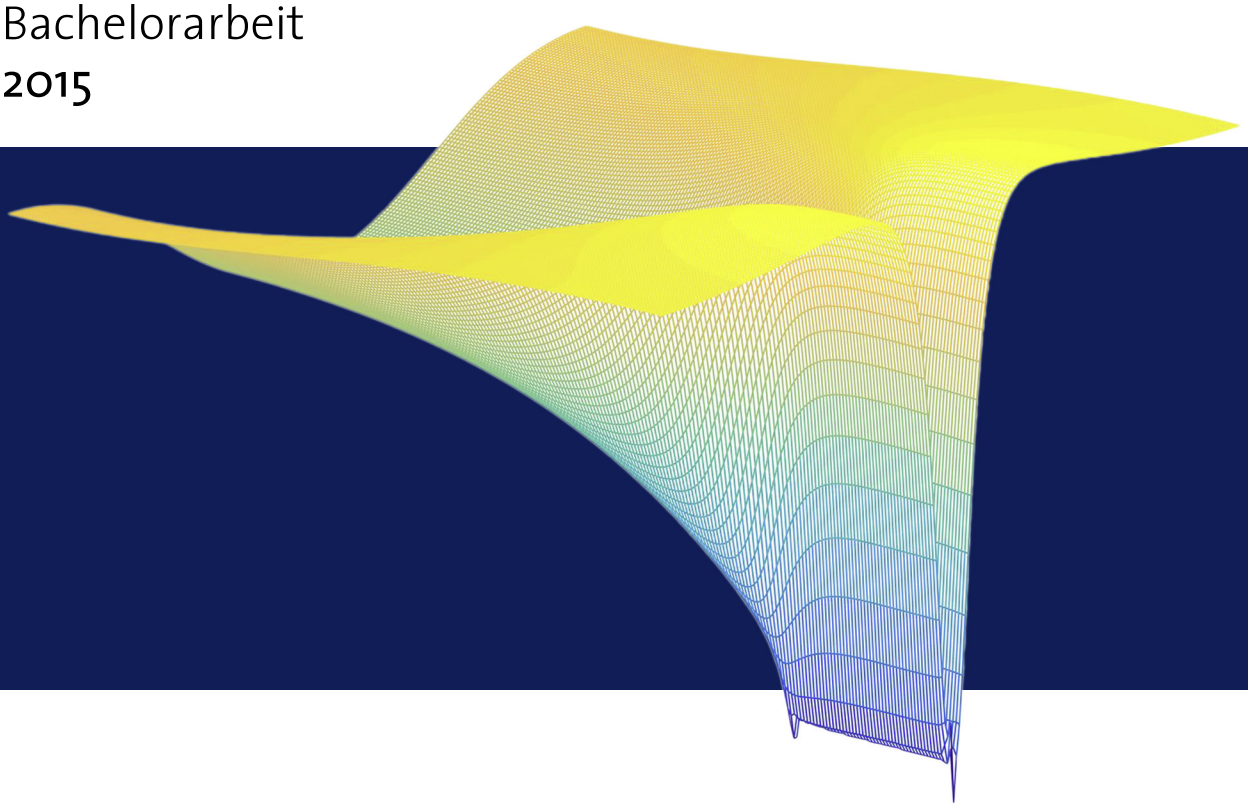
Westfälische
Wilhelms-Universität
Münster

Lösung eines Phasenfeld Modells für Rissentstehung mittels semiglatter Newton Methode

Ines Ahrens

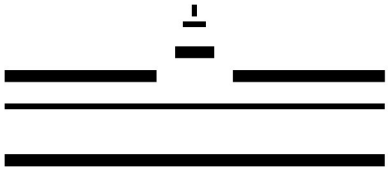
Bachelorarbeit

2015



wissen.leben
WWU Münster

Institut für Numerische und
Angewandte Mathematik
Westfälische Wilhelms-Universität
Münster



Westfälische
Wilhelms-Universität
Münster

Lösung eines Phasenfeld Modells für Rissentstehung mittels semiglatter Newton Methode

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science

Westfälische Wilhelms-Universität Münster
Fachbereich Mathematik und Informatik
Institut für Numerische und Angewandte Mathematik

Betreuung:
Prof. Dr. Benedikt Wirth

Eingereicht von:
Ines Ahrens

Münster, September 2015

Inhaltsverzeichnis

Eidesstattliche Erklärung

Hiermit versichere ich, *Ines Ahrens*, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Gedanklich, inhaltlich oder wörtlich übernommenes habe ich durch Angabe von Herkunft und Text oder Anmerkung belegt bzw. kenntlich gemacht. Dies gilt in gleicher Weise für Bilder, Tabellen, Zeichnungen und Skizzen, die nicht von mir selbst erstellt wurden.

