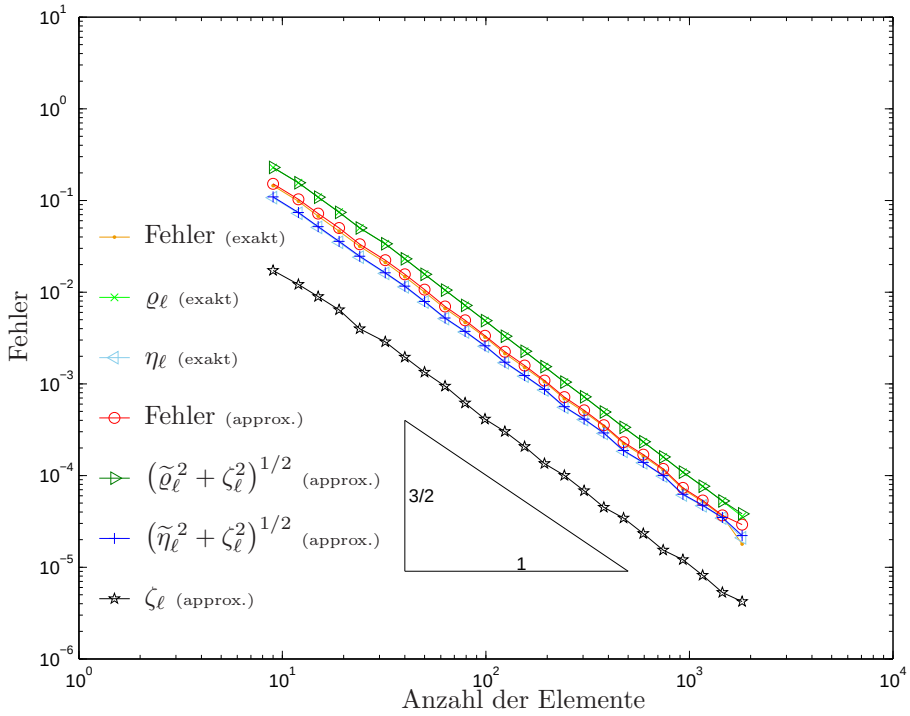


Abbildung 7.31: Fehler $\left|\!\left|\!\left|\phi-\Phi_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\eta_\ell$, $\varrho_\ell$ für eine Berechnung mit exakter rechter Seite und Fehler $\left|\!\left|\!\left|\phi-\widetilde{\Phi}_\ell\right|\!\right|\!\right|$ und Fehlerschätzer $\zeta_\ell$, $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$, $\left(\widetilde{\eta}_\ell^2+\zeta_\ell^2\right)^{1/2}$ für eine Berechnung mit approximativer rechter Seite des Laplaceproblems 7.3.

Anhand von Abbildung 7.28, die den exakten Fehler $e_\ell=\left|\!\left|\!\left|\phi-\widetilde{\Phi}_\ell\right|\!\right|\!\right|$, und die Fehlerschätzer

$$\begin{aligned}
&\zeta_\ell=\left\|h_\ell^{1/2}(g-g_\ell)'\right\|_{L^2(\Gamma)},\\
&\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}\quad\text{mit }\widetilde{\varrho}_\ell=\left\|h_\ell^{1/2}\big((K+\tfrac12)g_\ell-V\widetilde{\Phi}_\ell\big)'\right\|,\\
&\left(\widetilde{\mu}_\ell^2+\zeta_\ell^2\right)^{1/2}\quad\text{mit }\widetilde{\mu}_\ell=\left\|h_\ell^{1/2}\big(\widehat{\widetilde{\Phi}}_\ell-\widetilde{\Phi}_\ell\big)\right\|_{L^2(\Gamma)},\\
&\left(\widetilde{\eta}_\ell^2+\zeta_\ell^2\right)^{1/2}\quad\text{mit }\widetilde{\eta}_\ell=\left|\!\left|\!\left|\widehat{\widetilde{\Phi}}_\ell-\widetilde{\Phi}_\ell\right|\!\right|\!\right|
\end{aligned}$$

zeigt, kann man erkennen, dass sich eine Konvergenzrate von $\alpha=\frac47\approx0.57$ ergibt, was mit dem Reziprokwert des Innenwinkels des Gebiets bei der Singularität übereinstimmt und daher zu erwarten war. Auch hier erkennt man eine Verbesserung der Konvergenzrate bei adaptiver Rechnung zu $\alpha=\frac32$. Die Zuverlässigkeit und Effizienz des Residualschätzers und des Gesamtfehlerschätzers wird durch die parallel zum Fehler verlaufenden Kurven numerisch bestätigt.

Die Daten für die uniforme Verfeinerung finden sich in Tabelle 7.20, die für adaptive Verfeinerung mit $\omega_\ell=\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$ in Tabelle 7.21.

Die Abbildungen 7.29 und 7.30 zeigen eine Folge adaptiver Partitionen. Man kann deutlich die Verfeinerung zur Ecke, in der sich die Singularität befindet, erkennen.

Zum Vergleich der Berechnungen mit exakter rechter Seite und gestörter rechter Seite sind in Abbildung 7.31 sowohl der exakte Fehler $\left|\!\left|\!\left|\phi-\Phi_\ell\right|\!\right|\!\right|$ und die Fehlerschätzer

$$\varrho_\ell=\left\|h_\ell^{1/2}\big((K+\tfrac12)g-V\Phi_\ell\big)'\right\|_{L^2(\Gamma)},\qquad \eta_\ell=\left|\!\left|\!\left|\widehat{\Phi}_\ell-\Phi_\ell\right|\!\right|\!\right|$$

| $N_\ell$ | $\|\|\|\phi - \Phi_\ell\|\|\|$ | $\|\|\|\phi - \widetilde{\Phi}_\ell\|\|\|$ | $\zeta_\ell$ | $\varrho_\ell$ | $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ | $\eta_\ell$ | $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ |
|---|---|---|---|---|---|---|---|
| 9 | $1.449_{-1}$ | $1.521_{-1}$ | $1.721_{-2}$ | $2.285_{-1}$ | $2.289_{-1}$ | $1.075_{-1}$ | $1.093_{-1}$ |
| 12 | $9.797_{-2}$ | $1.029_{-1}$ | $1.212_{-2}$ | $1.562_{-1}$ | $1.562_{-1}$ | $7.255_{-2}$ | $7.378_{-2}$ |
| 15 | $6.753_{-2}$ | $7.204_{-2}$ | $8.926_{-3}$ | $1.086_{-1}$ | $1.085_{-1}$ | $5.075_{-2}$ | $5.182_{-2}$ |
| 19 | $4.619_{-2}$ | $5.000_{-2}$ | $6.408_{-3}$ | $7.462_{-2}$ | $7.425_{-2}$ | $3.500_{-2}$ | $3.585_{-2}$ |
| 24 | $3.156_{-2}$ | $3.332_{-2}$ | $3.987_{-3}$ | $5.006_{-2}$ | $5.000_{-2}$ | $2.409_{-2}$ | $2.460_{-2}$ |
| 32 | $2.107_{-2}$ | $2.232_{-2}$ | $2.865_{-3}$ | $3.368_{-2}$ | $3.366_{-2}$ | $1.602_{-2}$ | $1.627_{-2}$ |
| 40 | $1.471_{-2}$ | $1.565_{-2}$ | $1.953_{-3}$ | $2.310_{-2}$ | $2.306_{-2}$ | $1.139_{-2}$ | $1.161_{-2}$ |
| 50 | $1.001_{-2}$ | $1.067_{-2}$ | $1.336_{-3}$ | $1.596_{-2}$ | $1.566_{-2}$ | $7.784_{-3}$ | $7.923_{-3}$ |
| 63 | $6.650_{-3}$ | $6.963_{-3}$ | $9.420_{-4}$ | $1.052_{-2}$ | $1.053_{-2}$ | $5.141_{-3}$ | $5.213_{-3}$ |
| 79 | $4.670_{-3}$ | $4.926_{-3}$ | $6.182_{-4}$ | $7.164_{-3}$ | $7.160_{-3}$ | $3.679_{-3}$ | $3.736_{-3}$ |
| 99 | $3.224_{-3}$ | $3.342_{-3}$ | $4.120_{-4}$ | $4.891_{-3}$ | $4.895_{-3}$ | $2.567_{-3}$ | $2.602_{-3}$ |
| 124 | $2.142_{-3}$ | $2.241_{-3}$ | $3.012_{-4}$ | $3.302_{-3}$ | $3.305_{-3}$ | $1.696_{-3}$ | $1.719_{-3}$ |
| 155 | $1.507_{-3}$ | $1.586_{-3}$ | $2.064_{-4}$ | $2.265_{-3}$ | $2.265_{-3}$ | $1.213_{-3}$ | $1.231_{-3}$ |
| 194 | $1.052_{-3}$ | $1.085_{-3}$ | $1.351_{-4}$ | $1.540_{-3}$ | $1.542_{-3}$ | $8.580_{-4}$ | $8.690_{-4}$ |
| 243 | $6.874_{-4}$ | $7.152_{-4}$ | $1.001_{-4}$ | $1.043_{-3}$ | $1.046_{-3}$ | $5.550_{-4}$ | $5.632_{-4}$ |
| 304 | $4.935_{-4}$ | $5.151_{-4}$ | $6.812_{-5}$ | $7.214_{-4}$ | $7.225_{-4}$ | $4.069_{-4}$ | $4.124_{-4}$ |
| 380 | $3.448_{-4}$ | $3.542_{-4}$ | $4.500_{-5}$ | $4.911_{-4}$ | $4.921_{-4}$ | $2.875_{-4}$ | $2.910_{-4}$ |
| 475 | $2.228_{-4}$ | $2.305_{-4}$ | $3.435_{-5}$ | $3.344_{-4}$ | $3.355_{-4}$ | $1.840_{-4}$ | $1.869_{-4}$ |
| 594 | $1.623_{-4}$ | $1.686_{-4}$ | $2.328_{-5}$ | $2.327_{-4}$ | $2.333_{-4}$ | $1.373_{-4}$ | $1.392_{-4}$ |
| 743 | $1.145_{-4}$ | $1.180_{-4}$ | $1.537_{-5}$ | $1.594_{-4}$ | $1.598_{-4}$ | $9.852_{-5}$ | $9.970_{-5}$ |
| 929 | $7.116_{-5}$ | $7.321_{-5}$ | $1.208_{-5}$ | $1.087_{-4}$ | $1.092_{-4}$ | $6.134_{-5}$ | $6.246_{-5}$ |
| 1162 | $5.185_{-5}$ | $5.350_{-5}$ | $8.155_{-6}$ | $7.620_{-5}$ | $7.651_{-5}$ | $4.672_{-5}$ | $4.740_{-5}$ |
| 1453 | $3.587_{-5}$ | $3.682_{-5}$ | $5.299_{-6}$ | $5.257_{-5}$ | $5.275_{-5}$ | $3.439_{-5}$ | $3.479_{-5}$ |
| 1817 | $1.788_{-5}$ | $2.909_{-5}$ | $4.222_{-6}$ | $3.597_{-5}$ | $3.825_{-5}$ | $2.095_{-5}$ | $2.212_{-5}$ |

Tabelle 7.22: Fehler $\|\|\|\phi - \Phi_\ell\|\|\|$, $\|\|\|\phi - \widetilde{\Phi}_\ell\|\|\|$ sowie Fehlerschätzer $\zeta_\ell$, $\widetilde{\varrho}_\ell$, $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$, $\widetilde{\eta}_\ell$, $\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ für eine $\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.3 mit gestörter rechter Seite.

für die exakte rechte Seite als auch der exakte Fehler $\|\|\|\phi - \widetilde{\Phi}_\ell\|\|\|$ und die Fehlerschätzer

$$\begin{aligned}
&\zeta_\ell = \left\|h_\ell^{1/2}(g - g_\ell)'\right\|_{L^2(\Gamma)}, \\
&\left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\varrho}_\ell = \left\|h_\ell^{1/2}\left((K + \tfrac{1}{2})g_\ell - V\widetilde{\Phi}_\ell\right)'\right\|, \\
&\left(\widetilde{\eta}_\ell^2 + \zeta_\ell^2\right)^{1/2} \quad \text{mit } \widetilde{\eta}_\ell = \|\|\widehat{\widetilde{\Phi}}_\ell - \widetilde{\Phi}_\ell\|\|
\end{aligned}$$

für die approximierte rechte Seite angegeben. Der adaptive Algorithmus wird mittels $\omega_\ell = \left(\widetilde{\varrho}_\ell^2 + \zeta_\ell^2\right)^{1/2}$ und $\theta = \frac{1}{4}$ gesteuert. Man sieht, dass sich die Fehler und auch Fehlerschätzer kaum unterscheiden. Deshalb sind die genauen Größen in Tabelle 7.22 angegeben.

Auch für dieses Beispiel werden die Konditionszahlen der unveränderten Matrix $A$, der diagonal vorkonditionierten Matrix $A_{\text{eq}}$ und der mittels Zeilenäquilibrierung vorkonditionierten Matrix $A_{\text{diag}}$ in Tabelle 7.23 und Abbildung 7.32 dargestellt.

Die Beschränktheit von $\kappa(\mathcal{T}_\ell)$ wird numerisch überprüft und in Tabelle 7.24 ausgegeben. Zum Vergleich ist wiederum der maximale Quotient der lokalen Netzweiten von beliebigen, also nicht benachbarten Elementen, d.h $\frac{h_{\max}}{h_{\min}}$, angegeben.

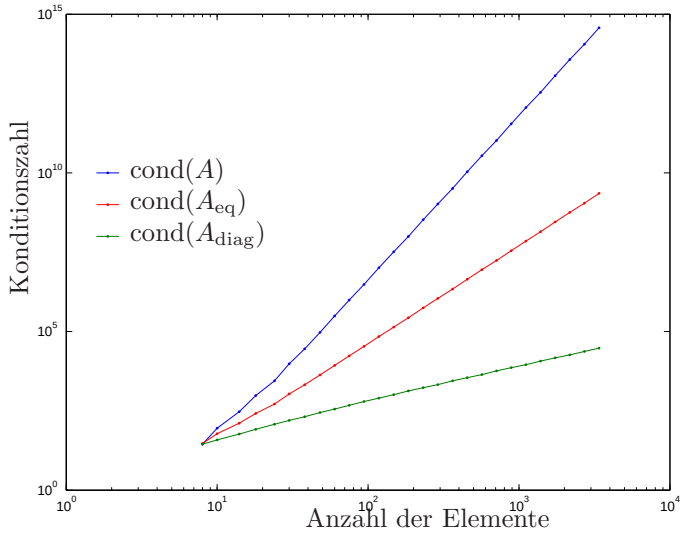


Abbildung 7.32: Konditionszahlen der Steifigkeitsmatrix ohne, mit zeilenäquilibrierter und mit diagonaler Vorkonditionierung für eine $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.3 über der Anzahl der Elemente.

| $N_\ell$ | $\operatorname{cond}(A)$ | $\operatorname{cond}(A_{\mathrm{eq}})$ | $\operatorname{cond}(A_{\mathrm{diag}})$ |
|---|---|---|---|
| 9 | $3.4_1$ | $3.8_1$ | $3.7_1$ |
| 12 | $1.0_2$ | $7.4_1$ | $5.2_1$ |
| 15 | $3.5_2$ | $1.6_2$ | $7.6_1$ |
| 19 | $1.2_3$ | $3.2_2$ | $1.1_2$ |
| 24 | $3.7_3$ | $6.3_2$ | $1.4_2$ |
| 32 | $1.2_4$ | $1.3_3$ | $2.1_2$ |
| 40 | $4.1_4$ | $2.7_3$ | $2.8_2$ |
| 50 | $1.3_5$ | $5.3_3$ | $3.7_2$ |
| 63 | $3.9_5$ | $1.1_4$ | $4.8_2$ |
| 79 | $1.3_6$ | $2.1_4$ | $6.2_2$ |
| 99 | $3.9_6$ | $4.2_4$ | $7.9_2$ |
| 124 | $1.3_7$ | $8.5_4$ | $1.1_3$ |
| 155 | $3.9_7$ | $1.7_5$ | $1.4_3$ |
| 194 | $1.3_8$ | $3.4_5$ | $1.7_3$ |
| 243 | $4.3_8$ | $6.7_5$ | $2.2_3$ |
| 304 | $1.3_9$ | $1.4_6$ | $2.8_3$ |
| 380 | $4.2_9$ | $2.7_6$ | $3.5_3$ |
| 475 | $1.4_{10}$ | $5.4_6$ | $4.7_3$ |
| 594 | $4.5_{10}$ | $1.1_7$ | $5.8_3$ |
| 743 | $1.4_{11}$ | $2.2_7$ | $7.3_3$ |
| 929 | $4.4_{11}$ | $4.3_7$ | $9.7_3$ |
| 1162 | $1.5_{12}$ | $8.7_7$ | $1.2_4$ |
| 1453 | $4.6_{12}$ | $1.7_8$ | $1.5_4$ |
| 1817 | $1.4_{13}$ | $3.4_8$ | $2.0_4$ |

Tabelle 7.23: Konditionszahlen der Steifigkeitsmatrix ohne, mit zeilenäquilibrierter und mit diagonaler Vorkonditionierung für eine $\left(\widetilde{\varrho}_\ell^2+\zeta_\ell^2\right)^{1/2}$-adaptive Netzfolge im Laplaceproblem 7.3.

| $N_\ell$ | $\kappa(\mathcal{T}_\ell)$ | $\frac{h_{\max}}{h_{\min}}$ |
|---|---|---|
| 9 | 1 | $1.4_0$ |
| 12 | 2 | $2.0_0$ |
| 15 | 2 | $4.0_0$ |
| 19 | 2 | $8.0_0$ |
| 24 | 2 | $1.6_1$ |
| 32 | 2 | $1.6_1$ |
| 40 | 2 | $3.2_1$ |
| 50 | 2 | $6.4_1$ |
| 63 | 2 | $9.1_1$ |
| 79 | 2 | $1.3_2$ |
| 99 | 2 | $2.6_2$ |
| 124 | 2 | $3.6_2$ |
| 155 | 2 | $5.1_2$ |
| 194 | 2 | $1.0_3$ |
| 243 | 2 | $1.4_3$ |
| 304 | 2 | $2.9_3$ |
| 380 | 2 | $4.1_3$ |
| 475 | 2 | $5.8_3$ |
| 594 | 2 | $1.2_4$ |
| 743 | 2 | $1.6_4$ |
| 929 | 2 | $2.3_4$ |
| 1162 | 2 | $4.6_4$ |
| 1453 | 2 | $6.6_4$ |
| 1817 | 2 | $9.3_4$ |

Tabelle 7.24: Anzahl der Elemente $N_\ell$ in $\mathcal{T}_\ell$, $\kappa(\mathcal{T}_\ell)$ und $\frac{h_{\max}}{h_{\min}}$ bei Rechnung wie in Tabelle 7.23.

