



TECHNISCHE
UNIVERSITÄT
WIEN
Vienna | Austria

DIPLOMARBEIT

Abstrakte a-priori-Analyse der FEM für elliptische, unrestringierte Optimal-Steuer-Probleme

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

unter der Anleitung von
Univ.Prof. Jens Markus Melenk, Ph.D.

durch
Niklas Angleitner
Westbahnstraße 13/1/11
1070 Wien.

Datum: 23. September 2015

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Ort, Datum:

Unterschrift:

Motivation

Wir motivieren den Ausdruck *Optimal-Steuer-Problem* durch das folgende Beispiel: Sei $\Omega \subseteq \mathbb{R}^d$, ($d \in \{1, 2, 3\}$), ein physikalischer Körper (z.B. ein Bauteil aus Metall, Glas oder Ähnlichem) und bezeichne mit $u \in H_0^1(\Omega)$ die (stationäre) Temperatur-Verteilung dieses Körpers. Wird der Körper von außen durch eine auf Volumina wirkende Wärmequelle $f \in L^2(\Omega)$ (z.B. Strahlung oder magnetische Induktion) erhitzt, dann gehorcht die Temperatur-Verteilung u der *Laplace-Gleichung*

$$\begin{aligned} -\Delta u &= f & \text{in } \Omega, \\ u|_{\partial\Omega} &= 0. \end{aligned}$$

Man bemerke, dass dadurch jeder Wärmequelle f eine eindeutige Temperatur-Verteilung u zugeordnet wird, d.h. die *Zustands-Größe* u lässt sich über die Wahl der *Steuer-Größe* f steuern. (Um diesen funktionalen Zusammenhang $f \mapsto u$ klarer hervorzuheben, werden wir bis zum Ende dieser Einleitung die Notation $u[f]$ für das aus f resultierende u verwenden.)

Wir geben uns nun eine gewünschte Temperatur-Verteilung $u_0 \in H_0^1(\Omega)$ vor und stellen uns die folgende Frage:

Wie muss die Wärmequelle f gewählt werden, sodass die aus ihr resultierende Temperatur-Verteilung $u[f]$ möglichst gut mit der vorgegebenen Verteilung u_0 übereinstimmt und gleichzeitig die Kosten für f klein bleiben?

Diese Wärmequelle f heißt dann *optimale Steuerung* und $u[f]$ der zugehörige *Zustand*. Entsprechend wird die Fragestellung an sich als *Optimal-Steuer-Problem* bezeichnet.

Eine mathematisch präzisere Formulierung dieses Optimal-Steuer-Problems lautet wie folgend:

Seien $u_0 \in H_0^1(\Omega)$ und $\kappa \in (0, 1)$ gegeben^a. Welche Wärmequelle $f \in L^2(\Omega)$ minimiert das Funktional

$$J : \begin{cases} L^2(\Omega) & \longrightarrow \\ f & \longmapsto \end{cases} \underbrace{(1-\kappa)}_{\text{Balance}} \cdot \underbrace{\|u_0 - u[f]\|_{H^1(\Omega)}^2}_{\text{Approx.-Fehler}} + \underbrace{\kappa}_{\text{Balance}} \cdot \underbrace{\|f\|_{L^2(\Omega)}^2}_{\text{Kosten}} \quad ?$$

^aDer Parameter $\kappa \in (0, 1)$ ist für die Balance zwischen dem Fehler-Term $\|u_0 - u[f]\|_{H^1(\Omega)}^2$ und dem Kosten-Term $\|f\|_{L^2(\Omega)}^2$ verantwortlich. Kleine Werte $\kappa \approx 0$ führen dazu, dass der Zustand $u[f]$ die Vorgabe u_0 besser approximiert, aber gleichzeitig auch die Kosten für die Steuerung f wachsen. Große Werte $\kappa \approx 1$ hingegen haben genau den umgekehrten Effekt. (Vgl. hierzu insbesondere die beiden Ergebnis-Plots der MATLAB-Funktionen `kappaComparison.m` (1D) und `kappaComparison.m` (2D) in Untersektion 3.2.)

Referenzen

Die Theorie der *Optimalen Steuerung von partiellen Differentialgleichungen* ist bereits gut erforscht, eine Einführung findet man beispielsweise in den Büchern [Lions, 1971], [Hinze et al., 2009] und [Tröltzsch, 2009]. **Diese Bücher dienen als Anhaltspunkt beim Verfassen dieser Arbeit.** Auch zur *Numerik* von Optimal-Steuer-Problemen findet man zahlreiche Arbeiten, wir erwähnen hier exemplarisch das Paper [Tröltzsch, 2010].

Weiters geben wir eine unvollständige Liste von Autoren/-innen, die sich unter Anderem mit der optimalen Steuerung von partiellen Differentialgleichungen und der zugehörigen Numerik beschäftigen, an:

- Klaus Deckelnick (Univ. Magdeburg): <http://www-ian.math.uni-magdeburg.de/home/deckelnick>
- Michael Hinze (Univ. Hamburg): <http://www.math.uni-hamburg.de/home/hinze>

- Irena Lasiecka (Univ. of Virginia): <http://pi.math.virginia.edu/Faculty/Lasiecka>
- Rolf Rannacher (Univ. Heidelberg): <http://ganymed.iwr.uni-heidelberg.de>
- Jean-Pierre Raymond (Univ. de Toulouse): <http://www.math.univ-toulouse.fr/~raymond>
- Arnd Rösch (Univ. Duisburg-Essen): <https://www.uni-due.de/mathematik/agroesch/roesch.shtml>
- Fredi Tröltzsch (TU Berlin): https://www.math.tu-berlin.de/fachgebiete_ag_modnumdiff/fg_optimierung_bei_partiellen_differentialgleichungen/v-menue/mitarbeiter/prof_dr_fredi_troeltzsch
- Michael Ulbrich (TU München):
<https://www-m1.ma.tum.de/bin/view/Lehrstuhl/MichaelUlbrich/>
- Stefan Ulbrich (TU Darmstadt): <http://www3.mathematik.tu-darmstadt.de/hp/optimization/ulbrich-stefan/stefan-ulbrich.html>
- Boris Vexler (TU München): <https://www-m17.ma.tum.de/Lehrstuhl/BorisVexler>
- Stefan Volkwein (Univ. Konstanz): <http://www.math.uni-konstanz.de/numerik/personen/volkwein/index.php>
- ...

Klassifikation dieser Arbeit

Wir fassen hier einige wesentliche Merkmale der vorliegenden Arbeit zusammen:

- **Elliptische Probleme mit verteilter Steuerung:** Das funktionalanalytische Framework dieser Arbeit wurde auf die Behandlung von *elliptischen* Problemen mit *verteilter Steuerung* zugeschnitten. Im obigen Beispiel zur Laplace-Gleichung ist f eine solche *verteilte* Steuerung, da sie auf die Volumina in Ω wirkt. Alternativ betrachten viele Autoren in der Literatur auch Probleme mit *Rand-Steuerung*, d.h. die Steuer-Größe wirkt nur auf die Oberfläche $\partial\Omega$ ein.
- **Unrestringierte Probleme:** In der Literatur findet man zumeist Aussagen über *restringierte* Optimal-Steuer-Probleme, d.h. es wird ein Optimierer von $J|_C$ gesucht, wobei J z.B. wie oben und $C \subseteq L^2(\Omega)$ eine (meist abgeschlossene und konvexe) Teilmenge ist. Wir beschränken uns in dieser Arbeit jedoch auf die einfacheren, *unrestringierten* Probleme, da sich diese in ein äquivalentes, einfach zu analysierendes *Optimalitäts-System* umformulieren lassen (vgl. Satz 2.4.1.1). Sämtliche Resultate in dieser Arbeit basieren essentiell auf dieser Umformulierung, insbesondere können die erarbeiteten Aussagen *nicht* ohne Weiteres auf restringierte Probleme verallgemeinert werden.
- **Abstrakte Analyse:** Die wichtigsten Resultate (Existenz und Eindeutigkeit von kontinuierlichen und diskreten Lösungen, Konvergenz in starken und schwachen Normen, Formulierung als LGS, Auswirkungen von sog. *variational crimes*) werden in einem rein abstrakten Rahmen formuliert und bewiesen. Insbesondere können die Untersektionen 2.1, 2.2, 2.3, 2.4 *ohne* Vorwissen zur Approximations-Theorie von Spline-Räumen aus Sektion 1 gelesen werden. Auf diese Vorgehensweise wurde hier viel Wert gelegt und sie ist *nicht* Standard in der Literatur.
- **A-priori-Abschätzungen:** Wir befassen uns ausschließlich mit der Herleitung von *a-priori*-Fehler-Abschätzungen. *A-posteriori*-Abschätzungen zur Fehler-Kontrolle für adaptive Gitter werden hier *nicht* behandelt.
- **LBB-Bedingung:** Das obige Beispiel zur Laplace-Gleichung ist ein sehr einfaches Optimal-Steuer-Problem, da die zur Differentialgleichung gehörige Bilinearform koerziv ist (vgl. Lemma 2.1.2.5). Wir entwickeln in dieser Arbeit darüber hinaus die Theorie für kompliziertere partielle Differentialgleichungen, deren beschreibende Bilinearform nur die LBB-Bedingung erfüllt (vgl. Korollar 2.1.2.3). Eine Einführung zur inf-sup-Theorie findet man z.B. in [Brezzi and Fortin, 1991] und [Braess, 2013].

Zusammenfassung der einzelnen Sektionen

Wir geben nun eine kurze Zusammenfassung der einzelnen Untersektionen an. **Den Kern dieser Arbeit bildet die Untersektion 2.4.**

-) **Sektion 1: Funktionen-Räume:**

-) *Untersektion 1.1: Der Raum $H^k(\Omega)$:* Wir definieren den Sobolev-Raum $H^k(\Omega)$ und fassen bereits bekannte Aussagen darüber zusammen.
-) *Untersektion 1.2: Der Raum $\mathbb{S}_{(0)}^{g,k}(\mathcal{T})$:* Wir definieren zuerst Gitter in den Raum-Dimensionen $d \in \{1, 2, 3\}$ und untersuchen dann die Approximations-Eigenschaften der Spline-Räume $\mathbb{S}_{(0)}^{g,k}(\mathcal{T})$ für $[g \geq 0, k = 0]$ und für $[g \geq 1, k = 1]$. Außerdem geben wir explizite Basen dieser Räume an.

-) **Sektion 2: Optimal-Steuer-Probleme:**

-) *Untersektion 2.1: Die inf-sup-Bedingung $[\alpha(e) > 0]$:* Hier werden die wesentlichen Resultate rund um die sog. inf-sup-Bedingung erarbeitet.
-) *Untersektion 2.2: Variations-Probleme:* Es werden allgemeine Variations-Probleme in einem abstrakten Setting formuliert. Es folgen Existenz und Eindeutigkeit von kontinuierlicher und diskreter Lösung, Konvergenz in starken und schwachen Normen, eine Formulierung als LGS sowie die Auswirkungen von sog. *variational crimes*. **Alle weiteren Aussagen aus den Untersektionen 2.3 und 2.4 werden auf diese Resultate zurückgeführt.** Wir beschreiben diesen Umstand schematisch mit

$$\{\text{Optimal-Steuer-Probleme}\} \subseteq \{\text{Sattelpunkt-Probleme}\} \subseteq \{\text{Variations-Probleme}\}.$$

-) *Untersektion 2.3: Sattelpunkt-Probleme:* Die Ergebnisse aus Untersektion 2.2 werden direkt auf Sattelpunkt-Probleme angewendet.
-) *Untersektion 2.4: Optimal-Steuer-Probleme:* **Diese Untersektion ist der Kern der vorliegenden Arbeit.** Die Ergebnisse aus Untersektion 2.3 werden direkt auf Optimal-Steuer-Probleme angewendet.
-) *Untersektion 2.5: Ein konkretes Optimal-Steuer-Problem: Die Laplace-Gleichung:* Wir betrachten ein konkretes Optimal-Steuer-Problem am Beispiel der Laplace-Gleichung und weisen alle Voraussetzungen aus Untersektion 2.4 nach.

-) **Sektion 3: Implementierung in 1D und 2D: Optimale Steuerung der Laplace-Gleichung:**

-) *Untersektion 3.1: Exakte Lösungen, Assemblierung und Fehler-Berechnung:* Wir geben die exakten Lösungen des Optimal-Steuer-Problems zur Laplace-Gleichung in $d \in \{1, 2\}$ für die Gebiete $\Omega := (0, 1)$ bzw. $\Omega := (0, 1) \times (0, 1)$ an. Außerdem leiten wir explizite Formeln für die Assemblierung und Fehler-Berechnung her.
-) *Untersektion 3.2: Numerische Ergebnisse:* Hier findet man die numerischen Ergebnisse (Plots der berechneten Lösungen, Fehler-Plots, Konditionszahl-Plots) des MATLAB-Programms.
-) *Untersektion 3.3: MATLAB Programm-Codes:* Sämtliche MATLAB-Codes.

Inhaltsverzeichnis

1	Funktionen-Räume	9
1.1	Der Raum $H^k(\Omega)$	9
1.1.1	Lipschitz-Gebiete Ω	9
1.1.2	Der Raum $H^k(\Omega)$	9
1.2	Der Raum $\mathbb{S}_{(0)}^{g,k}(\mathcal{T})$	11
1.2.1	Gitter $\mathcal{T}$	11
1.2.2	Der Raum $\mathbb{S}_{(0)}^{g,k}(\mathcal{T})$	15
1.2.3	Elementare Resultate auf $\hat{T}$	16
1.2.4	Skalierungs-Argument $\hat{T} \leftrightarrow T$	24
1.2.5	Größenordnung der Referenz-Abbildungen F_T	24
1.2.6	Inverse Ungleichung auf $\mathbb{S}^{g,0}(\mathcal{T}_h)$	25
1.2.7	Approximations-Eigenschaften von $\mathbb{S}^{g,0}(\mathcal{T}_h)$ für $g \geq 0$	26
1.2.8	Basis von $\mathbb{S}^{g,0}(\mathcal{T}_h)$ für $g \geq 0$	29
1.2.9	Approximations-Eigenschaften von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ für $g \geq 1$	31
1.2.10	Basis von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ für $g \geq 1$	36
2	Optimal-Steuer-Probleme	42
2.1	Die inf-sup-Bedingung $[\alpha(e) > 0]$	42
2.1.1	Grundlegende Eigenschaften von Hilbert-Räumen	42
2.1.2	Die inf-sup-Bedingung $[\alpha(e) > 0]$	44
2.2	Variations-Probleme	48
2.2.1	(Kontinuierliche) Variations-Probleme	48
2.2.2	Diskretisierung	49
2.2.3	Konvergenz	49
2.2.4	Aubin-Nitsche-Trick	51
2.2.5	LGS	52
2.2.6	Strang-Lemma	53
2.2.7	Bemerkungen	54
2.3	Sattelpunkt-Probleme	55
2.3.1	(Kontinuierliche) Sattelpunkt-Probleme	55
2.3.2	Diskretisierung	57
2.3.3	Konvergenz	58
2.3.4	Aubin-Nitsche-Trick	59
2.3.5	LGS	61
2.3.6	Strang-Lemma	63
2.3.7	Bemerkungen	64

2.4	Optimal-Steuer-Probleme	64
2.4.1	(Kontinuierliche) Optimal-Steuer-Probleme	64
2.4.2	Diskretisierung	67
2.4.3	Konvergenz	67
2.4.4	Aubin-Nitsche-Trick	68
2.4.5	LGS	70
2.4.6	Strang-Lemma	72
2.4.7	Bemerkungen	73
2.5	Ein konkretes Optimal-Steuer-Problem: Die Laplace-Gleichung	74
2.5.1	Das (kontinuierliche) Optimal-Steuer-Problem	74
2.5.2	Diskretisierung	75
2.5.3	Konvergenz	75
2.5.4	Aubin-Nitsche-Trick	77
2.5.5	LGS	78
2.5.6	Strang-Lemma	79
3	Implementierung in 1D und 2D: Optimale Steuerung der Laplace-Gleichung	80
3.1	Exakte Lösungen, Assemblierung und Fehler-Berechnung	80
3.1.1	1D	80
3.1.2	2D	86
3.2	Numerische Ergebnisse	89
3.2.1	1D	89
3.2.2	2D	94
3.3	MATLAB Programm-Codes	97
3.3.1	1D	97
3.3.2	2D	117
	Literatur	149

1 Funktionen-Räume

1.1 Der Raum $H^k(\Omega)$

1.1.1 Lipschitz-Gebiete Ω

Definition 1.1.1.1 (Lipschitz-Gebiet).

Ein offenes, beschränktes Intervall $\Omega \subseteq \mathbb{R}$ heißt *Lipschitz'sch*. Eine offene, beschränkte, zusammenhängende Menge $\Omega \subseteq \mathbb{R}^d$ mit $d \geq 2$ heißt *Lipschitz'sch*, wenn es endlich viele orthogonale Koordinaten-Transformationen $Q_1, \dots, Q_n : \mathbb{R}^d \rightarrow \mathbb{R}^d$, Bandbreiten $\delta_1, \dots, \delta_n > 0$ und Lipschitz-stetige Funktionen $f_1, \dots, f_n \in C^{0,1}(\mathbb{R}^{d-1}, \mathbb{R})$ mit folgenden Eigenschaften gibt:

- Die Quader $Q_k([0, 1]^d)$ überdecken $\partial\Omega$:

$$\partial\Omega \subseteq \bigcup_{k=1}^n Q_k([0, 1]^d).$$

- Der Rand $\partial\Omega$ lässt sich lokal (bis auf Koordinaten-Transformation) als Graph einer Lipschitz-stetigen Funktionen darstellen:

$$\forall k \in \{1, \dots, n\} : \quad \partial\Omega \cap Q_k([0, 1]^d) = Q_k(\{(y, f_k(y)) \mid y \in [0, 1]^{d-1}\}).$$

- Das Gebiet Ω liegt lokal auf nur einer Seite von $\partial\Omega$:

$$\begin{aligned} \forall k \in \{1, \dots, n\} : \quad Q_k(\{(y, z) \mid y \in [0, 1]^{d-1}, -\delta_k + f_k(y) < z < f_k(y)\}) &\subseteq \Omega, \\ Q_k(\{(y, z) \mid y \in [0, 1]^{d-1}, f_k(y) < z < f_k(y) + \delta_k\}) &\subseteq \Omega^c. \end{aligned}$$

In diesem Fall kann eine normierte äußere Normale $n : \partial\Omega \rightarrow \mathbb{R}^d$ definiert werden und es gilt die Formel der *partiellen Integration*:

$$\forall u, v \in C^1(\bar{\Omega}) : \forall i \in \{1, \dots, d\} : \quad \int_{\Omega} (\partial_i u) \cdot v \, dV = - \int_{\Omega} u \cdot (\partial_i v) \, dV + \int_{\partial\Omega} u \cdot v \cdot n_i \, dS.$$

1.1.2 Der Raum $H^k(\Omega)$

Lemma 1.1.2.1 (Schwache Ableitung).

Sei $\Omega \subseteq \mathbb{R}^d$ Lipschitz'sch. Eine Funktion $u \in L^1_{\text{loc}}(\Omega)$ heißt *schwach differenzierbar zur Ordnung $\alpha \in \mathbb{N}_0^d$* , wenn es eine Funktion $D^\alpha u \in L^1_{\text{loc}}(\Omega)$ gibt mit

$$\forall \varphi \in C_0^\infty(\Omega) : \quad \int_{\Omega} u \cdot (D^\alpha \varphi) \, dV = (-1)^{|\alpha|} \cdot \int_{\Omega} (D^\alpha u) \cdot \varphi \, dV.$$

Es gelten die folgenden Eigenschaften:

- **Eindeutigkeit:** Die schwache Ableitung $D^\alpha u$ ist eindeutig.
- **k -fache schwache Differenzierbarkeit:** Sei $k \in \mathbb{N}_0$. Ein $u \in L^1_{\text{loc}}(\Omega)$ heißt *k -mal schwach differenzierbar*, wenn alle Ableitungen $D^\alpha u, |\alpha| \leq k$, existieren.
- **Verallgemeinerung klassischer Differenzierbarkeit:** Jedes $u \in C^k(\Omega)$ ist k -mal schwach differenzierbar und die schwachen Ableitungen stimmen mit den klassischen jeweils überein.
- **Linearität:** Sind $u, v \in L^1_{\text{loc}}(\Omega)$ k -mal schwach differenzierbar, dann ist für alle $a, b \in \mathbb{R}$ auch

$au + bv \in L^1_{\text{loc}}(\Omega)$ k -mal schwach differenzierbar mit

$$\forall |\alpha| \leq k : \quad D^\alpha(au + bv) = a \cdot D^\alpha u + b \cdot D^\alpha v.$$

-) **Produkt-Regel:** Sind $u, v \in L^2(\Omega)$ k -mal schwach differenzierbar mit $\forall |\alpha| \leq k : D^\alpha u, D^\alpha v \in L^2(\Omega)$, dann ist auch $u \cdot v \in L^1(\Omega)$ k -mal schwach differenzierbar und es gilt

$$\forall |\alpha| \leq k : \quad D^\alpha(u \cdot v) = \sum_{\beta \leq \alpha} \binom{\alpha}{\beta} \cdot (D^{\alpha-\beta} u) \cdot (D^\beta v)$$

-) **Ketten-Regel:** Seien $\tilde{\Omega} \subseteq \mathbb{R}^d$ Lipschitz'sch und $F : \tilde{\Omega} \rightarrow \Omega$ bi-Lipschitz'sch. Sei $u \in L^2(\Omega)$ schwach differenzierbar mit $\forall |\alpha| \leq 1 : D^\alpha u \in L^2(\Omega)$. Dann ist auch $u \circ F \in L^2(\tilde{\Omega})$ schwach differenzierbar und es gilt

$$\nabla(u \circ F) = [(\nabla u) \circ F] \cdot \nabla F.$$

Satz 1.1.2.2 (Der Raum $H^k(\Omega)$).

Seien $\Omega \subseteq \mathbb{R}^d$ Lipschitz'sch und $k \geq 0$. Der *Sobolev-Raum*

$$H^k(\Omega) := \{u \in L^2 \mid u \text{ ist } k\text{-mal schwach diff. bar und } \forall |\alpha| \leq k : D^\alpha u \in L^2\}$$

versehen mit dem Skalarprodukt

$$\forall u, v \in H^k(\Omega) : \quad \langle u, v \rangle_{H^k(\Omega)} := \sum_{|\alpha| \leq k} \langle D^\alpha u, D^\alpha v \rangle_{L^2(\Omega)}$$

ist ein Hilbert-Raum und es gilt

$$\overline{C^\infty(\bar{\Omega})}^{\|\cdot\|_{H^k(\Omega)}} = H^k(\Omega).$$

Wir definieren weiters den Hilbert-Raum

$$H_0^k(\Omega) := \overline{C_0^\infty(\Omega)}^{\|\cdot\|_{H^k(\Omega)}} \leq H^k(\Omega).$$

Es gelten die folgenden Eigenschaften:

-) **Spur-Operator:** Der *Spur-Operator* $(\cdot)|_{\partial\Omega} : (C^\infty(\bar{\Omega}), \|\cdot\|_{H^1(\Omega)}) \rightarrow (L^2(\partial\Omega), \|\cdot\|_{L^2(\partial\Omega)})$ ist linear und stetig und kann zu einem linearen, stetigen Operator

$$(\cdot)|_{\partial\Omega} : H^1(\Omega) \rightarrow L^2(\partial\Omega)$$

mit $\|(\cdot)|_{\partial\Omega}\| \leq C(d, \Omega)$ fortgesetzt werden.

Es gilt

$$\forall k \geq 1 : \quad H_0^k(\Omega) = \{v \in H^k(\Omega) \mid \forall |\alpha| \leq k-1 : (D^\alpha v)|_{\partial\Omega} = 0\}.$$

-) **Semi-Norm:** Für alle $k \geq 0$ definiert der Ausdruck

$$\forall u \in H^k(\Omega) : \quad |u|_{H^k(\Omega)} := \left(\sum_{|\alpha|=k} \|D^\alpha u\|_{L^2(\Omega)}^2 \right)^{\frac{1}{2}}$$

eine Semi-Norm auf $H^k(\Omega)$. Mit dem *Polynom-Raum*

$$\forall g \geq 0 : \quad \mathbb{P}_g(\Omega) := \text{span}\{\Omega \ni x \mapsto x_1^{e_1} \cdots x_d^{e_d} \mid e_i \in \{0, \dots, g\}, e_1 + \dots + e_d \leq g\}$$

gilt:

$$\forall k \geq 1 : \quad \{v \in H^k(\Omega) \mid |v|_{H^k(\Omega)} = 0\} = \mathbb{P}_{k-1}(\Omega).$$

Aufgrund der daraus resultierenden *Poincaré-Ungleichung*

$$\forall u \in H_0^k(\Omega) : \quad \|u\|_{H^k(\Omega)} \leq C(d, \Omega) \cdot |u|_{H^k(\Omega)}$$

ist $|\cdot|_{H^k(\Omega)}$ sogar eine zu $\|\cdot\|_{H^k(\Omega)}$ äquivalente Norm auf $H_0^k(\Omega)$.

-) **Sobolev'scher Einbettungs-Satz:** Im Sinne von kompakten Einbettungen gilt

$$\begin{aligned} \forall k > l \geq 0 : \quad H^k(\Omega) &\subseteq H^l(\Omega) \quad \text{dicht bzgl. } \|\cdot\|_{H^l(\Omega)}, \\ H^k(\Omega) &\hookrightarrow_c H^l(\Omega), \\ \forall k > \frac{d}{2} : \quad H^k(\Omega) &\hookrightarrow_c C^0(\overline{\Omega}). \end{aligned}$$

Insbesondere lassen sich bei $d = 1$ bereits $H^1(\Omega)$ -Funktionen und bei $d \in \{2, 3\}$ erst $H^2(\Omega)$ -Funktionen Punkt-auswerten.

1.2 Der Raum $\mathbb{S}_{(0)}^{g,k}(\mathcal{T})$

1.2.1 Gitter $\mathcal{T}$

Definition 1.2.1.1 (Gitter).

Sei $d \geq 1$. Zu $D \geq d + 1$ vielen gegebenen Punkten $N_1(\hat{T}), \dots, N_D(\hat{T}) \in \mathbb{R}^d$ bezeichne $\hat{T} := (\text{conv}\{N_i(\hat{T}) \mid i \in \{1, \dots, D\}\})^\circ \subseteq \mathbb{R}^d$ ein *Referenz-Element*. Wir nehmen an, dass die Punkte $N_i(\hat{T})$ irredundant gewählt wurden.

Ein endliches Mengensystem $\mathcal{T} \subseteq \text{Pow}(\mathbb{R}^d)$ heißt $(\hat{T})$ -Gitter, falls:

-) Jedes *Element* $T \in \mathcal{T}$ ist von der Form $T = F_T(\hat{T})$ für eine affine, bijektive *Referenz-Abbildung* $F_T : \mathbb{R}^d \rightarrow \mathbb{R}^d$. Insbesondere ist $|T| = |\det \nabla F_T| \cdot |\hat{T}| > 0$. Wir gehen im Folgenden immer davon aus, dass für jedes $T \in \mathcal{T}$ genau ein derartiges F_T ausgewählt wurde.
-) Die Elemente sind Überschneidungs-frei (Lebesgue-Maß) und es gibt keine „hängenden“ Knoten:

$$\begin{aligned} \forall S \neq T \in \mathcal{T} : \quad |S \cap T| &= 0, \\ \{F_S(N_i(\hat{T})) \mid i \in \{1, \dots, D\}\} \cap \overline{T} &\subseteq \{F_T(N_i(\hat{T})) \mid i \in \{1, \dots, D\}\}. \end{aligned}$$

Wir führen noch einige Bezeichnungen ein:

-) **Gitterweite:** Für jedes $T \in \mathcal{T}$ bezeichnen wir mit

$$\begin{aligned} h_T &:= \max_{x, y \in \overline{T}} \|x - y\|_2, \\ r_T &:= \max\{r > 0 \mid \exists x \in \mathbb{R}^d \text{ mit } B_r(x) \subseteq T\} \end{aligned}$$

den *Durchmesser* und den *Inkugel-Radius* von T . Wir definieren weiters die *maximale* bzw. *minimale Gitterweite* von $\mathcal{T}$:

$$h_{\mathcal{T}} := \max_{T \in \mathcal{T}} h_T, \quad h_{\min, \mathcal{T}} := \min_{T \in \mathcal{T}} h_T$$

-) **Polygon:** Ein Lipschitz-Gebiet^a $\Omega \subseteq \mathbb{R}^d$ heißt $(\hat{T})$ -*Polygon*, wenn es ein $\hat{T}$ -Gitter $\mathcal{T} \subseteq \text{Pow}(\mathbb{R}^d)$ gibt mit

$$\bigcup_{T \in \mathcal{T}} T \subseteq \Omega \quad \wedge \quad \left| \bigcup_{T \in \mathcal{T}} T \right| = |\Omega|.$$

-) **Formregularität:** Eine Familie^{b,c} $(\mathcal{T}_h)_{h \in H}$ von Gittern auf Ω heißt *formregulär*, falls

$$\gamma := \sup_{h \in H} \max_{T \in \mathcal{T}_h} \frac{h_T}{r_T} < \infty$$

und *quasi-uniform*, falls

$$\nu := \sup_{h \in H} \max_{T \in \mathcal{T}_h} \frac{h_{\mathcal{T}_h}}{h_T} < \infty.$$

^aDie Lipschitz-Eigenschaft garantiert insbesondere, dass Ω keine „Risse“ oder einpunktigen Engstellen (z.B. Berührungspunkt der beiden Enden eines „C-förmigen“ Gebiets) haben kann.

^bMan beachte: Der Buchstabe „ h “ tritt im Folgenden auf zwei unterschiedliche Arten auf. Einerseits verwenden wir ein alleinstehendes, tiefgestelltes „ h “ als Index zur Indizierung von Gitter-Familien. Andererseits verwenden wir h_T und $h_{\mathcal{T}}$ für den Element-Durchmesser und die maximale Gitterweite (siehe oben). Für die Indexmenge H werden wir im Verlaufe der Arbeit stets o.B.d.A. $H \subseteq (0, 1]$ und $0 \in \overline{H}$ annehmen.

^cWir werden im Folgenden nur Gitter-Familien mit $\lim_{h \rightarrow 0} h_{\mathcal{T}_h} = 0$ betrachten, daher können wir diese Eigenschaft implizit als gegeben voraussetzen.

Definition 1.2.1.2 (Gitter in 1D, 2D und 3D).

Die wichtigsten Beispiele für Gitter in den Dimensionen $d \in \{1, 2, 3\}$ sind^a:

-) **1D Intervall-Gitter:** Sei $d := 1$. Wir betrachten das durch die Punkte $N_1(\hat{T}) := 0$ und $N_2(\hat{T}) := 1$ gegebene Referenz-Element

$$\hat{T} = (0, 1) \subseteq \mathbb{R}.$$

Ein entsprechendes Gitter $\mathcal{T} \subseteq \text{Pow}(\mathbb{R})$ heißt *Intervall-Gitter*. Eine Familie $(\mathcal{T}_h)_{h \in H}$ von Intervall-Gittern ist stets formregulär mit $\gamma = 2$.

-) **2D Dreiecks-Gitter:** Sei $d := 2$. Wir betrachten das durch die Punkte $N_1(\hat{T}) := (0, 0)$, $N_2(\hat{T}) := (1, 0)$ und $N_3(\hat{T}) := (0, 1)$ gegebene Referenz-Element

$$\hat{T} = \{(x, y) \mid x \in (0, 1), y \in (0, 1 - x)\} \subseteq \mathbb{R}^2.$$

Ein entsprechendes Gitter $\mathcal{T} \subseteq \text{Pow}(\mathbb{R}^2)$ heißt *Dreiecks-Gitter*. Eine Familie $(\mathcal{T}_h)_{h \in H}$ von Dreiecks-Gittern ist formregulär genau dann, wenn sämtliche Dreiecks-Innenwinkel nach unten hin beschränkt bleiben:

$$\inf_{h \in H} \min_{T \in \mathcal{T}_h} \min \{\alpha > 0 \mid \alpha \text{ ist Innenwinkel von } T\} > 0.$$

In diesem Fall ist insbesondere die Anzahl der an einen Knoten angrenzenden Dreiecke nach oben hin beschränkt (vgl. Bezeichnungen weiter unten):

$$\forall h \in H : \forall N \in \mathcal{N}_h : \quad \#\{T \in \mathcal{T}_h \mid N \in \mathcal{N}(T)\} \leq C(\gamma).$$

-) **3D Tetraeder-Gitter:** Sei $d := 3$. Wir betrachten das durch die Punkte $N_1(\hat{T}) := (0, 0, 0)$, $N_2(\hat{T}) := (1, 0, 0)$, $N_3(\hat{T}) := (0, 1, 0)$ und $N_4(\hat{T}) := (0, 0, 1)$ gegebene Referenz-Element

$$\hat{T} = \{(x, y, z) \mid x \in (0, 1), y \in (0, 1 - x), z \in (0, 1 - x - y)\} \subseteq \mathbb{R}^3.$$

Ein entsprechendes Gitter $\mathcal{T} \subseteq \text{Pow}(\mathbb{R}^3)$ heißt *Tetraeder-Gitter*. Eine Familie $(\mathcal{T}_h)_{h \in H}$ von Tetraeder-Gittern ist formregulär genau dann, wenn sämtliche Innenwinkel (Eckwinkel der Flächen und Raumwinkel zwischen benachbarten Flächen) nach unten hin beschränkt bleiben^b:

$$\inf_{h \in H} \min_{T \in \mathcal{T}_h} \min \{\alpha > 0 \mid \alpha \text{ ist Innenwinkel von } T\} > 0.$$

In diesem Fall ist insbesondere die Anzahl der an einen Knoten/eine Kante angrenzenden Dreiecke nach oben hin beschränkt (vgl. Bezeichnungen weiter unten):

$$\begin{aligned} \forall h \in H : \forall N \in \mathcal{N}_h : \quad & \#\{T \in \mathcal{T}_h \mid N \in \mathcal{N}(T)\} \leq C_1(\gamma), \\ \forall h \in H : \forall E \in \mathcal{E}_h : \quad & \#\{T \in \mathcal{T}_h \mid E \in \mathcal{E}(T)\} \leq C_2(\gamma). \end{aligned}$$

•) **Bezeichnungen:** Wir führen noch einige Bezeichnungen ein: Es sind für jedes $T \in \mathcal{T}$

$$\begin{aligned} (d \in \{1, 2, 3\}) \quad \mathcal{N}(T) &:= \{F_T(N_i(\hat{T})) \mid i \in \{1, \dots, D\}\}, \\ (d \in \{2, 3\}) \quad \mathcal{E}(T) &:= \{\text{conv}\{M, N\} \setminus \{M, N\} \mid M \neq N \in \mathcal{N}(T)\}, \\ (d = 3) \quad \mathcal{F}(T) &:= \{\text{conv}(E \cup F \cup G) \setminus (\bar{E} \cup \bar{F} \cup \bar{G}) \mid E \neq F \neq G \in \mathcal{E}(T)\} \end{aligned}$$

die Menge der *Knoten*, die Menge der *Kanten* und die Menge der *Flächen* von T . Wir setzen weiters

$$\begin{aligned} \mathcal{N} &:= \bigcup_{T \in \mathcal{T}} \mathcal{N}(T), \\ \mathcal{E} &:= \bigcup_{T \in \mathcal{T}} \mathcal{E}(T), \\ \mathcal{F} &:= \bigcup_{T \in \mathcal{T}} \mathcal{F}(T). \end{aligned}$$

Schließlich sei $\mathcal{N}_D := \mathcal{N} \cap \partial\Omega$ die Menge der *Rand-Knoten*, $\mathcal{N}_I := \mathcal{N} \setminus \mathcal{N}_D$ die Menge der *inneren Knoten*, $\mathcal{E}_D := \mathcal{E} \cap \text{Pow}(\partial\Omega)$ die Menge der *Rand-Kanten*, $\mathcal{E}_I := \mathcal{E} \setminus \mathcal{E}_D$ die Menge der *inneren Kanten*, $\mathcal{F}_D := \mathcal{F} \cap \text{Pow}(\partial\Omega)$ die Menge der *Rand-Flächen* und $\mathcal{F}_I := \mathcal{F} \setminus \mathcal{F}_D$ die Menge der *inneren Flächen* von $\mathcal{T}$.

^aEs gibt für $d \in \{2, 3\}$ auch Gitter basierend auf $\hat{T} = (0, 1)^d \subseteq \mathbb{R}^d$, sog. *Parallelogramm-* bzw. *Parallelepiped-*Gitter. Wir werden jedoch derartige Gitter in dieser Arbeit nicht weiter behandeln.

^bDass man tatsächlich die Beschränktheit beider Arten von Winkeln benötigt, zeigen die beiden folgenden Beispiele: Die Tetraeder-Familie $(\text{conv}\{e_2, -e_2, -e_1, n \cdot e_3\})_{n \geq 1}$ hat nach unten beschränkte Raumwinkel, aber beliebig kleine Eckwinkel. Die Tetraeder-Familie $(\text{conv}\{e_2, -e_2, -e_1, (\cos(\alpha), 0, \sin(\alpha))\})_{\alpha \in (0, \frac{\pi}{2}]}$ hingegen hat nach unten beschränkte Eckwinkel, aber beliebig kleine Raumwinkel.

Beweis: (der Charakterisierungen der Formregularitäten für $d \in \{2, 3\}$)

Schritt 1: Formregularität bei Dreiecks-Gittern

Für jedes $T \in \mathcal{T}_h$ bezeichnen wir die Seitenlängen mit $0 < c \leq b \leq h$ und die jeweils gegenüberliegenden Innenwinkel mit $\alpha_{\min}, \beta, \alpha \in (0, \pi)$. Es gilt $\alpha_{\min} \leq \beta \leq \alpha$ und $\alpha_{\min} \in (0, \frac{\pi}{3}]$, $\beta \in (0, \frac{\pi}{2}]$, $\alpha \in [\frac{\pi}{3}, \pi)$. Wir bezeichnen weiters mit $r > 0$ den Inkreis-Radius und mit $H > 0$ die Höhe auf die Seite h .

Der Flächeninhalt von T lässt sich auf zwei Arten berechnen:

$$\frac{1}{2} h \cdot b \cdot \sin(\alpha_{\min}) = \frac{1}{2} h H = |T| = \frac{1}{2} r \cdot (h + b + c).$$

Die Richtung „ $\Rightarrow$ “ der Behauptung folgt aus

$$\sin(\alpha_{\min}) = \frac{r}{h} \cdot \frac{h + b + c}{b} \geq \frac{1}{\gamma} \cdot \frac{b}{b} = \frac{1}{\gamma} > 0$$

und die Richtung „ $\Leftarrow$ “ aus

$$\frac{h}{r} = \frac{1}{\sin(\alpha_{\min})} \cdot \left(\underbrace{\frac{\sin(\alpha)}{\sin(\beta)}}_{\geq \sin(\alpha_{\min})} + \frac{b+c}{b} \right) \leq \left(\frac{1}{\sin\left(\inf_{h \in H} \min_{T \in \mathcal{T}_h} \alpha_{\min, T}\right)} + 2 \right)^2 < \infty.$$

□

Schritt 2: Formregularität bei Tetraeder-Gittern

Sei $T \in \mathcal{T}_h$ gegeben. Es bezeichne $h > 0$ die längste Seite und $B_r(x_0) \subseteq T$ die größte Inkugel.

„ $\Rightarrow$ “: Gelte $\gamma > 0$. Wir betrachten zwei beliebige Flächen $F \neq G \in \mathcal{F}(T)$ und bezeichnen jenen Eckpunkt von F , der G gegenüber liegt mit $N \in \mathcal{N}(F) \setminus \mathcal{N}(G)$. Es gilt

$$\frac{4\pi}{3\gamma^3} \cdot h^3 \leq \frac{4\pi}{3} \cdot r^3 = |B_r(x_0)| \leq |T| = \frac{1}{3} \cdot |G| \cdot \text{dist}(G, N) \leq \frac{1}{3} \cdot h^2 \cdot \text{dist}(G, N).$$

Für den Raumwinkel $\alpha \in (0, \pi)$ zwischen F und G ergibt sich mit

$$\text{dist}(G, N) = \text{dist}(\overline{F} \cap \overline{G}, N) \cdot \sin(\alpha) \leq h \cdot \sin(\alpha)$$

die Abschätzung

$$\sin(\alpha) \geq \frac{4\pi}{\gamma^3} > 0.$$

Für jeden der beiden Innenwinkel $\beta \in (0, \pi)$ von F an den Knoten $M \in \mathcal{N}(F) \setminus \{N\}$ ergibt sich mit

$$\text{dist}(G, N) \leq \text{dist}(\overline{F} \cap \overline{G}, N) = \|M - N\|_2 \cdot \sin(\beta) \leq h \cdot \sin(\beta)$$

die Abschätzung

$$\sin(\beta) \geq \frac{4\pi}{\gamma^3} > 0.$$

Da der kleinste Raum- und der kleinste Eckwinkel in T die Bedingung $\alpha_{\min}, \beta_{\min} \in (0, \frac{\pi}{2})$ erfüllen, folgt durch Anwendung von $\arcsin$ die Behauptung.

„ $\Leftarrow$ “: Gelte $\alpha_{\min} := \inf_{h \in H} \min_{T \in \mathcal{T}_h} \min \{ \alpha > 0 \mid \alpha \text{ ist Innenwinkel von } T \} > 0$. Es gibt ein gerades Parallelepiped $Q \subseteq T$ mit Länge $\frac{1}{2} \cdot h$, Breite und Höhe $\frac{1}{4} \cdot h \cdot \tan(\alpha_{\min})$ und Basiswinkel $\alpha_{\min}$. Damit

$$\left[\frac{1}{16} \cdot h^2 \cdot \tan^2(\alpha_{\min}) \cdot \sin(\alpha_{\min}) \right] \cdot \frac{1}{2} \cdot h = |Q| \leq |T| = \frac{1}{3} \cdot r \cdot \sum_{F \in \mathcal{F}(T)} |F| \leq \frac{4}{3} \cdot r \cdot h^2$$

und schließlich

$$\frac{h}{r} \leq \frac{128}{3 \cdot \tan^2(\alpha_{\min}) \cdot \sin(\alpha_{\min})} < \infty.$$

□

■

Beispiel 1.2.1.3 (Ein 1D-Intervall- und ein 2D-Dreiecks-Gitter).

Wir geben je ein explizites Beispiel für ein 1D-Intervall- und ein 2D-Dreiecks-Gitter an:

- **1D Intervall-Gitter:** Seien $d := 1$, $\Omega := (0, 1) \subseteq \mathbb{R}$ und $\hat{T} \subseteq \mathbb{R}$ das Referenz-Element für Intervall-Gitter. Zu jedem $h \in H := \{ \frac{1}{n-1} \mid n \in \mathbb{N}^\times \}$ definieren wir

$$\begin{aligned} n &:= \frac{1}{h} + 1 \in \mathbb{N}^\times, \\ \forall i \in \{1, \dots, n\} : \quad N_i &:= \frac{1}{n-1} \cdot (i-1) \in \overline{\Omega}, \\ \forall i \in \{1, \dots, n-1\} : \quad F_{T_i}(x) &:= \frac{1}{n-1} \cdot x + N_i, \\ \forall i \in \{1, \dots, n-1\} : \quad T_i &:= F_{T_i}(\hat{T}) = (N_i, N_{i+1}) \subseteq \Omega \end{aligned}$$

und damit das Intervall-Gitter

$$\mathcal{T}_h := \{T_i \mid i \in \{1, \dots, n-1\}\}.$$

Es gilt $h_{\mathcal{T}_h} = \frac{1}{n-1} = h$. Es ist $(\mathcal{T}_h)_{h \in H}$ formregulär und quasi-uniform mit $\gamma = 2$ und $\nu = 1$.

-) **2D Dreiecks-Gitter:** Seien $d := 2$, $\Omega := (0, 1) \times (0, 1) \subseteq \mathbb{R}^2$ und $\hat{T} \subseteq \mathbb{R}^2$ das Referenz-Element für Dreiecks-Gitter. Zu jedem $h \in H := \{\frac{\sqrt{2}}{n-1} \mid n \in \mathbb{N}^\times\}$ definieren wir

$$\begin{aligned} n &:= \frac{\sqrt{2}}{h} + 1 \in \mathbb{N}^\times, \\ \forall i, j \in \{1, \dots, n\} : \quad N_{ij} &:= \frac{1}{n-1} \cdot (i-1, j-1) \in \overline{\Omega}, \\ \forall i, j \in \{1, \dots, n-1\} : \quad F_{T_{ij1}}(x, y) &:= \frac{1}{n-1} \cdot (x, y) + N_{ij}, \\ F_{T_{ij2}}(x, y) &:= -\frac{1}{n-1} \cdot (x, y) + N_{i+1, j+1}, \\ \forall i, j \in \{1, \dots, n-1\}, k \in \{1, 2\} : \quad T_{ijk} &:= F_{T_{ijk}}(\hat{T}) \subseteq \Omega \end{aligned}$$

und damit das Dreiecks-Gitter

$$\mathcal{T}_h := \{T_{ijk} \mid i, j \in \{1, \dots, n-1\}, k \in \{1, 2\}\}.$$

Es gilt $h_{\mathcal{T}_h} = \frac{\sqrt{2}}{n-1} = h$. Es ist $(\mathcal{T}_h)_{h \in H}$ formregulär und quasi-uniform mit $\gamma = 2 + \sqrt{8}$ und $\nu = 1$.

1.2.2 Der Raum $\mathbb{S}_{(0)}^{g,k}(\mathcal{T})$

Definition 1.2.2.1 (Der Raum $\mathbb{S}_{(0)}^{g,k}(\mathcal{T})$).

Seien $d \geq 1$, $\hat{T} \subseteq \mathbb{R}^d$ ein Referenz-Element, $\Omega \subseteq \mathbb{R}^d$ ein Polygon und $\mathcal{T} \subseteq \text{Pow}(\mathbb{R}^d)$ ein Gitter auf Ω . Wir definieren die endlich-dimensionalen *Spline-Räume*^{ab}

$$\begin{aligned} \forall g \geq 0, k \geq 0 : \quad \mathbb{S}^{g,k}(\mathcal{T}) &:= \{v \in H^k(\Omega) \mid \forall T \in \mathcal{T} : v \circ F_T \in \mathbb{P}_g(\hat{T})\}, \\ \forall g \geq 1, k \geq 1 : \quad \mathbb{S}_0^{g,k}(\mathcal{T}) &:= \mathbb{S}^{g,k}(\mathcal{T}) \cap H_0^1(\Omega). \end{aligned}$$

Es gilt:

$$\forall g \geq 1, k \geq 1 : \quad \mathbb{S}^{g,k}(\mathcal{T}) = \{v \in C^{k-1}(\overline{\Omega}) \mid \forall T \in \mathcal{T} : v \circ F_T \in \mathbb{P}_g(\hat{T})\}.$$

^aWir fordern von den Funktionen $v \in \mathbb{S}_0^{g,k}(\mathcal{T})$ also nur $v|_{\partial\Omega} = 0$, aber *nicht* unbedingt $[\forall |\alpha| \leq k-1 : (D^\alpha v)|_{\partial\Omega} = 0]$.

^bWir werden später an mehreren Stellen Aussagen angeben, die für $\mathbb{S}^{g,k}(\mathcal{T})$ und $\mathbb{S}_0^{g,k}(\mathcal{T})$ gleichermaßen zutreffen. Wir verwenden dann die verkürzte Notation $\mathbb{S}_{(0)}^{g,k}(\mathcal{T})$.

Beweis: (der Identität $\mathbb{S}^{g,k}(\mathcal{T}) = \{v \in C^{k-1}(\overline{\Omega}) \mid \forall T \in \mathcal{T} : v \circ F_T \in \mathbb{P}_g(\hat{T})\}$)

Wir zeigen, dass für jede Funktion $v \in L^2(\Omega)$ mit $\forall T \in \mathcal{T} : v|_T \in C^\infty(\overline{T})$ die Äquivalenz $v \in H^1(\Omega) \Leftrightarrow v \in C^0(\overline{\Omega})$ gilt. Die entsprechende Aussage für alle $k \geq 1$ folgt dann mittels vollständiger Induktion. Außerdem beschränken wir uns auf den Fall $d = 2$, denn für $d = 1$ ist die Aussage elementar zu zeigen und für $d \geq 3$ müssen im Beweis nur die Kanten durch (Hyper-)Flächen und γ durch eine $(d-1)$ -dimensionale affine Parametrisierung ersetzt werden.

Schritt 1: „ $\Rightarrow$ “

Gelte $v \in H^1(\Omega)$. Wir müssen die Stetigkeit von v an allen inneren Kanten $E \in \mathcal{E}_I$ zeigen. Bezeichne mit $T_1, T_2 \in \mathcal{T}$ die beiden an E angrenzenden Elemente und mit $\gamma : (0, 1) \rightarrow E$ eine affine Parametrisierung von E .

Jede Testfunktion $\psi \in C_0^\infty((0, 1))$ lässt sich in der Form $\psi = \varphi|_E \circ \gamma$ mit einer Testfunktion $\varphi \in C_0^\infty(\Omega)$

mit $\text{supp}(\varphi) \subseteq \overline{T_1 \cup T_2}$ schreiben¹. Damit

$$\begin{aligned} \forall i \in \{1, 2\} : \forall \psi \in C_0^\infty((0, 1)) : \\ \int_{(0,1)} [(v|_{T_1} - v|_{T_2}) \cdot n_i|_{T_1}] \circ \gamma \cdot \psi \, dV &= \int_{(0,1)} [(v|_{T_1} - v|_{T_2}) \cdot n_i|_{T_1} \cdot \varphi] \circ \gamma \, dV \\ &= \frac{1}{\|\gamma'\|_2} \cdot \int_E (v|_{T_1} - v|_{T_2}) \cdot n_i|_{T_1} \cdot \varphi \, dS \\ &\stackrel{v \in H^1}{=} 0. \end{aligned}$$

Das Fundamental-Lemma der Variationsrechnung liefert $(v|_{T_1} - v|_{T_2}) \circ \gamma = 0$. □

Schritt 2: „ $\Leftarrow$ “

Gelte $v \in C^0(\overline{\Omega})$. Dann ist v schwach differenzierbar:

$$\begin{aligned} \forall \varphi \in C_0^\infty(\Omega) : \quad \int_{\Omega} v \cdot (\partial_i \varphi) \, dV &= \sum_{T \in \mathcal{T}} \int_T v \cdot (\partial_i \varphi) \, dV \\ &= \sum_{E \in \mathcal{E}_I} \underbrace{\sum_{\substack{T \in \mathcal{T}: \\ E \in \mathcal{E}(T)}} \int_E v|_T \cdot \varphi \cdot n_i \, dS}_{=0} - \sum_{T \in \mathcal{T}} \int_T (\partial_i v) \cdot \varphi \, dV. \end{aligned}$$

Die L^2 -Integrierbarkeit von $v, \partial_1 v, \partial_2 v$ folgt aus der Beschränktheit auf den endlichen Maßräumen $T \in \mathcal{T}$. □

□

■

1.2.3 Elementare Resultate auf $\hat{T}$

Definition 1.2.3.1 (Lagrange-Polynome auf $\hat{T}$).

Seien $d \in \{1, 2, 3\}$, $\hat{T} \subseteq \mathbb{R}^d$ das Referenz-Element für Intervall-, Dreiecks- bzw. Tetraeder-Gitter und $g \geq 0$. Wir definieren nun *Referenz-Stützstellen*^a und *Referenz-Formfunktionen* (als die zugehörigen *Lagrange-Polynome*) auf $\hat{T}$:

•) **1D Intervall-Gitter:** Sei $d := 1$. Wir definieren die Index-Mengen^b

$$\begin{aligned} (g \geq 0) \quad I_g &:= \{0, \dots, g\}, \\ (g \geq 2) \quad I_g^\times &:= \{1, \dots, g-1\}, \end{aligned}$$

die *Referenz-Stützstellen*

$$\begin{aligned} (g = 0) \quad \hat{x}_0^0 &:= \frac{1}{2} \in \overline{\hat{T}}, \\ (g \geq 1) \quad \forall i \in I_g : \hat{x}_i^g &:= \frac{1}{g} \cdot i \in \overline{\hat{T}} \end{aligned}$$

und die *Referenz-Formfunktionen* (als die *Lagrange-Polynome*)

$$\begin{aligned} (g = 0) \quad \hat{\varphi}_0^0(x) &:= 1 \in \mathbb{P}_0(\hat{T}), \\ (g \geq 1) \quad \forall i \in I_g : \hat{\varphi}_i^g(x) &:= \prod_{\substack{l=0 \\ l \neq i}}^g \frac{x - \hat{x}_l^g}{\hat{x}_i^g - \hat{x}_l^g} \in \mathbb{P}_g(\hat{T}). \end{aligned}$$

¹Beweis-Skizze: Im Falle $E = (0, 1) \times \{0\}$ kann man $\varphi(x_1, x_2) := \psi(x_1) \cdot \chi(x_2)$ wählen, wobei $\chi \in C_0^\infty(\mathbb{R})$ eine geeignete Testfunktion mit $\chi(0) = 1$ und $\text{supp}(\chi) \subseteq B_\varepsilon(0)$ ist. Der Parameter $\varepsilon > 0$ hängt von der Lage von $\text{supp}(\psi) \subseteq (0, 1)$ und von den Innenwinkeln von T_1, T_2 ab und muss klein genug für $\text{supp}(\varphi) \subseteq \overline{T_1 \cup T_2}$ gewählt werden.

•) **2D Dreiecks-Gitter:** Sei $d := 2$. Wir definieren die Index-Mengen

$$\begin{aligned} (g \geq 0) \quad IJ_g &:= \{(i, j) \mid i \in \{0, \dots, g\}, j \in \{0, \dots, g - i\}\}, \\ (g \geq 3) \quad IJ_g^\times &:= \{(i, j) \mid i \in \{1, \dots, g - 1\}, j \in \{1, \dots, g - i - 1\}\}, \end{aligned}$$

die *Referenz-Stützstellen*

$$\begin{aligned} (g = 0) \quad (\hat{x}_0^0, \hat{y}_0^0) &:= \left(\frac{1}{3}, \frac{1}{3}\right) \in \overline{\hat{T}}, \\ (g \geq 1) \quad \forall (i, j) \in IJ_g : (\hat{x}_i^g, \hat{y}_j^g) &:= \frac{1}{g} \cdot (i, j) \in \overline{\hat{T}} \end{aligned}$$

und die *Referenz-Formfunktionen* (als die *Lagrange-Polynome*)

$$\begin{aligned} (g = 0) \quad \hat{\varphi}_{00}^0(x, y) &:= 1 \in \mathbb{P}_0(\hat{T}), \\ (g \geq 1) \quad \forall (i, j) \in IJ_g : \hat{\varphi}_{ij}^g(x, y) &:= \prod_{l=0}^{i-1} \frac{x - \hat{x}_l^g}{\hat{x}_i^g - \hat{x}_l^g} \cdot \prod_{l=0}^{j-1} \frac{y - \hat{y}_l^g}{\hat{y}_j^g - \hat{y}_l^g} \cdot \prod_{l=i+j+1}^g \frac{x + y - \hat{y}_l^g}{\hat{x}_i^g + \hat{y}_j^g - \hat{y}_l^g} \in \mathbb{P}_g(\hat{T}). \end{aligned}$$

•) **3D Tetraeder-Gitter:** Sei $d := 3$. Wir definieren die Index-Mengen

$$\begin{aligned} (g \geq 0) \quad IJK_g &:= \{(i, j, k) \mid i \in \{0, \dots, g\}, j \in \{0, \dots, g - i\}, k \in \{0, \dots, g - i - j\}\}, \\ (g \geq 4) \quad IJK_g^\times &:= \{(i, j, k) \mid i \in \{1, \dots, g - 1\}, j \in \{1, \dots, g - i - 1\}, \\ &\quad k \in \{1, \dots, g - i - j - 1\}\}, \end{aligned}$$

die *Referenz-Stützstellen*

$$\begin{aligned} (g = 0) \quad (\hat{x}_0^0, \hat{y}_0^0, \hat{z}_0^0) &:= \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) \in \overline{\hat{T}}, \\ (g \geq 1) \quad \forall (i, j, k) \in IJK_g : (\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) &:= \frac{1}{g} \cdot (i, j, k) \in \overline{\hat{T}} \end{aligned}$$

und die *Referenz-Formfunktionen* (als die *Lagrange-Polynome*)

$$\begin{aligned} (g = 0) \quad \hat{\varphi}_{000}^0(x, y, z) &:= 1 \in \mathbb{P}_0(\hat{T}), \\ (g \geq 1) \quad \forall (i, j, k) \in IJK_g : \hat{\varphi}_{ijk}^g(x, y, z) &:= \prod_{l=0}^{i-1} \frac{x - \hat{x}_l^g}{\hat{x}_i^g - \hat{x}_l^g} \cdot \prod_{l=0}^{j-1} \frac{y - \hat{y}_l^g}{\hat{y}_j^g - \hat{y}_l^g} \cdot \prod_{l=0}^{k-1} \frac{z - \hat{z}_l^g}{\hat{z}_k^g - \hat{z}_l^g} \\ &\quad \cdot \prod_{l=i+j+k+1}^g \frac{x + y + z - \hat{z}_l^g}{\hat{x}_i^g + \hat{y}_j^g + \hat{z}_k^g - \hat{z}_l^g} \in \mathbb{P}_g(\hat{T}). \end{aligned}$$

^aFür $g \geq 1$ sind bei der konkreten Verteilung der Referenz-Stützstellen auf $\hat{T}$ insbesondere folgende Punkte zu beachten: 1) Alle Knoten von $\hat{T}$ müssen als Stützstellen auftreten und es müssen hinreichend viele Stützstellen an den Kanten und Flächen auftreten (vgl. Lemma 1.2.3.6). 2) Es muss die Kronecker- δ -Eigenschaft gelten. 3) Die Stützstellen an den Kanten und Flächen müssen geeignet symmetrisch liegen. (Diese Punkte sind später essentiell für die Stetigkeit der in dieser Arbeit betrachteten Basis-Elemente von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T})$, vgl. Satz 1.2.10.2.)

^bDie Indexmengen $I_g^\times$, $IJ_g^\times$ und $IJK_g^\times$ werden so gewählt, dass für die Referenz-Stützstellen gilt (exemplarisch für $d = 3$): $(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \in \partial \hat{T} \Leftrightarrow (i, j, k) \in IJK_g \setminus IJK_g^\times$.

Lemma 1.2.3.2 (Lagrange-Polynome bilden Basis von $\mathbb{P}_g(\hat{T})$).

Sei $g \geq 0$. Die Lagrange-Polynome aus Definition 1.2.3.1 erfüllen (exemplarisch für $d = 3$)

$$\forall (i, j, k), (\tilde{i}, \tilde{j}, \tilde{k}) \in IJK_g : \quad \hat{\varphi}_{ijk}^g(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) = \delta_{[(i,j,k)=(\tilde{i},\tilde{j},\tilde{k})]}$$

und bilden somit eine Basis von $\mathbb{P}_g(\hat{T})$ (exemplarisch für $d = 3$):

$$\text{span}\{\hat{\varphi}_{ijk}^g \mid (i, j, k) \in IJK_g\} = \mathbb{P}_g(\hat{T}).$$

Die Dimension von $\mathbb{P}_g(\hat{T})$ ist gegeben durch

$$\begin{aligned} (d=1) \quad \dim \mathbb{P}_g(\hat{T}) &= g+1, \\ (d=2) \quad \dim \mathbb{P}_g(\hat{T}) &= \frac{1}{2} \cdot (g+1) \cdot (g+2), \\ (d=3) \quad \dim \mathbb{P}_g(\hat{T}) &= \frac{1}{6} \cdot (g+1) \cdot (g+2) \cdot (g+3). \end{aligned}$$

Wir geben die Dimensionen für kleines $g \geq 0$ explizit an:

$\dim \mathbb{P}_g(\hat{T})$	$g=0$	$g=1$	$g=2$	$g=3$	$g=4$
$d=1$	1	2	3	4	5
$d=2$	1	3	6	10	15
$d=3$	1	4	10	20	35

Beweis:

Wir zeigen exemplarisch nur den Fall $[d=3, g \geq 1]$:

Schritt 1: $\forall (i, j, k), (\tilde{i}, \tilde{j}, \tilde{k}) \in IJK_g : \hat{\varphi}_{ijk}^g(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) = \delta_{[(i,j,k)=(\tilde{i},\tilde{j},\tilde{k})]}$

Die Identität $\forall (i, j, k) \in IJK_g : \hat{\varphi}_{ijk}^g(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) = 1$ ist elementar nachzurechnen. Für beliebige $(\tilde{i}, \tilde{j}, \tilde{k}) \in IJK_g$ gilt:

$$\begin{aligned} [\hat{\varphi}_{ijk}^g(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) = 0] &\Leftrightarrow [(\tilde{i} \in \{0, \dots, i-1\}) \vee (\tilde{j} \in \{0, \dots, j-1\}) \vee (\tilde{k} \in \{0, \dots, k-1\}) \\ &\quad \vee (\tilde{i} + \tilde{j} + \tilde{k} \in \{i+j+k+1, \dots, g\})] \\ &\Leftrightarrow [(\tilde{i}, \tilde{j}, \tilde{k}) \neq (i, j, k)]. \end{aligned}$$

□

Schritt 2: $\text{span}\{\hat{\varphi}_{ijk}^g \mid (i, j, k) \in IJK_g\} = \mathbb{P}_g(\hat{T}), \quad \dim \mathbb{P}_g(\hat{T}) = \frac{1}{6} \cdot (g+1) \cdot (g+2) \cdot (g+3)$

Wegen Schritt 1 ist $\{\hat{\varphi}_{ijk}^g \mid (i, j, k) \in IJK_g\} \subseteq \mathbb{P}_g(\hat{T})$ linear unabhängig. Aus Dimensions-Gründen muss diese Menge auch ein Erzeugenden-System von $\mathbb{P}_g(\hat{T})$ sein (Beachte: Die Monome $\{x^i y^j z^k \mid (i, j, k) \in IJK_g\} \subseteq \mathbb{P}_g(\hat{T})$ bilden eine Basis.):

$$\dim \mathbb{P}_g(\hat{T}) = \dim \text{span} \left\{ \sum_{i=0}^g \sum_{j=0}^{g-i} \sum_{k=0}^{g-i-j} p_{ijk} \cdot x^i y^j z^k \mid p_{ijk} \in \mathbb{R} \right\} = \sum_{i=0}^g \sum_{j=0}^{g-i} \sum_{k=0}^{g-i-j} 1 = \#(IJK_g).$$

Die Dimensions-Formel $\dim \mathbb{P}_g(\hat{T}) = \frac{1}{6} \cdot (g+1) \cdot (g+2) \cdot (g+3)$ erhält man durch explizites Berechnen dieser Summe.

□

■

Lemma 1.2.3.3 (Lagrange-Polynome Koordinaten-Abbildungen).

Sei $g \geq 0$. Die Lagrange-Polynome aus Definition 1.2.3.1 erfüllen (exemplarisch für $d=3$)

$$\forall v \in \mathbb{P}_g(\hat{T}) : \quad v = \sum_{(i,j,k) \in IJK_g} v(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \cdot \hat{\varphi}_{ijk}^g.$$

Insbesondere ist die Koordinaten-Abbildung der Basis der Lagrange-Polynome gegeben durch (exemplarisch für $d=3$)

$$\hat{\chi}_g : \begin{cases} \mathbb{P}_g(\hat{T}) & \longrightarrow \mathbb{R}^{1/6(g+1)(g+2)(g+3)} \\ v & \longmapsto (v(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g))_{(i,j,k) \in IJK_g} \end{cases}.$$

Es sind $\hat{\chi}_g$ und ihre Inverse $\hat{\chi}_g^{-1}$ linear und stetig mit^a

$$\|\hat{\chi}_g\|_{l^2 \leftarrow L^2} \leq C(d, \hat{T}, g), \quad \|\hat{\chi}_g^{-1}\|_{L^2 \leftarrow l^2} \leq C(d, \hat{T}, g).$$

^aFür unsere Wahl der Basis-Elemente $\hat{\varphi}_{ijk}^g$ hängt die Konstante $C(d, \hat{T}, g)$ auf der rechten Seite im Allgemeinen auch tatsächlich von g ab. Wären die $\hat{\varphi}_{ijk}^g$ hingegen eine $\langle \cdot, \cdot \rangle_{L^2(\hat{T})}$ -ONB von $\mathbb{P}_g(\hat{T})$, dann würde die zugehörige Koordinaten-Abbildung $\hat{\chi}_g$ wegen $[\forall v \in \mathbb{P}_g(\hat{T}) : \|v\|_{L^2(\hat{T})}^2 = \sum_{(i,j,k)} \langle v, \hat{\varphi}_{ijk}^g \rangle_{L^2(\hat{T})}^2 = \|\hat{\chi}_g v\|_{l^2}^2]$ sogar $\|\hat{\chi}_g\|_{l^2 \leftarrow L^2} = \|\hat{\chi}_g^{-1}\|_{L^2 \leftarrow l^2} = 1$ erfüllen. Wie wir sehen werden, würde sich das günstig auf die Konditionszahlen der später zu lösenden LGSe auswirken (vgl. Satz 1.2.8.1, Satz 1.2.10.2, Lemma 2.2.5.1, Lemma 2.3.5.1 und Lemma 2.4.5.1).

Beweis:

Wir zeigen exemplarisch nur den Fall $d = 3$:

Für jedes $v \in \mathbb{P}_g(\hat{T})$ erfüllen die eindeutigen Koeffizienten $p_{ijk} \in \mathbb{R}$ aus der Darstellung $v = \sum_{(i,j,k) \in IJK_g} p_{ijk} \cdot \hat{\varphi}_{ijk}^g$ bereits

$$\forall (\tilde{i}, \tilde{j}, \tilde{k}) \in IJK_g : \quad v(\hat{x}_{\tilde{i}}^g, \hat{y}_{\tilde{j}}^g, \hat{z}_{\tilde{k}}^g) = \sum_{(i,j,k) \in IJK_g} p_{ijk} \cdot \underbrace{\hat{\varphi}_{ijk}^g(\hat{x}_{\tilde{i}}^g, \hat{y}_{\tilde{j}}^g, \hat{z}_{\tilde{k}}^g)}_{=\delta_{[(i,j,k)=(\tilde{i},\tilde{j},\tilde{k})]}} \stackrel{\text{L.1.2.3.2}}{=} p_{\tilde{i}\tilde{j}\tilde{k}}.$$

Insbesondere ist die angegebene Funktion $\hat{\chi}_g : \mathbb{P}_g(\hat{T}) \longrightarrow \mathbb{R}^{1/6(g+1)(g+2)(g+3)}$ tatsächlich die Koordinaten-Abbildung der Basis der Lagrange-Polynome.

Die Abbildungen $\|\cdot\|_{L^2(\hat{T})}$ und $\|\hat{\chi}_g(\cdot)\|_{l^2}$ definieren Normen auf dem endlich-dimensionalen Raum $\mathbb{P}_g(\hat{T})$ und sind damit äquivalent:

$$C_1(d, \hat{T}, g) \cdot \|\cdot\|_{L^2(\hat{T})} \leq \|\hat{\chi}_g(\cdot)\|_{l^2} \leq C_2(d, \hat{T}, g) \cdot \|\cdot\|_{L^2(\hat{T})} \quad \text{auf } \mathbb{P}_g(\hat{T}).$$

Daher $\|\hat{\chi}_g\|_{l^2 \leftarrow L^2} \leq C_2(d, \hat{T}, g)$ und $\|\hat{\chi}_g^{-1}\|_{L^2 \leftarrow l^2} \leq C_1(d, \hat{T}, g)$. ■

Lemma 1.2.3.4 (Deny-Lions).

Seien $d \geq 1$, $\hat{T} \subseteq \mathbb{R}^d$ Lipschitz'sch und $l \geq 1$. Dann gilt:

$$\forall v \in H^l(\hat{T}) : \quad \inf_{p \in \mathbb{P}_{l-1}(\hat{T})} \|v - p\|_{H^l(\hat{T})} \leq C(d, \hat{T}, l) \cdot |v|_{H^l(\hat{T})}.$$

Beweis:

Schritt 1: $\forall v \in H^l(\hat{T}) : \quad \|v\|_{H^l(\hat{T})} \leq C(d, \hat{T}, l) \cdot [\|\hat{P}_{l-1} v\|_{L^2(\hat{T})} + |v|_{H^l(\hat{T})}]$

Es bezeichne $\hat{P}_{l-1} : L^2(\hat{T}) \longrightarrow \mathbb{P}_{l-1}(\hat{T})$ die $\langle \cdot, \cdot \rangle_{L^2(\hat{T})}$ -Orthogonal-Projektion.

Wir führen einen Widerspruchs-Beweis: Angenommen es gäbe eine Folge $(v_n)_{n \in \mathbb{N}} \subseteq H^l(\hat{T})$ mit $\|v_n\|_{H^l(\hat{T})} = 1$ und

$$\frac{1}{n} = \frac{1}{n} \cdot \|v_n\|_{H^l(\hat{T})} \geq \|\hat{P}_{l-1} v_n\|_{L^2(\hat{T})} + |v_n|_{H^l(\hat{T})}.$$

Wegen der kompakten Einbettung $H^l(\hat{T}) \hookrightarrow_c H^{l-1}(\hat{T})$ gibt es eine in $H^{l-1}(\hat{T})$ konvergente Teilfolge. Diese ist sogar eine $H^l(\hat{T})$ -Cauchy-Folge, denn

$$\|v_{n(k)} - v_{n(j)}\|_{H^l(\hat{T})}^2 = \|v_{n(k)} - v_{n(j)}\|_{H^{l-1}(\hat{T})}^2 + |v_{n(k)} - v_{n(j)}|_{H^l(\hat{T})}^2 \xrightarrow{(k,j) \rightarrow \infty} 0.$$

Also gibt es ein $v \in H^l(\hat{T})$ mit $v_{n(k)} \xrightarrow[\|\cdot\|_{H^l(\hat{T})}]{k \rightarrow \infty} v$. Wegen

$$|v|_{H^l(\hat{T})} \leq \|v - v_{n(k)}\|_{H^l(\hat{T})} + |v_{n(k)}|_{H^l(\hat{T})} \xrightarrow{k \rightarrow \infty} 0$$

haben wir $v \in \mathbb{P}_{l-1}(\hat{T})$ (vgl. Satz 1.1.2.2). Mit

$$\|v\|_{L^2(\hat{T})} = \|\hat{P}_{l-1}v\|_{L^2(\hat{T})} \leq \|\hat{P}_{l-1}\| \cdot \|v - v_{n(k)}\|_{L^2(\hat{T})} + \|\hat{P}_{l-1}v_{n(k)}\|_{L^2(\hat{T})} \xrightarrow{k \rightarrow \infty} 0$$

folgt $v = 0$ und daher der Widerspruch

$$1 = \lim_{k \rightarrow \infty} \|v_{n(k)}\|_{H^l(\hat{T})} = \|v\|_{H^l(\hat{T})} = 0.$$

□

Schritt 2: $\forall v \in H^l(\hat{T}) : \quad \inf_{p \in \mathbb{P}_{l-1}(\hat{T})} \|v - p\|_{H^l(\hat{T})} \leq C(d, \hat{T}, l) \cdot |v|_{H^l(\hat{T})}.$

$$\begin{aligned} \forall v \in H^l(\hat{T}) : \quad \inf_{p \in \mathbb{P}_{l-1}(\hat{T})} \|v - p\|_{H^l(\hat{T})} &\leq \|v - \hat{P}_{l-1}v\|_{H^l(\hat{T})} \\ &\stackrel{\text{S.1}}{\leq} C(d, \hat{T}, l) \cdot [\|\hat{P}_{l-1}(v - \hat{P}_{l-1}v)\|_{L^2(\hat{T})} + |v - \hat{P}_{l-1}v|_{H^l(\hat{T})}] \\ &= C(d, \hat{T}, l) \cdot |v|_{H^l(\hat{T})}. \end{aligned}$$

□

■

Lemma 1.2.3.5 (Orthogonal-Projektion auf $\mathbb{P}_g(\hat{T})$).

Sei $g \geq 0$. Die $\langle \cdot, \cdot \rangle_{H^0(\hat{T})}$ -Orthogonal-Projektion^a

$$\hat{P}_g : H^0(\hat{T}) \longrightarrow \mathbb{P}_g(\hat{T}).$$

erfüllt die folgenden Eigenschaften:

-) **Stetigkeit:** $\|\hat{P}_g\|_{H^0 \leftarrow H^0} \leq 1.$
-) **Fehler-Abschätzung:** Seien $k \geq 1$ und $l := \min\{k, g+1\} \geq 1$ gegeben. Dann gilt die Fehler-Abschätzung

$$\forall v \in H^k(\hat{T}) : \quad \|v - \hat{P}_g v\|_{H^0(\hat{T})} \leq C(d, \hat{T}, k, g) \cdot |v|_{H^l(\hat{T})}.$$

^aWir schreiben $H^0(\hat{T})$ anstelle von $L^2(\hat{T})$, um die Aussagen besser mit jenen in Lemma 1.2.3.7 vergleichen zu können.

Beweis:

Es gilt:

$$\begin{aligned} \forall v \in H^k(\hat{T}) : \quad \|v - \hat{P}_g v\|_{H^0(\hat{T})} &\stackrel{\text{Def. } \hat{P}_g}{=} \inf_{p \in \mathbb{P}_g(\hat{T})} \|v - p\|_{H^0(\hat{T})} \\ &\stackrel{0 \leq l \leq g+1}{\leq} \inf_{p \in \mathbb{P}_{l-1}(\hat{T})} \|v - p\|_{H^l(\hat{T})} \\ &\stackrel{\text{L.1.2.3.4}}{\leq} C(d, \hat{T}, k, g) \cdot |v|_{H^l(\hat{T})}. \end{aligned}$$

■

Lemma 1.2.3.6 (Referenz-Stützstellen auf $\partial\hat{T}$).