Alle auf der CD beigefügten Programme sind von mir selbst programmiert worden.

Berlin, 30. September 2015

Ines Ahrens

1. Einleitung

Ein Material wird an zwei Seiten eingespannt und Kraft wird an diesen Seiten ausgeübt. Die daraus resultierende Verzerrung und der Riss im Material wird durch ein Optimierungsproblem modelliert. In dieser Arbeit befasse ich mich mit der Lösung dieses Optimierungsproblems.

Betrachten wir einführend ein Beispiel. Ein Blatt Papier befindet sich in Ruheposition auf dem Gebiet $\Omega \subset \mathbb{R}^3$. Nun wird es an zwei gegenüberliegenden Seiten eingespannt und an diesen Seiten wird gezogen. Zunächst dehnt sich das Blatt Papier ein wenig. Diese Verschiebung des Papiers gibt $u : \Omega \to \mathbb{R}^2$ an. Wenn genügend Kräfte auf dem Papier wirken, reißt es. Dieser Riss wird durch die Abbildung $v : \Omega \to \mathbb{R}$ modelliert. v ist 0, falls das Papier vollständig gerissen ist und 1, wenn kein Riss vorhanden ist.

Das folgende Minimierungsproblem beschreibt, wie der Riss im Papier und die Verschiebung des Gebietes modelliert werden.

$$\begin{aligned} &\min_{u\in H^1(\Omega)^2, v\in H^1(\Omega)} \int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 + \nu\left(\epsilon_2|\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\right) dx \\ &\text{s.d. } 0 \le v \le v_0 \\ &u = u_0 \text{ auf } \Gamma_1 \cup \Gamma_2 \end{aligned}$$

Dabei ist ν eine Konstante, $\epsilon_i > 0$ für alle $i \in \{1,2\}$ ein kleiner Parameter und $\Gamma_1, \Gamma_2 \subset \Omega$ der rechte bzw. linke Rand eines rechteckigen Gebietes $\Omega \subset \mathbb{R}^2$, d.h.,

$$\Omega = [0,a] \times [0,b] \;\; \Gamma_1 = \{0\} \times [0,b] \;\; \Gamma_2 = \{a\} \times [0,b].$$

Die Erklärung der genauen Modellierung erfolgt im Kapitel Modellierung 2.

Zur Lösung des Minimierungsproblems werden analytische und numerische Grundlagen benötigt. Für die analytische Betrachtung werden partielle Differentialgleichungen und Optimierung genutzt. Da die Lösung des Problems analytisch nicht zu finden ist, diskretisiere ich das Problem mit dreieckig linearen Lagrange Elementen und imple-

mentiere es mittels semiglatter Newton Methoden. Die Grundlagen dazu sind ebenfalls im zweiten Kapitel zu finden. Da jedes der eben genannten Themen sehr umfangreich ist, wird es mir nur möglich sein, die wichtigsten Begriffe einzuführen.

Die Untersuchung des Problems folgt in Kapitel drei. Beim ersten Betrachten fällt auf, dass die Optimierung nicht gleichzeitig nach u und v ausgeführt werden muss. Man kann zwei getrennte Optimierungsprobleme betrachten. Dieses wird im ersten Teil des dritten Kapitels erläutert. Im zweiten Teil wird die Optimierung nach u betrachtet. Zuerst wird die Existenz und Eindeutigkeit gesichert, um dann numerisch die Lösung mit dreieckig linearen Lagrange Elementen zu suchen. Die Optimierung nach v im dritten Teil hat den gleichen Aufbau. Nur ist hier das Problem komplizierter, da die Ableitung nicht berechnet werden kann; die numerische Lösung erfolgt mit der semiglatten Newton Methode. Im letzten Kapitel werte ich die numerischen Resultate aus und veranschauliche meine Ergebnisse anhand mehrerer Anwendungsbeispiele.