# Anhang A

# Quelltexte

## A.1 elementare Integrale

### A.1.1 integrals.h

```
1  #include "math.h"
2  #include "mex.h"
3  #define PI 3.141592653589793116
4  #define EPS 1e-15
5
6  /* computes \int_{-1}^1 t^k / ( |p|^2*t^2 + 2*<p,q>*t + |q|^2 ) dt     */
7  /* input: integer k                                                    */
8  /*        vectors p,q                                                 */
9  /*        outputvector val containing values of all integrals until k */
10 void G (double* val, int k, double* p, double* q);
11
12
13 /* computes \int_{-1}^1 t^k * log ( |p|^2*t^2 + 2*<p,q>*t + |q|^2 ) dt */
14 /* input: integer k                                                    */
15 /*        vectors p,q                                                 */
16 /*        outputvector val containing values of all integrals until k */
17 void L (double* val, int k, double* p, double* q);
18
19
20 /* computes double integrals                                                    */
21 /* \int_{-1}^1 \int_{-1}^1 s^k * t^ell * <su+tv+w,n> / |su+tv+w|^2 dt ds        */
22 /* input: integers k,ell                                                       */
23 /*        vectors u,v,w                                                       */
24 /*        integer type: double integral: \int_[aa bb] \int [a,b] ...          */
25 /*             type 0: intervals don't touch                                  */
26 /*             type 1: bb == a                                                */
27 /*             type 2: aa == b                                                */
28 /*             outputmatrix val containing values of all integrals until (k,ell) */
29 void I (double** val, int k, int ell, double* u, double* v, double* w,
30         double* n, int type);
```

### A.1.2 integrals.c

```
1  #include "integrals.h"
2
```

```
3  void G (double* val, int k, double* p, double* q)
4  {
5    double norm_p, norm_q, norm_p2, norm_q2;
6    double sc_p_q, delta;
7    double tmp;
8    double* ppq, *pmq;
9    int j;
10
11   norm_p2 = p[0]*p[0]+p[1]*p[1];
12   norm_p = sqrt(norm_p2);
13   norm_q2 = q[0]*q[0]+q[1]*q[1];
14   norm_q = sqrt(norm_q2);
15   sc_p_q = p[0]*q[0]+p[1]*q[1];
16   delta = 4*(norm_p2*norm_q2-sc_p_q*sc_p_q);
17
18   /* case k==0 */
19   if (fabs(delta)>EPS)
20   {
21    if (norm_p<norm_q)
22      val[0] = atan( sqrt(delta) / (norm_q2-norm_p2) );
23    else if (norm_p > norm_q)
24      val[0] = atan ( sqrt(delta) / (norm_q2-norm_p2) ) + PI;
25    else
26      val[0] = PI/2;
27    val[0] = val[0]*2 / sqrt(delta);
28   }
29   else
30     val[0] = 2 / (norm_q2-norm_p2);
31
32   /* case k==1 */
33   if (k>=1)
34   {
35     ppq = mxCalloc(2,sizeof(double));
36     pmq = mxCalloc(2,sizeof(double));
37     ppq[0] = p[0]+q[0];
38     ppq[1] = p[1]+q[1];
39     pmq[0] = p[0]-q[0];
40     pmq[1] = p[1]-q[1];
41     val[1] = ( log( sqrt(ppq[0]*ppq[0]+ppq[1]*ppq[1]) )
42            - log( sqrt(pmq[0]*pmq[0]+pmq[1]*pmq[1]) )
43            - sc_p_q*val[0] )/norm_p2;
44     mxFree(ppq);
45     mxFree(pmq);
46   }
47   /* case k>1 */
48   for(j=2; j<=k; j++)
49   {
50     tmp = 2 * ( (j+1)%2 ) / (j-1);
51     val[j] =  ( tmp - 2*sc_p_q*val[j-1] - norm_q2*val[j-2] )/norm_p2;
52   }
53 }
54
55
56 /*****************************************************************/
57 void L (double* val, int k, double* p, double* q)
58 {
59   double norm_p, norm_q, norm_p2, norm_q2;
60   double norm_ppq2, norm_pmq2;
61   double sc_p_q, delta;
62   double tmp, tmp1, tmp2;
63   double* ppq, *pmq;
```

```
64    double G_p_q_0;
65    int j;
66
67    norm_p2 = p[0]*p[0]+p[1]*p[1];
68    norm_p = sqrt(norm_p2);
69    norm_q2 = q[0]*q[0]+q[1]*q[1];
70    norm_q = sqrt(norm_q2);
71    sc_p_q = p[0]*q[0]+p[1]*q[1];
72    delta = 4*(norm_p2*norm_q2-sc_p_q*sc_p_q);
73    tmp = sc_p_q/norm_p2;
74
75    ppq = mxCalloc(2,sizeof(double));
76    pmq = mxCalloc(2,sizeof(double));
77    ppq[0] = p[0]+q[0];
78    ppq[1] = p[1]+q[1];
79    pmq[0] = p[0]-q[0];
80    pmq[1] = p[1]-q[1];
81
82    norm_ppq2 = ppq[0]*ppq[0]+ppq[1]*ppq[1];
83    norm_pmq2 = pmq[0]*pmq[0]+pmq[1]*pmq[1];
84
85    /* case k==0 */
86    if (delta>0)
87    {
88      G(&G_p_q_0,0,p,q);
89      val[0] = (1+tmp) * log( norm_ppq2 ) + (1-tmp) * log( norm_pmq2 ) -4
90             + 2*(norm_q2-tmp*sc_p_q)*G_p_q_0;
91    }
92    else
93      val[0] = (1+tmp) * log( (sc_p_q+norm_p2)*(sc_p_q+norm_p2) / norm_p2 )
94             + (1-tmp) * log( (sc_p_q-norm_p2)*(sc_p_q-norm_p2) / norm_p2 )
95             -4;
96
97    /* case k==1 */
98    if (k>=1)
99      val[1] = ( norm_ppq2*log(norm_ppq2) - norm_pmq2*log(norm_pmq2)
100            - 4*sc_p_q - 2*sc_p_q*val[0] ) / ( 2*norm_p2 );
101
102   /*case k>1 */
103   for (j=2; j<=k; ++j)
104   {
105     tmp = 2 * ( (j+1)%2 ) / (j+1);
106     tmp1 = 1-2*(j%2);
107     tmp2 = 2 * (j%2) / j;
108       val[j] = ( norm_ppq2*log(norm_ppq2) + tmp1*norm_pmq2*log(norm_pmq2)
109              - 2*norm_p2*tmp - 2*sc_p_q*tmp2 - 2*sc_p_q*j*val[j-1]
110              - norm_q2*(j-1)*val[j-2] ) / ( (j+1)*norm_p2 );
111   }
112   mxFree(ppq);
113   mxFree(pmq);
114 }
115
116
117 /*****************************************************************/
118 void I (double** val, int k, int ell, double* u, double* v, double* w,
119         double* n, int type)
120 {
121   int i,j;
122   double *wpv, *wmv, *wpu, *wmu;
123   double sc_u_n, sc_w_n, sc_wpu_n, sc_wmu_n;
124   double *G_u_wpv, *G_u_wmv, *G_v_wpu, *G_v_wmu;
```

```
125   double mu, mu1, mu2;
126   int kpell = k+ell;
127
128   wpv = mxCalloc(2,sizeof(double));
129   wmv = mxCalloc(2,sizeof(double));
130   wpu = mxCalloc(2,sizeof(double));
131   wmu = mxCalloc(2,sizeof(double));
132   wpv[0] = w[0]+v[0];
133   wpv[1] = w[1]+v[1];
134   wmv[0] = w[0]-v[0];
135   wmv[1] = w[1]-v[1];
136   wpu[0] = w[0]+u[0];
137   wpu[1] = w[1]+u[1];
138   wmu[0] = w[0]-u[0];
139   wmu[1] = w[1]-u[1];
140
141   sc_u_n = u[0]*n[0]+u[1]*n[1];
142   sc_w_n = w[0]*n[0]+w[1]*n[1];
143   sc_wpu_n = wpu[0]*n[0]+wpu[1]*n[1];
144   sc_wmu_n = wmu[0]*n[0]+wmu[1]*n[1];
145
146   /* case: integrals don't touch */
147   if (type == 0)
148   {
149     /* case u || v   <=>   u = mu*v */
150     if (fabs(u[0]*v[1]-u[1]*v[0]) < EPS)
151     {
152       G_u_wmv = mxCalloc(k+2,sizeof(double));
153       G_u_wpv = mxCalloc(k+2,sizeof(double));
154       G_v_wmu = mxCalloc(kpell+2,sizeof(double));
155       G_v_wpu = mxCalloc(kpell+2,sizeof(double));
156       G(G_u_wpv,k+1,u,wpv);
157       G(G_u_wmv,k+1,u,wmv);
158       G(G_v_wpu,kpell+1,v,wpu);
159       G(G_v_wmu,kpell+1,v,wmu);
160
161       if (fabs(v[0]) < EPS)
162         mu = u[1]/v[1];
163       else
164         mu = u[0]/v[0];
165
166       /* I(0,j) */
167       for (j=0; j<=kpell; j++)
168       {
169         val[0][j] =  ( sc_w_n*G_u_wpv[0]
170                   + ( 1-2*(j%2) )*sc_w_n*G_u_wmv[0]
171                   - mu*sc_w_n*G_v_wpu[j+1]
172                   + mu*sc_w_n*G_v_wmu[j+1] ) / (j+1);
173         /* I(j,ell+k-j) */
174         for (i=1; i<=k; i++)
175         {
176           val[i][j] = ( sc_w_n*G_u_wpv[j]
177                     + ( 1-2*((kpell-j)%2) )*sc_w_n*G_u_wmv[j]
178                     - mu*sc_w_n*G_v_wpu[kpell-j+1]
179                     + ( 1-2*(j%2) )*mu*sc_w_n*G_v_wmu[kpell-j+1]
180                     + j*mu*val[i-1][j+1] ) / (kpell-j+1);
181         }
182       }
183     }
184     /* case u not || v   <=>   w = mu1*u + mu2*v */
185     else
```

```
186     {
187       G_u_wmv = mxCalloc(k+2,sizeof(double));
188       G_u_wpv = mxCalloc(k+2,sizeof(double));
189       G_v_wmu = mxCalloc(ell+2,sizeof(double));
190       G_v_wpu = mxCalloc(ell+2,sizeof(double));
191       G(G_u_wpv,k+1,u,wpv);
192       G(G_u_wmv,k+1,u,wmv);
193       G(G_v_wmu,ell+1,v,wmu);
194       G(G_v_wpu,ell+1,v,wpu);
195
196       mu1 = ( w[0]*v[1]-w[1]*v[0] ) / ( u[0]*v[1]-u[1]*v[0] );
197       mu2 = ( w[0]*u[1]-w[1]*u[0] ) / ( v[0]*u[1]-v[1]*u[0] );
198
199       /* I(0,0) */
200       val[0][0] = (mu1+1)*sc_wpu_n*G_v_wpu[0]
201                 - (mu1-1)*sc_wmu_n*G_v_wmu[0]
202                 + (mu2+1)*sc_u_n*G_u_wpv[1] + (mu2+1)*sc_w_n*G_u_wpv[0]
203                 - (mu2-1)*sc_u_n*G_u_wmv[1] - (mu2-1)*sc_w_n*G_u_wmv[0];
204
205       /* I(0,j) */
206       for (j=1; j<=ell; j++)
207       {
208         val[0][j] = ( (mu1+1)*sc_wpu_n*G_v_wpu[j]
209                     - (mu1-1)*sc_wmu_n*G_v_wmu[j]
210                     + (mu2+1)*sc_u_n*G_u_wpv[1] + (mu2+1)*sc_w_n*G_u_wpv[0]
211                     - ( 1-2*(j%2) )*(mu2-1)*sc_u_n*G_u_wmv[1]
212                     - ( 1-2*(j%2) )*(mu2-1)*sc_w_n*G_u_wmv[0]
213                     - j*mu2*val[0][j-1] ) / (j+1);
214       }
215
216       /* I(i,j) */
217       for (i=1; i<=k; i++)
218       {
219         val[i][0] = ( (mu1+1)*sc_wpu_n*G_v_wpu[0]
220                     - ( 1-2*(i%2) )*(mu1-1)*sc_wmu_n*G_v_wmu[0]
221                     + (mu2+1)*sc_u_n*G_u_wpv[i+1]
222                     + (mu2+1)*sc_w_n*G_u_wpv[i]
223                     - (mu2-1)*sc_u_n*G_u_wmv[i+1]
224                     - (mu2-1)*sc_w_n*G_u_wmv[i]
225                     - i*mu1*val[i-1][0] ) / (i+1);
226         for (j=1; j<=ell; j++)
227         {
228           val[i][j] = ( (mu1+1)*sc_wpu_n*G_v_wpu[j]
229                       - ( 1-2*(i%2) )*(mu1-1)*sc_wmu_n*G_v_wmu[j]
230                       + (mu2+1)*sc_u_n*G_u_wpv[i+1]
231                       + (mu2+1)*sc_w_n*G_u_wpv[i]
232                       - ( 1-2*(j%2) )*(mu2-1)*sc_u_n*G_u_wmv[i+1]
233                       - ( 1-2*(j%2) )*(mu2-1)*sc_w_n*G_u_wmv[i]
234                       - i*mu1*val[i-1][j] - j*mu2*val[i][j-1] ) / (i+j+1);
235         }
236       }
237     }
238     mxFree(G_v_wmu);
239     mxFree(G_u_wpv);
240     mxFree(G_v_wpu);
241     mxFree(G_u_wmv);
242   }
243
244   /* case: bb == a */
245   else if (type == 1)
246   {
```

```
247       G_u_wpv = mxCalloc(k+2,sizeof(double));
248       G_v_wmu = mxCalloc(ell+1,sizeof(double));
249       G(G_u_wpv,k+1,u,wpv);
250       G(G_v_wmu,ell,v,wmu);
251
252       /* I(0,0) */
253       val[0][0] = 2* (sc_wmu_n*G_v_wmu[0] + sc_u_n*G_u_wpv[1]
254                 + sc_w_n*G_u_wpv[0] );
255       /* I(0,j) */
256       for (j=1; j<=ell; j++)
257       {
258         val[0][j] = ( 2*sc_wmu_n*G_v_wmu[j] + 2*sc_u_n*G_u_wpv[1]
259                   + 2*sc_w_n*G_u_wpv[0] - j*val[0][j-1] ) / (j+1);
260       }
261
262       /* I(i,j) */
263       for (i=1; i<=k; i++)
264       {
265         val[i][0] = ( 2*( 1-2*(i%2) )*sc_wmu_n*G_v_wmu[0]
266                   + 2*sc_u_n*G_u_wpv[i+1] + 2*sc_w_n*G_u_wpv[i]
267                   + i*val[i-1][0] ) / (i+1);
268         for (j=1; j<=ell; j++)
269         {
270           val[i][j] = ( 2*( 1-2*(i%2) )*sc_wmu_n*G_v_wmu[j]
271                     + 2*sc_u_n*G_u_wpv[i+1] + 2*sc_w_n*G_u_wpv[i]
272                     + i*val[i-1][j] + j*val[i][j-1] ) / (i+j+1);
273         }
274       }
275       mxFree(G_v_wmu);
276       mxFree(G_u_wpv);
277     }
278
279     /* case: aa == b */
280     else if (type == 2)
281     {
282       G_u_wmv = mxCalloc(k+2,sizeof(double));
283       G_v_wpu = mxCalloc(ell+1,sizeof(double));
284       G(G_u_wmv,k+1,u,wmv);
285       G(G_v_wpu,ell,v,wpu);
286
287       /* I(0,0) */
288       val[0][0] = 2* (sc_wpu_n*G_v_wpu[0] + sc_u_n*G_u_wmv[1]
289                 + sc_w_n*G_u_wmv[0] );
290       /* I(0,j) */
291       for (j=1; j<=ell; j++)
292       {
293         val[0][j] = ( 2*sc_wpu_n*G_v_wpu[j]
294                   + 2*( 1-2*(j%2) )*sc_u_n*G_u_wmv[1]
295                   + 2*( 1-2*(j%2) )*sc_w_n*G_u_wmv[0]
296                   + j*val[0][j-1] ) / (j+1);
297       }
298       /* I(i,j) */
299       for (i=1; i<=k; i++)
300       {
301         val[i][0] = ( 2*sc_wpu_n*G_v_wpu[0] + 2*sc_u_n*G_u_wmv[i+1]
302                   + 2*sc_w_n*G_u_wmv[i] - i*val[i-1][0] ) / (i+1);
303         for (j=1; j<=ell; j++)
304         {
305           val[i][j] = ( 2*sc_wpu_n*G_v_wpu[j]
306                     + ( 1-2*(j%2) )*2*sc_u_n*G_u_wmv[i+1]
307                     - ( 1-2*(j%2) )*2*sc_w_n*G_u_wmv[i]
```

```
                    - i*val[i-1][j] - j*val[i][j-1] ) / (i+j+1);
      }
    }
    mxFree(G_u_wmv);
    mxFree(G_v_wpu);
  }

  else
    mexPrintf("Something is wrong! this type does not exist!");
}
```

## A.2 rechte Seite der Symm'schen Integralgleichung

### A.2.1 exakte rechte Seite

```
#include <math.h>
#include "mex.h"

#define EPS 1e-13
#define PI 3.141592653589793116

void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[]);

void rhs(double* b, double* coordinates, double* elements, int nC, int nE, char* g);

void adlp(double* returnvec, double* sx, double a0, double a1, double b0, double b1,
          double n0, double n1, int order);

double arctan(double x0, double x1, double a0, double a1, double b0, double b1);


/* transfers data from matlab to C vice versa */
/* replaces main - function                   */
void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[])
{
  const char* functionName = mexFunctionName();
  char errorMessage[255];
  int j;
  int nE, nC;
  double* elements;
  double* coordinates;
  char* g;
  double* b;


  /* check data */
  if (nlhs != 1) {
    sprintf(errorMessage,"Use b = %s(coordinates,elements,'g').",
            functionName);
    mexErrMsgTxt(errorMessage);
  }

  if (nrhs != 3) {
    sprintf(errorMessage,"Use b = %s(coordinates,elements,'g').",
            functionName);
    mexErrMsgTxt(errorMessage);
  }