Für $g \geq 1$ wurden die Referenz-Stützstellen auf $\partial\hat{T}$ in Definition 1.2.3.1 so gewählt, dass die folgenden Implikationen gelten:

$$\begin{aligned} (d=2) \quad & \forall E \in \mathcal{E}(\hat{T}) : \quad [\forall v \in \mathbb{P}_g(\hat{T}) \text{ mit } (\forall(\hat{x}_i^g, \hat{y}_j^g) \in \overline{E} : v(\hat{x}_i^g, \hat{y}_j^g) = 0) : \quad v|_{\overline{E}} \equiv 0], \\ (d=3) \quad & \forall F \in \mathcal{F}(\hat{T}) : \quad [\forall v \in \mathbb{P}_g(\hat{T}) \text{ mit } (\forall(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \in \overline{F} : v(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) = 0) : \quad v|_{\overline{F}} \equiv 0]. \end{aligned}$$

Beweis:

Wir zeigen exemplarisch nur den Fall $d = 3$:

Eine der Seitenflächen des Referenz-Tetraeders $\hat{T}_3 \subseteq \mathbb{R}^3$ ist gegeben durch $F := \hat{T}_2 \times \{0\} \in \mathcal{F}(\hat{T}_3)$, wobei $\hat{T}_2 \subseteq \mathbb{R}^2$ das Referenz-Dreieck aus dem Fall $d = 2$ bezeichnet. Wir bezeichnen weiters mit $(\hat{x}_i^{g,3}, \hat{y}_j^{g,3}, \hat{z}_k^{g,3}) \in \hat{T}_3$ die Referenz-Stützstellen für den Fall $d = 3$ und mit $(\hat{x}_i^{g,2}, \hat{y}_j^{g,2}) \in \hat{T}_2$ jene für $d = 2$. Laut Konstruktion gilt

$$\forall(i, j) \in IJ_g : \quad (\hat{x}_i^{g,3}, \hat{y}_j^{g,3}, 0) = (\hat{x}_i^{g,2}, \hat{y}_j^{g,2}, 0).$$

Wir zeigen die Aussage nur für die obige Seitenfläche F : Sei also $v \in \mathbb{P}_g(\hat{T}_3)$ ein Polynom mit

$$\forall(\hat{x}_i^{g,3}, \hat{y}_j^{g,3}, 0) \in \overline{F} = \overline{\hat{T}_2} \times \{0\} : \quad v(\hat{x}_i^{g,3}, \hat{y}_j^{g,3}, 0) = 0.$$

Dann erfüllt das Polynom $v|_{\overline{F}} \in \mathbb{P}_g(\hat{T}_2)$ aber bereits

$$v|_{\overline{F}} \stackrel{\text{L.1.2.3.3}}{=} \sum_{(i,j) \in IJ_g} (v|_{\overline{F}})(\hat{x}_i^{g,2}, \hat{y}_j^{g,2}) \cdot \hat{\varphi}_{ij}^g = \sum_{(i,j) \in IJ_g} \underbrace{v(\hat{x}_i^{g,3}, \hat{y}_j^{g,3}, 0)}_{=0} \cdot \hat{\varphi}_{ij}^g \equiv 0.$$

■

Lemma 1.2.3.7 (Interpolations-Operator auf $\mathbb{P}_g(\hat{T})$).

Sei $g \geq 1$. Der lineare *Interpolations-Operator* (exemplarisch für $d = 3$)

$$\hat{I}_g : \begin{cases} C^0(\overline{\hat{T}}) & \longrightarrow \\ v & \longmapsto \sum_{(i,j,k) \in IJK_g} \mathbb{P}_g(\hat{T}) \\ & v(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \cdot \hat{\varphi}_{ijk}^g \end{cases}$$

erfüllt die folgenden Eigenschaften:

•) **Stetigkeit:** $\|\hat{I}_g\|_{C^0 \leftarrow C^0} \leq C(d, \hat{T}, g)$.

•) **Interpolations-Eigenschaft:**

$$\forall v \in C^0(\overline{\hat{T}}) : \forall(i, j, k) \in IJK_g : \quad (\hat{I}_g v)(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) = v(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g).$$

•) **Projektions-Eigenschaft:** $\hat{I}_g|_{\mathbb{P}_g(\hat{T})} = \text{id}$.

•) **Erhaltung der Randwerte:** Es gelten die folgenden Implikationen^a:

$$\begin{aligned} (d=2) \quad & \forall E \in \mathcal{E}(\hat{T}) : \quad [\forall v \in C^0(\overline{\hat{T}}) \text{ mit } v|_{\overline{E}} \equiv 0 : \quad (\hat{I}_g v)|_{\overline{E}} \equiv 0], \\ (d=3) \quad & \forall F \in \mathcal{F}(\hat{T}) : \quad [\forall v \in C^0(\overline{\hat{T}}) \text{ mit } v|_{\overline{F}} \equiv 0 : \quad (\hat{I}_g v)|_{\overline{F}} \equiv 0]. \end{aligned}$$

•) **Fehler-Abschätzung:** Seien zusätzlich $k > \frac{d}{2}$, $l := \min\{k, g+1\} > \frac{d}{2}$ und $m \in \{0, 1\}$ gegeben. Dann ist $\hat{I}_g|_{H^l(\hat{T})}$ stetig mit

$$\|\hat{I}_g|_{H^l}\|_{H^m \leftarrow H^l} \leq C(d, \hat{T}, k, g, m)$$

und es gilt die Fehler-Abschätzung

$$\forall v \in H^k(\hat{T}) : \quad \|v - \hat{I}_g v\|_{H^m(\hat{T})} \leq C(d, \hat{T}, k, g, m) \cdot |v|_{H^l(\hat{T})}.$$

^aWir werden dieses Resultat nicht für den *Beweis* von Satz 1.2.9.1 benötigen. Wir geben es aber trotzdem an, um die Eigenschaften von $\hat{I}_g$ und $I_{h,g}$ besser vergleichen zu können.

Beweis:

Wir zeigen exemplarisch nur den Fall $d = 3$:

Schritt 1: Stetigkeit, Interpolations-Eigenschaft, Projektions-Eigenschaft

Der Interpolations-Operator $\hat{I}_g$ ist linear und stetig:

$$\forall v \in C^0(\bar{\hat{T}}) : \quad \|\hat{I}_g v\|_{C^0(\bar{\hat{T}})} \leq \sup_{(x,y,z) \in \bar{\hat{T}}} \sum_{(i,j,k) \in IJK_g} \underbrace{|v(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)|}_{\leq \|v\|_{C^0(\bar{\hat{T}})}} \cdot \underbrace{|\hat{\varphi}_{ijk}^g(x, y, z)|}_{\leq C(d, \hat{T}, g)} \leq C(d, \hat{T}, g) \cdot \|v\|_{C^0(\bar{\hat{T}})}.$$

Die Interpolations-Eigenschaft ergibt sich aus

$$\begin{aligned} \forall v \in C^0(\bar{\hat{T}}) : \forall (\tilde{i}, \tilde{j}, \tilde{k}) \in IJK_g : \quad (\hat{I}_g v)(\hat{x}_{\tilde{i}}^g, \hat{y}_{\tilde{j}}^g, \hat{z}_{\tilde{k}}^g) &= \sum_{(i,j,k) \in IJK_g} v(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \cdot \underbrace{\hat{\varphi}_{ijk}^g(\hat{x}_{\tilde{i}}^g, \hat{y}_{\tilde{j}}^g, \hat{z}_{\tilde{k}}^g)}_{=\delta_{[(i,j,k)=(\tilde{i},\tilde{j},\tilde{k})]}} \\ &\stackrel{\text{L.1.2.3.2}}{=} v(\hat{x}_{\tilde{i}}^g, \hat{y}_{\tilde{j}}^g, \hat{z}_{\tilde{k}}^g). \end{aligned}$$

Die Projektions-Eigenschaft folgt aus

$$\forall v \in \mathbb{P}_g(\hat{T}) : \quad \hat{I}_g v = \sum_{(i,j,k) \in IJK_g} v(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \cdot \hat{\varphi}_{ijk}^g \stackrel{\text{L.1.2.3.3}}{=} v.$$

Seien nun $F \in \mathcal{F}(\hat{T})$ und $v \in C^0(\bar{\hat{T}})$ mit $v|_F \equiv 0$ gegeben. Dann gilt

$$\forall (\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \in \bar{F} : \quad \underbrace{(\hat{I}_g v)(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)}_{\in \mathbb{P}_g(\hat{T})} = v \underbrace{(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)}_{\in \bar{F}} = 0,$$

also wegen Lemma 1.2.3.6 bereits $(\hat{I}_g v)|_{\bar{F}} \equiv 0$.

□

Schritt 2: Fehler-Abschätzung

Seien nun zusätzlich $k > \frac{d}{2}$, $l := \min\{k, g+1\} > \frac{d}{2}$ und $m \in \{0, 1\}$ gegeben. Wegen $l > \frac{d}{2}$ und Satz 1.1.2.2 gilt die stetige Einbettung $H^l(\hat{T}) \hookrightarrow C^0(\bar{\hat{T}})$:

$$\|\cdot\|_{C^0(\bar{\hat{T}})} \leq C(d, \hat{T}, k, g) \cdot \|\cdot\|_{H^l(\hat{T})} \quad \text{auf } H^l(\hat{T}).$$

Außerdem sind die Normen $\|\cdot\|_{H^m(\hat{T})}$ und $\|\cdot\|_{C^0(\bar{\hat{T}})}$ auf $\mathbb{P}_g(\hat{T})$ äquivalent:

$$\|\cdot\|_{H^m(\hat{T})} \leq C(d, \hat{T}, g, m) \cdot \|\cdot\|_{C^0(\bar{\hat{T}})} \quad \text{auf } \mathbb{P}_g(\hat{T}).$$

Damit

$$\begin{aligned} \forall v \in H^l(\hat{T}) : \quad \|\hat{I}_g v\|_{H^m(\hat{T})} &\leq C(d, \hat{T}, g, m) \cdot \|\hat{I}_g v\|_{C^0(\bar{\hat{T}})} \\ &\leq C(d, \hat{T}, g, m) \cdot \|\hat{I}_g\|_{C^0 \leftarrow C^0} \cdot \|v\|_{C^0(\bar{\hat{T}})} \\ &\stackrel{\text{S.1}}{\leq} C(d, \hat{T}, k, g, m) \cdot \|v\|_{H^l(\hat{T})} \end{aligned}$$

und

$$\begin{aligned}
\forall v \in H^k(\hat{T}) : \quad \|v - \hat{I}_g v\|_{H^m(\hat{T})} &\leq \inf_{p \in \mathbb{P}_g(\hat{T})} \|v - p\|_{H^m(\hat{T})} + \|\hat{I}_g(p - v)\|_{H^m(\hat{T})} \\
&\stackrel{m \leq l \leq k}{\leq} (1 + \|\hat{I}_g\|_{H^l \leftarrow H^m}) \cdot \inf_{p \in \mathbb{P}_g(\hat{T})} \|v - p\|_{H^l(\hat{T})} \\
&\stackrel{l \leq g+1}{\leq} (1 + \|\hat{I}_g\|_{H^l \leftarrow H^m}) \cdot \inf_{p \in \mathbb{P}_{l-1}(\hat{T})} \|v - p\|_{H^l(\hat{T})} \\
&\stackrel{l \geq 1, \text{ L.1.2.3.4}}{\leq} C(d, \hat{T}, k, g, m) \cdot |v|_{H^l(\hat{T})}.
\end{aligned}$$

□

■

Lemma 1.2.3.8 (Inverse Ungleichung auf $\mathbb{P}_g(\hat{T})$).

Seien $d \geq 1$ und $\hat{T} \subseteq \mathbb{R}^d$ Lipschitz'sch. Seien weiters $g \geq 0$ und $0 \leq m \leq l$. Dann gilt die *inverse Ungleichung*

$$\forall v \in \mathbb{P}_g(\hat{T}) : \quad |v|_{H^l(\hat{T})} \leq C(d, \hat{T}, l, g, m) \cdot |v|_{H^m(\hat{T})}.$$

Beweis:

Schritt 1: $(X, \|\cdot\|)$ endlich-dimensional, $|\cdot|$ Semi-Norm $\Rightarrow |\cdot| \leq C(X, \|\cdot\|, |\cdot|) \cdot \|\cdot\|$

Der vom Unterraum $N := \{n \in X \mid |n| = 0\} \leq X$ induzierte Vektorraum $X/N := \{x + N \mid x \in X\}$ ist endlich-dimensional mit $\dim(X/N) \leq \dim X < \infty$. Die Abbildungen

$$\|x + N\| := \inf_{n \in N} \|x + n\|, \quad |x + N| := \inf_{n \in N} |x + n| = |x|$$

sind Normen auf X/N und damit äquivalent:

$$\forall x \in X : \quad |x| = |x + N| \leq C(X, \|\cdot\|, |\cdot|) \cdot \|x + N\| \leq C(X, \|\cdot\|, |\cdot|) \cdot \|x\|.$$

□

Schritt 2: $\forall v \in \mathbb{P}_g(\hat{T}) : \quad |v|_{H^l(\hat{T})} \leq C(d, \hat{T}, l, g, m) \cdot |v|_{H^m(\hat{T})}$

Wir wenden Schritt 1 nun auf den Raum $X := \tilde{\mathbb{P}}_g(\hat{T}) := \{D^\alpha v \mid v \in \mathbb{P}_g(\hat{T}), |\alpha| = m\}$, die Norm $\|\cdot\| := \|\cdot\|_{L^2(\hat{T})}$ und die Semi-Norm $|\cdot| := |\cdot|_{H^{l-m}(\hat{T})}$ an:

$$|\cdot|_{H^{l-m}(\hat{T})} \leq C(d, \hat{T}, l, g, m) \cdot \|\cdot\|_{L^2(\hat{T})} \quad \text{auf } \tilde{\mathbb{P}}_g(\hat{T}).$$

Es folgt

$$\begin{aligned}
\forall v \in \mathbb{P}_g(\hat{T}) : \quad |v|_{H^l(\hat{T})}^2 &\leq \sum_{|\alpha|=m} |D^\alpha v|_{H^{l-m}(\hat{T})}^2 \\
&\leq \sum_{|\alpha|=m} C(d, \hat{T}, l, g, m) \cdot \|D^\alpha v\|_{L^2(\hat{T})}^2 \\
&= C(d, \hat{T}, l, g, m) \cdot |v|_{H^m(\hat{T})}^2.
\end{aligned}$$

□

■

1.2.4 Skalierungs-Argument $\hat{T} \leftrightarrow T$

Lemma 1.2.4.1 (Skalierungs-Argument $\hat{T} \leftrightarrow T$).

Seien $d \geq 1$, $\hat{T}, T \subseteq \mathbb{R}^d$ Lipschitz'sch, $F_T : \hat{T} \rightarrow T$ eine affine Bijektion und $m \geq 0$. Die Sobolev-Seminormen auf T hängen mit jenen auf $\hat{T}$ auf folgende Weise zusammen:

$$\begin{aligned} \forall v \in H^m(T) : \quad |v \circ F_T|_{H^m(\hat{T})} &\leq C(d, m) \cdot \|\nabla F_T\|_2^m \cdot \sqrt{|\det \nabla(F_T^{-1})|} \cdot |v|_{H^m(T)}, \\ |v|_{H^m(T)} &\leq C(d, m) \cdot \|\nabla(F_T^{-1})\|_2^m \cdot \sqrt{|\det \nabla F_T|} \cdot |v \circ F_T|_{H^m(\hat{T})}. \end{aligned}$$

Beweis:

Für $m = 0$ gilt:

$$|v \circ F_T|_{H^0(\hat{T})}^2 = \int_{\hat{T}} |v \circ F_T|^2 \, dV = |\det \nabla(F_T^{-1})| \cdot |v|_{H^0(T)}^2.$$

Für $m \geq 1$ gilt:

$$\begin{aligned} |v \circ F_T|_{H^m(\hat{T})}^2 &\leq \sum_{i_1, \dots, i_m=1}^d \int_{\hat{T}} |\partial_{i_m} \cdots \partial_{i_1} (v \circ F_T)|^2 \, dV \\ &= \sum_{i_1, \dots, i_m=1}^d \int_{\hat{T}} \left| \sum_{l_1, \dots, l_m=1}^d [(\partial_{l_m} \cdots \partial_{l_1} v) \circ F_T] \cdot \prod_{\mu=1}^m (\nabla F_T)_{l_\mu, i_\mu} \right|^2 \, dV \\ &\stackrel{\text{C.S.}}{\leq} \sum_{i_1, \dots, i_m=1}^d \int_{\hat{T}} \left[\sum_{l_1, \dots, l_m=1}^d [(\partial_{l_m} \cdots \partial_{l_1} v) \circ F_T]^2 \right] \cdot \left[\sum_{l_1, \dots, l_m=1}^d \prod_{\mu=1}^m (\nabla F_T)_{l_\mu, i_\mu}^2 \right] \, dV \\ &= \left[\prod_{\mu=1}^m \sum_{i_\mu=1}^d \sum_{l_\mu=1}^d (\nabla F_T)_{l_\mu, i_\mu}^2 \right] \cdot |\det \nabla(F_T^{-1})| \cdot \left[\sum_{l_1, \dots, l_m=1}^d \int_T [\partial_{l_m} \cdots \partial_{l_1} v]^2 \, dV \right] \\ &\leq C(d, m) \cdot \|\nabla F_T\|_2^{2m} \cdot |\det \nabla(F_T^{-1})| \cdot |v|_{H^m(T)}^2. \end{aligned}$$

■

1.2.5 Größenordnung der Referenz-Abbildungen F_T

Lemma 1.2.5.1 (Größenordnung Referenz-Abbildung auf T).

Seien $d \geq 1$, $\hat{T} \subseteq \mathbb{R}^d$ ein Referenz-Element und $(\mathcal{T}_h)_{h \in H}$ eine formreguläre Familie von Gittern. Dann gilt

$$\begin{aligned} \forall h \in H : \forall T \in \mathcal{T}_h : \quad & C_1(d, \hat{T}) \cdot h_T \leq \|\nabla F_T\|_2 \leq C_2(d, \hat{T}) \cdot h_T, \\ & C_3(d, \hat{T}) \cdot \frac{1}{h_T} \leq \|\nabla(F_T^{-1})\|_2 \leq C_1(d, \hat{T}, \gamma) \cdot \frac{1}{h_T}, \\ & C_2(d, \hat{T}, \gamma) \cdot h_T^d \leq |\det \nabla F_T| \leq C_4(d, \hat{T}) \cdot h_T^d, \\ & C_5(d, \hat{T}) \cdot \frac{1}{h_T^d} \leq |\det \nabla(F_T^{-1})| \leq C_3(d, \hat{T}, \gamma) \cdot \frac{1}{h_T^d}. \end{aligned}$$

Beweis:

Schritt 1: Abschätzungen für $\|\nabla F_T\|_2$ und $\|\nabla(F_T^{-1})\|_2$

Sei $T \in \mathcal{T}_h$ gegeben. Bezeichne mit $B_{r_T}(m_{\hat{T}}) \subseteq \hat{T}$ die Inkugel von $\hat{T}$. Aus der Kompaktheit der Einheits-

Kugel in $\mathbb{R}^d$ folgt die Existenz eines $z_0 \in \overline{B}_1(0) \subseteq \mathbb{R}^d$ mit

$$\begin{aligned}
\|\nabla F_T\|_2 &= \max_{\|z\|_2 \leq 1} \|\nabla F_T \cdot z\|_2 \\
&= \|\nabla F_T \cdot z_0\|_2 \\
&= \frac{1}{r_{\hat{T}}} \cdot \|\nabla F_T \cdot (m_{\hat{T}} + \frac{r_{\hat{T}}}{2} z_0) - \nabla F_T \cdot (m_{\hat{T}} - \frac{r_{\hat{T}}}{2} z_0)\|_2 \\
&= \frac{1}{r_{\hat{T}}} \cdot \|F_T(m_{\hat{T}} + \frac{r_{\hat{T}}}{2} z_0) - F_T(m_{\hat{T}} - \frac{r_{\hat{T}}}{2} z_0)\|_2 \\
&\leq \frac{1}{r_{\hat{T}}} \cdot h_T.
\end{aligned}$$

Weiters gilt für zwei geeignete Punkt $x, y \in \overline{\hat{T}}$ sicherlich

$$h_T = \|F_T(x) - F_T(y)\|_2 = \|\nabla F_T \cdot (x - y)\|_2 \leq h_{\hat{T}} \cdot \|\nabla F_T\|_2.$$

Für die Inverse F_T^{-1} ergibt sich durch Vertauschung der Rollen von $\hat{T}$ und T :

$$h_{\hat{T}} \cdot \frac{1}{h_T} \leq \|\nabla(F_T^{-1})\|_2 \leq \frac{1}{r_T} \cdot h_{\hat{T}} \leq \gamma \cdot h_{\hat{T}} \cdot \frac{1}{h_T}.$$

□

Schritt 2: Abschätzungen für $|\det \nabla F_T|$ und $|\det \nabla(F_T^{-1})|$

Bezeichne $B_{r_T}(m_T) \subseteq T$ die Inkugel von T und $B_{h_T}(M) \supseteq T$ eine Kugel mit beliebigem Mittelpunkt $M \in T$. Dann gilt:

$$C(d, \gamma) \cdot h_T^d \leq C(d) \cdot r_T^d \leq |B_{r_T}(m_T)| \leq |T| \leq |B_{h_T}(M)| \leq C(d) \cdot h_T^d.$$

Die behaupteten Abschätzungen für $|\det \nabla F_T|$ und $|\det \nabla(F_T^{-1})|$ ergeben sich nun aus

$$|T| = \int_T 1 \, dV = \int_{\hat{T}} 1 \cdot |\det \nabla F_T| \, dV = |\det \nabla F_T| \cdot |\hat{T}|$$

und $|\det \nabla(F_T^{-1})| = |\det \nabla F_T|^{-1}$.

□

■

1.2.6 Inverse Ungleichung auf $\mathbb{S}^{g,0}(\mathcal{T}_h)$

Lemma 1.2.6.1 (Inverse Ungleichung auf $\mathbb{S}^{g,0}(\mathcal{T}_h)$).

Seien $d \geq 1$, $\hat{T} \subseteq \mathbb{R}^d$ ein Referenz-Element, $\Omega \subseteq \mathbb{R}^d$ ein Polygon und $(\mathcal{T}_h)_{h \in H}$ eine formreguläre Familie von Gittern auf Ω . Seien weiters $g \geq 0$ und $0 \leq m \leq l$. Dann gilt die *inverse Ungleichung*

$$\forall v \in \mathbb{S}^{g,0}(\mathcal{T}_h) : \sum_{T \in \mathcal{T}_h} h_T^{2(l-m)} \cdot |v|_{H^l(T)}^2 \leq C(d, \hat{T}, \gamma, l, g, m) \cdot \sum_{T \in \mathcal{T}_h} |v|_{H^m(T)}^2.$$

Beweis:

Für alle $v \in \mathbb{S}^{g,0}(\mathcal{T}_h)$ gilt:

$$\begin{aligned}
\sum_{T \in \mathcal{T}_h} h_T^{2(l-m)} \cdot |v|_{H^l(T)}^2 &\stackrel{\text{L.1.2.4.1}}{\leq} C(d, l) \cdot \sum_{T \in \mathcal{T}_h} h_T^{2(l-m)} \cdot \|\nabla(F_T^{-1})\|_2^{2l} \cdot |\det \nabla F_T| \cdot |v \circ F_T|_{H^l(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.3.8}}{\leq} C(d, \hat{T}, l, g, m) \cdot \sum_{T \in \mathcal{T}_h} h_T^{2(l-m)} \cdot \|\nabla(F_T^{-1})\|_2^{2l} \cdot |\det \nabla F_T| \cdot |v \circ F_T|_{H^m(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.4.1}}{\lesssim} C(d, \hat{T}, l, g, m) \cdot \sum_{T \in \mathcal{T}_h} h_T^{2(l-m)} \cdot \|\nabla F_T\|_2^{2m} \cdot \|\nabla(F_T^{-1})\|_2^{2l} \cdot |v|_{H^m(T)}^2 \\
&\stackrel{\text{L.1.2.5.1}}{\leq} C(d, \hat{T}, \gamma, l, g, m) \cdot \sum_{T \in \mathcal{T}_h} |v|_{H^m(T)}^2.
\end{aligned}$$

■

1.2.7 Approximations-Eigenschaften von $\mathbb{S}^{g,0}(\mathcal{T}_h)$ für $g \geq 0$

Satz 1.2.7.1 (Approximations-Operator nach $\mathbb{S}^{g,0}(\mathcal{T}_h)$ für $g \geq 0$).

Seien $d \in \{1, 2, 3\}$, $\hat{T} \subseteq \mathbb{R}^d$ das Referenz-Element für Intervall-, Dreiecks- bzw. Tetraeder-Gitter, $\Omega \subseteq \mathbb{R}^d$ ein Polygon und $(\mathcal{T}_h)_{h \in H}$ eine Familie von Gittern auf Ω . Sei weiters $g \geq 0$ gegeben. Wir bezeichnen wieder mit $\hat{P}_g : H^0(\hat{T}) \rightarrow \mathbb{P}_g(\hat{T})$ die $\langle \cdot, \cdot \rangle_{H^0(\hat{T})}$ -Orthogonal-Projektion aus Lemma 1.2.3.5.

Dann erfüllt für jedes $h \in H$ der Operator^a

$$P_{h,g} : \begin{cases} (H^0(\Omega), \|\cdot\|_{H^0(\Omega)}) & \longrightarrow (\mathbb{S}^{g,0}(\mathcal{T}_h), \|\cdot\|_{H^0(\Omega)}) \\ v & \longmapsto \sum_{T \in \mathcal{T}_h} [\hat{P}_g(v \circ F_T) \circ F_T^{-1}] \cdot \mathbb{I}_T \end{cases}$$

die folgenden Eigenschaften:

- **Orthogonal-Projektion:** Es ist $P_{h,g}$ genau die $\langle \cdot, \cdot \rangle_{H^0(\Omega)}$ -Orthogonal-Projektion auf $\mathbb{S}^{g,0}(\mathcal{T}_h) \leq H^0(\Omega)$.
- **Fehler-Abschätzung:** Für alle $k \geq 1$ gilt mit $l := \min\{k, g+1\} \geq 1$ die Fehler-Abschätzung

$$\forall v \in H^k(\Omega) : \quad \|v - P_{h,g}v\|_{H^0(\Omega)} \leq C(d, \hat{T}, k, g) \cdot h_{\mathcal{T}_h}^{l-0} \cdot |v|_{H^l(\Omega)}.$$

^aWir schreiben wieder $H^0(\Omega)$ anstelle von $L^2(\Omega)$, um die Aussagen besser mit jenen von Satz 1.2.9.1 vergleichen zu können.

Beweis:

Schritt 1: $P_{h,g}$ ist die Orthogonal-Projektion auf $\mathbb{S}^{g,0}(\mathcal{T}_h)$

Der Operator $P_{h,g}$ ist wohldefiniert, linear und stetig mit

$$\begin{aligned}
\forall v \in H^0(\Omega) : \quad \|P_{h,g}v\|_{H^0(\Omega)}^2 &= \sum_{T \in \mathcal{T}_h} \|\hat{P}_g(v \circ F_T) \circ F_T^{-1}\|_{H^0(T)}^2 \\
&\stackrel{\text{L.1.2.4.1}}{=} \sum_{T \in \mathcal{T}_h} |\det \nabla F_T| \cdot \|\hat{P}_g(v \circ F_T)\|_{H^0(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.3.5}}{\leq} \sum_{T \in \mathcal{T}_h} |\det \nabla F_T| \cdot 1 \cdot \|v \circ F_T\|_{H^0(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.4.1}}{=} \sum_{T \in \mathcal{T}_h} \|v\|_{H^0(T)}^2 \\
&= \|v\|_{H^0(\Omega)}^2.
\end{aligned}$$

Weiters gilt

$$\begin{aligned}
\forall v \in H^0(\Omega) : \forall w \in \mathbb{S}^{g,0}(\mathcal{T}_h) : \\
\|v - P_{h,g}v\|_{H^0(\Omega)}^2 &= \sum_{T \in \mathcal{T}_h} \|v - \hat{P}_g(v \circ F_T) \circ F_T^{-1}\|_{H^0(T)}^2 \\
&\stackrel{\text{L.1.2.4.1}}{=} \sum_{T \in \mathcal{T}_h} |\det \nabla F_T| \cdot \|v \circ F_T - \hat{P}_g(v \circ F_T)\|_{H^0(\hat{T})}^2 \\
&\stackrel{\text{Def. } \hat{P}_g}{\leq} \sum_{T \in \mathcal{T}_h} |\det \nabla F_T| \cdot \|v \circ F_T - w \circ F_T\|_{H^0(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.4.1}}{=} \sum_{T \in \mathcal{T}_h} \|v - w\|_{H^0(T)}^2 \\
&= \|v - w\|_{H^0(\Omega)}^2.
\end{aligned}$$

Daher folgt für die $\langle \cdot, \cdot \rangle_{H^0(\Omega)}$ -Orthogonal-Projektion $\tilde{P}_{h,g} : H^0(\Omega) \rightarrow \mathbb{S}^{g,0}(\mathcal{T}_h)$ bereits

$$\forall v \in H^0(\Omega) : \quad \|v - \tilde{P}_{h,g}v\|_{H^0(\Omega)} \stackrel{\text{Def. } \tilde{P}_{h,g}}{\leq} \|v - \underbrace{P_{h,g}v}_{\in \mathbb{S}^{g,0}}\|_{H^0(\Omega)} \leq \|v - \underbrace{\tilde{P}_{h,g}v}_{\in \mathbb{S}^{g,0}}\|_{H^0(\Omega)},$$

wodurch $[\forall v \in H^0(\Omega) : P_{h,g}v = \tilde{P}_{h,g}v]$.

□

Schritt 2: Fehler-Abschätzung

Mit der Fehler-Abschätzung für $\hat{P}_g$ aus Lemma 1.2.3.5 ergibt sich:

$$\begin{aligned}
\forall v \in H^k(\Omega) : \quad \|v - P_{h,g}v\|_{H^0(\Omega)}^2 &= \sum_{T \in \mathcal{T}_h} \|v - \hat{P}_g(v \circ F_T) \circ F_T^{-1}\|_{H^0(T)}^2 \\
&\stackrel{\text{L.1.2.4.1}}{=} \sum_{T \in \mathcal{T}_h} |\det \nabla F_T| \cdot \|\underbrace{v \circ F_T}_{\in H^k(\hat{T})} - \hat{P}_g(v \circ F_T)\|_{H^0(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.3.5}}{\leq} C(d, \hat{T}, k, g) \cdot \sum_{T \in \mathcal{T}_h} |\det \nabla F_T| \cdot |v \circ F_T|_{H^1(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.4.1}}{\lesssim} C(d, \hat{T}, k, g) \cdot \sum_{T \in \mathcal{T}_h} \|\nabla F_T\|_2^{2l} \cdot |v|_{H^1(T)}^2 \\
&\stackrel{\text{S.1.2.5.1}}{\lesssim} C(d, \hat{T}, k, g) \cdot \sum_{T \in \mathcal{T}_h} h_T^{2l} \cdot |v|_{H^1(T)}^2 \\
&\leq C(d, \hat{T}, k, g) \cdot h_{\mathcal{T}_h}^{2l} \cdot |v|_{H^1(\Omega)}^2.
\end{aligned}$$

□

■

Korollar 1.2.7.2 (Bramble-Hilbert für $\mathbb{S}^{g,0}(\mathcal{T}_h)$ mit $g \geq 0$).

Seien $d \in \{1, 2, 3\}$, $\hat{T} \subseteq \mathbb{R}^d$ das Referenz-Element für Intervall-, Dreiecks- bzw. Tetraeder-Gitter, $\Omega \subseteq \mathbb{R}^d$ ein Polygon und $(\mathcal{T}_h)_{h \in H}$ eine Familie von Gittern auf Ω . Seien weiters $k \geq 1$, $g \geq 0$ und $l := \min\{k, g+1\} \geq 1$. Dann gilt ^{ab}:

$$\forall v \in H^k(\Omega) : \quad \inf_{w \in \mathbb{S}^{g,0}(\mathcal{T}_h)} \|v - w\|_{H^0(\Omega)} \leq C(d, \hat{T}, k, g) \cdot h_{\mathcal{T}_h}^{l-0} \cdot |v|_{H^1(\Omega)}.$$

Ist zusätzlich $(\mathcal{T}_h)_{h \in H}$ formregulär und quasi-uniform sowie $g+1 \leq k$, so ist die Konvergenz-Ordnung

$\mathcal{O}(h_{\mathcal{T}_h}^{l-0})$ (bezüglich $h \rightarrow 0$) maximal^c.

^aWir schreiben wieder $H^0(\Omega)$ anstelle von $L^2(\Omega)$, um die Aussagen besser mit jenen von Korollar 1.2.9.2 vergleichen zu können.

^bMan beachte: Wegen $l = \min \{k, g + 1\}$ liefert eine Erhöhung des Polynom-Grades g nur dann höhere Konvergenz-Geschwindigkeit, wenn die zu approximierende Funktion hinreichend glatt ist. Die Raum-Dimension d hingegen spielt für die Konvergenz-Geschwindigkeit keine Rolle.

^cDa diese Konvergenz-Ordnung mit Hilfe des Operators $P_{h,g} : H^0(\Omega) \rightarrow \mathbb{S}^{g,0}(\mathcal{T}_h)$ aus Satz 1.2.7.1 gezeigt wird, muss sie *nicht* zwingend die größtmögliche sein. A priori könnte es ja auch einen weiteren Operator $\tilde{P}_{h,g} : H^0(\Omega) \rightarrow \mathbb{S}^{g,0}(\mathcal{T}_h)$ geben, der auf eine noch höhere Konvergenz-Ordnung führt. Wie wir im Beweis sehen werden, ist dies jedoch nicht der Fall.

Beweis:

Schritt 1: Fehler-Abschätzung

Mit dem Operator $P_{h,g} : H^0(\Omega) \rightarrow \mathbb{S}^{g,0}(\mathcal{T}_h)$ aus Satz 1.2.7.1 ergibt sich

$$\forall v \in H^k(\Omega) : \quad \inf_{w \in \mathbb{S}^{g,0}(\mathcal{T}_h)} \|v - w\|_{H^0(\Omega)} = \|v - P_{h,g}v\|_{H^0(\Omega)} \leq C(d, \hat{T}, k, g) \cdot h_{\mathcal{T}_h}^{l-0} \cdot |v|_{H^l(\Omega)}.$$

□

Schritt 2: Optimalität der Konvergenz-Ordnung

Es bezeichne $\alpha \in \mathbb{R}$ den maximalen Exponenten mit

$$\forall v \in H^k(\Omega) : \quad \inf_{w \in \mathbb{S}^{g,0}(\mathcal{T}_h)} \|v - w\|_{H^0(\Omega)} \leq C(d, \hat{T}, k, g) \cdot h_{\mathcal{T}_h}^\alpha \cdot |v|_{H^l(\Omega)}.$$

Insbesondere gilt $\alpha \geq l - 0$.

Für jedes feste $\tilde{g} \geq 0$ gilt²:

$$\begin{aligned} \forall v \in \mathbb{S}^{\tilde{g},k}(\mathcal{T}_h) : \quad & \frac{1}{\nu^{2(l-m)}} \cdot h_{\mathcal{T}_h}^{2(l-0)} \cdot |v|_{H^l(\Omega)}^2 \leq \sum_{T \in \mathcal{T}_h} h_T^{2(l-0)} \cdot |v|_{H^l(T)}^2 \\ & \stackrel{l \geq g}{=} \sum_{T \in \mathcal{T}_h} h_T^{2(l-0)} \cdot \underbrace{|v - \underbrace{P_{h,g}v}_{\in \mathbb{S}^{\tilde{g},0}(\mathcal{T}_h)}|_{H^l(T)}^2}_{\in \mathbb{P}_g(T)} \\ & \stackrel{0 \leq l, \text{ L.1.2.6.1}}{\leq} C(d, \hat{T}, \gamma, k, \tilde{g}) \cdot \sum_{T \in \mathcal{T}_h} \|v - P_{h,g}v\|_{H^0(T)}^2 \\ & = C(d, \hat{T}, \gamma, k, \tilde{g}) \cdot \|v - P_{h,g}v\|_{H^0(\Omega)}^2 \\ & \stackrel{\text{S.1.2.7.1}}{=} C(d, \hat{T}, \gamma, k, \tilde{g}) \cdot \inf_{w \in \mathbb{S}^{g,0}(\mathcal{T}_h)} \|v - w\|_{H^0(\Omega)}^2 \\ & \stackrel{\text{Def. } \alpha}{\lesssim} C(d, \hat{T}, \gamma, k, \tilde{g}) \cdot h_{\mathcal{T}_h}^{2\alpha} \cdot |v|_{H^l(\Omega)}^2. \end{aligned}$$

Wir dürfen das Polynom

$$v(x) := x_1^l \in \mathbb{P}_l(\Omega) \subseteq \mathbb{S}^{l,k}(\mathcal{T}_h)$$

einsetzen und erhalten unter Beachtung von $|v|_{H^l(\Omega)}^2 = (l!)^2 \cdot |\Omega| \neq 0$ und $\lim_{h \rightarrow 0} h_{\mathcal{T}_h} = 0$ unmittelbar $\alpha \leq l - 0$.

□

■

²Die Ungleichungen sind eigentlich nur für alle $\tilde{g} \geq g + 1$ interessant, da sonst wegen $l > g \geq \tilde{g}$ bereits $[\forall v \in \mathbb{S}^{\tilde{g},k}(\mathcal{T}_h) : |v|_{H^l(\Omega)} = 0]$ gilt.

1.2.8 Basis von $\mathbb{S}^{g,0}(\mathcal{T}_h)$ für $g \geq 0$

Satz 1.2.8.1 (Basis von $\mathbb{S}^{g,0}(\mathcal{T}_h)$ für $g \geq 0$).

Seien $d \in \{1, 2, 3\}$, $\hat{T} \subseteq \mathbb{R}^d$ das Referenz-Element für Intervall-, Dreiecks- bzw. Tetraeder-Gitter, $\Omega \subseteq \mathbb{R}^d$ ein Polygon und $(\mathcal{T}_h)_{h \in H}$ eine Familie von Gittern auf Ω . Wir bezeichnen wieder für festes $g \geq 0$ mit $\{\hat{x}_i^g \mid i \in I_g\}$, $\{(\hat{x}_i^g, \hat{y}_j^g) \mid (i, j) \in IJ_g\}$ bzw. $\{(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \mid (i, j, k) \in IJK_g\}$ die Referenz-Stützstellen und mit $\{\hat{\varphi}_i^g \mid i \in I_g\}$, $\{\hat{\varphi}_{ij}^g \mid (i, j) \in IJ_g\}$ bzw. $\{\hat{\varphi}_{ijk}^g \mid (i, j, k) \in IJK_g\}$ die Referenz-Formfunktionen aus Definition 1.2.3.1.

- **Definition der Basis-Elemente:** Wir können die Basis-Elemente von $\mathbb{S}^{g,0}(\mathcal{T}_h)$ explizit anschreiben (exemplarisch für $d = 3$):

$$\forall T \in \mathcal{T}_h : \forall (i, j, k) \in IJK_g : \quad \varphi_{T,ijk}^g := [\hat{\varphi}_{ijk}^g \circ F_T^{-1}] \cdot \mathbb{I}_T \in \mathbb{S}^{g,0}(\mathcal{T}_h).$$

- **Beschreibung der Basis:** Wir geben nun eine Basis von $\mathbb{S}^{g,0}(\mathcal{T}_h)$ an:

$$\mathbb{S}^{g,0}(\mathcal{T}_h) = \text{span}\{\varphi_{T,ijk}^g \mid T \in \mathcal{T}_h, (i, j, k) \in IJK_g\}.$$

- **Koordinaten-Abbildung:** Wir definieren die *Element-Stützstellen* (exemplarisch für $d = 3$)

$$\forall T \in \mathcal{T}_h : \forall (i, j, k) \in IJK_g : \quad N_{ijk}^g(T) := F_T(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \in \bar{T}.$$

Für jedes $v \in \mathbb{S}^{g,0}(\mathcal{T}_h)$ stimmen die Koeffizienten bezüglich der obigen Basis mit den Funktionswerten (bzw. einseitigen Grenzwerten) an den Element-Stützstellen überein:

$$\forall v \in \mathbb{S}^{g,0}(\mathcal{T}_h) : \quad v = \sum_{\substack{T \in \mathcal{T}_h, \\ (i,j,k) \in IJK_g}} [(v|_T)(N_{ijk}^g(T))] \cdot \varphi_{T,ijk}^g.$$

Insbesondere ist die Koordinaten-Abbildung bezüglich der obigen Basis gegeben durch (exemplarisch für $d = 3$)

$$\chi_{h,g} : \begin{cases} \mathbb{S}^{g,0}(\mathcal{T}_h) & \longrightarrow & \mathbb{R}^{\# \mathcal{T}_h \cdot \# IJK_g} \\ v & \longmapsto & ((v|_T)(N_{ijk}^g(T)))_{T \in \mathcal{T}_h, (i,j,k) \in IJK_g} \end{cases}.$$

Ist die Familie $(\mathcal{T}_h)_{h \in H}$ formregulär, dann gelten die Abschätzungen

$$\|\chi_{h,g}\|_{l^2 \leftarrow L^2} \leq C(d, \hat{T}, \gamma, g) \cdot \frac{1}{h_{\min, \mathcal{T}_h}^{d/2}}, \quad \|\chi_{h,g}^{-1}\|_{L^2 \leftarrow l^2} \leq C(d, \hat{T}, g) \cdot h_{\mathcal{T}_h}^{d/2}.$$

Ist $(\mathcal{T}_h)_{h \in H}$ zusätzlich quasi-uniform, dann gilt insbesondere

$$\|\chi_{h,g}\|_{l^2 \leftarrow L^2} \cdot \|\chi_{h,g}^{-1}\|_{L^2 \leftarrow l^2} \leq C(d, \hat{T}, \gamma, \nu, g).$$

Beweis:

Wir zeigen exemplarisch nur den Fall $d = 3$:

Schritt 1: Basis-Elemente bilden Basis von $\mathbb{S}^{g,0}(\mathcal{T}_h)$

Für die angegebenen Basis-Elemente $\varphi_{T,ijk}^g$ gilt:

$$\forall T, \tilde{T} \in \mathcal{T}_h : \forall (i, j, k), (\tilde{i}, \tilde{j}, \tilde{k}) \in IJK_g : \quad (\varphi_{T,ijk}^g|_T)(N_{\tilde{i}\tilde{j}\tilde{k}}^g(\tilde{T})) = \delta_{[(T,i,j,k)=(\tilde{T},\tilde{i},\tilde{j},\tilde{k})]}.$$

Insbesondere ist die Menge $\{\varphi_{T,ijk}^g \mid T \in \mathcal{T}_h, (i, j, k) \in IJK_g\} \subseteq \mathbb{S}^{g,0}(\mathcal{T}_h)$ linear unabhängig.

Dass die Basis-Elemente auch tatsächlich ein Erzeugenden-System von $\mathbb{S}^{g,0}(\mathcal{T}_h)$ bilden, sieht man auf

folgende Weise ein: Für jede Funktion $v \in \mathbb{S}^{g,0}(\mathcal{T}_h)$ erfüllt die Funktion

$$\tilde{v} := \sum_{\substack{T \in \mathcal{T}_h, \\ (i,j,k) \in IJK_g}} [(v|_T)(N_{ijk}^g(T))] \cdot \varphi_{T,ijk}^g \in \text{span}\{\varphi_{T,ijk}^g \mid T \in \mathcal{T}_h, (i,j,k) \in IJK_g\} \subseteq \mathbb{S}^{g,0}(\mathcal{T}_h)$$

bereits $[v = \tilde{v} \text{ punktweise auf } \Omega]$, denn

$$\begin{aligned} \forall T \in \mathcal{T}_h : \quad (v - \tilde{v})|_T &= \underbrace{((v - \tilde{v}) \circ F_T) \circ F_T^{-1}}_{\in \mathbb{P}_g(\hat{T})} \\ &\stackrel{\text{L.1.2.3.3}}{=} \left[\sum_{(i,j,k) \in IJK_g} \underbrace{((v - \tilde{v}) \circ F_T)(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)}_{=0} \hat{\varphi}_{ijk}^g \right] \circ F_T^{-1} \\ &= 0. \end{aligned}$$

□

Schritt 2: Abschätzungen der Koordinaten-Abbildungen

Die Koordinaten-Abbildung $\chi_{h,g}$ stimmt auf jedem Element $T \in \mathcal{T}_h$ mit der Koordinaten-Abbildung $\hat{\chi}_g : \mathbb{P}_g(\hat{T}) \longrightarrow \mathbb{R}^{1/6(g+1)(g+2)(g+3)}$ aus Lemma 1.2.3.3 überein:

$$\begin{aligned} \forall v \in \mathbb{S}^{g,0}(\mathcal{T}_h) : \quad \chi_{h,g} v &= ((v|_T)(N_{ijk}^g(T)))_{T \in \mathcal{T}_h, (i,j,k) \in IJK_g} \\ &= \underbrace{((v \circ F_T)(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g))}_{\in \mathbb{P}_g(\hat{T})}_{T \in \mathcal{T}_h, (i,j,k) \in IJK_g} \\ &= (\hat{\chi}_g(v \circ F_T))_{T \in \mathcal{T}_h}. \end{aligned}$$

Es folgt

$$\begin{aligned} \forall v \in \mathbb{S}^{g,0}(\mathcal{T}_h) : \quad \|\chi_{h,g} v\|_{l^2}^2 &= \sum_{T \in \mathcal{T}_h} \|\hat{\chi}_g(v \circ F_T)\|_{l^2}^2 \\ &\leq \|\hat{\chi}_g\|_{l^2 \leftarrow L^2}^2 \cdot \sum_{T \in \mathcal{T}_h} \|v \circ F_T\|_{L^2(\hat{T})}^2 \\ &\stackrel{\text{L.1.2.4.1}}{=} \|\hat{\chi}_g\|_{l^2 \leftarrow L^2}^2 \cdot \sum_{T \in \mathcal{T}_h} |\det \nabla(F_T^{-1})| \cdot \|v\|_{L^2(T)}^2 \\ &\stackrel{\text{L.1.2.5.1, L.1.2.3.3}}{\leq} C(d, \hat{T}, \gamma, g) \cdot \sum_{T \in \mathcal{T}_h} \frac{1}{h_T^d} \cdot \|v\|_{L^2(T)}^2 \\ &\leq C(d, \hat{T}, \gamma, g) \cdot \frac{1}{h_{\min, \mathcal{T}_h}^d} \cdot \|v\|_{L^2(\Omega)}^2 \end{aligned}$$

und damit

$$\|\chi_{h,g}\|_{l^2 \leftarrow L^2} \leq C(d, \hat{T}, \gamma, g) \cdot \frac{1}{h_{\min, \mathcal{T}_h}^{d/2}}.$$

Weiters folgt aus

$$\begin{aligned}
\forall v \in \mathbb{S}^{g,0}(\mathcal{T}_h) : \quad \|v\|_{L^2(\Omega)}^2 &= \sum_{T \in \mathcal{T}_h} \|v\|_{L^2(T)}^2 \\
&\stackrel{\text{L.1.2.4.1}}{=} \sum_{T \in \mathcal{T}_h} |\det \nabla F_T| \cdot \|(\hat{\chi}_g^{-1} \circ \hat{\chi}_g)(v \circ F_T)\|_{L^2(\hat{T})}^2 \\
&\leq \sum_{T \in \mathcal{T}_h} |\det \nabla F_T| \cdot \|\hat{\chi}_g^{-1}\|_{L^2 \leftarrow l^2}^2 \cdot \|\hat{\chi}_g(v \circ F_T)\|_{l^2}^2 \\
&\stackrel{\text{L.1.2.5.1, L.1.2.3.3}}{\leq} C(d, \hat{T}, g) \cdot h_{\mathcal{T}_h}^d \cdot \sum_{T \in \mathcal{T}_h} \|\hat{\chi}_g(v \circ F_T)\|_{l^2}^2 \\
&= C(d, \hat{T}, g) \cdot h_{\mathcal{T}_h}^d \cdot \|\chi_{h,g} v\|_{l^2}^2
\end{aligned}$$

bereits

$$\|\chi_{h,g}^{-1}\|_{L^2 \leftarrow l^2} \leq C(d, \hat{T}, g) \cdot h_{\mathcal{T}_h}^{d/2}.$$

□

■

1.2.9 Approximations-Eigenschaften von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ für $g \geq 1$

Satz 1.2.9.1 (Approximations-Operator nach $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ für $g \geq 1$).

Seien $d \in \{1, 2, 3\}$, $\hat{T} \subseteq \mathbb{R}^d$ das Referenz-Element für Intervall-, Dreiecks- bzw. Tetraeder-Gitter, $\Omega \subseteq \mathbb{R}^d$ ein Polygon und $(\mathcal{T}_h)_{h \in H}$ eine formreguläre Familie von Gittern auf Ω . Sei weiters $g \geq 1$ gegeben. Wir bezeichnen wieder mit $\hat{I}_g : C^0(\hat{T}) \rightarrow \mathbb{P}_g(\hat{T})$ den Interpolations-Operator aus Lemma 1.2.3.7.

Dann erfüllt für jedes $h \in H$ der lineare *Interpolations-Operator*

$$I_{h,g} : \begin{cases} C^0(\bar{\Omega}) & \longrightarrow \mathbb{S}^{g,1}(\mathcal{T}_h) \\ v & \longmapsto \sum_{T \in \mathcal{T}_h} [\hat{I}_g(v \circ F_T) \circ F_T^{-1}] \cdot \mathbb{I}_T \end{cases}$$

die folgenden Eigenschaften:

•) **Stetigkeit:** $\|I_{h,g}\|_{C^0 \leftarrow C^0} \leq C(d, \hat{T}, g)$.

•) **Interpolations-Eigenschaft:**

$$\forall v \in C^0(\bar{\Omega}) : \forall x \in \{F_T(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \mid T \in \mathcal{T}_h, (i, j, k) \in IJK_g\} : \quad (I_{h,g}v)(x) = v(x).$$

•) **Projektions-Eigenschaft:** $I_{h,g}|_{\mathbb{S}^{g,1}(\mathcal{T}_h)} = \text{id}$.

•) **Erhaltung der Randwerte:** $\forall v \in C^0(\bar{\Omega})$ mit $v|_{\partial\Omega} = 0$: $I_{h,g}v \in \mathbb{S}_0^{g,1}(\mathcal{T}_h)$.

•) **Fehler-Abschätzung:** Seien $k > \frac{d}{2}$, $l := \min\{k, g+1\} > \frac{d}{2}$ und $m \in \{0, 1\}$ gegeben. Dann ist $I_{h,g}|_{H^l(\Omega)}$ stetig mit

$$\|I_{h,g}|_{H^l}\|_{H^m \leftarrow H^l} \leq C(d, \hat{T}, \gamma, h_{\min, \mathcal{T}_h}, k, g, m)$$

und es gilt die Fehler-Abschätzung^a

$$\forall v \in H^k(\Omega) : \quad \|v - I_{h,g}v\|_{H^m(\Omega)} \leq C(d, \hat{T}, \gamma, k, g, m) \cdot h_{\mathcal{T}_h}^{l-m} \cdot |v|_{H^l(\Omega)}.$$

^aWir setzen o.B.d.A. $[\forall h \in H : \forall T \in \mathcal{T}_h : h_T \leq 1]$ voraus.

Beweis:

Wir zeigen exemplarisch nur den Fall $d = 3$:

Schritt 1: $I_{h,g}v \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$

Laut Definition 1.2.2.1 reicht es, $I_{h,g}v \in C^0(\bar{\Omega})$ zu zeigen. Dafür wiederum reicht es, die Stetigkeit an allen Element-Grenzen $F \in \mathcal{F}_h$ zu zeigen. Wir zeigen die Stetigkeit aber nur an den Grenzen $F \in \mathcal{F}_{I,h}$ (Der Beweis der Stetigkeit bis hin zu den verbleibenden Rand-Flächen $F \in \mathcal{F}_{D,h}$ verläuft analog und zeigt im Falle $v|_{\partial\Omega} = 0$ auch gleich $I_{h,g}v \in \mathbb{S}_0^{g,1}(\mathcal{T}_h)$):

Sei $F \in \mathcal{F}_{I,h}$ eine der inneren Element-Grenzen. Wir bezeichnen mit $S, T \in \mathcal{T}_h$ die beiden angrenzenden Elemente. Seien $v \in C^0(\bar{S})$ und $w \in C^0(\bar{T})$ Funktionen mit

$$v|_{\bar{F}} = w|_{\bar{F}}.$$

Wir möchten $(\hat{I}_g(v \circ F_S) \circ F_S^{-1})|_{\bar{F}} = (\hat{I}_g(w \circ F_T) \circ F_T^{-1})|_{\bar{F}}$ zeigen:

Wir zeigen, dass die Polynome $\hat{I}_g(v \circ F_S) \circ F_S^{-1} \in \mathbb{P}_g(S)$ und $\hat{I}_g(w \circ F_T) \circ F_T^{-1} \in \mathbb{P}_g(T)$ die Voraussetzungen von Satz 1.2.10.2, Beweis-Schritt 1, erfüllen (Die *Flächen-Stützstellen* $N_{mn}^g(F)$ werden dort definiert.):

$$\begin{aligned} \forall (m, n) \in MN_g : \\ (\hat{I}_g(v \circ F_S) \circ F_S^{-1})(N_{mn}^g(F)) &= (\hat{I}_g(v \circ F_S) \circ F_S^{-1})(F_S((\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)_{(i,j,k)=ijk(g,F,S,m,n)})) \\ &= (\hat{I}_g(v \circ F_S))((\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)_{(i,j,k)=ijk(g,F,S,m,n)}) \\ &\stackrel{\text{L.1.2.3.7}}{=} (v \circ F_S)((\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)_{(i,j,k)=ijk(g,F,S,m,n)}) \\ &= v(\underbrace{N_{mn}^g(F)}_{\in \bar{F}}) \\ &= w(N_{mn}^g(F)) \\ &= (\hat{I}_g(w \circ F_T) \circ F_T^{-1})(N_{mn}^g(F)). \end{aligned}$$

Laut Satz 1.2.10.2, Beweis-Schritt 1, gilt damit schon

$$(\hat{I}_g(v \circ F_S) \circ F_S^{-1})|_{\bar{F}} = (\hat{I}_g(w \circ F_T) \circ F_T^{-1})|_{\bar{F}}.$$

□

Schritt 2: Eigenschaften von $I_{h,g}$

Die verbleibenden Eigenschaften von $I_{h,g}$ können unmittelbar auf jene von $\hat{I}_g$ aus Lemma 1.2.3.7 zurückgeführt werden:

Es ist $I_{h,g}$ linear und stetig:

$$\begin{aligned} \forall v \in C^0(\bar{\Omega}) : \quad \|I_{h,g}v\|_{C^0(\bar{\Omega})} &= \max_{T \in \mathcal{T}_h} \|\hat{I}_g(v \circ F_T) \circ F_T^{-1}\|_{C^0(\bar{T})} \\ &= \max_{T \in \mathcal{T}_h} \|\hat{I}_g(v \circ F_T)\|_{C^0(\bar{T})} \\ &\leq \|\hat{I}_g\|_{C^0 \leftarrow C^0} \cdot \max_{T \in \mathcal{T}_h} \|v \circ F_T\|_{C^0(\bar{T})} \\ &= \|\hat{I}_g\|_{C^0 \leftarrow C^0} \cdot \max_{T \in \mathcal{T}_h} \|v\|_{C^0(\bar{T})} \\ &\stackrel{\text{L.1.2.3.7}}{\leq} C(d, \hat{T}, g) \cdot \|v\|_{C^0(\bar{\Omega})}. \end{aligned}$$

Die Interpolations-Eigenschaft ergibt sich aus

$$\begin{aligned}
\forall v \in C^0(\bar{\Omega}) : \forall x = F_T(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) : \quad (I_{h,g}v)(x) &= [\hat{I}_g(v \circ F_T) \circ F_T^{-1}](F_T(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)) \\
&= [\hat{I}_g(\underbrace{v \circ F_T}_{\in C^0(\hat{T})})](\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \\
&\stackrel{\text{L.1.2.3.7}}{=} (v \circ F_T)(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \\
&= v(x).
\end{aligned}$$

Die Projektions-Eigenschaft folgt aus

$$\forall v \in \mathbb{S}^{g,1}(\mathcal{T}_h) : \quad I_{h,g}v = \sum_{T \in \mathcal{T}_h} [\underbrace{\hat{I}_g(v \circ F_T)}_{\in \mathbb{P}_g(\hat{T})} \circ F_T^{-1}] \cdot \mathbb{I}_T \stackrel{\text{L.1.2.3.7}}{=} \sum_{T \in \mathcal{T}_h} [(v \circ F_T) \circ F_T^{-1}] \cdot \mathbb{I}_T = v.$$

□

Schritt 3: Fehler-Abschätzung

Seien nun $k > \frac{d}{2}$, $l := \min\{k, g+1\} > \frac{d}{2}$ und $m \in \{0, 1\}$ gegeben. Wegen $l > \frac{d}{2}$ und Satz 1.1.2.2 gilt die stetige Einbettung $H^l(\Omega) \hookrightarrow C^0(\bar{\Omega})$.

Damit

$\forall v \in H^l(\Omega) :$

$$\begin{aligned}
\|I_{h,g}v\|_{H^m(\Omega)}^2 &= \sum_{T \in \mathcal{T}_h} \sum_{i=0}^m |\hat{I}_g(v \circ F_T) \circ F_T^{-1}|_{H^i(T)}^2 \\
&\stackrel{\text{L.1.2.4.1}}{\leq} C(d, m) \cdot \sum_{T \in \mathcal{T}_h} \sum_{i=0}^m \|\nabla(F_T^{-1})\|_2^{2i} \cdot |\det \nabla F_T| \cdot |\hat{I}_g(\underbrace{v \circ F_T}_{\in H^l(\hat{T})})|_{H^i(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.3.7}}{\leq} C(d, m) \cdot \|\hat{I}_g\|_{H^l} \|H^m \leftarrow H^l\| \cdot \sum_{T \in \mathcal{T}_h} \sum_{i=0}^m \|\nabla(F_T^{-1})\|_2^{2i} \cdot |\det \nabla F_T| \cdot \sum_{j=0}^l |v \circ F_T|_{H^j(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.3.7, L.1.2.4.1}}{\leq} C(d, \hat{T}, k, g, m) \cdot \sum_{T \in \mathcal{T}_h} \sum_{i=0}^m \|\nabla(F_T^{-1})\|_2^{2i} \cdot \sum_{j=0}^l \|\nabla F_T\|_2^{2j} \cdot |v|_{H^j(T)}^2 \\
&\stackrel{\text{L.1.2.5.1}}{\leq} C(d, \hat{T}, \gamma, k, g, m) \cdot \sum_{T \in \mathcal{T}_h} \sum_{i=0}^m \frac{1}{h_T^{2i}} \cdot \sum_{j=0}^l h_T^{2j} \cdot |v|_{H^j(T)}^2 \\
&\stackrel{h_T \leq 1}{\leq} C(d, \hat{T}, \gamma, h_{\min, \mathcal{T}_h}, k, g, m) \cdot \|v\|_{H^l(\Omega)}^2.
\end{aligned}$$

Die Fehler-Abschätzung ergibt sich nun aus

$$\begin{aligned}
\forall v \in H^k(\Omega) : \\
\|v - I_{h,g}v\|_{H^m(\Omega)}^2 &= \sum_{T \in \mathcal{T}_h} \sum_{i=0}^m |v - \hat{I}_g(v \circ F_T) \circ F_T^{-1}|_{H^i(T)}^2 \\
&\stackrel{\text{L.1.2.4.1}}{\leq} C(d, m) \cdot \sum_{T \in \mathcal{T}_h} \sum_{i=0}^m \|\nabla(F_T^{-1})\|_2^{2i} \cdot |\det \nabla F_T| \cdot \underbrace{|v \circ F_T - \hat{I}_g(v \circ F_T)|}_{\in H^k(\hat{T})}^2_{H^i(\hat{T})} \\
&\stackrel{\text{L.1.2.3.7}}{\leq} C(d, \hat{T}, k, g, m) \cdot \sum_{T \in \mathcal{T}_h} \sum_{i=0}^m \|\nabla(F_T^{-1})\|_2^{2i} \cdot |\det \nabla F_T| \cdot |v \circ F_T|_{H^i(\hat{T})}^2 \\
&\stackrel{\text{L.1.2.4.1}}{\lesssim} C(d, \hat{T}, k, g, m) \cdot \sum_{T \in \mathcal{T}_h} \sum_{i=0}^m \|\nabla(F_T^{-1})\|_2^{2i} \cdot \|\nabla F_T\|_2^{2l} \cdot |v|_{H^i(T)}^2 \\
&\stackrel{\substack{h_T \leq 1, \\ \text{L.1.2.5.1}}}{\leq} C(d, \hat{T}, \gamma, k, g, m) \cdot \sum_{T \in \mathcal{T}_h} h_T^{2(l-m)} \cdot |v|_{H^l(T)}^2 \\
&\stackrel{m \leq l}{\leq} C(d, \hat{T}, \gamma, k, g, m) \cdot h_{\mathcal{T}_h}^{2(l-m)} \cdot |v|_{H^l(\Omega)}^2.
\end{aligned}$$

□

■

Korollar 1.2.9.2 (Bramble-Hilbert für $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ mit $g \geq 1$).

Seien $d \in \{1, 2, 3\}$, $\hat{T} \subseteq \mathbb{R}^d$ das Referenz-Element für Intervall-, Dreiecks- bzw. Tetraeder-Gitter, $\Omega \subseteq \mathbb{R}^d$ ein Polygon und $(\mathcal{T}_h)_{h \in H}$ eine formreguläre Familie von Gittern auf Ω . Seien weiters $k > \frac{d}{2}$, $g \geq 1$, $l := \min\{k, g+1\} \geq 1$ und $m \in \{0, 1\}$. Dann gilt^a:

$$\begin{aligned}
\forall v \in H^k(\Omega) : \quad \inf_{w \in \mathbb{S}^{g,1}(\mathcal{T}_h)} \|v - w\|_{H^m(\Omega)} &\leq C(d, \hat{T}, \gamma, k, g, m) \cdot h_{\mathcal{T}_h}^{l-m} \cdot |v|_{H^l(\Omega)}, \\
\forall v \in H^k(\Omega) \cap H_0^1(\Omega) : \quad \inf_{w \in \mathbb{S}_0^{g,1}(\mathcal{T}_h)} \|v - w\|_{H^m(\Omega)} &\leq C(d, \hat{T}, \gamma, k, g, m) \cdot h_{\mathcal{T}_h}^{l-m} \cdot |v|_{H^l(\Omega)}.
\end{aligned}$$

Ist zusätzlich $(\mathcal{T}_h)_{h \in H}$ quasi-uniform und $g+1 \leq k$, so sind die Konvergenz-Ordnungen $\mathcal{O}(h_{\mathcal{T}_h}^{l-m})$ (bezüglich $h \rightarrow 0$) jeweils maximal^b.

^aMan beachte: Wegen $l = \min\{k, g+1\}$ liefert eine Erhöhung des Polynom-Grades g nur dann höhere Konvergenz-Geschwindigkeit, wenn die zu approximierende Funktion hinreichend glatt ist. Die Raum-Dimension d hingegen spielt für die Konvergenz-Geschwindigkeit keine Rolle.

^bDa die Konvergenz-Ordnung mit Hilfe des Operators $I_{h,g} : C^0(\bar{\Omega}) \rightarrow \mathbb{S}^{g,1}(\mathcal{T}_h)$ aus Satz 1.2.9.1 gezeigt wird, muss sie *nicht* zwingend die größtmögliche sein. A priori könnte es ja auch einen weiteren Operator $\tilde{I}_{h,g} : C^0(\bar{\Omega}) \rightarrow \mathbb{S}^{g,1}(\mathcal{T}_h)$ geben, der auf eine noch höhere Konvergenz-Ordnung führt. Wie wir im Beweis sehen werden, ist dies jedoch nicht der Fall.

Beweis:

Schritt 1: Fehler-Abschätzung

Mit dem Operator $I_{h,g} : C^0(\bar{\Omega}) \rightarrow \mathbb{S}^{g,1}(\mathcal{T}_h)$ aus Satz 1.2.9.1 ergibt sich

$$\forall v \in H^k(\Omega) : \quad \inf_{w \in \mathbb{S}^{g,1}(\mathcal{T}_h)} \|v - w\|_{H^m(\Omega)} \leq \|v - I_{h,g}v\|_{H^m(\Omega)} \stackrel{\text{S.1.2.9.1}}{\leq} C(d, \hat{T}, \gamma, k, g, m) \cdot h_{\mathcal{T}_h}^{l-m} \cdot |v|_{H^l(\Omega)}.$$

Die entsprechende Aussage für alle $v \in H^k(\Omega) \cap H_0^1(\Omega)$ folgt aus der Tatsache

$$I_{h,g}(H^k \cap H_0^1) \subseteq I_{h,g}(C^0 \cap H_0^1) \stackrel{\text{S.1.2.9.1}}{\subseteq} \mathbb{S}_0^{g,1}(\mathcal{T}_h).$$

□

Schritt 2: Optimalität der Konvergenz-Ordnung

Es bezeichne $\alpha_{(0)} \in \mathbb{R}$ den maximalen Exponenten mit

$$\forall v \in H^k(\Omega) \cap H_0^1(\Omega) : \inf_{w \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)} \|v - w\|_{H^m(\Omega)} \leq C(d, \hat{T}, \gamma, k, g, m) \cdot h_{\mathcal{T}_h}^{\alpha_{(0)}} \cdot |v|_{H^l(\Omega)}.$$

Insbesondere gilt $\alpha_{(0)} \geq l - m$.

Es bezeichne weiters $P_{h,g} : H^m(\Omega) \rightarrow \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ die $\langle \cdot, \cdot \rangle_{H^m(\Omega)}$ -Orthogonal-Projektion³. Für jedes feste $\tilde{g} \geq 0$ gilt⁴:

$$\begin{aligned} \forall v \in \mathbb{S}_{(0)}^{\tilde{g},k}(\mathcal{T}_h) : \quad & \frac{1}{\nu^{2(l-m)}} \cdot h_{\mathcal{T}_h}^{2(l-m)} \cdot |v|_{H^l(\Omega)}^2 \leq \sum_{T \in \mathcal{T}_h} h_T^{2(l-m)} \cdot |v|_{H^l(T)}^2 \\ & \stackrel{l > g}{=} \sum_{T \in \mathcal{T}_h} h_T^{2(l-m)} \cdot \underbrace{|v - \overbrace{P_{h,g} v}^{\mathbb{P}_g(T)}|_{H^l(T)}^2}_{\in \mathbb{S}_{(0)}^{\tilde{g},0}(\mathcal{T}_h)} \\ & \stackrel{\substack{m \leq l, \\ \text{L.1.2.6.1}}}{\leq} C(d, \hat{T}, \gamma, k, \tilde{g}, m) \cdot \sum_{T \in \mathcal{T}_h} |v - P_{h,g} v|_{H^m(T)}^2 \\ & \leq C(d, \hat{T}, \gamma, k, \tilde{g}, m) \cdot \|v - P_{h,g} v\|_{H^m(\Omega)}^2 \\ & \stackrel{\text{Def. } P_{h,g}}{=} C(d, \hat{T}, \gamma, k, \tilde{g}, m) \cdot \inf_{w \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)} \|v - w\|_{H^m(\Omega)}^2 \\ & \stackrel{\text{Def. } \alpha_{(0)}}{\lesssim} C(d, \hat{T}, \gamma, k, \tilde{g}, m) \cdot h_{\mathcal{T}_h}^{2\alpha_{(0)}} \cdot |v|_{H^l(\Omega)}^2. \end{aligned}$$

Wir dürfen das Polynom

$$v(x) := x_1^l \in \mathbb{P}_l(\Omega) \subseteq \mathbb{S}^{l,k}(\mathcal{T}_h)$$

einsetzen und erhalten unter Beachtung von $|v|_{H^l(\Omega)}^2 = (l!)^2 \cdot |\Omega| \neq 0$ und $\lim_{h \rightarrow 0} h_{\mathcal{T}_h} = 0$ unmittelbar $\alpha \leq l - m$.

Um auch $\alpha_0 \leq l - m$ zu zeigen⁵, müssen wir nur v um einen geeigneten Faktor erweitern (exemplarisch für $d = 3$): Für jede Rand-Fläche $F \in \mathcal{F}_{D,h}$ bezeichne $n(F) \in F^\perp$ einen Normal-Vektor und $N(F) \in F$ einen beliebigen Punkt auf F . Dann erfüllt das Polynom

$$v_0(x) := x_1^l \cdot \prod_{F \in \mathcal{F}_{D,h}} \underbrace{\langle x - N(F), n(F) \rangle_2}_{\substack{\in \mathbb{P}_1(\Omega), \\ (\cdot)|_F = 0}} \in \mathbb{S}_0^{l+\#\mathcal{F}_{D,h},k}(\mathcal{T}_h)$$

wegen $v_0 \notin \mathbb{P}_{l-1}(\Omega)$ die Bedingung $|v_0|_{H^l(\Omega)} \neq 0$ (vgl. Satz 1.1.2.2). Wir dürfen diese Funktion oben einsetzen und erhalten auch hier $\alpha_0 \leq l - m$.

□

■

³Man beachte den Unterschied zum gleichnamigen Operator aus Satz 1.2.7.1.

⁴Die Ungleichungen sind eigentlich nur für alle $\tilde{g} \geq g + 1$ interessant, da sonst wegen $l > g \geq \tilde{g}$ bereits $[\forall v \in \mathbb{S}_{(0)}^{\tilde{g},k}(\mathcal{T}_h) : |v|_{H^l(\Omega)} = 0]$ gilt.

⁵Wir könnten natürlich auch gleich die Funktion v_0 in die Ungleichung für α einsetzen, um $\alpha \leq l - m$ zu erhalten.

1.2.10 Basis von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ für $g \geq 1$

Definition 1.2.10.1 (Indizes zur Basis von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ für $g \geq 1$).

Seien $d \in \{1, 2, 3\}$, $\hat{T} \subseteq \mathbb{R}^d$ das Referenz-Element für Intervall-, Dreiecks- bzw. Tetraeder-Gitter, $\Omega \subseteq \mathbb{R}^d$ ein Polygon und $(\mathcal{T}_h)_{h \in H}$ eine Familie von Gittern auf Ω . Wir bezeichnen wieder für festes $g \geq 1$ mit $\{\hat{x}_i^g \mid i \in I_g\}$, $\{(\hat{x}_i^g, \hat{y}_j^g) \mid (i, j) \in IJ_g\}$ bzw. $\{(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \mid (i, j, k) \in IJK_g\}$ die Referenz-Stützstellen aus Definition 1.2.3.1.

Wir werden in Satz 1.2.10.2 folgende Indizes und Index-Mengen benötigen:

- **Indizes bzgl. $N \in \mathcal{N}_h$ ($d \in \{1, 2, 3\}$, $g \geq 1$):** Für jeden Knoten $N \in \mathcal{N}_h$ gilt: Für jedes der angrenzenden Elemente $T \in \{T \in \mathcal{T}_h \mid N \in \mathcal{N}(T)\}$ gibt es ein eindeutiges Index-Tupel $i(g, N, T) \in I_g$, $ij(g, N, T) \in IJ_g$ bzw. $ijk(g, N, T) \in IJK_g$ mit (exemplarisch für $d = 3$)

$$F_T((\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \mid (i, j, k) = ijk(g, N, T)) = N.$$

- **Indizes bzgl. $E \in \mathcal{E}_h$ ($d \in \{2, 3\}$, $g \geq 2$):** Wir definieren die Index-Mengen

$$\begin{aligned} (g \geq 1) \quad L_g &:= \{0, \dots, g\}, \\ (g \geq 2) \quad L_g^\times &:= \{1, \dots, g-1\}. \end{aligned}$$

Wir wählen für jede Kante $E \in \mathcal{E}_h$ eine Orientierung aus, indem wir ihre Knoten durchnummerieren: $\{N \in \mathcal{N}_h \mid N \in \overline{E}\} = \{N_0^g(E), N_g^g(E)\}$. Mit den *Kanten-Stützstellen*

$$\forall l \in L_g : \quad N_l^g(E) := (1 - \frac{l}{g}) \cdot N_0^g(E) + \frac{l}{g} \cdot N_g^g(E) \in \overline{E}$$

gilt: Für jedes der angrenzenden Elemente $T \in \{T \in \mathcal{T}_h \mid E \in \mathcal{E}(T)\}$ und jedes $l \in L_g$ gibt es ein eindeutiges Index-Tupel $ij(g, E, T, l) \in IJ_g$ bzw. $ijk(g, E, T, l) \in IJK_g$ mit (exemplarisch für $d = 3$)

$$F_T((\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \mid (i, j, k) = ijk(g, E, T, l)) = N_l^g(E).$$

- **Indizes bzgl. $F \in \mathcal{F}_h$ ($d = 3$, $g \geq 3$):** Wir definieren die Index-Mengen

$$\begin{aligned} (g \geq 1) \quad MN_g &:= \{(m, n) \mid m \in \{0, \dots, g\}, n \in \{0, \dots, g-m\}\}, \\ (g \geq 3) \quad MN_g^\times &:= \{(m, n) \mid m \in \{1, \dots, g-1\}, n \in \{1, \dots, g-m-1\}\}. \end{aligned}$$

Wir wählen für jede Fläche $F \in \mathcal{F}_h$ eine Orientierung aus, indem wir ihre Knoten durchnummerieren: $\{N \in \mathcal{N}_h \mid N \in \overline{F}\} = \{N_{00}^g(F), N_{g0}^g(F), N_{0g}^g(F)\}$. Mit den *Flächen-Stützstellen*

$$\forall (m, n) \in MN_g : \quad N_{mn}^g(F) := (1 - \frac{m}{g} - \frac{n}{g}) \cdot N_{00}^g(F) + \frac{m}{g} \cdot N_{g0}^g(F) + \frac{n}{g} \cdot N_{0g}^g(F) \in \overline{F}$$

gilt: Für jedes der angrenzenden Elemente $T \in \{T \in \mathcal{T}_h \mid F \in \mathcal{F}(T)\}$ und jedes $(m, n) \in MN_g$ gibt es ein eindeutiges Index-Tupel $ijk(g, F, T, m, n) \in IJK_g$ mit

$$F_T((\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \mid (i, j, k) = ijk(g, F, T, m, n)) = N_{mn}^g(F).$$

- **Indizes bzgl. $T \in \mathcal{T}_h$ ($d \in \{1, 2, 3\}$):** Für jedes Element $T \in \mathcal{T}_h$ ist durch die zugehörige Referenz-Abbildung F_T eine kanonische Orientierung vorgegeben. Wir definieren die *Element-Stützstellen* (exemplarisch für $[d = 3, g \geq 4]$)

$$\forall (i, j, k) \in IJK_g : \quad N_{ijk}^g(T) := F_T(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \in T.$$

Satz 1.2.10.2 (Basis von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ für $g \geq 1$).