2. Modellierung

Erinnern wir uns an das Beispiel am Anfang: Ein Blatt Papier befindet sich in Ruheposition auf dem Gebiet $\Omega \subset \mathbb{R}^3$. Nun wird es an zwei gegenüberliegenden Seiten eingespannt und an diesen Seiten wird gezogen. Zunächst deformiert sich das Papier. Folgende Modellierung stammt aus [1]. Die Funktion $\varphi : \Omega \to \mathbb{R}^3$ beschreibt die neue Position des Papiers. Also gilt $\varphi = u + id$, wobei $u : \Omega \to \mathbb{R}^3$ die Verschiebung des Körpers ist.

Das Ziel ist es, ein φ zu finden, bei dem die Verzerrung des Papiers minimal ist. Dazu betrachten wir den Abstand zwischen $x \in \Omega$ und $x + h \in \Omega$ im aktuellen Zustand. Durch taylorn erhalten wir

$$\|\varphi(x+h) - \varphi(x)\|^2 = \|\nabla\varphi\|^2 + o(\|h\|^2) = h^T \nabla\varphi^T \nabla\varphi h + o(\|h\|^2).$$

Damit haben wir die Matrix $C := \nabla\varphi^T\nabla\varphi$ erhalten, die die lokale Änderung der Längen angibt und Cauchy-Grennscher Verzerrungstensor genannt wird. Die Verzerrung E lässt sich über $\frac{1}{2}(C - I)$ berechnen. Da $u = \varphi - \mathrm{id}$ gilt, können wir die Ableitung von u einsetzen und erhalten

$$E_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \frac{1}{2}\sum_k \frac{\partial u_i}{\partial x_k}\frac{\partial u_j}{\partial x_k}.$$

Da im Linearen die quadratischen Terme vernachlässigt werden können, erhalten wir

$$\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) = \frac{\nabla u^T + \nabla u}{2}$$

Irgendwann hält das Papier den auf es wirkenden Kräften nicht mehr stand und reißt. Sobald dies geschieht, gilt die Formel für die Verzerrung nicht mehr. Also brauchen wir einen Term, der angibt, wo und wie stark der Körper gerissen ist. Dazu definieren wir $v : \Omega \to \mathbb{R}$. v ist 0, falls der Körper vollständig gerissen ist und 1, wenn kein Riss

vorhanden ist. Multiplizieren wir also $\frac{1}{2}(\nabla u^T + \nabla u)$ mit $v^2 + \epsilon_1$. Es ergibt sich

$$\min_u \int_\Omega |\nabla u|^2 (v^2 + \epsilon_1)$$
$$u = u_0 \text{ auf } \Gamma_1 \cup \Gamma_2$$

Da wir u nur im zweidimensionalen betrachten, können wir $\frac{\nabla u + \nabla u^T}{2}^2$ zu $|\nabla u|^2$ vereinfachen. Am rechten bzw. linken Rand $\Gamma_1, \Gamma_2 \subset \Omega$ ist das Material befestigt, was $u = u_0$ aussagt.

Ideal wäre es, wenn der Körper niemals reißen würde. Da dies aber in der Anwendung nicht der Fall sein kann, möchten wir einen möglichst kleinen Riss erhalten. Die Oberfläche, auf die der Riss auftritt sollte also minimal sein. Wir wollen also, dass $\int_\Omega (1-v)^2$ minimal ist. Aber in der Realität stellt man fest, dass der Riss glatt ist. Um dies zu modellieren, addieren wir die Ableitung zum Quadrat mit ϵ_2^2 multipliziert hinzu. Daraus ergibt sich eine Darstellung vom Riss nach [2]

$$\epsilon_2 |\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2$$

Zusammengefasst ist das gesamte Problem gegeben durch

$$\min_{u \in H^1(\Omega)^2, v \in H^1(\Omega)} \int_\Omega \left(v^2 + \epsilon_1\right) |\nabla u|^2 + \nu \left(\epsilon_2 |\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\right) dx$$
$$\text{s.d. } 0 \leq v \leq v_0$$
$$u = u_0 \text{ auf } \Gamma_1 \cup \Gamma_2,$$

dabei ist ν eine Konstante, $\epsilon_i > 0$ für alle $i \in \{1, 2\}$ ein kleiner Parameter und $\Gamma_1, \Gamma_2 \subset \Omega$ der rechte bzw. linke Rand eines rechteckigen Gebietes $\Omega \subset \mathbb{R}^2$, d.h.,

$$\Omega = [0, a] \times [0, b] \;\; \Gamma_1 = \{0\} \times [0, b] \;\; \Gamma_2 = \{a\} \times [0, b].$$

3. Mathematische Grundlagen

Damit eine Lösung des Optimierungsproblems gefunden werden kann, Grundlagen in Optimierung, partieller Differentialgleichungen, Finiter Elemente und semidifferenzierbarer Newton Methoden gebraucht. Diese werden hier eingeführt.