```

```
44    /* Read input data */
45    coordinates = mxGetPr(prhs[0]);
46    nC = mxGetM(prhs[0]);          /* number of nodes */
47
48    elements = mxGetPr(prhs[1]);
49    nE = mxGetM(prhs[1]);          /* number of elements */
50
51    /* (K+1/2)g has to be computed -> g is the name (string!) of a matlab */
52    /* function realizing the evaluation                                  */
53    g = mxArrayToString(prhs[2]);
54
55    /* create output vector vecb */
56    plhs[0] = mxCreateDoubleMatrix(nE,1,mxREAL);
57    b = mxGetPr(plhs[0]);
58
59    rhs(b,coordinates,elements,nC,nE,g);
60  }
61
62  /* returns approximation of right-hand-side for Galerkin scheme */
63  /* val(j) = < (K+1/2)*g ; \chi_{T_j} >                          */
64  /*        = (1/2) * < g ; \chi_{T_j} > + < g ; K' \chi_{T_j} >  */
65  /* where K is built with respect to the fundamental solution    */
66  /* of the Laplacian G(z) = -1/(2*PI) * log |z|                  */
67  void rhs(double* brhs, double* coordinates, double* elements,
68           int nC, int nE, char* g) {
69
70    const order = 16;
71    const log_order = 16;
72    int i,j,k,ell;
73    int aidx, bidx, cidx, didx;
74    double a0,a1, b0,b1, c0,c1, d0,d1, n0,n1;
75    double hj, hk;
76    double D, u[2*order], v[2], vn;
77    double singular, regular, regular1[order], regular2[order];
78    double tmp;
79    double *sx, *gsx, *tmpgsx, *tmpgsxlog;
80    double tmpsx[2*order],tmpsxlog[2*log_order];
81    double norm_sxam, norm_sxbm;
82    double *sxlog1, *sxlog2;
83    double* wl1, *wl2;
84    double* tmpvec;
85    mxArray *g_input, *g_output;
86    double *pr;
87
88    /* 16-point Gaussian quadrature on [-1,1] */
89    const double wht[] = { 0.027152459411754, 0.062253523938648, 0.095158511682493,
90                           0.124628971255534, 0.149595988816577, 0.169156519395003,
91                           0.182603415044924, 0.189450610455069, 0.189450610455069,
92                           0.182603415044924, 0.169156519395003, 0.149595988816577,
93                           0.124628971255534, 0.095158511682492, 0.062253523938648,
94                           0.027152459411753 };
95    const double pt[] = { -0.989400934991650,-0.944575023073233,-0.865631202387832,
96                          -0.755404408355003,-0.617876244402644,-0.458016777657227,
97                          -0.281603550779259,-0.095012509837637, 0.095012509837637,
98                           0.281603550779259, 0.458016777657228, 0.617876244402644,
99                           0.755404408355003, 0.865631202387832, 0.944575023073232,
100                          0.989400934991650 };
101
102   /* 16-point logarithmic-weighted Gaussian quadrature on [-1,1] */
103   const double log_pt[] = { 0.38978344871159e-02, 0.23028945616873e-01,
104                             0.58280398306240e-01, 0.10867836509105,
```

```
105                              0.17260945490984,      0.24793705447058,
106                              0.33209454912992,      0.42218391058195,
107                              0.51508247338146,      0.60755612044773,
108                              0.69637565322821,      0.77843256587327,
109                              0.85085026971539,      0.91108685722227,
110                              0.95702557170354,      0.98704780024798};
111    const double log_wht[]= {-0.60791710043591e-01,-0.10291567751758,
112                             -0.12235566204601,     -0.12756924693702,
113                             -0.12301357460007,     -0.11184724485549,
114                             -0.96596385152124e-01,-0.79356664351473e-01,
115                             -0.61850494581965e-01,-0.45435246507727e-01,
116                             -0.31098974751582e-01,-0.19459765927361e-01,
117                             -0.10776254963206e-01,-0.49725428900876e-02,
118                             -0.16782011100512e-02,-0.28235376466844e-03};
119
120    /* collect all evaluations of the Dirichlet data in one matrix   */
121    sx = mxMalloc(2*order*nE*sizeof(double));
122    gsx = mxMalloc(order*nE*sizeof(double));
123    sxlog1 = mxMalloc(2*log_order*nE*sizeof(double));
124    sxlog2 = mxMalloc(2*log_order*nE*sizeof(double));
125    wl1 = mxMalloc(nE*sizeof(double));
126    wl2 = mxMalloc(nE*sizeof(double));
127
128    for (j=0; j<nE; ++j) {
129
130      aidx = (int) elements[j]-1;       /* 1st node of element Tj = [a,b] */
131      a0 = coordinates[aidx];
132      a1 = coordinates[aidx+nC];
133
134      bidx = (int) elements[j+nE]-1;    /* 2nd node of element Tj = [a,b] */
135      b0 = coordinates[bidx];
136      b1 = coordinates[bidx+nC];
137
138      v[0] = b0-a0;
139      v[1] = b1-a1;
140
141      /* gauss quadrature*/
142      g_input = mxCreateDoubleMatrix(order,2,mxREAL);
143      pr = mxGetPr(g_input);
144      for (i=0; i<order; ++i)
145      {
146        pr[i]= 0.5 * ( (1.-pt[i]) * a0 + (1.+pt[i]) * b0 );
147        pr[i+order] = 0.5 * ( (1.-pt[i]) * a1 + (1.+pt[i]) * b1);
148        sx[2*j*order+i] = pr[i];
149        sx[(2*j+1)*order+i] = pr[i+order];
150      }
151      mexCallMATLAB(1,&g_output,1,&g_input,g);
152      tmpgsx = mxGetPr(g_output);
153      for(i=0; i<order; ++i)
154        gsx[j*order+i] = tmpgsx[i];
155      mxDestroyArray(g_input);
156      mxDestroyArray(g_output);
157
158      /* log-weighted gauss quadrature     */
159      /* this is just needed in case a=bb */
160      g_input = mxCreateDoubleMatrix(order,2,mxREAL);
161      pr = mxGetPr(g_input);
162      for (i=0; i<log_order; ++i)
163      {
164        pr[i]= a0 + (1.-log_pt[i]) * v[0];
165        pr[i+log_order] = a1 + (1.-log_pt[i]) * v[1];
```

```
166       sxlog1[2*j*log_order+i] = a0 + (1.-log_pt[i]) * v[0];
167       sxlog1[(2*j+1)*log_order+i] = a1 + (1.-log_pt[i]) * v[1];
168     }
169     mexCallMATLAB(1,&g_output,1,&g_input,g);
170     tmpgsxlog = mxGetPr(g_output);
171     wl1[j] = tmpgsxlog[0] * log_wht[0];
172     for (i=1; i<log_order; ++i)
173       wl1[j] += tmpgsxlog[i]*log_wht[i];
174     mxDestroyArray(g_input);
175     mxDestroyArray(g_output);
176
177     /* log-weighted gauss quadrature    */
178     /* this is just needed in case b=aa */
179     g_input = mxCreateDoubleMatrix(order,2,mxREAL);
180     pr = mxGetPr(g_input);
181     for (i=0; i<log_order; ++i)
182     {
183       pr[i]= a0 + log_pt[i] * v[0];
184       pr[i+log_order] = a1 + log_pt[i] * v[1];
185       sxlog2[2*j*log_order+i] = pr[i];
186       sxlog2[(2*j+1)*log_order+i] = pr[i+log_order];
187     }
188     mexCallMATLAB(1,&g_output,1,&g_input,g);
189     tmpgsxlog = mxGetPr(g_output);
190     wl2[j] = tmpgsxlog[0]*log_wht[0];
191     for (i=1; i<log_order; ++i)
192       wl2[j] += tmpgsxlog[i]*log_wht[i];
193     mxDestroyArray(g_input);
194     mxDestroyArray(g_output);
195   }
196
197   /* starting calculation*/
198   for (j=0; j<nE; ++j)
199   {
200     regular = 0.;
201
202     aidx = (int) elements[j]-1;       /* 1st node of element Tj = [a,b] */
203     a0 = coordinates[aidx];
204     a1 = coordinates[aidx+nC];
205
206     bidx = (int) elements[j+nE]-1;    /* 2nd node of element Tj = [a,b] */
207     b0 = coordinates[bidx];
208     b1 = coordinates[bidx+nC];
209
210     v[0] = b0-a0;
211     v[1] = b1-a1;
212
213     hj = v[0]*v[0] + v[1]*v[1] ;      /* hj = norm(b-a)^2 */
214
215     /* ( \int_{T_j} g(x) ds_x ) / 2 */
216     tmp = wht[0] * gsx[j*order];
217     for (i=1; i<order; ++i)
218       tmp += wht[i] * gsx[j*order+i];
219     brhs[j] = 0.25 * tmp * sqrt(hj);
220
221     for (k=0; k<nE; ++k)
222     {
223       cidx = (int) elements[k]-1;         /* 1st node of element Tk = [c,d] */
224       c0 = coordinates[cidx];
225       c1 = coordinates[cidx+nC];
226
```

```
227       didx = (int) elements[k+nE]-1;      /* 2nd node of element Tk = [c,d] */
228       d0 = coordinates[didx];
229       d1 = coordinates[didx+nC];
230
231       hk = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1);        /* hk = norm(d-c)^2 */
232
233       n0 =  (d1-c1) / sqrt(hk);           /* n = n_x normale vector */
234       n1 =  (c0-d0) / sqrt(hk);
235
236       vn = v[0] * n0 + v[1] * n1;         /* < b-a , n > */
237
238       /* case 2.2: intervals [a,b] and [c,d] are in contact */
239       if ( (a0==d0 && a1==d1) || (c0==b0 && c1==b1) )
240       {
241
242         /* the normal vectors of [a,b] and [c,d] differ */
243         if ( fabs(vn) >= EPS*sqrt(hj) )
244         {
245
246           for (i=0; i<order; ++i)
247           {
248             u[i] = a0 + b0 - 2*sx[2*k*order+i];
249             u[i+order] = a1 + b1 - 2*sx[(2*k+1)*order+i];
250           }
251
252           /* case a = d */
253           if ( a0==d0 && a1==d1 )
254           {
255             for (ell=0; ell<order; ++ell)
256             {
257               D = fabs( u[ell] * v[1] - u[ell+order] * v[0] );
258
259               norm_sxbm = sqrt( (sx[2*k*order+ell]-b0) * (sx[2*k*order+ell]-b0)
260                         + (sx[(2*k+1)*order+ell]-b1) * (sx[(2*k+1)*order+ell]-b1) );
261
262               /* regular1 = log (|sx-b|/|d-c|) */
263               regular1[ell] = log( norm_sxbm / sqrt(hk) );
264               /* regular2 = 1/D * arctan */
265               regular2[ell] = arctan( sx[2*k*order+ell],sx[(2*k+1)*order+ell],a0,a1,
                      b0,b1 ) / D;
266
267             }
268             singular = -vn * sqrt(hk) * wl1[k];
269
270             for (i=0; i<order; ++i)
271             {
272               /* regular = Gauss quadrature of
                                                                  */
273               /* <d,normvec> * regular1 + (<u,n>*<b-a,b-a> - <b-a,n>*<u,b-a>) *
                      regular2 */
274               regular += wht[i] * gsx[k*order+i] * 0.5 * sqrt(hk) * (
275                       ( ( u[i] * n0 + u[i+order]*n1 ) * hj
276                       - ( u[i] * v[0] + u[i+order] * v[1] ) *vn ) *regular2[i]
277                       + vn * regular1[i] ) ;
278             }
279           }
280
281           /* case  b = c */
282           else
283           {
284             for (ell=0; ell<order; ++ell)
```

```
285             {
286               D = fabs( u[ell] * v[1] - u[ell+order] * v[0] );
287
288               norm_sxam = sqrt( (sx[2*k*order+ell]-a0) * (sx[2*k*order+ell]-a0)
289                         + ( sx[(2*k+1)*order+ell]-a1) * (sx[(2*k+1)*order+ell]-a1) );
290               /* regular1 = log (|d-c|/|sx-a|) */
291               regular1[ell] = log( sqrt(hk) / norm_sxam );
292               /* regular2 = 1/D * arctan */
293               regular2[ell] = arctan( sx[2*k*order+ell],sx[(2*k+1)*order+ell],a0,a1,
                      b0,b1 ) / D;
294             }
295
296             /* singular = <d,n> * |d-c| * gauss-log-quadrature */
297             singular = vn * sqrt(hk) * wl2[k];
298
299             for (i=0; i<order; ++i)
300             {
301               /* regular = Gauss quadrature of                                    */
302               /* <d,normvec> * regular1 + (<u,n>*<d,d> - <d,n>*<u,d>) * regular2 */
303               regular += wht[i] * gsx[k*order+i] * 0.5 * sqrt(hk) *
304                       ( ( (u[i] * n0 + u[i+order] * n1) * hj
305                       -(u[i] * v[0] + u[i+order] * v[1] ) *vn ) * regular2[i]
306                       + vn * regular1[i] );
307             }
308           }
309           /* b(j) = rhs(j) = sum_[a,b]  1/(2*pi*|b-a|)(regular+singular) */
310           brhs[j] += ( regular + singular ) / ( 2. * PI * sqrt(hj) );
311         }
312       }
313       /* case 1: intervals [a,b] and [c,d] are not in contact */
314       else
315       {
316         tmp = 0.;
317         for (i=0; i<order; ++i)
318         {
319           tmpsx[i] = sx[2*k*order+i];
320           tmpsx[i+order] = sx[(2*k+1)*order+i];
321         }
322         tmpvec = mxMalloc(order*sizeof(double));
323         adlp(tmpvec,tmpsx,a0,a1,b0,b1,n0,n1,order);
324         for (i=0; i<order; ++i)
325           tmp += wht[i]*gsx[k*order+i]*tmpvec[i];
326         /* b(j) = rhs(j) = sum_[a,b]  1/(2*pi*|b-a|) < g;K'_chi_{[a,b]} > */
327         brhs[j] += tmp*0.5*sqrt(hk);
328
329         mxFree(tmpvec);
330       }
331     }
332   }
333
334   /* Free memory */
335   mxFree(sx);
336   mxFree(gsx);
337   mxFree(sxlog1);
338   mxFree(sxlog2);
339   mxFree(wl1);
340   mxFree(wl2);
341 }
342
343
344
```

```
345 /* ********************************************************* *
346  * computes double layer potential K'                        *
347  * K'\phi = -1/(2pi) \int_Gamma <x-y,n(x)>/|x-y|^2 g(y) ds_y *
348  * ********************************************************* */
349 void adlp(double* returnvec, double* sx, double a0, double a1, double b0, double b1,
350           double n0, double n1, int order)
351 {
352   int j;
353   double d[2], c[2];
354   double sxam[2], sxbm[2];
355   double D, norm_d, norm2_c, norm2_d, dn;
356   double tmp1, tmp2, maxtmp;
357
358   d[0] = b0-a0;                       /* d = b-a */
359   d[1] = b1-a1;
360   norm2_d = d[0]*d[0]+d[1]*d[1];      /* norm2_d = |d|^2 */
361   norm_d = sqrt(norm2_d);             /* norm_d = |d| */
362   dn = d[0]*n0+d[1]*n1;               /* dn = < d,n > */
363
364   for(j=0; j<order; ++j)
365   {
366     c[0] = a0+b0-2.*sx[j];              /* c = a+b-2*x */
367     c[1] = a1+b1-2.*sx[j+order];
368     D = fabs( c[0]*d[1]-c[1]*d[0] );    /* D = |c,d| */
369     norm2_c = c[0]*c[0]+c[1]*c[1];      /* norm2_c = |c|^2 */
370
371     /* case 2.*/
372     if ( D<=EPS*norm_d )
373     {
374       if ( (norm2_c <= norm2_d) || (fabs(dn) <= EPS*norm_d) )
375         returnvec[j] = 0.;
376
377       /* case 2.2. */
378       else
379       {
380         tmp1 = b0+b1-sx[j]-sx[j+order];
381         tmp2 = a0+a1-sx[j]-sx[j+order];
382
383         maxtmp = fabs(tmp1);
384         if(fabs(tmp2)>maxtmp)
385           maxtmp = fabs(tmp2);
386
387         if ( fabs(tmp1-tmp2) <= EPS*maxtmp )
388           returnvec[j] = 0.;
389         else if ( (tmp1==0. && fabs(tmp2)<=EPS) || (tmp2==0. && fabs(tmp1)<=EPS) )
390           returnvec[j] = 0.;
391         else if ( tmp1*tmp2 < 0. && maxtmp<=EPS )
392           returnvec[j] = 0.;
393         else
394           returnvec[j] = dn*log(tmp1/tmp2)/(2.*PI*norm_d);
395       }
396     }
397     /* case 1. */
398     else
399     {
400       sxam[0] = sx[j]-a0;
401       sxam[1] = sx[j+order]-a1;
402       sxbm[0] = sx[j]-b0;
403       sxbm[1] = sx[j+order]-b1;
404       returnvec[j] = ( dn*log( sqrt(sxbm[0]*sxbm[0]+sxbm[1]*sxbm[1])/
405                       sqrt(sxam[0]*sxam[0]+sxam[1]*sxam[1]) )
```

```
406                     + (n1*d[0]-n0*d[1])*(c[1]*d[0]-c[0]*d[1])
407                       *arctan(sx[j],sx[j+order],a0,a1,b0,b1)/D ) /(2*PI*norm_d);
408       }
409     }
410   }
411
412   /* ********************************************** *
413    * computes function arctan                       *
414    *   arctan ( ( |b-a|^2 + < a+b-2*x , b-a > ) / D ) *
415    * + arctan ( ( |b-a|^2 - < a+b-2*x , b-a > ) / D ) *
416    * ********************************************** */
417   double arctan(double x0, double x1, double a0, double a1, double b0, double b1)
418   {
419
420     double d[2], c[2], axm[2], bxm[2];
421     double D;
422     double val;
423     double norm2_c, norm2_d;
424
425     c[0] = a0 + b0 - 2. * x0;
426     c[1] = a1 + b1 - 2. * x1;
427     d[0] = b0 - a0;
428     d[1] = b1 - a1;
429     axm[0] = a0 - x0;
430     axm[1] = a1 - x1;
431     bxm[0] = b0 - x0;
432     bxm[1] = b1 - x1;
433     norm2_c = c[0] * c[0] + c[1] * c[1];
434     norm2_d = d[0] * d[0] + d[1] * d[1];
435
436     D = fabs( c[0] * d[1] - c[1] * d[0] );
437
438     /* in this case we use the addition theorem */
439     if (norm2_d != norm2_c)
440     {
441       val = atan( D / (2.* (axm[0]*bxm[0]+axm[1]*bxm[1]) ) );
442
443       /* in this case we have to add pi*/
444       if (norm2_d > norm2_c)
445         val += PI;
446       return val;
447     }
448
449     /* in this case we cannot use the addition theorem */
450     else
451     {
452       val = atan( ( bxm[0] * d[0] + bxm[1] * d[1] ) * 2. / D )
453           + atan( ( axm[0] * d[0] + axm[1] * d[1] ) * (-2.) / D );
454       return val;
455     }
456   }
```