Seien $d \in \{1, 2, 3\}$, $\hat{T} \subseteq \mathbb{R}^d$ das Referenz-Element für Intervall-, Dreiecks- bzw. Tetraeder-Gitter, $\Omega \subseteq \mathbb{R}^d$ ein Polygon und $(\mathcal{T}_h)_{h \in H}$ eine Familie von Gittern auf Ω . Wir bezeichnen wieder für festes $g \geq 1$ mit $\{\hat{x}_i^g \mid i \in I_g\}$, $\{(\hat{x}_i^g, \hat{y}_j^g) \mid (i, j) \in IJ_g\}$ bzw. $\{(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \mid (i, j, k) \in IJK_g\}$ die Referenz-Stützstellen und mit $\{\hat{\varphi}_i^g \mid i \in I_g\}$, $\{\hat{\varphi}_{ij}^g \mid (i, j) \in IJ_g\}$ bzw. $\{\hat{\varphi}_{ijk}^g \mid (i, j, k) \in IJK_g\}$ die Referenz-Formfunktionen aus Definition 1.2.3.1.

- **Definition der Basis-Elemente:** Mit Hilfe der Indizes und Index-Mengen aus Definition 1.2.10.1 lassen sich die Basis-Elemente von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ explizit anschreiben (exemplarisch für $d = 3$)^a:

$$\begin{aligned}
(g \geq 1) \quad & \forall N \in \mathcal{N}_{(I),h} : \\
& \varphi_N^g := \sum_{N \in \mathcal{N}(T)}^{T \in \mathcal{T}_h} [\hat{\varphi}_{ijk(g,N,T)}^g \circ F_T^{-1}] \cdot \mathbb{I}_T \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h), \\
(g \geq 2) \quad & \forall E \in \mathcal{E}_{(I),h} : \forall l \in L_g^\times : \\
& \varphi_{E,l}^g := \sum_{E \in \mathcal{E}(T)}^{T \in \mathcal{T}_h} [\hat{\varphi}_{ijk(g,E,T,l)}^g \circ F_T^{-1}] \cdot \mathbb{I}_T \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h), \\
(g \geq 3) \quad & \forall F \in \mathcal{F}_{(I),h} : \forall (m, n) \in MN_g^\times : \\
& \varphi_{F,mn}^g := \sum_{F \in \mathcal{F}(T)}^{T \in \mathcal{T}_h} [\hat{\varphi}_{ijk(g,F,T,m,n)}^g \circ F_T^{-1}] \cdot \mathbb{I}_T \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h), \\
(g \geq 4) \quad & \forall T \in \mathcal{T}_h : \forall (i, j, k) \in IJK_g^\times : \\
& \varphi_{T,ijk}^g := [\hat{\varphi}_{ijk}^g \circ F_T^{-1}] \cdot \mathbb{I}_T \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h).
\end{aligned}$$

- **Beschreibung der Basis:** Wir geben nun eine Basis von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ an:

o) *1D Intervall-Gitter:* Für $d = 1$ gilt

$$\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) = \text{span}(\underbrace{\{\varphi_N^g \mid N \in \mathcal{N}_{(I),h}\}}_{g \geq 1} \cup \underbrace{\{\varphi_{T,i}^g \mid T \in \mathcal{T}_h, i \in I_g^\times\}}_{g \geq 2}).$$

o) *2D Dreiecks-Gitter:* Für $d = 2$ gilt

$$\begin{aligned}
\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) = & \text{span}(\underbrace{\{\varphi_N^g \mid N \in \mathcal{N}_{(I),h}\}}_{g \geq 1} \cup \underbrace{\{\varphi_{E,l}^g \mid E \in \mathcal{E}_{(I),h}, l \in L_g^\times\}}_{g \geq 2} \\
& \cup \underbrace{\{\varphi_{T,ij}^g \mid T \in \mathcal{T}_h, (i, j) \in IJ_g^\times\}}_{g \geq 3}).
\end{aligned}$$

o) *3D Tetraeder-Gitter:* Für $d = 3$ gilt

$$\begin{aligned}
\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) = & \text{span}(\underbrace{\{\varphi_N^g \mid N \in \mathcal{N}_{(I),h}\}}_{g \geq 1} \cup \underbrace{\{\varphi_{E,l}^g \mid E \in \mathcal{E}_{(I),h}, l \in L_g^\times\}}_{g \geq 2} \\
& \cup \underbrace{\{\varphi_{F,mn}^g \mid F \in \mathcal{F}_{(I),h}, (m, n) \in MN_g^\times\}}_{g \geq 3} \cup \underbrace{\{\varphi_{T,ijk}^g \mid T \in \mathcal{T}_h, (i, j, k) \in IJK_g^\times\}}_{g \geq 4}).
\end{aligned}$$

- **Koordinaten-Abbildung:** Für jedes $v \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ stimmen die Koeffizienten bezüglich der obigen Basis mit den Funktionswerten an den Knoten, Kanten-, Flächen- und Element-Stützstellen

überein (exemplarisch für $[d = 3, g \geq 4]$):

$$\begin{aligned} \forall v \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) : \quad v &= \sum_{N \in \mathcal{N}_{(I),h}} v(N) \cdot \varphi_N^g + \sum_{\substack{E \in \mathcal{E}_{(I),h}, \\ l \in L_g^\times}} v(N_l^g(E)) \cdot \varphi_{E,l}^g \\ &+ \sum_{\substack{F \in \mathcal{F}_{(I),h}, \\ (m,n) \in MN_g^\times}} v(N_{mn}^g(F)) \cdot \varphi_{F,mn}^g + \sum_{\substack{T \in \mathcal{T}_h, \\ (i,j,k) \in IJK_g^\times}} v(N_{ijk}^g(T)) \cdot \varphi_{T,ijk}^g. \end{aligned}$$

Insbesondere ist die Koordinaten-Abbildung bezüglich der obigen Basis gegeben durch (exemplarisch für $[d = 3, g \geq 4]$)

$$\chi_{h,g} : \begin{cases} \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) & \longrightarrow \mathbb{R}^{\# \mathcal{N}_{(I),h} + \# \mathcal{E}_{(I),h} \cdot \# L_g^\times + \# \mathcal{F}_{(I),h} \cdot \# MN_g^\times + \# \mathcal{T}_h \cdot \# IJK_g^\times} \\ v & \longmapsto ((v(N))_N, (v(N_l^g(E)))_{E,l}, (v(N_{mn}^g(F)))_{F,(m,n)}, (v(N_{ijk}^g(T)))_{T,(i,j,k)}) \end{cases}.$$

Ist die Familie $(\mathcal{T}_h)_{h \in H}$ formregulär, dann gilt^b

$$\|\chi_{h,g}\|_{l^2 \leftarrow H^1} \leq C(d, \hat{T}, \gamma, g) \cdot \frac{1}{h_{\min, \mathcal{T}_h}^{d/2}}, \quad \|\chi_{h,g}^{-1}\|_{H^1 \leftarrow l^2} \leq C(d, \hat{T}, \gamma, g) \cdot \frac{h_{\mathcal{T}_h}^{d/2}}{h_{\min, \mathcal{T}_h}}.$$

Ist $(\mathcal{T}_h)_{h \in H}$ zusätzlich quasi-uniform, dann gilt insbesondere

$$\|\chi_{h,g}\|_{l^2 \leftarrow H^1} \cdot \|\chi_{h,g}^{-1}\|_{H^1 \leftarrow l^2} \leq C(d, \hat{T}, \gamma, \nu, g) \cdot \frac{1}{h_{\min, \mathcal{T}_h}}.$$

^aFür $k \geq 2$ ist die Konstruktion einer Basis von $\mathbb{S}^{g,k}(\mathcal{T}_h)$ komplizierter: Die Basiselemente müssen dann an den Element-Grenzen nicht nur stetig, sondern sogar (mindestens) stetig differenzierbar sein. Die Referenz-Formfunktionen $\hat{\varphi}_i^g, \hat{\varphi}_{ij}^g, \hat{\varphi}_{ijk}^g$ aus Definition 1.2.3.1 ergeben beim Element-weisen Zusammensetzen jedoch nur stetige Funktionen.

^bWir setzen o.B.d.A. $[\forall h \in H : h_{\min, \mathcal{T}_h} \leq 1]$ voraus.

Beweis:

Wir zeigen exemplarisch nur den Fall $[d = 3, g \geq 4]$:

Schritt 1: $\forall F \in \mathcal{F}_{I,h} : [\forall v \in \mathbb{P}_g(S), w \in \mathbb{P}_g(T) \text{ mit } (\forall (m,n) : v(N_{mn}^g(F)) = w(N_{mn}^g(F))) : v|_{\overline{F}} = w|_{\overline{F}}]$
Sei $F \in \mathcal{F}_{I,h}$ eine der inneren Element-Grenzen. Wir bezeichnen mit $S, T \in \mathcal{T}_h$ die beiden angrenzenden Elemente. Seien $v \in \mathbb{P}_g(S)$ und $w \in \mathbb{P}_g(T)$ Polynome mit

$$\forall (m,n) \in MN_g : \quad v(N_{mn}^g(F)) = w(N_{mn}^g(F)).$$

Wir möchten $v|_{\overline{F}} = w|_{\overline{F}}$ zeigen:

Die Polynom-Funktion $w \in \mathbb{P}_g(T)$ kann klarerweise zu einer Polynom-Funktion $w \in \mathbb{P}_g(S \cup T)$ fortgesetzt werden. Für die eindeutige Seitenfläche $\hat{F} \in \mathcal{F}(\hat{T})$ mit $F_S(\hat{F}) = F$ gilt:

$$\forall (\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) \in \overline{\hat{F}} : \quad \underbrace{(v \circ F_S - w|_{\overline{S}} \circ F_S)}_{\in \mathbb{P}_g(\hat{T})}(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) = (v - w|_{\overline{S}})(\underbrace{F_S(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)}_{\in \{N_{mn}^g(F) \mid (m,n) \in MN_g\}}) = 0.$$

Laut Lemma 1.2.3.6 gilt damit schon

$$\forall x \in \overline{F} : \quad (v|_{\overline{F}} - w|_{\overline{F}})(x) = (v - w|_{\overline{S}})|_{\overline{F}}(x) = (v \circ F_S - w|_{\overline{S}} \circ F_S)|_{\overline{F}}(\underbrace{F_S^{-1}(x)}_{\in \hat{F}}) \stackrel{\text{L.1.2.3.6}}{=} 0.$$

□

Schritt 2: $\varphi_N^g, \varphi_{E,l}^g, \varphi_{F,mn}^g, \varphi_{T,ijk}^g \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$

Wir zeigen exemplarisch nur die Stetigkeit der Funktionen $\{\varphi_N^g \mid N \in \mathcal{N}_h\}$:

Seien also $N \in \mathcal{N}_h$ ein fester Knoten und $\varphi_N^g \in \mathbb{S}^{g,0}(\mathcal{T}_h)$ wie oben. Laut Definition 1.2.2.1 reicht es, $\varphi_N^g \in C^0(\bar{\Omega})$ zu zeigen. Dafür wiederum reicht es, die Stetigkeit an allen Element-Grenzen $F \in \mathcal{F}_h$ mit $F \subseteq \text{supp}(\varphi_N^g)$ zu zeigen. Wir zeigen die Stetigkeit aber nur an den Grenzen $F \in \mathcal{F}_{I,h}$ mit $F \subseteq \text{supp}(\varphi_N^g)$ und $N \in \bar{F}$ (Der Beweis für die verbleibenden Grenzen $F \subseteq \text{supp}(\varphi_N^g)$ mit $N \notin \bar{F}$ verläuft analog und zeigt im Falle $N \in \mathcal{N}_{I,h}$ auch gleich $\varphi_N^g \in \mathbb{S}_0^{g,1}(\mathcal{T}_h)$.):

Es bezeichne $S, T \in \mathcal{T}_h$ die beiden Nachbar-Elemente von F , d.h. $F \in \mathcal{F}(S) \cap \mathcal{F}(T)$. Dann gilt:

$$\begin{aligned}
\forall (m, n) \in MN_g : \quad & \underbrace{(\varphi_N^g|_S)(N_{mn}^g(F))}_{\in \mathbb{P}_g(S)} = [\hat{\varphi}_{ijk(N,S)} \circ F_S^{-1}](F_S((\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)|_{(i,j,k)=ijk(g,F,S,m,n)})) \\
& = \delta_{[ijk(N,S)=ijk(g,F,S,m,n)]} \\
& = \mathbb{I}_{[N=N_{mn}^g(F)]} \\
& = \delta_{[ijk(g,N,T)=ijk(g,F,T,m,n)]} \\
& = [\hat{\varphi}_{ijk(g,N,T)}^g \circ F_T^{-1}](F_T((\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)|_{(i,j,k)=ijk(g,F,T,m,n)})) \\
& = \underbrace{(\varphi_N^g|_T)(N_{mn}^g(F))}_{\in \mathbb{P}_g(T)}.
\end{aligned}$$

Mit Schritt 1 folgt bereits $(\varphi_N^g|_S)|_{\bar{F}} = (\varphi_N^g|_T)|_{\bar{F}}$. □

Schritt 3: $\varphi_N^g, \varphi_{E,l}^g, \varphi_{F,mn}^g, \varphi_{T,ijk}^g$ bilden Basis von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$

Die Menge der durch die Familie $\{F_T | T \in \mathcal{T}_h\}$ transformierten Referenz-Stützstellen zerfällt in eine disjunkte Vereinigung von Knoten, Kanten-, Flächen- und Element-Stützstellen:

$$\begin{aligned}
\{F_T(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g) | T \in \mathcal{T}_h, (i, j, k) \in IJK_g\} &= \{N | N \in \mathcal{N}_h\} \\
&\cup \{N_l^g(E) | E \in \mathcal{E}_h, l \in L_g^\times\} \\
&\cup \{N_{mn}^g(F) | F \in \mathcal{F}_h, (m, n) \in MN_g^\times\} \\
&\cup \{N_{ijk}^g(T) | T \in \mathcal{T}_h, (i, j, k) \in IJK_g^\times\}.
\end{aligned}$$

Jedes der angegebenen Basis-Elemente $\varphi_N^g, \varphi_{E,l}^g, \varphi_{F,mn}^g, \varphi_{T,ijk}^g$ nimmt an genau einer dieser transformierten Referenz-Stützstellen den Wert $1 \in \mathbb{R}$ und an allen anderen den Wert $0 \in \mathbb{R}$ an. Insbesondere ist die Menge der Basis-Elemente linear unabhängig. Dass sie auch tatsächlich ein Erzeugenden-System von $\mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ bildet, sieht man auf folgende Weise ein: Für jede Funktion $v \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)$ erfüllt die Funktion

$$\begin{aligned}
\tilde{v} &:= \sum_{N \in \mathcal{N}_{(I),h}} v(N) \cdot \varphi_N^g + \sum_{\substack{E \in \mathcal{E}_{(I),h}, \\ l \in L_g^\times}} v(N_l^g(E)) \cdot \varphi_{E,l}^g \\
&+ \sum_{\substack{F \in \mathcal{F}_{(I),h}, \\ (m,n) \in MN_g^\times}} v(N_{mn}^g(F)) \cdot \varphi_{F,mn}^g + \sum_{\substack{T \in \mathcal{T}_h, \\ (i,j,k) \in IJK_g^\times}} v(N_{ijk}^g(T)) \cdot \varphi_{T,ijk}^g \\
&\in \text{span}(\{\varphi_N^g | N\} \cup \{\varphi_{E,l}^g | E, l\} \cup \{\varphi_{F,mn}^g | F, (m, n)\} \cup \{\varphi_{T,ijk}^g | T, (i, j, k)\}) \subseteq \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)
\end{aligned}$$

bereits $[v = \tilde{v} \text{ punktweise auf } \Omega]$, denn

$$\begin{aligned}
\forall T \in \mathcal{T}_h : \quad (v - \tilde{v})|_T &= \underbrace{((v - \tilde{v}) \circ F_T) \circ F_T^{-1}}_{\in \mathbb{P}_g(\hat{T})} \\
&\stackrel{\text{L.1.2.3.3}}{=} \left[\sum_{(i,j,k) \in IJK_g} \underbrace{((v - \tilde{v}) \circ F_T)(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g)}_{=0} \hat{\varphi}_{ijk}^g \right] \circ F_T^{-1} \\
&= 0.
\end{aligned}$$

□

Schritt 4: Abschätzungen der Koordinaten-Abbildungen

Wir bezeichnen mit $\hat{\chi}_g : \mathbb{P}_g(\hat{T}) \rightarrow \mathbb{R}^{1/6(g+1)(g+2)(g+3)}$ wieder die Koordinaten-Abbildung aus Lemma 1.2.3.3. In Satz 1.2.8.1, Beweis-Schritt 2, haben wir eigentlich die folgenden Abschätzungen gezeigt:

$$\forall v \in \mathbb{S}^{g,0}(\mathcal{T}_h) : \quad C(d, \hat{T}, g) \cdot \frac{1}{h_{\mathcal{T}_h}^d} \cdot \|v\|_{L^2(\Omega)}^2 \leq \sum_{T \in \mathcal{T}_h} \|\hat{\chi}_g(v \circ F_T)\|_{l^2}^2 \leq C(d, \hat{T}, \gamma, g) \cdot \frac{1}{h_{\min, \mathcal{T}_h}^d} \cdot \|v\|_{L^2(\Omega)}^2.$$

Für die Koordinaten-Abbildung $\chi_{h,g} : \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) \rightarrow \mathbb{R}^{\dim \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h)}$ stimmt jetzt zwar $\|\chi_{h,g}v\|_{l^2}^2$ nicht mehr mit dem mittleren Term überein, aber es gelten zumindest die folgenden Abschätzungen:

$$\begin{aligned} \forall v \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) : \quad \|\chi_{h,g}v\|_{l^2}^2 &= \sum_{N \in \mathcal{N}_{(I),h}} v(N)^2 + \sum_{\substack{E \in \mathcal{E}_{(I),h}, \\ l \in L_g^\times}} v(N_l^g(E))^2 \\ &\quad + \sum_{\substack{F \in \mathcal{F}_{(I),h}, \\ (m,n) \in MN_g^\times}} v(N_{mn}^g(F))^2 + \sum_{\substack{T \in \mathcal{T}_h, \\ (i,j,k) \in IJK_g^\times}} v(N_{ijk}^g(T))^2 \\ &\leq \sum_{T \in \mathcal{T}_h} \left[\sum_{N \in \mathcal{N}(T)} v(N)^2 + \sum_{\substack{E \in \mathcal{E}(T), \\ l \in L_g^\times}} v(N_l^g(E))^2 \right. \\ &\quad \left. + \sum_{\substack{F \in \mathcal{F}(T), \\ (m,n) \in MN_g^\times}} v(N_{mn}^g(F))^2 + \sum_{(i,j,k) \in IJK_g^\times} v(N_{ijk}^g(T))^2 \right] \\ &= \sum_{T \in \mathcal{T}_h} \sum_{(i,j,k) \in IJK_g} ((v \circ F_T)(\hat{x}_i^g, \hat{y}_j^g, \hat{z}_k^g))^2 \\ &= \sum_{T \in \mathcal{T}_h} \|\hat{\chi}_g(v \circ F_T)\|_{l^2}^2 \end{aligned}$$

und weiters

$$\begin{aligned} \forall v \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) : \quad \sum_{T \in \mathcal{T}_h} \|\hat{\chi}_g(v \circ F_T)\|_{l^2}^2 &= \sum_{T \in \mathcal{T}_h} \left[\sum_{N \in \mathcal{N}(T)} v(N)^2 + \sum_{\substack{E \in \mathcal{E}(T), \\ l \in L_g^\times}} v(N_l^g(E))^2 \right. \\ &\quad \left. + \sum_{\substack{F \in \mathcal{F}(T), \\ (m,n) \in MN_g^\times}} v(N_{mn}^g(F))^2 + \sum_{(i,j,k) \in IJK_g^\times} v(N_{ijk}^g(T))^2 \right] \\ &\stackrel{v \in C^0(\bar{\Omega})}{=} \sum_{N \in \mathcal{N}_{(I),h}} \underbrace{\#\{T \in \mathcal{T}_h \mid N \in \mathcal{N}(T)\}}_{\leq C(d, \hat{T}, \gamma)} v(N)^2 \\ &\quad + \sum_{\substack{E \in \mathcal{E}_{(I),h}, \\ l \in L_g^\times}} \underbrace{\#\{T \in \mathcal{T}_h \mid E \in \mathcal{E}(T)\}}_{\leq C(d, \hat{T}, \gamma)} v(N_l^g(E))^2 \\ &\quad + \sum_{\substack{F \in \mathcal{F}_{(I),h}, \\ (m,n) \in MN_g^\times}} 2 \cdot v(N_{mn}^g(F))^2 + \sum_{\substack{T \in \mathcal{T}_h, \\ (i,j,k) \in IJK_g^\times}} v(N_{ijk}^g(T))^2 \\ &\stackrel{\text{D.1.2.1.2}}{\leq} C(d, \hat{T}, \gamma) \cdot \|\chi_{h,g}v\|_{l^2}^2. \end{aligned}$$

Die Abschätzungen für $\|\chi_{h,g}\|_{l^2 \leftarrow H^1}$ und $\|\chi_{h,g}^{-1}\|_{H^1 \leftarrow l^2}$ folgen nun aus

$$\forall v \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) : \quad \|\chi_{h,g}v\|_{l^2}^2 \leq \sum_{T \in \mathcal{T}_h} \|\hat{\chi}_g(v \circ F_T)\|_{l^2}^2 \leq C(d, \hat{T}, \gamma, g) \cdot \frac{1}{h_{\min, \mathcal{T}_h}^d} \cdot \|v\|_{H^1(\Omega)}^2$$

und

$$\begin{aligned}
\forall v \in \mathbb{S}_{(0)}^{g,1}(\mathcal{T}_h) : \quad h_{\min, \mathcal{T}_h}^2 \cdot \|v\|_{H^1(\Omega)}^2 &\stackrel{h_{\min, \mathcal{T}_h} \leq 1}{\leq} 1 \cdot \|v\|_{L^2(\Omega)}^2 + \sum_{T \in \mathcal{T}_h} h_T^2 \cdot |v|_{H^1(T)}^2 \\
&\stackrel{\text{L.1.2.6.1}}{\leq} \|v\|_{L^2(\Omega)}^2 + C(d, \hat{T}, \gamma, g) \cdot \sum_{T \in \mathcal{T}_h} \|v\|_{L^2(T)}^2 \\
&\lesssim C(d, \hat{T}, \gamma, g) \cdot \|v\|_{L^2(\Omega)}^2 \\
&\lesssim C(d, \hat{T}, \gamma, g) \cdot h_{\mathcal{T}_h}^d \cdot \sum_{T \in \mathcal{T}_h} \|\hat{\chi}_g(v \circ F_T)\|_{l^2}^2 \\
&\lesssim C(d, \hat{T}, \gamma, g) \cdot h_{\mathcal{T}_h}^d \cdot \|\chi_{h,g} v\|_{l^2}^2.
\end{aligned}$$

□

■

2 Optimal-Steuer-Probleme

2.1 Die inf-sup-Bedingung [$\alpha(e) > 0$]

2.1.1 Grundlegende Eigenschaften von Hilbert-Räumen

Satz 2.1.1.1 (Zusammenfassung Hilbert-Räume).

In einem Hilbert-Raum $(X, \langle \cdot, \cdot \rangle_X)$ gelten die folgenden Eigenschaften:

-) **Abschluss $\leftrightarrow$ Orthogonalraum:** Für jeden Unterraum $M \leq X$ gilt:

$$M^\perp = (\overline{M})^\perp, \quad M^{\perp\perp} = \overline{M}, \quad X = \overline{M} \oplus M^\perp.$$

-) **Riesz-Fréchet-Isomorphismus:** Der *Riesz-Fréchet-Isomorphismus*

$$\tau_X : \begin{cases} X & \longrightarrow & X' \\ x & \longmapsto & \langle x, \cdot \rangle_X \end{cases}$$

erfüllt $\|\tau_X\| = 1$ und induziert ein Skalarprodukt auf X' :

$$\forall x'_1, x'_2 \in X' : \quad \langle x'_1, x'_2 \rangle_{X'} := \langle \tau_X^{-1} x'_1, \tau_X^{-1} x'_2 \rangle_X.$$

Die davon induzierte Norm ist genau die bekannte Operator-Norm $\|\cdot\|_{X'}$, d.h. X' ist ein Hilbert-Raum.

Für jeden Unterraum $M \leq X$ gilt

$$\tau_X(M^\perp) = \tau_X(M)^\perp.$$

-) **Summen-Skalarprodukt:** Ist Y ein weiterer Hilbert-Raum, dann ist $X \times Y$ bezüglich dem *Summen-Skalarprodukt*

$$\forall (x_1, y_1), (x_2, y_2) \in X \times Y : \quad \langle (x_1, y_1), (x_2, y_2) \rangle_{X \times Y} := \langle x_1, x_2 \rangle_X + \langle y_1, y_2 \rangle_Y$$

ebenfalls ein Hilbert-Raum. Die davon induzierte Norm erfüllt

$$\forall (x, y) \in X \times Y : \quad \|(x, y)\|_{X \times Y}^2 = \|x\|_X^2 + \|y\|_Y^2.$$

-) **Einschränkungs-Operator:** Sei $X_{\text{stg}} \leq X$ ein weiterer Hilbert-Raum mit stetiger Einbettung $\text{id}_{X_{\text{stg}} \rightarrow X} : X_{\text{stg}} \longrightarrow X$, d.h.

$$\|\text{id}_{X_{\text{stg}} \rightarrow X}\| = \sup_{x \in X_{\text{stg}}} \frac{\|x\|_X}{\|x\|_{X_{\text{stg}}}} < \infty.$$

Dann ist der *Einschränkungs-Operator*

$$(\cdot)|_{X_{\text{stg}}} : X' \longrightarrow X'_{\text{stg}}$$

linear und stetig mit $\|(\cdot)|_{X_{\text{stg}}}\| \leq \|\text{id}_{X_{\text{stg}} \rightarrow X}\|$. Außerdem ist $\text{ran}((\cdot)|_{X_{\text{stg}}}) \subseteq X'_{\text{stg}}$ dicht^a

Seien nun zusätzlich $X_h \leq X$ für alle^b $h \in H$ endlich-dimensionale Unterräume. Erfüllen die Orthogonal-Projektionen $P_{X_h} : X \longrightarrow X_h$ die Konvergenz^c

$$\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\|_{X \leftarrow X_{\text{stg}}} \xrightarrow{h \rightarrow 0} 0,$$

dann muss die Einbettung $\text{id}_{X_{\text{stg}} \rightarrow X}$ schon kompakt sein.

^aMan beachte: Im Allgemeinen ist $(\cdot)|_{X_{\text{stg}}}$ jedoch *nicht* surjektiv: Ein Gegenbeispiel: $\Omega := (0, 1) \subseteq \mathbb{R}$, $X := L^2(\Omega)$, $X_{\text{stg}} := H_0^1(\Omega)$. Dann gilt mit den Riesz-Fréchet-Isomorphismen $\tau_X : X \rightarrow X'$ und $\tau_{X_{\text{stg}}} : X_{\text{stg}} \rightarrow X'_{\text{stg}}$ die Identität $\text{ran}(\tau_{X_{\text{stg}}}^{-1} \circ (\cdot)|_{X_{\text{stg}}} \circ \tau_X) = H^2 \cap H_0^1 \subsetneq H_0^1$. Da τ_X und $\tau_{X_{\text{stg}}}$ bijektiv sind, kann also $(\cdot)|_{X_{\text{stg}}}$ nicht surjektiv sein.

^bWir nehmen für die Indexmenge wieder o.B.d.A. $H \subseteq (0, 1]$ und $0 \in \overline{H}$ an.

^cWir schreiben im Folgenden kurz $\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\|$ an Stelle von $\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\|_{X \leftarrow X_{\text{stg}}}$.

Beweis:

Wir beweisen nur die letzte Aussage über die Kompaktheit der Einbettung $\text{id}_{X_{\text{stg}} \rightarrow X}$.

Schritt 1: $\forall (x_n)_{n \in \mathbb{N}} \subseteq X_{\text{stg}}$ beschr. : $\forall \varepsilon > 0 : \exists n_\varepsilon : \mathbb{N} \rightarrow \mathbb{N}$ mit $[\forall i, j \in \mathbb{N} : \|x_{n_\varepsilon(i)} - x_{n_\varepsilon(j)}\|_X \leq \varepsilon]$

Seien $(x_n)_{n \in \mathbb{N}} \subseteq X_{\text{stg}}$ eine beschränkte Folge und $\varepsilon > 0$ gegeben.

Wegen $\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0$ gibt es ein hinreichend kleines $h \in H$ mit

$$2 \cdot \left(\sup_{n \in \mathbb{N}} \|x_n\|_{X_{\text{stg}}} + 1 \right) \cdot \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \leq \varepsilon.$$

Die Folge $(P_{X_h} x_n)_{n \in \mathbb{N}} \subseteq X_h$ ist beschränkt, denn

$$\begin{aligned} \forall n \in \mathbb{N} : \quad \|P_{X_h} x_n\|_X &\leq [\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| + \|\text{id}_{X_{\text{stg}} \rightarrow X}\|] \cdot \|x_n\|_{X_{\text{stg}}} \\ &\leq (1 + \|\text{id}_{X_{\text{stg}} \rightarrow X}\|) \cdot \sup_{m \in \mathbb{N}} \|x_m\|_{X_{\text{stg}}} < \infty. \end{aligned}$$

Wegen $\dim X_h < \infty$ ist die Einheitskugel in $(X_h, \|\cdot\|_X)$ kompakt, also gibt es ein $x_\varepsilon \in X_h$ und eine Teilfolgen-Indizierung $n_\varepsilon : \mathbb{N} \rightarrow \mathbb{N}$ mit

$$P_{X_h} x_{n_\varepsilon(i)} \xrightarrow[\|\cdot\|_X]{i \rightarrow \infty} x_\varepsilon.$$

Wir können die Teilfolge o.B.d.A. gleich so wählen, dass $[\forall i \in \mathbb{N} : \|x_\varepsilon - P_{X_h} x_{n_\varepsilon(i)}\|_X \leq \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\|]$.

Damit

$$\begin{aligned} \forall i, j \in \mathbb{N} : \quad \|x_{n_\varepsilon(i)} - x_{n_\varepsilon(j)}\|_X &\leq \|(\text{id} - P_{X_h})|_{X_{\text{stg}}} x_{n_\varepsilon(i)}\|_X + \|P_{X_h} x_{n_\varepsilon(i)} - x_\varepsilon\|_X \\ &\quad + \|x_\varepsilon - P_{X_h} x_{n_\varepsilon(j)}\|_X + \|(\text{id} - P_{X_h})|_{X_{\text{stg}}} x_{n_\varepsilon(j)}\|_X \\ &\leq \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \cdot [\|x_{n_\varepsilon(i)}\|_{X_{\text{stg}}} + 1 + 1 + \|x_{n_\varepsilon(j)}\|_{X_{\text{stg}}}] \\ &\leq 2 \cdot \left(\sup_{n \in \mathbb{N}} \|x_n\|_{X_{\text{stg}}} + 1 \right) \cdot \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \\ &\stackrel{\text{Def. } h}{\leq} \varepsilon. \end{aligned}$$

□

Schritt 2: $\text{id}_{X_{\text{stg}} \rightarrow X}$ ist kompakt

Sei $(x_n)_{n \in \mathbb{N}} \subseteq X_{\text{stg}}$ eine beschränkte Folge. Wendet man die in Schritt 1 gezeigte Aussage induktiv an, so erhält man Teilfolgen-Indizierungen $\{n_{1/k} : \mathbb{N} \rightarrow \mathbb{N} \mid k \in \mathbb{N}\}$ mit

$$\forall k \in \mathbb{N} : \quad n_{1/(k+1)}(\mathbb{N}) \subseteq n_{1/k}(\mathbb{N})$$

und

$$\forall k \in \mathbb{N} : \quad \left[\forall i, j \in \mathbb{N} : \quad \|x_{n_{1/k}(i)} - x_{n_{1/k}(j)}\|_X \leq \frac{1}{k} \right].$$

Die durch $[\forall k \in \mathbb{N} : n(k) := n_{1/k}(k)]$ definierte Teilfolge $(x_{n(k)})_{k \in \mathbb{N}}$ ist nun eine $\|\cdot\|_X$ -Cauchy-Folge (und damit konvergent), denn

$$\begin{aligned} \forall \varepsilon > 0 : \forall k \geq l \geq \text{ceil}\left(\frac{1}{\varepsilon}\right) : \quad \|x_{n(k)} - x_{n(l)}\|_X &= \|x_{n_{1/k}(k)} - x_{n_{1/l}(l)}\|_X \\ &\stackrel{n_{1/k}(\mathbb{N}) \subseteq n_{1/l}(\mathbb{N})}{=} \|x_{n_{1/l}(\tilde{l})} - x_{n_{1/l}(l)}\|_X \\ &\stackrel{\text{Def. } n_{1/l}}{\leq} \frac{1}{\tilde{l}} \\ &\leq \varepsilon. \end{aligned}$$

□

■

2.1.2 Die inf-sup-Bedingung $[\alpha(e) > 0]$

Lemma 2.1.2.1 (Von Bilinearform induzierte Operatoren).

Seien X, Y Hilbert-Räume. Eine Bilinearform $e : X \times Y \rightarrow \mathbb{R}$ ist stetig, genau falls

$$\|e\| := \sup_{\substack{x \in X^\times \\ y \in Y^\times}} \frac{e(x, y)}{\|x\|_X \cdot \|y\|_Y} < \infty.$$

In diesem Fall definieren wir die linearen, stetigen, *von e induzierten Operatoren*

$$E : \begin{cases} X & \longrightarrow & Y' \\ x & \longmapsto & e(x, \cdot) \end{cases}, \quad E^t : \begin{cases} Y & \longrightarrow & X' \\ y & \longmapsto & e(\cdot, y) \end{cases},$$

wobei $\|E\| = \|E^t\| = \|e\|$.

Es gilt:

- Die *Hilbert-Raum-Transponierte* $E^T : Y' \rightarrow X$ von E hängt mit E^t über die Riesz-Fréchet-Isomorphismen $\tau_X : X \rightarrow X'$ und $\tau_Y : Y \rightarrow Y'$ zusammen:

$$E^t = \tau_X \circ E^T \circ \tau_Y.$$

- Außerdem gilt

$$\begin{aligned} \tau_X(\ker(E)) &= \text{ran}(E^t)^\perp, \\ \tau_Y(\ker(E^t)) &= \text{ran}(E)^\perp. \end{aligned}$$

Beweis:

Schritt 1: $E^t = \tau_X \circ E^T \circ \tau_Y$, $\tau_X(\ker(E)) = \text{ran}(E^t)^\perp$, $\tau_Y(\ker(E^t)) = \text{ran}(E)^\perp$

Aus

$$\forall x \in X : \forall y \in Y :$$

$$\langle (\tau_X^{-1} \circ E^t)(y), x \rangle_X = (E^t y)(x) = e(x, y) = (Ex)(y) = \langle \tau_Y y, Ex \rangle_{Y'} = \langle (E^T \circ \tau_Y)(y), x \rangle_X$$

folgt $E^t = \tau_X \circ E^T \circ \tau_Y$.

Zusammen mit

$$\ker(E) = \{x \in X \mid \forall y' \in Y' : \langle x, E^T y' \rangle_X = \langle Ex, y' \rangle_{Y'} = 0\} = \{x \in X \mid x \perp \text{ran}(E^T)\} = \text{ran}(E^T)^\perp$$

ergibt das

$$\tau_X(\ker(E)) = \tau_X(\operatorname{ran}(E^T)^\perp) = \tau_X(\operatorname{ran}(E^T))^\perp = \operatorname{ran}(\tau_X \circ E^T)^\perp = \operatorname{ran}(E^t \circ \tau_Y^{-1})^\perp = \operatorname{ran}(E^t)^\perp.$$

Die Identität $\tau_Y(\ker(E^t)) = \operatorname{ran}(E)^\perp$ zeigt man völlig analog. □

Schritt 2: $\|E\| = \|E^t\| = \|e\| \geq \alpha(e)$

Es gilt

$$\|E\| = \sup_{x \in X^\times} \frac{\|Ex\|_{Y'}}{\|x\|_X} = \sup_{x \in X^\times} \sup_{y \in Y^\times} \frac{e(x, y)}{\|x\|_X \cdot \|y\|_Y} = \|e\|$$

und analog $\|E^t\| = \|e\|$. Weiters

$$\alpha(e) = \inf_{x \in X^\times} \sup_{y \in Y^\times} \frac{e(x, y)}{\|x\|_X \cdot \|y\|_Y} \leq \sup_{x \in X^\times} \sup_{y \in Y^\times} \frac{e(x, y)}{\|x\|_X \cdot \|y\|_Y} = \|e\|.$$

□

■

Satz 2.1.2.2 (inf-sup-Bedingung).

Seien X, Y Hilbert-Räume und $e : X \times Y \rightarrow \mathbb{R}$ eine stetige Bilinearform.

- **(Kontinuierliche) inf-sup-Bedingung:** Es erfüllt e die *(kontinuierliche) inf-sup-Bedingung*

$$\alpha(e) := \inf_{x \in X^\times} \sup_{y \in Y^\times} \frac{e(x, y)}{\|x\|_X \cdot \|y\|_Y} = \inf_{x \in X^\times} \frac{\|Ex\|_{Y'}}{\|x\|_X} > 0,$$

genau falls E injektiv und $\operatorname{ran}(E)$ abgeschlossen sind. In diesem Fall ist E^t surjektiv und es gelten die Relationen

$$\begin{aligned} \alpha(e) &\leq \|e\|, \\ \frac{1}{\alpha(e)} &= \|E^{-1}\|, \\ \forall x \in X : \quad \alpha(e) \cdot \|x\|_X &\leq \|Ex\|_{Y'}, \\ \forall y \in \ker(E^t)^\perp : \quad \alpha(e) \cdot \|y\|_Y &\leq \|E^t y\|_{X'}. \end{aligned}$$

- **(Kontinuierliche) Transponierte inf-sup-Bedingung:** Für E^t gelten analoge Aussagen bezüglich der *(kontinuierlichen) transponierten inf-sup-Bedingung*:

$$\alpha^t(e) := \inf_{y \in Y^\times} \sup_{x \in X^\times} \frac{e(x, y)}{\|x\|_X \cdot \|y\|_Y} = \inf_{y \in Y^\times} \frac{\|E^t y\|_{X'}}{\|y\|_Y} > 0.$$

- **Diskrete (transponierte) inf-sup-Bedingung:** Seien nun zusätzlich $X_h \leq X$ und $Y_h \leq Y$ mit $\dim X_h < \infty$ und $\dim Y_h < \infty$ für alle^a $h \in H$ sog. *Ansatz-Räume*. Wir bezeichnen mit $E_h : X_h \rightarrow Y_h'$ und $E_h^t : Y_h \rightarrow X_h'$ die von $e|_{X_h \times Y_h}$ induzierten Operatoren. Für sie gelten ebenfalls analoge Aussagen bezüglich der *diskreten inf-sup-Bedingung*^b

$$\alpha_h(e) := \alpha(e|_{X_h \times Y_h}) = \inf_{x \in X_h^\times} \sup_{y \in Y_h^\times} \frac{e(x, y)}{\|x\|_X \cdot \|y\|_Y} = \inf_{x \in X_h^\times} \frac{\|E_h x\|_{Y_h'}}{\|x\|_X} > 0$$

und der *diskreten transponierten inf-sup-Bedingung*:

$$\alpha_h^t(e) := \alpha^t(e|_{X_h \times Y_h}) = \inf_{y \in Y_h^\times} \sup_{x \in X_h^\times} \frac{e(x, y)}{\|x\|_X \cdot \|y\|_Y} = \inf_{y \in Y_h^\times} \frac{\|E_h^t y\|_{X'_h}}{\|y\|_Y} > 0.$$

Schließlich setzen wir noch

$$\alpha_0(e) := \inf_{h \in H} \alpha_h(e) \geq 0, \quad \alpha_0^t(e) := \inf_{h \in H} \alpha_h^t(e) \geq 0.$$

^aWir nehmen für die Indexmenge wieder o.B.d.A. $H \subseteq (0, 1]$ und $0 \in \overline{H}$ an.

^bMan beachte: Gilt zwar $[\alpha(e) > 0]$, aber erfüllt ein gegebenes Paar (X_h, Y_h) von Unterräumen *nicht* $[\alpha_h(e) > 0]$, so kann dies manchmal durch Verkleinern von X_h und immer durch Vergrößern von Y_h erreicht werden (Spätestens bei $Y_h := Y$).

Beweis:

Schritt 1: $[\alpha(e) > 0] \Rightarrow [\forall x \in X : \alpha(e) \cdot \|x\|_X \leq \|Ex\|_{Y'}, E \text{ injektiv, } \text{ran}(E) \text{ abgeschlossen}]$
 Aus der Definition von $\alpha(e)$ folgt unmittelbar

$$\forall x \in X : \quad \alpha(e) \cdot \|x\|_X \leq \|Ex\|_{Y'}.$$

Insbesondere ist E injektiv. Außerdem ist $\text{ran}(E)$ abgeschlossen, denn die Urbild-Folgen von konvergenten Folgen aus $\text{ran}(E)$ sind Cauchy-Folgen. □

Schritt 2: $[\alpha(e) > 0] \Rightarrow [\forall y \in \ker(E^t)^\perp : \alpha(e) \cdot \|y\|_Y \leq \|E^t y\|_{X'}, E^t \text{ surjektiv}]$
 Für jedes $y \in \ker(E^t)^\perp$ gibt es wegen

$$\tau_Y y \in \tau_Y(\ker(E^t)^\perp) = \tau_Y(\ker(E^t))^\perp \stackrel{\text{L.2.1.2.1}}{=} \text{ran}(E)^{\perp\perp} = \overline{\text{ran}(E)} \stackrel{\text{S.1}}{=} \text{ran}(E)$$

ein $x \in X$ mit $\tau_Y y = Ex$, wodurch

$$\|y\|_Y^2 = (\tau_Y y)(y) = (Ex)(y) = (E^t y)(x) \leq \|E^t y\|_{X'} \|x\|_X \stackrel{\text{S.1}}{\leq} \frac{1}{\alpha(e)} \|E^t y\|_{X'} \|Ex\|_{Y'} = \frac{1}{\alpha(e)} \|E^t y\|_{X'} \|y\|_Y.$$

Die daraus resultierende Ungleichung

$$\forall y \in \ker(E^t)^\perp : \quad \alpha(e) \cdot \|y\|_Y \leq \|E^t y\|_{X'}$$

hat analog zu Schritt 1 die Abgeschlossenheit von $\text{ran}(E^t)$ zur Folge: Die Urbild-Folgen von konvergenten Folgen aus $\text{ran}(E^t)$ liegen wegen $Y = \ker(E^t) \oplus \ker(E^t)^\perp$ o.B.d.A. in $\ker(E^t)^\perp$ und sind laut obiger Ungleichung Cauchy-Folgen.

Damit lässt sich schließlich auch die Surjektivität von E^t zeigen:

$$\text{ran}(E^t) = \overline{\text{ran}(E^t)} = \text{ran}(E^t)^{\perp\perp} \stackrel{\text{L.2.1.2.1}}{=} \tau_X(\underbrace{\ker(E)}_{=\{0\}})^\perp = \{0\}^\perp = X'.$$

□

Schritt 3: $[E \text{ injektiv, } \text{ran}(E) \text{ abgeschlossen}] \Rightarrow [\alpha(e) > 0]$

Sind E injektiv und $\text{ran}(E)$ abgeschlossen, dann liefert der Satz von der offenen Abbildung die Stetigkeit von $E^{-1} : \text{ran}(E) \rightarrow X$ und wir erhalten

$$\alpha(e) = \inf_{x \in X^\times} \frac{\|Ex\|_{Y'}}{\|x\|_X} = \inf_{x \in X^\times} \frac{\|Ex\|_{Y'}}{\|E^{-1}Ex\|_X} = \frac{1}{\sup_{y' \in \text{ran}(E)^\times} \frac{\|E^{-1}y'\|_X}{\|y'\|_{Y'}}} = \frac{1}{\|E^{-1}\|} > 0.$$

□

■

Korollar 2.1.2.3 (Ladyzhenskaya-Babuška-Brezzi-Bedingung^a).

Seien X, Y Hilbert-Räume und $e : X \times Y \rightarrow \mathbb{R}$ eine stetige Bilinearform.

Dann sind die folgenden Aussagen äquivalent^b:

-) $\alpha(e) > 0$ und E^t ist injektiv.
-) E^t ist bijektiv.
-) E ist bijektiv.
-) $\alpha^t(e) > 0$ und E ist injektiv.

In diesem Fall gilt

$$\alpha(e) = \alpha^t(e).$$

^aWir kürzen im Folgenden mit *LBB-Bedingung* ab.

^bDie Injektivität von E^t ist eine etwas schwächere Forderung als die transponierte inf-sup-Bedingung $\alpha^t(e) > 0$. Das bedeutet: Die Forderung $[\alpha(e) > 0 \text{ und } \alpha^t(e) > 0]$ wäre zu viel, die Forderung $[E \text{ inj. und } E^t \text{ inj.}]$ zu wenig für die Invertierbarkeit von E .

Beweis:

Wir zeigen nur $\alpha^t(e) = \alpha(e)$. Mit Satz 2.1.2.2 gilt:

$$\frac{1}{\alpha^t(e)} = \|(E^t)^{-1}\| = \|(\tau_X \circ E^T \circ \tau_Y)^{-1}\| = \|(E^T)^{-1}\| = \|(E^{-1})^T\| = \|E^{-1}\| = \frac{1}{\alpha(e)}.$$

■

Lemma 2.1.2.4 (Fortin-Kriterium).

Seien X, Y Hilbert-Räume und $e : X \times Y \rightarrow \mathbb{R}$ eine stetige Bilinearform.

Es gelte:

- *) e erfüllt $\alpha(e) > 0$.
- *) $\dim X_h = \dim Y_h < \infty$.

Dann sind folgende Aussagen äquivalent:

-) e erfüllt $\alpha_h(e) > 0$.
-) Es existiert eine lineare, stetige Abbildung $P_{Y_h} : Y \rightarrow Y_h$ (sog. *Galerkin-Projektion*) mit

$$\forall y \in Y : \quad [e(\cdot, P_{Y_h} y) = e(\cdot, y) \quad \text{auf } X_h].$$

In diesem Fall gilt $P_{Y_h} = (E_h^t)^{-1} \circ (\cdot)|_{X_h} \circ E^t$ und $\alpha_h(e) \geq \frac{\alpha(e)}{\|P_{Y_h}\|} > 0$.

Beweis:

Wir bezeichnen mit $E : X \rightarrow Y'$ und $E^t : Y \rightarrow X'$ bzw. mit $E_h : X_h \rightarrow Y'_h$ und $E_h^t : Y_h \rightarrow X'_h$ die von e bzw. $e|_{X_h \times Y_h}$ induzierten Operatoren.

Schritt 1: „ $\Downarrow$ “

Wegen $\alpha_h(e) > 0$ ist $E_h^t : Y_h \rightarrow X'_h$ surjektiv und wegen $\dim X'_h = \dim X_h = \dim Y_h$ auch injektiv. Die Abbildung $P_{Y_h} := (E_h^t)^{-1} \circ (\cdot)|_{X_h} \circ E^t : Y \rightarrow Y_h$ ist linear und stetig und erfüllt

$$\forall y \in Y : \quad [e(\cdot, P_{Y_h} y) = (E_h^t \circ P_{Y_h})(y) = ((\cdot)|_{X_h} \circ E^t)(y) = e(\cdot, y) \quad \text{auf } X_h].$$

Die Abbildung P_{Y_h} ist auf Grund der Bijektivität von E_h^t außerdem eindeutig.

□

Schritt 2: „ $\uparrow$ “
Für jedes $x \in X_h$ gilt

$$\alpha(e) \cdot \|x\|_X \leq \|Ex\|_{Y'} = \sup_{y \in Y^\times} \frac{e(x, y)}{\|y\|_Y} \leq \|P_{Y_h}\| \cdot \sup_{y \in Y^\times} \frac{e(x, P_{Y_h}y)}{\|P_{Y_h}y\|_Y} \leq \|P_{Y_h}\| \cdot \sup_{y \in Y_h^\times} \frac{e(x, y)}{\|y\|_Y} = \|P_{Y_h}\| \cdot \|E_h x\|_{Y'_h}$$

und damit insbesondere $\|P_{Y_h}\| \neq 0$. Es folgt

$$\alpha_h(e) = \inf_{x \in X_h^\times} \frac{\|E_h x\|_{Y'_h}}{\|x\|_X} \geq \frac{\alpha(e)}{\|P_{Y_h}\|} > 0.$$

□

■

Lemma 2.1.2.5 (Koerzive Bilinearformen).

Seien X ein Hilbert-Raum, $X_h \leq X$ für alle $h \in H$ Ansatz-Räume und $e : X \times X \rightarrow \mathbb{R}$ eine stetige Bilinearform.

Ist e koerziv, d.h.

$$\alpha_{\text{kzv}}(e) := \inf_{x \in X^\times} \frac{e(x, x)}{\|x\|_X^2} > 0,$$

dann gilt bereits

$$\begin{aligned} \alpha_{\text{kzv}}(e) &\leq \alpha(e), \\ \alpha_{\text{kzv}}(e) &\leq \alpha^t(e), \\ \forall h \in H : \alpha_{\text{kzv}}(e) &\leq \alpha_0(e) \leq \alpha_h(e), \\ \forall h \in H : \alpha_{\text{kzv}}(e) &\leq \alpha_0^t(e) \leq \alpha_h^t(e). \end{aligned}$$

Insbesondere sind dann alle induzierten Operatoren E, E^t, E_h, E_h^t bijektiv.

Beweis:

Wir zeigen exemplarisch nur die erste Ungleichung:

$$\alpha_{\text{kzv}}(e) = \inf_{x \in X^\times} \frac{e(x, x)}{\|x\|_X^2} \leq \inf_{x \in X^\times} \sup_{y \in X^\times} \frac{e(x, y)}{\|x\|_X \cdot \|y\|_X} = \alpha(e).$$

■

2.2 Variations-Probleme

2.2.1 (Kontinuierliche) Variations-Probleme

Lemma 2.2.1.1 (Ein (kontinuierliches) Variations-Problem).

Seien X, Y Hilbert-Räume, $e : X \times Y \rightarrow \mathbb{R}$ eine stetige Bilinearform und $l \in Y'$.

Es gelte:

*) e erfüllt $\alpha(e) > 0$ und $E^t : Y \rightarrow X'$ ist injektiv.

Dann existiert eine eindeutige (kontinuierliche) Lösung $x \in X$ des folgenden (kontinuierlichen) Variations-Problems:

$$\text{Finde } x \in X \text{ mit : } [e(x, \cdot) = l \text{ auf } Y].$$

Die Lösung x hängt linear und stetig von den Daten l ab mit $\|x\|_X \leq \frac{1}{\alpha(e)} \cdot \|l\|_{Y'}$.

Beweis:

Laut Korollar 2.1.2.3 ist der von e induzierte Operator $E : X \rightarrow Y'$ bijektiv. Damit ist das kontinuierliche Variations-Problem eindeutig lösbar:

$$\forall l \in Y' : \exists! x \in X : [e(x, \cdot) = Ex = l \text{ auf } Y].$$

Der Lösungs-Operator ist gegeben durch $E^{-1} : Y' \rightarrow X$, also insbesondere linear und stetig mit

$$\|x\|_X = \|E^{-1}l\|_X \leq \|E^{-1}\| \cdot \|l\|_{Y'} \stackrel{\text{S. 2.1.2.2}}{=} \frac{1}{\alpha(e)} \cdot \|l\|_{Y'}.$$

■

2.2.2 Diskretisierung

Lemma 2.2.2.1 (Variations-Problem Diskretisierung).

Seien nun zusätzlich $X_h \leq X$ und $Y_h \leq Y$ für alle $h \in H$ Ansatz-Räume mit folgenden Eigenschaften:

- *) $\dim X_h = \dim Y_h < \infty$.
- *) e erfüllt $\alpha_h(e) > 0$.

Dann existiert auch eine eindeutige (diskrete) Lösung $x_h \in X_h$ des (diskreten) Variations-Problems:

$$\text{Finde } x_h \in X_h \text{ mit : } [e(x_h, \cdot) = l \text{ auf } Y_h].$$

Beweis:

Auch der von $e|_{X_h \times Y_h}$ induzierte Operator $E_h : X_h \rightarrow Y'_h$ ist bijektiv, denn aus $\alpha_h(e) > 0$ folgt mit Satz 2.1.2.2 die Injektivität und aus $\dim Y'_h = \dim Y_h = \dim X_h$ auch schon die Bijektivität. Also ist auch das diskrete Variations-Problem eindeutig lösbar und der Lösungs-Operator ist gegeben durch $E_h^{-1} \circ (\cdot)|_{Y'_h} : Y' \rightarrow X_h$.

■

2.2.3 Konvergenz

Satz 2.2.3.1 (Variations-Problem Konvergenz).

Sei X_{stg} ein weiterer Hilbert-Raum mit folgenden Eigenschaften:

- *) $X_{\text{stg}} \leq X$ dicht.
- *) Für die Orthogonal-Projektionen $P_{X_h} : X \rightarrow X_h$ gilt:

$$\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0.$$

Für die Ansatz-Räume X_h gelte zusätzlich:

- *) e erfüllt $\alpha_0(e) > 0$.

Dann gilt^a

$$x_h \xrightarrow[\|\cdot\|_X]{h \rightarrow 0} x$$

und unter der Voraussetzung

- *) $x \in X_{\text{stg}}$

sogar

$$\|x - x_h\|_X \leq \frac{2 \cdot \|e\|}{\alpha_0(e)} \cdot \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \cdot \|x\|_{X_{\text{stg}}}.$$

■

^aDie Konvergenz $x_h \xrightarrow{h \rightarrow 0} x$ lässt sich ohne explizite Voraussetzungen an die Approximations-Güte der Räume Y_h zeigen. Dennoch können diese *nicht* einfach trivial gewählt werden, da ja trotzdem noch $\alpha_h(e) > 0$ erfüllt sein muss.

Beweis:

Schritt 1: Definition der Galerkin-Projektionen

Die Bijektivität von E_h garantiert die eindeutige Lösbarkeit des folgenden Variations-Problems:

$$\text{Für geg. } x \in X, \text{ finde } x_h \in X_h \text{ mit : } [e(x_h, \cdot) = e(x, \cdot) \quad \text{auf } Y_h].$$

Der Lösungs-Operator ist gegeben durch die lineare, stetige *Galerkin-Projektion*

$$G_{X_h} := E_h^{-1} \circ (\cdot)|_{Y_h} \circ E : X \longrightarrow X_h.$$

Sie erfüllt $1 \leq \|G_{X_h}\| \leq \|E_h^{-1}\| \cdot \|E\| = \frac{\|e\|}{\alpha_h(e)} \leq \frac{\|e\|}{\alpha_0(e)}$ und $G_{X_h}|_{X_h} = \text{id}$. Für die kontinuierliche Lösung $x \in X$ und die diskrete Lösung $x_h \in X_h$ von oben gilt auf Grund der *Galerkin-Orthogonalität* $[e(x - x_h, \cdot) = 0 \text{ auf } Y_h]$ auch $x_h = G_{X_h}x$.

Analog garantiert die Bijektivität von E_h^t die eindeutige Lösbarkeit des folgenden Variations-Problems:

$$\text{Für geg. } y \in Y, \text{ finde } y_h \in Y_h \text{ mit : } [e(\cdot, y_h) = e(\cdot, y) \quad \text{auf } X_h].$$

Der Lösungs-Operator ist gegeben durch die lineare, stetige *Galerkin-Projektion*

$$G_{Y_h} := (E_h^t)^{-1} \circ (\cdot)|_{X_h} \circ E^t : Y \longrightarrow Y_h.$$

Sie erfüllt $1 \leq \|G_{Y_h}\| \leq \|(E_h^t)^{-1}\| \cdot \|E^t\| = \frac{\|e\|}{\alpha_h^t(e)} = \frac{\|e\|}{\alpha_h(e)} \leq \frac{\|e\|}{\alpha_0(e)}$ und $G_{Y_h}|_{Y_h} = \text{id}$.

□

Schritt 2: Konvergenz

Aus

$$\begin{aligned} \forall z \in X_{\text{stg}} : \quad \|z - G_{X_h}z\|_X &\leq \|z - P_{X_h}z\|_X + \|G_{X_h}(P_{X_h}z - z)\|_X \\ &\leq (1 + \|G_{X_h}\|) \cdot \|(\text{id} - P_{X_h})z\|_X \\ &\stackrel{1 \leq \|G_{X_h}\|}{\leq} \frac{2 \cdot \|e\|}{\alpha_0(e)} \cdot \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \cdot \|z\|_{X_{\text{stg}}} \end{aligned}$$

folgt

$$\begin{aligned} \forall z \in X_{\text{stg}} : \\ \|x - x_h\|_X &\leq \|x - z\|_X + \|z - G_{X_h}z\|_X + \|G_{X_h}(z - x)\|_X \\ &\leq (1 + \|G_{X_h}\|) \cdot \|x - z\|_X + (1 + \|G_{X_h}\|) \cdot \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \cdot \|z\|_{X_{\text{stg}}} \\ &\stackrel{1 \leq \|G_{X_h}\|}{\leq} \frac{2 \cdot \|e\|}{\alpha_0(e)} \cdot \left[\|x - z\|_X + \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \cdot \|z\|_{X_{\text{stg}}} \right]. \end{aligned}$$

Auf Grund der Dichtheit von $X_{\text{stg}} \subseteq X$ und der Konvergenz $\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0$ gibt es zu jedem $\varepsilon > 0$ ein $h_{\max} > 0$ mit

$$\forall h \in (0, h_{\max}) \cap H : \quad \|x - x_h\|_X \leq \frac{2 \cdot \|e\|}{\alpha_0(e)} \cdot \varepsilon.$$

□

■

2.2.4 Aubin-Nitsche-Trick

Satz 2.2.4.1 (Variations-Problem Aubin-Nitsche-Trick).

Seien^a X_{wk} und Y_{stg}^t weitere Hilbert-Räume mit folgenden Eigenschaften:

- *) $\iota_{X_{\text{wk}}} : X \longrightarrow X_{\text{wk}}$ linear, stetig.
- *) $Y_{\text{stg}}^t \leq Y$.
- *) Für die Orthogonal-Projektionen $P_{Y_h} : Y \longrightarrow Y_h$ gilt:

$$\|(\text{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0.$$

Die Bijektivität von E^t garantiert die eindeutige Lösbarkeit des folgenden (*kontinuierlichen*) *transponierten Variations-Problems*:

$$\text{Für geg. } x \in X, \text{ finde } y \in Y \text{ mit : } [e(\cdot, y) = \langle \iota_{X_{\text{wk}}}(\cdot), \iota_{X_{\text{wk}}} x \rangle_{X_{\text{wk}}} \text{ auf } X].$$

Der zugehörige Lösungs-Operator $S : X \longrightarrow Y$ ist linear, stetig und erfüllt

$$\forall x \in X : \quad \|Sx\|_Y \leq \frac{\|\iota_{X_{\text{wk}}}\|}{\alpha(e)} \cdot \|\iota_{X_{\text{wk}}} x\|_{X_{\text{wk}}}.$$

Erfüllt S darüber hinaus die Voraussetzungen

- *) $\text{ran}(S) \subseteq Y_{\text{stg}}^t$ und $[\forall x \in X : \|Sx\|_{Y_{\text{stg}}^t} \leq C_S \cdot \|\iota_{X_{\text{wk}}} x\|_{X_{\text{wk}}}]$,

dann gilt im Falle

- *) $x \in X_{\text{stg}}$

auch die Fehler-Abschätzung^b

$$\|\iota_{X_{\text{wk}}}(x - x_h)\|_{X_{\text{wk}}} \leq \frac{2 \cdot \|e\|^2 \cdot C_S}{\alpha_0(e)} \cdot \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \cdot \|(\text{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\| \cdot \|x\|_{X_{\text{stg}}}.$$

^aWir verstehen den Raum Y_{stg}^t mit dem Index „t“, um die Zugehörigkeit zum *transponierten* Variations-Problem zu verdeutlichen. Diese Notation ist an dieser Stelle noch überflüssig, wird sich aber in Satz 2.3.4.1 und Satz 2.4.4.1 als vorteilhaft erweisen.

^bDurch diese Abschätzung wird die Rolle der Ansatz-Räume Y_h klarer: Für fest gewählte Ansatz-Räume $(X_h)_{h \in H}$ wirkt sich eine *Vergrößerung* der Räume $(Y_h)_{h \in H}$ auf folgende Weise aus: In der Abschätzung für $\|x - x_h\|_X$ wird nur die Konstante $\alpha_0(e)$ besser, die Konvergenz-Ordnung bleibt jedoch gleich. In der Abschätzung der schwächeren Norm $\|\iota_{X_{\text{wk}}}(x - x_h)\|_{X_{\text{wk}}}$ hingegen verbessert sich neben der Konstante $\alpha_0(e)$ auch die Konvergenz-Ordnung, nämlich um den Faktor $\|(\text{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\|$.

Beweis:

Der Lösungs-Operator S ist linear, stetig und erfüllt

$$\begin{aligned} \forall x \in X : \quad \|Sx\|_Y &= \|(E^t)^{-1} \langle \iota_{X_{\text{wk}}}(\cdot), \iota_{X_{\text{wk}}} x \rangle_{X_{\text{wk}}}\|_Y \\ &\leq \|(E^t)^{-1}\| \cdot \|\langle \iota_{X_{\text{wk}}}(\cdot), \iota_{X_{\text{wk}}} x \rangle_{X_{\text{wk}}}\|_{X'} \\ &\stackrel{\text{S.2.1.2.2}}{\leq} \frac{\|\iota_{X_{\text{wk}}}\|}{\alpha(e)} \cdot \|\iota_{X_{\text{wk}}} x\|_{X_{\text{wk}}}. \end{aligned}$$

Weiters gilt:

$$\begin{aligned}
\forall z \in X : \quad \|\iota_{X_{\text{wk}}}(z - G_{X_h} z)\|_{X_{\text{wk}}}^2 &= \langle \underbrace{\iota_{X_{\text{wk}}}(z - G_{X_h} z)}_{\in X}, \underbrace{\iota_{X_{\text{wk}}}(z - G_{X_h} z)}_{\in X} \rangle_{X_{\text{wk}}} \\
&\stackrel{\text{Def. } S}{=} e(z - G_{X_h} z, S(z - G_{X_h} z)) \\
&\stackrel{\text{Def. } G_{X_h}}{=} e(z - G_{X_h} z, (\text{id} - P_{Y_h})S(z - G_{X_h} z)) \\
&\leq \|e\| \cdot \|z - G_{X_h} z\|_X \cdot \|(\text{id} - P_{Y_h}) \underbrace{S(z - G_{X_h} z)}_{\in Y_{\text{stg}}^t}\|_Y \\
&\leq \|e\| \cdot \|z - G_{X_h} z\|_X \cdot \|(\text{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\| \cdot \| \underbrace{S(z - G_{X_h} z)}_{\in X} \|_{Y_{\text{stg}}^t} \\
&\leq \|e\| \cdot C_S \cdot \|z - G_{X_h} z\|_X \cdot \|(\text{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\| \cdot \|\iota_{X_{\text{wk}}}(z - G_{X_h} z)\|_{X_{\text{wk}}}.
\end{aligned}$$

Insbesondere können wir die kontinuierliche Lösung $z := x \in X$ einsetzen und erhalten unter Beachtung von $x_h = G_{X_h} x$ die Abschätzung (wieder im Falle $x \in X_{\text{stg}}$)

$$\begin{aligned}
\|\iota_{X_{\text{wk}}}(x - x_h)\|_{X_{\text{wk}}} &\leq \|e\| \cdot C_S \cdot \|(\text{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\| \cdot \|x - x_h\|_X \\
&\stackrel{\text{S.2.2.3.1}}{\leq} \frac{2 \cdot \|e\|^2 \cdot C_S}{\alpha_0(e)} \cdot \|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \cdot \|(\text{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\| \cdot \|x\|_{X_{\text{stg}}}.
\end{aligned}$$

■

2.2.5 LGS

Lemma 2.2.5.1 (Variations-Problem LGS).

Seien nun $\{x_1, \dots, x_n\} \subseteq X_h$ und $\{y_1, \dots, y_n\} \subseteq Y_h$ Basen. Wir bezeichnen mit

$$\chi_h : \begin{cases} X_h & \longrightarrow \mathbb{R}^n \\ \sum_{j=1}^n \chi_j x_j & \longmapsto (\chi_j)_{j=1}^n \end{cases}, \quad \gamma_h : \begin{cases} Y_h & \longrightarrow \mathbb{R}^n \\ \sum_{i=1}^n \gamma_i y_i & \longmapsto (\gamma_i)_{i=1}^n \end{cases}$$

die zugehörigen Koordinaten-Abbildungen. Die Koeffizienten $(\chi_j)_{j=1}^n \in \mathbb{R}^n$ aus der Darstellung

$$x_h = \sum_{j=1}^n \chi_j x_j$$

lassen sich durch Lösen eines regulären LGS bestimmen: Mit

$$\begin{aligned}
\mathbf{E} &:= (e(x_j, y_i))_{i,j=1}^n \in \mathbb{R}^{n \times n}, \\
\boldsymbol{\chi} &:= (\chi_j)_{j=1}^n \in \mathbb{R}^n, \\
\mathbf{l} &:= (l(y_i))_{i=1}^n \in \mathbb{R}^n
\end{aligned}$$

gilt:

$$\mathbf{E} \cdot \boldsymbol{\chi} = \mathbf{l}.$$

Die Konditionszahl der System-Matrix $\mathbf{E}$ (sog. *Steifigkeits-Matrix*) erfüllt die Abschätzung^a

$$\text{cond}(\mathbf{E}) \leq \frac{\|e\|}{\alpha_0(e)} \cdot \|\chi_h\| \cdot \|\chi_h^{-1}\| \cdot \|\gamma_h\| \cdot \|\gamma_h^{-1}\|.$$

Die rechte Seite $\mathbf{l}$ des LGS wird als *Last-Vektor* bezeichnet.

^aHier sind immer die Operator-Normen bezüglich der Standard-Normen $\|\cdot\|_X$ (bzw. $\|\cdot\|_Y$) und $\|\cdot\|_2$ gemeint.

Beweis:

Die System-Matrix $\mathbf{E} \in \mathbb{R}^{n \times n}$ ist genau die darstellende Matrix der linearen Abbildung $(\gamma_h^{-1})^T \circ \tau_{Y_h}^{-1} \circ E_h \circ \chi_h^{-1} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ bezüglich der euklidischen Basis $\{e_1, \dots, e_n\} \subseteq \mathbb{R}^n$, denn

$$\forall i, j : \quad \langle ((\gamma_h^{-1})^T \circ \tau_{Y_h}^{-1} \circ E_h \circ \chi_h^{-1})(e_j), e_i \rangle_2 = \langle (\tau_{Y_h}^{-1} \circ E_h)(x_j), y_i \rangle_Y = (E_h x_j)(y_i) = e(x_j, y_i) = \langle \mathbf{E} \cdot e_j, e_i \rangle_2.$$

Es folgt

$$\begin{aligned} \|\mathbf{E}\| &= \|(\gamma_h^{-1})^T \circ \tau_{Y_h}^{-1} \circ E_h \circ \chi_h^{-1}\| \leq \|\gamma_h^{-1}\| \cdot \|E_h\| \cdot \|\chi_h^{-1}\| \leq \|e\| \cdot \|\chi_h^{-1}\| \cdot \|\gamma_h^{-1}\|, \\ \|\mathbf{E}^{-1}\| &= \|\chi_h \circ E_h^{-1} \circ \tau_{Y_h} \circ \gamma_h^T\| \leq \|\chi_h\| \cdot \|E_h^{-1}\| \cdot \|\gamma_h\| \leq \frac{1}{\alpha_0(e)} \cdot \|\chi_h\| \cdot \|\gamma_h\| \end{aligned}$$

und damit die behauptete Abschätzung für $\text{cond}(\mathbf{E}) = \|\mathbf{E}\| \cdot \|\mathbf{E}^{-1}\|$.

■

2.2.6 Strang-Lemma

Lemma 2.2.6.1 (Variations-Problem Strang-Lemma).

Seien nun zusätzlich $l_h \in Y'_h$ für alle $h \in H$ stetige Linearformen mit

$$*) \quad \|l - l_h\|_{Y'_h} \xrightarrow{h \rightarrow 0} 0.$$

Dann erfüllen auch die eindeutigen Lösungen $\tilde{x}_h \in X_h$ der diskreten Variations-Probleme

$$[e(\tilde{x}_h, \cdot) = l_h \quad \text{auf } Y_h]$$

die Konvergenz $\tilde{x}_h \xrightarrow[\|\cdot\|_X]{h \rightarrow 0} x$ und im Falle

$$*) \quad x \in X_{\text{stg}}$$

die Fehler-Abschätzungen

$$\begin{aligned} \|x - \tilde{x}_h\|_X &\leq \frac{2 \cdot \|e\| + 1}{\alpha_0(e)} \cdot [\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| + \|l - l_h\|_{Y'_h}] \cdot (\|x\|_{X_{\text{stg}}} + 1), \\ \|\iota_{X_{\text{wk}}}(x - \tilde{x}_h)\|_{X_{\text{wk}}} &\leq \frac{2 \cdot \|e\|^2 \cdot C_S + \|\iota_{X_{\text{wk}}}\|}{\alpha_0(e)} \cdot (\|x\|_{X_{\text{stg}}} + 1) \\ &\quad \cdot [\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \cdot \|(\text{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\| + \|l - l_h\|_{Y'_h}]. \end{aligned}$$

Für die Berechnung der Koeffizienten von $\tilde{x}_h$ muss nun der Last-Vektor $\mathbf{l}_h := (l_h(y_i))_{i=1}^n \in \mathbb{R}^n$ verwendet werden.

Beweis:

Schritt 1: Existenz und Eindeutigkeit von $\tilde{x}_h$

Die Bijektivität von $E_h : X_h \longrightarrow Y'_h$ garantiert auch hier wieder die eindeutige Existenz einer Lösung $\tilde{x}_h \in X_h$ von

$$[e(\tilde{x}_h, \cdot) = l_h \quad \text{auf } Y_h].$$

□

Schritt 2: Abschätzung von $\|x - \tilde{x}_h\|_X$

Aus

$$\alpha_h(e) \cdot \underbrace{\|x_h - \tilde{x}_h\|_X}_{\in X_h} \leq \sup_{y \in Y_h^\times} \frac{e(x_h - \tilde{x}_h, y)}{\|y\|_Y} = \sup_{y \in Y_h^\times} \frac{(l - l_h)(y)}{\|y\|_Y} = \|l - l_h\|_{Y_h'}$$

folgt

$$\|x - \tilde{x}_h\|_X \leq \|x - x_h\|_X + \|x_h - \tilde{x}_h\|_X \leq \|x - x_h\|_X + \frac{1}{\alpha_0(e)} \cdot \|l - l_h\|_{Y_h'} \xrightarrow{h \rightarrow 0} 0.$$

Im Falle $x \in X_{\text{stg}}$ zeigt die Fehler-Abschätzung aus Satz 2.2.3.1 bereits

$$\|x - \tilde{x}_h\|_X \leq \frac{2 \cdot \|e\| + 1}{\alpha_0(e)} \cdot [\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| + \|l - l_h\|_{Y_h'}] \cdot (\|x\|_{X_{\text{stg}}} + 1).$$

□

Schritt 3: Abschätzung von $\|\iota_{X_{\text{wk}}}(x - \tilde{x}_h)\|_{X_{\text{wk}}}$

Wir haben

$$\|\iota_{X_{\text{wk}}}(x_h - \tilde{x}_h)\|_{X_{\text{wk}}} \leq \|\iota_{X_{\text{wk}}}\| \cdot \|x_h - \tilde{x}_h\|_X \stackrel{\text{S.2}}{\leq} \|\iota_{X_{\text{wk}}}\| \cdot \frac{1}{\alpha_0(e)} \cdot \|l - l_h\|_{Y_h'}$$

und damit

$$\begin{aligned} \|\iota_{X_{\text{wk}}}(x - \tilde{x}_h)\|_{X_{\text{wk}}} &\leq \|\iota_{X_{\text{wk}}}(x - x_h)\|_{X_{\text{wk}}} + \|\iota_{X_{\text{wk}}}(x_h - \tilde{x}_h)\|_{X_{\text{wk}}} \\ &\stackrel{\text{S.2.2.4.1}}{\leq} \frac{2 \cdot \|e\|^2 \cdot C_S + \|\iota_{X_{\text{wk}}}\|}{\alpha_0(e)} \cdot (\|x\|_{X_{\text{stg}}} + 1) \\ &\quad \cdot [\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \cdot \|(\text{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\| + \|l - l_h\|_{Y_h'}]. \end{aligned}$$

□

■

2.2.7 Bemerkungen

Bemerkung 2.2.7.1 (Variations-Problem Bemerkungen).

Man beachte:

- Ist die Einbettung $\text{id}_{X_{\text{stg}} \rightarrow X}$ stetig, dann ist sie auf Grund der Voraussetzung $\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0$ schon kompakt (vgl. Satz 2.1.1.1). Insbesondere kann im Falle $\dim X = \infty$ *nicht* $X_{\text{stg}} = X$ gelten. Gleiches gilt für $Y_{\text{stg}}^t \leq Y$.
- Sei $\text{id}_{Y_{\text{stg}}^t \rightarrow Y}$ stetig (und damit bereits kompakt). Dann ist $\iota_{X_{\text{wk}}}$ auf Grund der Voraussetzungen an S bereits kompakt. Insbesondere kann dann bei $\dim X = \infty$ *nicht* $X = X_{\text{wk}}$ gelten.

Beweis: (der Kompaktheit von $\iota_{X_{\text{wk}}}$)

Sei $(x_n)_{n \in \mathbb{N}} \subseteq X$ eine Folge mit $\sup_{n \in \mathbb{N}} \|x_n\|_X < \infty$. Dann ist die Folge $(Sx_n)_{n \in \mathbb{N}} \subseteq \text{ran}(S) \subseteq Y_{\text{stg}}^t$ ebenfalls beschränkt, denn:

$$\forall n \in \mathbb{N} : \quad \|Sx_n\|_{Y_{\text{stg}}^t} \leq C_S \cdot \|\iota_{X_{\text{wk}}} x_n\|_{X_{\text{wk}}} \leq C_S \cdot \|\iota_{X_{\text{wk}}}\| \cdot \sup_{m \in \mathbb{N}} \|x_m\|_X < \infty.$$

Auf Grund der Kompaktheit der Einbettung $\text{id}_{Y_{\text{stg}}^t \rightarrow Y}$ gibt es eine Teilfolgen-Indizierung $n : \mathbb{N} \rightarrow \mathbb{N}$ und ein $y \in Y$ mit

$$Sx_{n(k)} \xrightarrow[\|\cdot\|_Y]{k \rightarrow \infty} y.$$

Nun ist $(\iota_{X_{\text{wk}}} x_{n(k)})_{k \in \mathbb{N}}$ eine $\|\cdot\|_{X_{\text{wk}}}$ -Cauchy-Folge (und damit konvergent), denn

$$\begin{aligned} \|\iota_{X_{\text{wk}}} x_{n(k)} - \iota_{X_{\text{wk}}} x_{n(l)}\|_{X_{\text{wk}}}^2 &= \langle \underbrace{\iota_{X_{\text{wk}}} (x_{n(k)} - x_{n(l)})}_{\in X}, \underbrace{\iota_{X_{\text{wk}}} (x_{n(k)} - x_{n(l)})}_{\in X} \rangle_{X_{\text{wk}}} \\ &\stackrel{\text{Def. } S}{=} e(x_{n(k)} - x_{n(l)}, S(x_{n(k)} - x_{n(l)})) \\ &\leq \|e\| \cdot \|x_{n(k)} - x_{n(l)}\|_X \cdot \|S(x_{n(k)} - x_{n(l)})\|_Y \\ &\leq \|e\| \cdot [\|x_{n(k)}\|_X + \|x_{n(l)}\|_X] \\ &\quad \cdot [\|Sx_{n(k)} - y\|_Y + \|y - Sx_{n(l)}\|_Y] \xrightarrow{(k,l) \rightarrow \infty} 0. \end{aligned}$$

■

2.3 Sattelpunkt-Probleme

2.3.1 (Kontinuierliche) Sattelpunkt-Probleme

Satz 2.3.1.1 (Ein (kontinuierliches) Sattelpunkt-Problem).