3.1. Optimierung

Das Phasenfeldmodell für die Rissentstehung ist ein Optimierungsproblem mit Ungleichungsnebenbedingung. Um die Eindeutigkeit und Existenz einer Lösung zu zeigen, werden Grundlagen in der Optimierung benötigt. Außerdem werden Bedingungen vorgestellt, mit denen sich das Optimierungsproblem in eine Nullstellensuche umschreiben lässt. Grundsätzlich lassen sich Optimierungsprobleme in Probleme mit und ohne Nebenbedingung aufteilen.

3.1.1. Optimierungsproblem ohne Nebenbedingung

Optimierungsprobleme ohne Nebenbedingung kennt man im Endlichdimensionalen bereits aus der Schule. Ein Minimum oder Maximum soll gefunden werden, wozu die zu optimierende Funktion abgeleitet und Null gesetzt wird. Für etwas kompliziertere Probleme reicht Optimierung im Endlichdimensionalen nicht aus. Grundlagen für die unendlichdimensionale Optimierung werden benötigt.

Sei also W ein Banachraum und $J : W \to \mathbb{R}$ ein Funktional. Das Optimierungsproblem ohne Nebenbedingung hat folgende Form

$$\min_{w \in W} J(w) \tag{3.1}$$

Damit die Ableitung bestimmt werden kann, muss erst der Ableitungsbegriff in Banachräumen definiert werden. Dies ist die Gâteaux-Ableitung. Die Definitionen stam-

men aus [3, S. 50].

Sei $F : U \subset X \to Y$ ein Operator zwischen Banachräumen und $U \neq \emptyset$ offen.
Definition 3.1.1 (Richtungsableitung)**.** *F heißt Richtungsableitbar in $x \in U$, falls*

$$\partial F(x)(h) = \lim_{t \to 0^+} \frac{F(x+th) - F(x)}{t} \in Y$$

für alle $h \in X$ existiert. Dann heißt $\delta F(x,h)$ Richtungsableitung von F in Richtung h.
Definition 3.1.2 (Gâteaux differenzierbar)**.** *F heißt Gâteaux differenzierbar in $x \in U$, falls F richtungsableitbar ist und die Richtungsableitung*

$$F'(x) : X \to Y$$
$$h \mapsto \partial F(x)(h)$$

beschränkt und linear ist, d.h. $F'(x) \in L(X,Y)$.
Definition 3.1.3 (Fréchet differenzierbar)**.** *F heißt Fréchet differenzierbar in $x \in U$, falls F Gâteaux differenzierbar ist und folgende Approximation gilt*

$$\|F(x+h) - F(x) - F'(x)h\|_Y = o\left(\|h\|_X\right) \text{ für } \|h\|_X \to 0$$

Nun kann die Ableitung von J bestimmt werden und daraus resultierend das Optimeriungsproblem gelöst werden. Theorem und Beweis stammen aus der Vorlesung „Optimierung 2“, gelesen von Prof. B. Wirth [5].
Theorem 3.1.4. *Sei das Optimierungsproblem* (3.1) *gegeben. Sei $J : W \to \mathbb{R}$ Gâteaux differenzierbar in $\tilde{w} \in W$. Wenn $\tilde{w}$ das Optimierungsproblem löst, gilt*

$$\partial J(\tilde{w})(h) = 0 \quad \forall\ h \in W \tag{3.2}$$

Dabei ist h die Richtung der Ableitung.

Beweis. Für alle $h \in W$ muss $\alpha \mapsto J(\tilde{w} + \alpha h)$ minimal in $\alpha = 0$ sein. Daraus folgt

$$\frac{\partial}{\partial \alpha} f(x + \alpha h)|_{\alpha=0} = 0$$

□

Oftmals ist (3.2) eine partielle Differentialgleichung. In den Grundlagen partieller Differentialgleichungen 3.2 wird erklärt, wann eine Lösung existiert und ob diese eindeutig

ist.