### A.2.2 approximative rechte Seite

```
1   /* ********************************************************** *
2    * returns approximation of right-hand-side for Galerkin scheme *
3    * val(j) = < (K+1/2)*g ; \chi_{T_j} >                        *
4    *        = (1/2) * < g ; \chi_{T_j} > + < g ; K' \chi_{T_j} >  *
5    * where K is built with respect to the fundamental solution    *
6    * of the Laplacian G(z) = -1/(2*PI) * log |                    *
```

```
7   ************************************************************ */
8  #include <math.h>
9  #include "mex.h"
10
11 #define EPS 1e-15
12 #define PI 3.141592653589793116
13
14 void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[]);
15
16 void rhs(double* returnvalue, double* coordinates, double* elements,
17          int nC, int nE, double* gx, double eta);
18
19 void computeKij(double* val0, double* val1,
20                 double a0, double a1, double b0, double b1,
21                 double c0, double c1, double d0, double d1, double eta);
22 void computeKij_analytic(double* I0, double* I1,
23                          double a0, double a1, double b0, double b1,
24                          double c0, double c1, double d0, double d1);
25 void computeKijT_analytic(double* I0, double* I1,
26                           double a0, double a1, double b0, double b1,
27                           double c0, double c1, double d0, double d1);
28 void computeKij_semianalytic(double* I0, double* I1,
29                              double a0, double a1, double b0, double b1,
30                              double c0, double c1, double d0, double d1);
31 void computeKijT_semianalytic(double* I0, double* I1,
32                               double a0, double a1, double b0, double b1,
33                               double c0, double c1, double d0, double d1);
34
35 double dlpWrapper(int k, double u[2],double v[2]);
36 double dlp(int k, double a, double b, double c);
37
38 double ptoseg(double p0, double p1, double a0, double a1, double b0, double b1);
39 double dist2(double a0, double a1, double b0, double b1,
40              double c0, double c1, double d0, double d1);
41
42 /* ----------------- MEX interface ----------------- */
43 void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[])
44 {
45   int i;
46   const char* functionName = mexFunctionName();
47   char errorMessage[255];
48   int nE, nC;
49   double* elements;
50   double* coordinates;
51   double eta;
52   char* g;
53   double* vecb;
54   mxArray *g_input, *g_output;
55   double *pr, *gx;
56
57   /* check data */
58   if (nlhs !=1) {
59     sprintf(errorMessage,"Use either b = %s(coordinates,elements,'g')\n or b = %s(
         coordinates,elements,'g',eta).",functionName,functionName);
60     mexErrMsgTxt(errorMessage);
61   }
62
63   if ((nrhs != 3) && (nrhs != 4) ) {
64     sprintf(errorMessage,"Use either b = %s(coordinates,elements,'g')\n or b = %s(
         coordinates,elements,'g',eta).",functionName,functionName);
65     mexErrMsgTxt(errorMessage);
```

```
66    }
67
68    if (nrhs == 4) {
69      pr = mxGetPr(prhs[3]);
70      eta = *pr;
71    }
72    else {
73      eta = 0.;
74    }
75
76    /* Read input data */
77    coordinates = mxGetPr(prhs[0]);
78    nC = mxGetM(prhs[0]);          /* number of nodes */
79
80    elements = mxGetPr(prhs[1]);
81    nE = mxGetM(prhs[1]);          /* number of elements */
82
83    /* (K+1/2)g has to be computed -> g is the name (string!) of a matlab */
84    /* function realizing the evaluation                                   */
85    g = mxArrayToString(prhs[2]);
86
87    /* create output vector vecb */
88    plhs[0] = mxCreateDoubleMatrix(nE,1,mxREAL);
89    vecb = mxGetPr(plhs[0]);
90
91    /* compute all evaluations of function g */
92    mexCallMATLAB(1,&g_output,1,prhs,g);
93    gx = mxGetPr(g_output);
94
95    mxDestroyArray(g_input);
96
97    rhs(vecb,coordinates,elements,nC,nE,gx,eta);
98
99    mxDestroyArray(g_output);
100 }
101
102 /* ********************************************************** *
103  * returns approximation of right-hand-side for Galerkin scheme *
104  * val(j) = < (K+1/2)*g ; \chi_{T_j} >                        *
105  *        = (1/2) * < g ; \chi_{T_j} > + < g ; K' \chi_{T_j} >  *
106  * where K is built with respect to the fundamental solution    *
107  * of the Laplacian G(z) = -1/(2*PI) * log |                    *
108  ********************************************************** */
109 void rhs(double* returnvalue, double* coordinates, double* elements,
110          int nC, int nE, double* gx, double eta)
111 {
112   int i,j;
113   double linetest1, linetest2;
114   double I0,I1;
115   int aidx, bidx, cidx, didx;
116   double a0,a1, b0,b1, c0,c1, d0,d1;
117   double hi, hj;
118   double *K = mxCalloc(nE*nC,sizeof(double));
119
120   /* Fill matrix < (K+1/2)g,\chi> */
121   for (j=0; j<nE; ++j) {
122
123     cidx = (int) elements[j]-1;        /* 1st node of element Tj = [c,d] */
124     c0 = coordinates[cidx];
125     c1 = coordinates[cidx+nC];
126
```

```
127     didx = (int) elements[j+nE]-1;      /* 2nd node of element Tj = [c,d] */
128     d0 = coordinates[didx];
129     d1 = coordinates[didx+nC];
130
131     for (i=0; i<nE; ++i) {
132
133       aidx = (int) elements[i]-1;       /* 1st node of element Ti = [a,b] */
134       a0 = coordinates[aidx];
135       a1 = coordinates[aidx+nC];
136
137       bidx = (int) elements[i+nE]-1;    /* 2nd node of element Ti = [a,b] */
138       b0 = coordinates[bidx];
139       b1 = coordinates[bidx+nC];
140
141       hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
142       hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
143
144       /* if two intervals are in line the scalar-product with */
145       /* the normal vector is zero                            */
146       /*  --> double-layer-potential is zero!                 */
147       /* this is tested beneath and in this case nothing      */
148       /* should be calculated!                                */
149       linetest1 = fabs( (a0+b0-2*c0)*(b1-a1)-(a1+b1-2*c1)*(b0-a0) );
150       linetest2 = fabs( (a0+b0-2*d0)*(b1-a1)-(a1+b1-2*d1)*(b0-a0) );
151
152       if ( linetest1 > EPS || linetest2 > EPS ) {
153         computeKij(&I0,&I1, a0,a1,b0,b1, c0,c1,d0,d1,eta);
154         K[j+aidx*nE] += I0 - I1;
155         K[j+bidx*nE] += I0 + I1;
156       }
157     }
158
159     /* 1/2 * <\eta_j,\chi_k> */
160     K[j+cidx*nE] += 0.25*sqrt(hj);
161     K[j+didx*nE] += 0.25*sqrt(hj);
162   }
163   /* compute (K+1/2)*g */
164   for (j=0; j<nE; ++j)
165   {
166     returnvalue[j] = 0.;
167     for (i=0; i<nC; ++i)
168       returnvalue[j] += K[j+i*nE]*gx[i];
169   }
170   mxFree(K);
171 }
172
173 /* ----------------- WRAPPER: Double Integrals for DLP ----------------- */
174
175 void computeKij(double* I0, double* I1,
176                 double a0, double a1, double b0, double b1,
177                 double c0, double c1, double d0, double d1,
178                 double eta) {
179   /*
180    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
181    * OUTPUT: Galerkin integral
182    *         -1/(2pi) \int_{Tj} \int_{Ti} 1/2 <y-x,n(y)>/|x-y|^2 \phi(y) ds_y ds_x
183    *         \phi(\gamma(t))\in[-1,1]: \phi(\gamma(t)) = 1/2 + 1/2t
184    *                                  resp. \phi(\gamma(t)) = 1/2 - 1/2t
185    *         thus \phi(\gamma) is the hat function on [-1,1],
186    *         resp. \phi is the hat function on T_i
187    */
```

```
188
189   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
190   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
191   double tmp, det;
192   double n[2];
193   double x[2], y[2], z[2];
194   double zxp[2], zxm[2], zyp[2], zym[2];
195   double yn, zn, zypn, zymn;
196   double lambda, mu;
197   int swap;
198
199   /* ************************************************************* *
200    * For stability reasons, we guarantee   hj <= hi   to ensure that *
201    * outer integration is over smaller domain. This is done by       *
202    * swapping Tj and Ti if necessary.                                *
203    * ************************************************************* */
204   if ( hi<hj )
205     swap = 1;
206   else
207     swap = 0;
208
209   if (eta == 0) { /* compute all matrix entries analytically */
210
211     if (swap == 0)
212       computeKij_analytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
213     else
214       computeKij_analytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
215   }
216   else { /* compute admissible matrix entries semi-analytically */
217
218     if ( dist2(a0,a1,b0,b1,c0,c1,d0,d1) > eta*sqrt(hj) ) {
219       if (swap == 0)
220         computeKij_semianalytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
221       else
222         computeKijT_semianalytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
223     }
224     else {
225       if (swap == 0)
226         computeKij_analytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
227       else
228         computeKijT_analytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
229     }
230   }
231 }
232
233
234 /* -------------- Double Integrals for DLP: analytic, no swap -------------- */
235
236 void computeKij_analytic(double* I0, double* I1,
237                          double a0, double a1, double b0, double b1,
238                          double c0, double c1, double d0, double d1) {
239   /*
240    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
241    * OUTPUT: Galerkin integral
242    *         -1/(2pi) \int_{Tj} \int_{Ti} 1/2 <y-x,n(y)>/|x-y|^2 \phi(y) ds_y ds_x
243    *         \gamma(\phi(t))\in[-1,1]: \gamma(\phi(t)) = 1/2 + 1/2t
244    *                                  resp. \gamma(\phi(t)) = 1/2 - 1/2t
245    *         thus \phi(\gamma) is the hat function on [-1,1],
246    *         resp. \phi is the hat function on T_i
247    */
248
```

```
249   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
250   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
251   double tmp, det;
252   double n[2];
253   double x[2], y[2], z[2];
254   double zxp[2], zxm[2], zyp[2], zym[2];
255   double yn, zn, zypn, zymn;
256   double lambda, mu;
257
258   n[0] = ( b1-a1 ) / sqrt(hi);  /* n = n_y normale vector */
259   n[1] = ( a0-b0 ) / sqrt(hi);
260
261   x[0] = b0 - a0;               /* x = b-a */
262   x[1] = b1 - a1;
263   y[0] = c0 - d0;               /* y = c-d */
264   y[1] = c1 - d1;
265   z[0] = a0 + b0 - c0 - d0;     /* z = a+b-c-d */
266   z[1] = a1 + b1 - c1 - d1;
267
268   zxp[0] = z[0] + x[0];           /* zxp = z+x = 2b-c-d */
269   zxp[1] = z[1] + x[1];
270   zxm[0] = z[0] - x[0];           /* zxm = z-x = 2a-c-d */
271   zxm[1] = z[1] - x[1];
272   zyp[0] = z[0] + y[0];           /* zyp = z+y = a+b-2d */
273   zyp[1] = z[1] + y[1];
274   zym[0] = z[0] - y[0];           /* zym = z-y = a+b-2c */
275   zym[1] = z[1] - y[1];
276
277   yn = y[0]*n[0]+y[1]*n[1];       /* yn = <y,n> */
278   zn = z[0]*n[0]+z[1]*n[1];       /* zn = <z,n> */
279   zypn = zyp[0]*n[0]+zyp[1]*n[1]; /* zypn = <z+y,n> = <a+b-2d,n> */
280   zymn = zym[0]*n[0]+zym[1]*n[1]; /* zymn = <z-y,n> = <a+b-2c,n> */
281
282   /* There hold different recursion formulae if Ti and Tj *
283    * are parallel (det = 0) or not                        */
284   det = x[0]*y[1] - x[1]*y[0];
285
286   /* case that x and y are linearly dependent, i.e., Ti and Tj are parallel */
287   if ( fabs(det) <= EPS*sqrt(hi*hj) ) {
288
289     if ( fabs(y[0]) < fabs(y[1]) )
290       lambda = x[1] / y[1];
291     else
292       lambda = x[0] / y[0];
293
294     *I0 = zn * ( dlpWrapper(0, y, zxp) + dlpWrapper(0, y, zxm)
295         - lambda * dlpWrapper(1, x, zyp) + lambda * dlpWrapper(1, x, zym) );
296
297     *I1 = 0.5 * zn * ( dlpWrapper(0, y, zxp) - dlpWrapper(0, y, zxm)
298         - lambda * dlpWrapper(2, x, zyp) + lambda * dlpWrapper(2, x, zym) );
299   }
300
301   else { /* case that x and y are linearly independent */
302
303     /* if intervals are in touch recursion formulae reduce */
304     if (b0==c0 && b1==c1) {
305
306       *I0 = 2 * ( zypn * dlpWrapper(0, x, zyp) + yn * dlpWrapper(1, y, zxm)
307           + zn * dlpWrapper(0, y, zxm) );
308
309       *I1 = 0.5 * ( 2 * zypn * dlpWrapper(1, x, zyp)
```

```
310             - 2 * yn * dlpWrapper(1, y, zxm) - 2 * zn * dlpWrapper(0, y, zxm) + *I0 );
311       }
312
313       else if (a0==d0 && a1==d1) {
314
315         *I0 = 2 * ( zymn * dlpWrapper(0, x, zym) + yn * dlpWrapper(1, y, zxp)
316             + zn * dlpWrapper(0, y, zxp) );
317
318         *I1 = 0.5 * ( 2 * zymn * dlpWrapper(1, x ,zym)
319             + 2 * yn * dlpWrapper(1, y, zxp) + 2 * zn * dlpWrapper(0, y, zxp) - *I0 );
320       }
321
322       else {
323
324         lambda = (z[0]*y[1] - z[1]*y[0]) /det;
325         mu = (x[0]*z[1] - x[1]*z[0]) /det;
326
327         *I0 = (mu+1) * zypn * dlpWrapper(0, x, zyp)
328             - (mu-1) * zymn * dlpWrapper(0, x ,zym)
329             + (lambda+1) * yn * dlpWrapper(1, y, zxp)
330             + (lambda+1) * zn * dlpWrapper(0, y, zxp)
331             - (lambda-1) * yn * dlpWrapper(1, y, zxm)
332             - (lambda-1) * zn * dlpWrapper(0, y, zxm);
333
334         *I1 = 0.5 * ( (mu+1) * zypn * dlpWrapper(1, x, zyp)
335             - (mu-1) * zymn * dlpWrapper(1, x ,zym)
336             + (lambda+1) * yn * dlpWrapper(1, y, zxp)
337             + (lambda+1) * zn * dlpWrapper(0, y, zxp)
338             + (lambda-1) * yn * dlpWrapper(1, y, zxm)
339             + (lambda-1) * zn * dlpWrapper(0, y, zxm) - lambda * *I0 );
340       }
341     }
342
343     *I0 *= -0.125 * sqrt(hi*hj) / PI; /* = - 1/(2*PI) * |Ti|/2 * |Tj|/2 * I0 */
344     *I1 *= -0.125 * sqrt(hi*hj) / PI; /* = - 1/(2*PI) * |Ti|/2 * |Tj|/2 * I1 */
345 }
346
347
348  /* ----------------- Double Integrals for DLP: analytic, swap ----------------- */
349
350 void computeKijT_analytic(double* I0, double* I1,
351                           double a0, double a1, double b0, double b1,
352                           double c0, double c1, double d0, double d1) {
353   /*
354    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
355    * OUTPUT: Galerkin integral
356    *         -1/(2pi) \int_{Ti} \int_{Tj} 1/2 <y-x,n(y)>/|x-y|^2 \phi(y) ds_y ds_x
357    *         \gamma(\phi(t))\in[-1,1]: \gamma(\phi(t)) = 1/2 + 1/2t
358    *         resp. \gamma(\phi(t)) = 1/2 - 1/2t
359    *         thus \phi(\gamma) is the hat function on [-1,1],
360    *         resp. \phi is the hat function on T_i
361    */
362
363   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
364   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
365   double tmp, det;
366   double n[2];
367   double x[2], y[2], z[2];
368   double zxp[2], zxm[2], zyp[2], zym[2];
369   double xn, zn, zxpn, zxmn;
370   double lambda, mu;
```

```
371
372   /* n = n_y normale vector */
373   n[0] = ( b1-a1 ) / sqrt(hi);
374   n[1] = ( a0-b0 ) / sqrt(hi);
375
376   x[0] = c0 - d0;               /* x = c-d */
377   x[1] = c1 - d1;
378   y[0] = b0 - a0;               /* y = b-a */
379   y[1] = b1 - a1;
380   z[0] = a0 + b0 - c0 - d0;     /* z = a+b-c-d */
381   z[1] = a1 + b1 - c1 - d1;
382
383   zxp[0] = z[0] + x[0];           /* zxp = z+x = a+b-2d */
384   zxp[1] = z[1] + x[1];
385   zxm[0] = z[0] - x[0];           /* zxm = z-x = a+b-2c */
386   zxm[1] = z[1] - x[1];
387   zyp[0] = z[0] + y[0];           /* zyp = z+y = 2b-c-d */
388   zyp[1] = z[1] + y[1];
389   zym[0] = z[0] - y[0];           /* zym = z-y = 2a-c-d */
390   zym[1] = z[1] - y[1];
391
392   xn = x[0]*n[0]+x[1]*n[1];       /* yn = <y,n> */
393   zn = z[0]*n[0]+z[1]*n[1];       /* zn = <z,n> */
394   zxpn = zxp[0]*n[0]+zxp[1]*n[1]; /* zxpn = <z+x,n> = <a+b,n> */
395   zxmn = zxm[0]*n[0]+zxm[1]*n[1]; /* zxmn = <z-x,n> = <a+b-2c,n> */
396
397
398   /* There hold different recursion formulae if Ti and Tj *
399    * are parallel (det = 0) or not                        */
400   det = x[0]*y[1] - x[1]*y[0];
401
402   /* case that x and y are linearly dependent, i.e., Ti and Tj are parallel */
403   if ( fabs(det) <= EPS*sqrt(hi*hj) ) {
404
405     if ( fabs(y[0]) < fabs(y[1]) )
406       lambda = x[1] / y[1];
407     else
408       lambda = x[0] / y[0];
409
410     *I0 = zn * ( dlpWrapper(0, y, zxp) + dlpWrapper(0, y, zxm)
411         - lambda * dlpWrapper(1, x, zyp) + lambda * dlpWrapper(1, x, zym) );
412
413     *I1 = zn * ( dlpWrapper(1, y, zxp) + dlpWrapper(1, y, zxm)
414         - lambda * dlpWrapper(1, x, zyp) - lambda * dlpWrapper(1, x, zym)
415         + 0.5* lambda * ( dlpWrapper(0, y, zxp) - dlpWrapper(0, y, zxm)
416         - lambda * dlpWrapper(2, x, zyp) + lambda * dlpWrapper(2, x, zym) ) );
417   }
418
419   else { /* case that x and y are linearly independent */
420
421     /* if intervals are in touch recursion formulae reduce */
422     if (b0==c0 && b1==c1) {
423
424       *I0 = 2 * ( zxpn * dlpWrapper(0, y, zxp) + xn * dlpWrapper(1, x, zym)
425           + zn * dlpWrapper(0, x, zym) );
426
427       *I1 = 0.5 * ( 2 * zxpn * dlpWrapper(1, y, zxp)
428           - 2 * xn * dlpWrapper(1, x, zym) - 2 * zn * dlpWrapper(0, x, zym) + *I0 );
429    }
430
431     else if (a0==d0 && a1==d1) {
```

```
432
433       *I0 = 2 * ( zxmn * dlpWrapper(0, y, zxm) + xn * dlpWrapper(1, x, zyp)
434           + zn * dlpWrapper(0, x, zyp) );
435
436       *I1 = 0.5 * ( 2 * zxmn * dlpWrapper(1, y ,zxm)
437           + 2 * xn * dlpWrapper(1, x, zyp) + 2 * zn * dlpWrapper(0, x, zyp) - *I0 );
438     }
439
440     else {
441
442       mu = (z[0]*y[1] - z[1]*y[0]) /det;
443       lambda = (x[0]*z[1] - x[1]*z[0]) /det;
444
445       *I0 = (mu+1) * zxpn * dlpWrapper(0, y, zxp)
446           - (mu-1) * zxmn * dlpWrapper(0, y ,zxm)
447           + (lambda+1) * xn * dlpWrapper(1, x, zyp)
448           + (lambda+1) * zn * dlpWrapper(0, x, zyp)
449           - (lambda-1) * xn * dlpWrapper(1, x, zym)
450           - (lambda-1) * zn * dlpWrapper(0, x, zym);
451
452       *I1 = 0.5 * ( (mu+1) * zxpn * dlpWrapper(1, y, zxp)
453           - (mu-1) * zxmn * dlpWrapper(1, y ,zxm)
454           + (lambda+1) * xn * dlpWrapper(1, x, zyp)
455           + (lambda+1) * zn * dlpWrapper(0, x, zyp)
456           + (lambda-1) * xn * dlpWrapper(1, x, zym)
457           + (lambda-1) * zn * dlpWrapper(0, x, zym) - lambda * *I0 );
458     }
459   }
460
461   *I0 *= -0.125 * sqrt(hi*hj) / PI; /* = - 1/(2*PI) * |Ti|/2 * |Tj|/2 * I0 */
462   *I1 *= -0.125 * sqrt(hi*hj) / PI; /* = - 1/(2*PI) * |Ti|/2 * |Tj|/2 * I1 */
463 }
464
465
466 /* -------------- Double Integrals for DLP: semianalytic, no swap -------------- */
467
468 void computeKij_semianalytic(double* I0, double* I1,
469                              double a0, double a1, double b0, double b1,
470                              double c0, double c1, double d0, double d1) {
471   /*
472    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
473    * OUTPUT: Galerkin integral
474    *         -1/(2pi) \int_{Tj} \int_{Ti} 1/2 <y-x,n(y)>/|x-y|^2 \phi(y) ds_y ds_x
475    *         \gamma(\phi(t))\in[-1,1]: \gamma(\phi(t)) = 1/2 + 1/2t
476    *         resp. \gamma(\phi(t)) = 1/2 - 1/2t
477    *         thus \phi(\gamma) is the hat function on [-1,1],
478    *         resp. \phi is the hat function on T_i
479    */
480
481   int i;
482   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
483   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
484   double n[2];
485   double x[2], y[2], z[2];
486   double v[2];
487   double vn;
488   double I0tmp = 0., I1tmp = 0.;
489
490   /* 16-point Gaussian quadrature on [-1,1] */
491   const order = 16;
492   const double wht[] = { 0.027152459411754, 0.062253523938648, 0.095158511682493,
```

```
493                          0.124628971255534, 0.149595988816577, 0.169156519395003,
494                          0.182603415044924, 0.189450610455069, 0.189450610455069,
495                          0.182603415044924, 0.169156519395003, 0.149595988816577,
496                          0.124628971255534, 0.095158511682492, 0.062253523938648,
497                          0.027152459411753 };
498   const double pt[] = { -0.989400934991650,-0.944575023073233,-0.865631202387832,
499                         -0.755404408355003,-0.617876244402644,-0.458016777657227,
500                         -0.281603550779259,-0.095012509837637, 0.095012509837637,
501                          0.281603550779259, 0.458016777657228, 0.617876244402644,
502                          0.755404408355003, 0.865631202387832, 0.944575023073232,
503                          0.989400934991650 };
504
505   /* n = n_y normale vector */
506   n[0] = ( b1-a1 ) / sqrt(hi);
507   n[1] = ( a0-b0 ) / sqrt(hi);
508
509   x[0] = b0 - a0;               /* x = b-a */
510   x[1] = b1 - a1;
511   y[0] = c0 - d0;               /* y = c-d */
512   y[1] = c1 - d1;
513   z[0] = a0 + b0 - c0 - d0;     /* z = a+b-c-d */
514   z[1] = a1 + b1 - c1 - d1;
515
516
517   for (i=0; i<order; ++i) {
518
519     v[0] = pt[i] * y[0] + z[0];
520     v[1] = pt[i] * y[1] + z[1];
521
522     vn = v[0]*n[0]+v[1]*n[1];
523
524     I0tmp +=  wht[i] * vn * dlpWrapper(0, x, v);
525     I1tmp +=  wht[i] * vn * dlpWrapper(1, x, v);
526   }
527
528   *I0 = -0.125 * sqrt(hi*hj) * I0tmp / PI; /* = - 1/(2*PI) * |Ti|/2 * |Tj|/2 * val0
        */
529   *I1 = -0.125 * sqrt(hi*hj) * I1tmp / PI; /* = - 1/(2*PI) * |Ti|/2 * |Tj|/2 * val1
        */
530  }
531
532
533  /* -------------- Double Integrals for DLP: semianalytic, swap-------------- */
534
535 void computeKijT_semianalytic(double* I0, double* I1,
536                               double a0, double a1, double b0, double b1,
537                               double c0, double c1, double d0, double d1) {
538   /*
539    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
540    * OUTPUT: Galerkin integral
541    *         -1/(2pi) \int_{Ti} \int_{Tj} 1/2 <y-x,n>/|x-y|^2 \phi(y) ds_y ds_x
542    *         \gamma(\phi(t))\in[-1,1]: \gamma(\phi(t)) = 1/2 + 1/2t
543    *         resp. \gamma(\phi(t)) = 1/2 - 1/2t
544    *         thus \phi(\gamma) is the hat function on [-1,1],
545    *         resp. \phi is the hat function on T_i
546    */
547
548   int i;
549   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
550   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
551   double n[2];
```

```
552   double x[2], y[2], z[2];
553   double v[2];
554   double xn, zn;
555   double I0tmp = 0., I1tmp = 0.;
556
557   /* 16-point Gaussian quadrature on [-1,1] */
558   const order = 16;
559   const double wht[] = { 0.027152459411754, 0.062253523938648, 0.095158511682493,
560                          0.124628971255534, 0.149595988816577, 0.169156519395003,
561                          0.182603415044924, 0.189450610455069, 0.189450610455069,
562                          0.182603415044924, 0.169156519395003, 0.149595988816577,
563                          0.124628971255534, 0.095158511682492, 0.062253523938648,
564                          0.027152459411753 };
565   const double pt[] = { -0.989400934991650,-0.944575023073233,-0.865631202387832,
566                         -0.755404408355003,-0.617876244402644,-0.458016777657227,
567                         -0.281603550779259,-0.095012509837637, 0.095012509837637,
568                          0.281603550779259, 0.458016777657228, 0.617876244402644,
569                          0.755404408355003, 0.865631202387832, 0.944575023073232,
570                          0.989400934991650 };
571
572   /* n = n_y normale vector */
573   n[0] = ( b1-a1 ) / sqrt(hi);
574   n[1] = ( a0-b0 ) / sqrt(hi);
575
576   x[0] = c0 - d0;                /* x = c-d */
577   x[1] = c1 - d1;
578   y[0] = b0 - a0;                /* y = b-a */
579   y[1] = b1 - a1;
580   z[0] = a0 + b0 - c0 - d0;      /* z = a+b-c-d */
581   z[1] = a1 + b1 - c1 - d1;
582
583
584   for (i=0; i<order; ++i) {
585
586     v[0] = pt[i] * y[0] + z[0];
587     v[1] = pt[i] * y[1] + z[1];
588
589     xn = x[0]*n[0]+x[1]*n[1];
590     zn = z[0]*n[0]+z[1]*n[1];
591
592
593     I0tmp +=  wht[i] * ( xn * dlpWrapper(1, x, v) + zn * dlpWrapper(0,x,v) );
594     I1tmp +=  wht[i] * pt[i] * ( xn * dlpWrapper(1, x, v) + zn * dlpWrapper(0,x,v));
595   }
596
597   *I0 = -0.125 * sqrt(hi*hj) * I0tmp / PI;
598   *I1 = -0.125 * sqrt(hi*hj) * I1tmp / PI;
599  }
600
601 /* ----------------- DLP ----------------- */
602
603 double dlpWrapper(int k, double u[2],double v[2]) {
604
605   /*
606    * INPUT: Vectors u,v,w \in \R^2, integer k
607    * OUTPUT: value of DLP-type integral
608    *          \int_{-1}^{+1} t^k 1 / |t*u+v|^2 dt
609    */
610
611   double a = u[0]*u[0] + u[1]*u[1];          /* a = <u,u> */
612   double b = 2 * ( u[0]*v[0] + u[1]*v[1] ); /* b = 2 <u,v> */
```

```
613   double c = v[0]*v[0] + v[1]*v[1];          /* c = <v,v> */
614   return dlp(k,a,b,c);
615 }
616
617
618 double dlp(int k, double a, double b, double c) {
619   /*
620    * INPUT: scalars a,c>0 and b \in \R, integer k
621    * OUPUT: value of DLP-type integral
622    *        \int_{-1}^{+1} t^k 1 / (a*t^2+b*t+c) dt
623    */
624
625
626   double val, val0, val1, val2;
627   double tmp;
628   double D;
629
630   /* Ensure that discriminant is either positive or zero */
631   tmp = 4*a*c - b*b; /* Note that by theory tmp >= 0 */
632
633   if (tmp > EPS * 4*a*c)
634     D = sqrt(tmp);
635   else
636     D = 0;
637
638   /* The case k=0 */
639   if (D == 0)
640   {
641     val0 = 2 / ( c - a );
642   }
643   else { /* also D > 0 */
644     tmp = c - a;
645     if (fabs(tmp) < EPS * fabs(c))
646       val0 = 0.5*PI;
647     else if (a < c)
648       val0 = atan( D /tmp );
649     else
650       val0 = atan( D /tmp ) + PI;
651     val0 *= 2/D;
652   }
653   val = val0;
654
655   /* The case k=1 */
656   if (k>=1) {
657     val1 = -b*val0;
658
659     tmp = a+b+c;
660     if (fabs(tmp) > EPS*fabs(a))
661       val1 += log(tmp);
662
663     tmp = a-b+c;
664     if (fabs(tmp) > EPS*fabs(a))
665       val1 -= log(tmp);
666
667     val1 /= (2*a);
668     val = val1;
669   }
670
671   /* The case k=2 */
672   if (k==2) {
673     val2 = ( 2 - b * val1 - c * val0 ) / a;
```

```
674       val = val2;
675     }
676
677     return val;
678   }
679
680   /* ---------------------- Dist(Ti,Tj) ---------------------- */
681   /* The closest distance between two segments is either zero  */
682   /* if they intersect or the distance from one of the lines'  */
683   /* end points to the other line. "dist" first asks if the    */
684   /* segments intersect. If they do, it returns zero.          */
685   /* Otherwise the function calculates the shortest distance   */
686   /* from the first segment's end points to the second segment */
687   /* and vice versa and returns the shortest distance.         */
688   /* ------------------------------------------------------- */
689   double dist2(double a0, double a1, double b0, double b1,
690                double c0, double c1, double d0, double d1) {
691
692   double test;
693   double best = sqrt((a0-c0)*(a0-c0)+(a1-c1)*(a1-c1));
694   double val;
695
696   /* The only possibility that these two segments intersect: */
697   if ((fabs(a0-d0)<=EPS && fabs(a1-d1)<=EPS) ||
698       (fabs(b0-c0)<=EPS && fabs(b1-c1)<=EPS))
699       best=0;
700   /* If segements do not intersect: */
701   else{
702       /* Calculate the distance from A to segment [C,D]: */
703       test = ptoseg(a0, a1, c0, c1, d0, d1);
704       if ((test-best)<=EPS){
705           best = test;
706       }
707       /* Calculate the distance from B to segment [C,D]: */
708       test = ptoseg(b0, b1, c0, c1, d0, d1);
709       if ((test-best)<=EPS){
710           best = test;
711       }
712       /* Calculate the distance from C to segment [A,B]: */
713       test = ptoseg(c0, c1, a0, a1, b0, b1);
714       if ((test-best)<=EPS){
715           best = test;
716       }
717       /* Calculate the distance from D to segment [A,B]: */
718       test = ptoseg(d0, d1, a0, a1, b0, b1);
719       if ((test-best)<=EPS){
720           best = test;
721       }
722   }
723
724   return best;
725   }
726
727
728   /* ---------------------- ptoseg ---------------------- */
729   /* This function treats the segment as a parameterized */
730   /* vector where the parameter t varies from 0 to 1.    */
731   /* It finds the value of t that minimizes the distance */
732   /* from the point to the line. If t is between 0.0 and */
733   /* 1.0, then the closest point lies on the segment,    */
734   /* otherwise the closest point is one of the segment's */
```

```
735 /* end points. The program finds this closest point     */
736 /* and calculates the distance between it and the       */
737 /* target point.                                        */
738 /* ---------------------------------------------------- */
739 double ptoseg(double p0, double p1, double a0, double a1,double b0, double b1){
740
741 double t, tmp1, tmp2;
742 double val;
743
744 tmp1 = b0-a0;
745 tmp2 = b1-a1;
746
747 if (tmp1<=EPS && tmp2<=EPS){
748     tmp1 = p0-a0;
749     tmp2 = p1-a1;
750 }
751 else{
752     t = ((p0-a0)*(b0-a0)+(p1-a1)*(b1-a1))/((b0-a0)*(b0-a0)+(b1-a1)*(b1-a1));
753
754     if (t<=EPS){
755         tmp1 = p0-a0;
756         tmp2 = p1-a1;
757     }
758     else if ((1-t)<=EPS){
759         tmp1 = p0-b0;
760         tmp2 = p1-b1;
761     }
762     else{
763         tmp1 = p0 - a0 - t*tmp1;
764         tmp2 = p1 - a1 - t*tmp2;
765     }
766 }
767 val  = sqrt(tmp1*tmp1 + tmp2*tmp2);
768 return val;
769 }
```