Seien V, P Hilbert-Räume, $c : V \times V \rightarrow \mathbb{R}$ und $d : V \times P \rightarrow \mathbb{R}$ stetige Bilinearformen, $r \in V'$ und $s \in P'$. Es bezeichne $D : V \rightarrow P'$ den von d induzierten Operator.

Es gelte:

- *) c erfüllt $\alpha(c|_{\ker(D)^2}) > 0$ und der von $c|_{\ker(D)^2}$ induzierte Operator $C_{\ker}^t : \ker(D) \rightarrow \ker(D)'$ ist injektiv.
- *) d erfüllt $\alpha^t(d) > 0$.

Dann existiert eine eindeutige (kontinuierliche) Lösung des folgenden (kontinuierlichen) Sattelpunkt-Problems:

$$\text{Finde } (v, p) \in V \times P \text{ mit :} \quad \begin{aligned} c(v, \cdot) + d(\cdot, p) &= r \quad \text{auf } V, \\ d(v, \cdot) &= s \quad \text{auf } P. \end{aligned}$$

Die Lösung (v, p) hängt linear und stetig von den Daten (r, s) ab mit $\|(v, p)\|_{V \times P} \leq \frac{6(\|c\| + \|d\|)^2}{\alpha(c|_{\ker(D)^2})\alpha^t(d)^2} \cdot \|(r, s)\|_{V' \times P'}$.

Beweis:

Schritt 1: Rückführung auf Lemma 2.2.1.1

Das kontinuierliche Sattelpunkt-Problem ist äquivalent zu dem kontinuierlichen Variations-Problem, das nach Summation der beiden Zeilen entsteht. Wir können also Lemma 2.2.1.1 auf folgende Situation anwenden:

$$\begin{aligned} X := Y &:= V \times P, \\ \forall (v, p), (w, q) \in V \times P : \quad e((v, p), (w, q)) &:= c(v, w) + d(w, p) + d(v, q), \\ \forall (w, q) \in V \times P : \quad l(w, q) &:= r(w) + s(q), \\ x &= (v, p) \quad (\text{kontin. Lösung}). \end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Lemma 2.2.1.1:

Schritt 2: X, Y sind Hilbert-Räume, e ist stetige Bilinearform, l ist stetige Linearform

Es sind e eine stetige Bilinearform mit $\|e\| \leq \|c\| + 2 \cdot \|d\|$ und $l \in Y'$ mit $\|l\|_{(V \times P)'} = \|r\|_{V'} + \|s\|_{P'} \leq 2 \cdot \|(r, s)\|_{V' \times P'}$.

□

Schritt 3: $\alpha(e) > 0$

Sei $(v, p) \in V \times P$ beliebig. Wir zerlegen

$$v = v_{\ker} + v_{\perp} \in \ker(D) \oplus \ker(D)^{\perp} = V.$$

Der Anteil $v_{\perp} \in \ker(D)^{\perp}$ erfüllt⁶

$$\begin{aligned} \alpha^t(d) \cdot \|v_{\perp}\|_V &\stackrel{\text{S.2.1.2.2}}{\leq} \|Dv_{\perp}\|_{P'} \\ &= \|Dv\|_{P'} \\ &= \sup_{q \in P^{\times}} \frac{d(v, q)}{\|q\|_P} \\ &= \sup_{q \in P^{\times}} \frac{e((v, p), (0, q))}{\|(0, q)\|_{V \times P}} \\ &\leq \sup_{(w, q) \in (V \times P)^{\times}} \frac{e((v, p), (w, q))}{\|(w, q)\|_{V \times P}} \\ &= \|E(v, p)\|_{(V \times P)'}. \end{aligned}$$

Der Anteil $v_{\ker} \in \ker(D)$ erfüllt ($C_{\ker} : \ker(D) \rightarrow \ker(D)'$ ist der von $c|_{\ker(D)^2}$ induzierte Operator)

$$\begin{aligned} \alpha(c|_{\ker(D)^2}) \cdot \|v_{\ker}\|_V &\leq \|C_{\ker} v_{\ker}\|_{\ker(D)'} \\ &= \sup_{w \in \ker(D)^{\times}} \frac{c(v_{\ker}, w)}{\|w\|_V} \\ &= \sup_{w \in \ker(D)^{\times}} \frac{e((v, p), (w, 0)) - c(v_{\perp}, w) - \overbrace{d(w, p)}^{=0}}{\|(w, 0)\|_{V \times P}} \\ &\leq \|c\| \cdot \|v_{\perp}\|_V + \sup_{(w, q) \in (V \times P)^{\times}} \frac{e((v, p), (w, q))}{\|(w, q)\|_{V \times P}} \\ &= \|c\| \cdot \|v_{\perp}\|_V + \|E(v, p)\|_{(V \times P)'}. \end{aligned}$$

Die zweite Komponente $p \in P$ erfüllt ($D^t : P \rightarrow V'$ ist der von d induzierte Operator)

$$\begin{aligned} \alpha^t(d) \cdot \|p\|_P &\leq \|D^t p\|_{V'} \\ &= \sup_{w \in V^{\times}} \frac{d(w, p)}{\|w\|_V} \\ &= \sup_{w \in V^{\times}} \frac{e((v, p), (w, 0)) - c(v, w)}{\|(w, 0)\|_{V \times P}} \\ &\leq \|c\| \cdot \|v\|_V + \sup_{(w, q) \in (V \times P)^{\times}} \frac{e((v, p), (w, q))}{\|(w, q)\|_{V \times P}} \\ &\leq \|c\| \cdot \|v\|_V + \|E(v, p)\|_{(V \times P)'}. \end{aligned}$$

Mit

$$\begin{aligned} \|v\|_V &\leq \|v_{\perp}\|_V + \|v_{\ker}\|_V \\ &\leq \left(1 + \underbrace{\frac{\|c\|}{\alpha(c|_{\ker(D)^2})}}_{\geq 1}\right) \cdot \|v_{\perp}\|_V + \frac{1}{\alpha(c|_{\ker(D)^2})} \cdot \|E(v, p)\|_{(V \times P)'} \\ &\leq \left[2 \cdot \frac{\|c\|}{\alpha(c|_{\ker(D)^2})} \cdot \frac{1}{\alpha^t(d)} + \frac{1}{\alpha(c|_{\ker(D)^2})}\right] \cdot \|E(v, p)\|_{(V \times P)'} \\ &\leq 2 \cdot \left(1 + \frac{\|c\|}{\alpha^t(d)}\right) \cdot \frac{1}{\alpha(c|_{\ker(D)^2})} \cdot \|E(v, p)\|_{(V \times P)'} \end{aligned}$$

⁶Man beachte: $E(v, p) = E((v, p)) \in (V \times P)'$ ist das Bild des Vektors $(v, p) \in V \times P$ unter der linearen Abbildung $E : V \times P \rightarrow (V \times P)'$. Hier ist natürlich *keine* Bilinearform gemeint.

ergibt sich

$$\begin{aligned}
\|(v, p)\|_{V \times P} &\leq \|v\|_V + \|p\|_P \\
&\leq \left(1 + \frac{\|c\|}{\alpha^t(d)}\right) \cdot \|v\|_V + \frac{1}{\alpha^t(d)} \cdot \|E(v, p)\|_{(V \times P)'} \\
&\leq \left[2 \cdot \left(1 + \frac{\|c\|}{\alpha^t(d)}\right)^2 \cdot \frac{1}{\alpha(c|_{\ker(D)^2})} + \frac{1}{\alpha^t(d)} \cdot \underbrace{\frac{\|c\|}{\alpha(c|_{\ker(D)^2})}}_{\geq 1}\right] \cdot \|E(v, p)\|_{(V \times P)'} \\
&\leq \frac{3}{\alpha(c|_{\ker(D)^2})} \cdot \left(1 + \frac{\|c\|}{\alpha^t(d)}\right)^2 \cdot \|E(v, p)\|_{(V \times P)'} \\
&\stackrel{\alpha^t(d) \leq \|d\|}{\leq} \frac{3 \cdot (\|c\| + \|d\|)^2}{\alpha(c|_{\ker(D)^2}) \cdot \alpha^t(d)^2} \cdot \|E(v, p)\|_{(V \times P)'}
\end{aligned}$$

und schließlich

$$\alpha(e) = \inf_{(v, p) \in (V \times P)^\times} \frac{\|E(v, p)\|_{(V \times P)'}}{\|(v, p)\|_{V \times P}} \geq \frac{\alpha(c|_{\ker(D)^2}) \cdot \alpha^t(d)^2}{3 \cdot (\|c\| + \|d\|)^2} > 0.$$

□

Schritt 4: $E^t : Y \longrightarrow X'$ ist injektiv

Sei $(w, q) \in V \times P$ ein Element mit

$$\forall (v, p) \in V \times P : \quad 0 = (E^t(w, q))(v, p) = e((v, p), (w, q)) = c(v, w) + d(w, p) + d(v, q).$$

Durch Einsetzen von $v := 0$ sieht man $w \in \ker(D)$, was dann wegen

$$\forall v \in \ker(D) : \quad 0 = c(v, w) + \underbrace{d(v, q)}_{=0} = (C_{\ker}^t w)(v)$$

und der Injektivität von $C_{\ker}^t$ schon in $w = 0$ resultiert. Die verbleibende Identität $0 = d(\cdot, q) = D^t q$ liefert zusammen mit der Injektivität von $D^t : P \longrightarrow V'$ schließlich $q = 0$.

□

Also existiert eine eindeutige Lösung $x = (v, p) \in V \times P$ des kontinuierlichen Sattelpunkt-Problems. Die Abschätzung für $\|(v, p)\|_{V \times P}$ ergibt sich unmittelbar aus jener von $\|x\|_X$ in Lemma 2.2.1.1.

■

2.3.2 Diskretisierung

Lemma 2.3.2.1 (Sattelpunkt-Problem Diskretisierung).

Seien nun zusätzlich $V_h \leq V$ und $P_h \leq P$ Ansatz-Räume mit folgenden Eigenschaften ($D_h : V_h \longrightarrow P_h'$ ist der von $d|_{V_h \times P_h}$ induzierte Operator):

- *) $\dim V_h < \infty$ und $\dim P_h < \infty$.
- *) c erfüllt $\alpha_h(c|_{\ker(D_h)^2}) > 0$.
- *) d erfüllt $\alpha_h^t(d) > 0$.

Dann existiert auch eine eindeutige (diskrete) Lösung $(v_h, p_h) \in V_h \times P_h$ des (diskreten) Sattelpunkt-Problems:

$$\begin{aligned}
\text{Finde } (v_h, p_h) \in V_h \times P_h \text{ mit :} \quad & c(v_h, \cdot) + d(\cdot, p_h) = r \quad \text{auf } V_h, \\
& d(v_h, \cdot) = s \quad \text{auf } P_h.
\end{aligned}$$

Beweis:

Schritt 1: Rückführung auf Lemma 2.2.2.1

Das diskrete Sattelpunkt-Problem ist äquivalent zu dem diskreten Variations-Problem, das nach Summation der beiden Zeilen entsteht. Wir wenden also Lemma 2.2.2.1 auf folgende Situation an:

$$\begin{aligned} X_h := Y_h &:= V_h \times P_h, \\ x_h &= (v_h, p_h) \quad (\text{diskr. Lösung}). \end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Lemma 2.2.2.1:

Schritt 2: $\dim X_h = \dim Y_h < \infty$, $\alpha_h(e) > 0$

Es gilt klarerweise $\dim X_h = \dim Y_h = \dim(V_h \times P_h) = \dim V_h + \dim P_h < \infty$.

Analog zum Beweis von Satz 2.3.1.1 ergibt sich aus den Voraussetzungen $\alpha_h(c|_{\ker(D_h)^2}) > 0$ und $\alpha_h^t(d) > 0$ die Abschätzung

$$\alpha_h(e) \geq \frac{\alpha_h(c|_{\ker(D_h)^2}) \cdot \alpha_h^t(d)^2}{3 \cdot (\|c\| + \|d\|)^2} > 0.$$

□

Also existiert eine eindeutige Lösung $x_h = (v_h, p_h) \in V_h \times P_h$ des diskreten Sattelpunkt-Problems.

■

2.3.3 Konvergenz

Satz 2.3.3.1 (Sattelpunkt-Problem Konvergenz).

Seien V_{stg} und P_{stg} weitere Hilbert-Räume mit folgenden Eigenschaften:

- *) $V_{\text{stg}} \leq V$ und $P_{\text{stg}} \leq P$ jeweils dicht.
- *) Für die Orthogonal-Projektionen $P_{V_h} : V \rightarrow V_h$ und $P_{P_h} : P \rightarrow P_h$ gilt

$$\|(\text{id} - P_{V_h})|_{V_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0, \quad \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0.$$

Für die Ansatz-Räume gelte zusätzlich

- *) c erfüllt $\alpha_0(c) := \inf_{h \in H} \alpha_h(c|_{\ker(D_h)^2}) > 0$.
- *) d erfüllt $\alpha_0^t(d) > 0$.

Dann gilt

$$(v_h, p_h) \xrightarrow[\|\cdot\|_{V \times P}]{h \rightarrow 0} (v, p)$$

und unter der Voraussetzung

- *) $(v, p) \in V_{\text{stg}} \times P_{\text{stg}}$

sogar

$$\|v - v_h\|_V + \|p - p_h\|_P \leq \frac{24 \cdot (\|c\| + \|d\|)^3}{\alpha_0(c) \cdot \alpha_0^t(d)^2} \cdot (\|(\text{id} - P_{V_h})|_{V_{\text{stg}}}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\|) \cdot \|(v, p)\|_{V_{\text{stg}} \times P_{\text{stg}}}.$$

Beweis:

Schritt 1: Rückführung auf Satz 2.2.3.1

Wir wenden Satz 2.2.3.1 auf folgende Situation an:

$$X_{\text{stg}} := V_{\text{stg}} \times P_{\text{stg}}.$$

□

Wir zeigen jetzt die Voraussetzungen von Satz 2.2.3.1:

Schritt 2: $X_{\text{stg}} \leq X$ dicht, $\|(\text{id} - P_{X_h})|_{X_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0$, $\alpha_0(e) > 0$

Es ist $X_{\text{stg}} = V_{\text{stg}} \times P_{\text{stg}} \leq V \times P = X$ dicht.

Die Abbildung

$$P_{V_h \times P_h} : \begin{cases} V \times P & \longrightarrow & V_h \times P_h \\ (v, p) & \longmapsto & (P_{V_h} v, P_{P_h} p) \end{cases}$$

ist wegen $(V_h \times P_h)^\perp = V_h^\perp \times P_h^\perp$ genau die Orthogonal-Projektion auf $V_h \times P_h$. Aus

$$\begin{aligned} \forall (v, p) \in V_{\text{stg}} \times P_{\text{stg}} : \\ \|(\text{id} - P_{V_h \times P_h})|_{V_{\text{stg}} \times P_{\text{stg}}} (v, p)\|_{V \times P}^2 &= \| ((\text{id} - P_{V_h})|_{V_{\text{stg}}} v, (\text{id} - P_{P_h})|_{P_{\text{stg}}} p) \|_{V \times P}^2 \\ &= \| (\text{id} - P_{V_h})|_{V_{\text{stg}}} v \|_V^2 + \| (\text{id} - P_{P_h})|_{P_{\text{stg}}} p \|_P^2 \\ &\leq \| (\text{id} - P_{V_h})|_{V_{\text{stg}}} \|^2 \cdot \|v\|_{V_{\text{stg}}}^2 + \| (\text{id} - P_{P_h})|_{P_{\text{stg}}} \|^2 \cdot \|p\|_{P_{\text{stg}}}^2 \\ &\leq (\| (\text{id} - P_{V_h})|_{V_{\text{stg}}} \| + \| (\text{id} - P_{P_h})|_{P_{\text{stg}}} \|)^2 \cdot \| (v, p) \|_{V_{\text{stg}} \times P_{\text{stg}}}^2 \end{aligned}$$

folgt

$$\| (\text{id} - P_{V_h \times P_h})|_{V_{\text{stg}} \times P_{\text{stg}}} \| \leq \| (\text{id} - P_{V_h})|_{V_{\text{stg}}} \| + \| (\text{id} - P_{P_h})|_{P_{\text{stg}}} \| \xrightarrow{h \rightarrow 0} 0.$$

Außerdem

$$\alpha_0(e) = \inf_{h \in H} \alpha_h(e) \stackrel{\text{L.2.3.2.1}}{\geq} \inf_{h \in H} \frac{\alpha_h(c|_{\ker(D_h)^2}) \cdot \alpha_h^t(d)^2}{3 \cdot (\|c\| + \|d\|)^2} \geq \frac{\alpha_0(c) \cdot \alpha_0^t(d)^2}{3 \cdot (\|c\| + \|d\|)^2} > 0.$$

□

Also konvergiert $(v_h, p_h) \xrightarrow[\|\cdot\|_{V \times P}]{h \rightarrow 0} (v, p)$. Die Abschätzung für $\|v - v_h\|_V + \|p - p_h\|_P$ ergibt sich unmittelbar aus jener für $\|x - x_h\|_X$ in Satz 2.2.3.1. ■

2.3.4 Aubin-Nitsche-Trick

Satz 2.3.4.1 (Sattelpunkt-Problem Aubin-Nitsche-Trick).

Seien^a V_{wk} , P_{wk} und V_{stg}^t , P_{stg}^t weitere Hilbert-Räume mit folgenden Eigenschaften:

- *) $\iota_{V_{\text{wk}}} : V \longrightarrow V_{\text{wk}}$ und $\iota_{P_{\text{wk}}} : P \longrightarrow P_{\text{wk}}$ jeweils linear und stetig.
- *) $V_{\text{stg}}^t \leq V$ und $P_{\text{stg}}^t \leq P$.
- *) Für die Orthogonal-Projektionen $P_{V_h} : V \longrightarrow V_h$ und $P_{P_h} : P \longrightarrow P_h$ gilt

$$\| (\text{id} - P_{V_h})|_{V_{\text{stg}}^t} \| \xrightarrow{h \rightarrow 0} 0, \quad \| (\text{id} - P_{P_h})|_{P_{\text{stg}}^t} \| \xrightarrow{h \rightarrow 0} 0.$$

Die Voraussetzungen an c und d garantieren auch die eindeutige Lösbarkeit des folgenden (*kontinu-*

ierlichen) transponierten Sattelpunkt-Problems:

$$\begin{aligned} \text{Für geg. } (v, p) \in V \times P, \text{ finde } (w, q) \in V \times P \text{ mit :} \\ c(\cdot, w) + d(\cdot, q) &= \langle \iota_{V_{\text{wk}}}(\cdot), \iota_{V_{\text{wk}}} v \rangle_{V_{\text{wk}}} \quad \text{auf } V, \\ d(w, \cdot) &= \langle \iota_{P_{\text{wk}}}(\cdot), \iota_{P_{\text{wk}}} p \rangle_{P_{\text{wk}}} \quad \text{auf } P. \end{aligned}$$

Der zugehörige Lösungs-Operator $S : V \times P \longrightarrow V \times P$ ist linear, stetig und erfüllt

$$\forall (v, p) \in V \times P : \|S(v, p)\|_{V \times P} \leq \frac{3 \cdot (\|c\| + \|d\|)^2 \cdot (\|\iota_{V_{\text{wk}}}\| + \|\iota_{P_{\text{wk}}}\|)}{\alpha(c|_{\ker(D)^2}) \cdot \alpha^t(d)^2} \cdot \|(\iota_{V_{\text{wk}}} v, \iota_{P_{\text{wk}}} p)\|_{V_{\text{wk}} \times P_{\text{wk}}}.$$

Erfüllt S darüber hinaus die Voraussetzungen

$$*) \operatorname{ran}(S) \subseteq V_{\text{stg}}^t \times P_{\text{stg}}^t \text{ und}$$

$$\forall (v, p) \in V \times P : \|S(v, p)\|_{V_{\text{stg}}^t \times P_{\text{stg}}^t} \leq C_S \cdot \|(\iota_{V_{\text{wk}}} v, \iota_{P_{\text{wk}}} p)\|_{V_{\text{wk}} \times P_{\text{wk}}},$$

dann gilt im Falle

$$*) (v, p) \in V_{\text{stg}} \times P_{\text{stg}}$$

auch die Fehler-Abschätzung

$$\begin{aligned} \|\iota_{V_{\text{wk}}}(v - v_h)\|_{V_{\text{wk}}} + \|\iota_{P_{\text{wk}}}(p - p_h)\|_{P_{\text{wk}}} &\leq \frac{48 \cdot (\|c\| + \|d\|)^4 \cdot C_S}{\alpha_0(c) \cdot \alpha_0^t(d)^2} \cdot \|(v, p)\|_{V_{\text{stg}} \times P_{\text{stg}}} \\ &\quad \cdot (\|(\operatorname{id} - P_{V_h})|_{V_{\text{stg}}}\| + \|(\operatorname{id} - P_{P_h})|_{P_{\text{stg}}}\|) \\ &\quad \cdot (\|(\operatorname{id} - P_{V_h})|_{V_{\text{stg}}^t}\| + \|(\operatorname{id} - P_{P_h})|_{P_{\text{stg}}^t}\|). \end{aligned}$$

^aWir verstehen die Räume V_{stg}^t und P_{stg}^t mit dem Index „t“, um die Zugehörigkeit zum *transponierten* Sattelpunkt-Problem zu verdeutlichen. Wir fordern aber *nicht* zwingend einen Zusammenhang mit den Räumen V_{stg} und P_{stg} von oben.

Beweis:

Schritt 1: Rückführung auf Satz 2.2.4.1

Wir wenden Satz 2.2.4.1 auf folgende Situation an:

$$\begin{aligned} X_{\text{wk}} &:= V_{\text{wk}} \times P_{\text{wk}}, \\ Y_{\text{stg}}^t &:= V_{\text{stg}}^t \times P_{\text{stg}}^t, \\ S &= S. \end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Satz 2.2.4.1:

Schritt 2: $X_{\text{wk}}, Y_{\text{stg}}^t$ Hilbert-Räume, $\|X_{\text{wk}}\| < \infty$, $\|(\operatorname{id} - P_{Y_h})|_{Y_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0$, Voraussetzungen an S

Es sind $X_{\text{wk}} = V_{\text{wk}} \times P_{\text{wk}}$ und $Y_{\text{stg}}^t = V_{\text{stg}}^t \times P_{\text{stg}}^t$ Hilbert-Räume und

$$\iota_{V_{\text{wk}} \times P_{\text{wk}}} : \begin{cases} V \times P & \longrightarrow V_{\text{wk}} \times P_{\text{wk}} \\ (v, p) & \longmapsto (\iota_{V_{\text{wk}}} v, \iota_{P_{\text{wk}}} p) \end{cases}$$

linear und stetig mit $\|\iota_{V_{\text{wk}} \times P_{\text{wk}}}\| \leq \|\iota_{V_{\text{wk}}}\| + \|\iota_{P_{\text{wk}}}\|$.

Analog zum Beweis von Satz 2.3.3.1 sieht man

$$\|(\operatorname{id} - P_{V_h \times P_h})|_{V_{\text{stg}}^t \times P_{\text{stg}}^t}\| \leq \|(\operatorname{id} - P_{V_h})|_{V_{\text{stg}}^t}\| + \|(\operatorname{id} - P_{P_h})|_{P_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0.$$

Der Lösungs-Operator $S : V \times P \longrightarrow V \times P$ stimmt mit dem Lösungs-Operator $S : X \longrightarrow Y$ aus Satz 2.2.4.1 überein und erfüllt $\operatorname{ran}(S) \subseteq V_{\text{stg}}^t \times P_{\text{stg}}^t = Y_{\text{stg}}^t$ sowie

$$\forall (v, p) \in V \times P : \|S(v, p)\|_{V_{\text{stg}}^t \times P_{\text{stg}}^t} \leq C_S \cdot \|\iota_{V_{\text{wk}} \times P_{\text{wk}}}(v, p)\|_{V_{\text{wk}} \times P_{\text{wk}}}.$$

□

Die Abschätzungen für $\|S(\cdot, \cdot)\|_{V \times P}$ und $\|\iota_{V_{\text{wk}}}(v - v_h)\|_{V_{\text{wk}}} + \|\iota_{P_{\text{wk}}}(p - p_h)\|_{P_{\text{wk}}}$ ergeben sich unmittelbar aus jenen für $\|S(\cdot)\|_Y$ und $\|\iota_{X_{\text{wk}}}(x - x_h)\|_{X_{\text{wk}}}$ in Satz 2.2.4.1.

■

2.3.5 LGS

Lemma 2.3.5.1 (Sattelpunkt-Problem LGS).

Seien nun $\{v_1, \dots, v_{n_V}\} \subseteq V_h$ und $\{p_1, \dots, p_{n_P}\} \subseteq P_h$ Basen. Wir bezeichnen mit

$$\nu_h : \begin{cases} V_h & \longrightarrow \mathbb{R}^{n_V} \\ \sum_{i=1}^{n_V} \nu_i v_i & \longmapsto (\nu_i)_{i=1}^{n_V} \end{cases}, \quad \pi_h : \begin{cases} P_h & \longrightarrow \mathbb{R}^{n_P} \\ \sum_{j=1}^{n_P} \pi_j p_j & \longmapsto (\pi_j)_{j=1}^{n_P} \end{cases}$$

die zugehörigen Koordinaten-Abbildungen. Die Koeffizienten $(\nu_k)_{k=1}^{n_V} \in \mathbb{R}^{n_V}$ und $(\pi_l)_{l=1}^{n_P} \in \mathbb{R}^{n_P}$ aus den Darstellungen

$$v_h = \sum_{k=1}^{n_V} \nu_k v_k, \quad p_h = \sum_{l=1}^{n_P} \pi_l p_l$$

lassen sich durch Lösen eines regulären LGS bestimmen: Mit

$$\begin{aligned} \mathbf{C} &:= (c(v_k, v_i))_{i=1, k=1}^{n_V, n_V} \in \mathbb{R}^{n_V \times n_V}, \\ \mathbf{D} &:= (d(v_k, p_j))_{j=1, k=1}^{n_P, n_V} \in \mathbb{R}^{n_P \times n_V}, \\ \boldsymbol{\nu} &:= (\nu_k)_{k=1}^{n_V} \in \mathbb{R}^{n_V}, \\ \boldsymbol{\pi} &:= (\pi_l)_{l=1}^{n_P} \in \mathbb{R}^{n_P}, \\ \mathbf{r} &:= (r(v_i))_{i=1}^{n_V} \in \mathbb{R}^{n_V}, \\ \mathbf{s} &:= (s(p_j))_{j=1}^{n_P} \in \mathbb{R}^{n_P} \end{aligned}$$

gilt:

$$\begin{pmatrix} \mathbf{C} & \mathbf{D}^T \\ \mathbf{D} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\nu} \\ \boldsymbol{\pi} \end{pmatrix} = \begin{pmatrix} \mathbf{r} \\ \mathbf{s} \end{pmatrix}.$$

Die Konditionszahl der System-Matrix erfüllt die Abschätzung

$$\text{cond} \begin{pmatrix} \mathbf{C} & \mathbf{D}^T \\ \mathbf{D} & \mathbf{0} \end{pmatrix} \leq \frac{24 \cdot (\|c\| + \|d\|)^3}{\alpha_0(c) \cdot \alpha_0^t(d)^2} \cdot (\|\nu_h\|^2 + \|\pi_h\|^2) \cdot (\|\nu_h^{-1}\|^2 + \|\pi_h^{-1}\|^2).$$

Es gelte zusätzlich:

*) c ist symmetrisch und koerziv auf ganz V .

Dann ist die System-Matrix symmetrisch indefinit mit genau n_V positiven und genau n_P negativen Eigenwerten.

Beweis:

Schritt 1: Rückführung auf Lemma 2.2.5.1

Wir wenden Lemma 2.2.5.1 auf folgende Situation an:

$$\{x_1, \dots, x_n\} := \{y_1, \dots, y_n\} := \{(v_i, 0) \mid i \in \{1, \dots, n_V\}\} \cup \{(0, p_j) \mid j \in \{1, \dots, n_P\}\}.$$

□

Wir zeigen jetzt die Voraussetzungen von Lemma 2.2.5.1:

Schritt 2: $\{x_1, \dots, x_n\} \subseteq X_h$ und $\{y_1, \dots, y_n\} \subseteq Y_h$ sind Basen, Abschätzung Konditionszahl
Klarerweise ist $\{(v_i, 0) \mid i \in \{1, \dots, n_V\}\} \cup \{(0, p_j) \mid j \in \{1, \dots, n_P\}\} \subseteq V_h \times P_h$ eine Basis.

Das LGS aus Lemma 2.2.5.1 nimmt genau die angegebene Form an.

Nun zur Abschätzung der Konditionszahl: Die Koordinaten-Abbildung zur obigen Basis von $V_h \times P_h$ lässt sich mit Hilfe der Koordinaten-Abbildungen ν_h und π_h explizit angeben:

$$\chi_h : \begin{cases} V_h \times P_h & \longrightarrow \mathbb{R}^{n_V + n_P} \\ (v, p) & \longmapsto (\nu_h v, \pi_h p) \end{cases}.$$

Wegen

$$\forall (v, p) \in V_h \times P_h : \quad \|\chi_h(v, p)\|_{l^2}^2 = \|(\nu_h v, \pi_h p)\|_{l^2}^2 = \|\nu_h v\|_{l^2}^2 + \|\pi_h p\|_{l^2}^2 \leq (\|\nu_h\| + \|\pi_h\|)^2 \cdot \|(v, p)\|_{V \times P}^2$$

ist $\|\chi_h\| \leq \|\nu_h\| + \|\pi_h\|$ und wegen

$$\forall (\nu, \pi) \in \mathbb{R}^{n_V + n_P} :$$

$$\|\chi_h^{-1}(\nu, \pi)\|_{V \times P}^2 = \|(\nu_h^{-1} \nu, \pi_h^{-1} \pi)\|_{V \times P}^2 = \|\nu_h^{-1} \nu\|_V^2 + \|\pi_h^{-1} \pi\|_P^2 \leq (\|\nu_h^{-1}\| + \|\pi_h^{-1}\|)^2 \cdot \|(\nu, \pi)\|_{l^2}^2$$

weilers $\|\chi_h^{-1}\| \leq \|\nu_h^{-1}\| + \|\pi_h^{-1}\|$.

Die Abschätzung für die Konditionszahl der System-Matrix ergibt sich unmittelbar aus jener für $\text{cond}(\mathbf{E})$ in Lemma 2.2.5.1. □

Schritt 3: Symmetrie und Indefinitheit der System-Matrix

Wir bezeichnen im Folgenden mit $C_h : V_h \longrightarrow V'_h$ und $D_h : V_h \longrightarrow P'_h$ die von $c|_{V_h \times V_h}$ bzw. $d|_{V_h \times P_h}$ induzierten Operatoren.

Die Matrix $\mathbf{C} \in \mathbb{R}^{n_V \times n_V}$ ist genau die darstellende Matrix der linearen Abbildung $(\nu_h^{-1})^T \circ \tau_{V_h}^{-1} \circ C_h \circ \nu_h^{-1} : \mathbb{R}^{n_V} \longrightarrow \mathbb{R}^{n_V}$ bezüglich der euklidischen Basis $\{e_1, \dots, e_{n_V}\} \subseteq \mathbb{R}^{n_V}$, denn

$$\forall i, k : \quad \langle ((\nu_h^{-1})^T \circ \tau_{V_h}^{-1} \circ C_h \circ \nu_h^{-1})(e_k), e_i \rangle_2 = \langle (\tau_{V_h}^{-1} \circ C_h)(v_k), v_i \rangle_V = (C_h v_k)(v_i) = c(v_k, v_i) = \langle \mathbf{C} \cdot e_k, e_i \rangle_2.$$

Weiters ist $\mathbf{C}$ symmetrisch und positiv definit:

$$\forall \nu \in (\mathbb{R}^{n_V})^\times : \quad \langle \mathbf{C} \cdot \nu, \nu \rangle_2 = \langle (\tau_{V_h}^{-1} \circ C_h)(\nu_h^{-1} \nu), \nu_h^{-1} \nu \rangle_V = c(\nu_h^{-1} \nu, \nu_h^{-1} \nu) \geq \alpha_{\text{kzv}}(c) \cdot \|\nu_h^{-1} \nu\|_V^2 > 0.$$

Insbesondere ist auch $\mathbf{C}^{-1} \in \mathbb{R}^{n_V \times n_V}$ SPD.

Die Matrix $\mathbf{D}^T \in \mathbb{R}^{n_V \times n_P}$ ist genau die darstellende Matrix der linearen Abbildung $(\nu_h^{-1})^T \circ \tau_{V_h}^{-1} \circ D_h^t \circ \pi_h^{-1} : \mathbb{R}^{n_P} \longrightarrow \mathbb{R}^{n_V}$ bezüglich den euklidischen Basen $\{e_1, \dots, e_{n_P}\} \subseteq \mathbb{R}^{n_P}$ und $\{e_1, \dots, e_{n_V}\} \subseteq \mathbb{R}^{n_V}$, denn

$$\forall j, k : \quad \langle ((\nu_h^{-1})^T \circ \tau_{V_h}^{-1} \circ D_h^t \circ \pi_h^{-1})(e_j), e_k \rangle_2 = \langle (\tau_{V_h}^{-1} \circ D_h^t)(p_j), v_k \rangle_V = (D_h^t p_j)(v_k) = d(v_k, p_j) = \langle \mathbf{D}^T e_j, e_k \rangle_2.$$

Auf Grund der Voraussetzung $\alpha_h^t(d) > 0$ ist der Operator $D_h^t : P_h \longrightarrow V'_h$ und damit auch die Matrix $\mathbf{D}^T$ injektiv. Insbesondere ist auch die Matrix $\mathbf{D} \mathbf{C}^{-1} \mathbf{D}^T \in \mathbb{R}^{n_P \times n_P}$ SPD.

Aus der Zerlegung

$$\begin{pmatrix} \mathbf{C} & \mathbf{D}^T \\ \mathbf{D} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{D} & \text{id} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{C}^{-1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{D} \mathbf{C}^{-1} \mathbf{D}^T \end{pmatrix} \cdot \begin{pmatrix} \mathbf{C} & \mathbf{D}^T \\ \mathbf{0} & \text{id} \end{pmatrix}$$

und dem Trägheitssatz von Sylvester folgt: Die System-Matrix hat genau $n_V = \text{rowdim}(\mathbf{C}^{-1})$ positive und genau $n_P = \text{rowdim}(-\mathbf{D} \mathbf{C}^{-1} \mathbf{D}^T)$ negative Eigenwerte. □

■

2.3.6 Strang-Lemma

Lemma 2.3.6.1 (Sattelpunkt-Problem Strang-Lemma).

Seien nun zusätzlich $r_h \in V'_h$ und $s_h \in P'_h$ für alle $h \in H$ stetige Linearformen mit

$$*) \quad \|r - r_h\|_{V'_h} \xrightarrow{h \rightarrow 0} 0 \text{ und } \|s - s_h\|_{P'_h} \xrightarrow{h \rightarrow 0} 0.$$

Dann erfüllen auch die eindeutigen Lösungen $(\tilde{v}_h, \tilde{p}_h) \in V_h \times P_h$ der diskreten Sattelpunkt-Probleme

$$\begin{bmatrix} c(\tilde{v}_h, \cdot) + d(\cdot, \tilde{p}_h) & = & r_h & \text{auf } V_h, \\ d(\tilde{v}_h, \cdot) & = & s_h & \text{auf } P_h \end{bmatrix}$$

die Konvergenz $(\tilde{v}_h, \tilde{p}_h) \xrightarrow[\|\cdot\|_{V \times P}]{h \rightarrow 0} (v, p)$ und im Falle

$$*) \quad (v, p) \in V_{\text{stg}} \times P_{\text{stg}}$$

die Fehler-Abschätzungen

$$\begin{aligned} \|v - \tilde{v}_h\|_V + \|p - \tilde{p}_h\|_P &\leq \frac{24 \cdot (\|c\| + \|d\| + 1)^3}{\alpha_0(c) \cdot \alpha_0^t(d)^2} \cdot [\|(\text{id} - P_{V_h})|_{V_{\text{stg}}}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\| \\ &\quad + \|r - r_h\|_{V'_h} + \|s - s_h\|_{P'_h}] \cdot (\|(v, p)\|_{V_{\text{stg}} \times P_{\text{stg}}} + 1) \end{aligned}$$

und^a

$$\begin{aligned} \|\iota_{V_{\text{wk}}}(v - \tilde{v}_h)\|_{V_{\text{wk}}} + \|\iota_{P_{\text{wk}}}(p - \tilde{p}_h)\|_{P_{\text{wk}}} &\leq \frac{48 \cdot (\|c\| + \|d\| + \|\iota_{V_{\text{wk}}}\| + \|\iota_{P_{\text{wk}}}\|)^4 \cdot (C_S + 1)}{\alpha_0(c) \cdot \alpha_0^t(d)^2} \\ &\quad \cdot [\|(\text{id} - P_{V_h})|_{V_{\text{stg}}}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\|] \\ &\quad \cdot [\|(\text{id} - P_{V_h})|_{V_{\text{stg}}^t}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}^t}\|] \\ &\quad + \|r - r_h\|_{V'_h} + \|s - s_h\|_{P'_h} \cdot (\|(v, p)\|_{V_{\text{stg}} \times P_{\text{stg}}} + 1). \end{aligned}$$

Für die Berechnung der Koeffizienten von $\tilde{v}_h$ und $\tilde{p}_h$ muss nun der Last-Vektor bestehend aus $\mathbf{r}_h := (r_h(v_i))_{i=1}^{n_V} \in \mathbb{R}^{n_V}$ und $\mathbf{s}_h := (s_h(p_j))_{j=1}^{n_P} \in \mathbb{R}^{n_P}$ verwendet werden.

^aBei der zweiten Abschätzung setzen wir o.B.d.A. $\|\iota_{V_{\text{wk}}}\| \geq 1$ und $\|\iota_{P_{\text{wk}}}\| \geq 1$ voraus.

Beweis:

Schritt 1: Rückführung auf Lemma 2.2.6.1

Wir wenden Lemma 2.2.6.1 auf folgende Situation an:

$$\begin{aligned} \forall (w, q) \in V_h \times P_h : \quad l_h(w, q) &:= r_h(w) + s_h(q), \\ \tilde{x}_h &= (\tilde{v}_h, \tilde{p}_h) \quad (\text{diskr. Lösung zu } l_h). \end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Lemma 2.2.6.1:

$$\text{Schritt 2: } \|l - l_h\|_{(V_h \times P_h)'} \xrightarrow{h \rightarrow 0} 0$$

Es ist l_h eine stetige Linearform mit $\|l_h\|_{(V_h \times P_h)'} = \|r_h\|_{V'_h} + \|s_h\|_{P'_h}$. Also existiert eine eindeutige Lösung $\tilde{x}_h = (\tilde{v}_h, \tilde{p}_h) \in V_h \times P_h$ des diskreten Sattelpunkt-Problems mit rechter Seite (r_h, s_h) . Es gilt $\|l - l_h\|_{(V_h \times P_h)'} \leq \|r - r_h\|_{V'_h} + \|s - s_h\|_{P'_h} \xrightarrow{h \rightarrow 0} 0$ und damit $(\tilde{v}_h, \tilde{p}_h) \xrightarrow[\|\cdot\|_{V \times P}]{h \rightarrow 0} (v, p)$.

□

Die Abschätzungen für $\|v - \tilde{v}_h\|_V + \|p - \tilde{p}_h\|_P$ und $\|\iota_{V_{\text{wk}}}(v - \tilde{v}_h)\|_{V_{\text{wk}}} + \|\iota_{P_{\text{wk}}}(p - \tilde{p}_h)\|_{P_{\text{wk}}}$ ergeben sich unmittelbar aus jenen für $\|x - \tilde{x}_h\|_X$ und $\|\iota_{X_{\text{wk}}}(x - \tilde{x}_h)\|_{X_{\text{wk}}}$ in Lemma 2.2.6.1. ■

2.3.7 Bemerkungen

Bemerkung 2.3.7.1 (Sattelpunkt-Problem Bemerkungen).

Man beachte:

- Wegen $\alpha_h^t(d) > 0$ ist der von $d|_{V_h \times P_h}$ induzierte Operator $D_h^t : P_h \rightarrow V_h'$ injektiv. Also muss bereits $\dim P_h \leq \dim V_h$ gelten.
- Ist die Einbettung $\text{id}_{V_{\text{stg}} \rightarrow V}$ stetig, dann ist sie auf Grund der Voraussetzung $\|(\text{id} - P_{V_h})|_{V_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0$ schon kompakt (vgl. Satz 2.1.1.1). Insbesondere kann im Falle $\dim V = \infty$ nicht $V_{\text{stg}} = V$ gelten. Gleiches gilt für $P_{\text{stg}} \leq P$, $V_{\text{stg}}^t \leq V$ und $P_{\text{stg}}^t \leq P$.

2.4 Optimal-Steuer-Probleme

2.4.1 (Kontinuierliche) Optimal-Steuer-Probleme

Satz 2.4.1.1 (Ein (kontinuierliches) Optimal-Steuer-Problem).

Seien F, U, P Hilbert-Räume, $a : U \times P \rightarrow \mathbb{R}$ und $b : F \times P \rightarrow \mathbb{R}$ stetige Bilinearformen, $u_0 \in U$ und $\kappa \in (0, 1)$. Es bezeichne $B : F \rightarrow P'$ den von b induzierten Operator.

- **(Kontinuierliches) Optimal-Steuer-Problem:** Es gelte:

*) a erfüllt $\alpha^t(a) > 0$ und $A : U \rightarrow P'$ ist injektiv.

Dann existiert eine eindeutige (kontinuierliche) Lösung $f \in F$ des folgenden (kontinuierlichen) Optimal-Steuer-Problems:

$$\text{Finde den Minimierer } f \in F \text{ von } J : \begin{cases} F & \rightarrow \\ g & \mapsto \end{cases} \begin{matrix} \mathbb{R} \\ (1 - \kappa) \cdot \|u_0 - A^{-1}Bg\|_U^2 + \kappa \cdot \|g\|_F^2 \end{matrix}.$$

Der Minimierer f wird durch die folgende Optimalitäts-Bedingung charakterisiert:

$$\frac{\kappa}{1 - \kappa} \cdot f = (A^{-1}B)^T(u_0 - A^{-1}Bf).$$

Außerdem ist f Teil der eindeutigen Lösung $(f, u, p) \in F \times U \times P$ des Optimalitäts-Systems:

$$\begin{aligned} \frac{\kappa}{1 - \kappa} \cdot \langle f, \cdot \rangle_F &= b(\cdot, p) && \text{auf } F, \\ a(\cdot, p) &= \langle u_0 - u, \cdot \rangle_U && \text{auf } U, \\ a(u, \cdot) &= b(f, \cdot) && \text{auf } P. \end{aligned}$$

Der Minimierer f wird als optimale Steuerung bezeichnet und u ist der zugehörige Zustand.

- **Rückführung auf ein Sattelpunkt-Problem:** Mit den stetigen Bilinearformen

$$\begin{aligned} \forall (f, u), (g, v) \in F \times U : & \quad c((f, u), (g, v)) := \frac{\kappa}{1 - \kappa} \cdot \langle f, g \rangle_F + \langle u, v \rangle_U, \\ \forall (f, u) \in F \times U, q \in P : & \quad d((f, u), q) := a(u, q) - b(f, q) \end{aligned}$$

gilt: Die Voraussetzungen an a und b garantieren die eindeutige Lösbarkeit des folgenden (kontinuierlichen) Sattelpunkt-Problems:

$$\text{Finde } (f, u, p) \in F \times U \times P \text{ mit : } \begin{aligned} c((f, u), \cdot) + d(\cdot, p) &= \langle u_0, \cdot \rangle_U && \text{auf } F \times U, \\ d((f, u), \cdot) &= 0 && \text{auf } P. \end{aligned}$$

Die (kontinuierliche) Lösung (f, u, p) hängt linear und stetig von den Daten u_0 ab mit $\|(f, u, p)\|_{F \times U \times P} \leq \frac{6(\|a\| + \|b\| + 1)^2}{\kappa(1 - \kappa)^2 \alpha^t(a)^2} \cdot \|u_0\|_U$.

Das kontinuierliche Sattelpunkt-Problem ist äquivalent zum Optimalitäts-System. Insbesondere ist f der eindeutige Minimierer von J .

^aDie Fälle $\kappa \in \{0, 1\}$ sind nicht weiter von Interesse. Bei $\kappa := 1$ ist $f = 0$ der eindeutige Minimierer von J . Bei $\kappa := 0$ hat J im Allgemeinen keinen Minimierer, wie das folgende Beispiel zeigt: $\Omega := (0, 1) \subseteq \mathbb{R}$, $F := L^2$, $U := P := H_0^1$, $a(u, p) := \langle u, p \rangle_{H^1}$, $b(f, p) := \langle f, p \rangle_{L^2}$. Dann erfüllt $A^{-1}B : L^2 \rightarrow H_0^1$ die Identität $\text{ran}(A^{-1}B) = H^2 \cap H_0^1$. Das von einem $u_0 \in H_0^1 \setminus H^2$ induzierte Funktional $[\forall f \in L^2 : J(f) := \|u_0 - A^{-1}Bf\|_{H^1}^2]$ kann keinen Minimierer haben, da es sonst ein $f \in L^2$ gäbe mit $0 < \|u_0 - A^{-1}Bf\|_{H^1} = \min_{g \in L^2} \|u_0 - A^{-1}Bg\|_{H^1} = \min_{u \in H^2 \cap H_0^1} \|u_0 - u\|_{H^1} = 0$.

Beweis:

Schritt 1: $[f \in F \text{ minimiert } J] \Leftrightarrow [\frac{\kappa}{1-\kappa} \cdot f = (A^{-1}B)^T(u_0 - A^{-1}Bf)]$

Ein $f \in F$ minimiert J , genau wenn für jedes $g \in F$ die nach oben geöffnete Parabel

$$\begin{aligned} [0, 1] \ni \varepsilon \mapsto J(f + \varepsilon \cdot (g - f)) &= (1 - \kappa) \cdot \|u_0 - A^{-1}Bf - \varepsilon \cdot A^{-1}B(g - f)\|_U^2 + \kappa \cdot \|f + \varepsilon \cdot (g - f)\|_F^2 \\ &= \underbrace{J(f)}_{\geq 0} + \varepsilon \cdot [2\kappa \cdot \langle f, g - f \rangle_F - 2(1 - \kappa) \cdot \langle u_0 - A^{-1}Bf, A^{-1}B(g - f) \rangle_U] \\ &\quad + \varepsilon^2 \cdot \underbrace{[(1 - \kappa) \cdot \|A^{-1}B(g - f)\|_U^2 + \kappa \cdot \|g - f\|_F^2]}_{\geq 0} \end{aligned}$$

bei $\varepsilon = 0$ minimal wird, also genau falls für jedes $g \in F$ der Koeffizient bei ε nicht-negativ ist:

$$\forall g \in F : \quad 2 \cdot \kappa \cdot \langle f, g - f \rangle_F - 2 \cdot (1 - \kappa) \cdot \langle u_0 - A^{-1}Bf, A^{-1}B(g - f) \rangle_U \geq 0.$$

Das ist wiederum äquivalent zu

$$\kappa \cdot f - (1 - \kappa) \cdot (A^{-1}B)^T(u_0 - A^{-1}Bf) = 0.$$

□

Schritt 2: Kont. Sattelp.-P. eindeutig lösbar $\Rightarrow$ Opt.-Sys. eindeutig lösbar $\Rightarrow \exists!$ Minimierer von J

Wir werden später in diesem Beweis zeigen, dass das kontinuierliche Sattelpunkt-Problem eine eindeutige Lösung $(f, u, p) \in F \times U \times P$ besitzt. Wir verwenden diese Tatsache aber bereits jetzt, um die eindeutige Existenz eines Minimierers von J zu zeigen:

Einfaches Nachrechnen zeigt die Äquivalenz von kontinuierlichem Sattelpunkt-Problem und Optimalitäts-System. Also ist auch dieses eindeutig lösbar mit der gleichen Lösung (f, u, p) .

Das Optimalitäts-System lässt sich mit Hilfe der Identitäten $A^t = \tau_U \circ A^T \circ \tau_P$ und $B^t = \tau_F \circ B^T \circ \tau_P$ (vgl. Lemma 2.1.2.1) auf folgende Form bringen:

$$\begin{aligned} \frac{\kappa}{1 - \kappa} \cdot f &= (B^T \circ \tau_P)(p), \\ (A^T \circ \tau_P)(p) &= u_0 - u, \\ Au &= Bf. \end{aligned}$$

Insbesondere erfüllt f auch die Optimalitäts-Bedingung:

$$\frac{\kappa}{1 - \kappa} \cdot f = (B^T \circ \tau_P)(p) = [(B^T \circ \tau_P) \circ (A^T \circ \tau_P)^{-1}](u_0 - u) = (A^{-1}B)^T(u_0 - A^{-1}Bf).$$

Ist andererseits $\tilde{f} \in F$ ein weiteres Element, das der Optimalitäts-Bedingung genügt, dann bildet es zusammen mit $\tilde{u} := A^{-1}B\tilde{f} \in U$ und $\tilde{p} := (A^T \circ \tau_P)^{-1}(u_0 - \tilde{u}) \in P$ eine Lösung des Optimalitäts-Systems. Da (f, u, p) aber schon die *eindeutige* Lösung des Optimalitäts-System ist, muss $\tilde{f} = f$ gelten.

Zusammen mit Schritt 1 ergibt sich gesamt: J besitzt einen eindeutigen Minimierer und dieser ist gegeben durch f .

□

Schritt 3: Rückführung auf Satz 2.3.1.1

Wir wenden nun Satz 2.3.1.1 auf folgende Situation an:

$$\begin{aligned}
V &:= F \times U, \\
P &= P, \\
\forall (f, u), (g, v) \in F \times U : \quad c((f, u), (g, v)) &:= \frac{\kappa}{1-\kappa} \cdot \langle f, g \rangle_F + \langle u, v \rangle_U, \\
\forall (f, u) \in F \times U, q \in P : \quad d((f, u), q) &:= a(u, q) - b(f, q), \\
\forall (g, v) \in F \times U : \quad r(g, v) &:= \langle u_0, v \rangle_U, \\
\forall q \in P : \quad s(q) &:= 0, \\
(v, p) &= (f, u, p) \quad (\text{kontin. Lösung}).
\end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Satz 2.3.1.1:

Schritt 4: V, P sind Hilbert-Räume, c, d sind stetige Bilinearformen, r, s sind stetige Linearformen

Es sind c und d stetige Bilinearformen mit $\|c\| \leq \frac{1}{1-\kappa}$ und $\|d\| \leq \|a\| + \|b\|$. Weiters sind $r \in (F \times U)'$ mit $\|r\|_{(F \times U)'} \leq \|u_0\|_U$ und $s \in P'$ mit $\|s\|_{P'} = 0$.

□

Schritt 5: $\alpha(c|_{\ker(D)^2}) > 0$, $C_{\ker}^t : \ker(D) \rightarrow \ker(D)'$ ist injektiv, $\alpha^t(d) > 0$

Es ist c sogar koerziv auf ganz $F \times U$ mit

$$\alpha_{\text{kzv}}(c) = \inf_{(f,u) \in (F \times U)^\times} \frac{c((f,u), (f,u))}{\|(f,u)\|_{F \times U}^2} = \inf_{(f,u) \in (F \times U)^\times} \frac{\frac{\kappa}{1-\kappa} \cdot \|f\|_F^2 + \|u\|_U^2}{\|f\|_F^2 + \|u\|_U^2} \geq \min\left\{\frac{\kappa}{1-\kappa}, 1\right\} \geq \kappa > 0.$$

Insbesondere gilt mit dem von d induzierten Operator $D : F \times U \rightarrow P'$ bereits

$$\alpha(c|_{\ker(D)^2}) \geq \alpha_{\text{kzv}}(c) \geq \kappa > 0.$$

Eine analoge Rechnung zeigt auch $\alpha^t(c|_{\ker(D)^2}) \geq \kappa > 0$ und damit die Injektivität des von $c|_{\ker(D)^2}$ induzierten Operators $C_{\ker}^t : \ker(D) \rightarrow \ker(D)'$.

Weiters gilt

$$\begin{aligned}
\alpha^t(d) &= \inf_{q \in P^\times} \sup_{(f,u) \in (F \times U)^\times} \frac{d((f,u), q)}{\|(f,u)\|_{F \times U} \cdot \|q\|_P} \\
&\geq \inf_{q \in P^\times} \sup_{u \in U^\times} \frac{d((0,u), q)}{\|(0,u)\|_{F \times U} \cdot \|q\|_P} \\
&= \inf_{q \in P^\times} \sup_{u \in U^\times} \frac{a(u, q)}{\|u\|_U \cdot \|q\|_P} \\
&= \alpha^t(a) \\
&> 0.
\end{aligned}$$

□

Also existiert eine eindeutige Lösung $(v, p) = (f, u, p) \in F \times U \times P$ des kontinuierlichen Sattelpunkt-Problems. Die Abschätzung für $\|(f, u, p)\|_{F \times U \times P}$ ergibt sich unmittelbar aus jener von $\|(v, p)\|_{V \times P}$ in Satz 2.3.1.1.

■

2.4.2 Diskretisierung

Lemma 2.4.2.1 (Optimal-Steuer-Problem Diskretisierung).

Seien nun zusätzlich $F_h \leq F, U_h \leq U, P_h \leq P$ Ansatz-Räume mit folgenden Eigenschaften:

*) $\dim F_h < \infty, \dim U_h < \infty$ und $\dim P_h < \infty$.

*) a erfüllt $\alpha_h^t(a) > 0$.

Dann existiert auch eine eindeutige (diskrete) Lösung $(f_h, u_h, p_h) \in F_h \times U_h \times P_h$ des zugehörigen (diskreten) Sattelpunkt-Problems:

$$\begin{aligned} \text{Finde } (f_h, u_h, p_h) \in F_h \times U_h \times P_h \text{ mit : } \quad & c((f_h, u_h), \cdot) + d(\cdot, p_h) = \langle u_0, \cdot \rangle_U \quad \text{auf } F_h \times U_h, \\ & d((f_h, u_h), \cdot) = 0 \quad \text{auf } P_h. \end{aligned}$$

Beweis:

Schritt 1: Rückführung auf Lemma 2.3.2.1

Wir wenden nun Lemma 2.3.2.1 auf folgende Situation an:

$$\begin{aligned} V_h &:= F_h \times U_h, \\ P_h &= P_h, \\ (v_h, p_h) &= (f_h, u_h, p_h) \quad (\text{diskr. Lösung}). \end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Lemma 2.3.2.1:

Schritt 2: $\dim V_h < \infty, \dim P_h < \infty, \quad \alpha_h(c|_{\ker(D_h)^2}) > 0, \quad \alpha_h^t(d) > 0$

Es gilt klarerweise $\dim V_h = \dim F_h + \dim U_h < \infty$ und $\dim P_h < \infty$.

Analog zum Beweis von Satz 2.4.1.1 erhält man

$$\alpha_h(c|_{\ker(D_h)^2}) \geq \kappa > 0, \quad \alpha_h^t(d) \geq \alpha_h^t(a) > 0.$$

□

Also existiert eine eindeutige Lösung $(v_h, p_h) = (f_h, u_h, p_h) \in F_h \times U_h \times P_h$ des diskreten Sattelpunkt-Problems.

■

2.4.3 Konvergenz

Satz 2.4.3.1 (Optimal-Steuer-Problem Konvergenz).

Seien $F_{\text{stg}}, U_{\text{stg}}$ und P_{stg} weitere Hilbert-Räume mit folgenden Eigenschaften:

*) $F_{\text{stg}} \leq F, U_{\text{stg}} \leq U$ und $P_{\text{stg}} \leq P$ jeweils dicht.

*) Für die Orthogonal-Projektionen $P_{F_h} : F \rightarrow F_h, P_{U_h} : U \rightarrow U_h$ und $P_{P_h} : P \rightarrow P_h$ gilt

$$\|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0, \quad \|(\text{id} - P_{U_h})|_{U_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0, \quad \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0.$$

Für die Ansatz-Räume gelte zusätzlich

*) a erfüllt $\alpha_0^t(a) > 0$.

Dann gilt

$$(f_h, u_h, p_h) \xrightarrow[\|\cdot\|_{F \times U \times P}]{h \rightarrow 0} (f, u, p)$$

und unter der Voraussetzung

$$*) (f, u, p) \in F_{\text{stg}} \times U_{\text{stg}} \times P_{\text{stg}}$$

sogar

$$\|f - f_h\|_F + \|u - u_h\|_U + \|p - p_h\|_P \leq \frac{48 \cdot (\|a\| + \|b\| + 1)^3}{\kappa \cdot (1 - \kappa)^3 \cdot \alpha_0^t(a)^2} \cdot \|(f, u, p)\|_{F_{\text{stg}} \times U_{\text{stg}} \times P_{\text{stg}}} \cdot (\|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| + \|(\text{id} - P_{U_h})|_{U_{\text{stg}}}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\|).$$

Beweis:

Schritt 1: Rückführung auf Satz 2.3.3.1

Wir wenden nun Satz 2.3.3.1 auf folgende Situation an:

$$\begin{aligned} V_{\text{stg}} &:= F_{\text{stg}} \times U_{\text{stg}}, \\ P_{\text{stg}} &= P_{\text{stg}}. \end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Satz 2.3.3.1:

Schritt 2: $V_{\text{stg}} \leq V, P_{\text{stg}} \leq P$ dicht, $\|(\text{id} - P_{V_h})|_{V_{\text{stg}}}\| \rightarrow 0, \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\| \rightarrow 0, \alpha_0(c) > 0, \alpha_0^t(d) > 0$

Es sind $V_{\text{stg}} = F_{\text{stg}} \times U_{\text{stg}} \leq F \times U = V$ und $P_{\text{stg}} \leq P$ jeweils dicht.

Die Abbildung

$$P_{F_h \times U_h} : \begin{cases} F \times U & \longrightarrow & F_h \times U_h \\ (f, u) & \longmapsto & (P_{F_h}f, P_{U_h}u) \end{cases}$$

ist wegen $(F_h \times U_h)^\perp = F_h^\perp \times U_h^\perp$ genau die Orthogonal-Projektion auf $F_h \times U_h$. Analog zum Beweis von Satz 2.3.3.1 sieht man

$$\|(\text{id} - P_{F_h \times U_h})|_{F_{\text{stg}} \times U_{\text{stg}}}\| \leq \|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| + \|(\text{id} - P_{U_h})|_{U_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0.$$

Weiters haben wir

$$\alpha_0(c) = \inf_{h \in H} \alpha_h(c|_{\ker(D_h)^2}) \stackrel{\text{L.2.4.2.1}}{\geq} \kappa > 0, \quad \alpha_0^t(d) = \inf_{h \in H} \alpha_h^t(d) \stackrel{\text{L.2.4.2.1}}{\geq} \inf_{h \in H} \alpha_h^t(a) = \alpha_0^t(a) > 0.$$

□

Also konvergiert $(f_h, u_h, p_h) \xrightarrow[\|\cdot\|_{F \times U \times P}]{h \rightarrow 0} (f, u, p)$. Die Abschätzung für $\|f - f_h\|_F + \|u - u_h\|_U + \|p - p_h\|_P$ ergibt sich unmittelbar aus jener für $\|v - v_h\|_V + \|p - p_h\|_P$ in Satz 2.3.3.1.

■

2.4.4 Aubin-Nitsche-Trick

Satz 2.4.4.1 (Optimal-Steuer-Problem Aubin-Nitsche-Trick).

Seien^a $F_{\text{wk}}, U_{\text{wk}}, P_{\text{wk}}$ und $F_{\text{stg}}^t, U_{\text{stg}}^t, P_{\text{stg}}^t$ weitere Hilbert-Räume mit folgenden Eigenschaften:

*) $\iota_{F_{\text{wk}}} : F \longrightarrow F_{\text{wk}}, \iota_{U_{\text{wk}}} : U \longrightarrow U_{\text{wk}}$ und $\iota_{P_{\text{wk}}} : P \longrightarrow P_{\text{wk}}$ jeweils linear und stetig.

*) $F_{\text{stg}}^t \leq F, U_{\text{stg}}^t \leq U$ und $P_{\text{stg}}^t \leq P$.

*) Für die Orthogonal-Projektionen $P_{F_h} : F \rightarrow F_h$, $P_{U_h} : U \rightarrow U_h$ und $P_{P_h} : P \rightarrow P_h$ gilt

$$\|(\text{id} - P_{F_h})|_{F_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0, \quad \|(\text{id} - P_{U_h})|_{U_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0, \quad \|(\text{id} - P_{P_h})|_{P_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0.$$

Die Voraussetzungen an a und b garantieren auch die eindeutige Lösbarkeit des folgenden (kontinuierlichen) transponierten Sattelpunkt-Problems:

$$\begin{aligned} \text{Für geg. } (f, u, p) \in F \times U \times P, \text{ finde } (g, v, q) \in F \times U \times P \text{ mit :} \\ c(\cdot, (g, v)) + d(\cdot, q) &= \langle \iota_{F_{\text{wk}}}(\cdot), \iota_{F_{\text{wk}}} f \rangle_{F_{\text{wk}}} + \langle \iota_{U_{\text{wk}}}(\cdot), \iota_{U_{\text{wk}}} u \rangle_{U_{\text{wk}}} \quad \text{auf } F \times U, \\ d((g, v), \cdot) &= \langle \iota_{P_{\text{wk}}}(\cdot), \iota_{P_{\text{wk}}} p \rangle_{P_{\text{wk}}} \quad \text{auf } P. \end{aligned}$$

Der zugehörige Lösungs-Operator $S : F \times U \times P \rightarrow F \times U \times P$ ist linear, stetig und erfüllt

$$\begin{aligned} \forall (f, u, p) \in F \times U \times P : \|S(f, u, p)\|_{F \times U \times P} &\leq \frac{3 \cdot (\|a\| + \|b\| + 1)^2 \cdot (\|\iota_{F_{\text{wk}}}\| + \|\iota_{U_{\text{wk}}}\| + \|\iota_{P_{\text{wk}}}\|)}{\kappa \cdot (1 - \kappa)^2 \cdot \alpha^t(a)^2} \\ &\cdot \|(\iota_{F_{\text{wk}}} f, \iota_{U_{\text{wk}}} u, \iota_{P_{\text{wk}}} p)\|_{F_{\text{wk}} \times U_{\text{wk}} \times P_{\text{wk}}}. \end{aligned}$$

Erfüllt S darüber hinaus die Voraussetzungen

*) $\text{ran}(S) \subseteq F_{\text{stg}}^t \times U_{\text{stg}}^t \times P_{\text{stg}}^t$ und

$$\forall (f, u, p) \in F \times U \times P : \|S(f, u, p)\|_{F_{\text{stg}}^t \times U_{\text{stg}}^t \times P_{\text{stg}}^t} \leq C_S \cdot \|(\iota_{F_{\text{wk}}} f, \iota_{U_{\text{wk}}} u, \iota_{P_{\text{wk}}} p)\|_{F_{\text{wk}} \times U_{\text{wk}} \times P_{\text{wk}}},$$

dann gilt im Falle

*) $(f, u, p) \in F_{\text{stg}} \times U_{\text{stg}} \times P_{\text{stg}}$

auch die Fehler-Abschätzung

$$\begin{aligned} &\|\iota_{F_{\text{wk}}}(f - f_h)\|_{F_{\text{wk}}} \\ &+ \|\iota_{U_{\text{wk}}}(u - u_h)\|_{U_{\text{wk}}} \\ &+ \|\iota_{P_{\text{wk}}}(p - p_h)\|_{P_{\text{wk}}} \leq \frac{96 \cdot (\|a\| + \|b\| + 1)^4 \cdot C_S}{\kappa \cdot (1 - \kappa)^4 \cdot \alpha_0^t(a)^2} \cdot \|(f, u, p)\|_{F_{\text{stg}} \times U_{\text{stg}} \times P_{\text{stg}}} \\ &\cdot (\|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| + \|(\text{id} - P_{U_h})|_{U_{\text{stg}}}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\|) \\ &\cdot (\|(\text{id} - P_{F_h})|_{F_{\text{stg}}^t}\| + \|(\text{id} - P_{U_h})|_{U_{\text{stg}}^t}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}^t}\|). \end{aligned}$$

^aWir verstehen die Räume F_{stg}^t , U_{stg}^t und P_{stg}^t mit dem Index „t“, um die Zugehörigkeit zum *transponierten* Sattelpunkt-Problem zu verdeutlichen. Wir fordern aber *nicht* zwingend einen Zusammenhang mit den Räumen F_{stg} , U_{stg} und P_{stg} von oben.

Beweis:

Schritt 1: Rückführung auf Satz 2.3.4.1

Wir wenden nun Satz 2.3.4.1 auf folgende Situation an:

$$\begin{aligned} V_{\text{wk}} &:= F_{\text{wk}} \times U_{\text{wk}}, \\ P_{\text{wk}} &= P_{\text{wk}}, \\ V_{\text{stg}}^t &:= F_{\text{stg}}^t \times U_{\text{stg}}^t, \\ P_{\text{stg}}^t &= P_{\text{stg}}^t, \\ S &= S. \end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Satz 2.3.4.1:

Schritt 2: $\|\iota_{V_{\text{wk}}}\|, \|\iota_{P_{\text{wk}}}\| < \infty, \|(\text{id} - P_{V_h})|_{V_{\text{stg}}^t}\|, \|(\text{id} - P_{P_h})|_{P_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0$, Voraussetzungen an S
 Es sind $V_{\text{wk}} = F_{\text{wk}} \times U_{\text{wk}}$ und P_{wk} sowie $V_{\text{stg}}^t = F_{\text{stg}}^t \times U_{\text{stg}}^t$ und P_{stg}^t Hilbert-Räume und

$$\iota_{F_{\text{wk}} \times U_{\text{wk}}} : \begin{cases} F \times U & \longrightarrow F_{\text{wk}} \times U_{\text{wk}} \\ (f, u) & \longmapsto (\iota_{F_{\text{wk}}} f, \iota_{U_{\text{wk}}} u) \end{cases}$$

linear und stetig mit $\|\iota_{F_{\text{wk}} \times U_{\text{wk}}}\| \leq \|\iota_{F_{\text{wk}}}\| + \|\iota_{U_{\text{wk}}}\|$.

Analog zum Beweis von Satz 2.3.3.1 sieht man

$$\|(\text{id} - P_{F_h \times U_h})|_{F_{\text{stg}}^t \times U_{\text{stg}}^t}\| \leq \|(\text{id} - P_{F_h})|_{F_{\text{stg}}^t}\| + \|(\text{id} - P_{U_h})|_{U_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0.$$

Der Lösungs-Operator $S : F \times U \times P \longrightarrow F \times U \times P$ stimmt mit dem Lösungs-Operator $S : V \times P \longrightarrow V \times P$ aus Satz 2.3.4.1 überein und erfüllt $\text{ran}(S) \subseteq F_{\text{stg}}^t \times U_{\text{stg}}^t \times P_{\text{stg}}^t = V_{\text{stg}}^t \times P_{\text{stg}}^t$ sowie

$$\forall (f, u, p) \in F \times U \times P : \|S(f, u, p)\|_{F_{\text{stg}}^t \times U_{\text{stg}}^t \times P_{\text{stg}}^t} \leq C_S \cdot \|(\iota_{F_{\text{wk}} \times U_{\text{wk}}}(f, u), \iota_{P_{\text{wk}}} p)\|_{F_{\text{wk}} \times U_{\text{wk}} \times P_{\text{wk}}}.$$

□

Die Abschätzung für $\|S(\cdot, \cdot, \cdot)\|_{F \times U \times P}$ und $\|\iota_{F_{\text{wk}}}(f - f_h)\|_{F_{\text{wk}}} + \|\iota_{U_{\text{wk}}}(u - u_h)\|_{U_{\text{wk}}} + \|\iota_{P_{\text{wk}}}(p - p_h)\|_{P_{\text{wk}}}$ ergeben sich unmittelbar aus jenen für $\|S(\cdot, \cdot, \cdot)\|_{V \times P}$ und $\|\iota_{V_{\text{wk}}}(v - v_h)\|_{V_{\text{wk}}} + \|\iota_{P_{\text{wk}}}(p - p_h)\|_{P_{\text{wk}}}$ in Satz 2.3.4.1.

■

2.4.5 LGS

Lemma 2.4.5.1 (Optimal-Steuer-Problem LGS).

Seien nun $\{f_1, \dots, f_{n_F}\} \subseteq F_h$, $\{u_1, \dots, u_{n_U}\} \subseteq U_h$ und $\{p_1, \dots, p_{n_P}\} \subseteq P_h$ Basen. Wir bezeichnen mit

$$\gamma_h : \begin{cases} F_h & \longrightarrow \mathbb{R}^{n_F} \\ \sum_{i=1}^{n_F} \gamma_i f_i & \longmapsto (\gamma_i)_{i=1}^{n_F} \end{cases}, \mu_h : \begin{cases} U_h & \longrightarrow \mathbb{R}^{n_U} \\ \sum_{j=1}^{n_U} \mu_j u_j & \longmapsto (\mu_j)_{j=1}^{n_U} \end{cases}, \pi_h : \begin{cases} P_h & \longrightarrow \mathbb{R}^{n_P} \\ \sum_{k=1}^{n_P} \pi_k p_k & \longmapsto (\pi_k)_{k=1}^{n_P} \end{cases}$$

die zugehörigen Koordinaten-Abbildungen. Die Koeffizienten $(\gamma_l)_{l=1}^{n_F} \in \mathbb{R}^{n_F}$, $(\mu_m)_{m=1}^{n_U} \in \mathbb{R}^{n_U}$ und $(\pi_n)_{n=1}^{n_P} \in \mathbb{R}^{n_P}$ aus den Darstellungen

$$f_h = \sum_{l=1}^{n_F} \gamma_l f_l, \quad u_h = \sum_{m=1}^{n_U} \mu_m u_m, \quad p_h = \sum_{n=1}^{n_P} \pi_n p_n$$

lassen sich durch Lösen eines regulären LGS bestimmen: Mit

$$\begin{aligned} \mathbf{F} &:= \left(\frac{\kappa}{1 - \kappa} \cdot \langle f_l, f_i \rangle_F \right)_{i=1, l=1}^{n_F, n_F} \in \mathbb{R}^{n_F \times n_F}, \\ \mathbf{U} &:= (\langle u_m, u_j \rangle_U)_{j=1, m=1}^{n_U, n_U} \in \mathbb{R}^{n_U \times n_U}, \\ \mathbf{A} &:= (a(u_m, p_k))_{k=1, m=1}^{n_P, n_U} \in \mathbb{R}^{n_P \times n_U}, \\ \mathbf{B} &:= (-b(f_l, p_k))_{k=1, l=1}^{n_P, n_F} \in \mathbb{R}^{n_P \times n_F}, \\ \boldsymbol{\gamma} &:= (\gamma_l)_{l=1}^{n_F} \in \mathbb{R}^{n_F}, \\ \boldsymbol{\mu} &:= (\mu_m)_{m=1}^{n_U} \in \mathbb{R}^{n_U}, \\ \boldsymbol{\pi} &:= (\pi_n)_{n=1}^{n_P} \in \mathbb{R}^{n_P}, \\ \mathbf{u}_0 &:= (\langle u_0, u_j \rangle_U)_{j=1}^{n_U} \in \mathbb{R}^{n_U} \end{aligned}$$

gilt:

$$\begin{pmatrix} F & 0 & B^T \\ 0 & U & A^T \\ B & A & 0 \end{pmatrix} \cdot \begin{pmatrix} \gamma \\ \mu \\ \pi \end{pmatrix} = \begin{pmatrix} 0 \\ u_0 \\ 0 \end{pmatrix}.$$

Die System-Matrix ist symmetrisch und indefinit mit genau $n_F + n_U$ positiven und genau n_P negativen Eigenwerten. Die Konditionszahl erfüllt die Abschätzung

$$\text{cond} \begin{pmatrix} F & 0 & B^T \\ 0 & U & A^T \\ B & A & 0 \end{pmatrix} \leq \frac{96 \cdot (\|a\| + \|b\| + 1)^3}{\kappa \cdot (1 - \kappa)^3 \cdot \alpha_0^t(a)^2} \cdot (\|\gamma_h\|^2 + \|\mu_h\|^2 + \|\pi_h\|^2) \cdot (\|\gamma_h^{-1}\|^2 + \|\mu_h^{-1}\|^2 + \|\pi_h^{-1}\|^2).$$

Beweis:

Schritt 1: Rückführung auf Lemma 2.3.5.1

Wir wenden nun Lemma 2.3.5.1 auf folgende Situation an:

$$\begin{aligned} \{v_1, \dots, v_{n_V}\} &:= \{(f_i, 0) \mid i \in \{1, \dots, n_F\}\} \cup \{(0, u_j) \mid j \in \{1, \dots, n_U\}\}, \\ \{p_1, \dots, p_{n_P}\} &= \{p_1, \dots, p_{n_P}\}. \end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Lemma 2.3.5.1:

Schritt 2: $\{v_1, \dots, v_{n_V}\} \subseteq V_h$ und $\{p_1, \dots, p_{n_P}\} \subseteq P_h$ sind Basen

Klarerweise sind $\{(f_i, 0) \mid i \in \{1, \dots, n_F\}\} \cup \{(0, u_j) \mid j \in \{1, \dots, n_U\}\} \subseteq F_h \times U_h$ und $\{p_1, \dots, p_{n_P}\} \subseteq P_h$ Basen.

Das LGS aus Satz 2.3.5.1 nimmt genau die angegebene Form an.