3.1.2. Optimierungsproblem mit Ungleichungsnebenbedingung

Oftmals tauchen als Nebenbedingungen Ungleichungsbedingungen wie $a \leq u \leq b$ auf, wobei $a, b, u \in X$ gilt und X ein Vektorraum ist. Dabei ist u die zu optimierende Funktion, a die untere und b die obere Schranke. Zum besseren Verständnis der Bedeutung des Ungleichheitszeichens, wird ein positiver Kegel nach der Vorlesung „Optimierung 2“von Prof. Wirth [5] definiert.

Definition 3.1.5 (positiver Kegel)**.** *Sei X ein Vektorraum, $P \subset X$ ein konvexer Kegel. Für $x, y \in X$ schreiben wir $x \leq_P y$ oder $y \geq_p x$, falls $y - x \in P$. P heißt positiver Kegel.*

$x <_P y$ oder $y >_P x$ bedeutet $y - x \in \mathring{P}$

Wir werden Probleme der Form

$$\min_{w \in W} J(w) \quad s.d. \quad G(w) \leq_p 0$$

betrachten, wobei W, Z Banachräume sind, $J : W \to \mathbb{R}$ Gâteaux differenzierbar und $G : W \to Z$ die Nebenbedingung des Optimierungsproblems ist. $P \subset Z$ ist ein positiver Kegel. Die Nebenbedingung lässt sich in eine Raumnebenbedingung umschreiben, also $C := \{w \in W | G(w) \leq_P 0\}$. Dabei ist C nichtleer, abgeschlossen und konvex. Das Problem lautet

$$\min_{w \in W} J(w) \quad s.d. \quad w \in C \tag{3.3}$$

Je nachdem welche Notation praktischer ist, wird die eine oder andere benutzt. Bei Optimierungen dieser Art muss zunächst die Existenz und Eindeutigkeit der Lösung gesichert werden.

Theorem 3.1.6. *Sei*

1. *W reflexiver Banachraum*
2. *$C \subset W$ nichtleer, konvex und abgeschlossen*
3. *$J : W \to \mathbb{R}$ strikt konvex und stetig auf C*
4. *J Gâteaux differenzierbar*

5. $\lim_{w \in C, \|w\|_W \to \infty} J(w) = \infty$

Dann existiert genau eine Lösung von (3.3).

Beweis. Der Beweis und das Theorem sind in [3, S.66] zu finden □

Bei Optimierungsproblemen mit Nebenbedingung reicht als Bedingung für das Optimum nicht aus, dass die Ableitung 0 ist. Da das Optimum auf dem Rand des zulässigen Gebietes sein könnte, muss die Ableitung nicht zwingend 0 sein. Jedoch gibt es andere Bedingungen, die ausreichend für ein Optimum sind. Die Herleitung dieser Bedingungen, die im folgenden Karush-Kuhn-Tucker Bedingungen (kurz: KKT) genannt werden, werden aufgrund des Umfanges nicht eingeführt, sondern nur angegeben.

Theorem 3.1.7 (Lagrangefunktion)**.** *Seien X,Y normierte Räume, $P \subset Z$ ein positiver Kegel mit $\mathring{P} \neq \emptyset$. Sei $J : W \to \mathbb{R} \cup \{\infty\}$, $G : W \to Z$ konvex. Es existiert ein $\hat{w}$ im Bild(J), sodass $G(\hat{w}) <_P 0$. Außerdem gelte $\mu = \inf\{J(w) | G(w) \leq_P 0\} < \infty$.*

Dann existiert $z' \in Z^*$ mit $z' \geq_{P^*} 0$, sodass $\mu = \inf_{w \in W} J(w) + \langle G(w), z' \rangle_{Z,Z^*}$. Falls ein optimales $\overline{w}$ existiert, dann minimiert $\overline{w}$

$$J(w) + \langle G(w), z' \rangle_{Z,Z^*}$$

Beweis. Der Beweis ist im Script zur Vorlesung „Optimierung II “, gelesen von Prof. B. Wirth [5], zu finden. □

Mit dieser Bedingungen kann von (3.3) das KKT System aufgestellt werden. Dabei ist $\overline{w}$ die Lösung des Problems. μ und λ sind die Lagrange Multiplikatoren.

$$\begin{aligned} &\nabla J(\overline{w}) + \lambda - \mu = 0 \\ &\overline{w} \geq a \quad \mu \geq 0 \quad \mu(\overline{w} - a) = 0 \\ &\overline{w} \leq b \quad \lambda \geq 0 \quad \lambda(b - \overline{w}) = 0 \end{aligned} \tag{3.4}$$