## A.3 Einfach- und Doppelschichtpotential

### A.3.1 Einfachschichtpotential

```
1  /* ********************************************** *
2   * Compute Galerkin-Matrix for V and P0xP0        *
3   * Vjk = < Vchi_j , chi_k > with char. functions   *
4   * ********************************************** *
5   * Vphi = -1/(2pi) \int_Gamma log|x-y| phi(y) ds_y *
6   * ********************************************** */
7
8  #include <math.h>
9  #include "mex.h"
10
11 #define EPS 1e-12
12 #define PI 3.141592653589793116
13
14 void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[]);
15
16 double computeVij(double a0, double a1, double b0, double b1,
17                   double c0, double c1, double d0, double d1,
18                   double eta);
```

```
19 double computeVij_analytic(double a0, double a1, double b0, double b1,
20                            double c0, double c1, double d0, double d1);
21 double computeVij_semianalytic(double a0, double a1, double b0, double b1,
22                                double c0, double c1, double d0, double d1);
23 double slpWrapper(int k, double u[2],double v[2]);
24 double slp(int k, double a, double b, double c);
25
26 double ptoseg(double p0, double p1, double a0, double a1, double b0, double b1);
27 double dist2(double a0, double a1, double b0, double b1,
28              double c0, double c1, double d0, double d1);
29
30
31
32 /* ----------------- MEX interface ----------------- */
33
34
35 void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[]) {
36
37   const char* functionName = mexFunctionName();
38   char errorMessage[255];
39   int nC, nE;
40   int i, j;
41   double* V;
42   double* elements;
43   double* coordinates;
44   double* ptr;
45   double eta;
46   double a0,a1, b0,b1, c0,c1, d0,d1;
47   int aidx, bidx, cidx, didx;
48
49   if (nlhs != 1) {
50     sprintf(errorMessage,"Use either V = %s(coordinates,elements)\n        or V = %s
         (coordinates,elements,eta).",
51             functionName,functionName);
52     mexErrMsgTxt(errorMessage);
53   }
54
55   if ((nrhs != 2) && (nrhs != 3) ) {
56     sprintf(errorMessage,"Use either V = %s(coordinates,elements)\n        or V = %s
         (coordinates,elements,eta).",
57             functionName,functionName);
58     mexErrMsgTxt(errorMessage);
59   }
60
61   /* Read input data */
62   coordinates = mxGetPr(prhs[0]);
63   nC = mxGetM(prhs[0]);          /* number of nodes */
64
65   elements = mxGetPr(prhs[1]);
66   nE = mxGetM(prhs[1]);          /* number of elements */
67
68   if (nrhs == 3) {
69     ptr = mxGetPr(prhs[2]);
70     eta = *ptr;
71   }
72   else {
73     eta = 0;
74   }
75
76   /* Allocate output data */
77   plhs[0] = mxCreateDoubleMatrix(nE,nE,mxREAL);
```

```
78    V = mxGetPr(plhs[0]);
79
80    /* Fill matrix V and use symmetry to reduce building time */
81    for (i=0; i<nE; ++i) {
82
83      aidx = (int) elements[i]-1;         /* 1st node of element Ti = [a,b] */
84      a0 = coordinates[aidx];
85      a1 = coordinates[aidx+nC];
86
87      bidx = (int) elements[i+nE]-1;     /* 2nd node of element Ti = [a,b] */
88      b0 = coordinates[bidx];
89      b1 = coordinates[bidx+nC];
90
91      for (j=i; j<nE; ++j) {
92
93        cidx = (int) elements[j]-1;       /* 1st node of element Tj = [c,d] */
94        c0 = coordinates[cidx];
95        c1 = coordinates[cidx+nC];
96
97        didx = (int) elements[j+nE]-1;  /* 2nd node of element Tj = [c,d] */
98        d0 = coordinates[didx];
99        d1 = coordinates[didx+nC];
100
101       V[i + j*nE] = computeVij(a0,a1,b0,b1, c0,c1,d0,d1,eta);
102       V[j + i*nE] = V[i + j*nE];
103
104     }
105   }
106 }
107
108
109 /* ----------------- WRAPPER: Galerkin-Element SLP ----------------- */
110
111
112 double computeVij(double a0, double a1, double b0, double b1,
113                   double c0, double c1, double d0, double d1, double eta) {
114   /*
115    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
116    * OUTPUT: Galerkin integral -1/(2pi) \int_{Tj} \int_{Ti} log|x-y| ds_y ds_x
117    */
118
119   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
120   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
121   double tmp;
122
123   /* For stability reasons, we guarantee   hj <= hi   to ensure that *
124    * outer integration is over smaller domain. This is done by       *
125    * swapping Tj and Ti if necessary.                                */
126
127   if (hj > hi) {
128     tmp = a0; a0 = c0; c0 = tmp;   /* swap a and c */
129     tmp = a1; a1 = c1; c1 = tmp;
130     tmp = b0; b0 = d0; d0 = tmp;   /* swap b and d */
131     tmp = b1; b1 = d1; d1 = tmp;
132     tmp = hi; hi = hj; hj = tmp;   /* ensure that hj <= hi */
133   }
134
135   if ( eta == 0) { /* compute all matrix entries analytically */
136
137     return computeVij_analytic(a0,a1, b0,b1, c0,c1, d0,d1);
138
```

```
139   }
140   else { /* compute admissible matrix entries semi-analytically */
141
142     if ( dist2(a0,a1,b0,b1,c0,c1,d0,d1) > eta*sqrt(hj) ) {
143       return computeVij_semianalytic(a0,a1, b0,b1, c0,c1, d0,d1);
144     }
145     else {
146       return computeVij_analytic(a0,a1, b0,b1, c0,c1, d0,d1);
147     }
148   }
149 }
150
151
152 /* ------------------ Galerkin-Element SLP (analytic computation) ------------------
      */
153
154
155 double computeVij_analytic(double a0, double a1, double b0, double b1,
156                            double c0, double c1, double d0, double d1) {
157
158   /*
159    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
160    * OUTPUT: Galerkin integral -1/(2pi) \int_{Tj} \int_{Ti} log|x-y| ds_y ds_x
161    */
162
163   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
164   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
165   double tmp, val, det;
166   double x[2], y[2], z[2];
167   double zxp[2], zxm[2], zyp[2], zym[2];
168   double lambda, mu;
169
170   x[0] = 0.5*(b0 - a0);             /* x = (b-a)/2 */
171   x[1] = 0.5*(b1 - a1);
172   y[0] = 0.5*(c0 - d0);             /* y = (c-d)/2 */
173   y[1] = 0.5*(c1 - d1);
174   z[0] = 0.5*(a0 + b0 - c0 - d0);   /* z = (a+b-c-d)/2 */
175   z[1] = 0.5*(a1 + b1 - c1 - d1);
176
177   zxp[0] = z[0] + x[0];             /* zxp = z+x = (2b-c-d)/2 */
178   zxp[1] = z[1] + x[1];
179   zxm[0] = z[0] - x[0];             /* zxm = z-x = (2a-c-d)/2 */
180   zxm[1] = z[1] - x[1];
181   zyp[0] = z[0] + y[0];             /* zyp = z+y = (a+b-2d)/2 */
182   zyp[1] = z[1] + y[1];
183   zym[0] = z[0] - y[0];             /* zym = z-y = (a+b-2c)/2 */
184   zym[1] = z[1] - y[1];
185
186   /* There hold different recursion formulae if Ti and Tj *
187    * are parallel (det = 0) or not                        */
188
189   det = x[0]*y[1] - x[1]*y[0];
190
191   if ( fabs(det) <= EPS*sqrt(hi*hj) ) { /* case that Ti and Tj are parallel */
192
193     if ( fabs(x[0]) < fabs(x[1]) )
194       lambda = y[1] / x[1];
195     else
196       lambda = y[0] / x[0];
197
198     val = 0.5*( lambda * ( slpWrapper(1, y, zxm) - slpWrapper(1, y, zxp) )
```

```
199         + slpWrapper(0, x, zyp) + slpWrapper(0, x, zym) );
200   }
201
202   else { /* case that x and y are linearly independent */
203
204     lambda = (z[0]*y[1] - z[1]*y[0]) /det;
205     mu = (x[0]*z[1] - x[1]*z[0]) /det;
206
207     val = 0.25 * ( -8 + (lambda+1)*slpWrapper(0, y, zxp)
208         - (lambda-1)*slpWrapper(0, y, zxm) + (mu+1)*slpWrapper(0, x, zyp)
209         - (mu-1)*slpWrapper(0, x, zym) );
210   }
211
212   return -0.125*sqrt(hi*hj)*val /PI; /* = - 1/(2*PI) * |Ti|/2 * |Tj|/2 * val */
213 }
214
215
216 /* -------------- Galerkin-Element SLP (semianalytic computation) -------------- */
217
218
219 double computeVij_semianalytic(double a0, double a1, double b0, double b1,
220                                double c0, double c1, double d0, double d1) {
221
222   /*
223    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
224    * OUTPUT: approximate Galerkin integral -1/(2pi) \int_{Tj} \int_{Ti} log|x-y|
         ds_y ds_x,
225    *         where outer integration is performed by Gaussian quadrature
226    */
227
228   int k;
229   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
230   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
231   double u[2], v[2];
232   double val = 0;
233   double sx0 = 0;
234   double sx1 = 0;
235
236   /* 16-point Gaussian quadrature on [-1,1] */
237   const order = 16;
238   const double wht[] = { 0.027152459411754, 0.062253523938648, 0.095158511682493,
239                          0.124628971255534, 0.149595988816577, 0.169156519395003,
240                          0.182603415044924, 0.189450610455069, 0.189450610455069,
241                          0.182603415044924, 0.169156519395003, 0.149595988816577,
242                          0.124628971255534, 0.095158511682492, 0.062253523938648,
243                          0.027152459411753 };
244   const double pt[] = { -0.989400934991650,-0.944575023073233,-0.865631202387832,
245                         -0.755404408355003,-0.617876244402644,-0.458016777657227,
246                         -0.281603550779259,-0.095012509837637, 0.095012509837637,
247                          0.281603550779259, 0.458016777657228, 0.617876244402644,
248                          0.755404408355003, 0.865631202387832, 0.944575023073232,
249                          0.989400934991650 };
250
251   u[0] = 0.5*(a0-b0);
252   u[1] = 0.5*(a1-b1);
253
254   for (k=0; k<order; ++k){
255       /* transformation of quadrature nodes from [-1,1] to [a,b] */
256       sx0 = ((1-pkt[k])*c0+(1+pkt[k])*d0)*0.5;
257       sx1 = ((1-pkt[k])*c1+(1+pkt[k])*d1)*0.5;
258
```

```
259         v[0] = sx0 - 0.5*(a0+b0);
260         v[1] = sx1 - 0.5*(a1+b1);
261
262         /* inner product wht*func(sx) */
263         val += wht[k]*slpWrapper(0, u, v);
264     }
265
266     return -0.0625*sqrt(hi*hj)*val /PI;
267 }
268
269
270 /* ----------------- SLP ----------------- */
271
272
273 double slpWrapper(int k, double u[2],double v[2]) {
274     /*
275      * INPUT: Vectors u,v \in \R^2, integer k
276      * OUTPUT: value of SLP-type integral
277      *          \int_{-1}^{+1} s^k \log |s*u+v|^2 ds
278      */
279
280     double a = u[0]*u[0] + u[1]*u[1];          /* a = <u,u> */
281     double b = 2 * ( u[0]*v[0] + u[1]*v[1] ); /* b = 2 <u,v> */
282     double c = v[0]*v[0] + v[1]*v[1];          /* c = <v,v> */
283     return slp(k,a,b,c);
284 }
285
286
287 double slp(int k, double a, double b, double c) {
288     /*
289      * INPUT: scalars a,c>0 and b \in \R, integer k
290      * OUPUT: value of SLP-type integral
291      *         \int_{-1}^{+1} s^k \log |a*s^2+b*s+c| ds
292      */
293
294     double val;
295     double tmp;
296     double D;
297
298     /* Ensure that discriminant is either positive or zero */
299     tmp = 4*a*c - b*b; /* Note that by theory tmp >= 0 */
300     if (tmp > EPS*4*a*c)
301       D = sqrt(tmp);
302     else
303       D = 0;
304
305     /* The case k=0 */
306     if (D == 0) {
307       tmp = b + 2*a;
308       if (fabs(tmp) > EPS*a)
309         val = tmp * log( 0.25*tmp*tmp /a );
310       else
311         val = 0;
312       tmp = b - 2*a;
313       if (fabs(tmp) > EPS*a)
314         val -= tmp * log( 0.25*tmp*tmp /a );
315       val = 0.5*val /a - 4;
316     }
317     else { /* case D > 0 */
318       tmp = c - a;
319       if (fabs(tmp) < EPS*c)
```

```
320       val = 0.5*PI;
321     else if (a < c)
322       val = atan( D /tmp );
323     else
324       val = atan( D /tmp ) + PI;
325
326     val = ( 0.5*( (b+2*a) * log(a+b+c) - (b-2*a) * log(a-b+c) ) + D*val )/a - 4;
327   }
328
329   /* The case k=1 */
330   if (k==1) {
331     val = -b*(2+val);
332
333
334     tmp = a+b+c;
335     if (fabs(tmp) > EPS*a)
336       val += tmp * log(tmp);
337
338     tmp = a-b+c;
339     if (fabs(tmp) > EPS*a)
340       val -= tmp * log(tmp);
341
342     val /= (2*a);
343   }
344
345   return val;
346 }
347
348
349 /* ---------------------- Dist(Ti,Tj) ---------------------- */
350 /* The closest distance between two segments is either zero  */
351 /* if they intersect or the distance from one of the lines'  */
352 /* end points to the other line. "dist" first asks if the    */
353 /* segments intersect. If they do, it returns zero.          */
354 /* Otherwise the function calculates the shortest distance   */
355 /* from the first segment's end points to the second segment */
356 /* and vice versa and returns the shortest distance.         */
357 /* ------------------------------------------------------- */
358 double dist2(double a0, double a1, double b0, double b1,
359              double c0, double c1, double d0, double d1) {
360
361 double test;
362 double best = sqrt((a0-c0)*(a0-c0)+(a1-c1)*(a1-c1));
363 double val;
364
365 /* The only possibility that these two segments intersect: */
366 if ((fabs(a0-d0)<=EPS && fabs(a1-d1)<=EPS) ||
367     (fabs(b0-c0)<=EPS && fabs(b1-c1)<=EPS))
368     best=0;
369 /* If segements do not intersect: */
370 else{
371     /* Calculate the distance from A to segment [C,D]: */
372     test = ptoseg(a0, a1, c0, c1, d0, d1);
373     if ((test-best)<=EPS){
374         best = test;
375     }
376     /* Calculate the distance from B to segment [C,D]: */
377     test = ptoseg(b0, b1, c0, c1, d0, d1);
378     if ((test-best)<=EPS){
379         best = test;
380     }
```

```
381     /* Calculate the distance from C to segment [A,B]: */
382     test = ptoseg(c0, c1, a0, a1, b0, b1);
383     if ((test-best)<=EPS){
384         best = test;
385     }
386     /* Calculate the distance from D to segment [A,B]: */
387     test = ptoseg(d0, d1, a0, a1, b0, b1);
388     if ((test-best)<=EPS){
389         best = test;
390     }
391 }
392 
393 return best;
394 }
395 
396 
397 /* --------------------- ptoseg ---------------------- */
398 /* This function treats the segment as a parameterized */
399 /* vector where the parameter t varies from 0 to 1.    */
400 /* It finds the value of t that minimizes the distance */
401 /* from the point to the line. If t is between 0.0 and */
402 /* 1.0, then the closest point lies on the segment,    */
403 /* otherwise the closest point is one of the segment's */
404 /* end points. The program finds this closest point    */
405 /* and calculates the distance between it and the      */
406 /* target point.                                       */
407 /* ----------------------------------------------------- */
408 double ptoseg(double p0, double p1, double a0, double a1,double b0, double b1){
409 
410 double t, tmp1, tmp2;
411 double val;
412 
413 tmp1 = b0-a0;
414 tmp2 = b1-a1;
415 
416 if (tmp1<=EPS && tmp2<=EPS){
417     tmp1 = p0-a0;
418     tmp2 = p1-a1;
419 }
420 else{
421     t = ((p0-a0)*(b0-a0)+(p1-a1)*(b1-a1))/((b0-a0)*(b0-a0)+(b1-a1)*(b1-a1));
422 
423     if (t<=EPS){
424         tmp1 = p0-a0;
425         tmp2 = p1-a1;
426     }
427     else if ((1-t)<=EPS){
428         tmp1 = p0-b0;
429         tmp2 = p1-b1;
430     }
431     else{
432         tmp1 = p0 - a0 - t*tmp1;
433         tmp2 = p1 - a1 - t*tmp2;
434     }
435 }
436 val  = sqrt(tmp1*tmp1 + tmp2*tmp2);
437 return val;
438 }
```