Nun zur Abschätzung der Konditionszahl: Die Koordinaten-Abbildung zur obigen Basis von $F_h \times U_h$ lässt sich mit Hilfe der Koordinaten-Abbildungen γ_h und μ_h explizit angeben:

$$\nu_h : \begin{cases} F_h \times U_h & \longrightarrow \mathbb{R}^{n_F + n_U} \\ (f, u) & \longmapsto (\gamma_h f, \mu_h u) \end{cases}.$$

Analog zum Beweis von Lemma 2.3.5.1 sieht man

$$\|\nu_h\| \leq \|\gamma_h\| + \|\mu_h\|, \quad \|\nu_h^{-1}\| \leq \|\gamma_h^{-1}\| + \|\mu_h^{-1}\|.$$

Die Abschätzung für die Konditionszahl der System-Matrix ergibt sich unmittelbar aus jener in Satz 2.3.5.1.

□

Schritt 3: Symmetrie und Indefinitheit der System-Matrix

Die Bilinearform c ist symmetrisch und koerziv auf ganz $F \times U$, wobei $\alpha_{\text{kzv}}(c) \stackrel{\text{S.2.4.1.1}}{\geq} \kappa > 0$. Also ist die System-Matrix laut Lemma 2.3.5.1 symmetrisch indefinit mit genau $n_F + n_U$ positiven und genau n_P negativen Eigenwerten.

□

■

2.4.6 Strang-Lemma

Lemma 2.4.6.1 (Optimal-Steuer-Problem Strang-Lemma).

Es bezeichne $(u_{0,h})_{h \in H} \subseteq U$ eine Folge mit

$$*) \quad u_{0,h} \xrightarrow[\|\cdot\|_U]{h \rightarrow 0} u_0.$$

Dann erfüllen auch die eindeutigen Lösungen $(\tilde{f}_h, \tilde{u}_h, \tilde{p}_h) \in F_h \times U_h \times P_h$ der diskreten Sattelpunkt-Probleme

$$\begin{bmatrix} c((\tilde{f}_h, \tilde{u}_h), \cdot) + d(\cdot, \tilde{p}_h) &= \langle u_{0,h}, \cdot \rangle_U & \text{auf } F_h \times U_h, \\ d((\tilde{f}_h, \tilde{u}_h), \cdot) &= 0 & \text{auf } P_h \end{bmatrix}$$

die Konvergenz $(\tilde{f}_h, \tilde{u}_h, \tilde{p}_h) \xrightarrow[\|\cdot\|_{F \times U \times P}]{h \rightarrow 0} (f, u, p)$ und im Falle

$$*) \quad (f, u, p) \in F_{\text{stg}} \times U_{\text{stg}} \times P_{\text{stg}}$$

die Fehler-Abschätzungen

$$\begin{aligned} \|f - \tilde{f}_h\|_F + \|u - \tilde{u}_h\|_U + \|p - \tilde{p}_h\|_P &\leq \frac{48 \cdot (\|a\| + \|b\| + 2)^3}{\kappa \cdot (1 - \kappa)^3 \cdot \alpha_0^t(a)^2} \cdot (\|(f, u, p)\|_{F_{\text{stg}} \times U_{\text{stg}} \times P_{\text{stg}}} + 1) \\ &\quad \cdot (\|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| + \|(\text{id} - P_{U_h})|_{U_{\text{stg}}}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\| \\ &\quad + \|u_0 - u_{0,h}\|_U) \end{aligned}$$

und

$$\begin{aligned} &\|\iota_{F_{\text{wk}}}(f - \tilde{f}_h)\|_{F_{\text{wk}}} \\ &\quad + \|\iota_{U_{\text{wk}}}(u - \tilde{u}_h)\|_{U_{\text{wk}}} \\ &\quad + \|\iota_{P_{\text{wk}}}(p - \tilde{p}_h)\|_{P_{\text{wk}}} \leq \frac{96 \cdot (\|a\| + \|b\| + \|\iota_{F_{\text{wk}}}\| + \|\iota_{U_{\text{wk}}}\| + \|\iota_{P_{\text{wk}}}\| + 1)^4 \cdot (C_S + 1)}{\kappa \cdot (1 - \kappa)^4 \cdot \alpha_0^t(a)^2} \\ &\quad \cdot (\|(f, u, p)\|_{F_{\text{stg}} \times U_{\text{stg}} \times P_{\text{stg}}} + 1) \\ &\quad \cdot [\|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| + \|(\text{id} - P_{U_h})|_{U_{\text{stg}}}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\|] \\ &\quad \cdot (\|(\text{id} - P_{F_h})|_{F_{\text{stg}}^t}\| + \|(\text{id} - P_{U_h})|_{U_{\text{stg}}^t}\| + \|(\text{id} - P_{P_h})|_{P_{\text{stg}}^t}\|) \\ &\quad + \|u_0 - u_{0,h}\|_U]. \end{aligned}$$

Für die Berechnung der Koeffizienten von $\tilde{f}_h$, $\tilde{u}_h$ und $\tilde{p}_h$ muss nun der Last-Vektor bestehend aus $\mathbf{0} \in \mathbb{R}^{n_F}$, $\mathbf{u}_{0,h} := (\langle u_{0,h}, u_j \rangle_U)_{j=1}^{n_U} \in \mathbb{R}^{n_U}$ und $\mathbf{0} \in \mathbb{R}^{n_P}$ verwendet werden.

Beweis:

Schritt 1: Rückführung auf Lemma 2.3.6.1

Wir wenden nun Lemma 2.3.6.1 auf folgende Situation an:

$$\begin{aligned} \forall (g, v) \in F_h \times U_h : \quad r_h(g, v) &:= \langle u_{0,h}, v \rangle_U, \\ \forall q \in P_h : \quad s_h(q) &:= 0, \\ (\tilde{v}_h, \tilde{p}_h) &= (\tilde{f}_h, \tilde{u}_h, \tilde{p}_h) \quad (\text{diskr. Lösung zu } (r_h, s_h)). \end{aligned}$$

□

Wir zeigen jetzt die Voraussetzungen von Lemma 2.3.6.1:

$$\text{Schritt 2:} \quad \|r - r_h\|_{(F_h \times U_h)'} \xrightarrow{h \rightarrow 0} 0, \quad \|s - s_h\|_{P_h'} \xrightarrow{h \rightarrow 0} 0$$

Es sind r_h und s_h stetige Linearformen mit $\|r_h\|_{(F_h \times U_h)'} = \|u_{0,h}\|_U$ und $\|s_h\|_{P_h'} = 0$. Also existiert eine eindeutige Lösung $(\tilde{v}_h, \tilde{p}_h) = (\tilde{f}_h, \tilde{u}_h, \tilde{p}_h) \in F_h \times U_h \times P_h$ des diskreten Sattelpunkt-Problems mit rechter

Seite $(\langle u_{0,h}, \cdot \rangle_U, 0)$. Es gilt

$$\|r - r_h\|_{(F_h \times U_h)'} = \sup_{(g,v) \in (F_h \times U_h) \times} \frac{|(r - r_h)(g, v)|}{\|(g, v)\|_{F \times U}} = \sup_{v \in U_h^\times} \frac{|\langle u_0 - u_{0,h}, v \rangle_U|}{\|v\|_U} \leq \|u_0 - u_{0,h}\|_U \xrightarrow{h \rightarrow 0} 0$$

sowie $\|s - s_h\|_{P_h'} = 0$ und damit $(\tilde{f}_h, \tilde{u}_h, \tilde{p}_h) \xrightarrow[\|\cdot\|_{F \times U \times P}]{h \rightarrow 0} (f, u, p)$.

□

Die Abschätzungen für $\|f - \tilde{f}_h\|_F + \|u - \tilde{u}_h\|_U + \|p - \tilde{p}_h\|_P$ und $\|\iota_{F_{\text{wk}}}(f - \tilde{f}_h)\|_{F_{\text{wk}}} + \|\iota_{U_{\text{wk}}}(u - \tilde{u}_h)\|_{U_{\text{wk}}} + \|\iota_{P_{\text{wk}}}(p - \tilde{p}_h)\|_{P_{\text{wk}}}$ ergeben sich unmittelbar aus jenen für $\|v - \tilde{v}_h\|_V + \|p - \tilde{p}_h\|_P$ und $\|\iota_{V_{\text{wk}}}(v - \tilde{v}_h)\|_{V_{\text{wk}}} + \|\iota_{P_{\text{wk}}}(p - \tilde{p}_h)\|_{P_{\text{wk}}}$ in Lemma 2.3.6.1.

■

2.4.7 Bemerkungen

Bemerkung 2.4.7.1 (Optimal-Steuer-Problem Bemerkungen).

Man beachte:

- Wegen $\alpha_h^t(a) > 0$ ist der von $a|_{U_h \times P_h}$ induzierte Operator $A_h^t : P_h \rightarrow U_h'$ injektiv. Also muss bereits $\dim P_h \leq \dim U_h$ gelten.
- Für die Verifikation der Voraussetzungen an S bei konkreten Beispielen ist die folgende äquivalente Formulierung des transponierten Sattelpunkt-Problems geeigneter:

Für geg. $(f, u, p) \in F \times U \times P$, finde $(g, v, q) \in F \times U \times P$ mit :

$$\begin{aligned} \frac{\kappa}{1-\kappa} \cdot \langle \cdot, g \rangle_F - b(\cdot, q) &= \langle \iota_{F_{\text{wk}}}(\cdot), \iota_{F_{\text{wk}}} f \rangle_{F_{\text{wk}}} && \text{auf } F, \\ \langle \cdot, v \rangle_U + a(\cdot, q) &= \langle \iota_{U_{\text{wk}}}(\cdot), \iota_{U_{\text{wk}}} u \rangle_{U_{\text{wk}}} && \text{auf } U, \\ a(v, \cdot) - b(g, \cdot) &= \langle \iota_{P_{\text{wk}}}(\cdot), \iota_{P_{\text{wk}}} p \rangle_{P_{\text{wk}}} && \text{auf } P. \end{aligned}$$

- Ist die Einbettung $\text{id}_{F_{\text{stg}}^t \rightarrow F}$ stetig und betrachtet man die Wahl $F_{\text{wk}} := (F_{\text{stg}}^t)'$ und $\iota_{F_{\text{wk}}} := (\cdot)|_{F_{\text{stg}}^t} \circ \tau_F : F \rightarrow F_{\text{wk}}$, dann gibt es für jedes $f \in F$ ein $f_{\text{stg}}^t \in F_{\text{stg}}^t$ mit

$$[\langle \iota_{F_{\text{wk}}}(\cdot), \iota_{F_{\text{wk}}} f \rangle_{F_{\text{wk}}} = \langle \cdot, f_{\text{stg}}^t \rangle_F \quad \text{auf } F]$$

und

$$\|f_{\text{stg}}^t\|_{F_{\text{stg}}^t} = \|\iota_{F_{\text{wk}}} f\|_{F_{\text{wk}}}.$$

Eine analoge Aussage gilt für $U_{\text{wk}} := (U_{\text{stg}}^t)'$ bzw. $P_{\text{wk}} := (P_{\text{stg}}^t)'$.

- Ist die Einbettung $\text{id}_{F_{\text{stg}}^t \rightarrow F}$ stetig, dann ist sie auf Grund der Voraussetzung $\|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0$ schon kompakt (vgl. Satz 2.1.1.1). Insbesondere kann im Falle $\dim F = \infty$ nicht $F_{\text{stg}} = F$ gelten. Gleiches gilt für $U_{\text{stg}} \leq U$, $P_{\text{stg}} \leq P$, $F_{\text{stg}}^t \leq F$, $U_{\text{stg}}^t \leq U$ und $P_{\text{stg}}^t \leq P$.
- Seien $\text{id}_{F_{\text{stg}}^t \rightarrow F}$, $\text{id}_{U_{\text{stg}}^t \rightarrow U}$ und $\text{id}_{P_{\text{stg}}^t \rightarrow P}$ stetig (und damit bereits kompakt). Dann sind $\iota_{F_{\text{wk}}}$, $\iota_{U_{\text{wk}}}$ und $\iota_{P_{\text{wk}}}$ auf Grund der Voraussetzungen an S bereits kompakt. Insbesondere kann bei $\dim F = \infty$ nicht $F = F_{\text{wk}}$, bei $\dim U = \infty$ nicht $U = U_{\text{wk}}$ und bei $\dim P = \infty$ nicht $P = P_{\text{wk}}$ gelten.

Beweis: (der Existenz von f_{stg}^t)

Seien $\text{id}_{F_{\text{stg}}^t \rightarrow F}$ stetig, $F_{\text{wk}} := (F_{\text{stg}}^t)'$ und $\iota_{F_{\text{wk}}} := (\cdot)|_{F_{\text{stg}}^t} \circ \tau_F : F \rightarrow F_{\text{wk}}$. Dann gibt es für jedes $f \in F$ ein $f_{\text{stg}}^t \in F_{\text{stg}}^t$ mit

$$\underbrace{\langle f, \cdot \rangle_F|_{F_{\text{stg}}^t}}_{\in (F_{\text{stg}}^t)'} = \langle f_{\text{stg}}^t, \cdot \rangle_{F_{\text{stg}}^t} \quad \text{auf } F_{\text{stg}}^t.$$

Damit

$$\begin{aligned}
\forall g \in F : \quad \langle \iota_{F_{\text{wk}}} g, \iota_{F_{\text{wk}}} f \rangle_{F_{\text{wk}}} &= \underbrace{\langle \langle g, \cdot \rangle_F |_{F_{\text{stg}}^t}, \langle f, \cdot \rangle_F |_{F_{\text{stg}}^t} \rangle_{(F_{\text{stg}}^t)'}}_{=:\langle g_{\text{stg}}^t, \cdot \rangle_{F_{\text{stg}}^t}} =: \langle f_{\text{stg}}^t, \cdot \rangle_{F_{\text{stg}}^t} \\
&= \langle g_{\text{stg}}^t, f_{\text{stg}}^t \rangle_{F_{\text{stg}}^t} \\
&\stackrel{\text{Def. } g_{\text{stg}}^t}{=} \langle g, f_{\text{stg}}^t \rangle_F.
\end{aligned}$$

Weiters

$$\|f_{\text{stg}}^t\|_{F_{\text{stg}}^t}^2 = \langle f_{\text{stg}}^t, \underbrace{f_{\text{stg}}^t}_{\in F_{\text{stg}}^t} \rangle_{F_{\text{stg}}^t} \stackrel{\text{Def. } f}{=} \langle f, f_{\text{stg}}^t \rangle_F = \langle \iota_{F_{\text{wk}}} f, \iota_{F_{\text{wk}}} f \rangle_{F_{\text{wk}}} = \|\iota_{F_{\text{wk}}} f\|_{F_{\text{wk}}}^2.$$

■

2.5 Ein konkretes Optimal-Steuer-Problem: Die Laplace-Gleichung

2.5.1 Das (kontinuierliche) Optimal-Steuer-Problem

Korollar 2.5.1.1 (Das (kontinuierliche) Optimal-Steuer-Problem).

Seien $d \in \{1, 2, 3\}$, $\hat{T} \subseteq \mathbb{R}^d$ das Referenz-Element für Intervall-, Dreiecks- bzw. Tetraeder-Gitter und $\Omega \subseteq \mathbb{R}^d$ ein konvexes Polygon. Wir betrachten die Hilbert-Räume $F := L^2(\Omega)$, $U := P := H_0^1(\Omega)$ und die stetigen Bilinearformen

$$\begin{aligned}
\forall u, p \in H_0^1(\Omega) : \quad a(u, p) &:= \int_{\Omega} \nabla u \cdot \nabla p \, dV, \\
\forall f \in L^2(\Omega), p \in H_0^1(\Omega) : \quad b(f, p) &:= \int_{\Omega} f \cdot p \, dV
\end{aligned}$$

mit $\|a\| \leq 1$ und $\|b\| \leq 1$. Seien weiters $u_0 \in H^{k(u_0)}(\Omega) \cap H_0^1(\Omega)$ für ein $k(u_0)$ mit^a

$$\begin{aligned}
(d = 1) \quad k(u_0) &\geq 2, \\
(d \in \{2, 3\}) \quad k(u_0) &:= 2
\end{aligned}$$

und $\kappa \in (0, 1)$ gegeben. Es bezeichne $B : F \longrightarrow P'$ den von b induzierten Operator.

Dann existiert laut Satz 2.4.1.1 ein eindeutiger Minimierer $f \in L^2(\Omega)$ von

$$J : \begin{cases} L^2(\Omega) & \longrightarrow \mathbb{R} \\ g & \longmapsto (1 - \kappa) \cdot \|u_0 - A^{-1}Bg\|_{H^1(\Omega)}^2 + \kappa \cdot \|g\|_{L^2(\Omega)}^2 \end{cases}.$$

Wir bezeichnen im Folgenden mit $u \in H_0^1(\Omega)$ und $p \in H_0^1(\Omega)$ die entsprechenden Teil-Lösungen des Optimalitäts-Systems aus Satz 2.4.1.1.

Die Lösung (f, u, p) hängt linear und stetig von den Daten u_0 ab mit $\|(f, u, p)\|_{L^2 \times H^1 \times H^1} \leq \frac{C(d, \Omega)}{\kappa(1-\kappa)^2} \cdot \|u_0\|_{H^1}$.

^aIm Fall $d = 1$ setzen wir $k(u_0)$ nicht zwingend als maximal voraus, d.h. es kann durchaus auch $u_0 \in H^k(\Omega)$ für ein $k \geq k(u_0) + 1$ gelten. Es sollte $k(u_0)$ vielmehr nur als Parameter zur Auswahl der Ansatz-Räume F_h , U_h und P_h verstanden werden.

Beweis:

Wir prüfen die Voraussetzungen von Satz 2.4.1.1:

Laut Poincaré-Ungleichung (vgl. Satz 1.1.2.2) ist a koerziv mit $\alpha_{\text{kzv}}(a) \geq C(d, \Omega) > 0$. Laut Lemma 2.1.2.5 gilt insbesondere $\alpha^t(a) \geq \alpha_{\text{kzv}}(a) \geq C(d, \Omega) > 0$ und der von a induzierte Operator $A : U \longrightarrow P'$ ist injektiv.

■

2.5.2 Diskretisierung

Korollar 2.5.2.1 (Optimal-Steuer-Problem Laplace-Gleichung Diskretisierung).

Seien zusätzlich $(\mathcal{T}_h)_{h \in H}$ eine formreguläre, quasi-uniforme Familie von Gittern auf Ω und

$$g_F := k(u_0) - 2 \geq 0, \quad g_U := k(u_0) - 1 \geq 1, \quad g_P := k(u_0) - 1 \geq 1.$$

Wir betrachten die folgenden Ansatz-Räume (vgl. Definition 1.2.2.1):

$$\forall h \in H : \quad F_h := \mathbb{S}^{g_F, 0}(\mathcal{T}_h) \leq F, \quad U_h := \mathbb{S}_0^{g_U, 1}(\mathcal{T}_h) \leq U, \quad P_h := \mathbb{S}_0^{g_P, 1}(\mathcal{T}_h) \leq P.$$

Dann existiert auch eine eindeutige diskrete Lösung $(f_h, u_h, p_h) \in \mathbb{S}^{g_F, 0}(\mathcal{T}_h) \times \mathbb{S}_0^{g_U, 1}(\mathcal{T}_h) \times \mathbb{S}_0^{g_P, 1}(\mathcal{T}_h)$ des diskreten Sattelpunkt-Problems aus Lemma 2.4.2.1.

Beweis:

Wir prüfen die Voraussetzungen von Lemma 2.4.2.1:

Schritt 1: $\dim F_h < \infty$, $\dim U_h < \infty$ und $\dim P_h < \infty$

Das ist klar. □

Schritt 2: a erfüllt $\alpha_h^t(a) > 0$

Die Koerzivität von a garantiert wieder laut Lemma 2.1.2.5 die Relationen $[\forall h \in H : \alpha_h^t(a) \geq \alpha_0^t(a) \geq \alpha_{\text{kzv}}(a) \geq C(d, \Omega) > 0]$. □

■

2.5.3 Konvergenz

Korollar 2.5.3.1 (Optimal-Steuer-Problem Laplace-Gleichung Konvergenz).

Seien nun zusätzlich

$$k_F := k(u_0) - 1 \geq 1, \quad k_U := k(u_0) \geq 2, \quad k_P := k(u_0) \geq 2.$$

Wir betrachten die folgenden Unterräume:

$$F_{\text{stg}} := H^{k_F}(\Omega) \leq F, \quad U_{\text{stg}} := H^{k_U}(\Omega) \cap H_0^1(\Omega) \leq U, \quad P_{\text{stg}} := H^{k_P}(\Omega) \cap H_0^1(\Omega) \leq P.$$

Dann gilt laut Satz 2.4.3.1 die Fehler-Abschätzung

$$\|f - f_h\|_{L^2(\Omega)} + \|u - u_h\|_{H^1(\Omega)} + \|p - p_h\|_{H^1(\Omega)} \leq \frac{C(d, \Omega, k(u_0), \hat{T}, \gamma)}{\kappa \cdot (1 - \kappa)^3} \cdot \|(f, u, p)\|_{H^{k_F} \times H^{k_U} \times H^{k_P}} \cdot h_{\mathcal{T}_h}^{g_U}.$$

Beweis:

Wir prüfen die Voraussetzungen von Satz 2.4.3.1:

Schritt 1: $F_{\text{stg}} \leq F$, $U_{\text{stg}} \leq U$ und $P_{\text{stg}} \leq P$ jeweils dicht

Vgl. Satz 1.1.2.2. □

Schritt 2: $\|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0$, $\|(\text{id} - P_{U_h})|_{U_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0$, $\|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\| \xrightarrow{h \rightarrow 0} 0$

Die Orthogonal-Projektionen $P_{F_h} : F \rightarrow F_h$, $P_{U_h} : U \rightarrow U_h$ und $P_{P_h} : P \rightarrow P_h$ erfüllen laut Korollar

1.2.7.2 und Korollar 1.2.9.2 die Abschätzungen

$$\begin{aligned}\|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| &\leq C(d, \hat{T}, k(u_0)) \cdot h_{\mathcal{T}_h}^{g_F+1}, \\ \|(\text{id} - P_{U_h})|_{U_{\text{stg}}}\| &\leq C(d, \hat{T}, \gamma, k(u_0)) \cdot h_{\mathcal{T}_h}^{g_U}, \\ \|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\| &\leq C(d, \hat{T}, \gamma, k(u_0)) \cdot h_{\mathcal{T}_h}^{g_P}.\end{aligned}$$

□

Schritt 3: a erfüllt $\alpha_0^t(a) > 0$

Wir haben in Korollar 2.5.2.1 bereits gesehen, dass für die obigen Ansatz-Räume gilt: $\alpha_0^t(a) \geq C(d, \Omega) > 0$.

□

Schritt 4: $(f, u, p) \in F_{\text{stg}} \times U_{\text{stg}} \times P_{\text{stg}}$

Das Optimalitäts-Systems aus Satz 2.4.1.1 vereinfacht sich wegen $\langle \cdot, \cdot \rangle_{H^1} = a(\cdot, \cdot) + \langle \cdot, \cdot \rangle_{L^2}$ zu folgendem System:

$$\begin{aligned}\frac{\kappa}{1-\kappa} \cdot f &= p, \\ a(\cdot, p - (u_0 - u)) &= \langle u_0 - u, \cdot \rangle_{L^2} \quad \text{auf } H_0^1, \\ a(u, \cdot) &= \langle f, \cdot \rangle_{L^2} \quad \text{auf } H_0^1.\end{aligned}$$

Die spezielle Form dieses Systems zeigt, dass die kontinuierliche Lösung (f, u, p) die Regularität $(f, u, p) \in F_{\text{stg}} \times U_{\text{stg}} \times P_{\text{stg}}$ besitzt:

1. *Fall: $d = 1$:* Eine einfache Rechnung zeigt, dass für jedes $k \geq 0$ und $f \in H^k$ die Lösung $u \in H_0^1$ von $[a(u, \cdot) = \langle f, \cdot \rangle_{L^2} \text{ auf } H_0^1]$ die Regularität $u \in H^{k+2}$ besitzt, wobei $\|u\|_{H^{k+2}} \leq C(d, \Omega) \cdot \|f\|_{H^k}$. Für die Lösung (f, u, p) des obigen Systems gelten daher die folgenden Implikationen:

$$\forall k \geq 0 : \quad [u_0, u \in H^k \Rightarrow p \in H^k, u \in H^{k+2}].$$

Sukzessives Anwenden dieser Implikationen liefert⁷

$$u \in H^{k(u_0)+2} \subseteq H^{k_U}, \quad p \in H^{k(u_0)} = H^{k_P}, \quad f = \frac{1-\kappa}{\kappa} \cdot p \in H^{k(u_0)} \subseteq H^{k_F}.$$

2. *Fall: $d \in \{2, 3\}$:* Die Konvexität von Ω liefert wegen Theorem 3.2.1.2. in [Grisvard, 1985], dass für jedes $f \in H^0$ die Lösung $u \in H_0^1$ von $[a(u, \cdot) = \langle f, \cdot \rangle_{L^2} \text{ auf } H_0^1]$ die Regularität $u \in H^2$ besitzt, wobei $\|u\|_{H^2} \leq C(d, \Omega) \cdot \|f\|_{H^0}$. Für die Lösung (f, u, p) des obigen Systems gilt daher die folgende Implikation:

$$[u_0 \in H^2 \Rightarrow p \in H^2].$$

Damit

$$u \in H^2 = H^{k_U}, \quad p \in H^2 = H^{k_P}, \quad f = \frac{1-\kappa}{\kappa} \cdot p \in H^2 \subseteq H^{k_F}.$$

□

■

⁷Da die Anteile f und u der kontinuierlichen Lösung die Regularitäten $f \in H^{k(u_0)}$ und $u \in H^{k(u_0)+2}$ besitzen, würde die Wahl $g_F := k(u_0) - 1$ und $g_U := k(u_0)$ der Polynom-Grade zu einer besseren Konvergenz-Rate von $\|(\text{id} - P_{F_h})|_{F_{\text{stg}}}\| \leq \mathcal{O}(h_{\mathcal{T}_h}^{k(u_0)})$ und $\|(\text{id} - P_{U_h})|_{U_{\text{stg}}}\| \leq \mathcal{O}(h_{\mathcal{T}_h}^{k(u_0)})$ führen. Das würde aber wegen $\|(\text{id} - P_{P_h})|_{P_{\text{stg}}}\| \leq \mathcal{O}(h_{\mathcal{T}_h}^{k(u_0)-1})$ die Gesamt-Konvergenz-Rate von $\|f - f_h\|_{L^2(\Omega)} + \|u - u_h\|_{H^1(\Omega)} + \|p - p_h\|_{H^1(\Omega)}$ nicht erhöhen. Die zu geringe Regularität des Anteils p der kontinuierlichen Lösung legt also eine obere Schranke für die Gesamt-Konvergenz-Rate fest.

2.5.4 Aubin-Nitsche-Trick

Korollar 2.5.4.1 (Optimal-Steuer-Problem Laplace-Gleichung Aubin-Nitsche-Trick).

Wir betrachten zusätzlich die folgenden Hilbert-Räume und Einbettungen:

$$\begin{aligned} F_{\text{wk}} &:= (H^1(\Omega))', & U_{\text{wk}} &:= L^2(\Omega), & P_{\text{wk}} &:= L^2(\Omega), \\ \iota_{F_{\text{wk}}} &:= (\cdot)|_{H^1 \circ \tau_{L^2}}, & \iota_{U_{\text{wk}}} &:= \text{id}_{H_0^1 \rightarrow L^2}, & \iota_{P_{\text{wk}}} &:= \text{id}_{H_0^1 \rightarrow L^2}, \\ F_{\text{stg}}^t &:= H^1(\Omega), & U_{\text{stg}}^t &:= H^2(\Omega) \cap H_0^1(\Omega), & P_{\text{stg}}^t &:= H^2(\Omega) \cap H_0^1(\Omega). \end{aligned}$$

Dann gilt laut Satz 2.4.4.1 die Fehler-Abschätzung

$$\begin{aligned} &\|\iota_{(H^1)'}(f - f_h)\|_{(H^1(\Omega))'} \\ &+ \|u - u_h\|_{L^2(\Omega)} + \|p - p_h\|_{L^2(\Omega)} \leq \frac{C(d, \Omega, k(u_0), \hat{T}, \gamma)}{\kappa^3 \cdot (1 - \kappa)^6} \cdot \|(f, u, p)\|_{H^k_F \times H^k_U \times H^k_P} \cdot h_{\mathcal{T}_h}^{g_U+1}. \end{aligned}$$

Beweis:

Wir prüfen die Voraussetzungen von Satz 2.4.4.1:

Schritt 1: $\iota_{F_{\text{wk}}} : F \longrightarrow F_{\text{wk}}, \iota_{U_{\text{wk}}} : U \longrightarrow U_{\text{wk}}, \iota_{P_{\text{wk}}} : P \longrightarrow P_{\text{wk}}$ linear und stetig

Die Einbettungen $\iota_{F_{\text{wk}}}, \iota_{U_{\text{wk}}}$ und $\iota_{P_{\text{wk}}}$ sind linear und stetig mit $\|\iota_{F_{\text{wk}}}\| \leq 1, \|\iota_{U_{\text{wk}}}\| \leq 1$ und $\|\iota_{P_{\text{wk}}}\| \leq 1$. $\square$

Schritt 2: $F_{\text{stg}}^t \leq F, U_{\text{stg}}^t \leq U$ und $P_{\text{stg}}^t \leq P$

Das ist klar. $\square$

Schritt 3: $\|(\text{id} - P_{F_h})|_{F_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0, \quad \|(\text{id} - P_{U_h})|_{U_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0, \quad \|(\text{id} - P_{P_h})|_{P_{\text{stg}}^t}\| \xrightarrow{h \rightarrow 0} 0$

Die Orthogonal-Projektionen $P_{F_h} : F \longrightarrow F_h, P_{U_h} : U \longrightarrow U_h$ und $P_{P_h} : P \longrightarrow P_h$ erfüllen laut Korollar 1.2.7.2 und Korollar 1.2.9.2 die Abschätzungen

$$\begin{aligned} \|(\text{id} - P_{F_h})|_{F_{\text{stg}}^t}\| &\leq C(d, \hat{T}, k(u_0)) \cdot h_{\mathcal{T}_h}, \\ \|(\text{id} - P_{U_h})|_{U_{\text{stg}}^t}\| &\leq C(d, \hat{T}, \gamma, k(u_0)) \cdot h_{\mathcal{T}_h}, \\ \|(\text{id} - P_{P_h})|_{P_{\text{stg}}^t}\| &\leq C(d, \hat{T}, \gamma, k(u_0)) \cdot h_{\mathcal{T}_h}. \end{aligned}$$

$\square$

Schritt 4: Voraussetzungen an S

Die äquivalente Form des transponierten Sattelpunkt-Problems aus Bemerkung 2.4.7.1 vereinfacht sich wegen $\langle \cdot, \cdot \rangle_{H^1} = a(\cdot, \cdot) + \langle \cdot, \cdot \rangle_{L^2}$ zu folgendem System:

Für geg. $(f, u, p) \in L^2 \times H_0^1 \times H_0^1$, finde $(g, v, q) \in L^2 \times H_0^1 \times H_0^1$ mit :

$$\begin{aligned} \frac{\kappa}{1-\kappa} \cdot g - q &= f_{\text{stg}}^t, \\ a(\cdot, v + q) &= \langle \cdot, u - v \rangle_{L^2} \quad \text{auf } H_0^1, \\ a(v, \cdot) &= \langle \cdot, g + p \rangle_{L^2} \quad \text{auf } H_0^1. \end{aligned}$$

Dabei ist $f_{\text{stg}}^t \in F_{\text{stg}}^t = H^1$ eine Funktion mit $[\langle \iota_{(H^1)'}(\cdot), \iota_{(H^1)'} f \rangle_{(H^1)'} = \langle \cdot, f_{\text{stg}}^t \rangle_{L^2} \text{ auf } L^2]$ und $\|f_{\text{stg}}^t\|_{H^1} = \|\iota_{(H^1)'} f\|_{(H^1)'}$ (vgl. erneut die Bemerkung 2.4.7.1).

Die spezielle Form dieses Systems zeigt, dass für jedes $(f, u, p) \in L^2 \times H_0^1 \times H_0^1$ die Lösung (g, v, q) die Regularität $(g, v, q) \in F_{\text{stg}}^t \times U_{\text{stg}}^t \times P_{\text{stg}}^t$ besitzt und dass eine Abschätzung der Form $\|(g, v, q)\|_{F_{\text{stg}}^t \times U_{\text{stg}}^t \times P_{\text{stg}}^t} \leq C_S \cdot \|(\iota_{F_{\text{wk}}} f, \iota_{U_{\text{wk}}} u, \iota_{P_{\text{wk}}} p)\|_{F_{\text{wk}} \times U_{\text{wk}} \times P_{\text{wk}}}$ gilt: Die erwähnten Regularisierungseigenschaften der Bilinearform a (vgl. Korollar 2.5.3.1) liefern

$$\begin{aligned} v &\in H^2, & \|v\|_{H^2} &\leq C(d, \Omega) \cdot \|g + p\|_{L^2}, \\ q &\in H^2, & \|v + q\|_{H^2} &\leq C(d, \Omega) \cdot \|u - v\|_{L^2}, \\ g &= \frac{1-\kappa}{\kappa} \cdot (q + f_{\text{stg}}) \in H^1, & \|g\|_{H^1} &= \frac{1-\kappa}{\kappa} \cdot \|q + f_{\text{stg}}\|_{H^1}. \end{aligned}$$

Zusammen mit der Abschätzung

$$\|(g, v, q)\|_{L^2 \times H^1 \times H^1} \leq \frac{C(d, \Omega)}{\kappa \cdot (1 - \kappa)^2} \cdot \|(\iota_{(H^1)'} f, u, p)\|_{(H^1)' \times L^2 \times L^2}$$

aus Satz 2.4.4.1 erhalten wir

$$\begin{aligned} \|(g, v, q)\|_{H^1 \times H^2 \times H^2} &\leq \|g\|_{H^1} + \|v\|_{H^2} + \|q\|_{H^2} \\ &\leq \|g\|_{H^1} + 2 \cdot \|v\|_{H^2} + \|v + q\|_{H^2} \\ &\leq \frac{C(d, \Omega)}{\kappa} \cdot [\|q + f_{\text{stg}}^t\|_{H^1} + \|g + p\|_{L^2} + \|u - v\|_{L^2}] \\ &\leq \frac{C(d, \Omega)}{\kappa} \cdot [\|g\|_{L^2} + 1 \cdot \|v\|_{H^1} + \|q\|_{H^1} \\ &\quad + \|f_{\text{stg}}^t\|_{H^1} + \|u\|_{L^2} + \|p\|_{L^2}] \\ &\lesssim \frac{C(d, \Omega)}{\kappa} \cdot [\|(g, v, q)\|_{L^2 \times H^1 \times H^1} \\ &\quad + \|(\iota_{(H^1)'} f, u, p)\|_{(H^1)' \times L^2 \times L^2}] \\ &\lesssim \frac{C(d, \Omega)}{\kappa^2 \cdot (1 - \kappa)^2} \cdot \|(\iota_{(H^1)'} f, u, p)\|_{(H^1)' \times L^2 \times L^2}. \end{aligned}$$

□

■

2.5.5 LGS

Korollar 2.5.5.1 (Optimal-Steuer-Problem Laplace-Gleichung LGS).

Wir wählen nun die Basen aus Satz 1.2.8.1 und Satz 1.2.10.2:^a

$$\begin{array}{llll} (d = 1) & \{f_1, \dots, f_{n_F}\} & := \{\psi_{T,r}^{g_F} \mid T \in \mathcal{T}_h, r \in I_{g_F}\} & \subseteq \mathbb{S}^{g_F,0}(\mathcal{T}_h) = F_h, \\ & \{u_1, \dots, u_{n_U}\} & := \{\varphi_N^{g_U} \mid N \in \mathcal{N}_{I,h}\} \cup \{\varphi_{T,s}^{g_U} \mid T \in \mathcal{T}_h, s \in I_{g_U}^\times\} & \subseteq \mathbb{S}_0^{g_U,1}(\mathcal{T}_h) = U_h, \\ & \{p_1, \dots, p_{n_P}\} & := \{\varphi_N^{g_P} \mid N \in \mathcal{N}_{I,h}\} \cup \{\varphi_{T,s}^{g_P} \mid T \in \mathcal{T}_h, s \in I_{g_P}^\times\} & \subseteq \mathbb{S}_0^{g_P,1}(\mathcal{T}_h) = P_h, \\ (d \in \{2, 3\}) & \{f_1, \dots, f_{n_F}\} & := \{\psi_{T,00(0)}^0 \mid T \in \mathcal{T}_h\} & \subseteq \mathbb{S}^{0,0}(\mathcal{T}_h) = F_h, \\ & \{u_1, \dots, u_{n_U}\} & := \{\varphi_N^1 \mid N \in \mathcal{N}_{I,h}\} & \subseteq \mathbb{S}_0^{1,1}(\mathcal{T}_h) = U_h, \\ & \{p_1, \dots, p_{n_P}\} & := \{\varphi_N^1 \mid N \in \mathcal{N}_{I,h}\} & \subseteq \mathbb{S}_0^{1,1}(\mathcal{T}_h) = P_h. \end{array}$$

Damit können wir die Konditionszahl der System-Matrix aus Lemma 2.4.5.1 abschätzen:

$$\text{cond} \begin{pmatrix} F & 0 & B^T \\ 0 & U & A^T \\ B & A & 0 \end{pmatrix} \leq \frac{C(d, \Omega, k(u_0), \hat{T}, \gamma, \nu)}{\kappa \cdot (1 - \kappa)^3} \cdot \frac{1}{h_{\min, \mathcal{T}_h}^2}.$$

^aZur besseren Unterscheidung verwenden wir für die Basis-Funktionen von $\mathbb{S}^{g,0}(\mathcal{T}_h)$ den Buchstaben „ ψ “ und für jene von $\mathbb{S}_0^{g,1}(\mathcal{T}_h)$ den Buchstaben „ φ “.

Beweis:

Die zu den angegebenen Basen zugehörigen Koordinaten-Abbildungen $\gamma_h : F_h \rightarrow \mathbb{R}^{\dim F_h}$, $\mu_h : U_h \rightarrow \mathbb{R}^{\dim U_h}$ und $\pi_h : P_h \rightarrow \mathbb{R}^{\dim P_h}$ erfüllen laut Satz 1.2.8.1 und Satz 1.2.10.2 die Abschätzung

$$(\|\gamma_h\|^2 + \|\mu_h\|^2 + \|\pi_h\|^2) \cdot (\|\gamma_h^{-1}\|^2 + \|\mu_h^{-1}\|^2 + \|\pi_h^{-1}\|^2) \leq C(d, \hat{T}, k(u_0), \gamma, \nu) \cdot \frac{1}{h_{\min, \mathcal{T}_h}^2}.$$

Die behauptete Abschätzung der Konditionszahl folgt dann unmittelbar aus jener in Lemma 2.4.5.1.

■

2.5.6 Strang-Lemma

Korollar 2.5.6.1 (Optimal-Steuer-Problem Laplace-Gleichung Strang-Lemma).

Wir definieren

$$\begin{aligned} & \text{(falls } u_0 \in H^{k(u_0)} \setminus H^{k(u_0)+1}) & g(u_0) &:= g_U, \\ & \text{(falls } u_0 \in H^{k(u_0)+1}) & g(u_0) &:= g_U + 1 \end{aligned}$$

und bezeichnen wieder mit $I_{h,g(u_0)} : C^0(\overline{\Omega}) \rightarrow \mathbb{S}^{g(u_0),1}(\mathcal{T}_h)$ den Interpolations-Operator aus Satz 1.2.9.1. Wir betrachten die Approximationen

$$\forall h \in H : \quad u_{0,h} := I_{h,g(u_0)} u_0 \in U.$$

Dann gelten laut Lemma 2.4.6.1 für die diskreten Lösungen $(\tilde{f}_h, \tilde{u}_h, \tilde{p}_h) \in \mathbb{S}^{g_F,0}(\mathcal{T}_h) \times \mathbb{S}_0^{g_U,1}(\mathcal{T}_h) \times \mathbb{S}_0^{g_P,1}(\mathcal{T}_h)$ der diskreten Sattelpunkt-Probleme mit rechten Seiten $\langle I_{h,g(u_0)} u_0, \cdot \rangle_{H^1}$ die Fehler-Abschätzungen (in beiden Fällen)

$$\begin{aligned} \|f - \tilde{f}_h\|_{L^2(\Omega)} + \|u - \tilde{u}_h\|_{H^1(\Omega)} + \|p - \tilde{p}_h\|_{H^1(\Omega)} &\leq \frac{C(d, \Omega, k(u_0), \hat{T}, \gamma)}{\kappa \cdot (1 - \kappa)^3} \cdot h_{\mathcal{T}_h}^{g_U} \\ &\quad \cdot (\|(f, u, p)\|_{H^{k_F} \times H^{k_U} \times H^{k_P}} + 1) \\ &\quad \cdot (|u_0|_{H^{g(u_0)+1}} + 1) \end{aligned}$$

und (nur für $[u_0 \in H^{k(u_0)+1}, g(u_0) = g_U + 1]$)

$$\begin{aligned} \|\iota_{(H^1)'}(f - \tilde{f}_h)\|_{(H^1(\Omega))'} + \|u - \tilde{u}_h\|_{L^2(\Omega)} + \|p - \tilde{p}_h\|_{L^2(\Omega)} &\leq \frac{C(d, \Omega, k(u_0), \hat{T}, \gamma)}{\kappa^3 \cdot (1 - \kappa)^6} \cdot h_{\mathcal{T}_h}^{g_U+1} \\ &\quad \cdot (\|(f, u, p)\|_{H^{k_F} \times H^{k_U} \times H^{k_P}} + 1) \\ &\quad \cdot (|u_0|_{H^{g(u_0)+1}} + 1). \end{aligned}$$

Beweis:

Wir prüfen die Voraussetzungen von Lemma 2.4.6.1:

Die angegebenen Approximationen $u_{0,h}$ erfüllen laut Satz 1.2.9.1 die Abschätzungen

$$\|u_0 - u_{0,h}\|_{H^1} \leq C(d, k(u_0), \hat{T}, \gamma) \cdot h_{\mathcal{T}_h}^{g(u_0)} \cdot |u_0|_{H^{g(u_0)+1}} \xrightarrow{h \rightarrow 0} 0.$$

■

3 Implementierung in 1D und 2D: Optimale Steuerung der Laplace-Gleichung

3.1 Exakte Lösungen, Assemblierung und Fehler-Berechnung

3.1.1 1D

Lemma 3.1.1.1 (Ein 1D Matlab Beispiel Exakte Lösungen).

Seien $d := 1$, $\Omega := (0, 1) \subseteq \mathbb{R}$ und $\kappa \in (0, 1)$ gegeben. Definiere

$$\begin{aligned} u(x) &:= x^3 \cdot (1-x)^3 \in H_0^1(\Omega), \\ f(x) &:= -u''(x) \in H_0^1(\Omega) \subseteq L^2(\Omega), \\ p(x) &:= \frac{\kappa}{1-\kappa} \cdot f(x) \in H_0^1(\Omega). \end{aligned}$$

Dann ist (f, u, p) genau die Lösung des Optimal-Steuer-Problems aus Korollar 2.5.1.1 bezüglich der Daten

$$u_0(x) := u(x) + \frac{\kappa}{1-\kappa} \cdot \left[792 \cdot \left(\frac{e^x + e^{1-x}}{1+e} - 1 \right) + 360 \cdot x \cdot (1-x) \right] \in H_0^1(\Omega).$$

Es gilt $u_0 \in H^{k(u_0)}(\Omega)$ für alle $k(u_0) \geq 2$.

Beweis:

Wegen $u_0, f, u, p \in C^\infty(\bar{\Omega})$ löst (f, u, p) das Optimal-Steuer-Problem aus Korollar 2.5.1.1 sogar im klassischen Sinne. ■

Lemma 3.1.1.2 (Ein 1D Matlab Beispiel Assemblierung).

Wir verwenden das formreguläre, quasi-uniforme 1D-Gitter aus Beispiel 1.2.1.3 und die Basen aus Lemma 2.5.5.1.

Die Einträge von Steifigkeits-Matrix und Last-Vektor können mit Hilfe der folgenden Formeln berechnet werden:

$$\begin{aligned} \langle \psi_{T,r}^{gF}, \psi_{\tilde{T},\tilde{r}}^{gF} \rangle_{L^2(\Omega)} &= \delta_{[T=\tilde{T}]} \cdot |\nabla F_T| \cdot \langle \hat{\varphi}_r^{gF}, \hat{\varphi}_{\tilde{r}}^{gF} \rangle_{L^2(\hat{T})}, \\ \langle \varphi_N^{gU}, \varphi_{\tilde{N}}^{gU} \rangle_{H^1(\Omega)} &= \sum_{\substack{T \in \mathcal{T}_h: \\ N, \tilde{N} \in \mathcal{N}(T)}} \left[|\nabla F_T| \cdot \langle \hat{\varphi}_{i(gU, N, T)}^{gU}, \hat{\varphi}_{i(gU, \tilde{N}, T)}^{gU} \rangle_{L^2(\hat{T})} \right. \\ &\quad \left. + |\nabla F_T|^{-1} \cdot \langle \nabla \hat{\varphi}_{i(gU, N, T)}^{gU}, \nabla \hat{\varphi}_{i(gU, \tilde{N}, T)}^{gU} \rangle_{L^2(\hat{T})} \right], \\ \langle \varphi_N^{gU}, \varphi_{T,s}^{gU} \rangle_{H^1(\Omega)} &= \delta_{[N \in \mathcal{N}(T)]} \cdot \left[|\nabla F_T| \cdot \langle \hat{\varphi}_{i(gU, N, T)}^{gU}, \hat{\varphi}_s^{gU} \rangle_{L^2(\hat{T})} \right. \\ &\quad \left. + |\nabla F_T|^{-1} \cdot \langle \nabla \hat{\varphi}_{i(gU, N, T)}^{gU}, \nabla \hat{\varphi}_s^{gU} \rangle_{L^2(\hat{T})} \right], \\ \langle \varphi_{T,s}^{gU}, \varphi_{\tilde{T},\tilde{s}}^{gU} \rangle_{H^1(\Omega)} &= \delta_{[T=\tilde{T}]} \cdot \left[|\nabla F_T| \cdot \langle \hat{\varphi}_s^{gU}, \hat{\varphi}_{\tilde{s}}^{gU} \rangle_{L^2(\hat{T})} + |\nabla F_T|^{-1} \cdot \langle \nabla \hat{\varphi}_s^{gU}, \nabla \hat{\varphi}_{\tilde{s}}^{gU} \rangle_{L^2(\hat{T})} \right] \end{aligned}$$

und

$$\begin{aligned}
a(\varphi_N^{g_U}, \varphi_{\tilde{N}}^{g_U}) &= \sum_{\substack{T \in \mathcal{T}_h: \\ N, \tilde{N} \in \mathcal{N}(T)}} |\nabla F_T|^{-1} \cdot \langle \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U}, \nabla \hat{\varphi}_{i(g_U, \tilde{N}, T)}^{g_U} \rangle_{L^2(\hat{T})}, \\
a(\varphi_N^{g_U}, \varphi_{T,s}^{g_U}) &= \delta_{[N \in \mathcal{N}(T)]} \cdot |\nabla F_T|^{-1} \cdot \langle \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U}, \nabla \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})}, \\
a(\varphi_{T,s}^{g_U}, \varphi_{\tilde{T},\tilde{s}}^{g_U}) &= \delta_{[T=\tilde{T}]} \cdot |\nabla F_T|^{-1} \cdot \langle \nabla \hat{\varphi}_s^{g_U}, \nabla \hat{\varphi}_{\tilde{s}}^{g_U} \rangle_{L^2(\hat{T})}, \\
b(\psi_{T,r}^{g_F}, \varphi_N^{g_U}) &= \delta_{[N \in \mathcal{N}(T)]} \cdot |\nabla F_T| \cdot \langle \hat{\varphi}_r^{g_F}, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})}, \\
b(\psi_{T,r}^{g_F}, \varphi_{\tilde{T},s}^{g_U}) &= \delta_{[T=\tilde{T}]} \cdot |\nabla F_T| \cdot \langle \hat{\varphi}_r^{g_F}, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})}
\end{aligned}$$

sowie

$$\begin{aligned}
\langle u_0, \varphi_N^{g_U} \rangle_{H^1(\Omega)} &= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \left[\nabla F_T^{-1} \cdot \langle (\nabla u_0) \circ F_T, \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right. \\
&\quad \left. + \langle u_0 \circ F_T, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right], \\
\langle u_0, \varphi_{T,s}^{g_U} \rangle_{H^1(\Omega)} &= |\nabla F_T| \cdot \left[\nabla F_T^{-1} \cdot \langle (\nabla u_0) \circ F_T, \nabla \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} + \langle u_0 \circ F_T, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} \right], \\
\langle I_{h,g(u_0)} u_0, \varphi_N^{g_U} \rangle_{H^1(\Omega)} &= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \sum_{i \in I_{g(u_0)}} (u_0 \circ F_T)(\hat{x}_i^{g(u_0)}) \cdot \left[\langle \hat{\varphi}_i^{g(u_0)}, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right. \\
&\quad \left. + \nabla F_T^{-2} \cdot \langle \nabla \hat{\varphi}_i^{g(u_0)}, \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right], \\
\langle I_{h,g(u_0)} u_0, \varphi_{T,s}^{g_U} \rangle_{H^1(\Omega)} &= |\nabla F_T| \cdot \sum_{i \in I_{g(u_0)}} (u_0 \circ F_T)(\hat{x}_i^{g(u_0)}) \cdot \left[\langle \hat{\varphi}_i^{g(u_0)}, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} \right. \\
&\quad \left. + \nabla F_T^{-2} \cdot \langle \nabla \hat{\varphi}_i^{g(u_0)}, \nabla \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} \right].
\end{aligned}$$

Beweis:

Wir erinnern an die Definition der Basis-Elemente aus Satz 1.2.8.1 und Satz 1.2.10.2:

$$\begin{aligned}
(g_F \geq 0) \quad \forall T \in \mathcal{T}_h : \forall r \in I_{g_F} : \quad \psi_{T,r}^{g_F} &:= [\hat{\varphi}_r^{g_F} \circ F_T^{-1}] \cdot \mathbb{I}_T && \in \mathbb{S}^{g_F,0}(\mathcal{T}_h), \\
(g_U \geq 1) \quad \forall N \in \mathcal{N}_{I,h} : \quad \varphi_N^{g_U} &:= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} [\hat{\varphi}_{i(g_U, N, T)}^{g_U} \circ F_T^{-1}] \cdot \mathbb{I}_T && \in \mathbb{S}_0^{g_U,1}(\mathcal{T}_h), \\
(g_U \geq 2) \quad \forall T \in \mathcal{T}_h : \forall r \in I_{g_U}^\times : \quad \varphi_{T,s}^{g_U} &:= [\hat{\varphi}_s^{g_U} \circ F_T^{-1}] \cdot \mathbb{I}_T && \in \mathbb{S}_0^{g_U,1}(\mathcal{T}_h).
\end{aligned}$$

Außerdem sei an die Definition der Interpolations-Operatoren aus Lemma 1.2.3.7 und Satz 1.2.9.1 erinnert:

$$\begin{aligned}
\hat{I}_{g(u_0)} : \begin{cases} C^0(\bar{\hat{T}}) & \longrightarrow \mathbb{P}_{g_U}(\hat{T}) \\ v & \longmapsto \sum_{i \in I_{g(u_0)}} v(\hat{x}_i^{g(u_0)}) \cdot \hat{\varphi}_i^{g(u_0)} \end{cases}, \\
I_{h,g(u_0)} : \begin{cases} C^0(\bar{\Omega}) & \longrightarrow \mathbb{S}^{g(u_0),1}(\mathcal{T}_h) \\ v & \longmapsto \sum_{T \in \mathcal{T}_h} [\hat{I}_{g(u_0)}(v \circ F_T) \circ F_T^{-1}] \cdot \mathbb{I}_T \end{cases}.
\end{aligned}$$

Schritt 1: Assemblierung von $\mathbf{F}$

Für die Assemblierung der Matrix $\mathbf{F}$ benötigen wir die folgenden Werte:

$$\langle \psi_{T,r}^{g_F}, \psi_{\tilde{T},\tilde{r}}^{g_F} \rangle_{L^2(\Omega)} = \delta_{[T=\tilde{T}]} \cdot \int_T [\hat{\varphi}_r^{g_F} \circ F_T^{-1}] \cdot [\hat{\varphi}_{\tilde{r}}^{g_F} \circ F_T^{-1}] \, dV = \delta_{[T=\tilde{T}]} \cdot |\nabla F_T| \cdot \langle \hat{\varphi}_r^{g_F}, \hat{\varphi}_{\tilde{r}}^{g_F} \rangle_{L^2(\hat{T})}.$$

□

Schritt 2: Assemblierung von $\mathbf{U}$

Für die Assemblierung der Matrix $\mathbf{U}$ benötigen wir unter Anderem die folgenden Werte:

$$\begin{aligned}\langle \varphi_N^{g_U}, \varphi_{\tilde{N}}^{g_U} \rangle_{L^2(\Omega)} &= \sum_{\substack{T \in \mathcal{T}_h: \\ N, \tilde{N} \in \mathcal{N}(T)}} \int_T [\hat{\varphi}_{i(g_U, N, T)}^{g_U} \circ F_T^{-1}] \cdot [\hat{\varphi}_{i(g_U, \tilde{N}, T)}^{g_U} \circ F_T^{-1}] \, dV \\ &= \sum_{\substack{T \in \mathcal{T}_h: \\ N, \tilde{N} \in \mathcal{N}(T)}} |\nabla F_T| \cdot \langle \hat{\varphi}_{i(g_U, N, T)}^{g_U}, \hat{\varphi}_{i(g_U, \tilde{N}, T)}^{g_U} \rangle_{L^2(\hat{T})}.\end{aligned}$$

Analog

$$\begin{aligned}\langle \varphi_N^{g_U}, \varphi_{T,s}^{g_U} \rangle_{L^2(\Omega)} &= \delta_{[N \in \mathcal{N}(T)]} \cdot |\nabla F_T| \cdot \langle \hat{\varphi}_{i(g_U, N, T)}^{g_U}, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})}, \\ \langle \varphi_{T,s}^{g_U}, \varphi_{\tilde{T},\tilde{s}}^{g_U} \rangle_{L^2(\Omega)} &= \delta_{[T=\tilde{T}]} \cdot |\nabla F_T| \cdot \langle \hat{\varphi}_s^{g_U}, \hat{\varphi}_{\tilde{s}}^{g_U} \rangle_{L^2(\hat{T})}.\end{aligned}$$

□

Schritt 3: Assemblierung von $\mathbf{U}$ und $\mathbf{A}$

Für die Assemblierung der Matrizen $\mathbf{U}$ und $\mathbf{A}$ benötigen wir unter Anderem die folgenden Werte:

$$\begin{aligned}a(\varphi_N^{g_U}, \varphi_{\tilde{N}}^{g_U}) &= \sum_{\substack{T \in \mathcal{T}_h: \\ N, \tilde{N} \in \mathcal{N}(T)}} \int_T \langle \nabla(\hat{\varphi}_{i(g_U, N, T)}^{g_U} \circ F_T^{-1}), \nabla(\hat{\varphi}_{i(g_U, \tilde{N}, T)}^{g_U} \circ F_T^{-1}) \rangle_2 \, dV \\ &= \sum_{\substack{T \in \mathcal{T}_h: \\ N, \tilde{N} \in \mathcal{N}(T)}} |\nabla F_T| \cdot \int_{\hat{T}} \langle \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U} \cdot \nabla F_T^{-1}, \nabla \hat{\varphi}_{i(g_U, \tilde{N}, T)}^{g_U} \cdot \nabla F_T^{-1} \rangle_2 \, dV \\ &= \sum_{\substack{T \in \mathcal{T}_h: \\ N, \tilde{N} \in \mathcal{N}(T)}} |\nabla F_T|^{-1} \cdot \langle \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U}, \nabla \hat{\varphi}_{i(g_U, \tilde{N}, T)}^{g_U} \rangle_{L^2(\hat{T})}.\end{aligned}$$

Analog

$$\begin{aligned}a(\varphi_N^{g_U}, \varphi_{T,s}^{g_U}) &= \delta_{[N \in \mathcal{N}(T)]} \cdot |\nabla F_T|^{-1} \cdot \langle \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U}, \nabla \hat{\varphi}_s^{g_U} \rangle_2, \\ a(\varphi_{T,s}^{g_U}, \varphi_{\tilde{T},\tilde{s}}^{g_U}) &= \delta_{[T=\tilde{T}]} \cdot |\nabla F_T|^{-1} \cdot \langle \nabla \hat{\varphi}_s^{g_U}, \nabla \hat{\varphi}_{\tilde{s}}^{g_U} \rangle_2.\end{aligned}$$

□

Schritt 4: Assemblierung von $\mathbf{B}$

Für die Assemblierung der Matrix $\mathbf{B}$ benötigen wir unter Anderem die folgenden Werte:

$$b(\psi_{T,r}^{g_F}, \varphi_N^{g_U}) = \langle \psi_{T,r}^{g_F}, \varphi_N^{g_U} \rangle_{L^2(\Omega)} = \delta_{[N \in \mathcal{N}(T)]} \cdot |\nabla F_T| \cdot \langle \hat{\varphi}_r^{g_F}, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})}.$$

Analog

$$b(\psi_{T,r}^{g_F}, \varphi_{T,s}^{g_U}) = \delta_{[T=\tilde{T}]} \cdot |\nabla F_T| \cdot \langle \hat{\varphi}_r^{g_F}, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})}.$$

□

Schritt 5: Assemblierung von $\mathbf{u}_0$

Für die Assemblierung des Vektors $\mathbf{u}_0$ benötigen wir unter Anderem die folgenden Werte:

$$\begin{aligned}
\langle u_0, \varphi_N^{g_U} \rangle_{H^1(\Omega)} &= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} \int_T \langle \nabla u_0, \nabla(\hat{\varphi}_{i(g_U, N, T)}^{g_U} \circ F_T^{-1}) \rangle_2 + u_0 \cdot (\hat{\varphi}_{i(g_U, N, T)}^{g_U} \circ F_T^{-1}) \, dV \\
&= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \int_{\hat{T}} \langle (\nabla u_0) \circ F_T, \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U} \cdot \nabla F_T^{-1} \rangle_2 + (u_0 \circ F_T) \cdot \hat{\varphi}_{i(g_U, N, T)}^{g_U} \, dV \\
&= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \left[\nabla F_T^{-1} \cdot \langle (\nabla u_0) \circ F_T, \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right. \\
&\quad \left. + \langle u_0 \circ F_T, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right].
\end{aligned}$$

Analog

$$\langle u_0, \varphi_{T,s}^{g_U} \rangle_{H^1(\Omega)} = |\nabla F_T| \cdot \left[\nabla F_T^{-1} \cdot \langle (\nabla u_0) \circ F_T, \nabla \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} + \langle u_0 \circ F_T, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} \right].$$

□

Schritt 6: Assemblierung von $\mathbf{u}_{0,h}$

Für die Assemblierung des Vektors $\mathbf{u}_{0,h}$ benötigen wir unter Anderem die folgenden Werte:

$$\begin{aligned}
\langle I_{h,g(u_0)} u_0, \varphi_N^{g_U} \rangle_{H^1(\Omega)} &= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} \int_T \langle \nabla(\hat{I}_{g(u_0)}(u_0 \circ F_T) \circ F_T^{-1}), \nabla(\hat{\varphi}_{i(g_U, N, T)}^{g_U} \circ F_T^{-1}) \rangle_2 \, dV \\
&\quad + \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} \int_T [\hat{I}_{g(u_0)}(u_0 \circ F_T) \circ F_T^{-1}] \cdot [\hat{\varphi}_{i(g_U, N, T)}^{g_U} \circ F_T^{-1}] \, dV \\
&= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \int_{\hat{T}} \langle \nabla[\hat{I}_{g(u_0)}(u_0 \circ F_T)] \cdot \nabla F_T^{-1}, \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U} \cdot \nabla F_T^{-1} \rangle_2 \, dV \\
&\quad + \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \int_{\hat{T}} [\hat{I}_{g(u_0)}(u_0 \circ F_T)] \cdot \hat{\varphi}_{i(g_U, N, T)}^{g_U} \, dV \\
&= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \sum_{i \in I_{g(u_0)}} (u_0 \circ F_T)(\hat{x}_i^{g(u_0)}) \cdot \left[\langle \hat{\varphi}_i^{g(u_0)}, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right. \\
&\quad \left. + \nabla F_T^{-2} \cdot \langle \nabla \hat{\varphi}_i^{g(u_0)}, \nabla \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right].
\end{aligned}$$

Analog

$$\begin{aligned}
\langle I_{h,g(u_0)} u_0, \varphi_{T,s}^{g_U} \rangle_{H^1(\Omega)} &= |\nabla F_T| \cdot \sum_{i \in I_{g(u_0)}} (u_0 \circ F_T)(\hat{x}_i^{g(u_0)}) \cdot \left[\langle \hat{\varphi}_i^{g(u_0)}, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} \right. \\
&\quad \left. + \nabla F_T^{-2} \cdot \langle \nabla \hat{\varphi}_i^{g(u_0)}, \nabla \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} \right].
\end{aligned}$$

□

■

Lemma 3.1.1.3 (Ein 1D Matlab Beispiel Fehlerberechnung).

Wir bezeichnen mit $(\gamma_{T,r})_{T \in \mathcal{T}_h, r \in I_{g_F}} \subseteq \mathbb{R}$, $((\mu_N)_{N \in \mathcal{N}_{I,h}}, (\mu_{T,s})_{T \in \mathcal{T}_h, s \in I_{g_U}^\times}) \subseteq \mathbb{R}$ und $((\pi_N)_{N \in \mathcal{N}_{I,h}}, (\pi_{T,s})_{T \in \mathcal{T}_h, s \in I_{g_P}^\times}) \subseteq \mathbb{R}$ die Koeffizienten aus den Darstellungen

$$\begin{aligned} f_h &= \sum_{\substack{T \in \mathcal{T}_h, \\ r \in I_{g_F}}} \gamma_{T,r} \cdot \psi_{T,r}^{g_F}, \\ u_h &= \sum_{N \in \mathcal{N}_{I,h}} \mu_N \cdot \varphi_N^{g_U} + \sum_{\substack{T \in \mathcal{T}_h, \\ s \in I_{g_U}^\times}} \mu_{T,s} \cdot \varphi_{T,s}^{g_U}, \\ p_h &= \sum_{N \in \mathcal{N}_{I,h}} \pi_N \cdot \varphi_N^{g_P} + \sum_{\substack{T \in \mathcal{T}_h, \\ s \in I_{g_P}^\times}} \pi_{T,s} \cdot \varphi_{T,s}^{g_P} \end{aligned}$$

aus Korollar 2.5.5.1.

Dann können die Fehler

$$\begin{aligned} \|f - f_h\|_{L^2(\Omega)}^2 &= \|f\|_{L^2(\Omega)}^2 - 2 \cdot \langle f, f_h \rangle_{L^2(\Omega)} + \|f_h\|_{L^2(\Omega)}^2, \\ \|u - u_h\|_{L^2(\Omega)}^2 &= \|u\|_{L^2(\Omega)}^2 - 2 \cdot \langle u, u_h \rangle_{L^2(\Omega)} + \|u_h\|_{L^2(\Omega)}^2, \\ \|u - u_h\|_{H^1(\Omega)}^2 &= \|u\|_{H^1(\Omega)}^2 - 2 \cdot \langle u, u_h \rangle_{H^1(\Omega)} + \|u_h\|_{H^1(\Omega)}^2, \\ \|p - p_h\|_{L^2(\Omega)}^2 &= \|p\|_{L^2(\Omega)}^2 - 2 \cdot \langle p, p_h \rangle_{L^2(\Omega)} + \|p_h\|_{L^2(\Omega)}^2, \\ \|p - p_h\|_{H^1(\Omega)}^2 &= \|p\|_{H^1(\Omega)}^2 - 2 \cdot \langle p, p_h \rangle_{H^1(\Omega)} + \|p_h\|_{H^1(\Omega)}^2 \end{aligned}$$

mit Hilfe der folgenden Formeln berechnet werden:

$$\begin{aligned} \|f_h\|_{L^2(\Omega)}^2 &= \frac{1 - \kappa}{\kappa} \cdot \boldsymbol{\gamma}^T \cdot \mathbf{F} \cdot \boldsymbol{\gamma}, \\ \|u_h\|_{L^2(\Omega)}^2 &= \boldsymbol{\mu}^T \cdot (\mathbf{U} - \mathbf{A}) \cdot \boldsymbol{\mu}, \\ \|u_h\|_{H^1(\Omega)}^2 &= \boldsymbol{\mu}^T \cdot \mathbf{U} \cdot \boldsymbol{\mu}, \\ \|p_h\|_{L^2(\Omega)}^2 &= \boldsymbol{\pi}^T \cdot (\mathbf{U} - \mathbf{A}) \cdot \boldsymbol{\pi}, \\ \|p_h\|_{H^1(\Omega)}^2 &= \boldsymbol{\pi}^T \cdot \mathbf{U} \cdot \boldsymbol{\pi} \end{aligned}$$

und

$$\begin{aligned}
\langle f, f_h \rangle_{L^2(\Omega)} &= \sum_{T \in \mathcal{T}_h} \sum_{r \in I_{g_F}} \gamma_{T,r} \cdot |\nabla F_T| \cdot \langle f \circ F_T, \hat{\varphi}_r^{g_F} \rangle_{L^2(\hat{T})}, \\
\langle u, u_h \rangle_{L^2(\Omega)} &= \sum_{N \in \mathcal{N}_{I,h}} \mu_N \cdot \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \langle u \circ F_T, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \\
&\quad + \sum_{T \in \mathcal{T}_h} \sum_{s \in I_{g_U}^\times} \mu_{T,s} \cdot |\nabla F_T| \cdot \langle u \circ F_T, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})}, \\
\langle u, u_h \rangle_{H^1(\Omega)} &= \sum_{N \in \mathcal{N}_{I,h}} \mu_N \cdot \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \left[\langle u \circ F_T, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right. \\
&\quad \left. + \nabla F_T^{-1} \cdot \langle u' \circ F_T, (\hat{\varphi}_{i(g_U, N, T)}^{g_U})' \rangle_{L^2(\hat{T})} \right] \\
&\quad + \sum_{T \in \mathcal{T}_h} \sum_{s \in I_{g_U}^\times} \mu_{T,s} \cdot |\nabla F_T| \cdot \left[\langle u \circ F_T, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} + \nabla F_T^{-1} \cdot \langle u' \circ F_T, (\hat{\varphi}_s^{g_U})' \rangle_{L^2(\hat{T})} \right], \\
\langle p, p_h \rangle_{L^2(\Omega)} &= \sum_{N \in \mathcal{N}_{I,h}} \pi_N \cdot \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \langle p \circ F_T, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \\
&\quad + \sum_{T \in \mathcal{T}_h} \sum_{s \in I_{g_U}^\times} \pi_{T,s} \cdot |\nabla F_T| \cdot \langle p \circ F_T, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})}, \\
\langle p, p_h \rangle_{H^1(\Omega)} &= \sum_{N \in \mathcal{N}_{I,h}} \pi_N \cdot \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\nabla F_T| \cdot \left[\langle p \circ F_T, \hat{\varphi}_{i(g_U, N, T)}^{g_U} \rangle_{L^2(\hat{T})} \right. \\
&\quad \left. + \nabla F_T^{-1} \cdot \langle p' \circ F_T, (\hat{\varphi}_{i(g_U, N, T)}^{g_U})' \rangle_{L^2(\hat{T})} \right] \\
&\quad + \sum_{T \in \mathcal{T}_h} \sum_{s \in I_{g_U}^\times} \pi_{T,s} \cdot |\nabla F_T| \cdot \left[\langle p \circ F_T, \hat{\varphi}_s^{g_U} \rangle_{L^2(\hat{T})} + \nabla F_T^{-1} \cdot \langle p' \circ F_T, (\hat{\varphi}_s^{g_U})' \rangle_{L^2(\hat{T})} \right].
\end{aligned}$$

Beweis:

Es gilt

$$\|f_h\|_{L^2(\Omega)}^2 = \sum_{\substack{T \in \mathcal{T}_h, \hat{T} \in \mathcal{T}_h, \\ r \in I_{g_F}, \tilde{r} \in I_{g_F}}} \gamma_{T,r} \cdot \langle \psi_{T,r}^{g_F}, \psi_{\hat{T},\tilde{r}}^{g_F} \rangle_{L^2(\Omega)} \cdot \gamma_{\hat{T},\tilde{r}} = \frac{1-\kappa}{\kappa} \cdot \gamma^T \cdot \mathbf{F} \cdot \gamma$$

und

$$\langle f, f_h \rangle_{L^2(\Omega)} = \sum_{T \in \mathcal{T}_h} \sum_{r \in I_{g_F}} \gamma_{T,r} \cdot \langle f, \psi_{T,r}^{g_F} \rangle_{L^2(T)} = \sum_{T \in \mathcal{T}_h} \sum_{r \in I_{g_F}} \gamma_{T,r} \cdot |\nabla F_T| \cdot \langle f \circ F_T, \hat{\varphi}_r^{g_F} \rangle_{L^2(\hat{T})}.$$

Die Formeln für $\|u_h\|_{L^2(\Omega)}^2$, $\|u_h\|_{H^1(\Omega)}^2$, $\|p_h\|_{L^2(\Omega)}^2$, $\|p_h\|_{H^1(\Omega)}^2$ und $\langle u, u_h \rangle_{L^2(\Omega)}$, $\langle u, u_h \rangle_{H^1(\Omega)}$, $\langle p, p_h \rangle_{L^2(\Omega)}$, $\langle p, p_h \rangle_{H^1(\Omega)}$ zeigt man analog. ■

3.1.2 2D

Lemma 3.1.2.1 (Ein 2D Matlab Beispiel Exakte Lösungen).

Seien $d := 2$, $\Omega := (0, 1) \times (0, 1) \subseteq \mathbb{R}^2$ und $\kappa \in (0, 1)$ gegeben. Definiere

$$\begin{aligned} u(x, y) &:= x^3 \cdot (1-x)^3 \cdot y^3 \cdot (1-y)^3 \in H_0^1(\Omega), \\ f(x, y) &:= -\Delta u(x, y) \in H_0^1(\Omega) \subseteq L^2(\Omega), \\ p(x, y) &:= \frac{\kappa}{1-\kappa} \cdot f(x, y) \in H_0^1(\Omega). \end{aligned}$$

Definiere weiters

$$q(x, y) := \sum_{i=0}^8 \sum_{j=0}^{8-i} q_{ij} x^i y^j := -\Delta p(x, y) \in \mathbb{P}_8(\Omega)$$

und für alle $m \geq 1$:

$$\begin{aligned} \alpha_m^0 &:= \langle 1, \sin(\pi m x) \rangle_{L^2(0,1)} = \frac{1 - (-1)^m}{\pi m}, \\ \alpha_m^1 &:= \langle x, \sin(\pi m x) \rangle_{L^2(0,1)} = -\frac{(-1)^m}{\pi m}, \\ \forall i \geq 2: \quad \alpha_m^i &:= \langle x^i, \sin(\pi m x) \rangle_{L^2(0,1)} = -\frac{i \cdot (i-1)}{\pi^2 m^2} \cdot \alpha_m^{i-2} - \frac{(-1)^m}{\pi m}. \end{aligned}$$

Dann ist (f, u, p) genau die Lösung des Optimal-Steuer-Problems aus Korollar 2.5.1.1 bezüglich der Daten

$$u_0(x, y) := u(x, y) + \sum_{i=0}^8 \sum_{j=0}^{8-i} q_{ij} \cdot \sum_{m,n=1}^{\infty} \frac{1}{1 + \pi^2(m^2 + n^2)} \cdot \alpha_m^i \cdot \alpha_n^j \cdot 4 \cdot \sin(\pi m x) \cdot \sin(\pi n y) \in H_0^1(\Omega).$$

Es gilt $u_0 \in H^2(\Omega)$.

Beweis:

Die Funktionen

$$\{e_{m,n}(x, y) := 2 \cdot \sin(\pi m x) \cdot \sin(\pi n y) \mid m, n \geq 1\}$$

bilden eine $\langle \cdot, \cdot \rangle_{L^2(\Omega)}$ -ONB von $L^2(\Omega)$ und sind Eigenvektoren zu den Eigenwerten $\lambda_{m,n} := \pi^2(m^2 + n^2)$ des Laplace-Operators:

$$-\Delta e_{m,n} = \lambda_{m,n} \cdot e_{m,n}.$$

Die zweite Gleichung des Optimalitäts-Systems aus Satz 2.4.1.1 lautet mit $v := u_0 - u \in H_0^1(\Omega)$ in starker Form:

$$\begin{aligned} -\Delta v + v &= q \quad \text{in } \Omega, \\ v|_{\partial\Omega} &= 0. \end{aligned}$$

Die Konvexität von Ω liefert wegen Theorem 3.2.1.2. in [Grisvard, 1985] die Regularität $v \in H^2(\Omega)$, also insbesondere auch $u_0 \in H^2(\Omega)$.

Aus den Fourier-Reihendarstellungen

$$v = \sum_{m,n=1}^{\infty} \langle v, e_{m,n} \rangle_{L^2(\Omega)} \cdot e_{m,n}, \quad q = \sum_{m,n=1}^{\infty} \langle q, e_{m,n} \rangle_{L^2(\Omega)} \cdot e_{m,n}$$

ergibt sich

$$\begin{aligned}
\sum_{m,n=1}^{\infty} \langle q, e_{m,n} \rangle_{L^2(\Omega)} \cdot e_{m,n} &= q \\
&= -\Delta v + v \\
&= \sum_{m,n=1}^{\infty} (1 + \lambda_{m,n}) \cdot \langle v, e_{m,n} \rangle_{L^2(\Omega)} \cdot e_{m,n}
\end{aligned}$$

und damit

$$\begin{aligned}
v &= \sum_{m,n=1}^{\infty} \langle v, e_{m,n} \rangle_{L^2(\Omega)} \cdot e_{m,n} \\
&= \sum_{m,n=1}^{\infty} \frac{1}{1 + \lambda_{m,n}} \cdot \langle q, e_{m,n} \rangle_{L^2(\Omega)} \cdot e_{m,n} \\
&= \sum_{i=0}^8 \sum_{j=0}^{8-i} q_{ij} \cdot \sum_{m,n=1}^{\infty} \frac{1}{1 + \pi^2(m^2 + n^2)} \cdot \alpha_m^i \cdot \alpha_n^j \cdot 4 \cdot \sin(\pi m x) \cdot \sin(\pi n y).
\end{aligned}$$

■

Lemma 3.1.2.2 (Ein 2D Matlab Beispiel Assemblierung).