### A.3.2 Doppelschichtpotential

```
/* ************************************************** *
 * Compute Galerkin-Matrix for K and S1xP0            *
 * K_kj = < K phi_j , chi_k >                         *
 * with chi characteristic and phi affine functions   *
 * ************************************************** *
 * Kg = -1/(2pi) \int_Gamma <y-x,n(y)>/|x-y|^2 g(y) ds_y *
 * ************************************************** */

#include <math.h>
#include "mex.h"

#define EPS 1e-12
#define PI 3.141592653589793116

void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[]);

void computeKij(double* val0, double* val1,
                double a0, double a1, double b0, double b1,
                double c0, double c1, double d0, double d1, double eta);
void computeKij_analytic(double* I0, double* I1,
                         double a0, double a1, double b0, double b1,
                         double c0, double c1, double d0, double d1);
void computeKijT_analytic(double* I0, double* I1,
                          double a0, double a1, double b0, double b1,
                          double c0, double c1, double d0, double d1);
void computeKij_semianalytic(double* I0, double* I1,
                             double a0, double a1, double b0, double b1,
                             double c0, double c1, double d0, double d1);
void computeKijT_semianalytic(double* I0, double* I1,
                              double a0, double a1, double b0, double b1,
                              double c0, double c1, double d0, double d1);

double dlpWrapper(int k, double u[2],double v[2]);
double dlp(int k, double a, double b, double c);

double ptoseg(double p0, double p1, double a0, double a1, double b0, double b1);
double dist2(double a0, double a1, double b0, double b1,
             double c0, double c1, double d0, double d1);



/* ----------------- MEX interface ----------------- */

void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[]) {

  const char* functionName = mexFunctionName();
  char errorMessage[255];
  int nC, nE;
  int i, j;
  double* K;
  double* elements;
  double* coordinates;
  double* ptr;
  double eta;
  double a0,a1, b0,b1, c0,c1, d0,d1;
  int aidx, bidx, cidx, didx;
  double I0, I1;
  double linetest1, linetest2;

```

```
60    if (nlhs != 1) {
61      sprintf(errorMessage,"Use either K = %s(coordinates,elements)\n        or K = %s
          (coordinates,elements,eta).",
62            functionName,functionName);
63      mexErrMsgTxt(errorMessage);
64    }
65
66    if ((nrhs != 2) && (nrhs != 3) ) {
67      sprintf(errorMessage,"Use either K = %s(coordinates,elements)\n        or K = %s
          (coordinates,elements,eta).",
68            functionName,functionName);
69      mexErrMsgTxt(errorMessage);
70    }
71
72
73    /* Read input data */
74    coordinates = mxGetPr(prhs[0]);
75    nC = mxGetM(prhs[0]);          /* number of nodes */
76
77    elements = mxGetPr(prhs[1]);
78    nE = mxGetM(prhs[1]);          /* number of elements */
79
80    if (nrhs == 3) {
81      ptr = mxGetPr(prhs[2]);
82      eta = *ptr;
83    }
84    else {
85      eta = 0.;
86    }
87
88    /* Allocate output data and initialize with zeros*/
89    plhs[0] = mxCreateDoubleMatrix(nE,nC,mxREAL);
90    K = mxGetPr(plhs[0]);
91
92    /* Fill matrix K */
93    for (j=0; j<nE; ++j) {
94
95      cidx = (int) elements[j]-1;        /* 1st node of element Tj = [c,d] */
96      c0 = coordinates[cidx];
97      c1 = coordinates[cidx+nC];
98
99      didx = (int) elements[j+nE]-1;     /* 2nd node of element Tj = [c,d] */
100     d0 = coordinates[didx];
101     d1 = coordinates[didx+nC];
102
103     for (i=0; i<nE; ++i) {
104
105       aidx = (int) elements[i]-1;      /* 1st node of element Ti = [a,b] */
106       a0 = coordinates[aidx];
107       a1 = coordinates[aidx+nC];
108
109       bidx = (int) elements[i+nE]-1;   /* 2nd node of element Ti = [a,b] */
110       b0 = coordinates[bidx];
111       b1 = coordinates[bidx+nC];
112
113       /* if two intervals are in line the scalar-product with */
114       /* the normal vector is zero                            */
115       /*  --> double-layer-potential is zero!                 */
116       /* this is tested beneath and in this case nothing      */
117       /* should be calculated!                                */
118       linetest1 = fabs( (a0+b0-2*c0)*(b1-a1)-(a1+b1-2*c1)*(b0-a0) );
```

```
119       linetest2 = fabs( (a0+b0-2*d0)*(b1-a1)-(a1+b1-2*d1)*(b0-a0) );
120
121       if ( linetest1 > EPS || linetest2 > EPS ) {
122         computeKij(&I0,&I1, a0,a1,b0,b1, c0,c1,d0,d1,eta);
123         K[j+aidx*nE] += I0 - I1;
124         K[j+bidx*nE] += I0 + I1;
125       }
126     }
127   }
128
129 }
130
131
132 /* ----------------- WRAPPER: Double Integrals for DLP ----------------- */
133
134 void computeKij(double* I0, double* I1,
135                 double a0, double a1, double b0, double b1,
136                 double c0, double c1, double d0, double d1,
137                 double eta) {
138   /*
139    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
140    * OUTPUT: Galerkin integral
141    *         -1/(2pi) \int_{Tj} \int_{Ti} 1/2 <y-x,n(y)>/|x-y|^2 \phi(y) ds_y ds_x
142    *         \gamma(\phi(t))\in[-1,1]: \gamma(\phi(t)) = 1/2 + 1/2t
143    *         resp. \gamma(\phi(t)) = 1/2 - 1/2t
144    *         thus \phi(\gamma) is the hat function on [-1,1],
145    *         resp. \phi is the hat function on T_i
146    */
147
148   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
149   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
150   int swap;
151
152   /* For stability reasons, we guarantee   hj <= hi   to ensure that *
153    * outer integration is over smaller domain. This is done by       *
154    * swapping Tj and Ti if necessary.                                */
155   if ( (hj-hi)/hj > EPS )
156     swap = 1;
157   else
158     swap = 0;
159
160   if (eta == 0) { /* compute all matrix entries analytically */
161
162     if (swap == 0)
163       computeKij_analytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
164     else
165       computeKijT_analytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
166   }
167   else { /* compute admissible matrix entries semi-analytically */
168
169     if ( dist2(a0,a1,b0,b1,c0,c1,d0,d1) > eta*sqrt(hj) ) {
170       if (swap == 0)
171         computeKij_semianalytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
172       else
173         computeKijT_semianalytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
174     }
175     else {
176       if (swap == 0)
177         computeKij_analytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
178       else
179         computeKijT_analytic(I0,I1, a0,a1, b0,b1, c0,c1, d0,d1);
```

```
180       }
181     }
182   }
183
184
185   /* -------------- Double Integrals for DLP: analytic, no swap -------------- */
186
187   void computeKij_analytic(double* I0, double* I1,
188                            double a0, double a1, double b0, double b1,
189                            double c0, double c1, double d0, double d1) {
190     /*
191      * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
192      * OUTPUT: Galerkin integral
193      *         -1/(2pi) \int_{Tj} \int_{Ti} 1/2 <y-x,n(y)>/|x-y|^2 \phi(y) ds_y ds_x
194      *         \gamma(\phi(t))\in[-1,1]: \gamma(\phi(t)) = 1/2 + 1/2t
195      *         resp. \gamma(\phi(t)) = 1/2 - 1/2t
196      *         thus \phi(\gamma) is the hat function on [-1,1],
197      *         resp. \phi is the hat function on T_i
198      */
199
200     double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
201     double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
202     double det;
203     double n[2];
204     double x[2], y[2], z[2];
205     double zxp[2], zxm[2], zyp[2], zym[2];
206     double yn, zn, zypn, zymn;
207     double lambda, mu;
208
209     /* n = n_y normale vector */
210     n[0] = ( b1-a1 ) / sqrt(hi);
211     n[1] = ( a0-b0 ) / sqrt(hi);
212
213     x[0] = b0 - a0;               /* x = b-a */
214     x[1] = b1 - a1;
215     y[0] = c0 - d0;               /* y = c-d */
216     y[1] = c1 - d1;
217     z[0] = a0 + b0 - c0 - d0;     /* z = a+b-c-d */
218     z[1] = a1 + b1 - c1 - d1;
219
220     zxp[0] = z[0] + x[0];           /* zxp = z+x = 2b-c-d */
221     zxp[1] = z[1] + x[1];
222     zxm[0] = z[0] - x[0];           /* zxm = z-x = 2a-c-d */
223     zxm[1] = z[1] - x[1];
224     zyp[0] = z[0] + y[0];           /* zyp = z+y = a+b-2d */
225     zyp[1] = z[1] + y[1];
226     zym[0] = z[0] - y[0];           /* zym = z-y = a+b-2c */
227     zym[1] = z[1] - y[1];
228
229     yn = y[0]*n[0]+y[1]*n[1];       /* yn = <y,n> */
230     zn = z[0]*n[0]+z[1]*n[1];       /* zn = <z,n> */
231     zypn = zyp[0]*n[0]+zyp[1]*n[1]; /* zypn = <z+y,n> = <a+b-2d,n> */
232     zymn = zym[0]*n[0]+zym[1]*n[1]; /* zymn = <z-y,n> = <a+b-2c,n> */
233
234     /* There hold different recursion formulae if Ti and Tj *
235      * are parallel (det = 0) or not                        */
236     det = x[0]*y[1] - x[1]*y[0];
237
238     /* case that x and y are linearly dependent, i.e., Ti and Tj are parallel */
239     if ( fabs(det) <= EPS*sqrt(hi*hj) ) {
240
```

```
241     if ( fabs(y[0]) < fabs(y[1]) )
242       lambda = x[1] / y[1];
243     else
244       lambda = x[0] / y[0];
245 
246     *I0 = zn * ( dlpWrapper(0, y, zxp) + dlpWrapper(0, y, zxm)
247         - lambda * dlpWrapper(1, x, zyp) + lambda * dlpWrapper(1, x, zym) );
248 
249     *I1 = 0.5 * zn * ( dlpWrapper(0, y, zxp) - dlpWrapper(0, y, zxm)
250         - lambda * dlpWrapper(2, x, zyp) + lambda * dlpWrapper(2, x, zym) );
251   }
252 
253   else { /* case that x and y are linearly independent */
254 
255     /* if intervals are in touch recursion formulae reduce */
256     if (b0==c0 && b1==c1) {
257 
258       *I0 = 2 * ( zypn * dlpWrapper(0, x, zyp) + yn * dlpWrapper(1, y, zxm)
259           + zn * dlpWrapper(0, y, zxm) );
260 
261       *I1 = 0.5 * ( 2 * zypn * dlpWrapper(1, x, zyp)
262           - 2 * yn * dlpWrapper(1, y, zxm) - 2 * zn * dlpWrapper(0, y, zxm) + *I0 );
263     }
264 
265     else if (a0==d0 && a1==d1) {
266 
267       *I0 = 2 * ( zymn * dlpWrapper(0, x, zym) + yn * dlpWrapper(1, y, zxp)
268           + zn * dlpWrapper(0, y, zxp) );
269 
270       *I1 = 0.5 * ( 2 * zymn * dlpWrapper(1, x ,zym)
271           + 2 * yn * dlpWrapper(1, y, zxp) + 2 * zn * dlpWrapper(0, y, zxp) - *I0 );
272     }
273 
274     else {
275 
276       lambda = (z[0]*y[1] - z[1]*y[0]) /det;
277       mu = (x[0]*z[1] - x[1]*z[0]) /det;
278 
279       *I0 = (mu+1) * zypn * dlpWrapper(0, x, zyp)
280           - (mu-1) * zymn * dlpWrapper(0, x ,zym)
281           + (lambda+1) * yn * dlpWrapper(1, y, zxp)
282           + (lambda+1) * zn * dlpWrapper(0, y, zxp)
283           - (lambda-1) * yn * dlpWrapper(1, y, zxm)
284           - (lambda-1) * zn * dlpWrapper(0, y, zxm);
285 
286       *I1 = 0.5 * ( (mu+1) * zypn * dlpWrapper(1, x, zyp)
287           - (mu-1) * zymn * dlpWrapper(1, x ,zym)
288           + (lambda+1) * yn * dlpWrapper(1, y, zxp)
289           + (lambda+1) * zn * dlpWrapper(0, y, zxp)
290           + (lambda-1) * yn * dlpWrapper(1, y, zxm)
291           + (lambda-1) * zn * dlpWrapper(0, y, zxm)
292           - lambda * *I0 );
293     }
294   }
295 
296   *I0 *= -0.125 * sqrt(hi*hj) / PI; /* = - 1/(2*PI) * |Ti|/2 * |Tj|/2 * I0 */
297   *I1 *= -0.125 * sqrt(hi*hj) / PI; /* = - 1/(2*PI) * |Ti|/2 * |Tj|/2 * I1 */
298 }
299 
300 
301  /* ----------------- Double Integrals for DLP: analytic, swap ----------------- */
```

```
302
303 void computeKijT_analytic(double* I0, double* I1,
304                           double a0, double a1, double b0, double b1,
305                           double c0, double c1, double d0, double d1) {
306   /*
307    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
308    * OUTPUT: Galerkin integral
309    *         -1/(2pi) \int_{Tj} \int_{Ti} 1/2 <y-x,n(y)>/|x-y|^2 \phi(y) ds_y ds_x
310    *         \gamma(\phi(t))\in[-1,1]: \gamma(\phi(t)) = 1/2 + 1/2t
311    *         resp. \gamma(\phi(t)) = 1/2 - 1/2t
312    *         thus \phi(\gamma) is the hat function on [-1,1],
313    *         resp. \phi is the hat function on T_i
314    */
315
316   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
317   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
318   double det;
319   double n[2];
320   double x[2], y[2], z[2];
321   double zxp[2], zxm[2], zyp[2], zym[2];
322   double xn, zn, zxpn, zxmn;
323   double lambda, mu;
324
325   /* n = n_y normale vector */
326   n[0] = ( b1-a1 ) / sqrt(hi);
327   n[1] = ( a0-b0 ) / sqrt(hi);
328
329   x[0] = c0 - d0;               /* x = c-d */
330   x[1] = c1 - d1;
331   y[0] = b0 - a0;               /* y = b-a */
332   y[1] = b1 - a1;
333   z[0] = a0 + b0 - c0 - d0;     /* z = a+b-c-d */
334   z[1] = a1 + b1 - c1 - d1;
335
336   zxp[0] = z[0] + x[0];           /* zxp = z+x = a+b-2d */
337   zxp[1] = z[1] + x[1];
338   zxm[0] = z[0] - x[0];           /* zxm = z-x = a+b-2c */
339   zxm[1] = z[1] - x[1];
340   zyp[0] = z[0] + y[0];           /* zyp = z+y = 2b-c-d */
341   zyp[1] = z[1] + y[1];
342   zym[0] = z[0] - y[0];           /* zym = z-y = 2a-c-d */
343   zym[1] = z[1] - y[1];
344
345   xn = x[0]*n[0]+x[1]*n[1];       /* yn = <y,n> */
346   zn = z[0]*n[0]+z[1]*n[1];       /* zn = <z,n> */
347   zxpn = zxp[0]*n[0]+zxp[1]*n[1]; /* zxpn = <z+x,n> = <a+b,n> */
348   zxmn = zxm[0]*n[0]+zxm[1]*n[1]; /* zxmn = <z-x,n> = <a+b-2c,n> */
349
350
351   /* There hold different recursion formulae if Ti and Tj *
352    * are parallel (det = 0) or not                        */
353   det = x[0]*y[1] - x[1]*y[0];
354
355   /* case that x and y are linearly dependent, i.e., Ti and Tj are parallel */
356   if ( fabs(det) <= EPS*sqrt(hi*hj) ) {
357
358     if ( fabs(y[0]) < fabs(y[1]) )
359       lambda = x[1] / y[1];
360     else
361       lambda = x[0] / y[0];
362
```

```
363     *I0 = zn * ( dlpWrapper(0, y, zxp) + dlpWrapper(0, y, zxm)
364         - lambda * dlpWrapper(1, x, zyp) + lambda * dlpWrapper(1, x, zym) );
365
366     *I1 = zn * ( dlpWrapper(1, y, zxp) + dlpWrapper(1, y, zxm)
367         - lambda * dlpWrapper(1, x, zyp) - lambda * dlpWrapper(1, x, zym)
368         + 0.5* lambda * ( dlpWrapper(0, y, zxp) - dlpWrapper(0, y, zxm)
369         - lambda * dlpWrapper(2, x, zyp) + lambda * dlpWrapper(2, x, zym) ) );
370   }
371
372   else { /* case that x and y are linearly independent */
373
374     /* if intervals are in touch recursion formulae reduce */
375     if (b0==c0 && b1==c1) {
376
377       *I0 = 2 * ( zxpn * dlpWrapper(0, y, zxp) + xn * dlpWrapper(1, x, zym)
378           + zn * dlpWrapper(0, x, zym) );
379
380       *I1 = 0.5 * ( 2 * zxpn * dlpWrapper(1, y, zxp)
381           - 2 * xn * dlpWrapper(1, x, zym) - 2 * zn * dlpWrapper(0, x, zym) + *I0 );
382     }
383
384     else if (a0==d0 && a1==d1) {
385
386       *I0 = 2 * ( zxmn * dlpWrapper(0, y, zxm) + xn * dlpWrapper(1, x, zyp)
387           + zn * dlpWrapper(0, x, zyp) );
388
389       *I1 = 0.5 * ( 2 * zxmn * dlpWrapper(1, y ,zxm)
390           + 2 * xn * dlpWrapper(1, x, zyp) + 2 * zn * dlpWrapper(0, x, zyp) - *I0 );
391     }
392
393     else {
394
395       mu = (z[0]*y[1] - z[1]*y[0]) /det;
396       lambda = (x[0]*z[1] - x[1]*z[0]) /det;
397
398       *I0 = (mu+1) * zxpn * dlpWrapper(0, y, zxp)
399           - (mu-1) * zxmn * dlpWrapper(0, y ,zxm)
400           + (lambda+1) * xn * dlpWrapper(1, x, zyp)
401           + (lambda+1) * zn * dlpWrapper(0, x, zyp)
402           - (lambda-1) * xn * dlpWrapper(1, x, zym)
403           - (lambda-1) * zn * dlpWrapper(0, x, zym);
404
405       *I1 = 0.5 * ( (mu+1) * zxpn * dlpWrapper(1, y, zxp)
406           - (mu-1) * zxmn * dlpWrapper(1, y ,zxm)
407           + (lambda+1) * xn * dlpWrapper(1, x, zyp)
408           + (lambda+1) * zn * dlpWrapper(0, x, zyp)
409           + (lambda-1) * xn * dlpWrapper(1, x, zym)
410           + (lambda-1) * zn * dlpWrapper(0, x, zym)
411           - lambda * *I0 );
412     }
413   }
414
415   *I0 *= -0.125 * sqrt(hi*hj) / PI;
416   *I1 *= -0.125 * sqrt(hi*hj) / PI;
417 }
418
419
420 /* -------------- Double Integrals for DLP: semianalytic, no swap -------------- */
421
422 void computeKij_semianalytic(double* I0, double* I1,
423                              double a0, double a1, double b0, double b1,
```

```
424                                 double c0, double c1, double d0, double d1) {
425   /*
426    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
427    * OUTPUT: Galerkin integral
428    *         -1/(2pi) \int_{Tj} \int_{Ti} 1/2 <y-x,n(y)>/|x-y|^2 \phi(y) ds_y ds_x
429    *         \gamma(\phi(t))\in[-1,1]: \gamma(\phi(t)) = 1/2 + 1/2t
430    *         resp. \gamma(\phi(t)) = 1/2 - 1/2t
431    *         thus \phi(\gamma) is the hat function on [-1,1],
432    *         resp. \phi is the hat function on T_i
433    */
434
435   int i;
436   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
437   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
438   double n[2];
439   double x[2], y[2], z[2];
440   double v[2];
441   double vn;
442   double I0tmp = 0., I1tmp = 0.;
443
444   /* 16-point Gaussian quadrature on [-1,1] */
445   const order = 16;
446   const double wht[] = { 0.027152459411754, 0.062253523938648, 0.095158511682493,
447                          0.124628971255534, 0.149595988816577, 0.169156519395003,
448                          0.182603415044924, 0.189450610455069, 0.189450610455069,
449                          0.182603415044924, 0.169156519395003, 0.149595988816577,
450                          0.124628971255534, 0.095158511682492, 0.062253523938648,
451                          0.027152459411753 };
452   const double pt[] = { -0.989400934991650,-0.944575023073233,-0.865631202387832,
453                         -0.755404408355003,-0.617876244402644,-0.458016777657227,
454                         -0.281603550779259,-0.095012509837637, 0.095012509837637,
455                          0.281603550779259, 0.458016777657228, 0.617876244402644,
456                          0.755404408355003, 0.865631202387832, 0.944575023073232,
457                          0.989400934991650 };
458
459   /* n = n_y normale vector */
460   n[0] = ( b1-a1 ) / sqrt(hi);
461   n[1] = ( a0-b0 ) / sqrt(hi);
462
463   x[0] = b0 - a0;               /* x = b-a */
464   x[1] = b1 - a1;
465   y[0] = c0 - d0;               /* y = c-d */
466   y[1] = c1 - d1;
467   z[0] = a0 + b0 - c0 - d0;     /* z = a+b-c-d */
468   z[1] = a1 + b1 - c1 - d1;
469
470
471   for (i=0; i<order; ++i) {
472
473     v[0] = pt[i] * y[0] + z[0];
474     v[1] = pt[i] * y[1] + z[1];
475
476     vn = v[0]*n[0]+v[1]*n[1];
477
478     I0tmp +=  wht[i] * vn * dlpWrapper(0, x, v);
479     I1tmp +=  wht[i] * vn * dlpWrapper(1, x, v);
480   }
481
482   *I0 = -0.125 * sqrt(hi*hj) * I0tmp / PI;
483   *I1 = -0.125 * sqrt(hi*hj) * I1tmp / PI;
484 }
```

```
485
486
487  /* -------------- Double Integrals for DLP: semianalytic, swap -------------- */
488
489 void computeKijT_semianalytic(double* I0, double* I1,
490                               double a0, double a1, double b0, double b1,
491                               double c0, double c1, double d0, double d1) {
492   /*
493    * INPUT:  elements Ti = [a,b], Tj = [c,d] with a,b,c,d \in \R^2
494    * OUTPUT: Galerkin integral
495    *         -1/(2pi) \int_{Tj} \int_{Ti} 1/2 <y-x,n(y)>/|x-y|^2 \phi(y) ds_y ds_x
496    *         \gamma(\phi(t))\in[-1,1]: \gamma(\phi(t)) = 1/2 + 1/2t
497    *         resp. \gamma(\phi(t)) = 1/2 - 1/2t
498    *         thus \phi(\gamma) is the hat function on [-1,1],
499    *         resp. \phi is the hat function on T_i
500    */
501
502   int i;
503   double hi = (b0-a0)*(b0-a0) + (b1-a1)*(b1-a1); /* hi = norm(b-a)^2 */
504   double hj = (d0-c0)*(d0-c0) + (d1-c1)*(d1-c1); /* hj = norm(d-c)^2 */
505   double n[2];
506   double x[2], y[2], z[2];
507   double v[2];
508   double xn, zn;
509   double I0tmp = 0., I1tmp = 0.;
510
511   /* 16-point Gaussian quadrature on [-1,1] */
512   const order = 16;
513   const double wht[] = { 0.027152459411754, 0.062253523938648, 0.095158511682493,
514                          0.124628971255534, 0.149595988816577, 0.169156519395003,
515                          0.182603415044924, 0.189450610455069, 0.189450610455069,
516                          0.182603415044924, 0.169156519395003, 0.149595988816577,
517                          0.124628971255534, 0.095158511682492, 0.062253523938648,
518                          0.027152459411753 };
519   const double pt[] = { -0.989400934991650,-0.944575023073233,-0.865631202387832,
520                         -0.755404408355003,-0.617876244402644,-0.458016777657227,
521                         -0.281603550779259,-0.095012509837637, 0.095012509837637,
522                          0.281603550779259, 0.458016777657228, 0.617876244402644,
523                          0.755404408355003, 0.865631202387832, 0.944575023073232,
524                          0.989400934991650 };
525
526   /* n = n_y normale vector */
527   n[0] = ( b1-a1 ) / sqrt(hi);
528   n[1] = ( a0-b0 ) / sqrt(hi);
529
530   x[0] = c0 - d0;                /* x = c-d */
531   x[1] = c1 - d1;
532   y[0] = b0 - a0;                /* y = b-a */
533   y[1] = b1 - a1;
534   z[0] = a0 + b0 - c0 - d0;      /* z = a+b-c-d */
535   z[1] = a1 + b1 - c1 - d1;
536
537   xn = x[0]*n[0]+x[1]*n[1];
538   zn = z[0]*n[0]+z[1]*n[1];
539
540
541   for (i=0; i<order; ++i) {
542
543     v[0] = pt[i] * y[0] + z[0];
544     v[1] = pt[i] * y[1] + z[1];
545
```

```
546       I0tmp +=  wht[i] * ( xn * dlpWrapper(1, x, v) + zn * dlpWrapper(0,x,v) );
547       I1tmp +=  wht[i] * pt[i] * ( xn * dlpWrapper(1, x, v) + zn * dlpWrapper(0,x,v));
548     }
549
550     *I0 = -0.125 * sqrt(hi*hj) * I0tmp / PI;
551     *I1 = -0.125 * sqrt(hi*hj) * I1tmp / PI;
552
553    }
554
555  /* ----------------- DLP ----------------- */
556
557  double dlpWrapper(int k, double u[2],double v[2]) {
558
559    /*
560     * INPUT: Vectors u,v,w \in \R^2, integer k
561     * OUTPUT: value of DLP-type integral
562     *          \int_{-1}^{+1} t^k 1 / |t*u+v|^2 dt
563     */
564
565    double a = u[0]*u[0] + u[1]*u[1];          /* a = <u,u> */
566    double b = 2 * ( u[0]*v[0] + u[1]*v[1] ); /* b = 2 <u,v> */
567    double c = v[0]*v[0] + v[1]*v[1];          /* c = <v,v> */
568    return dlp(k,a,b,c);
569  }
570
571
572  double dlp(int k, double a, double b, double c) {
573    /*
574     * INPUT: scalars a,c>0 and b \in \R, integer k
575     * OUPUT: value of DLP-type integral
576     *         \int_{-1}^{+1} t^k 1 / (a*t^2+b*t+c) dt
577     */
578
579
580    double val, val0, val1, val2;
581    double tmp;
582    double D;
583
584    /* Ensure that discriminant is either positive or zero */
585    tmp = 4*a*c - b*b; /* Note that by theory tmp >= 0 */
586
587    if (tmp > EPS * 4*a*c)
588      D = sqrt(tmp);
589    else
590      D = 0;
591
592    /* The case k=0 */
593    if (D == 0)
594    {
595      val0 = 2 / ( c - a );
596    }
597    else { /* also D > 0 */
598      tmp = c - a;
599      if (fabs(tmp) < EPS * fabs(c))
600        val0 = 0.5*PI;
601      else if (a < c)
602        val0 = atan( D /tmp );
603      else
604        val0 = atan( D /tmp ) + PI;
605      val0 *= 2/D;
606    }
```

```
607   val = val0;
608
609   /* The case k=1 */
610   if (k>=1) {
611     val1 = -b*val0;
612
613     tmp = a+b+c;
614     if (fabs(tmp) > EPS*fabs(a))
615       val1 += log(tmp);
616
617     tmp = a-b+c;
618     if (fabs(tmp) > EPS*fabs(a))
619       val1 -= log(tmp);
620
621     val1 /= (2*a);
622     val = val1;
623   }
624
625   /* The case k=2 */
626   if (k==2) {
627     val2 = ( 2 - b * val1 - c * val0 ) / a;
628     val = val2;
629   }
630
631   return val;
632 }
633
634 /* ---------------------- Dist(Ti,Tj) ---------------------- */
635 /* The closest distance between two segments is either zero  */
636 /* if they intersect or the distance from one of the lines'  */
637 /* end points to the other line. "dist" first asks if the    */
638 /* segments intersect. If they do, it returns zero.          */
639 /* Otherwise the function calculates the shortest distance   */
640 /* from the first segment's end points to the second segment */
641 /* and vice versa and returns the shortest distance.         */
642 /* ------------------------------------------------------------ */
643 double dist2(double a0, double a1, double b0, double b1,
644              double c0, double c1, double d0, double d1) {
645
646 double test;
647 double best = sqrt((a0-c0)*(a0-c0)+(a1-c1)*(a1-c1));
648 double val;
649
650 /* The only possibility that these two segments intersect: */
651 if ((fabs(a0-d0)<=EPS && fabs(a1-d1)<=EPS) ||
652     (fabs(b0-c0)<=EPS && fabs(b1-c1)<=EPS))
653     best=0;
654 /* If segements do not intersect: */
655 else{
656     /* Calculate the distance from A to segment [C,D]: */
657     test = ptoseg(a0, a1, c0, c1, d0, d1);
658     if ((test-best)<=EPS){
659         best = test;
660     }
661     /* Calculate the distance from B to segment [C,D]: */
662     test = ptoseg(b0, b1, c0, c1, d0, d1);
663     if ((test-best)<=EPS){
664         best = test;
665     }
666     /* Calculate the distance from C to segment [A,B]: */
667     test = ptoseg(c0, c1, a0, a1, b0, b1);
```

```
668     if ((test-best)<=EPS){
669         best = test;
670     }
671     /* Calculate the distance from D to segment [A,B]: */
672     test = ptoseg(d0, d1, a0, a1, b0, b1);
673     if ((test-best)<=EPS){
674         best = test;
675     }
676 }
677 
678 return best;
679 }
680 
681 
682 /* ---------------------- ptoseg ----------------------- */
683 /* This function treats the segment as a parameterized */
684 /* vector where the parameter t varies from 0 to 1.    */
685 /* It finds the value of t that minimizes the distance */
686 /* from the point to the line. If t is between 0.0 and */
687 /* 1.0, then the closest point lies on the segment,    */
688 /* otherwise the closest point is one of the segment's */
689 /* end points. The program finds this closest point    */
690 /* and calculates the distance between it and the      */
691 /* target point.                                       */
692 /* ----------------------------------------------------- */
693 double ptoseg(double p0, double p1, double a0, double a1,double b0, double b1){
694 
695 double t, tmp1, tmp2;
696 double val;
697 
698 tmp1 = b0-a0;
699 tmp2 = b1-a1;
700 
701 if (tmp1<=EPS && tmp2<=EPS){
702     tmp1 = p0-a0;
703     tmp2 = p1-a1;
704 }
705 else{
706     t = ((p0-a0)*(b0-a0)+(p1-a1)*(b1-a1))/((b0-a0)*(b0-a0)+(b1-a1)*(b1-a1));
707 
708     if (t<=EPS){
709         tmp1 = p0-a0;
710         tmp2 = p1-a1;
711     }
712     else if ((1-t)<=EPS){
713         tmp1 = p0-b0;
714         tmp2 = p1-b1;
715     }
716     else{
717         tmp1 = p0 - a0 - t*tmp1;
718         tmp2 = p1 - a1 - t*tmp2;
719     }
720 }
721 val  = sqrt(tmp1*tmp1 + tmp2*tmp2);
722 return val;
723 }
```

## A.4 Fehlerschätzer

### A.4.1 Datenfehlerschätzer

```
function val = data_error_estimator(coordinates,elements)

% computes || h^(1/2)*(1-PI_h))dg/ds ||(L_2)

val = zeros(length(elements),1);

% computes h^(-1)*|| g_hat - g_h_hat ||^2(L_2)
% g_h_hat: nodale interpolation of g
% g_hat: approximative calculation of g using interpolating Gauss
% quadrature with two nodes
% || g_hat - g_h_hat ||^2(L_2) ~ gauss_quadrature( (g_hat-g_h_hat)^2 , 2 )

% gauss nodes and weights for quadrature - now: just two points
NOQP = 2;
[gauss_nodes,gauss_weights] = gauss(2);

for i=1:length(elements)
   a = coordinates(elements(i,1),:);
   b = coordinates(elements(i,2),:);
   ga = g(a);
   gb = g(b);
   g_h_hat = zeros(NOQP,1);
   g_hat = zeros(NOQP,1);
   for j=1:NOQP
      g_h_hat(j) = 0.5* (ga+gb+gauss_nodes(j)*(gb-ga) );
      g_hat(j) = g( 0.5*(a+b+gauss_nodes(j)*(b-a)) );
      val(i) = val(i) + gauss_weights(j)* ( g_hat(j)-g_h_hat(j) )^2;
   end
   val(i) = val(i)*0.5;
end
```

### A.4.2 Residualfehlerschätzer

```
#include <math.h>
#include "mex.h"

#define EPS 1e-15
#define PI 3.141592653589793116

void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[]);

double residual(double* elements, double* coordinates, int nE, int nC,
                double* g_nodes, double* x_tilde, double x0, double x1,
                double* aa, double* bb, double gaa, double gbb);

void residual_error_estimator(double* rho, double* elements, int nE,
                              double* coordinates, int nC, double* phi, int p,
                                 double* gx);

void divided_differences(double* lambda, double* x, double* y, int length);

void G (double* val, int k, double* p, double* q);
void L (double* val, int k, double* p, double* q);