Wir verwenden das formreguläre, quasi-uniforme 2D-Gitter aus Beispiel 1.2.1.3 und die Basen aus Lemma 2.5.5.1.

Die Einträge von Steifigkeits-Matrix und Last-Vektor können mit Hilfe der folgenden Formeln berechnet werden:

$$\begin{aligned}
\langle \psi_{T,00}^0, \psi_{\tilde{T},00}^0 \rangle_{L^2(\Omega)} &= \delta_{[T=\tilde{T}]} \cdot |\det \nabla F_T| \cdot \langle \hat{\varphi}_{00}^0, \hat{\varphi}_{00}^0 \rangle_{L^2(\hat{T})}, \\
\langle \varphi_N^1, \varphi_{\tilde{N}}^1 \rangle_{H^1(\Omega)} &= \sum_{\substack{T \in \mathcal{T}_h: \\ N, \tilde{N} \in \mathcal{N}(T)}} |\det \nabla F_T| \cdot \left[\langle \hat{\varphi}_{ij(1,N,T)}^1, \hat{\varphi}_{ij(1,\tilde{N},T)}^1 \rangle_{L^2(\hat{T})} \right. \\
&\quad \left. + \int_{\hat{T}} \langle \nabla \hat{\varphi}_{ij(1,N,T)}^1 \cdot \nabla F_T^{-1}, \nabla \hat{\varphi}_{ij(1,\tilde{N},T)}^1 \cdot \nabla F_T^{-1} \rangle_2 \, dV \right], \\
a(\varphi_N^1, \varphi_{\tilde{N}}^1) &= \sum_{\substack{T \in \mathcal{T}_h: \\ N, \tilde{N} \in \mathcal{N}(T)}} |\det \nabla F_T| \cdot \int_{\hat{T}} \langle \nabla \hat{\varphi}_{ij(1,N,T)}^1 \cdot \nabla F_T^{-1}, \nabla \hat{\varphi}_{ij(1,\tilde{N},T)}^1 \cdot \nabla F_T^{-1} \rangle_2 \, dV, \\
b(\psi_{T,00}^0, \varphi_N^1) &= \delta_{[N \in \mathcal{N}(T)]} \cdot |\det \nabla F_T| \cdot \langle \hat{\varphi}_{00}^0, \hat{\varphi}_{ij(1,N,T)}^1 \rangle_{L^2(\hat{T})}.
\end{aligned}$$

und

$$\langle u_0, \varphi_N^1 \rangle_{H^1(\Omega)} = \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\det \nabla F_T| \cdot \int_{\hat{T}} \langle (\nabla u_0) \circ F_T, \nabla \hat{\varphi}_{ij(1,N,T)}^1 \cdot \nabla F_T^{-1} \rangle_2 + (u_0 \circ F_T) \cdot \hat{\varphi}_{ij(1,N,T)}^1 \, dV$$

sowie

$$\begin{aligned}
\langle I_{h,g(u_0)} u_0, \varphi_N^1 \rangle_{H^1(\Omega)} &= \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\det \nabla F_T| \cdot \sum_{(i,j) \in IJ_{g(u_0)}} (u_0 \circ F_T)(\hat{x}_i^{g(u_0)}, \hat{y}_j^{g(u_0)}) \\
&\quad \cdot \left[\langle \hat{\varphi}_{ij}^{g(u_0)}, \hat{\varphi}_{ij(1,N,T)}^1 \rangle_{L^2(\hat{T})} + \int_{\hat{T}} \langle \nabla \hat{\varphi}_{ij}^{g(u_0)} \cdot \nabla F_T^{-1}, \nabla \hat{\varphi}_{ij(1,N,T)}^1 \cdot \nabla F_T^{-1} \rangle_2 \, dV \right].
\end{aligned}$$

Beweis:

Analog zu Lemma 3.1.1.2.

Lemma 3.1.2.3 (Ein 2D Matlab Beispiel Fehlerberechnung).

Wir bezeichnen mit $(\gamma_{T,00})_{T \in \mathcal{T}_h} \subseteq \mathbb{R}$, $(\mu_N)_{N \in \mathcal{N}_{I,h}} \subseteq \mathbb{R}$ und $(\pi_N)_{N \in \mathcal{N}_{I,h}} \subseteq \mathbb{R}$ die Koeffizienten aus den Darstellungen

$$f_h = \sum_{T \in \mathcal{T}_h} \gamma_{T,00} \cdot \psi_{T,00}^0, \quad u_h = \sum_{N \in \mathcal{N}_{I,h}} \mu_N \cdot \varphi_N^1, \quad p_h = \sum_{N \in \mathcal{N}_{I,h}} \pi_N \cdot \varphi_N^1$$

aus Korollar 2.5.5.1.

Dann können die Fehler

$$\begin{aligned} \|f - f_h\|_{L^2(\Omega)}^2 &= \|f\|_{L^2(\Omega)}^2 - 2 \cdot \langle f, f_h \rangle_{L^2(\Omega)} + \|f_h\|_{L^2(\Omega)}^2, \\ \|u - u_h\|_{L^2(\Omega)}^2 &= \|u\|_{L^2(\Omega)}^2 - 2 \cdot \langle u, u_h \rangle_{L^2(\Omega)} + \|u_h\|_{L^2(\Omega)}^2, \\ \|u - u_h\|_{H^1(\Omega)}^2 &= \|u\|_{H^1(\Omega)}^2 - 2 \cdot \langle u, u_h \rangle_{H^1(\Omega)} + \|u_h\|_{H^1(\Omega)}^2, \\ \|p - p_h\|_{L^2(\Omega)}^2 &= \|p\|_{L^2(\Omega)}^2 - 2 \cdot \langle p, p_h \rangle_{L^2(\Omega)} + \|p_h\|_{L^2(\Omega)}^2, \\ \|p - p_h\|_{H^1(\Omega)}^2 &= \|p\|_{H^1(\Omega)}^2 - 2 \cdot \langle p, p_h \rangle_{H^1(\Omega)} + \|p_h\|_{H^1(\Omega)}^2 \end{aligned}$$

mit Hilfe der folgenden Formeln berechnet werden:

$$\begin{aligned} \|f_h\|_{L^2(\Omega)}^2 &= \frac{1 - \kappa}{\kappa} \cdot \gamma^T \cdot \mathbf{F} \cdot \gamma, \\ \|u_h\|_{L^2(\Omega)}^2 &= \mu^T \cdot (\mathbf{U} - \mathbf{A}) \cdot \mu, \\ \|u_h\|_{H^1(\Omega)}^2 &= \mu^T \cdot \mathbf{U} \cdot \mu, \\ \|p_h\|_{L^2(\Omega)}^2 &= \pi^T \cdot (\mathbf{U} - \mathbf{A}) \cdot \pi, \\ \|p_h\|_{H^1(\Omega)}^2 &= \pi^T \cdot \mathbf{U} \cdot \pi \end{aligned}$$

und

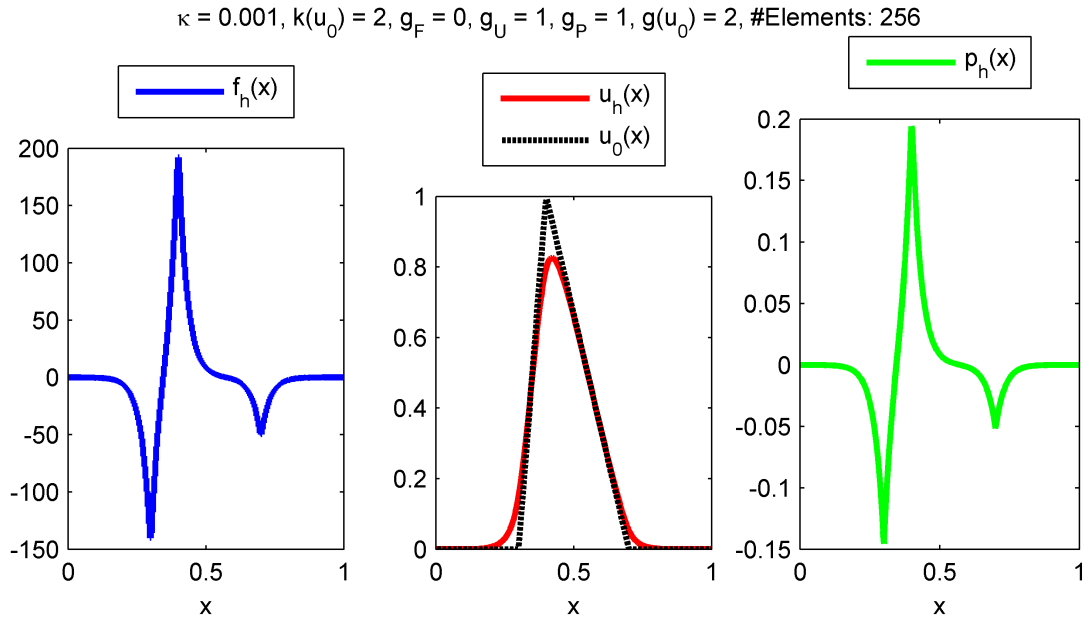
$$\begin{aligned} \langle f, f_h \rangle_{L^2(\Omega)} &= \sum_{T \in \mathcal{T}_h} \gamma_{T,00} \cdot |\det \nabla F_T| \cdot \langle f \circ F_T, \hat{\varphi}_{00}^0 \rangle_{L^2(\hat{T})}, \\ \langle u, u_h \rangle_{L^2(\Omega)} &= \sum_{N \in \mathcal{N}_{I,h}} \mu_N \cdot \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\det \nabla F_T| \cdot \langle u \circ F_T, \hat{\varphi}_{i(1,N,T)}^1 \rangle_{L^2(\hat{T})}, \\ \langle u, u_h \rangle_{H^1(\Omega)} &= \sum_{N \in \mathcal{N}_{I,h}} \mu_N \cdot \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\det \nabla F_T| \cdot \left[\langle u \circ F_T, \hat{\varphi}_{i(1,N,T)}^1 \rangle_{L^2(\hat{T})} \right. \\ &\quad \left. + \int_{\hat{T}} \langle (\nabla u) \circ F_T, \nabla \hat{\varphi}_{ij(1,N,T)}^1 \cdot \nabla F_T^{-1} \rangle_2 \, dV \right], \\ \langle p, p_h \rangle_{L^2(\Omega)} &= \sum_{N \in \mathcal{N}_{I,h}} \pi_N \cdot \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\det \nabla F_T| \cdot \langle p \circ F_T, \hat{\varphi}_{i(1,N,T)}^1 \rangle_{L^2(\hat{T})}, \\ \langle p, p_h \rangle_{H^1(\Omega)} &= \sum_{N \in \mathcal{N}_{I,h}} \pi_N \cdot \sum_{\substack{T \in \mathcal{T}_h: \\ N \in \mathcal{N}(T)}} |\det \nabla F_T| \cdot \left[\langle p \circ F_T, \hat{\varphi}_{i(1,N,T)}^1 \rangle_{L^2(\hat{T})} \right. \\ &\quad \left. + \int_{\hat{T}} \langle (\nabla p) \circ F_T, \nabla \hat{\varphi}_{ij(1,N,T)}^1 \cdot \nabla F_T^{-1} \rangle_2 \, dV \right]. \end{aligned}$$

Beweis:

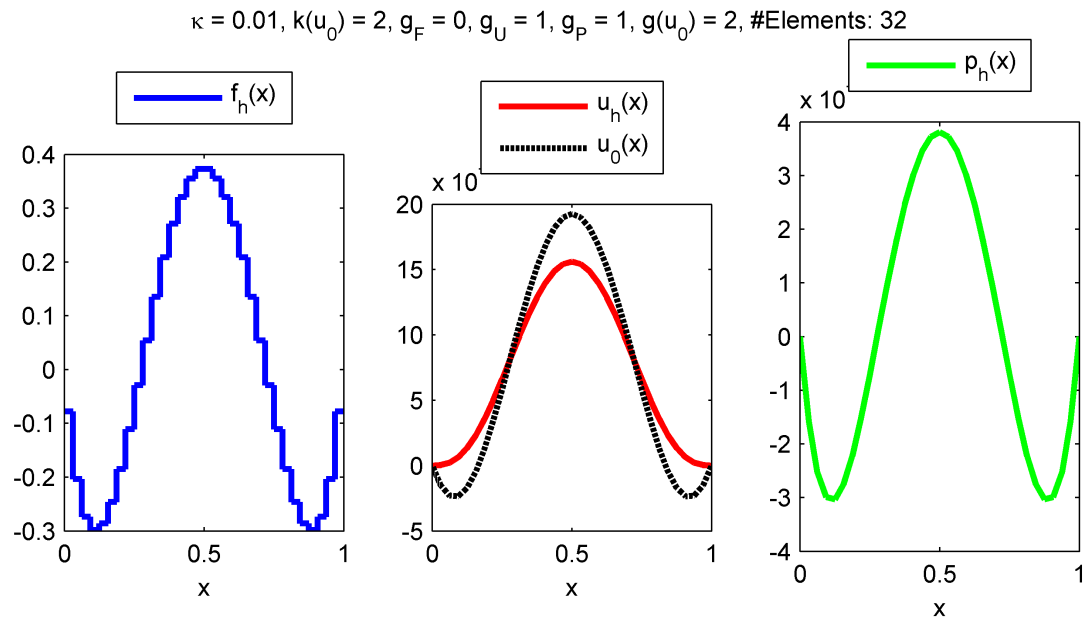
Analog zu Lemma 3.1.1.3.

3.2 Numerische Ergebnisse

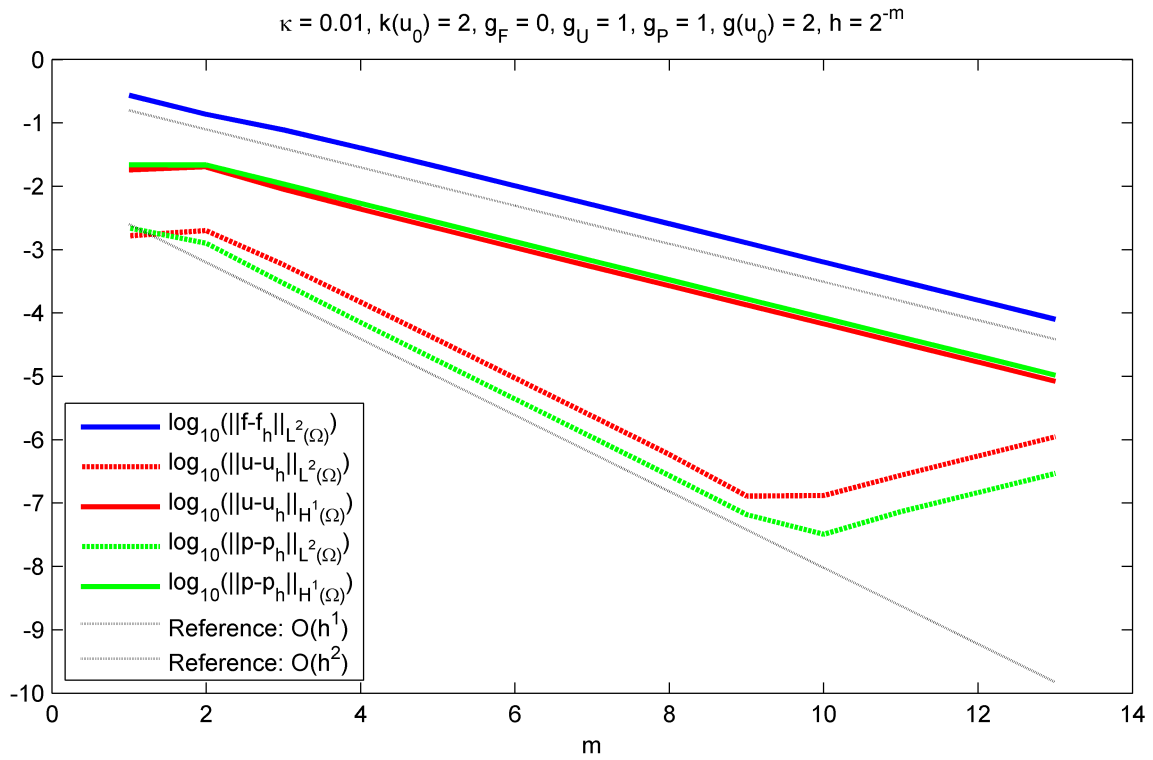
3.2.1 1D



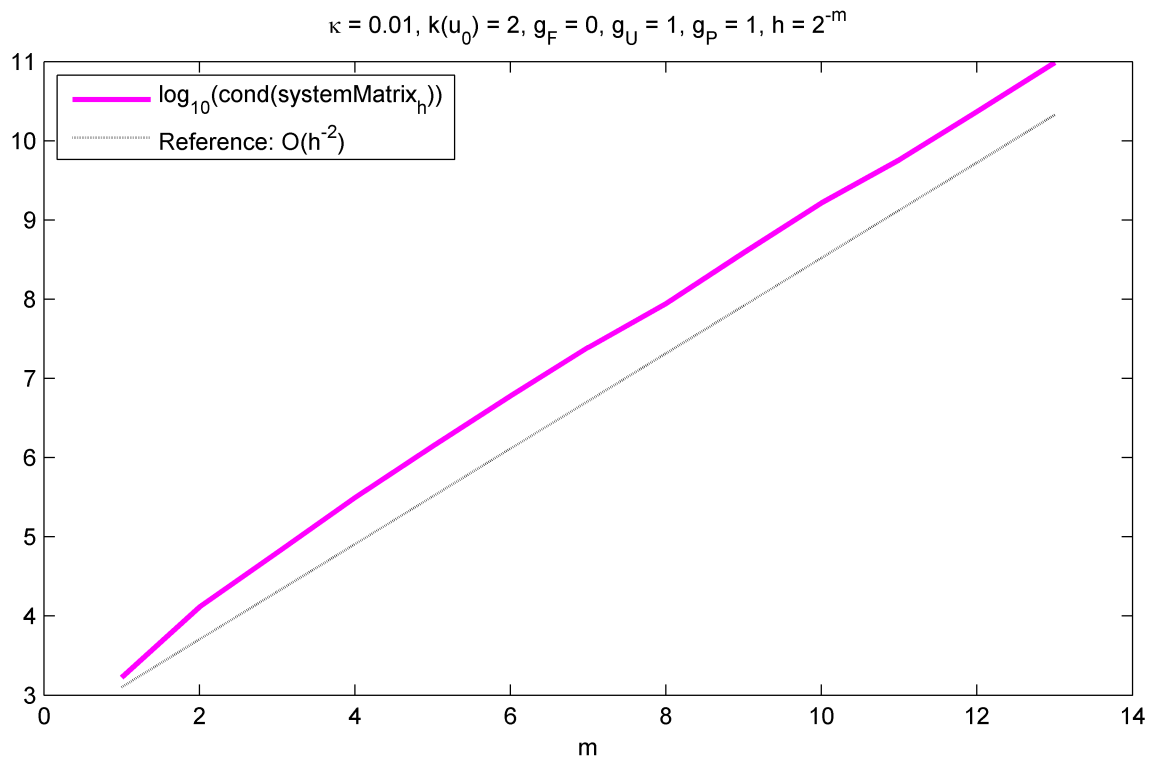
Plot der Lösungen. (Erzeugt durch `singleSolution.m` mittels Aufruf `singleSolution`.)



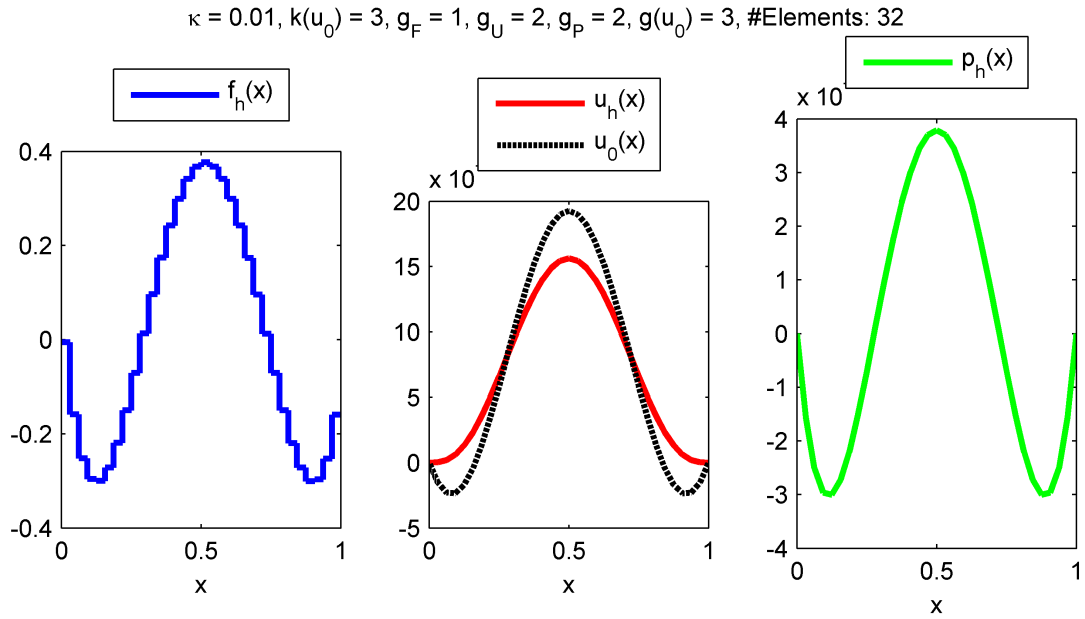
Plot der Lösungen für $g_F = 0$ und $g_U = 1$ bei $m = 5$ (i.e. bei $2^5 = 32$ Elementen). (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis(2)`.)



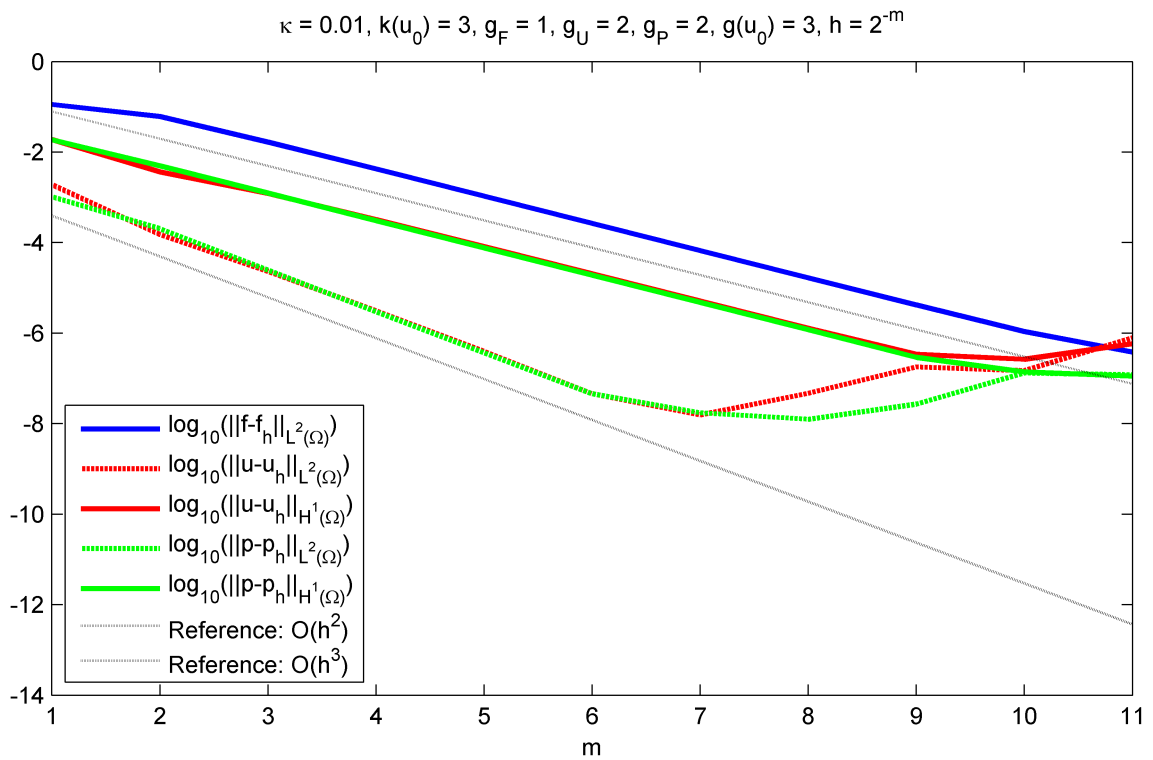
Plot der Fehler für $g_F = 0$ und $g_U = 1$. (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis(2).`)



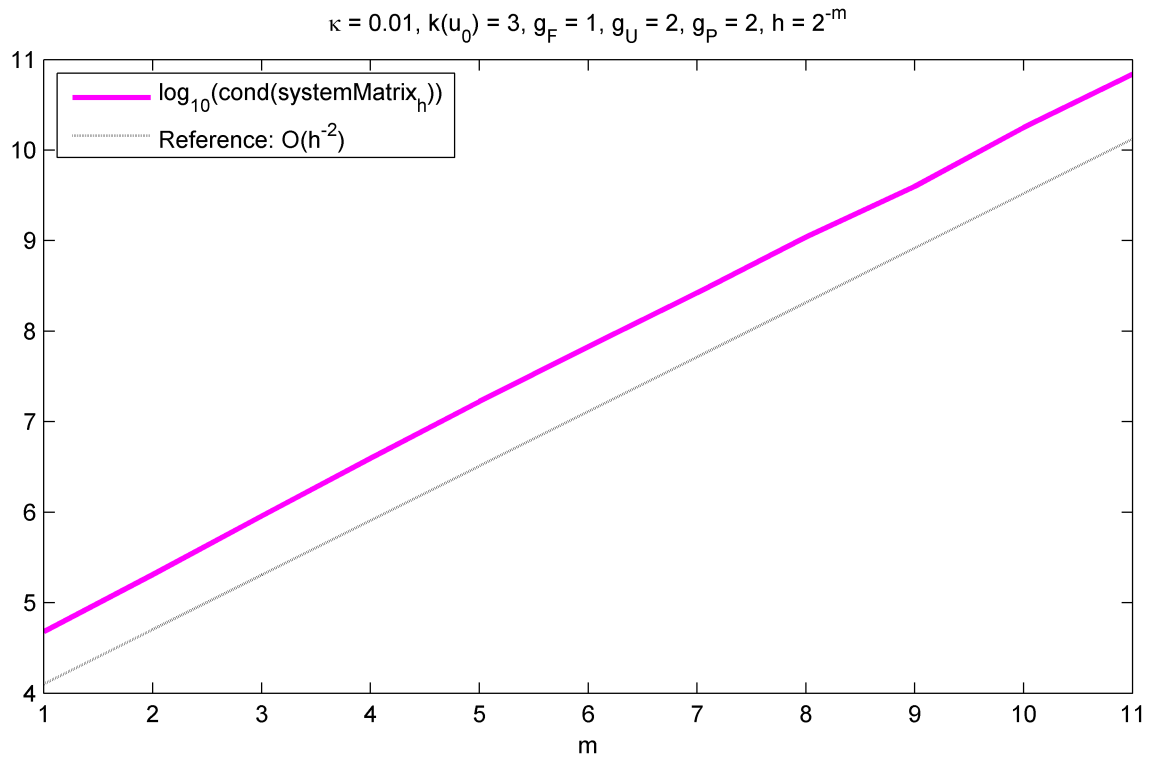
Plot der Konditionszahlen für $g_F = 0$ und $g_U = 1$. (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis(2).`)



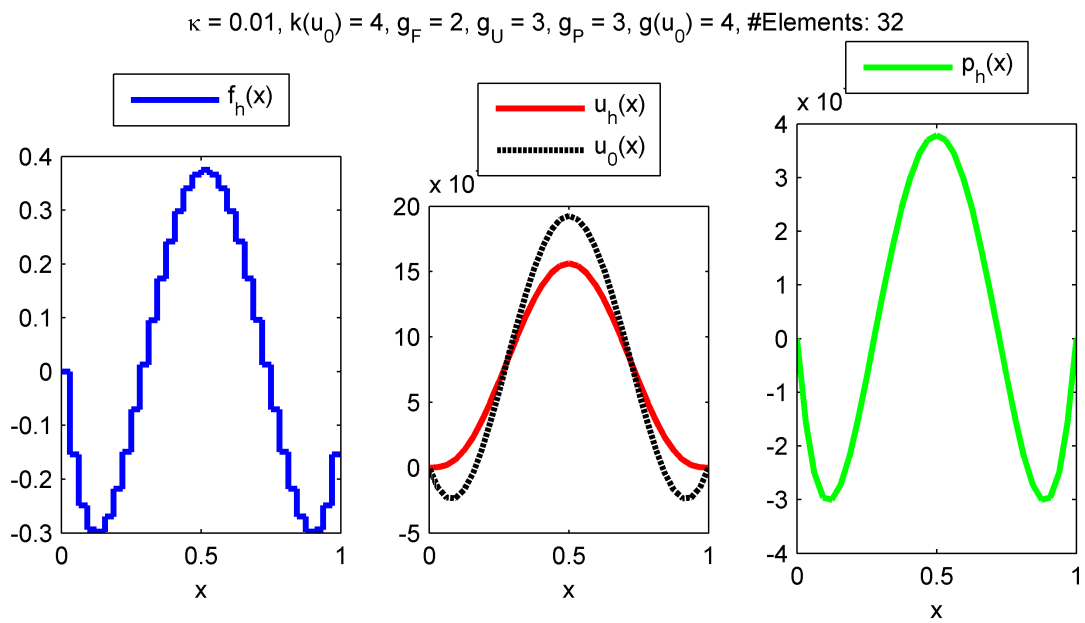
Plot der Lösungen für $g_F = 1$ und $g_U = 2$ bei $m = 5$ (i.e. bei $2^5 = 32$ Elementen). (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis(3).`)



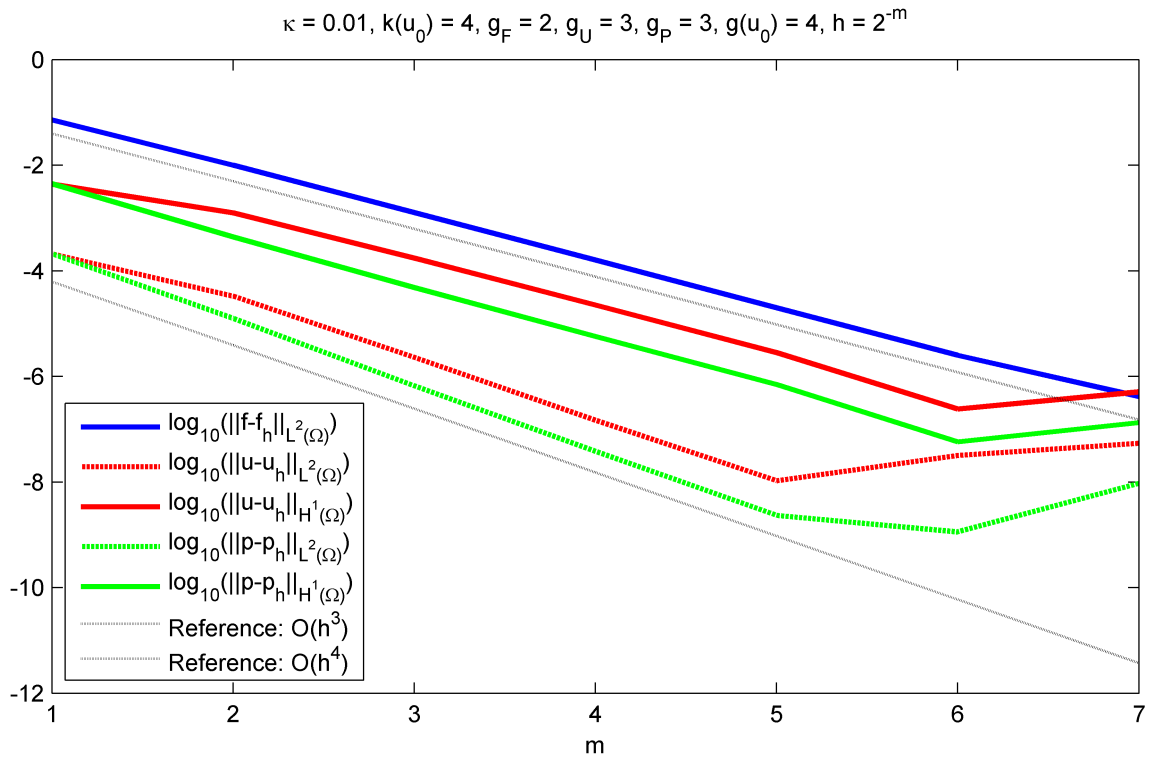
Plot der Fehler für $g_F = 1$ und $g_U = 2$. (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis(3).`)



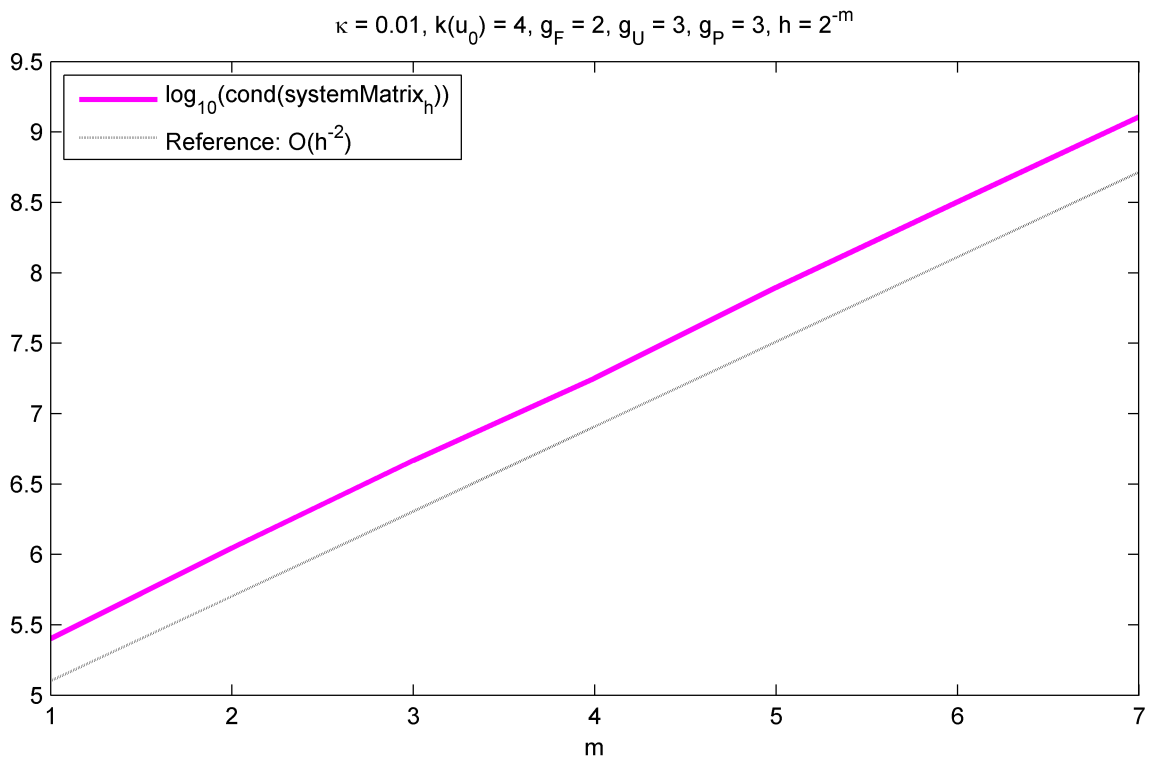
Plot der Konditionszahlen für $g_F = 1$ und $g_U = 2$. (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis(3).`)



Plot der Lösungen für $g_F = 2$ und $g_U = 3$ bei $m = 5$ (i.e. bei $2^5 = 32$ Elementen). (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis(4).`)

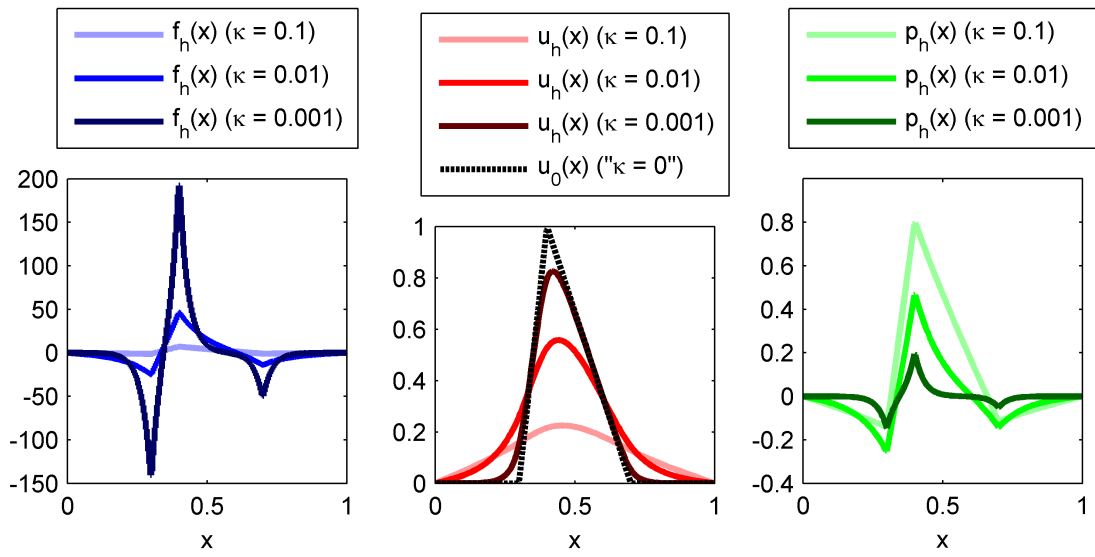


Plot der Fehler für $g_F = 2$ und $g_U = 3$. (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis(4)`.)



Plot der Konditionszahlen für $g_F = 2$ und $g_U = 3$. (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis(4)`.)

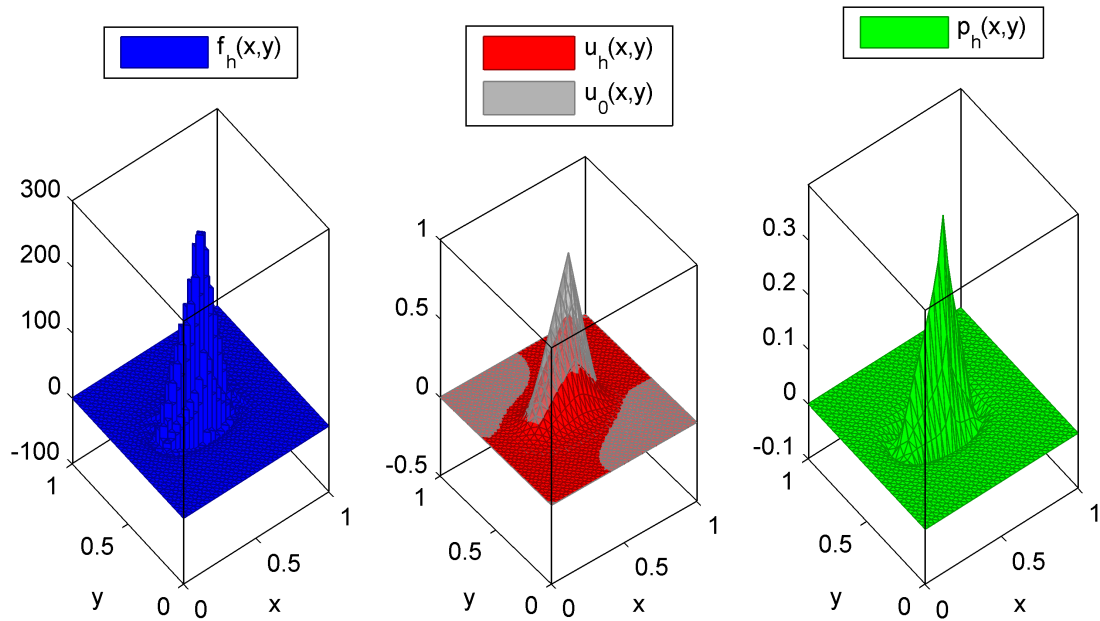
$k(u_0) = 2, g_F = 0, g_U = 1, g_P = 1, g(u_0) = 2, \#Elements: 256$



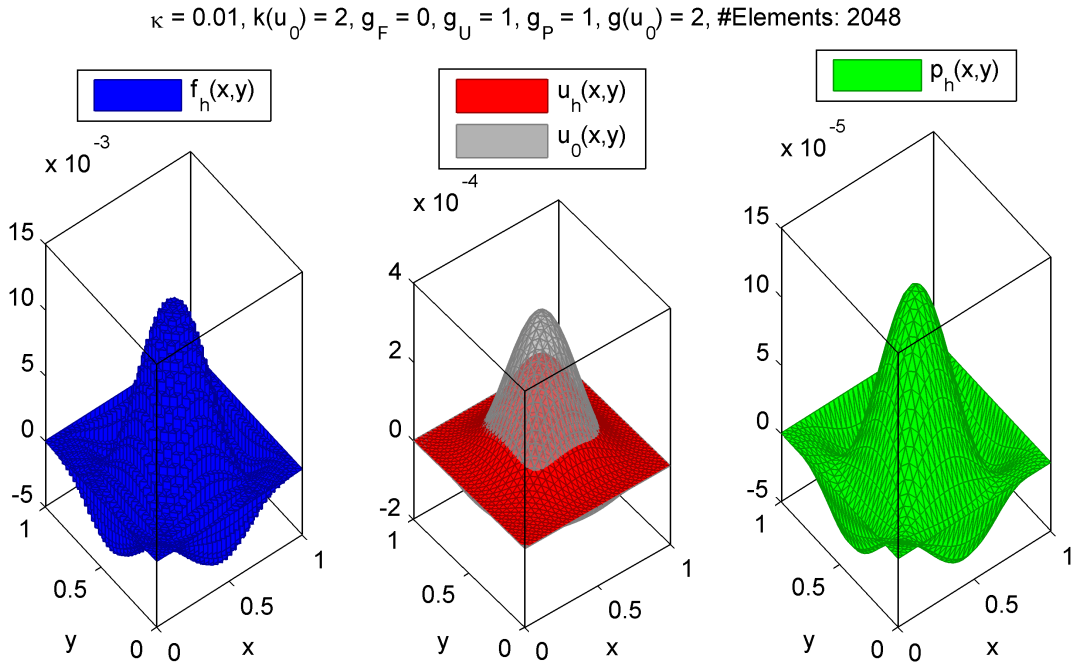
Plot der Lösungen für unterschiedliche Werte von κ . (Erzeugt durch `kappaComparison.m` mittels Aufruf `kappaComparison`.)

3.2.2 2D

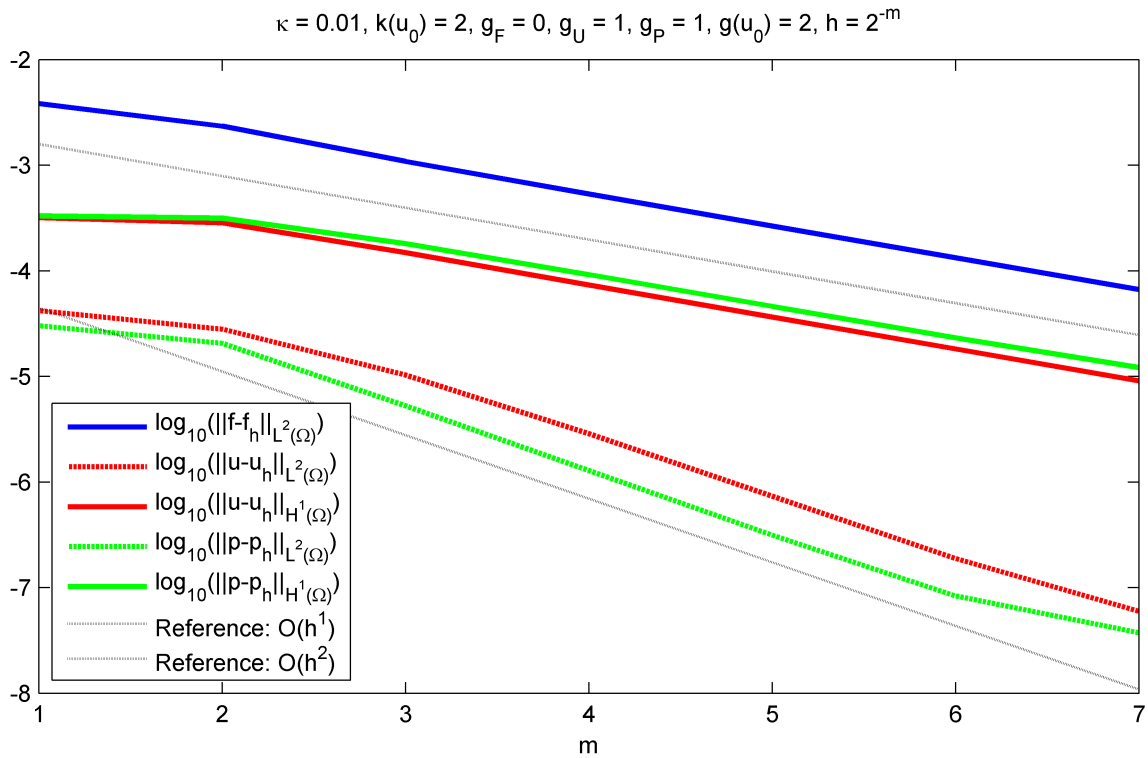
$\kappa = 0.001, k(u_0) = 2, g_F = 0, g_U = 1, g_P = 1, g(u_0) = 2, \#Elements: 2048$



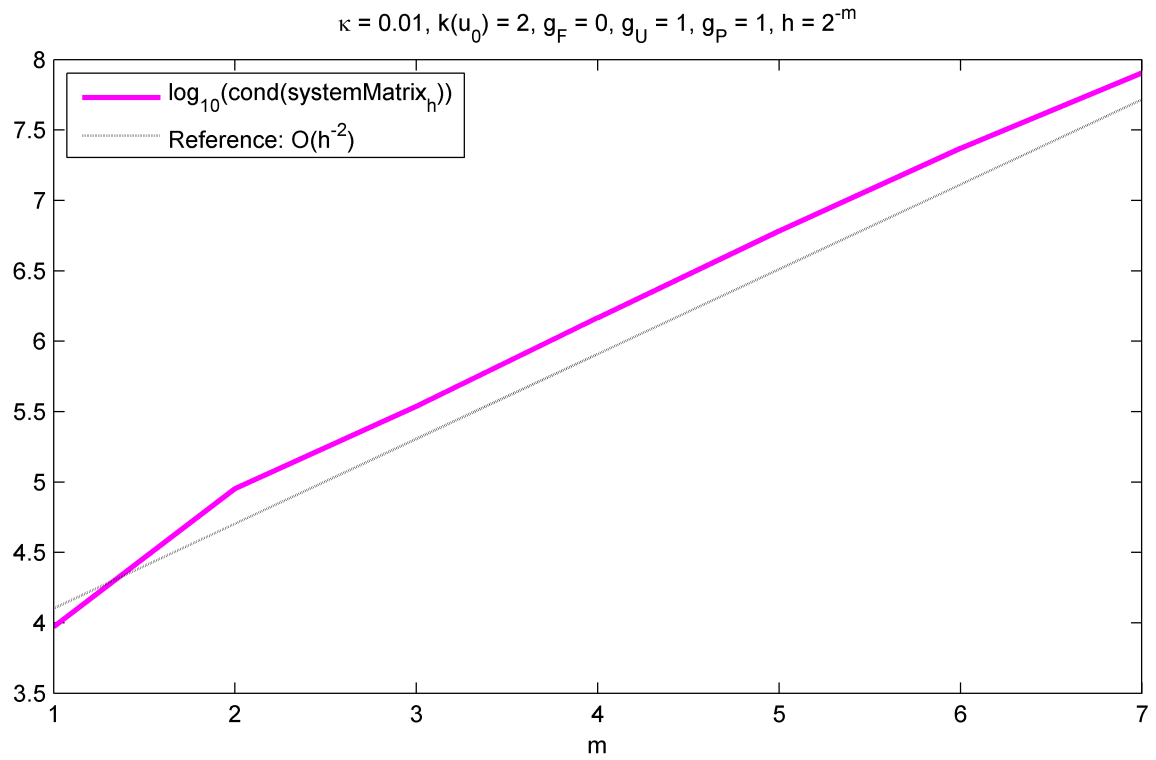
Plot der Lösungen. (Erzeugt durch `singleSolution.m` mittels Aufruf `singleSolution`.)



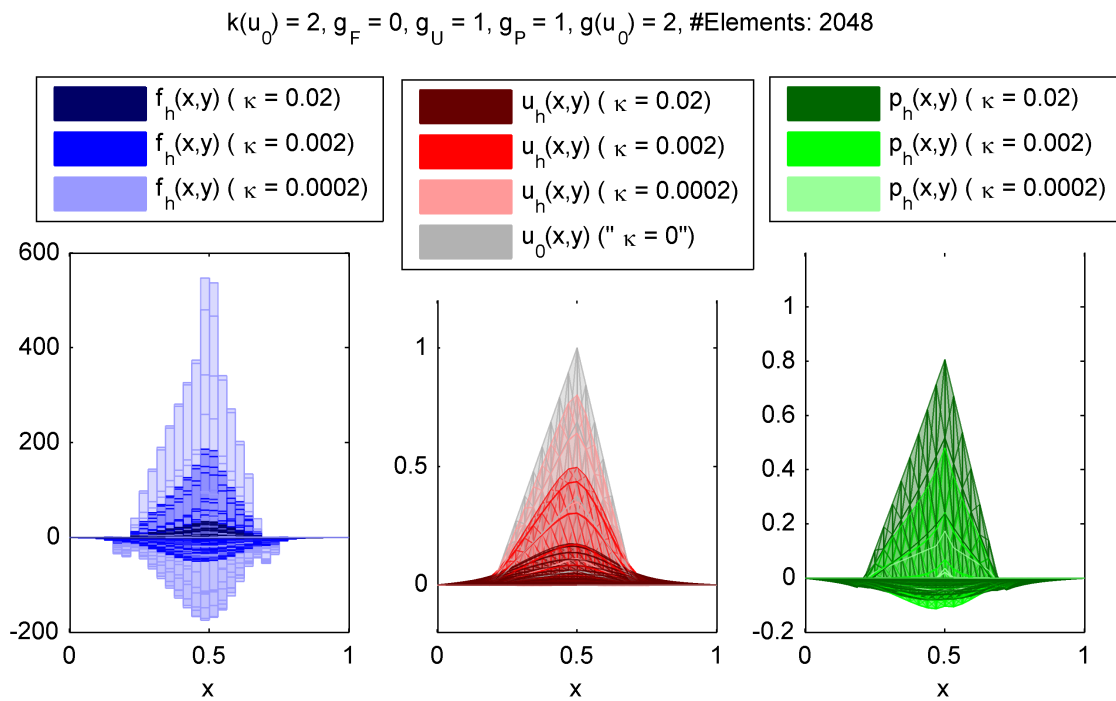
Plot der Lösungen für $g_F = 0$ und $g_U = 1$ bei $m = 5$ (i.e. bei $(2^5)^2 \cdot 2 = 2048$ Elementen). (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis`.)



Plot der Fehler für $g_F = 0$ und $g_U = 1$. (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis`.)



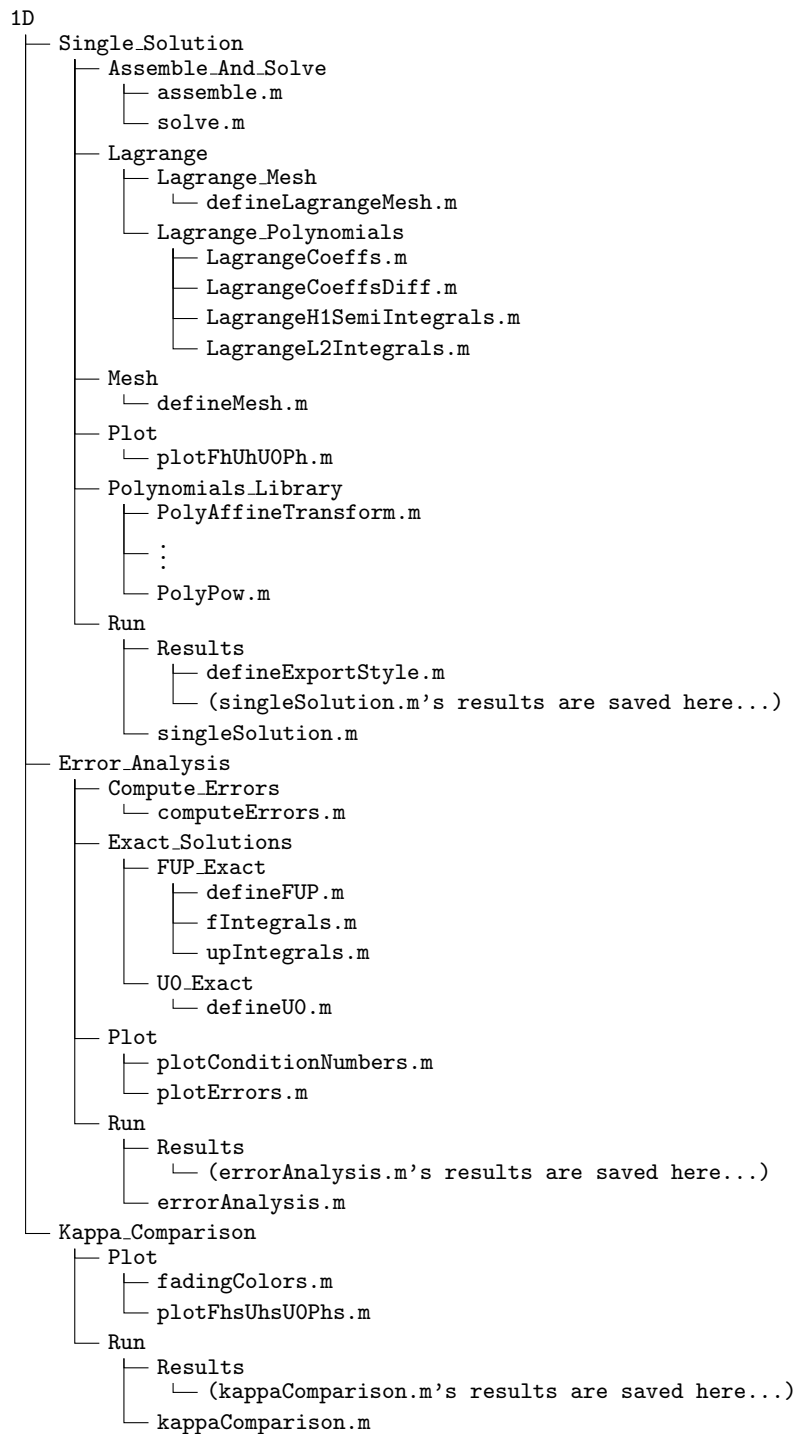
Plot der Konditionszahlen für $g_F = 0$ und $g_U = 1$. (Erzeugt durch `errorAnalysis.m` mittels Aufruf `errorAnalysis.`)



Plot der Lösungen für unterschiedliche Werte von κ in der Seitenansicht entlang der x -Achse. (Erzeugt durch `kappaComparison.m` mittels Aufruf `kappaComparison.`)

3.3 MATLAB Programm-Codes

3.3.1 1D



Ordnerstruktur 1D MATLAB Beispiel

Listing 1: assemble.m

```

1 function [F,U11,U12,U22,A11,A12,A22,B1,B2,U01,U02] = assemble(u0,kappa,gF,gU,gu0,...
    numberOfNodes)
2 % Assembles the components F,U11,U12,U22,A11,A12,A22,B1,B2 of the system
3 % matrix and the components U01,U02 of the load vector.
4
5 % Define Lagrange mesh on reference element:
6 [LagrangeNodes_gu0,I_gu0] = defineLagrangeMesh(gu0);
7
8 % Define main mesh on Omega:
9 [h,Nodes,...
10  FTiMat,FTiVect,...
11  I_gF,I_gU,I_gU_X,indicesElements,indicesNodes,indicesInnerNodes,...
12  idxTrafo_psiTr_iFromIndexTR,idxTrafo_psiTr_indexTRFromI,psiTr_dim,...
13  idxTrafo_phiN_iFromIndexN,idxTrafo_phiN_indexNFromI,phiN_dim,...
14  idxTrafo_phiTs_iFromIndexTS,idxTrafo_phiTs_indexTSFromI,phiTs_dim] = defineMesh(gF,...
    gU,numberOfNodes);
15
16 % Compute Lagrange polynomials integrals:
17 LagrangeL2Integrals_gF_gF = LagrangeL2Integrals(gF,gF);
18 LagrangeL2Integrals_gF_gU = LagrangeL2Integrals(gF,gU);
19 LagrangeL2Integrals_gU_gU = LagrangeL2Integrals(gU,gU);
20 LagrangeL2Integrals_gu0_gU = LagrangeL2Integrals(gu0,gU);
21 LagrangeH1SemiIntegrals_gU_gU = LagrangeH1SemiIntegrals(gU,gU);
22 LagrangeH1SemiIntegrals_gu0_gU = LagrangeH1SemiIntegrals(gu0,gU);
23
24 % Assemble the components of the system matrix:
25 % F:
26 F = sparse(psiTr_dim,psiTr_dim);
27 for indexT = indicesElements
28     for r = I_gF
29         for rTilde = I_gF(1):1:r % Exploit symmetry.
30             valueF = (kappa)/(1-kappa)*abs(FTiMat(indexT))*LagrangeL2Integrals_gF_gF(...
                rTilde+1,r+1);
31
32             i = idxTrafo_psiTr_iFromIndexTR(indexT,r);
33             l = idxTrafo_psiTr_iFromIndexTR(indexT,rTilde);
34             F(i,l) = valueF;
35             F(l,i) = valueF;
36         end
37     end
38 end
39
40 % U11,U12,U22,A11,A12,A22:
41 U11 = sparse(phiN_dim,phiN_dim);
42 A11 = sparse(phiN_dim,phiN_dim);
43 for indexN = indicesInnerNodes
44     for indexNTilde = [indexN:1:min(indexN+1,indicesInnerNodes(end))]
45         valueU = 0;
46         valueA = 0;
47         for indexT = [max(indexN,indexNTilde)-1:1:min(indexN,indexNTilde)] % ...
            Exploit symmetry.
48             iN = gU*(indexN~=indexT);
49             iTilde = gU*(indexNTilde~=indexT);
50             valueH1Semi = (1/abs(FTiMat(indexT)))*LagrangeH1SemiIntegrals_gU_gU(iNTilde...
                +1,iN+1);
51             valueL2 = abs(FTiMat(indexT))*LagrangeL2Integrals_gU_gU(iNTilde+1,iN+1);
52
53             valueU = valueU + valueH1Semi + valueL2;
54             valueA = valueA + valueH1Semi;
55         end
56
57         j = idxTrafo_phiN_iFromIndexN(indexN);
58         m = idxTrafo_phiN_iFromIndexN(indexNTilde);
59         U11(j,m) = valueU;
60         U11(m,j) = valueU;
61         A11(j,m) = valueA;
62         A11(m,j) = valueA;
63     end
64 end

```

```

65
66 U12 = sparse(phiN.dim,phiTs.dim);
67 A12 = sparse(phiN.dim,phiTs.dim);
68 for indexN = indicesInnerNodes
69     for indexT = [indexN-1,indexN]
70         for s = I_gU_X
71             iN = gU*(indexN~=indexT);
72
73             j = idxTrafo_phiN.iFromIndexN(indexN);
74             m = idxTrafo_phiTs.iFromIndexTS(indexT,s);
75             valueU = (1/abs(FTiMat(indexT)))*LagrangeH1SemiIntegrals_gU_gU(s+1,iN+1);
76             valueA = abs(FTiMat(indexT))*LagrangeL2Integrals_gU_gU(s+1,iN+1);
77
78             U12(j,m) = valueU + valueA;
79             A12(j,m) = valueU;
80         end
81     end
82 end
83
84 U22 = sparse(phiTs.dim,phiTs.dim);
85 A22 = sparse(phiTs.dim,phiTs.dim);
86 for indexT = indicesElements
87     for s = I_gU_X
88         for sTilde = I_gU_X(1):1:s % Exploit symmetry.
89             valueH1Semi = (1/abs(FTiMat(indexT)))*LagrangeH1SemiIntegrals_gU_gU(sTilde...
90                 +1,s+1);
91             valueL2 = abs(FTiMat(indexT))*LagrangeL2Integrals_gU_gU(sTilde+1,s+1);
92
93             j = idxTrafo_phiTs.iFromIndexTS(indexT,s);
94             m = idxTrafo_phiTs.iFromIndexTS(indexT,sTilde);
95             U22(j,m) = valueH1Semi + valueL2;
96             U22(m,j) = valueH1Semi + valueL2;
97             A22(j,m) = valueH1Semi;
98             A22(m,j) = valueH1Semi;
99         end
100     end
101 end
102 % B1,B2:
103 B1 = sparse(phiN.dim,psiTr.dim);
104 for indexN = indicesInnerNodes
105     for indexT = [indexN-1,indexN]
106         for r = I_gF
107             iN = gU*(indexN~=indexT);
108
109             k = idxTrafo_phiN.iFromIndexN(indexN);
110             l = idxTrafo_psiTr.iFromIndexTR(indexT,r);
111             B1(k,l) = -abs(FTiMat(indexT))*LagrangeL2Integrals_gF_gU(r+1,iN+1);
112         end
113     end
114 end
115
116 B2 = sparse(phiTs.dim,psiTr.dim);
117 for indexT = indicesElements
118     for s = I_gU_X
119         for r = I_gF
120             k = idxTrafo_phiTs.iFromIndexTS(indexT,s);
121             l = idxTrafo_psiTr.iFromIndexTR(indexT,r);
122             B2(k,l) = -abs(FTiMat(indexT))*LagrangeL2Integrals_gF_gU(r+1,s+1);
123         end
124     end
125 end
126
127 % Assemble the load vector:
128 % U01,U02:
129 U01 = sparse(phiN.dim,1);
130 for indexN = indicesInnerNodes
131     valueU01 = 0;
132     for indexT = [indexN-1,indexN]
133         for i = I_gu0

```

```

134         iN = gU*(indexN~=indexT);
135         valueU01 = valueU01 + abs(FTiMat(indexT))*u0(FTiMat(indexT)*...
            LagrangeNodes_gu0(i+1)+FTiVect(indexT))...
136         *( LagrangeL2Integrals_gu0_gU(i+1,iN+1)...
137         +(1/FTiMat(indexT)^2)*LagrangeH1SemiIntegrals_gu0_gU(i+1,iN+1) );
138     end
139 end
140
141     j = idxTrafo_phiN_iFromIndexN(indexN);
142     U01(j) = valueU01;
143 end
144
145     U02 = sparse(phiTs_dim,1);
146     for indexT = indicesElements
147         for s = I_gU_X
148             valueU02 = 0;
149             for i = I_gu0
150                 valueU02 = valueU02 + abs(FTiMat(indexT))*u0(FTiMat(indexT)*...
                    LagrangeNodes_gu0(i+1)+FTiVect(indexT))...
151                 *( LagrangeL2Integrals_gu0_gU(i+1,s+1)...
152                 +(1/FTiMat(indexT)^2)*LagrangeH1SemiIntegrals_gu0_gU(i+1,s+1) );
153             end
154
155             j = idxTrafo_phiTs_iFromIndexTS(indexT,s);
156             U02(j) = valueU02;
157         end
158     end
159
160 end

```

Listing 2: solve.m

```

1 function [fh_psiTr,uh_phiN,uh_phiTs,ph_phiN,ph_phiTs,conditionNumber] = solve(F,U11,U12...
    ,U22,A11,A12,A22,B1,B2,U01,U02)
2 % Assembles the system matrix from its components F,U11,U12,U22,A11,A12,A22,B1,B2
3 % and solves the resulting LSE.
4
5 % Final assemble of the system matrix:
6 psiTr_dim = size(F,1);
7 [phiN_dim,phiTs_dim] = size(U12);
8
9 systemMatrix = [F,sparse(psiTr_dim,phiN_dim+phiTs_dim),B1',B2';...
10     sparse(phiN_dim+phiTs_dim,psiTr_dim),[U11,U12;U12',U22],[A11',A12';A12',A22'];...
11     [B1;B2],[A11,A12;A12',A22],sparse(phiN_dim+phiTs_dim,phiN_dim+phiTs_dim)];
12 conditionNumber = condest(systemMatrix);
13
14 % Final assemble of the load vector:
15 loadVector = [sparse(psiTr_dim,1);U01;U02;sparse(phiN_dim+phiTs_dim,1)];
16
17 % Solve the LSE:
18 solution = full(systemMatrix\loadVector)';
19
20 % Extract the coefficients of the approximate solutions fh,uh,ph from the
21 % solution vector:
22 fh_psiTr = solution(1:1:psiTr_dim);
23 uh_phiN = solution(psiTr_dim+[1:1:phiN_dim]);
24 uh_phiTs = solution(psiTr_dim+phiN_dim+[1:1:phiTs_dim]);
25 ph_phiN = solution(psiTr_dim+phiN_dim+phiTs_dim+[1:1:phiN_dim]);
26 ph_phiTs = solution(psiTr_dim+2*phiN_dim+phiTs_dim+[1:1:phiTs_dim]);
27
28 end

```

Listing 3: defineLagrangeMesh.m

```

1 function [LagrangeNodes,I_g] = defineLagrangeMesh(g)
2 % Defines all necessary data necessary for representing the Lagrange mesh
3 % on the reference element:

```

```

4 % 1) The actual coordinates of all Lagrange nodes (LagrangeNodes).
5 % 2) An index array containing the indices of the Lagrange nodes (I_g).
6
7 % Define geometry data:
8 LagrangeNodes = [0:1:g]/g;
9
10 % Define index arrays:
11 I_g = [0:1:g];
12
13 end

```

Listing 4: LagrangeCoeffs.m

```

1 function LagrangeCoeffsMatrix = LagrangeCoeffs(g)
2 % Computes the coefficients of all Lagrange polynomials of degree g
3 % on the reference element. The vector LagrangeCoeffsMatrix(i,:)
4 % contains all coefficients of the i'th Lagrange polynomial.
5
6 LagrangeCoeffsMatrix = (fliplr(vander([0:1:g]/g))\eye(g+1))';
7
8 end

```

Listing 5: LagrangeCoeffsDiff.m

```

1 function LagrangeCoeffsDiffMatrix = LagrangeCoeffsDiff(g)
2 % Computes the derivatives of all Lagrange polynomials of degree g on the
3 % reference element. The vector LagrangeCoeffsDiffMatrix(i,:) contains all
4 % coefficients of the derivative of the i'th Lagrange polynomial.
5
6 % Compute Lagrange polynomials coefficients:
7 LagrangeCoeffsMatrix = LagrangeCoeffs(g);
8
9 % Compute derivatives:
10 LagrangeCoeffsDiffMatrix = zeros(size(LagrangeCoeffsMatrix,1),max(size(...
    LagrangeCoeffsMatrix,2)-1,1));
11 for i = 1:1:size(LagrangeCoeffsMatrix,1)
12     LagrangeCoeffsDiffMatrix(i,:) = PolyDiff(LagrangeCoeffsMatrix(i,:));
13 end
14
15 end

```

Listing 6: LagrangeH1SemiIntegrals.m

```

1 function LagrangeH1SemiIntegrals = LagrangeH1SemiIntegrals(g1,g2)
2 % Computes the integrals int(p'(x)*q'(x),x=0..1) for all
3 % pairs (p,q) of Lagrange polynomials.
4
5 % Compute Lagrange polynomials coefficients:
6 LagrangeCoeffsDiffMatrix_g1 = LagrangeCoeffsDiff(g1);
7 LagrangeCoeffsDiffMatrix_g2 = LagrangeCoeffsDiff(g2);
8
9 % Compute integrals:
10 LagrangeH1SemiIntegrals = zeros(g1+1,g2+1);
11 for i = 1:1:g1+1
12     for j = 1:1:g2+1
13         LagrangeH1SemiIntegrals(i,j) = PolyL2IntTRef(LagrangeCoeffsDiffMatrix_g1(i,:),...
            LagrangeCoeffsDiffMatrix_g2(j,:));
14     end
15 end
16
17 end

```

Listing 7: LagrangeL2Integrals.m

```

1 function LagrangeL2Integrals = LagrangeL2Integrals(g1,g2)
2 % Computes the integrals  $\int p(x)q(x), x=0..1$  for all
3 % pairs (p,q) of Lagrange polynomials.
4
5 % Compute Lagrange polynomials coefficients:
6 LagrangeCoeffsMatrix_g1 = LagrangeCoeffs(g1);
7 LagrangeCoeffsMatrix_g2 = LagrangeCoeffs(g2);
8
9 % Compute integrals:
10 LagrangeL2Integrals = zeros(g1+1,g2+1);
11 for i = 1:1:g1+1
12     for j = 1:1:g2+1
13         LagrangeL2Integrals(i,j) = PolyL2IntTRef(LagrangeCoeffsMatrix_g1(i,:),...
14             LagrangeCoeffsMatrix_g2(j,:));
15     end
16 end
17 end

```

Listing 8: defineMesh.m

```

1 function [h,Nodes,...
2     FTiMat,FTiVect,...
3     I_gF,I_gU,I_gU_X,indicesElements,indicesAllNodes,indicesInnerNodes,...
4     idxTrafo_psiTr_iFromIndexTR,idxTrafo_psiTr_indexTRFromI,psiTr_dim,...
5     idxTrafo_phiN_iFromIndexN,idxTrafo_phiN_indexNFromI,phiN_dim,...
6     idxTrafo_phiTs_iFromIndexTS,idxTrafo_phiTs_indexTSFromI,phiTs_dim] = defineMesh(gF,...
7     gU,numberOfNodes)
8 % Defines all necessary data necessary for representing the mesh on Omega:
9 % 1) The actual coordinates of all nodes (Nodes).
10 % 2) Matrix-vector-representations of the affine transformations between the reference
11 % element and each element in the mesh (FTiMat,FTiVect).
12 % 3) Index arrays containing the indices of elements and all/inner nodes
13 % (I_gF,...,indicesInnerNodes).
14 % 4) Index transformations used to transform e.g. (indexT,r)-indices into
15 % single indices needed e.g. in the assembly of the system matrix
16 % (idxTrafo_psiTr,...,idxTrafo_phiTs....).
17 % 5) The dimensions of the Ansatz-spaces (psiTr_dim,...,phiTs_dim).
18
19 % Define geometry data:
20 h = 1/(numberOfNodes-1);
21 Nodes = h*(1:1:numberOfNodes-1);
22
23 % Define affine transformations:
24 FTiMat = @(i) h;
25 FTiVect = @(i) Nodes(i);
26
27 % Define index arrays:
28 I_gF = [0:1:gF];
29 I_gU = [0:1:gU];
30 I_gU_X = [1:1:gU-1];
31 indicesElements = [1:1:numberOfNodes-1];
32 indicesAllNodes = [1:1:numberOfNodes];
33 indicesInnerNodes = [2:1:numberOfNodes-1];
34
35 % Define index transformations:
36
37 % Index transformations for  $\psi_{T,r}^{g_F}$ :
38 % (indexT,r) in  $\{1,\dots,n-1\} \times \{0,\dots,g_F\}$ 
39 %  $\leftrightarrow$ 
40 % (i) in  $\{1,\dots,(n-1)*(g_F+1)\}$ 
41 %
42 %  $i = (g_F+1)*(indexT-1)+r+1$ 
43 %  $\leftrightarrow$ 
44 %  $indexT = 1/(g_F+1)*(i-r-1)+1$ 
45 %  $r = \text{mod}(i-1,g_F+1)$ 
46 idxTrafo_psiTr_iFromIndexTR = @(indexT,r) (gF+1)*(indexT-1)+r+1;

```

```

46 idxTrafo_psiTr_indexTRFromI = @(i) deal(round(1/(gF+1)*(i-mod(i-1,gF+1)-1)+1),mod(i-1,...
    gF+1));
47 psiTr_dim = (numberOfNodes-1)*(gF+1);
48
49 % Index transformations for phi-{N}^{gU}:
50 % (indexN) in {2,...,n-1}
51 % <->
52 % (i) in {1,...,n-2}
53 %
54 % i = indexN-1
55 % <->
56 % indexN = i+1
57 idxTrafo_phiN_iFromIndexN = @(indexN) indexN-1;
58 idxTrafo_phiN_indexNFromI = @(i) i+1;
59 phiN_dim = numberOfNodes-2;
60
61 % Index transformations for phi-{T,s}^{gU}:
62 % (indexT,s) in {1,...,n-1} x {1,...,gU-1}
63 % <->
64 % (i) in {1,...,(n-1)*(gU-1)}
65 %
66 % i = (gU-1)*(indexT-1)+s
67 % <->
68 % indexT = 1/(gU-1)*(i-s)+1
69 % s = mod(i-1,gU-1)+1
70 idxTrafo_phiTs_iFromIndexTS = @(indexT,s) (gU-1)*(indexT-1)+s;
71 idxTrafo_phiTs_indexTSFromI = @(i) deal(round(1/(gU-1)*(i-mod(i-1,gU-1)-1)+1),mod(i-1,...
    gU-1)+1);
72 phiTs_dim = (numberOfNodes-1)*(gU-1);
73
74 end

```

Listing 9: plotFhUhUOPh.m

```

1 function figureHandle = plotFhUhUOPh(u0,kappa,ku0,gF,gU,gu0,numberOfNodes,fh.psiTr,...
    uh.phiN,ph.phiN)
2 % Sets up a figure with three subplots containing the plots of fh,uh,u0 and
3 % ph.
4 %
5 % Note: Only the nodal values of the solutions are used for plotting.
6
7 % Define main mesh on Omega:
8 [h,Nodes,...
9     FTiMat,FTiVect,...
10     I_gF,I_gU,I_gU_X,indicesElements,indicesNodes,indicesInnerNodes,...
11     idxTrafo_psiTr_iFromIndexTR,idxTrafo_psiTr_indexTRFromI,psiTr_dim,...
12     idxTrafo_phiN_iFromIndexN,idxTrafo_phiN_indexNFromI,phiN_dim,...
13     idxTrafo_phiTs_iFromIndexTS,idxTrafo_phiTs_indexTSFromI,phiTs_dim] = defineMesh(gF,...
    gu0,numberOfNodes);
14
15 % Define plot specifiers:
16 colorF = 'blue';
17 colorU = 'red';
18 colorP = 'green';
19 linestyleFUP = '-';
20 linewidthFUP = 2;
21 colorU0 = 'black';
22 linestyleU0 = '--';
23 linewidthU0 = 2;
24
25 % Plot:
26 figureHandle = figure;
27
28 % Subplot 1:
29 subplot(1,3,1);
30 stairs(Nodes,[fh.psiTr(1:gF+1:end),fh.psiTr(end-gF)],'color',colorF,'linestyle',...
    linestyleFUP,'linewidth',linewidthFUP);
31 xlabel('x');
32 legend('f_h(x)', 'Location', 'northoutside');

```

```

33 box on;
34
35 % Subplot 2:
36 subplot(1,3,2);
37 hold on;
38 plot(Nodes,[0,u_h_phiN,0], 'color',colorU, 'linestyle',linestyleFUP, 'linewidth',...
    linewidthFUP);
39 plot(Nodes,u0(Nodes), 'color',colorU0, 'linestyle',linestyleU0, 'linewidth',linewidthU0);
40 xlabel('x');
41 legend('u_h(x)', 'u_0(x)', 'Location', 'northoutside');
42 box on;
43
44 % Subplot 3:
45 subplot(1,3,3);
46 plot(Nodes,[0,ph_phiN,0], 'color',colorP, 'linestyle',linestyleFUP, 'linewidth',...
    linewidthFUP);
47 xlabel('x');
48 legend('p_h(x)', 'Location', 'northoutside');
49 box on;
50
51 % Title:
52 titleString = ['\kappa = ', num2str(kappa), ...
53     ', k(u_0) = ', num2str(ku0), ...
54     ', g_F = ', num2str(gF), ...
55     ', g_U = ', num2str(gU), ...
56     ', g_P = ', num2str(gU), ...
57     ', g(u_0) = ', num2str(gu0), ...
58     ', #Elements: ', num2str(numberOfNodes-1)];
59 annotation('textbox', [0,0.94,1,0.06], ...
60     'String',titleString, ...
61     'EdgeColor','none', ...
62     'HorizontalAlignment','center', ...
63     'VerticalAlignment','bottom');
64
65 end

```

Listing 10: PolyAffineTransform.m

```

1 function pTransformed = PolyAffineTransform(p,a,b,PascalMatrix,PowersMatrix)
2 % Performs affine transformation of the polynomial p. The transformation is
3 % of the form  $F(x) = a \cdot x + b$ . The resulting polynomial is pTransformed(x) =
4 % p(F(x)). The given matrices PascalMatrix and PowersMatrix depend on the
5 % degree of p only and are computed in advance. They allow for much faster
6 % computation.
7
8 pTransformed = (diag(a.^[0:1:(length(p)-1)]) * ((b.^PowersMatrix) .* PascalMatrix) * p)';
9
10 end

```

Listing 11: PolyAffineTransformMatrices.m

```

1 function [PascalMatrix,PowersMatrix] = PolyAffineTransformMatrices(g)
2 % Computes the matrices PascalMatrix and PowersMatrix for later use in
3 % 'PolyAffineTransform.m'.
4
5 PascalMatrix = zeros(g+1,g+1);
6 for i = 0:1:g
7     for j = i:1:g
8         PascalMatrix(i+1,j+1) = nchoosek(j,i);
9     end
10 end
11
12 PowersMatrix = zeros(g+1,g+1);
13 for i = 0:1:g
14     for j = i:1:g
15         PowersMatrix(i+1,j+1) = j-i;
16     end

```

```

17 end
18
19 end

```

Listing 12: PolyDiff.m

```

1 function pDiff = PolyDiff(p)
2 % Differentiates the polynomial p. Uses MATLAB builtin function 'polyder'.
3
4 pDiff = fliplr(polyder(fliplr(p)));
5
6 end

```

Listing 13: PolyEval.m

```

1 function value = PolyEval(p,x)
2 % Evaluates the polynomial p at x. Uses MATLAB builtin function 'polyval'.
3
4 value = polyval(p(end:-1:1),x);
5
6 end

```

Listing 14: PolyL1IntOmega.m

```

1 function integral = PolyL1IntOmega(p)
2 % Computes the integral int(p(x),x=0..1) for a given polynomial p.
3
4 integral = p*(1./[1:1:size(p,2)])';
5
6 end

```

Listing 15: PolyL1IntTRef.m

```

1 function integral = PolyL1IntTRef(p)
2 % Computes the integral int(p(x),x=0..1) for a given polynomial p.
3
4 integral = p*(1./[1:1:size(p,2)])';
5
6 end

```

Listing 16: PolyL2IntOmega.m

```

1 function integral = PolyL2IntOmega(p,q)
2 % Computes the integral int(p(x)*q(x),x=0..1) for given polynomials p and q.
3
4 integral = PolyL1IntOmega(PolyMult(p,q));
5
6 end

```

Listing 17: PolyL2IntTRef.m

```

1 function integral = PolyL2IntTRef(p,q)
2 % Computes the integral int(p(x)*q(x),x=0..1) for given polynomials p and q.
3
4 integral = PolyL1IntTRef(PolyMult(p,q));
5
6 end

```

Listing 18: PolyMult.m

```

1 function product = PolyMult(p,q)
2 % Computes the product p*q of two polynomials.
3
4 product = conv(p,q);
5
6 end

```

Listing 19: PolyPow.m

```

1 function pExp = PolyPow(p,exponent)
2 % Computes the polynomial p^exponent for given polynomial p and given exponent.
3
4 pExp = 1;
5 for i = 1:1:exponent
6     pExp = PolyMult(pExp,p);
7 end
8
9 end

```

Listing 20: defineExportStyle.m

```

1 function exportStyle = defineExportStyle(format,width,height,resolution)
2 % Defines the parameters used for saving (exporting) the result plots.
3
4 exportStyle = hgexport('factorystyle');
5 exportStyle.Format = format;
6 exportStyle.Units = 'centimeters';
7 exportStyle.Width = width;
8 exportStyle.Height = height;
9 exportStyle.Resolution = resolution;
10 exportStyle.Bounds = 'tight';
11
12 end

```

Listing 21: singleSolution.m

```

1 function singleSolution
2 % Computes approximate solutions fh,uh,ph to optimal control problem (Laplace equation)...
3 % and
4 % plots them.
5 %
6 % Inputs: None.
7 %
8 % Outputs: A plot showing the approximate solutions fh,uh,ph and the 'target' u0.
9
10 % Equation parameters:
11 kappa = 1/1000;
12 a = 0.3;
13 b = 0.4;
14 c = 0.7;
15 u0 = @(x) ((a<=x)&(x<=b)).*(x-a)/(b-a) + ((b<x)&(x<=c)).*((b-x)/(c-b)+1);
16
17 % Discretization parameters:
18 ku0 = 2;
19 gF = ku0-2;
20 gU = ku0-1;
21 gu0 = gU+1;
22 numberOfNodes = 2^8+1;
23
24 % Assemble system matrix (F,U11,U12,U22,A11,A12,A22,B1,B2) and load vector (U01,U02):
25 [F,U11,U12,U22,A11,A12,A22,B1,B2,U01,U02] = assemble(u0,kappa,gF,gU,gu0,numberOfNodes);
26
27 % Solve the LSE:

```

```

27 [fh_psiTr,uh_phiN,~,ph_phiN,~,~] = solve(F,U11,U12,U22,A11,A12,A22,B1,B2,U01,U02);
28
29 % Plot the solutions:
30 figureHandle = plotFhUhU0Ph(u0,kappa,ku0,gF,gU,gu0,numberOfNodes,fh_psiTr,uh_phiN,...
    ph_phiN);
31
32 % Save results:
33 hgexport(figureHandle,[fileparts(which('singleSolution.m')),'\Results\solutionsPlot'],...
    defineExportStyle('png',15,7,600));
34
35 end

```

Listing 22: computeErrors.m

```

1 function errors = computeErrors(kappa,f,u,p,...
2     gF,gU,numberOfNodes,...
3     fh_psiTr,uh_phiN,uh_phiTs,ph_phiN,ph_phiTs,...
4     F,U11,U12,U22,A11,A12,A22)
5 % Computes the L2- and H1-errors between approximate and exact solutions by
6 % means of the following decompositions:
7 % 1) ||f-fh||_L2^2 = ||f||_L2^2 - 2*<f,fh>_L2 + ||fh||_L2^2
8 % 2) ||u-uh||_L2^2 = ||u||_L2^2 - 2*<u,uh>_L2 + ||uh||_L2^2
9 % 3) ||u-uh||_H1^2 = ||u||_H1^2 - 2*<u,uh>_H1 + ||uh||_H1^2
10 % 4) ||p-ph||_L2^2 = ||p||_L2^2 - 2*<p,ph>_L2 + ||ph||_L2^2
11 % 5) ||p-ph||_H1^2 = ||p||_H1^2 - 2*<p,ph>_H1 + ||ph||_H1^2
12 %
13 % Note: There hold ||fh||_L2^2 = (1-kappa)/kappa*fh_psiTr*F*fh_psiTr' and analogous
14 % formulas for ||uh||_L2^2,...,||ph||_H1^2.
15
16 % Define main mesh on Omega:
17 [h,Nodes,...
18     FTiMat,FTiVect,...
19     I_gF,I_gU,I_gU_X,indicesElements,indicesNodes,indicesInnerNodes,...
20     idxTrafo_psiTr.iFromIndexTR,idxTrafo_psiTr.indexTRFromI,psiTr_dim,...
21     idxTrafo_phiN.iFromIndexN,idxTrafo_phiN.indexNFromI,phiN_dim,...
22     idxTrafo_phiTs.iFromIndexTS,idxTrafo_phiTs.indexTSFromI,phiTs_dim] = defineMesh(gF,...
    gU,numberOfNodes);
23
24 % Compute derivatives of u and p:
25 uDiff = PolyDiff(u);
26 pDiff = PolyDiff(p);
27
28 % Compute scalar products:
29 % <f,fh>_L2:
30 L2IntegralsF = fIntegrals(f,gF,numberOfNodes);
31
32 L2SP_f_fh = 0;
33 for indexT = indicesElements
34     for r = I_gF
35         i = idxTrafo_psiTr.iFromIndexTR(indexT,r);
36         L2SP_f_fh = L2SP_f_fh + fh_psiTr(i)*abs(FTiMat(indexT))*L2IntegralsF(indexT,r...
            +1);
37     end
38 end
39
40 % <u,uh>_L2, <u,uh>_H1, <p,ph>_L2, <p,ph>_H1:
41 [L2IntegralsU_phiN,H1SemiIntegralsU_phiN,L2IntegralsU_phiTs,H1SemiIntegralsU_phiTs] = ...
    upIntegrals(u,gU,numberOfNodes);
42 [L2IntegralsP_phiN,H1SemiIntegralsP_phiN,L2IntegralsP_phiTs,H1SemiIntegralsP_phiTs] = ...
    upIntegrals(p,gU,numberOfNodes);
43
44 L2SP_u_uh = 0;
45 H1SP_u_uh = 0;
46 L2SP_p_ph = 0;
47 H1SP_p_ph = 0;
48 for indexN = indicesInnerNodes
49     i = idxTrafo_phiN.iFromIndexN(indexN);
50     for indexT = [indexN-1,indexN]
51         iN = gU*(indexN~=indexT);

```

```

52
53     valueL2_u = uh_phiN(i)*abs(FTiMat(indexT))*L2IntegralsU_phiN(indexT,iN+1);
54     valueH1Semi_u = uh_phiN(i)*sign(FTiMat(indexT))*H1SemiIntegralsU_phiN(indexT,iN...
55         +1);
56     valueL2_p = ph_phiN(i)*abs(FTiMat(indexT))*L2IntegralsP_phiN(indexT,iN+1);
57     valueH1Semi_p = ph_phiN(i)*sign(FTiMat(indexT))*H1SemiIntegralsP_phiN(indexT,iN...
58         +1);
59
60     L2SP_u_uh = L2SP_u_uh + valueL2_u;
61     H1SP_u_uh = H1SP_u_uh + valueL2_u + valueH1Semi_u;
62     L2SP_p_ph = L2SP_p_ph + valueL2_p;
63     H1SP_p_ph = H1SP_p_ph + valueL2_p + valueH1Semi_p;
64
65 end
66 for indexT = indicesElements
67     for s = I_gU_X
68         i = idxTrafo_phiTs.iFromIndexTS(indexT,s);
69
70         valueL2_u = uh_phiTs(i)*abs(FTiMat(indexT))*L2IntegralsU_phiTs(indexT,s);
71         valueH1Semi_u = uh_phiTs(i)*sign(FTiMat(indexT))*H1SemiIntegralsU_phiTs(indexT,...
72             s);
73         valueL2_p = ph_phiTs(i)*abs(FTiMat(indexT))*L2IntegralsP_phiTs(indexT,s);
74         valueH1Semi_p = ph_phiTs(i)*sign(FTiMat(indexT))*H1SemiIntegralsP_phiTs(indexT,...
75             s);
76
77         L2SP_u_uh = L2SP_u_uh + valueL2_u;
78         H1SP_u_uh = H1SP_u_uh + valueL2_u + valueH1Semi_u;
79         L2SP_p_ph = L2SP_p_ph + valueL2_p;
80         H1SP_p_ph = H1SP_p_ph + valueL2_p + valueH1Semi_p;
81     end
82 end
83
84 % Compute squared norms of exact solutions:
85 % ||f||_L2^2, ||u||_L2^2, ||u||_H1^2, ||p||_L2^2, ||p||_H1^2:
86 L2NormSquared_f = PolyL2IntOmega(f,f);
87 L2NormSquared_u = PolyL2IntOmega(u,u);
88 H1NormSquared_u = PolyL2IntOmega(uDiff,uDiff) + L2NormSquared_u;
89 L2NormSquared_p = PolyL2IntOmega(p,p);
90 H1NormSquared_p = PolyL2IntOmega(pDiff,pDiff) + L2NormSquared_p;
91
92 % Compute squared norms of approximate solutions:
93 % ||fh||_L2^2, ||uh||_L2^2, ||uh||_H1^2, ||ph||_L2^2, ||ph||_H1^2:
94 L2NormSquared_fh = (1-kappa)/kappa*fh.psiTr*F*fh.psiTr';
95
96 H1NormSquared_uh = uh_phiN*(U11*uh_phiN')...
97     + 2*uh_phiN*(U12*uh_phiTs')...
98     + uh_phiTs*(U22*uh_phiTs');
99 L2NormSquared_uh = H1NormSquared_uh - uh_phiN*(A11*uh_phiN')...
100     - 2*uh_phiN*(A12*uh_phiTs')...
101     - uh_phiTs*(A22*uh_phiTs');
102
103 H1NormSquared_ph = ph_phiN*(U11*ph_phiN')...
104     + 2*ph_phiN*(U12*ph_phiTs')...
105     + ph_phiTs*(U22*ph_phiTs');
106 L2NormSquared_ph = H1NormSquared_ph - ph_phiN*(A11*ph_phiN')...
107     - 2*ph_phiN*(A12*ph_phiTs')...
108     - ph_phiTs*(A22*ph_phiTs');
109
110 % Compute total errors:
111 errors = [sqrt(abs(L2NormSquared_f-2*L2SP_f_fh+L2NormSquared_fh)),...
112     sqrt(abs(L2NormSquared_u-2*L2SP_u_uh+L2NormSquared_uh)),...
113     sqrt(abs(H1NormSquared_u-2*H1SP_u_uh+H1NormSquared_uh)),...
114     sqrt(abs(L2NormSquared_p-2*L2SP_p_ph+L2NormSquared_ph)),...
115     sqrt(abs(H1NormSquared_p-2*H1SP_p_ph+H1NormSquared_ph))];
116
117 end

```

Listing 23: defineFUP.m

```

1 function [f,u,p] = defineFUP(kappa)
2 % Computes the exact polynomial solutions f,u,p for given kappa. The
3 % 'target' u0 will be chosen accordingly afterwards (see
4 % 'defineU0.m').
5 %
6 % X(x) := x*(1-x)
7 % u(x) := X(x)^3 in H^1_0(Omega)
8 % f(x) := -u''(x) in H^1_0(Omega)
9 % p(x) := kappa/(1-kappa)*f(x) in H^1_0(Omega)
10
11 X = [0,1,-1];
12 u = PolyPow(X,3);
13 f = -PolyDiff(PolyDiff(u));
14 p = kappa/(1-kappa)*f;
15
16 end

```

Listing 24: fIntegrals.m

```

1 function L2Integrals = fIntegrals(f,gF,numberofNodes)
2 % Computes the integrals int(f(FTi(x))*p(x),x=0..1) for the
3 % exact solution f, all affine transformations FTi and all Lagrange
4 % polynomials p. These values will be needed to compute the error
5 % ||f-fh||_L2.
6
7 % Define main mesh on Omega:
8 [h,Nodes,...
9   FTiMat,FTiVect,...
10  I_gF,I_gU,I_gU_X,indicesElements,indicesNodes,indicesInnerNodes,...
11  idxTrafo.psiTr_iFromIndexTR,idxTrafo.psiTr.indexTRFromI,psiTr_dim,...
12  idxTrafo.phiN_iFromIndexN,idxTrafo.phiN.indexNFromI,phiN_dim,...
13  idxTrafo.phiTs_iFromIndexTS,idxTrafo.phiTs.indexTSFromI,phiTs_dim] = defineMesh(gF...
14  ,1,numberofNodes);
15
16 % Compute Lagrange polynomials coefficients:
17 LagrangeCoeffsMatrix = LagrangeCoeffs(gF);
18
19 % Compute pascal and powers matrices:
20 [PascalMatrix,PowersMatrix] = PolyAffineTransformMatrices(length(f)-1);
21
22 % Compute integrals:
23 L2Integrals = zeros(length(indicesElements),length(I_gF));
24 for indexT = indicesElements
25     fAffineTransformed = PolyAffineTransform(f,FTiMat(indexT),FTiVect(indexT),...
26     PascalMatrix,PowersMatrix);
27     for r = I_gF
28         L2Integrals(indexT,r+1) = PolyL2IntTRef(fAffineTransformed,LagrangeCoeffsMatrix...
29         (r+1,:));
30     end
31 end
32 end
33 end

```

Listing 25: upIntegrals.m

```

1 function [L2Integrals_phiN,H1SemiIntegrals_phiN,L2Integrals_phiTs,H1SemiIntegrals_phiTs...
2   ] = upIntegrals(u,gU,numberofNodes)
3 % Computes the integrals int(u(FTi(x))*q(x),x=0..1) and
4 % int(u'(FTi(x))*q(x),x=0..1) for the exact solution u (p resp.),
5 % all affine transformations FTi and all Lagrange polynomials q.
6 % These values will be needed to compute the errors
7 % ||u-uh||_L2 and ||u-uh||_H1 (||p-ph||_L2 and ||p-ph||_H1 resp.).
8
9 % Define main mesh on Omega:
10 [h,Nodes,...

```

```

10     FTiMat,FTiVect,...
11     I_gF,I_gU,I_gU_X,indicesElements,indicesNodes,indicesInnerNodes,...
12     idxTrafo.psiTr_iFromIndexTR,idxTrafo.psiTr_indexTRFromI,psiTr_dim,...
13     idxTrafo.phiN_iFromIndexN,idxTrafo.phiN_indexNFromI,phiN_dim,...
14     idxTrafo.phiTs_iFromIndexTS,idxTrafo.phiTs_indexTSFromI,phiTs_dim] = defineMesh(0,...
        gU,numberOfNodes);
15
16 % Compute derivatives of u (p resp.):
17 uDiff = PolyDiff(u);
18
19 % Compute Lagrange polynomials coefficients:
20 LagrangeCoeffsMatrix = LagrangeCoeffs(gU);
21 LagrangeCoeffsDiffMatrix = LagrangeCoeffsDiff(gU);
22
23 % Compute pascal and powers matrices:
24 [PascalMatrix_u,PowersMatrix_u] = PolyAffineTransformMatrices(length(u)-1);
25 [PascalMatrix_uDiff,PowersMatrix_uDiff] = PolyAffineTransformMatrices(length(uDiff)-1);
26
27 % Compute integrals:
28 L2Integrals_phiN = zeros(length(indicesElements),length(I_gU));
29 H1SemiIntegrals_phiN = zeros(length(indicesElements),length(I_gU));
30 L2Integrals_phiTs = zeros(length(indicesElements),length(I_gU_X));
31 H1SemiIntegrals_phiTs = zeros(length(indicesElements),length(I_gU_X));
32 for indexT = indicesElements
33     uAffineTransformed = PolyAffineTransform(u,FTiMat(indexT),FTiVect(indexT),...
        PascalMatrix_u,PowersMatrix_u);
34     uDiffAffineTransformed = PolyAffineTransform(uDiff,FTiMat(indexT),FTiVect(indexT),...
        PascalMatrix_uDiff,PowersMatrix_uDiff);
35
36     for iN = I_gU
37         L2Integrals_phiN(indexT,iN+1) = PolyL2IntTRef(uAffineTransformed,...
            LagrangeCoeffsMatrix(iN+1,:));
38         H1SemiIntegrals_phiN(indexT,iN+1) = PolyL2IntTRef(uDiffAffineTransformed,...
            LagrangeCoeffsDiffMatrix(iN+1,:));
39     end
40
41     for s = I_gU_X
42         L2Integrals_phiTs(indexT,s) = PolyL2IntTRef(uAffineTransformed,...
            LagrangeCoeffsMatrix(s+1,:));
43         H1SemiIntegrals_phiTs(indexT,s) = PolyL2IntTRef(uDiffAffineTransformed,...
            LagrangeCoeffsDiffMatrix(s+1,:));
44     end
45 end
46
47 end

```

Listing 26: defineU0.m

```

1 function u0 = defineU0(kappa,u)
2 % Defines the 'target' function u0.
3 %
4 % Note:
5 % The polynomials f,u,p from 'defineFUP.m' are the exact solutions of the optimal ...
    control
6 % problem (Laplace equation) with data u0.
7
8 u0 = @(x) PolyEval(u,x) + kappa/(1-kappa)*( 792*((exp(x)+exp(1-x))/(1+exp(1))-1) + 360*...
    PolyEval([0,1,-1],x) );
9
10 end

```

Listing 27: plotConditionNumbers.m

```

1 function figureHandle = plotConditionNumbers(kappa,ku0,gF,gU,mMin,mMax,conditionNumbers...
    )
2 % Plots the condition numbers of the system matrices for all values of h.
3

```

```

4 % Define plot specifiers:
5 colorConditionNumbers = 'magenta';
6 linestyleConditionNumbers = '-';
7 linewidthConditionNumbers = 2;
8 colorReferenceLine = 'black';
9 linestyleReferenceLine = ':';
10 linewidthReferenceLine = 1;
11
12 % Plot:
13 figureHandle = figure;
14 hold on;
15 plot(mMin:1:mMax, log10(conditionNumbers), 'color', colorConditionNumbers, 'linestyle', ...
    linestyleConditionNumbers, 'linewidth', linewidthConditionNumbers);
16 switch ku0
17     case 2
18         referenceLineGradient = 2;
19         referenceLineOffset = 2.5;
20     case 3
21         referenceLineGradient = 2;
22         referenceLineOffset = 3.5;
23     case 4
24         referenceLineGradient = 2;
25         referenceLineOffset = 4.5;
26 end
27 plot(mMin:1:mMax, log10(2.^([mMin:1:mMax]*referenceLineGradient))+referenceLineOffset, '...
    color', colorReferenceLine, 'linestyle', linestyleReferenceLine, 'linewidth', ...
    linewidthReferenceLine);
28
29 % Set xlabel, legend, title:
30 xlabel('m');
31 legend('log_{10}(cond(systemMatrix.h))', ['Reference: 0(h^{', num2str(-...
    referenceLineGradient), '})'], 'Location', 'northwest');
32 box on;
33 title(['\kappa = ', num2str(kappa), ...
    ', k(u_0) = ', num2str(ku0), ...
    ', g_F = ', num2str(gF), ...
    ', g_U = ', num2str(gU), ...
    ', g_P = ', num2str(gU), ...
    ', h = 2^{-m}']);
39
40 end

```

Listing 28: plotErrors.m

```

1 function figureHandle = plotErrors(kappa, ku0, gF, gU, gu0, mMin, mMax, errors)
2 % Plots the L2- and H1-errors between approximate and exact solutions for
3 % all values of h.
4
5 % Define plot specifiers:
6 colorF = 'blue';
7 colorU = 'red';
8 colorP = 'green';
9 linestyleFUPWeak = '--';
10 linestyleFUPStrong = '-';
11 linewidthFUPWeak = 2;
12 linewidthFUPStrong = 2;
13 colorReferenceLines = 'black';
14 linestyleReferenceLines = ':';
15 linewidthReferenceLines = 1;
16
17 % Plot:
18 figureHandle = figure;
19 hold on;
20 plot(mMin:1:mMax, log10(errors(:,1)), 'color', colorF, 'linestyle', linestyleFUPStrong, '...
    linewidth', linewidthFUPStrong);
21 plot(mMin:1:mMax, log10(errors(:,2)), 'color', colorU, 'linestyle', linestyleFUPWeak, '...
    linewidth', linewidthFUPWeak);
22 plot(mMin:1:mMax, log10(errors(:,3)), 'color', colorU, 'linestyle', linestyleFUPStrong, '...
    linewidth', linewidthFUPStrong);

```

```

23 plot(mMin:1:mMax,log10(errors(:,4)), 'color',colorP,'linestyle',linestyleFUPWeak, '...
    linewidth',linewidthFUPWeak);
24 plot(mMin:1:mMax,log10(errors(:,5)), 'color',colorP,'linestyle',linestyleFUPStrong, '...
    linewidth',linewidthFUPStrong);
25 switch ku0
26     case 2
27         referenceLinesGradients = [-1,-2];
28         referenceLinesOffsets = [-0.5,-2];
29     case 3
30         referenceLinesGradients = [-2,-3];
31         referenceLinesOffsets = [-0.5,-2.5];
32     case 4
33         referenceLinesGradients = [-3,-4];
34         referenceLinesOffsets = [-0.5,-3];
35 end
36 for i = 1:length(referenceLinesGradients)
37     plot(mMin:1:mMax,log10(2.^([mMin:1:mMax]*referenceLinesGradients(i))+...
        referenceLinesOffsets(i), 'color',colorReferenceLines,'linestyle',...
        linestyleReferenceLines,'linewidth',linewidthReferenceLines);
38 end
39
40 % Set xlabel, legend, title:
41 xlabel('m');
42
43 legendStrings = cell(5+length(referenceLinesGradients),1);
44 legendStrings(1:5) = {'log-10(||f-f_h|| -{L^2(\Omega)})',...
45     'log-10(||u-u_h|| -{L^2(\Omega)})',...
46     'log-10(||u-u_h|| -{H^1(\Omega)})',...
47     'log-10(||p-p_h|| -{L^2(\Omega)})',...
48     'log-10(||p-p_h|| -{H^1(\Omega)})'};
49 for i = 1:length(referenceLinesGradients)
50     legendStrings{i+5} = ['Reference: 0(h^{',num2str(-referenceLinesGradients(i)),'})'...
        ];
51 end
52 legend(legendStrings,'Location','southwest');
53
54 box on;
55 title(['\kappa = ',num2str(kappa),...
56     ', k(u_0) = ',num2str(ku0),...
57     ', g_F = ',num2str(gF),...
58     ', g_U = ',num2str(gU),...
59     ', g_P = ',num2str(gU),...
60     ', g(u_0) = ',num2str(gu0),...
61     ', h = 2^{-m}']);
62
63 end

```

Listing 29: errorAnalysis.m

```

1 function errorAnalysis(ku0)
2 % Computes approximate solutions fh,uh,ph to optimal control problem (Laplace equation)
3 % for different values of the mesh-size h -> 0. The L2- and H1-errors between
4 % the approximate solutions fh,uh,ph and known exact solutions f,u,p (for a
5 % specific 'target' u0) are computed for each value of h. The 'target' u0
6 % is chosen in a way that allows f,u,p to be polynomials on the entire
7 % domain Omega. This makes exact (!) error computation accessible.
8 %
9 % Inputs: ku0 in [2,3,4]: The main discretization parameter. Defines the
10 %           polynomials degrees for the Ansatz-spaces.
11 %
12 % Outputs:
13 % 1) Plot 1: A plot showing the approximate solutions fh,uh,ph and u0 for
14 % one value of h.
15 % 2) Plot 2: A plot showing the L2- and H1-errors between approximate and
16 % exact solutions for each value of h.
17 % 3) Plot 3: A plot showing the condition number of the system matrix for
18 % each value of h.
19
20 totalTimeStart = tic;

```

```

21
22 % Equation parameters and exact solutions:
23 kappa = 1/100;
24 [f,u,p] = defineFUP(kappa); % Exact polynomial solutions.
25 u0 = defineU0(kappa,u); % According 'target' function.
26
27 % Discretization parameters:
28 gF = ku0-2;
29 gU = ku0-1;
30 gu0 = gU+1;
31
32 % Main loop:
33 mMin = 1;
34 switch ku0
35     case 2
36         mMax = 13;
37     case 3
38         mMax = 11;
39     case 4
40         mMax = 7;
41 end
42
43 errors = zeros(mMax-mMin+1,5);
44 conditionNumbers = zeros(mMax-mMin+1,1);
45 for m = mMin:1:mMax
46     iterationTimeStart = tic;
47     numberOfNodes = 2^m+1;
48
49     % Assemble system matrix (F,U11,U12,U22,A11,A12,A22,B1,B2) and load vector (U01,U02...
50     ):
51     [F,U11,U12,U22,A11,A12,A22,B1,B2,U01,U02] = assemble(u0,kappa,gF,gU,gu0,...
52         numberOfNodes);
53
54     % Solve the LSE:
55     [fh_psiTr,u0_phiN,u0_phiTs,ph_phiN,ph_phiTs,conditionNumber] = solve(F,U11,U12,U22,...
56         A11,A12,A22,B1,B2,U01,U02);
57
58     % Compute the errors:
59     conditionNumbers(m-mMin+1) = conditionNumber;
60     errors(m-mMin+1,:) = computeErrors(kappa,f,u,p,gF,gU,numberOfNodes,fh_psiTr,u0_phiN...
61         ,u0_phiTs,ph_phiN,ph_phiTs,F,U11,U12,U22,A11,A12,A22);
62
63     % Plot the solutions:
64     if m==5
65         figureHandle_solutions = plotFhU0Ph(u0,kappa,ku0,gF,gU,gu0,numberOfNodes,...
66             fh_psiTr,u0_phiN,ph_phiN);
67     end
68
69     % Display iteration run time:
70     display(['Elements: ',num2str(numberOfNodes-1),' time: ',num2str(toc(...
71         iterationTimeStart)),' s.']);
72 end
73
74 % Compute and plot errors convergence rates:
75 figureHandle_errors = plotErrors(kappa,ku0,gF,gU,gu0,mMin,mMax,errors);
76
77 % Compute and plot condition number rate:
78 figureHandle_condNs = plotConditionNumbers(kappa,ku0,gF,gU,mMin,mMax,conditionNumbers);
79
80 % Display total run time:
81 display(['Total run time: ',num2str(toc(totalTimeStart)),' s.']);
82
83 % Save results:
84 hgexport(figureHandle_solutions,[fileparts(which('errorAnalysis.m')),'\Results\...
85     solutionsPlot.ku0-',num2str(ku0)],defineExportStyle('png',15,7,600));
86 hgexport(figureHandle_errors,[fileparts(which('errorAnalysis.m')),'\Results\...
87     errorsPlot.ku0-',num2str(ku0)],defineExportStyle('png',15,10,600));
88 hgexport(figureHandle_condNs,[fileparts(which('errorAnalysis.m')),'\Results\...
89     condNsPlot.ku0-',num2str(ku0)],defineExportStyle('png',15,10,600));
90

```

Listing 30: fadingColors.m

```

1 function colors = fadingColors(colorChooser,numberOfShades,fadingParameterMin,...
    fadingParameterMax)
2 % Computes the RGB-values of different shades of a given color (red, green
3 % or blue) and stores them rowwise in the matrix 'colors'.
4
5 switch colorChooser
6     case 'red'
7         colorFaderRed = @(s) min(max(2-2*s,0),1);
8         colorFaderGreen = @(s) min(max(1-2*s,0),1);
9         colorFaderBlue = @(s) min(max(1-2*s,0),1);
10    case 'green'
11        colorFaderRed = @(s) min(max(1-2*s,0),1);
12        colorFaderGreen = @(s) min(max(2-2*s,0),1);
13        colorFaderBlue = @(s) min(max(1-2*s,0),1);
14    case 'blue'
15        colorFaderRed = @(s) min(max(1-2*s,0),1);
16        colorFaderGreen = @(s) min(max(1-2*s,0),1);
17        colorFaderBlue = @(s) min(max(2-2*s,0),1);
18 end
19
20 colors= zeros(numberOfShades,3);
21 for i = 1:numberOfShades
22     fadingParameter = (1-(i-1)/(numberOfShades-1))*fadingParameterMin + (i-1)/(...
        numberOfShades-1)*fadingParameterMax;
23     colors(i,:) = [colorFaderRed(fadingParameter),colorFaderGreen(fadingParameter),...
        colorFaderBlue(fadingParameter)];
24 end
25
26 end

```

Listing 31: plotFhsUhsU0Phs.m

```

1 function figureHandle = plotFhsUhsU0Phs(u0,kappas,ku0,gF,gU,gu0,numberofNodes,fhs.psiTr...
    ,uhs.phIN,phs.phIN)
2 % Sets up a figure with three subplots containing the plots of fh,uh,u0 and
3 % ph.
4 % Subplot 1: Contains the solutions fh for all values of kappa.
5 % Subplot 2: Contains the solutions uh for all values of kappa and the
6 % 'target' u0.
7 % Subplot 3: Contains the solutions ph for all values of kappa.
8 %
9 % Note: Only the nodal values of the solutions are used for plotting.
10
11 % Define main mesh on Omega:
12 [h,Nodes,...
13     FTiMat,FTiVect,...
14     I_gF,I_gU,I_gU_X,indicesElements,indicesNodes,indicesInnerNodes,...
15     idxTrafo.psiTr_iFromIndexTR,idxTrafo.psiTr_indexTRFromI,psiTr_dim,...
16     idxTrafo.phIN_iFromIndexN,idxTrafo.phIN_indexNFromI,phiN_dim,...
17     idxTrafo.phiTs_iFromIndexTS,idxTrafo.phiTs_indexTSFromI,phiTs_dim] = defineMesh(gF,...
    gu0,numberofNodes);
18
19 % Define plot specifiers:
20 colorFaderMin = 0.2;
21 colorFaderMax = 0.8;
22 colorsF = fadingColors('blue',length(kappas),colorFaderMin,colorFaderMax);
23 colorsU = fadingColors('red',length(kappas),colorFaderMin,colorFaderMax);
24 colorsP = fadingColors('green',length(kappas),colorFaderMin,colorFaderMax);
25 linestyleFUP = '--';
26 linewidthFUP = 2;
27 colorU0 = 'black';
28 linestyleU0 = '--';
29 linewidthU0 = 2;

```

```

30
31 % Plot:
32 figureHandle = figure;
33
34 % Subplot 1:
35 subplot(1,3,1);
36 hold on;
37 legendStrings = cell(length(kappas),1);
38 for i = 1:length(kappas)
39     stairs(Nodes,[fhs_psiTr(i,1:gF+1:end),fhs_psiTr(i,end-gF)],'color',colorsF(i,:), '...
        linestyle',linestyleFUP,'linewidth',linewidthFUP);
40     legendStrings{i} = ['f_h(x) (\kappa = ',num2str(kappas(i)),')'];
41 end
42 xlabel('x');
43 legend(legendStrings,'Location','northoutside');
44 box on;
45
46 % Subplot 2:
47 subplot(1,3,2);
48 hold on;
49 legendStrings = cell(length(kappas)+1,1);
50 for i = 1:length(kappas)
51     plot(Nodes,[0,uhs_phiN(i,:),0],'color',colorsU(i,:), 'linestyle',linestyleFUP, '...
        linewidth',linewidthFUP);
52     legendStrings{i} = ['u_h(x) (\kappa = ',num2str(kappas(i)),')'];
53 end
54 legendStrings{end} = 'u_0(x) (\kappa = 0)';
55 plot(Nodes,u0(Nodes),'color',colorU0,'linestyle',linestyleU0,'linewidth',linewidthU0);
56 xlabel('x');
57 legend(legendStrings,'Location','northoutside');
58 box on;
59
60 % Subplot 3:
61 subplot(1,3,3);
62 hold on;
63 legendStrings = cell(length(kappas),1);
64 for i = 1:length(kappas)
65     plot(Nodes,[0,phs_phiN(i,:),0],'color',colorsP(i,:), 'linestyle',linestyleFUP, '...
        linewidth',linewidthFUP);
66     legendStrings{i} = ['p_h(x) (\kappa = ',num2str(kappas(i)),')'];
67 end
68 xlabel('x');
69 legend(legendStrings,'Location','northoutside');
70 box on;
71
72 % Title:
73 titleString = ['k(u_0) = ',num2str(ku0),...
74     ', g_F = ',num2str(gF),...
75     ', g_U = ',num2str(gU),...
76     ', g_P = ',num2str(gU),...
77     ', g(u_0) = ',num2str(gu0),...
78     ', #Elements: ',num2str(numberOfNodes-1)];
79 annotation('textbox',[0,0.94,1,0.06],...
80     'String',titleString,...
81     'EdgeColor','none',...
82     'HorizontalAlignment','center',...
83     'VerticalAlignment','bottom');
84
85 end

```

Listing 32: kappaComparison.m

```

1 function kappaComparison
2 % Computes approximate solutions fh,uh,ph to optimal control problem (Laplace equation)
3 % for different values of kappa and plots them together in one plot. This
4 % illustrates the role of the parameter kappa in the control problem.
5 %
6 % Inputs: None.
7 %

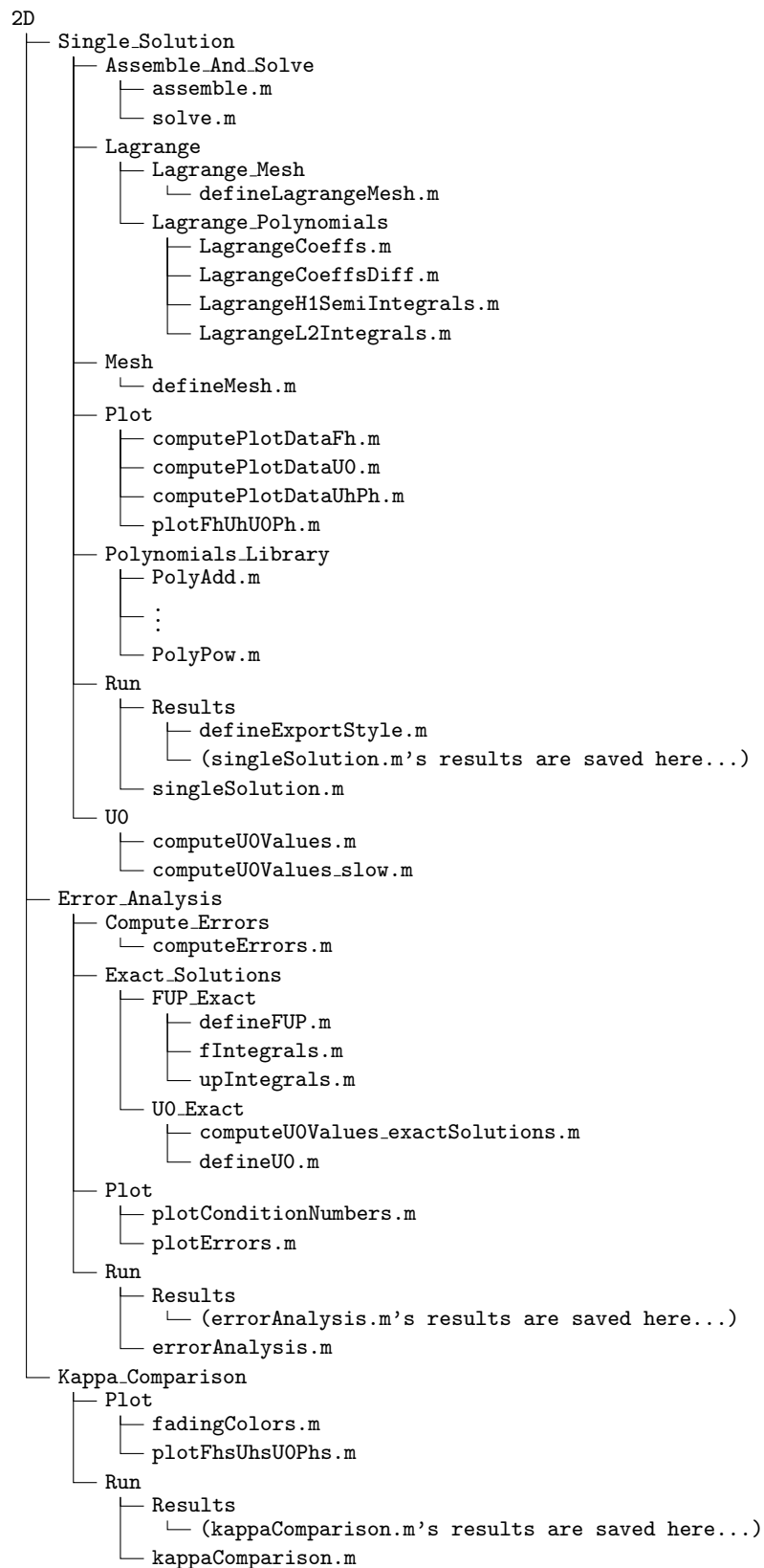
```

```

8 % Outputs: A plot showing the approximate solutions fh,uh,ph and the
9 % 'target' u0 for different values of kappa.
10
11 % Equation parameters:
12 a = 0.3;
13 b = 0.4;
14 c = 0.7;
15 u0 = @(x) ((a<=x)&(x<=b)).*(x-a)/(b-a) + ((b<x)&(x<=c)).*((b-x)/(c-b)+1);
16
17 % Discretization parameters:
18 ku0 = 2;
19 gF = ku0-2;
20 gU = ku0-1;
21 gu0 = gU+1;
22 numberOfNodes = 2^8+1;
23
24 % Define main mesh on Omega:
25 [h,Nodes,...
26     FTiMat,FTiVect,...
27     I_gF,I_gU,I_gU_X,indicesElements,indicesNodes,indicesInnerNodes,...
28     idxTrafo.psiTr.iFromIndexTR,idxTrafo.psiTr.indexTRFromI,psiTr.dim,...
29     idxTrafo.phiN.iFromIndexN,idxTrafo.phiN.indexNFromI,phiN.dim,...
30     idxTrafo.phiTs.iFromIndexTS,idxTrafo.phiTs.indexTSFromI,phiTs.dim] = defineMesh(gF,...
31     gU,numberOfNodes);
32
33 % Main loop:
34 kappas = [1/10,1/100,1/1000];
35 fhs_psiTr = zeros(length(kappas),psiTr.dim);
36 uhs_phiN = zeros(length(kappas),phiN.dim);
37 phs_phiN = zeros(length(kappas),phiN.dim);
38 for i = 1:length(kappas)
39     % Assemble system matrix (F,U11,U12,U22,A11,A12,A22,B1,B2) and load vector (U01,U02...
40     ):
41     [F,U11,U12,U22,A11,A12,A22,B1,B2,U01,U02] = assemble(u0,kappas(i),gF,gU,gu0,...
42     numberOfNodes);
43
44     % Solve the LSE:
45     [fh_psiTr,uh_phiN,~,ph_phiN,~,~] = solve(F,U11,U12,U22,A11,A12,A22,B1,B2,U01,U02);
46
47     % Store solutions in table:
48     fhs_psiTr(i,:) = fh_psiTr;
49     uhs_phiN(i,:) = uh_phiN;
50     phs_phiN(i,:) = ph_phiN;
51 end
52
53 % Plot the solutions:
54 figureHandle = plotFhsUhsU0Phs(u0,kappas,ku0,gF,gU,gu0,numberOfNodes,fhs_psiTr,uhs_phiN...
55     ,phs_phiN);
56
57 % Save results:
58 hgexport(figureHandle,[fileparts(which('kappaComparison.m')),'\Results\solutionsPlot'],...
59     defineExportStyle('png',15,7,600));
60
61 end

```

3.3.2 2D



Ordnerstruktur 2D MATLAB Beispiel

Listing 33: assemble.m

```

1 function [F,U,A,B,U0] = assemble(u0Values,kappa,gF,gU,gu0,numberofNodesPerSide)
2 % Assembles the components F,U,A,B of the system matrix and the component
3 % U0 of the load vector.
4
5 % Define Lagrange mesh on reference element:
6 [LagrangeNodesX,LagrangeNodesY,I_gu0,...
7     idxTrafo.LagrangeNodes.indexLNFromIndexIJLN,...
8     idxTrafo.LagrangeNodes.indexIJLNFromIndexLN,LagrangeNodes_dim] = ...
9     defineLagrangeMesh(gu0);
10
11 % Define main mesh on Omega:
12 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
13     indicesIElements,indicesJElements,indicesKElements,...
14     indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...
15     idxTrafo.Elements.indexTFromIndexIJKT,idxTrafo.Elements.indexIJKTFromIndexT,...
16     Elements_dim,...
17     idxTrafo.Nodes.indexNFromIndexIJN,idxTrafo.Nodes.indexIJNFromIndexN,Nodes_dim,...
18     idxTrafo.InnerNodes.indexNFromIndexIJN,idxTrafo.InnerNodes.indexIJNFromIndexN,...
19     InnerNodes_dim] = defineMesh(numberofNodesPerSide);
20
21 % Compute Lagrange polynomials integrals:
22 LagrangeL2Integrals.gF.gF = LagrangeL2Integrals(gF,gF);
23 LagrangeL2Integrals.gF.gU = LagrangeL2Integrals(gF,gU);
24 LagrangeL2Integrals.gU.gU = LagrangeL2Integrals(gU,gU);
25 LagrangeL2Integrals.gu0.gU = LagrangeL2Integrals(gu0,gU);
26 [LagrangeH1SemiXXIntegrals.gU.gU,LagrangeH1SemiXYIntegrals.gU.gU,...
27     LagrangeH1SemiYXIntegrals.gU.gU,LagrangeH1SemiYYIntegrals.gU.gU] = ...
28     LagrangeH1SemiIntegrals(gU,gU);
29 [LagrangeH1SemiXXIntegrals.gu0.gU,LagrangeH1SemiXYIntegrals.gu0.gU,...
30     LagrangeH1SemiYXIntegrals.gu0.gU,LagrangeH1SemiYYIntegrals.gu0.gU] = ...
31     LagrangeH1SemiIntegrals(gu0,gU);
32
33 % Assemble the components of the system matrix:
34 % F:
35 valueF.lookup = (kappa)/(1-kappa)*abs(det(FTijkMat(indicesIElements(1),indicesJElements...
36     (1),indicesKElements(1))))...
37     *LagrangeL2Integrals.gF.gF(1,1);
38
39 F = valueF.lookup*speye(Elements_dim,Elements_dim);
40
41 % U,A:
42 valueU.lookup = [0,0,0,0,0,0,0]; % Lookup tables for better performance.
43 valueA.lookup = [0,0,0,0,0,0,0];
44
45 indexIN = indicesIInnerNodes(1);
46 indexJN = indicesJInnerNodes(1);
47 neighbourNodesIds = [1:1:7];
48 for neighbourNodeId = neighbourNodesIds
49     indicesIT = [indexIN,indexIN-1,indexIN-1,indexIN-1,indexIN,indexIN];
50     indicesJT = [indexJN,indexJN,indexJN,indexJN-1,indexJN-1,indexJN-1];
51     indicesKT = [1,2,1,2,1,2];
52     iNs = [1,3,2,1,3,2];
53     neighbourElementsIds = [1:1:6];
54     useNeighbourElement = [false,false,false,false,false,false];
55     switch neighbourNodeId
56     case 1 % Node itself.
57         useNeighbourElement = [true,true,true,true,true,true];
58         iTildes = [1,3,2,1,3,2];
59     case 2 % Right neighbour.
60         useNeighbourElement(1) = true;
61         useNeighbourElement(6) = true;
62         iTildes = [2,0,0,0,0,1];
63     case 3 % Top neighbour.
64         useNeighbourElement(1) = true;
65         useNeighbourElement(2) = true;
66         iTildes = [3,1,0,0,0,0];
67     case 4 % Top left neighbour.
68         useNeighbourElement(2) = true;
69         useNeighbourElement(3) = true;

```

```

63     iNTildes = [0,2,3,0,0,0];
64     case 5 % Left neighbour.
65         useNeighbourElement(3) = true;
66         useNeighbourElement(4) = true;
67         iNTildes = [0,0,1,2,0,0];
68     case 6 % Bottom neighbour.
69         useNeighbourElement(4) = true;
70         useNeighbourElement(5) = true;
71         iNTildes = [0,0,0,3,1,0];
72     case 7 % Bottom right neighbour.
73         useNeighbourElement(5) = true;
74         useNeighbourElement(6) = true;
75         iNTildes = [0,0,0,0,2,3];
76 end
77
78 valueU = 0;
79 valueA = 0;
80 for neighbourElementId = neighbourElementsIds(useNeighbourElement) % iterate ...
81     through relevant elements
82     absDetMat = abs(det(FTijkMat(indicesIT(neighbourElementId), indicesJT(...
83         neighbourElementId), indicesKT(neighbourElementId))));
84     M = FTijkInvMat(indicesIT(neighbourElementId), indicesJT(neighbourElementId), ...
85         indicesKT(neighbourElementId)) ...
86         *FTijkInvMat(indicesIT(neighbourElementId), indicesJT(neighbourElementId), ...
87         indicesKT(neighbourElementId));
88     iN = iNs(neighbourElementId);
89     iNTilde = iNTildes(neighbourElementId);
90
91     valueH1Semi = absDetMat*(M(1,1)*LagrangeH1SemiXXIntegrals_gU_gU(iN, iNTilde) ...
92         + M(1,2)*LagrangeH1SemiXYIntegrals_gU_gU(iN, iNTilde) ...
93         + M(2,1)*LagrangeH1SemiYXIntegrals_gU_gU(iN, iNTilde) ...
94         + M(2,2)*LagrangeH1SemiYYIntegrals_gU_gU(iN, iNTilde));
95     valueL2 = absDetMat*LagrangeL2Integrals_gU_gU(iN, iNTilde);
96
97     valueU = valueU + valueH1Semi + valueL2;
98     valueA = valueA + valueH1Semi;
99 end
100
101 U = sparse(InnerNodes_dim, InnerNodes_dim);
102 A = sparse(InnerNodes_dim, InnerNodes_dim);
103 for indexJN = indicesJInnerNodes
104     for indexIN = indicesIInnerNodes
105         % Set neighbour nodes:
106         indicesINTilde = [indexIN, indexIN+1, indexIN, indexIN-1, indexIN-1, indexIN, indexIN...
107             +1];
108         indicesJNTilde = [indexJN, indexJN, indexJN+1, indexJN+1, indexJN, indexJN-1, indexJN...
109             -1];
110         neighbourNodesIds = [1:1:7];
111         useNeighbourNode = [true, false, false, false, false, false, false]; % node itself ...
112         always used
113         if indexIN < indicesIInnerNodes(end)
114             useNeighbourNode(2) = true;
115             if indexJN > indicesJInnerNodes(1)
116                 useNeighbourNode(7) = true;
117             end
118         end
119         if indexJN < indicesJInnerNodes(end)
120             useNeighbourNode(3) = true;
121             if indexIN > indicesIInnerNodes(1)
122                 useNeighbourNode(4) = true;
123             end
124         end
125         if indexIN > indicesIInnerNodes(1)
126             useNeighbourNode(5) = true;
127         end
128         if indexJN > indicesJInnerNodes(1)
129             useNeighbourNode(6) = true;
130         end
131     end
132 end

```

```

126         useNeighbourNode(6) = true;
127     end
128
129     % Iterate through neighbour nodes:
130     for neighbourNodeId = neighbourNodesIds(useNeighbourNode)
131         indexN = idxTrafo.InnerNodes.indexNFromIndexIJN(indexIN, indexJN);
132         indexNTilde = idxTrafo.InnerNodes.indexNFromIndexIJN(indicesINTilde(...
            neighbourNodeId), indicesJNTilde(neighbourNodeId));
133         U(indexN, indexNTilde) = valueU.lookup(neighbourNodeId);
134         A(indexN, indexNTilde) = valueA.lookup(neighbourNodeId);
135     end
136 end
137 end
138
139 % B:
140 valueB.lookup = -abs(det(FTijkMat(indicesIElements(1), indicesJElements(1), ...
    indicesKElements(1)))) * LagrangeL2Integrals.gF.gU(1,1);
141
142 B = sparse(InnerNodes_dim, Elements_dim);
143 for indexJN = indicesJInnerNodes
144     for indexIN = indicesIInnerNodes
145         indicesIT = [indexIN, indexIN-1, indexIN-1, indexIN-1, indexIN, indexIN];
146         indicesJT = [indexJN, indexJN, indexJN, indexJN-1, indexJN-1, indexJN-1];
147         indicesKT = [1, 2, 1, 2, 1, 2];
148         neighbourElementsIds = [1:1:6];
149
150         for neighbourElementId = neighbourElementsIds
151             indexN = idxTrafo.InnerNodes.indexNFromIndexIJN(indexIN, indexJN);
152             indexT = idxTrafo.Elements.indexTFromIndexIJKT(indicesIT(neighbourElementId...
                ), indicesJT(neighbourElementId), indicesKT(neighbourElementId));
153             B(indexN, indexT) = valueB.lookup;
154         end
155     end
156 end
157
158 % Assemble the load vector:
159 % U0:
160 absDetFTijkMat.lookup = abs(det(FTijkMat(indicesIElements(1), indicesJElements(1), ...
    indicesKElements(1))));
161 M.lookup = FTijkInvMat(indicesIElements(1), indicesJElements(1), indicesKElements(1)) ...
    *FTijkInvMat(indicesIElements(1), indicesJElements(1), indicesKElements(1));
162
163 U0 = sparse(InnerNodes_dim, 1);
164 for indexJN = indicesJInnerNodes
165     for indexIN = indicesIInnerNodes
166         indicesIT = [indexIN, indexIN-1, indexIN-1, indexIN-1, indexIN, indexIN];
167         indicesJT = [indexJN, indexJN, indexJN, indexJN-1, indexJN-1, indexJN-1];
168         indicesKT = [1, 2, 1, 2, 1, 2];
169         iNs = [1, 3, 2, 1, 3, 2];
170         neighbourElementsIds = [1:1:6];
171
172         sum = 0;
173         for neighbourElementId = neighbourElementsIds
174             for indexJLN = 1:gu0
175                 for indexILN = 0:1:(gu0-indexJLN)
176                     indexT = idxTrafo.Elements.indexTFromIndexIJKT(...
177                         indicesIT(neighbourElementId), ...
178                         indicesJT(neighbourElementId), ...
179                         indicesKT(neighbourElementId));
180                     indexLN = idxTrafo.LagrangeNodes.indexLNFromIndexIJLN(indexILN+1, ...
181                         indexJLN+1);
182                     iN = iNs(neighbourElementId);
183                     integral = M.lookup(1,1) * LagrangeH1SemiXXIntegrals.gu0.gU(indexLN, ...
184                         iN) ...
185                         + M.lookup(1,2) * LagrangeH1SemiXYIntegrals.gu0.gU(indexLN, iN) ...
186                         + M.lookup(2,1) * LagrangeH1SemiYXIntegrals.gu0.gU(indexLN, iN) ...
187                         + M.lookup(2,2) * LagrangeH1SemiYYIntegrals.gu0.gU(indexLN, iN);
188
189                     sum = sum + absDetFTijkMat.lookup * u0Values(indexT, indexLN) * (...
190                         LagrangeL2Integrals.gu0.gU(indexLN, iN) + integral);

```

```

189         end
190     end
191 end
192
193     indexN = idxTrafo.InnerNodes.indexNFromIndexIJN(indexIN, indexJN);
194     U0(indexN) = sum;
195 end
196 end
197
198 end

```

Listing 34: solve.m

```

1 function [fh,uh,ph,conditionNumber] = solve(F,U,A,B,U0)
2 % Assembles the system matrix from its components F,U,A,B and solves the
3 % resulting LSE.
4
5 % Final assemble of the system matrix:
6 Elements_dim = size(F,1);
7 InnerNodes_dim = size(U,1);
8
9 systemMatrix = [F,sparse(Elements_dim,InnerNodes_dim),B';...
10    sparse(InnerNodes_dim,Elements_dim),U,A';...
11    B,A,sparse(InnerNodes_dim,InnerNodes_dim)];
12 conditionNumber = condest(systemMatrix);
13
14 % Final assemble of the load vector:
15 loadVector = [sparse(Elements_dim,1);U0;sparse(InnerNodes_dim,1)];
16
17 % Solve the LSE:
18 solution = full(systemMatrix\loadVector)';
19
20 % Extract the coefficients of the approximate solutions fh,uh,ph from the
21 % solution vector:
22 fh = solution(1:1:Elements_dim);
23 uh = solution(Elements_dim+[1:1:InnerNodes_dim]);
24 ph = solution(Elements_dim+InnerNodes_dim+[1:1:InnerNodes_dim]);
25
26 end

```

Listing 35: defineLagrangeMesh.m

```

1 function [LagrangeNodesX,LagrangeNodesY,I_g,...
2     idxTrafo.LagrangeNodes.indexLNFromIndexIJLN,...
3     idxTrafo.LagrangeNodes.indexIJLNFromIndexLN,LagrangeNodes_dim] = ...
4     defineLagrangeMesh(g)
5 % Defines all necessary data necessary for representing the Lagrange mesh
6 % on the reference element:
7 % 1) The actual coordinates of all Lagrange nodes (LagrangeNodesX, LagrangeNodesY).
8 % 2) An index array containing the row-column-indices of the Lagrange nodes (I_g).
9 % 3) Index transformations used to transform row-column-indices into
10 % single indices needed e.g. in the assembly of the load vector (...
11 % idxTrafo.LagrangeNodes...).
12 % (Note: Using function handles for the index transformations
13 % would result in worse computation time.)
14 % 4) The total numbers of Lagrange nodes (LagrangeNodes_dim).
15
16 % Define geometry data:
17 LagrangeNodesX = [0:1:g]/g;
18 LagrangeNodesY = [0:1:g]/g;
19
20 % Define index arrays:
21 I_g = [0:1:g];
22
23 % Define index transformations:
24
25 % Index transformations for Lagrange nodes:

```

```

23 % (indexILN,indexJLN) in IJ_g
24 % <->
25 % (indexLN) in {1,...,1/2*(g+1)*(g+2)}
26 idxTrafo.LagrangeNodes.indexLNFromIndexIJLN = zeros(g+1,g+1);
27 idxTrafo.LagrangeNodes.indexIJLNFromIndexLN = zeros(1,2,1/2*(g+1)*(g+2));
28 indexLN = 1;
29 for indexJLN = 1:g
30     for indexILN = 0:1:(g-indexJLN)
31         idxTrafo.LagrangeNodes.indexLNFromIndexIJLN(indexILN+1,indexJLN+1) = indexLN;
32         idxTrafo.LagrangeNodes.indexIJLNFromIndexLN(:, :, indexLN) = [indexILN, indexJLN];
33         indexLN = indexLN + 1;
34     end
35 end
36 LagrangeNodes.dim = 1/2*(g+1)*(g+2);
37
38 end

```

Listing 36: LagrangeCoeffs.m

```

1 function LagrangeCoeffsMatrix = LagrangeCoeffs(g)
2 % Computes the coefficients of all Lagrange polynomials of degree g
3 % on the reference element. The matrix LagrangeCoeffsMatrix(:, :, indexLN)
4 % contains all coefficients of the indexLN'th Lagrange polynomial.
5
6 % Define Lagrange mesh on reference element:
7 [LagrangeNodesX, LagrangeNodesY, I_g, ...
8     idxTrafo.LagrangeNodes.indexLNFromIndexIJLN, ...
9     idxTrafo.LagrangeNodes.indexIJLNFromIndexLN, LagrangeNodes.dim] = ...
10     defineLagrangeMesh(g);
11
12 % Compute coefficients:
13 LagrangeCoeffsMatrix = zeros(g+1,g+1,1/2*(g+1)*(g+2));
14 for indexJLN = 1:g
15     for indexILN = 0:1:(g-indexJLN)
16         P = 1;
17         for l = 0:1:indexILN-1
18             P = 1/(indexILN-l)*PolyMult(P, [[-1,0]; [g,0]]);
19         end
20         for l = 0:1:indexJLN-1
21             P = 1/(indexJLN-l)*PolyMult(P, [[-1,g]; [0,0]]);
22         end
23         for l = indexILN+indexJLN+1:1:g
24             P = 1/(indexILN+indexJLN-l)*PolyMult(P, [[-1,g]; [g,0]]);
25         end
26         indexLN = idxTrafo.LagrangeNodes.indexLNFromIndexIJLN(indexILN+1,indexJLN+1);
27         LagrangeCoeffsMatrix(:, :, indexLN) = P;
28     end
29 end

```

Listing 37: LagrangeCoeffsDiff.m

```

1 function [LagrangeCoeffsDiffXMatrix, LagrangeCoeffsDiffYMatrix] = LagrangeCoeffsDiff(g)
2 % Computes the partial derivatives with respect to 'x' and 'y' of all Lagrange
3 % polynomials of degree g on the reference element. E.g.: The matrix
4 % LagrangeCoeffsDiffXMatrix(:, :, indexLN) contains all coefficients of the
5 % x-derivative of the indexLN'th Lagrange polynomial.
6
7 % Compute Lagrange polynomials coefficients:
8 LagrangeCoeffsMatrix = LagrangeCoeffs(g);
9
10 % Compute partial derivatives:
11 LagrangeCoeffsDiffXMatrix = zeros(max(size(LagrangeCoeffsMatrix(:, :, 1), 1)-1, 1), size(...
12     LagrangeCoeffsMatrix(:, :, 1), 2), size(LagrangeCoeffsMatrix, 3));
13 LagrangeCoeffsDiffYMatrix = zeros(size(LagrangeCoeffsMatrix(:, :, 1), 1), max(size(...
14     LagrangeCoeffsMatrix(:, :, 1), 2)-1, 1), size(LagrangeCoeffsMatrix, 3));

```

```

13 for indexLN = 1:1:size(LagrangeCoeffsMatrix,3)
14     LagrangeCoeffsDiffXMatrix(:, :, indexLN) = PolyDiff(LagrangeCoeffsMatrix(:, :, indexLN) ...
15         , 'x');
16     LagrangeCoeffsDiffYMatrix(:, :, indexLN) = PolyDiff(LagrangeCoeffsMatrix(:, :, indexLN) ...
17         , 'y');
18 end

```

Listing 38: LagrangeH1SemiIntegrals.m

```

1 function [LagrangeH1SemiXXIntegrals, LagrangeH1SemiXYIntegrals, ...
2     LagrangeH1SemiYXIntegrals, LagrangeH1SemiYYIntegrals] = LagrangeH1SemiIntegrals(g1, ...
3     g2)
4 % Computes the integrals  $\int_0^1 \int_0^{1-x} (p-\{x/y\})(x,y) \cdot q-\{x/y\}(x,y) dx dy$  for all
5 % pairs (p,q) of Lagrange polynomials.
6 % Compute Lagrange polynomials coefficients:
7 [LagrangeCoeffsDiffXMatrix_g1, LagrangeCoeffsDiffYMatrix_g1] = LagrangeCoeffsDiff(g1);
8 [LagrangeCoeffsDiffXMatrix_g2, LagrangeCoeffsDiffYMatrix_g2] = LagrangeCoeffsDiff(g2);
9
10 % Compute monomial integrals:
11 MonomialIntegrals = PolyL1IntTRef.MonomialIntegrals(g1+g2, g1+g2);
12
13 % Compute the integrals  $\int_0^1 \int_0^{1-x} (p-x)(x,y) \cdot q-x(x,y) dx dy$ :
14 LagrangeH1SemiXXIntegrals = zeros(size(LagrangeCoeffsDiffXMatrix_g1,3), size(...
15     LagrangeCoeffsDiffXMatrix_g2,3));
16 for i = 1:1:size(LagrangeCoeffsDiffXMatrix_g1,3)
17     for j = 1:1:size(LagrangeCoeffsDiffXMatrix_g2,3)
18         LagrangeH1SemiXXIntegrals(i, j) = PolyL2IntTRef(LagrangeCoeffsDiffXMatrix_g1...
19             (:, :, i), LagrangeCoeffsDiffXMatrix_g2(:, :, j), MonomialIntegrals);
20     end
21 end
22
23 % Compute the integrals  $\int_0^1 \int_0^{1-x} (p-x)(x,y) \cdot q-y(x,y) dx dy$ :
24 LagrangeH1SemiXYIntegrals = zeros(size(LagrangeCoeffsDiffXMatrix_g1,3), size(...
25     LagrangeCoeffsDiffYMatrix_g2,3));
26 for i = 1:1:size(LagrangeCoeffsDiffXMatrix_g1,3)
27     for j = 1:1:size(LagrangeCoeffsDiffYMatrix_g2,3)
28         LagrangeH1SemiXYIntegrals(i, j) = PolyL2IntTRef(LagrangeCoeffsDiffXMatrix_g1...
29             (:, :, i), LagrangeCoeffsDiffYMatrix_g2(:, :, j), MonomialIntegrals);
30     end
31 end
32
33 % Compute the integrals  $\int_0^1 \int_0^{1-x} (p-y)(x,y) \cdot q-x(x,y) dx dy$ :
34 LagrangeH1SemiYXIntegrals = zeros(size(LagrangeCoeffsDiffYMatrix_g1,3), size(...
35     LagrangeCoeffsDiffXMatrix_g2,3));
36 for i = 1:1:size(LagrangeCoeffsDiffYMatrix_g1,3)
37     for j = 1:1:size(LagrangeCoeffsDiffXMatrix_g2,3)
38         LagrangeH1SemiYXIntegrals(i, j) = PolyL2IntTRef(LagrangeCoeffsDiffYMatrix_g1...
39             (:, :, i), LagrangeCoeffsDiffXMatrix_g2(:, :, j), MonomialIntegrals);
40     end
41 end
42
43 % Compute the integrals  $\int_0^1 \int_0^{1-x} (p-y)(x,y) \cdot q-y(x,y) dx dy$ :
44 LagrangeH1SemiYYIntegrals = zeros(size(LagrangeCoeffsDiffYMatrix_g1,3), size(...
45     LagrangeCoeffsDiffYMatrix_g2,3));
46 for i = 1:1:size(LagrangeCoeffsDiffYMatrix_g1,3)
47     for j = 1:1:size(LagrangeCoeffsDiffYMatrix_g2,3)
48         LagrangeH1SemiYYIntegrals(i, j) = PolyL2IntTRef(LagrangeCoeffsDiffYMatrix_g1...
49             (:, :, i), LagrangeCoeffsDiffYMatrix_g2(:, :, j), MonomialIntegrals);
50     end
51 end
52

```

Listing 39: LagrangeL2Integrals.m

```

1 function LagrangeL2Integrals = LagrangeL2Integrals(g1,g2)
2 % Computes the integrals  $\int (p(x,y)*q(x,y), x=0..1, y=0..1-x)$  for all
3 % pairs (p,q) of Lagrange polynomials.
4
5 % Compute Lagrange polynomials coefficients:
6 LagrangeCoeffsMatrix_g1 = LagrangeCoeffs(g1);
7 LagrangeCoeffsMatrix_g2 = LagrangeCoeffs(g2);
8
9 % Compute monomial integrals:
10 MonomialIntegrals = PolyL1IntTRef_MonomialIntegrals(g1+g2,g1+g2);
11
12 % Compute integrals:
13 LagrangeL2Integrals = zeros(size(LagrangeCoeffsMatrix_g1,3),size(...
14     LagrangeCoeffsMatrix_g2,3));
15 for i = 1:size(LagrangeCoeffsMatrix_g1,3)
16     for j = 1:size(LagrangeCoeffsMatrix_g2,3)
17         LagrangeL2Integrals(i,j) = PolyL2IntTRef(LagrangeCoeffsMatrix_g1(:, :, i), ...
18             LagrangeCoeffsMatrix_g2(:, :, j), MonomialIntegrals);
19     end
20 end
21 end
22 end

```

Listing 40: defineMesh.m

```

1 function [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
2     indicesIElements,indicesJElements,indicesKElements,...
3     indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...
4     idxTrafo.Elements_indexTFromIndexIJKT,idxTrafo.Elements_indexIJKTFromIndexT,...
5     Elements_dim,...
6     idxTrafo.Nodes_indexNFromIndexIJN,idxTrafo.Nodes_indexIJNFromIndexN,Nodes_dim,...
7     idxTrafo.InnerNodes_indexNFromIndexIJN,idxTrafo.InnerNodes_indexIJNFromIndexN,...
8     InnerNodes_dim] = defineMesh(numberOfNodesPerSide)
9 % Defines all necessary data necessary for representing the mesh on Omega:
10 % 1) The actual coordinates of all nodes (NodesX, NodesY).
11 % 2) Matrix-vector-representations of the affine transformations between the reference
12 % element and each element in the mesh (FTijkMat,...,FTijkInvVect).
13 % 3) Index arrays containing the row-column-(diagonal)-indices of elements and all/...
14 % inner nodes
15 % (indicesIElements,...,indicesJInnerNodes).
16 % 4) Index transformations used to transform row-column-(diagonal)-indices into
17 % single indices needed e.g. in the assembly of the system matrix
18 % (idxTrafo.Elements,...,idxTrafo.InnerNodes...).
19 % (Note: Using function handles also for the index transformations
20 % would result in worse computation time.)
21 % 5) The total numbers of elements, nodes and inner nodes
22 % (Elements_dim,...,InnerNodes_dim).
23
24 % Define geometry data:
25 h = sqrt(2)/(numberOfNodesPerSide-1);
26 NodesX = 1/(numberOfNodesPerSide-1)*([1:1:numberOfNodesPerSide]-1);
27 NodesY = 1/(numberOfNodesPerSide-1)*([1:1:numberOfNodesPerSide]-1);
28
29 % Define affine transformations:
30 FTijkMat = @(indexIT,indexJT,indexKT) (indexKT==1)*1/(numberOfNodesPerSide-1)*eye(2)...
31     + (indexKT==2)*(-1)/(numberOfNodesPerSide-1)*eye(2);
32 FTijkVect = @(indexIT,indexJT,indexKT) (indexKT==1)*[NodesX(indexIT);NodesY(indexJT)]...
33     + (indexKT==2)*[NodesX(indexIT+1);NodesY(indexJT+1)];
34 FTijkInvMat = @(indexIT,indexJT,indexKT) (indexKT==1)*(numberOfNodesPerSide-1)*eye(2)...
35     + (indexKT==2)*(-(numberOfNodesPerSide-1))*eye(2);
36 FTijkInvVect = @(indexIT,indexJT,indexKT) (indexKT==1)*(-(numberOfNodesPerSide-1))*[...
37     NodesX(indexIT);NodesY(indexJT)]...
38     + (indexKT==2)*(numberOfNodesPerSide-1)*[NodesX(indexIT+1);NodesY(indexJT+1)];
39
40 % Define index arrays:

```

```

37 indicesIElements = [1:1:(numberOfNodesPerSide-1)];
38 indicesJElements = [1:1:(numberOfNodesPerSide-1)];
39 indicesKElements = [1,2];
40 indicesIAllNodes = [1:1:numberOfNodesPerSide];
41 indicesJAllNodes = [1:1:numberOfNodesPerSide];
42 indicesIInnerNodes = [2:1:(numberOfNodesPerSide-1)];
43 indicesJInnerNodes = [2:1:(numberOfNodesPerSide-1)];
44
45 % Define index transformations:
46
47 % Index transformations for elements:
48 % (indexIT,indexJT,indexKT) in {1,...,n-1} x {1,...,n-1} x {1,2}
49 % <->
50 % (indexT) in {1,...,(n-1)^2*2}
51 %
52 % indexT = (indexJT-1)*(n-1)*2 + (indexIT-1)*2 + indexKT
53 % <->
54 % indexKT = mod(indexT-1,2)+1
55 % indexIT = mod((indexT-indexKT)/2,n-1)+1
56 % indexJT = 1/(2*(n-1))*(i-indexKT-2*(indexIT-1)) + 1
57 idxTrafo_Elements.indexTFromIndexIJKT = zeros(numberOfNodesPerSide-1,...
    numberOfNodesPerSide-1,2);
58 for indexIT = indicesIElements
59     for indexJT = indicesJElements
60         for indexKT = indicesKElements
61             idxTrafo_Elements.indexTFromIndexIJKT(indexIT,indexJT,indexKT) = (indexJT...
                -1)*(numberOfNodesPerSide-1)*2 + (indexIT-1)*2 + indexKT;
62         end
63     end
64 end
65 Elements_dim = (numberOfNodesPerSide-1)^2*2;
66 idxTrafo_Elements.indexIJKTFromIndexT = zeros(1,3,(numberOfNodesPerSide-1)^2*2);
67 for indexT = 1:1:Elements_dim
68     idxTrafo_Elements.indexIJKTFromIndexT(:, :, indexT) = [round(mod((indexT-(mod(indexT...
        -1,2)+1))/2,numberOfNodesPerSide-1)+1),...
69         round(1/(2*(numberOfNodesPerSide-1))*(indexT-(mod(indexT-1,2)+1)-2*(mod(...
            indexT-(mod(indexT-1,2)+1))/2,numberOfNodesPerSide-1)+1)-1)) + 1),...
70         round(mod(indexT-1,2)+1)];
71 end
72
73 % Index transformations for all nodes:
74 % (indexIN,indexJN) in {1,...,n} x {1,...,n}
75 % <->
76 % (indexN) in {1,...,n^2}
77 %
78 % indexN = (indexJN-1)*n + indexIN
79 % <->
80 % indexIN = mod(indexN-1,n) + 1
81 % indexJN = 1/n*(indexN-indexIN) +
82 idxTrafo_Nodes.indexNFromIndexIJN = zeros(numberOfNodesPerSide,numberOfNodesPerSide);
83 for indexIN = indicesIAllNodes
84     for indexJN = indicesJAllNodes
85         idxTrafo_Nodes.indexNFromIndexIJN(indexIN,indexJN) = (indexJN-1)*...
            numberOfNodesPerSide + indexIN;
86     end
87 end
88 Nodes_dim = numberOfNodesPerSide^2;
89 idxTrafo_Nodes.indexIJNFromIndexN = zeros(1,2,numberOfNodesPerSide^2);
90 for indexN = 1:1:Nodes_dim
91     idxTrafo_Nodes.indexIJNFromIndexN(:, :, indexN) = [round(mod(indexN-1,...
        numberOfNodesPerSide)+1),...
92         round(1/numberOfNodesPerSide*(indexN-(mod(indexN-1,numberOfNodesPerSide)+1)) + ...
            1)];
93 end
94
95 % Index transformations for inner nodes:
96 % (indexIN,indexJN) in {2,...,n-1} x {2,...,n-1}
97 % <->
98 % (indexN) in {1,...,(n-2)^2}
99 %