```

```
22
23 /*******************************************/
24 /* transfers data from matlab to C vice versa */
25 /* replaces main - function                   */
26 /*******************************************/
27
28 void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[])
29 {
30   const char* functionName = mexFunctionName();
31   char errorMessage[255];
32   int j;
33   int nE, nC;
34   double* elements;
35   double* coordinates;
36   double* phi;
37   double* p;
38   int converted_p;
39   char* g;
40   double* rho;
41   mxArray *g_output;
42   double *gx;
43
44   /* check data */
45   if (nlhs != 1) {
46     sprintf(errorMessage,"Use rho_T = %s(coordinates,elements,'g').",functionName);
47     mexErrMsgTxt(errorMessage);
48   }
49
50   if (nrhs != 5) {
51     sprintf(errorMessage,"Use rho_T = %s(coordinates,elements,'g').",functionName);
52     mexErrMsgTxt(errorMessage);
53   }
54
55   /* Read input data */
56   coordinates = mxGetPr(prhs[0]);
57   nC = mxGetM(prhs[0]);          /* number of nodes */
58
59   elements = mxGetPr(prhs[1]);
60   nE = mxGetM(prhs[1]);          /* number of elements */
61
62   phi = mxGetPr(prhs[2]);
63
64   p = mxGetPr(prhs[3]);
65   converted_p = (int) p[0];
66
67   /* (K+1/2)g has to be computed -> g is the name (string!) of a matlab */
68   /* function realizing the evaluation                                  */
69   g = mxArrayToString(prhs[4]);
70
71   /* compute all evaluations of function g */
72   mexCallMATLAB(1,&g_output,1,prhs,g);
73   gx = mxGetPr(g_output);
74
75   /* create output vector rho */
76   plhs[0] = mxCreateDoubleMatrix(nE,1,mxREAL);
77   rho = mxGetPr(plhs[0]);
78
79   /* compute entries of rho */
80   residual_error_estimator(rho,elements,nE,coordinates,nC,phi,converted_p,gx);
81   mxDestroyArray(g_output);
82 }
```

```
83
84  /*****************************************/
85  /* computes residual Rh = V\phi - (K+1/2)g */
86  /*****************************************/
87  double residual(double* elements, double* coordinates, int nE, int nC,
88                  double* g_nodes, double* x_tilde, double x0, double x1,
89                  double* aa, double* bb, double gaa, double gbb)
90  {
91    int i;
92    double Rh = 0.;
93    int nodesi[2];
94    double a[2], b[2], d0, d1, n0, n1;
95    double norm_d, qn, dlp_ab, slp_ab;
96    double p[2], q[2], gp, gq;
97    double tmp[2], tmp1;
98    double ga,gb;
99
100   for (i=0; i<nE; ++i)
101   {
102     nodesi[0] = (int) elements[i]-1;
103     nodesi[1] = (int) elements[i+nE]-1;
104
105     a[0] = coordinates[nodesi[0]];
106     a[1] = coordinates[nodesi[0]+nC];
107     b[0] = coordinates[nodesi[1]];
108     b[1] = coordinates[nodesi[1]+nC];
109
110
111     q[0] = 0.5 * ( a[0] + b[0] ) - x0;
112     q[1] = 0.5 * ( a[1] + b[1] ) - x1;
113     p[0] = 0.5 * ( b[0] - a[0] );
114     p[1] = 0.5 * ( b[1] - a[1] );
115
116     ga = g_nodes[nodesi[0]];
117     gb = g_nodes[nodesi[1]];
118
119     gp = gb-ga;
120     gq = gb+ga;
121
122     d0 = b[0] - a[0];
123     d1 = b[1] - a[1];
124     norm_d = sqrt( d0 * d0 + d1 * d1 );
125
126     n0 = d1 / norm_d;
127     n1 = -d0 / norm_d;
128     qn = q[0] * n0 + q[1] * n1;
129
130     /* compute dlp                                                    */
131     /* tmp is a vector containing [G(0,p,q),G(1,p,q)]                 */
132     /* G(k,p,q) = \int_{-1}^1 t^k / ( |p|^2t^2 + 2<p,q>t + |q|^2 )    */
133     G(tmp,1,p,q);
134     dlp_ab = gq * tmp[0];
135     dlp_ab += gp * tmp[1];
136     dlp_ab = -0.125 * norm_d * qn * dlp_ab / PI;
137     /* compute slp            */
138     /* tmp contains L(0,p,q) */
139     /* L(k,p,q) = \int_{-1}^1 t^k * log ( |p|^2t^2 + 2<p,q>t + |q|^2 ) */
140     L(tmp,0,p,q);
141     slp_ab = -0.125 * norm_d * tmp[0] / PI;
142     /* build Kg-V\phi */
143     Rh = Rh + dlp_ab - x_tilde[i]*slp_ab;
```

```
144   }
145
146   tmp1 = sqrt ( (x0-aa[0]) * (x0-aa[0]) + (x1-aa[1]) * (x1-aa[1]) );
147   tmp1 = tmp1 / sqrt ( (bb[0]-aa[0]) * (bb[0]-aa[0])
148        + (bb[1]-aa[1]) * (bb[1]-aa[1]) );
149
150   tmp1 = tmp1 * ( gbb - gaa ) + gaa;
151
152   Rh = Rh + 0.5 * tmp1;
153   return Rh;
154 }
155
156 /*************************************/
157 /* returns residual error estimator rho */
158 /* val(j) = rho,T                       */
159 /*        = || h^(1/2) * Rh' ||         */
160 /*************************************/
161 void residual_error_estimator(double* rho, double* elements, int nE,
162                               double* coordinates, int nC, double* phi, int p,
163                               double* g_nodes)
164 {
165   int m,ell,j,i,k;
166   double *nodes, *weights;
167   double *trafo_nodes;
168   double *Rh;
169   int nodesm[2], nodesell[2];
170   double aa[2], bb[2], dd0, dd1;
171   double norm_dd;
172   double x0,x1;
173   double tmp, sum, product;
174   double* lambda;
175   double rh_prime;
176   double gaa, gbb;
177
178   nodes = mxMalloc(p*sizeof(double));
179   weights = mxMalloc(p*sizeof(double));
180   gauss(nodes, weights, p);
181   trafo_nodes = mxMalloc(2*p*nE*sizeof(double));
182
183   /* rh is a vector containing the residual of length p */
184   /* rh(x) = (K+1/2)g_\ell(x)-V\phi_hH(x)               */
185   Rh = mxCalloc(p,sizeof(double));
186
187   /* loop runs trough the elements */
188   for (m=0; m<nE; ++m)
189   {
190     tmp = 0.;
191
192     /* build vector containing the nodes of an element m */
193     /* be careful - matlab's indices: 1,...,n            */
194     /*                    C's indices: 0,...,n-1         */
195     nodesm[0] = (int) elements[m]-1;
196     nodesm[1] = (int) elements[m+nE]-1;
197
198     /* a contains the coordinates of the first node and */
199     /* b contains the coordinates of the second node of */
200     /* element k */
201     aa[0] = coordinates[nodesm[0]];
202     aa[1] = coordinates[nodesm[0]+nC];
203     bb[0] = coordinates[nodesm[1]];
204     bb[1] = coordinates[nodesm[1]+nC];
```

```
205
206       /* dd = bb-aa */
207       dd0 = bb[0]-aa[0];
208       dd1 = bb[1]-aa[1];
209       norm_dd = sqrt(dd0*dd0+dd1*dd1);
210
211       /* search g(aa) and g(bb) in vector g_nodes */
212       /* you don't have to compute it twice !!!   */
213       gaa = g_nodes[nodesm[0]];
214       gbb = g_nodes[nodesm[1]];
215
216       /* transformation of quadrature points [-1,1] -> [aa,bb] */
217       /* the residual is evaluated at the parameterized gauss nodes */
218       for (ell=0; ell<p; ++ell)
219       {
220         x0 = 0.5 * ( (1.-nodes[ell])*aa[0] + (1.+nodes[ell])*bb[0] );
221         x1 = 0.5 * ( (1.-nodes[ell])*aa[1] + (1.+nodes[ell])*bb[1]);
222         Rh[ell] = residual(elements,coordinates,nE,nC,g_nodes,phi,
223                            x0,x1,aa,bb,gaa,gbb);
224       }
225
226       /* interpolation polynomial rh through p points ( nodes,Rh(nodes) ) */
227       /* of degree p-1                                                    */
228       /* in Newton basis with coefficients lambda                         */
229       lambda = mxCalloc(p,sizeof(double));
230       divided_differences(lambda,nodes,Rh,p);
231
232       /* compute rh'                                                  */
233       /* compute approximation of rho(T) for an element T=[aa,bb] */
234       /* rho(T) \approx 2 * sum_{ell=1}^p w_ell (rh')^2           */
235       for (ell=0; ell<p; ++ell)
236       {
237         rh_prime = 0.;
238         for(j=0; j<p-1; ++j)
239         {
240           sum = 0.;
241           for (i=0; i<=j; ++i)
242           {
243             product = 1.;
244             for (k=0; k<=j; ++k)
245             {
246               if (k != i)
247                 product *= (nodes[ell]-nodes[k]);
248             }
249             sum += product;
250           }
251           rh_prime += lambda[j+1]*sum;
252         }
253         tmp += weights[ell]*rh_prime*rh_prime;
254       }
255       rho[m] += 2.*tmp;
256       mxFree(lambda);
257     }
258
259     mxFree(nodes);
260     mxFree(weights);
261     mxFree(trafo_nodes);
262     mxFree(Rh);
263   }
264
265   /*************************************************************/
```

```
266 /* divided differences compute polynomial in Newton basis   */
267 /* with coefficients lambda_i out of given points (x_i,y_i) */
268 /***********************************************************/
269 void divided_differences(double* lambda, double* x, double* y, int length)
270 {
271   int m,j;
272
273   for (m=0; m<length; m++)
274     lambda[m]=y[m];
275   for (m=1; m<length; m++)
276     for(j=m; j<length; j++)
277       lambda[j] = ( lambda[j]-lambda[j-1] ) / ( x[j]-x[j-m] );
278 }
279
280 /*************************************************************/
281 /* G(k,p,q) = \int_{-1}^1 t^k / ( |p|^2t^2 + 2<p,q>t + |q|^2 )*/
282 /*************************************************************/
283 void G (double* val, int k, double* p, double* q)
284 {
285   double norm_p, norm_q, norm_p2, norm_q2;
286   double sc_p_q, delta;
287   double tmp;
288   double* ppq, *pmq;
289   int j;
290
291   norm_p2 = p[0]*p[0]+p[1]*p[1];
292   norm_p = sqrt(norm_p2);
293   norm_q2 = q[0]*q[0]+q[1]*q[1];
294   norm_q = sqrt(norm_q2);
295   sc_p_q = p[0]*q[0]+p[1]*q[1];
296   tmp = 4*(norm_p2*norm_q2-sc_p_q*sc_p_q);
297
298   if (tmp > EPS * 4*norm_p2*norm_q2)
299    delta = sqrt(tmp);
300   else
301    delta = 0;
302
303   /* case k==0 */
304   if (delta == 0)
305     val[0] = 2 / (norm_q2-norm_p2);
306   else
307   {
308    if (fabs(norm_p-norm_q) < EPS * norm_q)
309      val[0] = PI/2;
310    else if (norm_p < norm_q)
311      val[0] = atan( delta / (norm_q2-norm_p2) );
312    else
313      val[0] = atan ( delta / (norm_q2-norm_p2) ) + PI;
314    val[0] *= 2 / delta;
315   }
316
317   /* case k==1 */
318   if (k>=1)
319   {
320     ppq = mxCalloc(2,sizeof(double));
321     pmq = mxCalloc(2,sizeof(double));
322     ppq[0] = p[0]+q[0];
323     ppq[1] = p[1]+q[1];
324     pmq[0] = p[0]-q[0];
325     pmq[1] = p[1]-q[1];
326     val[1] = ( log( ppq[0]*ppq[0]+ppq[1]*ppq[1] )
```

```
327             - log( pmq[0]*pmq[0]+pmq[1]*pmq[1] )
328             - 2*sc_p_q*val[0] )*0.5/norm_p2;
329     mxFree(ppq);
330     mxFree(pmq);
331   }
332     /* case k>1 */
333   for(j=2; j<=k; j++)
334   {
335     tmp = 2 * ( (j+1)%2 ) / (j-1);
336     val[j] =  ( tmp - 2*sc_p_q*val[j-1] - norm_q2*val[j-2] )/norm_p2;
337   }
338 }
339
340 /*****************************************************************/
341 /* L(k,p,q) = \int_{-1}^1 t^k * log ( |p|^2t^2 + 2<p,q>t + |q|^2 ) */
342 /*****************************************************************/
343 void L (double* val, int k, double* p, double* q)
344 {
345   double norm_p, norm_q, norm_p2, norm_q2;
346   double norm_ppq2, norm_pmq2;
347   double sc_p_q, delta;
348   double tmp, tmp1, tmp2;
349   double* ppq, *pmq;
350   double G_p_q_0;
351   int j;
352
353   norm_p2 = p[0]*p[0]+p[1]*p[1];
354   norm_p = sqrt(norm_p2);
355   norm_q2 = q[0]*q[0]+q[1]*q[1];
356   norm_q = sqrt(norm_q2);
357   sc_p_q = p[0]*q[0]+p[1]*q[1];
358
359   ppq = mxCalloc(2,sizeof(double));
360   pmq = mxCalloc(2,sizeof(double));
361   ppq[0] = p[0]+q[0];
362   ppq[1] = p[1]+q[1];
363   pmq[0] = p[0]-q[0];
364   pmq[1] = p[1]-q[1];
365
366   norm_ppq2 = ppq[0]*ppq[0]+ppq[1]*ppq[1];
367   norm_pmq2 = pmq[0]*pmq[0]+pmq[1]*pmq[1];
368
369   tmp = 4*(norm_p2*norm_q2-sc_p_q*sc_p_q);
370   if (tmp > EPS * 4*norm_p2*norm_q2)
371     delta = sqrt(tmp);
372   else
373     delta = 0;
374
375   tmp = sc_p_q/norm_p2;
376
377   /* case k==0 */
378   if (delta>0)
379   {
380     G(&G_p_q_0,0,p,q);
381     val[0] = (1+tmp) * log( norm_ppq2 ) + (1-tmp) * log( norm_pmq2 ) - 4
382            + 2*(norm_q2-tmp*sc_p_q)*G_p_q_0;
383   }
384   else
385     val[0] = (1+tmp) * log( 2*sc_p_q+norm_p2+2*sc_p_q*tmp*0.5 )
386            + (1-tmp) * log( -2*sc_p_q+norm_p2+2*sc_p_q*tmp*0.5 )
387            -4;
```

```
388
389   /* case k==1 */
390   if (k>=1)
391     val[1] = ( norm_ppq2*log(norm_ppq2) - norm_pmq2*log(norm_pmq2)
392            - 4*sc_p_q - 2*sc_p_q*val[0] ) / ( 2*norm_p2 );
393
394   /*case k>1 */
395   for (j=2; j<=k; ++j)
396   {
397     tmp = 2 * ( (j+1)%2 ) / (j+1);
398     tmp1 = 1-2*(j%2);
399     tmp2 = 2 * (j%2) / j;
400       val[j] = ( norm_ppq2*log(norm_ppq2) + tmp1*norm_pmq2*log(norm_pmq2)
401              - 2*norm_p2*tmp - 2*sc_p_q*tmp2 - 2*sc_p_q*j*val[j-1]
402              - norm_q2*(j-1)*val[j-2] ) / ( (j+1)*norm_p2 );
403   }
404   mxFree(ppq);
405   mxFree(pmq);
406 }
```

# Literaturverzeichnis

[1] M. Ainsworth, W. McLean, T. Tran: *The Conditioning of Boundary Element Equations on Locally Refined Meshes and Preconditioning by Diagonal Scaling*, SIAM J.Sci.Comp., volume 36, number 6 (1901-1932), 1999

[2] H. W. Alt: *Lineare Funktionalanalysis*, Springer-Verlag, Berlin, Heidelberg, New York, 2002

[3] J. Bergh, J. Löfström: *Interpolation Spaces: An Introduction*, Springer-Verlag, Berlin, Heidelberg, New York, 1976

[4] D. Braess: *Finite Elemente*, Springer-Verlag, Berlin, Heidelberg, New York, 2003

[5] S.C. Brenner, L.R. Scott: *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, Berlin, Heidelberg, New York, 2002

[6] C. Carstensen: *An A Posteriori Error Estimator for the First Kind Integral Equation*, Mathematics of Computation, volume 66, number 217 (139-155), 1997

[7] C. Carstensen, M.Maischak, E.P. Stephan: *A posteriori error estimate and h-adaptive algorithm on surfaces for Symm's integral equation*, Numer.Math.90, number 2 (197-213), 2001

[8] C. Carstensen, D. Praetorius: *Averaging techniques for the a posteriori BEM error control for a hypersingular integral equation in two dimensions*, SIAM J.Sci.Comp., volume 29, number 2 (782-810), 2007

[9] C. Carstensen, D. Praetorius: *Averaging techniques for the effective numerical solution of Symm's integral equation of the first kind*, SIAM J.Sci.Comp., volume 27, number 4 (1226-1260), 2006

[10] L. C. Evans: *Partial Differential Equations*, American Mathematical Society, Providence, Rhode Island, 1998

[11] S. Ferraz-Leite: *Simple A Posteriori Fehlerschätzer für die Symmsche Integralgleichung in 3D*, Diplomarbeit, Institute for Analysis and Scientific Computing, Vienna University of Technology, Wien, 2007

[12] S. Ferraz-Leite, D. Praetorius: *Simple A Posteriori Error Estimators for the h Version of the Boundary Element Method*, ASC Report 01/2007, Institute for Analysis and Scientific Computing, Vienna University of Technology, Wien, 2007

[13] I. G. Graham, W. Hackbusch, S. A. Sauter: *Finite Elements on degenerate meshes: inverse-type inequalities and applications*, IMA J. Numer. Anal. 25 (379-407), 2005

[14] M. Maischak: *The analytical computation of the Galerkin elements for the Laplace, Lamé and Helmholtz equation in 2D-BEM*, Preprint, Institut für Angewandte Mathematik, University of Hannover, Germany, 2001

[15] W. McLean: *Strongly Elliptic Systems and Boundary Integral Equations*, Cambridge University Press, Cambridge, New York, Melbourne, Madrid, 2000

[16] R. Plato: *Numerische Mathematik kompakt*, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 2000

[17] D. Praetorius: *Introduction to Boundary Element Method*, lecture notes, Institute for Analysis and Scientific Computing, Vienna University of Technology, Wien, 2007

[18] D. Praetorius: *Integral Equations and Boundary Element Methods*, lecture notes, Institute for Analysis and Scientific Computing, Vienna University of Technology, Wien, 2007

[19] S. Sauter, C. Schwab: *Analyse, Numerik und Implementierung schneller Algorithmen*, Teubner Verlag, Stuttgart, Leipzig, Wiesbaden, 2004

[20] M. Schechter: *Principles of Functional Analysis*, American Mathematical Society, Providence, Rhode Island, 2002

[21] S. Steinbach: *Numerische Näherungsverfahren für elliptische Randwertprobleme: Finite Elemente und Randelemente*, Teubner Verlag, Stuttgart, Leipzig, Wiesbaden, 2004

[22] D. Werner: *Funktionalanalysis*, Springer-Verlag, Berlin, Heidelberg, New York, 2000

[23] J. Wloka: *Partielle Differentialgleichungen*, Teubner Verlag, Stuttgart, 1982

[24] H. Woracek, M. Kaltenbäck: *Funktionalanalysis*, Vorlesungsskriptum, Institute for Analysis and Scientific Computing, Vienna University of Technology, Wien, 2007