```

```

100 % indexN = (indexJN-2)*(n-2) + indexIN - 1
101 % <->
102 % indexIN = mod(indexN-1,n-2) + 2
103 % indexJN = 1/(n-2)*(indexN-indexIN+1) + 2
104 idxTrafo.InnerNodes.indexNFromIndexIJN = zeros(numberOfNodesPerSide,...
    numberOfNodesPerSide);
105 for indexIN = indicesIInnerNodes
106     for indexJN = indicesJInnerNodes
107         idxTrafo.InnerNodes.indexNFromIndexIJN(indexIN,indexJN) = (indexJN-2)*(...
            numberOfNodesPerSide-2) + indexIN - 1;
108     end
109 end
110 InnerNodes_dim = (numberOfNodesPerSide-2)^2;
111 idxTrafo.InnerNodes.indexIJNFromIndexN = zeros(1,2,(numberOfNodesPerSide-2)^2);
112 for indexN = 1:1:InnerNodes_dim
113     idxTrafo.InnerNodes.indexIJNFromIndexN(:, :, indexN) = [round(mod(indexN-1,...
        numberOfNodesPerSide-2)+2),...
114         round(1/(numberOfNodesPerSide-2)*(indexN-(mod(indexN-1,numberOfNodesPerSide-2)...
            +2)+1) + 2)];
115 end
116
117 end

```

Listing 41: computePlotDataFh.m

```

1 function [xData,yData,fhData,xVertData,yVertData,fhVertData] = computePlotDataFh(...
    numberOfNodesPerSide,fh)
2 % Computes the plot data for fh needed in 'plotFhUhU0Ph.m'.
3
4 % Define main mesh on Omega:
5 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
6     indicesIElements,indicesJElements,indicesKElements,...
7     indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...
8     idxTrafo.Elements.indexTFromIndexIJKT,idxTrafo.Elements.indexIJKTFromIndexT,...
    Elements_dim,...
9     idxTrafo.Nodes.indexNFromIndexIJN,idxTrafo.Nodes.indexIJNFromIndexN,Nodes_dim,...
10    idxTrafo.InnerNodes.indexNFromIndexIJN,idxTrafo.InnerNodes.indexIJNFromIndexN,...
    InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
11
12 % Define plot data:
13 xData = zeros(3,Elements_dim);
14 yData = zeros(3,Elements_dim);
15 fhData = zeros(3,Elements_dim);
16
17 xVertData = zeros(4,(numberOfNodesPerSide-1)^2*3 + 2*(numberOfNodesPerSide-1));
18 yVertData = zeros(4,(numberOfNodesPerSide-1)^2*3 + 2*(numberOfNodesPerSide-1));
19 fhVertData = zeros(4,(numberOfNodesPerSide-1)^2*3 + 2*(numberOfNodesPerSide-1));
20 indexFhVerts = 1; % Makes edge indexing easier.
21
22 for indexT = 1:1:Elements_dim
23     indexIJKT = idxTrafo.Elements.indexIJKTFromIndexT(:, :, indexT);
24     indexIT = indexIJKT(1);
25     indexJT = indexIJKT(2);
26     indexKT = indexIJKT(3);
27
28     % Plot horizontal triangle:
29     if indexKT==1
30         indicesIN = [indexIT,indexIT+1,indexIT];
31         indicesJN = [indexJT,indexJT,indexJT+1];
32     else
33         indicesIN = [indexIT+1,indexIT,indexIT+1];
34         indicesJN = [indexJT+1,indexJT+1,indexJT];
35     end
36
37     xData(:,indexT) = [NodesX(indicesIN(1));NodesX(indicesIN(2));NodesX(indicesIN(3))];
38     yData(:,indexT) = [NodesY(indicesJN(1));NodesY(indicesJN(2));NodesY(indicesJN(3))];
39     fhData(:,indexT) = fh(indexT)*[1;1;1];
40
41 % Plot vertical rectangles:

```

```

42     if indexKT==1
43         for l = [1,2,3]
44             switch l
45                 case 1 % Bottom edge:
46                     indicesIN = [indexIT,indexIT+1];
47                     indicesJN = [indexJT,indexJT];
48                     if indexJT > indicesJElements(1)
49                         indexTTilde = idxTrafo.Elements.indexTFromIndexIJKT(indexIT,...
50                             indexJT-1,2);
51                         fhTilde = fh(indexTTilde);
52                     else
53                         fhTilde = 0;
54                     end
55                 case 2 % Diagonal edge:
56                     indicesIN = [indexIT+1,indexIT];
57                     indicesJN = [indexJT,indexJT+1];
58                     indexTTilde = idxTrafo.Elements.indexTFromIndexIJKT(indexIT,indexJT...
59                         ,2);
60                     fhTilde = fh(indexTTilde);
61                 case 3 % Left edge:
62                     indicesIN = [indexIT,indexIT];
63                     indicesJN = [indexJT+1,indexJT];
64                     if indexIT > indicesIElements(1)
65                         indexTTilde = idxTrafo.Elements.indexTFromIndexIJKT(indexIT-1,...
66                             indexJT,2);
67                         fhTilde = fh(indexTTilde);
68                     else
69                         fhTilde = 0;
70                     end
71                 end
72
73                 xVertData(:,indexFhVerts) = [NodesX(indicesIN(1));NodesX(indicesIN(2));...
74                     NodesX(indicesIN(2));NodesX(indicesIN(1))];
75                 yVertData(:,indexFhVerts) = [NodesY(indicesJN(1));NodesY(indicesJN(2));...
76                     NodesY(indicesJN(2));NodesY(indicesJN(1))];
77                 fhVertData(:,indexFhVerts) = [fhTilde;fhTilde;fh(indexT);fh(indexT)];
78                 indexFhVerts = indexFhVerts + 1;
79             end
80         else
81             if indexIT==indicesIElements(end) % Right edge:
82                 indicesIN = [indexIT+1,indexIT+1];
83                 indicesJN = [indexJT,indexJT+1];
84
85                 xVertData(:,indexFhVerts) = [NodesX(indicesIN(1));NodesX(indicesIN(2));...
86                     NodesX(indicesIN(2));NodesX(indicesIN(1))];
87                 yVertData(:,indexFhVerts) = [NodesY(indicesJN(1));NodesY(indicesJN(2));...
88                     NodesY(indicesJN(2));NodesY(indicesJN(1))];
89                 fhVertData(:,indexFhVerts) = [0;0;fh(indexT);fh(indexT)];
90                 indexFhVerts = indexFhVerts + 1;
91             end
92
93             if indexJT==indicesJElements(end) % Top edge:
94                 indicesIN = [indexIT,indexIT+1];
95                 indicesJN = [indexJT+1,indexJT+1];
96
97                 xVertData(:,indexFhVerts) = [NodesX(indicesIN(1));NodesX(indicesIN(2));...
98                     NodesX(indicesIN(2));NodesX(indicesIN(1))];
99                 yVertData(:,indexFhVerts) = [NodesY(indicesJN(1));NodesY(indicesJN(2));...
100                     NodesY(indicesJN(2));NodesY(indicesJN(1))];
101                 fhVertData(:,indexFhVerts) = [0;0;fh(indexT);fh(indexT)];
102                 indexFhVerts = indexFhVerts + 1;
103             end
104         end
105     end
106 end
107 end
108 end

```

Listing 42: computePlotDataU0.m

```

1 function [xData,yData,u0Data] = computePlotDataU0(gu0,numberOfNodesPerSide,u0Values)
2 % Computes the plot data for u0 needed in 'plotFhUhU0Ph.m'.
3
4 % Define Lagrange mesh on reference element:
5 [LagrangeNodesX,LagrangeNodesY,I_gu0,...
6   idxTrafo.LagrangeNodes.indexLNFromIndexIJLN,...
7   idxTrafo.LagrangeNodes.indexIJLNFromIndexLN,LagrangeNodes_dim] = ...
8   defineLagrangeMesh(gu0);
9
10 % Define main mesh on Omega:
11 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
12  indicesIElements,indicesJElements,indicesKElements,...
13  indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...
14  idxTrafo.Elements.indexTFromIndexIJKT,idxTrafo.Elements.indexIJKTFromIndexT,...
15  Elements_dim,...
16  idxTrafo.Nodes.indexNFromIndexIJN,idxTrafo.Nodes.indexIJNFromIndexN,Nodes_dim,...
17  idxTrafo.InnerNodes.indexNFromIndexIJN,idxTrafo.InnerNodes.indexIJNFromIndexN,...
18  InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
19
20 % Define plot data:
21 xData = zeros(3,Elements_dim);
22 yData = zeros(3,Elements_dim);
23 u0Data = zeros(3,Elements_dim);
24
25 for indexT = 1:1:Elements_dim
26   indexIJKT = idxTrafo.Elements.indexIJKTFromIndexT(:, :, indexT);
27   indexIT = indexIJKT(1);
28   indexJT = indexIJKT(2);
29   indexKT = indexIJKT(3);
30
31   if indexKT==1
32     indicesIN = [indexIT,indexIT+1,indexIT];
33     indicesJN = [indexJT,indexJT,indexJT+1];
34   else
35     indicesIN = [indexIT+1,indexIT,indexIT+1];
36     indicesJN = [indexJT+1,indexJT+1,indexJT];
37   end
38
39   xData(:,indexT) = [NodesX(indicesIN(1));NodesX(indicesIN(2));NodesX(indicesIN(3))];
40   yData(:,indexT) = [NodesY(indicesJN(1));NodesY(indicesJN(2));NodesY(indicesJN(3))];
41
42   indexLN1 = idxTrafo.LagrangeNodes.indexLNFromIndexIJLN(1,1);
43   indexLN2 = idxTrafo.LagrangeNodes.indexLNFromIndexIJLN(gu0+1,1);
44   indexLN3 = idxTrafo.LagrangeNodes.indexLNFromIndexIJLN(1,gu0+1);
45   u0Data(:,indexT) = [u0Values(indexT,indexLN1);u0Values(indexT,indexLN2);u0Values(...
46     indexT,indexLN3)];
47 end
48
49 end

```

Listing 43: computePlotDataUhPh.m

```

1 function [xData,yData,uhData] = computePlotDataUhPh(numberOfNodesPerSide,uh)
2 % Computes the plot data for uh (ph resp.) needed in 'plotFhUhU0Ph.m'.
3
4 % Define main mesh on Omega:
5 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
6  indicesIElements,indicesJElements,indicesKElements,...
7  indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...
8  idxTrafo.Elements.indexTFromIndexIJKT,idxTrafo.Elements.indexIJKTFromIndexT,...
9  Elements_dim,...
10  idxTrafo.Nodes.indexNFromIndexIJN,idxTrafo.Nodes.indexIJNFromIndexN,Nodes_dim,...
11  idxTrafo.InnerNodes.indexNFromIndexIJN,idxTrafo.InnerNodes.indexIJNFromIndexN,...
12  InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
13
14 % Define plot data:
15 xData = zeros(3,Elements_dim);

```



```

31 faceColorP = faceColorFader*[0,1,0];
32 faceAlphaP = faceAlpha;
33 edgeColorP = edgeColorFader*[0,1,0];
34 edgeAlphaP = edgeAlpha;
35
36 faceColorU0 = 0.7*[1,1,1];
37 faceAlphaU0 = 0.5;
38 edgeColorU0 = 0.5*[1,1,1];
39 edgeAlphaU0 = 1;
40
41 % Plot:
42 figureHandle = figure;
43
44 % Subplot 1:
45 subplot(1,3,1);
46 [xData,yData,fhData,xVertData,yVertData,fhVertData] = computePlotDataFh(...
    numberOfNodesPerSide,fh);
47 patch(xData,yData,fhData,faceColorF,'FaceAlpha',faceAlphaF,'EdgeColor',edgeColorF,...
    EdgeAlpha',edgeAlphaF);
48 patch(xVertData,yVertData,fhVertData,faceColorF,'FaceAlpha',faceAlphaF,'EdgeColor',...
    edgeColorF,'EdgeAlpha',edgeAlphaF);
49 view(viewSpecifier);
50 xlabel('x');
51 ylabel('y');
52 legend('f_h(x,y)', 'Location', 'northoutside');
53 box on;
54
55 % Subplot 2:
56 subplot(1,3,2);
57 [xData,yData,u0Data] = computePlotDataU0(numberOfNodesPerSide,u0);
58 patch(xData,yData,u0Data,faceColorU,'FaceAlpha',faceAlphaU,'EdgeColor',edgeColorU,...
    EdgeAlpha',edgeAlphaU);
59 [xData,yData,u0Data] = computePlotDataU0(gu0,numberOfNodesPerSide,u0Values);
60 patch(xData,yData,u0Data,faceColorU0,'FaceAlpha',faceAlphaU0,'EdgeColor',edgeColorU0,...
    EdgeAlpha',edgeAlphaU0);
61 view(viewSpecifier);
62 xlabel('x');
63 ylabel('y');
64 legend('u_h(x,y)', 'u_0(x,y)', 'Location', 'northoutside');
65 box on;
66
67 % Subplot 3:
68 subplot(1,3,3);
69 [xData,yData,phData] = computePlotDataPh(numberOfNodesPerSide,ph);
70 patch(xData,yData,phData,faceColorP,'FaceAlpha',faceAlphaP,'EdgeColor',edgeColorP,...
    EdgeAlpha',edgeAlphaP);
71 view(viewSpecifier);
72 xlabel('x');
73 ylabel('y');
74 legend('p_h(x,y)', 'Location', 'northoutside');
75 box on;
76
77 % Title:
78 titleString = ['\kappa = ',num2str(kappa),...
79     ', k(u_0) = ',num2str(ku0),...
80     ', g_F = ',num2str(gF),...
81     ', g_U = ',num2str(gU),...
82     ', g_P = ',num2str(gU),...
83     ', g(u_0) = ',num2str(gu0),...
84     ', #Elements: ',num2str(Elements_dim)];
85 annotation('textbox',[0,0.94,1,0.06],...
86     'String',titleString,...
87     'EdgeColor','none',...
88     'HorizontalAlignment','center',...
89     'VerticalAlignment','bottom');
90
91 end

```

Listing 45: PolyAdd.m

```

1 function pPlusQ = PolyAdd(p,q)
2 % Computes the sum p+q of two polynomials of arbitrary degree.
3
4 pPlusQ = zeros(max(size(p,1),size(q,1)),max(size(p,2),size(q,2)));
5 pPlusQ(1:1:size(p,1),1:1:size(p,2)) = p;
6 pPlusQ(1:1:size(q,1),1:1:size(q,2)) = pPlusQ(1:1:size(q,1),1:1:size(q,2)) + q;
7
8 end

```

Listing 46: PolyAffineTransform.m

```

1 function pTransformed = PolyAffineTransform(p,A11,A22,b1,b2,PascalMatrix_gX,...
    PowersMatrix_gX,PascalMatrix_gY,PowersMatrix_gY)
2 % Performs affine transformation of polynomial p. Only transformations of
3 % the form  $F(x,y) = [[A11,0];[0,A22]]*[x;y] + [b1;b2]$  are supported. The resulting ...
    polynomial
4 % is pTransformed(x,y) = p(F(x,y)).
5
6 pTransformed = PolyAffineTransform_1D(p',A22,b2,PascalMatrix_gY,PowersMatrix_gY)';
7 pTransformed = PolyAffineTransform_1D(pTransformed,A11,b1,PascalMatrix_gX,...
    PowersMatrix_gX);
8
9 end

```

Listing 47: PolyAffineTransform_1D.m

```

1 function pTransformed = PolyAffineTransform_1D(p,a,b,PascalMatrix,PowersMatrix)
2 % Performs affine transformation of the polynomial p. The transformation is
3 % of the form  $F(x) = a*x + b$ . The resulting polynomial is pTransformed(x) =
4 % p(F(x)). The given matrices PascalMatrix and PowersMatrix depend on the
5 % degree of p only and are computed in advance. They allow for much faster
6 % computation.
7
8 pTransformed = diag(a.^[0:1:(size(p,1)-1)]) * (((b.^PowersMatrix).*PascalMatrix)*p);
9
10 end

```

Listing 48: PolyAffineTransform_1D_Matrices.m

```

1 function [PascalMatrix,PowersMatrix] = PolyAffineTransform_1D_Matrices(g)
2 % Computes the matrices PascalMatrix and PowersMatrix for later use in
3 % 'PolyAffineTransform_1D.m'.
4
5 PascalMatrix = zeros(g+1,g+1);
6 for i = 0:1:g
7     for j = i:1:g
8         PascalMatrix(i+1,j+1) = nchoosek(j,i);
9     end
10 end
11
12 PowersMatrix = zeros(g+1,g+1);
13 for i = 0:1:g
14     for j = i:1:g
15         PowersMatrix(i+1,j+1) = j-i;
16     end
17 end
18
19 end

```

Listing 49: PolyDiff.m

```

1 function pDiff = PolyDiff(p,variable)
2 % Differentiates the polynomial p with respect to given variable 'x' or
3 % 'y'. Uses MATLAB builtin function 'polyder'.
4
5 switch variable
6     case 'x'
7         pDiff = zeros(max(size(p,1)-1,1),size(p,2));
8         for j = 1:1:size(p,2)
9             tmp = fliplr(polyder(fliplr(p(:,j)')));
10            pDiff(1:1:length(tmp),j) = tmp;
11        end
12    case 'y'
13        pDiff = zeros(size(p,1),max(size(p,2)-1,1));
14        for i = 1:1:size(p,1)
15            tmp = fliplr(polyder(fliplr(p(i,:))));
16            pDiff(i,1:1:length(tmp)) = tmp;
17        end
18 end
19
20 end

```

Listing 50: PolyEval.m

```

1 function value = PolyEval(p,x,y)
2 % Evaluates the polynomial p at (x,y).
3
4 value = (x.^[0:1:(size(p,1)-1)])*p*(y.^[0:1:(size(p,2)-1)]);
5
6 end

```

Listing 51: PolyL1IntOmega.m

```

1 function integral = PolyL1IntOmega(p)
2 % Computes the integral int(p(x,y),x=0..1,y=0..1) for a given polynomial p.
3
4 integral = (1./[1:1:size(p,1)])*p*(1./[1:1:size(p,2)]);
5
6 end

```

Listing 52: PolyL1IntTRef.m

```

1 function integral = PolyL1IntTRef(p,MonomialIntegrals)
2 % Computes the integral int(p(x,y),x=0..1,y=0..1-x) = sum_{i,j=0}^{gX,gY}
3 % p_{ij}*int(x^i*y^j,x=0..1,y=0..1-x) for a given polynomial p. The given
4 % monomial integrals are computed in 'PolyL1IntTRef_MonomialIntegrals.m'.
5
6 integral = sum(sum(p.*MonomialIntegrals(1:1:size(p,1),1:1:size(p,2))));
7
8 end

```

Listing 53: PolyL1IntTRef_MonomialIntegrals.m

```

1 function MonomialIntegrals = PolyL1IntTRef_MonomialIntegrals(gX,gY)
2 % Computes the monomial integrals int(x^i*y^j,x=0..1,y=0..1-x) for later
3 % use in 'PolyL1IntTRef.m'.
4
5 MonomialIntegrals = zeros(gX+1,gY+1);
6 for i = 0:1:gX
7     for j = 0:1:gY
8         sum = 0;
9         for k = 0:1:j+1
10            sum = sum + nchoosek(j+1,k)*(-1)^k*1/((j+1)*(i+k+1));
11        end

```

```

12
13     MonomialIntegrals(i+1,j+1) = sum;
14 end
15 end
16
17 end

```

Listing 54: PolyL2IntOmega.m

```

1 function integral = PolyL2IntOmega(p,q)
2 % Computes the integral int(p(x,y)*q(x,y),x=0..1,y=0..1) for given
3 % polynomials p and q.
4
5 integral = PolyL1IntOmega(PolyMult(p,q));
6
7 end

```

Listing 55: PolyL2IntTRef.m

```

1 function integral = PolyL2IntTRef(p,q,MonomialIntegrals)
2 % Computes the integral int(p(x,y)*q(x,y),x=0..1,y=0..1-x) for given
3 % polynomials p and q.
4
5 integral = PolyL1IntTRef(PolyMult(p,q),MonomialIntegrals);
6
7 end

```

Listing 56: PolyMult.m

```

1 function product = PolyMult(p,q)
2 % Computes the product p*q of two polynomials.
3
4 product = conv2(p,q);
5
6 end

```

Listing 57: PolyPow.m

```

1 function pExp = PolyPow(p,exponent)
2 % Computes the polynomial p^exponent for given polynomial p and given exponent.
3
4 pExp = 1;
5 for i = 1:1:exponent
6     pExp = PolyMult(pExp,p);
7 end
8
9 end

```

Listing 58: defineExportStyle.m

```

1 function exportStyle = defineExportStyle(format,width,height,resolution)
2 % Defines the parameters used for saving (exporting) the result plots.
3
4 exportStyle = hgexport('factorystyle');
5 exportStyle.Format = format;
6 exportStyle.Units = 'centimeters';
7 exportStyle.Width = width;
8 exportStyle.Height = height;
9 exportStyle.Resolution = resolution;
10 exportStyle.Bounds = 'tight';
11
12 end

```

Listing 59: singleSolution.m

```

1 function singleSolution
2 % Computes approximate solutions fh,uh,ph to optimal control problem (Laplace equation)...
  and
3 % plots them.
4 %
5 % Inputs: None.
6 %
7 % Outputs: A plot showing the approximate solutions fh,uh,ph and the 'target' u0.
8
9 % Equation parameters:
10 kappa = 1/1000;
11 aX = 0.2;
12 bX = 0.5;
13 cX = 0.7;
14 aY = 0.4;
15 bY = 0.5;
16 cY = 0.6;
17 u0 = @(x,y) (((aX<=x) & (x<=bX)) .* (x-aX) / (bX-aX) + ((bX<x) & (x<=cX)) .* ((bX-x) / (cX-bX)+1)) '...
    ...
    * (((aY<=y) & (y<=bY)) .* (y-aY) / (bY-aY) + ((bY<y) & (y<=cY)) .* ((bY-y) / (cY-bY)+1));
18
19
20 % Discretization parameters:
21 ku0 = 2;
22 gF = ku0-2;
23 gU = ku0-1;
24 gu0 = gU+1;
25 numberOfNodesPerSide = 2^5+1; % Number of nodes on each of both sides of the domain ...
    Omega.
26
27 % Compute values of u0 at all nodes of mesh (including all Lagrange nodes):
28 u0Values = computeU0Values(u0,gu0,numberOfNodesPerSide);
29
30 % Assemble system matrix (F,U,A,B) and load vector (U0):
31 [F,U,A,B,U0] = assemble(u0Values,kappa,gF,gU,gu0,numberOfNodesPerSide);
32
33 % Solve the LSE:
34 [fh,uh,ph,~] = solve(F,U,A,B,U0);
35
36 % Plot the solutions:
37 figureHandle = plotFhUhU0Ph(u0Values,kappa,ku0,gF,gU,gu0,numberOfNodesPerSide,fh,uh,ph) ...
    ;
38
39 % Save results:
40 hgexport (figureHandle,[fileparts(which('singleSolution.m')),'\Results\solutionsPlot'],...
    defineExportStyle('png',15,8,600));
41
42 end

```

Listing 60: computeU0Values.m

```

1 function u0Values = computeU0Values(u0,gu0,numberOfNodesPerSide)
2 % Computes the values of u0 at all nodes of mesh on Omega (including all Lagrange
3 % nodes). The values could also be computed 'on the fly' when they are
4 % needed later on, but this would result in bad computation time. This is
5 % mainly a MATLAB issue and is bypassed by means of vectorization.
6 % Therefore the function handle u0 has to support vectorization. The
7 % geometry of Omega is exploited massively.
8
9 % Define Lagrange mesh on reference element:
10 [LagrangeNodesX,LagrangeNodesY,I_gu0,...
11   idxTrafo.LagrangeNodes.indexLNFromIndexIJLN,...
12   idxTrafo.LagrangeNodes.indexIJLNFromIndexLN,LagrangeNodes.dim] = ...
    defineLagrangeMesh(gu0);
13
14 % Define main mesh on Omega:
15 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
    indicesIElements,indicesJElements,indicesKElements,...

```

```

16     indicesIAllNodes, indicesJAllNodes, indicesIInnerNodes, indicesJInnerNodes, ...
17     idxTrafo.Elements.indexTFromIndexIJKT, idxTrafo.Elements.indexIJKTFromIndexT, ...
        Elements_dim, ...
18     idxTrafo.Nodes.indexNFromIndexIJN, idxTrafo.Nodes.indexIJNFromIndexN, Nodes_dim, ...
19     idxTrafo.InnerNodes.indexNFromIndexIJN, idxTrafo.InnerNodes.indexIJNFromIndexN, ...
        InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
20
21 % Compute u0Values (vectorized):
22 n = (numberOfNodesPerSide-1)*gu0 + 1;
23 xs = 1/(n-1)*([1:1:n]-1);
24 ys = 1/(n-1)*([1:1:n]-1);
25 u0Values.tmp = u0(xs,ys); % u0 has to support vectorization.
26
27 % Store the values in other format (for easier use later):
28 u0Values = zeros(Elements_dim, LagrangeNodes_dim);
29 u0ValueHasBeenSet = zeros(Elements_dim, LagrangeNodes_dim);
30 for indexJT = indicesJElements
31     for indexIT = indicesIElements
32         for indexKT = indicesKElements
33             indexT = idxTrafo.Elements.indexTFromIndexIJKT(indexIT, indexJT, indexKT);
34             for indexJLN = 1:gu0
35                 for indexILN = 0:1:(gu0-indexJLN)
36                     indexLN = idxTrafo.LagrangeNodes.indexLNFromIndexIJLN(indexILN+1, ...
                        indexJLN+1);
37                     if ~u0ValueHasBeenSet(indexT, indexLN)
38                         switch indexKT
39                             case 1
40                                 i = (indexIT-1)*gu0 + 1 + indexILN;
41                                 j = (indexJT-1)*gu0 + 1 + indexJLN;
42                             case 2
43                                 i = (indexIT-1)*gu0 + 1 + (gu0-indexILN);
44                                 j = (indexJT-1)*gu0 + 1 + (gu0-indexJLN);
45                             end
46                             u0Values(indexT, indexLN) = u0Values.tmp(i, j);
47                             u0ValueHasBeenSet(indexT, indexLN) = true;
48                         end
49                     end
50                 end
51             end
52         end
53     end
54 end
55 end

```

Listing 61: computeU0Values_slow.m

```

1 function u0Values = computeU0Values_slow(u0, gu0, numberOfNodesPerSide)
2 % Computes the values of u0 at all nodes of mesh on Omega (including all Lagrange
3 % nodes). This is a much slower version of 'computeU0Values.m', but allows
4 % for u0 to be not necessarily vectorizable.
5
6 % Define Lagrange mesh on reference element:
7 [LagrangeNodesX, LagrangeNodesY, I_gu0, ...
8     idxTrafo.LagrangeNodes.indexLNFromIndexIJLN, ...
9     idxTrafo.LagrangeNodes.indexIJLNFromIndexLN, LagrangeNodes_dim] = ...
        defineLagrangeMesh(gu0);
10
11 % Define main mesh on Omega:
12 [h, NodesX, NodesY, FTijkMat, FTijkVect, FTijkInvMat, FTijkInvVect, ...
13     indicesIElements, indicesJElements, indicesKElements, ...
14     indicesIAllNodes, indicesJAllNodes, indicesIInnerNodes, indicesJInnerNodes, ...
15     idxTrafo.Elements.indexTFromIndexIJKT, idxTrafo.Elements.indexIJKTFromIndexT, ...
16     Elements_dim, ...
17     idxTrafo.Nodes.indexNFromIndexIJN, idxTrafo.Nodes.indexIJNFromIndexN, Nodes_dim, ...
18     idxTrafo.InnerNodes.indexNFromIndexIJN, idxTrafo.InnerNodes.indexIJNFromIndexN, ...
19     InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
20
21 % Build lookup table for affine transformed coordinates of Lagrange nodes:

```

```

19 LagrangeNodeAffineTransformed.XCoord.lookup = zeros(length(indicesKElements),...
    LagrangeNodes_dim);
20 LagrangeNodeAffineTransformed.YCoord.lookup = zeros(length(indicesKElements),...
    LagrangeNodes_dim);
21 for indexKT = indicesKElements
22     for indexJLN = 1:gu0
23         for indexILN = 0:1:(gu0-indexJLN)
24             FTijkMatrix = FTijkMat(indicesIInnerNodes(1),indicesJInnerNodes(1),indexKT)...
                ;
25             LagrangeNode = [LagrangeNodesX(indexILN+1);LagrangeNodesY(indexJLN+1)];
26
27             indexLN = idxTrafo.LagrangeNodes_indexLNFromIndexIJLN(indexILN+1,indexJLN...
                +1);
28             LagrangeNodeAffineTransformed.XCoord.lookup(indexKT,indexLN) = FTijkMatrix...
                (1,:)*LagrangeNode;
29             LagrangeNodeAffineTransformed.YCoord.lookup(indexKT,indexLN) = FTijkMatrix...
                (2,:)*LagrangeNode;
30         end
31     end
32 end
33
34 % Compute u0Values (not vectorized):
35 u0Values = zeros(Elements_dim,LagrangeNodes_dim);
36 u0ValueHasBeenSet = zeros(Elements_dim,LagrangeNodes_dim);
37 for indexJT = indicesJElements
38     for indexIT = indicesIElements
39         for indexKT = indicesKElements
40             indexT = idxTrafo.Elements_indexTFromIndexIJKT(indexIT,indexJT,indexKT);
41             for indexJLN = 1:gu0
42                 for indexILN = 0:1:(gu0-indexJLN)
43                     indexLN = idxTrafo.LagrangeNodes_indexLNFromIndexIJLN(indexILN+1,...
                        indexJLN+1);
44                     if ~u0ValueHasBeenSet(indexT,indexLN)
45                         FTijkVector = FTijkVect(indexIT,indexJT,indexKT);
46
47                         u0Values(indexT,indexLN) = u0(...
                            LagrangeNodeAffineTransformed.XCoord.lookup(indexKT,indexLN...
                                ) + FTijkVector(1),...
                            LagrangeNodeAffineTransformed.YCoord.lookup(indexKT,indexLN...
                                ) + FTijkVector(2));
48
49                         u0ValueHasBeenSet(indexT,indexLN) = true;
50                     end
51                 end
52             end
53         end
54     end
55 end
56
57 end

```

Listing 62: computeErrors.m

```

1 function errors = computeErrors(kappa,f,u,p,...
2     gF,gU,numberOfNodesPerSide,...
3     fh,uh,ph,...
4     F,U,A)
5 % Computes the L2- and H1-errors between approximate and exact solutions by
6 % means of the following decompositions:
7 % 1) ||f-fh||_L2^2 = ||f||_L2^2 - 2*<f,fh>_L2 + ||fh||_L2^2
8 % 2) ||u-uh||_L2^2 = ||u||_L2^2 - 2*<u,uh>_L2 + ||uh||_L2^2
9 % 3) ||u-uh||_H1^2 = ||u||_H1^2 - 2*<u,uh>_H1 + ||uh||_H1^2
10 % 4) ||p-ph||_L2^2 = ||p||_L2^2 - 2*<p,ph>_L2 + ||ph||_L2^2
11 % 5) ||p-ph||_H1^2 = ||p||_H1^2 - 2*<p,ph>_H1 + ||ph||_H1^2
12 %
13 % Note: There hold ||fh||_L2^2 = (1-kappa)/kappa*fh'*fh' and analogous
14 % formulas for ||uh||_L2^2,...,||ph||_H1^2.
15
16 % Define main mesh on Omega:
17 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...

```

```

18     indicesIElements, indicesJElements, indicesKElements, ...
19     indicesIAllNodes, indicesJAllNodes, indicesIInnerNodes, indicesJInnerNodes, ...
20     idxTrafo.Elements.indexTFromIndexIJKT, idxTrafo.Elements.indexIJKTFromIndexT, ...
        Elements_dim, ...
21     idxTrafo.Nodes.indexNFromIndexIJN, idxTrafo.Nodes.indexIJNFromIndexN, Nodes_dim, ...
22     idxTrafo.InnerNodes.indexNFromIndexIJN, idxTrafo.InnerNodes.indexIJNFromIndexN, ...
        InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
23
24 % Compute derivatives of u and p:
25 uDiffX = PolyDiff(u, 'x');
26 uDiffY = PolyDiff(u, 'y');
27 pDiffX = PolyDiff(p, 'x');
28 pDiffY = PolyDiff(p, 'y');
29
30 % Compute scalar products:
31 % <f, fh>_L2:
32 L2IntegralsF = fIntegrals(f, gF, numberOfNodesPerSide);
33 absDetMat_lookup = abs(det(FTijkMat(1,1,1)));
34
35 L2SP_f.fh = 0;
36 for indexJT = indicesJElements
37     for indexIT = indicesIElements
38         for indexKT = indicesKElements
39             indexT = idxTrafo.Elements.indexTFromIndexIJKT(indexIT, indexJT, indexKT);
40             L2SP_f.fh = L2SP_f.fh + fh(indexT)*absDetMat_lookup*L2IntegralsF(indexT);
41         end
42     end
43 end
44
45 % <u, uh>_L2, <u, uh>_H1, <p, ph>_L2, <p, ph>_H1:
46 [L2IntegralsU, H1SemiIntegralsU] = upIntegrals(u, gU, numberOfNodesPerSide);
47 [L2IntegralsP, H1SemiIntegralsP] = upIntegrals(p, gU, numberOfNodesPerSide);
48 absDetMat_lookup = abs(det(FTijkMat(1,1,1)));
49
50 L2SP_u.uh = 0;
51 H1SP_u.uh = 0;
52 L2SP_p.ph = 0;
53 H1SP_p.ph = 0;
54 for indexJN = indicesJInnerNodes
55     for indexIN = indicesIInnerNodes
56         indexN = idxTrafo.InnerNodes.indexNFromIndexIJN(indexIN, indexJN);
57
58         indicesIT = [indexIN, indexIN-1, indexIN-1, indexIN-1, indexIN, indexIN];
59         indicesJT = [indexJN, indexJN, indexJN, indexJN-1, indexJN-1, indexJN-1];
60         indicesKT = [1, 2, 1, 2, 1, 2];
61         iNs = [1, 3, 2, 1, 3, 2];
62         neighbourElementsIds = [1:1:6];
63
64         for neighbourElementId = neighbourElementsIds
65             indexT = idxTrafo.Elements.indexTFromIndexIJKT(indicesIT(neighbourElementId...
66                 ), ...
67                 indicesJT(neighbourElementId), ...
68                 indicesKT(neighbourElementId));
69
70             valueL2_u = uh(indexN)*absDetMat_lookup*L2IntegralsU(indexT, iNs(...
71                 neighbourElementId));
72             valueH1Semi_u = uh(indexN)*absDetMat_lookup*H1SemiIntegralsU(indexT, iNs(...
73                 neighbourElementId));
74             valueL2_p = ph(indexN)*absDetMat_lookup*L2IntegralsP(indexT, iNs(...
75                 neighbourElementId));
76             valueH1Semi_p = ph(indexN)*absDetMat_lookup*H1SemiIntegralsP(indexT, iNs(...
77                 neighbourElementId));
78
79             L2SP_u.uh = L2SP_u.uh + valueL2_u;
80             H1SP_u.uh = H1SP_u.uh + valueL2_u + valueH1Semi_u;
81             L2SP_p.ph = L2SP_p.ph + valueL2_p;
82             H1SP_p.ph = H1SP_p.ph + valueL2_p + valueH1Semi_p;
83         end
84     end
85 end

```

```

81
82 % Compute squared norms of exact solutions:
83 % ||f||_L2^2, ||u||_L2^2, ||u||_H1^2, ||p||_L2^2, ||p||_H1^2:
84 L2NormSquared_f = PolyL2IntOmega(f,f);
85 L2NormSquared_u = PolyL2IntOmega(u,u);
86 H1NormSquared_u = L2NormSquared_u + PolyL2IntOmega(uDiffX,uDiffX) + PolyL2IntOmega(...
    uDiffY,uDiffY);
87 L2NormSquared_p = PolyL2IntOmega(p,p);
88 H1NormSquared_p = L2NormSquared_p + PolyL2IntOmega(pDiffX,pDiffX) + PolyL2IntOmega(...
    pDiffY,pDiffY);
89
90 % Compute squared norms of approximate solutions:
91 % ||fh||_L2^2, ||uh||_L2^2, ||uh||_H1^2, ||ph||_L2^2, ||ph||_H1^2:
92 L2NormSquared_fh = (1-kappa)/kappa*fh*F*fh';
93 H1NormSquared_uh = uh*U*uh';
94 L2NormSquared_uh = H1NormSquared_uh - uh*A*uh';
95 H1NormSquared_ph = ph*U*ph';
96 L2NormSquared_ph = H1NormSquared_ph - ph*A*ph';
97
98 % Compute total errors:
99 errors = [sqrt(abs(L2NormSquared_f-2*L2SP_f_fh+L2NormSquared_fh)),...
100    sqrt(abs(L2NormSquared_u-2*L2SP_u_uh+L2NormSquared_uh)),...
101    sqrt(abs(H1NormSquared_u-2*H1SP_u_uh+H1NormSquared_uh)),...
102    sqrt(abs(L2NormSquared_p-2*L2SP_p_ph+L2NormSquared_ph)),...
103    sqrt(abs(H1NormSquared_p-2*H1SP_p_ph+H1NormSquared_ph))];
104
105 end

```

Listing 63: defineFUP.m

```

1 function [f,u,p] = defineFUP(kappa)
2 % Computes the exact polynomial solutions f,u,p for given kappa. The
3 % 'target' u0 will be chosen accordingly afterwards (see
4 % 'defineU0.m').
5 %
6 % X(x) := x*(1-x)
7 % Y(y) := y*(1-y)
8 % u(x,y) := X(x)^3*Y(y)^3 in H^1_0(Omega)
9 % f(x,y) := -Laplace(u(x,y)) in H^1_0(Omega)
10 % p(x,y) := kappa/(1-kappa)*f(x,y) in H^1_0(Omega)
11
12 X = [0,1,-1]';
13 Y = [0,1,-1]';
14 u = PolyPow(PolyMult(X,Y),3);
15 f = -PolyAdd(PolyDiff(PolyDiff(u,'x'),'x'),PolyDiff(PolyDiff(u,'y'),'y'));
16 p = kappa/(1-kappa)*f;
17
18 end

```

Listing 64: fIntegrals.m

```

1 function L2Integrals = fIntegrals(f,gF,numberofNodesPerSide)
2 % Computes the integrals int(f(FTijk(x,y))*p(x,y),x=0..1,y=0..1-x) for the
3 % exact solution f, all affine transformations FTijk and all Lagrange
4 % polynomials p. These values will be needed to compute the error
5 % ||f-fh||_L2.
6
7 % Define main mesh on Omega:
8 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
9    indicesIElements,indicesJElements,indicesKElements,...
10    indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...
11    idxTrafo.Elements.indexTFromIndexIJKT,idxTrafo.Elements.indexIJKTFromIndexT,...
    Elements_dim,...
12    idxTrafo.Nodes.indexNFromIndexIJN,idxTrafo.Nodes.indexIJNFromIndexN,Nodes_dim,...
13    idxTrafo.InnerNodes.indexNFromIndexIJN,idxTrafo.InnerNodes.indexIJNFromIndexN,...
    InnerNodes_dim] = defineMesh(numberofNodesPerSide);
14

```

```

15 % Compute Lagrange polynomials coefficients:
16 LagrangeCoeffsMatrix = LagrangeCoeffs(gF);
17
18 % Compute pascal and powers matrices:
19 [PascalMatrix.ATX,PowersMatrix.ATX] = PolyAffineTransform_1D_Matrices(size(f,1)-1);
20 [PascalMatrix.ATY,PowersMatrix.ATY] = PolyAffineTransform_1D_Matrices(size(f,2)-1);
21 MonomialIntegrals = PolyL1IntTRef_MonomialIntegrals(size(f,1),size(f,2));
22
23 % Compute integrals:
24 L2Integrals = zeros(Elements_dim,1);
25 for indexJT = indicesJElements
26     for indexIT = indicesIElements
27         for indexKT = indicesKElements
28             fAffineTransformed = PolyAffineTransform(f,...
29                 [1,0]*FTijkMat(indexIT,indexJT,indexKT)*[1;0],...
30                 [0,1]*FTijkMat(indexIT,indexJT,indexKT)*[0;1],...
31                 [1,0]*FTijkVect(indexIT,indexJT,indexKT),...
32                 [0,1]*FTijkVect(indexIT,indexJT,indexKT),...
33                 PascalMatrix.ATX,PowersMatrix.ATX,PascalMatrix.ATY,PowersMatrix.ATY);
34
35             indexT = idxTrafo.Elements.indexTFromIndexIJKT(indexIT,indexJT,indexKT);
36             L2Integrals(indexT) = PolyL2IntTRef(...
37                 fAffineTransformed,...
38                 LagrangeCoeffsMatrix(:, :, 1),...
39                 MonomialIntegrals...
40             );
41         end
42     end
43 end
44
45 end

```

Listing 65: upIntegrals.m

```

1 function [L2Integrals,H1SemiIntegrals] = upIntegrals(u,gU,numberOfNodesPerSide)
2 % Computes the integrals int(u(FTijk(x,y))*q(x,y),x=0..1,y=0..1-x) and
3 % int(u_{x/y}(FTijk(x,y))*q(x,y),x=0..1,y=0..1-x) for the exact
4 % solution u (p resp.), all affine transformations FTijk and all Lagrange
5 % polynomials q. These values will be needed to compute the errors
6 % ||u-uh||_L2 and ||u-uh||_H1 (||p-ph||_L2 and ||p-ph||_H1 resp.).
7
8 % Define main mesh on Omega:
9 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
10     indicesIElements,indicesJElements,indicesKElements,...
11     indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...
12     idxTrafo.Elements.indexTFromIndexIJKT,idxTrafo.Elements.indexIJKTFromIndexT,...
13     Elements_dim,...
14     idxTrafo.Nodes.indexNFromIndexIJN,idxTrafo.Nodes.indexIJNFromIndexN,Nodes_dim,...
15     idxTrafo.InnerNodes.indexNFromIndexIJN,idxTrafo.InnerNodes.indexIJNFromIndexN,...
16     InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
17
18 % Compute derivatives of u (p resp.):
19 uDiffX = PolyDiff(u,'x');
20 uDiffY = PolyDiff(u,'y');
21
22 % Compute Lagrange polynomials coefficients:
23 LagrangeCoeffsMatrix = LagrangeCoeffs(gU);
24 [LagrangeCoeffsDiffXMatrix,LagrangeCoeffsDiffYMatrix] = LagrangeCoeffsDiff(gU);
25
26 % Compute pascal and powers matrices:
27 [PascalMatrix.u.ATX,PowersMatrix.u.ATX] = PolyAffineTransform_1D_Matrices(size(u,1)-1);
28 [PascalMatrix.u.ATY,PowersMatrix.u.ATY] = PolyAffineTransform_1D_Matrices(size(u,2)-1);
29 [PascalMatrix.uDiffX.ATX,PowersMatrix.uDiffX.ATX] = PolyAffineTransform_1D_Matrices(...
30     size(uDiffX,1)-1);
31 [PascalMatrix.uDiffX.ATY,PowersMatrix.uDiffX.ATY] = PolyAffineTransform_1D_Matrices(...
32     size(uDiffX,2)-1);
33 [PascalMatrix.uDiffY.ATX,PowersMatrix.uDiffY.ATX] = PolyAffineTransform_1D_Matrices(...
34     size(uDiffY,1)-1);

```

```

30 [PascalMatrix.uDiffY_ATY, PowersMatrix.uDiffY_ATY] = PolyAffineTransform1D_Matrices(...
    size(uDiffY,2)-1);
31 MonomialIntegrals_U = PolyL1IntTRef_MonomialIntegrals(size(u,1),size(u,2));
32
33 % Compute integrals:
34 L2Integrals = zeros(Elements.dim,3);
35 H1SemiIntegrals = zeros(Elements.dim,3);
36 for indexJT = indicesJElements
37     for indexIT = indicesIElements
38         for indexKT = indicesKElements
39             FTijkMat11 = [1,0]*FTijkMat(indexIT,indexJT,indexKT)*[1;0];
40             FTijkMat22 = [0,1]*FTijkMat(indexIT,indexJT,indexKT)*[0;1];
41             FTijkVect1 = [1,0]*FTijkVect(indexIT,indexJT,indexKT);
42             FTijkVect2 = [0,1]*FTijkVect(indexIT,indexJT,indexKT);
43             FTijkInvMatrix = FTijkInvMat(indexIT,indexJT,indexKT)';
44
45             uAffineTransformed = PolyAffineTransform(u,FTijkMat11,FTijkMat22,FTijkVect1...
                ,FTijkVect2,...
46                 PascalMatrix.u_ATX, PowersMatrix.u_ATX, PascalMatrix.u_ATY, ...
                    PowersMatrix.u_ATY);
47             uDiffXAffineTransformed = PolyAffineTransform(uDiffX,FTijkMat11,FTijkMat22,...
                FTijkVect1,FTijkVect2,...
48                 PascalMatrix.uDiffX_ATX, PowersMatrix.uDiffX_ATX, PascalMatrix.uDiffX_ATY...
                    , PowersMatrix.uDiffX_ATY);
49             uDiffYAffineTransformed = PolyAffineTransform(uDiffY,FTijkMat11,FTijkMat22,...
                FTijkVect1,FTijkVect2,...
50                 PascalMatrix.uDiffY_ATX, PowersMatrix.uDiffY_ATX, PascalMatrix.uDiffY_ATY...
                    , PowersMatrix.uDiffY_ATY);
51
52             indexT = idxTrafo.Elements.indexTFromIndexIJKT(indexIT,indexJT,indexKT);
53             for iN = 1:1:3
54                 L2Integrals(indexT,iN) = PolyL2IntTRef(uAffineTransformed,...
                    LagrangeCoeffsMatrix(:, :, iN), MonomialIntegrals_U);
55
56                 H1SemiIntegrals(indexT,iN) = FTijkInvMatrix(1,1)*PolyL2IntTRef(...
                    uDiffXAffineTransformed, LagrangeCoeffsDiffXMatrix(:, :, iN), ...
                    MonomialIntegrals_U)...
57                 + FTijkInvMatrix(1,2)*PolyL2IntTRef(uDiffXAffineTransformed, ...
                    LagrangeCoeffsDiffYMatrix(:, :, iN), MonomialIntegrals_U)...
58                 + FTijkInvMatrix(2,1)*PolyL2IntTRef(uDiffYAffineTransformed, ...
                    LagrangeCoeffsDiffXMatrix(:, :, iN), MonomialIntegrals_U)...
59                 + FTijkInvMatrix(2,2)*PolyL2IntTRef(uDiffYAffineTransformed, ...
                    LagrangeCoeffsDiffYMatrix(:, :, iN), MonomialIntegrals_U);
60             end
61         end
62     end
63 end
64
65 end

```

Listing 66: computeU0Values_exactSolutions.m

```

1 function u0Values = computeU0Values_exactSolutions(u0,gu0,numberofNodesPerSide)
2 % Computes the values of u0 at all nodes of mesh on Omega (including all Lagrange
3 % nodes). The values could also be computed 'on the fly' when they are
4 % needed later on, but this would result in bad computation time. This is
5 % mainly a MATLAB issue and is bypassed by means of vectorization. The
6 % geometry of Omega is exploited massively.
7
8 % Define Lagrange mesh on reference element:
9 [LagrangeNodesX,LagrangeNodesY,I_gu0,...
10     idxTrafo.LagrangeNodes_indexLNFromIndexIJLN,...
    idxTrafo.LagrangeNodes_indexIJLNFromIndexLN,LagrangeNodes_dim] = ...
    defineLagrangeMesh(gu0);
11
12 % Define main mesh on Omega:
13 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
14     indicesIElements,indicesJElements,indicesKElements,...
15     indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...

```

```

16     idxTrafo.Elements_indexTFromIndexIJKT,idxTrafo.Elements_indexIJKTFromIndexT,...
        Elements_dim,...
17     idxTrafo.Nodes_indexNFromIndexIJN,idxTrafo.Nodes_indexIJNFromIndexN,Nodes_dim,...
18     idxTrafo.InnerNodes_indexNFromIndexIJN,idxTrafo.InnerNodes_indexIJNFromIndexN,...
        InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
19
20 % Store components of u0 in local variables:
21 u = u0.u;
22 q = u0.q;
23 M = u0.M;
24 Coeffs = u0.Coeffs;
25 Alphas = u0.Alphas;
26
27 % Compute u0Values (vectorized):
28 n = (numberOfNodesPerSide-1)*gu0+1;
29 xs = 1/(n-1)*([1:1:n]-1);
30 ys = 1/(n-1)*([1:1:n]-1);
31
32 uValues = zeros(n,n);
33 for i = 0:1:(size(u,1)-1)
34     for j = 0:1:(size(u,2)-1)
35         uValues = uValues + u(i+1,j+1)*(xs.^i)*(ys.^j);    % Monomials support ...
            vectorization.
36     end
37 end
38
39 vValues = zeros(n,n);
40 for i = 0:1:(size(q,1)-1)
41     for j = 0:1:(size(q,2)-1)
42         for m = 1:1:M
43             for n = 1:1:M
44                 if (q(i+1,j+1)~=0) && (Alphas(i+1,m)~=0) && (Alphas(j+1,n)~=0)
45                     vValues = vValues + q(i+1,j+1)*Coeffs(m,n)*Alphas(i+1,m)*Alphas(j...
                        +1,n)*4*sin(pi*m*xs)*sin(pi*n*ys);    % Builtin function 'sin'...
                        supports vectorization.
46                 end
47             end
48         end
49     end
50 end
51
52 % Store the values in other format (for easier use later):
53 u0Values = zeros(Elements_dim,LagrangeNodes_dim);
54 u0ValueHasBeenSet = zeros(Elements_dim,LagrangeNodes_dim);
55 for indexJT = indicesJElements
56     for indexIT = indicesIElements
57         for indexKT = indicesKElements
58             indexT = idxTrafo.Elements_indexTFromIndexIJKT(indexIT,indexJT,indexKT);
59             for indexJLN = 1:gu0
60                 for indexILN = 0:1:(gu0-indexJLN)
61                     indexLN = idxTrafo.LagrangeNodes_indexLNFromIndexIJLN(indexILN+1,...
                        indexJLN+1);
62                     if ~u0ValueHasBeenSet(indexT,indexLN)
63                         switch indexKT
64                             case 1
65                                 i = (indexIT-1)*gu0 + 1 + indexILN;
66                                 j = (indexJT-1)*gu0 + 1 + indexJLN;
67                             case 2
68                                 i = (indexIT-1)*gu0 + 1 + (gu0-indexILN);
69                                 j = (indexJT-1)*gu0 + 1 + (gu0-indexJLN);
70                             end
71                     u0Values(indexT,indexLN) = uValues(i,j) + vValues(i,j);
72                     u0ValueHasBeenSet(indexT,indexLN) = true;
73                 end
74             end
75         end
76     end
77 end
78 end
79

```

Listing 67: defineU0.m

```

1 function u0 = defineU0(u,p)
2 % Defines the representing components u,q,M,Coeffs,Alphas of the 'target' function u0.
3 %
4 % Note 1:
5 % With
6 %  $q(x,y) := -\text{Laplace}(p(x,y))$ ,
7 %  $\lambda_{mn} := \pi^2(m^2+n^2)$  and
8 %  $e_{mn}(x,y) := 2\sin(\pi m x)\sin(\pi n y)$ ,
9 % define the 'target' function
10 %
11 %  $u_0(x,y) := u(x,y) + \sum_{m,n=1}^{\infty} \frac{1}{(1+\lambda_{mn})} \langle q, e_{mn} \rangle_{L2} e_{mn}(x,y)$ .
12 %
13 % Then the polynomials f,u,p from 'defineFUP.m' are the exact solutions of the optimal ...
    control
14 % problem (Laplace equation) with data u0. (The infinite sum in the
15 % definition of u0 is approximated (!) by the according finite sum with
16 % upper index bound  $M < \infty$ .)
17 %
18 % Note 2:
19 % The integrals  $\alpha_i := \int_0^1 x^i \sin(\pi x) dx$  are needed to compute
20 % the Fourier coefficients  $\langle q, e_{mn} \rangle_{L2}$  and satisfy the following recursion:
21 %  $\alpha_0 = (1-(-1)^m)/(\pi m)$ ,
22 %  $\alpha_1 = -(-1)^m/(\pi m)$ ,
23 %  $\alpha_{i+1} = -i(i-1)/(\pi^2 m^2) \alpha_{i-1} - (-1)^m/(\pi m)$ .
24 %
25 % Compute the components:
26 q = -PolyAdd(PolyDiff(PolyDiff(p, 'x'), 'x'), PolyDiff(PolyDiff(p, 'y'), 'y'));
27
28 M = 40;
29
30 Coeffs = zeros(M,M);
31 for m = 1:1:M
32     for n = 1:1:M
33         Coeffs(m,n) = 1/(1+pi^2*(m^2+n^2));
34     end
35 end
36
37 g = size(q,1)-1;
38 AlphasRecursion = @(i,m,alpha) -i*(i-1)/(pi^2*m^2)*alpha - (-1)^m/(pi*m);
39 Alphas = zeros(g+1,M);
40 for m = 1:1:M
41     Alphas(1,m) = (1-(-1)^m)/(pi*m);
42     Alphas(2,m) = -(-1)^m/(pi*m);
43     for i = 2:1:g
44         Alphas(i+1,m) = AlphasRecursion(i,m,Alphas(i-1,m));
45     end
46 end
47
48 % Store the components in a structure:
49 u0.u = u;
50 u0.q = q;
51 u0.M = M;
52 u0.Coeffs = Coeffs;
53 u0.Alphas = Alphas;
54
55 end

```

Listing 68: plotConditionNumbers.m

```

1 function figureHandle = plotConditionNumbers(kappa,ku0,gF,gU,mMin,mMax,conditionNumbers...
    )
2 % Plots the condition numbers of the system matrices for all values of h.
3

```

```

4 % Define plot specifiers:
5 colorConditionNumbers = 'magenta';
6 linestyleConditionNumbers = '-';
7 linewidthConditionNumbers = 2;
8 colorReferenceLine = 'black';
9 linestyleReferenceLine = ':';
10 linewidthReferenceLine = 1;
11
12 % Plot:
13 figureHandle = figure;
14 hold on;
15 plot(mMin:1:mMax, log10(conditionNumbers), 'color', colorConditionNumbers, 'linestyle', ...
        linestyleConditionNumbers, 'linewidth', linewidthConditionNumbers);
16 referenceLineGradient = 2;
17 referenceLineOffset = 3.5;
18 plot(mMin:1:mMax, log10(2.^[mMin:1:mMax]*referenceLineGradient))+referenceLineOffset, '...
        color', colorReferenceLine, 'linestyle', linestyleReferenceLine, 'linewidth', ...
        linewidthReferenceLine);
19
20 % Set xlabel, legend, title:
21 xlabel('m');
22 legend('log_{10}(cond(systemMatrix_h))', ['Reference: O(h^{', num2str(-...
        referenceLineGradient), '})'], 'Location', 'northwest');
23 box on;
24 title(['\kappa = ', num2str(kappa), ...
        ', k(u_0) = ', num2str(ku0), ...
        ', g_F = ', num2str(gF), ...
        ', g_U = ', num2str(gU), ...
        ', g_P = ', num2str(gU), ...
        ', h = 2^{-m}']);
25
26
27
28
29
30
31 end

```

Listing 69: plotErrors.m

```

1 function figureHandle = plotErrors(kappa, ku0, gF, gU, gu0, mMin, mMax, errors)
2 % Plots the L2- and H1-errors between approximate and exact solutions for
3 % all values of h.
4
5 % Define plot specifiers:
6 colorF = 'blue';
7 colorU = 'red';
8 colorP = 'green';
9 linestyleFUPWeak = '--';
10 linestyleFUPStrong = '-';
11 linewidthFUPWeak = 2;
12 linewidthFUPStrong = 2;
13 colorReferenceLines = 'black';
14 linestyleReferenceLines = ':';
15 linewidthReferenceLines = 1;
16
17 % Plot:
18 figureHandle = figure;
19 hold on;
20 plot(mMin:1:mMax, log10(errors(:,1)), 'color', colorF, 'linestyle', linestyleFUPStrong, '...
        linewidth', linewidthFUPStrong);
21 plot(mMin:1:mMax, log10(errors(:,2)), 'color', colorU, 'linestyle', linestyleFUPWeak, '...
        linewidth', linewidthFUPWeak);
22 plot(mMin:1:mMax, log10(errors(:,3)), 'color', colorU, 'linestyle', linestyleFUPStrong, '...
        linewidth', linewidthFUPStrong);
23 plot(mMin:1:mMax, log10(errors(:,4)), 'color', colorP, 'linestyle', linestyleFUPWeak, '...
        linewidth', linewidthFUPWeak);
24 plot(mMin:1:mMax, log10(errors(:,5)), 'color', colorP, 'linestyle', linestyleFUPStrong, '...
        linewidth', linewidthFUPStrong);
25
26 referenceLinesGradients = [-1, -2];
27 referenceLinesOffsets = [-2.5, -3.75];
28 for i = 1:length(referenceLinesGradients)
29     plot(mMin:1:mMax, log10(2.^[mMin:1:mMax]*referenceLinesGradients(i))+...

```

```

        referenceLinesOffsets(i), 'color', colorReferenceLines, 'linestyle', ...
        linestyleReferenceLines, 'linewidth', linewidthReferenceLines);
30 end
31
32 % Set xlabel, legend, title:
33 xlabel('m');
34
35 legendStrings = cell(5+length(referenceLinesGradients),1);
36 legendStrings(1:5) = {'log- $\{10\}$  ( $\|f-f_h\|$  -  $L^2(\Omega)$ )', ...
37     'log- $\{10\}$  ( $\|u-u_h\|$  -  $L^2(\Omega)$ )', ...
38     'log- $\{10\}$  ( $\|u-u_h\|$  -  $H^1(\Omega)$ )', ...
39     'log- $\{10\}$  ( $\|p-p_h\|$  -  $L^2(\Omega)$ )', ...
40     'log- $\{10\}$  ( $\|p-p_h\|$  -  $H^1(\Omega)$ )'};
41 for i = 1:length(referenceLinesGradients)
42     legendStrings{i+5} = ['Reference:  $O(h^{\{$ ', num2str(-referenceLinesGradients(i)), ' $\})$  '...
43         '];
44 end
45 legend(legendStrings, 'Location', 'southwest');
46 box on;
47
48 title(['\kappa = ', num2str(kappa), ...
49     ', k(u_0) = ', num2str(ku0), ...
50     ', g_F = ', num2str(gF), ...
51     ', g_U = ', num2str(gU), ...
52     ', g_P = ', num2str(gU), ...
53     ', g(u_0) = ', num2str(gu0), ...
54     ', h = 2^{-m}']);
55 end

```

Listing 70: errorAnalysis.m

```

1 function errorAnalysis
2 % Computes approximate solutions fh,uh,ph to optimal control problem (Laplace equation)
3 % for different values of the mesh-size h -> 0. The L2- and H1-errors between
4 % the approximate solutions fh,uh,ph and known exact solutions f,u,p (for a
5 % specific 'target' u0) are computed for each value of h. The 'target' u0
6 % is chosen in a way that allows f,u,p to be polynomials on the entire
7 % domain Omega. This makes exact (!) error computation accessible.
8 %
9 % Inputs: None.
10 %
11 % Outputs:
12 % 1) Plot 1: A plot showing the approximate solutions fh,uh,ph and u0 for
13 % one value of h.
14 % 2) Plot 2: A plot showing the L2- and H1-errors between approximate and
15 % exact solutions for each value of h.
16 % 3) Plot 3: A plot showing the condition number of the system matrix for
17 % each value of h.
18
19 totalTimeStart = tic;
20
21 % Equation parameters and exact solutions:
22 kappa = 1/100;
23 [f,u,p] = defineFUP(kappa); % Exact polynomial solutions.
24 u0 = defineU0(u,p); % According 'target' function.
25
26 % Discretization parameters:
27 ku0 = 2;
28 gF = ku0-2;
29 gU = ku0-1;
30 gu0 = gU+1;
31
32 % Main loop:
33 mMin = 1;
34 mMax = 7;
35
36 errors = zeros(mMax-mMin+1,5);
37 conditionNumbers = zeros(mMax-mMin+1,1);

```

```

38 for m = mMin:1:mMax
39     iterationTimeStart = tic;
40     numberOfNodesPerSide = 2^m+1; % Number of nodes on each of both sides of the ...
        domain Omega.
41
42     % Compute u0Values:
43     u0Values = computeU0Values_exactSolutions(u0,gu0,numberOfNodesPerSide);
44
45     % Assemble system matrix (F,U,A,B) and load vector (U0):
46     [F,U,A,B,U0] = assemble(u0Values,kappa,gF,gU,gu0,numberOfNodesPerSide);
47
48     % Solve the LSE:
49     [fh,uh,ph,conditionNumber] = solve(F,U,A,B,U0);
50
51     % Compute the errors:
52     conditionNumbers(m-mMin+1) = conditionNumber;
53     errors(m-mMin+1,:) = computeErrors(kappa,f,u,p,gF,gU,numberOfNodesPerSide,fh,uh,ph,...
        F,U,A);
54
55     % Plot the solutions:
56     if m==5
57         figureHandle_solutions = plotFhUhU0Ph(u0Values,kappa,ku0,gF,gU,gu0,...
            numberOfNodesPerSide,fh,uh,ph);
58     end
59
60     % Display iteration run time:
61     display(['Elements: ',num2str((numberOfNodesPerSide-1)^2*2),' , time: ',num2str(toc(...
        iterationTimeStart)),' s.']);
62 end
63
64 % Compute and plot errors convergence rates:
65 figureHandle_errors = plotErrors(kappa,ku0,gF,gU,gu0,mMin,mMax,errors);
66
67 % Compute and plot condition number rate:
68 figureHandle_condNs = plotConditionNumbers(kappa,ku0,gF,gU,mMin,mMax,conditionNumbers);
69
70 % Display total run time:
71 display(['Total time: ',num2str(toc(totalTimeStart)),' s.']);
72
73 % Save results:
74 hgexport(figureHandle_solutions,[fileparts(which('errorAnalysis.m')),'\Results\...
        solutionsPlot'],defineExportStyle('png',15,8,600));
75 hgexport(figureHandle_errors,[fileparts(which('errorAnalysis.m')),'\Results\errorsPlot'...
        ],defineExportStyle('png',15,10,600));
76 hgexport(figureHandle_condNs,[fileparts(which('errorAnalysis.m')),'\Results\condNsPlot'...
        ],defineExportStyle('png',15,10,600));
77
78 end

```

Listing 71: fadingColors.m

```

1 function colors = fadingColors(colorChooser,numberOfShades,fadingParameterMin,...
        fadingParameterMax)
2 % Computes the RGB-values of different shades of a given color (red, green
3 % or blue) and stores them rowwise in the matrix 'colors'.
4
5 switch colorChooser
6     case 'red'
7         colorFaderRed = @(s) min(max(2-2*s,0),1);
8         colorFaderGreen = @(s) min(max(1-2*s,0),1);
9         colorFaderBlue = @(s) min(max(1-2*s,0),1);
10    case 'green'
11        colorFaderRed = @(s) min(max(1-2*s,0),1);
12        colorFaderGreen = @(s) min(max(2-2*s,0),1);
13        colorFaderBlue = @(s) min(max(1-2*s,0),1);
14    case 'blue'
15        colorFaderRed = @(s) min(max(1-2*s,0),1);
16        colorFaderGreen = @(s) min(max(1-2*s,0),1);
17        colorFaderBlue = @(s) min(max(2-2*s,0),1);

```

```

18 end
19
20 colors= zeros(numberOfShades,3);
21 for i = 1:1:numberOfShades
22     fadingParameter = (1-(i-1)/(numberOfShades-1))*fadingParameterMin + (i-1)/(...
        numberOfShades-1)*fadingParameterMax;
23     colors(i,:) = [colorFaderRed(fadingParameter),colorFaderGreen(fadingParameter),...
        colorFaderBlue(fadingParameter)];
24 end
25
26 end

```

Listing 72: plotFhsUhsU0Phs.m

```

1 function figureHandle = plotFhsUhsU0Phs(u0Values,kappas,ku0,gF,gU,gu0,...
    numberOfNodesPerSide,fhs,uhs,phs)
2 % Sets up a figure with three subplots containing the plots of fh,uh,u0 and
3 % ph.
4 % Subplot 1: Contains the solutions fh for all values of kappa.
5 % Subplot 2: Contains the solutions uh for all values of kappa and the
6 % 'target' u0.
7 % Subplot 3: Contains the solutions ph for all values of kappa.
8
9 % Define main mesh on Omega:
10 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
11     indicesIElements,indicesJElements,indicesKElements,...
12     indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...
13     idxTrafo.Elements_indexTFromIndexIJKT,idxTrafo.Elements_indexIJKTFromIndexT,...
        Elements_dim,...
14     idxTrafo.Nodes_indexNFromIndexIJN,idxTrafo.Nodes_indexIJNFromIndexN,Nodes_dim,...
15     idxTrafo.InnerNodes_indexNFromIndexIJN,idxTrafo.InnerNodes_indexIJNFromIndexN,...
        InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
16
17 % Define plot specifiers:
18 viewSpecifier = [0,0];
19 faceAlpha = 0.2;
20 edgeAlpha = 0.5;
21
22 colorFaderMin = 0.8;
23 colorFaderMax = 0.2;
24
25 faceColorsF = fadingColors('blue',length(kappas),colorFaderMin,colorFaderMax);
26 faceAlphaF = faceAlpha;
27 edgeColorsF = fadingColors('blue',length(kappas),colorFaderMin,colorFaderMax);
28 edgeAlphaF = edgeAlpha;
29
30 faceColorsU = fadingColors('red',length(kappas),colorFaderMin,colorFaderMax);
31 faceAlphaU = faceAlpha;
32 edgeColorsU = fadingColors('red',length(kappas),colorFaderMin,colorFaderMax);
33 edgeAlphaU = edgeAlpha;
34
35 faceColorsP = fadingColors('green',length(kappas),colorFaderMin,colorFaderMax);
36 faceAlphaP = faceAlpha;
37 edgeColorsP = fadingColors('green',length(kappas),colorFaderMin,colorFaderMax);
38 edgeAlphaP = edgeAlpha;
39
40 faceColorU0 = 0.7*[1,1,1];
41 faceAlphaU0 = faceAlpha;
42 edgeColorU0 = 0.7*[1,1,1];
43 edgeAlphaU0 = edgeAlpha;
44
45 % Plot:
46 figureHandle = figure;
47
48 % Subplot 1:
49 subplot(1,3,1);
50 plotHandles = zeros(1,length(kappas));
51 legendStrings = cell(length(kappas),1);
52 for i = 1:1:length(kappas)

```

```

53     [xData,yData,fhData,xVertData,yVertData,fhVertData] = computePlotDataFh(...
        numberOfNodesPerSide,fhs(i,:));
54     plotHandles(i) = patch(xData,yData,fhData,faceColorsF(i,:), 'FaceAlpha',faceAlphaF,...
        EdgeColor',edgeColorsF(i,:), 'EdgeAlpha',edgeAlphaF);
55     patch(xVertData,yVertData,fhVertData,faceColorsF(i,:), 'FaceAlpha',faceAlphaF,...
        EdgeColor',edgeColorsF(i,:), 'EdgeAlpha',edgeAlphaF);
56     legendStrings{i} = ['f_h(x,y) (\kappa = ',num2str(kappas(i)),')'];
57 end
58 view(viewSpecifier);
59 xlabel('x');
60 ylabel('y');
61 legend(plotHandles,legendStrings,'Location','northoutside');
62 box on;
63
64 % Subplot 2:
65 subplot(1,3,2);
66 legendStrings = cell(length(kappas)+1,1);
67 for i = 1:length(kappas)
68     [xData,yData,uhData] = computePlotDataUhPh(numberOfNodesPerSide,uhs(i,:));
69     patch(xData,yData,uhData,faceColorsU(i,:), 'FaceAlpha',faceAlphaU, 'EdgeColor',...
        edgeColorsU(i,:), 'EdgeAlpha',edgeAlphaU);
70     legendStrings{i} = ['u_h(x,y) (\kappa = ',num2str(kappas(i)),')'];
71 end
72 [xData,yData,u0Data] = computePlotDataU0(gu0,numberOfNodesPerSide,u0Values);
73 patch(xData,yData,u0Data,faceColorU0, 'FaceAlpha',faceAlphaU0, 'EdgeColor',edgeColorU0,...
        EdgeAlpha',edgeAlphaU0);
74 legendStrings{end} = 'u_0(x,y) (\kappa = 0)';
75 view(viewSpecifier);
76 xlabel('x');
77 ylabel('y');
78 legend(legendStrings,'Location','northoutside');
79 box on;
80
81 % Subplot 3:
82 subplot(1,3,3);
83 legendStrings = cell(length(kappas),1);
84 for i = 1:length(kappas)
85     [xData,yData,phData] = computePlotDataUhPh(numberOfNodesPerSide,phs(i,:));
86     patch(xData,yData,phData,faceColorsP(i,:), 'FaceAlpha',faceAlphaP, 'EdgeColor',...
        edgeColorsP(i,:), 'EdgeAlpha',edgeAlphaP);
87     legendStrings{i} = ['p_h(x,y) (\kappa = ',num2str(kappas(i)),')'];
88 end
89 view(viewSpecifier);
90 xlabel('x');
91 ylabel('y');
92 legend(legendStrings,'Location','northoutside');
93 box on;
94
95 % Title:
96 titleString = ['k(u_0) = ',num2str(ku0),...
97     ', g_F = ',num2str(gF),...
98     ', g_U = ',num2str(gU),...
99     ', g_P = ',num2str(gU),...
100     ', g(u_0) = ',num2str(gu0),...
101     ', #Elements: ',num2str(Elements_dim)];
102 annotation('textbox',[0,0.94,1,0.06],...
103     'String',titleString,...
104     'EdgeColor','none',...
105     'HorizontalAlignment','center',...
106     'VerticalAlignment','bottom');
107
108 end

```

Listing 73: kappaComparison.m

```

1 function kappaComparison
2 % Computes approximate solutions fh,uh,ph to optimal control problem (Laplace equation)
3 % for different values of kappa and plots them together in one plot. This
4 % illustrates the role of the parameter kappa in the control problem.

```

```

5 %
6 % Inputs: None.
7 %
8 % Outputs: A plot showing the approximate solutions fh,uh,ph and the
9 % 'target' u0 for different values of kappa.
10
11 % Equation parameters:
12 aX = 0.2;
13 bX = 0.5;
14 cX = 0.7;
15 aY = 0.4;
16 bY = 0.5;
17 cY = 0.6;
18 u0 = @(x,y) (((aX<=x) & (x<=bX)) .* (x-aX) / (bX-aX) + ((bX<x) & (x<=cX)) .* ((bX-x) / (cX-bX)+1))'...
19     ...
20     * (((aY<=y) & (y<=bY)) .* (y-aY) / (bY-aY) + ((bY<y) & (y<=cY)) .* ((bY-y) / (cY-bY)+1));
21
22 % Discretization parameters:
23 ku0 = 2;
24 gF = ku0-2;
25 gU = ku0-1;
26 gu0 = gU+1;
27 numberOfNodesPerSide = 2^5+1; % Number of nodes on each of both sides of the domain ...
28     Omega.
29
30 % Define main mesh on Omega:
31 [h,NodesX,NodesY,FTijkMat,FTijkVect,FTijkInvMat,FTijkInvVect,...
32     indicesIElements,indicesJElements,indicesKElements,...
33     indicesIAllNodes,indicesJAllNodes,indicesIInnerNodes,indicesJInnerNodes,...
34     idxTrafo.Elements_indexTFromIndexIJKT,idxTrafo.Elements_indexIJKTFromIndexT,...
35     Elements_dim,...
36     idxTrafo.Nodes_indexNFromIndexIJN,idxTrafo.Nodes_indexIJNFromIndexN,Nodes_dim,...
37     idxTrafo.InnerNodes_indexNFromIndexIJN,idxTrafo.InnerNodes_indexIJNFromIndexN,...
38     InnerNodes_dim] = defineMesh(numberOfNodesPerSide);
39
40 % Compute values of u0 at all nodes of mesh (including all Lagrange nodes):
41 u0Values = computeU0Values(u0,gu0,numberOfNodesPerSide);
42
43 % Main loop:
44 kappas = [1/50,1/500,1/5000];
45 fhs = zeros(length(kappas),Elements_dim);
46 uhs = zeros(length(kappas),InnerNodes_dim);
47 phs = zeros(length(kappas),InnerNodes_dim);
48 for i = 1:length(kappas)
49     % Assemble system matrix (F,U,A,B) and load vector (U0):
50     [F,U,A,B,U0] = assemble(u0Values,kappas(i),gF,gU,gu0,numberOfNodesPerSide);
51
52     % Solve the LSE:
53     [fh,uh,ph,~] = solve(F,U,A,B,U0);
54
55     % Store solutions in table:
56     fhs(i,:) = fh;
57     uhs(i,:) = uh;
58     phs(i,:) = ph;
59 end
60
61 % Plot the solutions:
62 figureHandle = plotFhsUhsU0Phs(u0Values,kappas,ku0,gF,gU,gu0,numberOfNodesPerSide,fhs,...
63     uhs,phs);
64
65 % Save results:
66 hgexport(figureHandle,[fileparts(which('kappaComparison.m')),'\Results\solutionsPlot'],...
67     defineExportStyle('png',15,8,600));
68
69 end

```

Literatur

- [Braess, 2013] Braess, D. (2013). *Finite Elemente. Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Berlin: Springer Spektrum, 5th revised ed. edition.
- [Brezzi and Fortin, 1991] Brezzi, F. and Fortin, M. (1991). *Mixed and hybrid finite element methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York.
- [Grisvard, 1985] Grisvard, P. (1985). *Elliptic problems in nonsmooth domains*, volume 24 of *Monographs and Studies in Mathematics*. Pitman (Advanced Publishing Program), Boston, MA.
- [Hinze et al., 2009] Hinze, M., Pinnau, R., Ulbrich, M., and Ulbrich, S. (2009). *Optimization with PDE constraints*, volume 23 of *Mathematical Modelling: Theory and Applications*. Springer, New York.
- [Lions, 1971] Lions, J.-L. (1971). *Optimal control of systems governed by partial differential equations*. Die Grundlehren der mathematischen Wissenschaften, Band 170. Springer-Verlag, New York-Berlin.
- [Tröltzsch, 2009] Tröltzsch, F. (2009). *Optimale Steuerung partieller Differentialgleichungen. Theorie, Verfahren und Anwendungen*. Wiesbaden: Vieweg+Teubner, 2nd revised ed. edition.
- [Tröltzsch, 2010] Tröltzsch, F. (2010). On finite element error estimates for optimal control problems with elliptic pdes. In *Large-Scale Scientific Computing*, volume 5910 of *Lecture Notes in Computer Science*, pages 40–53. Springer Berlin Heidelberg.