Die unteren beiden Zeilen in (3.4) kann man als Projektion darstellen. Es gilt für alle $c > 0$

$$\begin{aligned} \mu &= \max\{0, \mu + c(\overline{w} - a)\} \\ \lambda &= \max\{0, \lambda + c(b - \overline{w})\} \end{aligned}$$

Daraus ergibt sich nach [4]:

$$\begin{aligned}\mu - \lambda &= \max\{0, \mu + c(\overline{w} - a)\} - \max\{0, \lambda + c(b - \overline{w})\} \\ &= \max\{0, \mu + c(\overline{w} - a)\} + \min\{-\lambda + c(\overline{w} - b)\} \\ &= \max\{0, \mu - \lambda + c(\overline{w} - a)\} + \min\{\mu - \lambda + c(\overline{w} - b)\}\end{aligned} \tag{3.5}$$

Diese Darstellung wird später benutzt, um das Problem über die Rissentstehung zu lösen.

3.2. Partielle Differentialgleichungen

Optimierungsprobleme kann man oft umschreiben, sodass, statt dem Optimierungsproblem, eine partielle Differentialgleichung gelöst wird. Dadurch können Rückschlüsse auf die Existenz und Eindeutigkeit vom Optimierungsproblem gezogen werden. Die Theorie, die dazu verwendet wird, ist aus der Vorlesung „partielle Differentialgleichungen“gelesen vom Prof. B. Wirth [6].

Betrachte das elliptische Dirichlet-Problem auf einem beschränkten Gebiet $\Omega \subset \mathbb{R}^n$

$$\begin{aligned}Lu &= f \text{ auf } \Omega \\ u &= g \text{ auf } \partial\Omega\end{aligned} \tag{3.6}$$

mit $g \in H^1(\Omega)$, $f : \Omega \to \mathbb{R}$ und $Lu(x) := -\operatorname{div}(A(x)\nabla u(x)) + b(x)\nabla u(x) + c(x)u(x)$, wobei $A : \Omega \to \mathbb{R}^{n\times n}$, $b : \Omega \to \mathbb{R}^n$ und $c : \Omega \to \mathbb{R}$.

Definition 3.2.1 (schwache Lösung)**.** *$u \in g + H_0^1(\Omega)$ heißt schwache Lösung zu* (3.6)*, falls*

$$B(u, v) := \int_\Omega \nabla v^T A \nabla u + b\nabla uv + cuv \,\mathrm{d}x = \int_\Omega fv \,\mathrm{d}x \quad \forall v \in H_0^1(\Omega)$$

Damit eine schwache Lösung eindeutig ist, werden einige Voraussetzungen benötigt.

Annahme 3.2.2. *Es existieren $\lambda, \Lambda, \nu > 0$, sodass $\forall x \in \Omega$, $\forall \xi, \zeta \in \mathbb{R}^n$ gilt*

1. $\xi^T A(x)\xi \geq \lambda|\xi|^2$
2. $|\xi^T A(x)\zeta| \leq \Lambda|\xi||\zeta|$
3. $\lambda^{-2}|b(x)|^2 + \lambda^{-1}|c(x)| \leq \nu^2$

4. $c(x) \geq 0$

Theorem 3.2.3 (Eindeutigkeit der schwachen Lösung)**.** *Seien die Annahmen 3.2.2 für das Problem* (3.6) *erfüllt. Falls eine schwache Lösung für* (3.6) *existiert, ist sie eindeutig.*

Beweis. Der Beweis wird im Script von Prof. B. Wirth zur Vorlesung „Partielle Differentialgleichungen“[6] geführt. □

Theorem 3.2.4 (Existenz der schwachen Lösung)**.** *Sei* Ω *beschränkt mit Lipschitz Rand.* A, b, c *seien beschränkt,* $f \in L^2(\Omega)$*. Dann existiert eine schwache Lösung* $u \in H^1(\Omega)$ *von* (3.6)*.*

Beweis. Der Beweis wird im Script von Prof. B. Wirth zur Vorlesung „Partielle Differentialgleichungen“[6] geführt. □

3.3. Finite Elemente

Finite Elemente sind die Grundlage, um partielle Differentialgleichungen auf zweidimensionalen Gebieten numerisch darstellen zu können. Dazu wird zunächst das Gebiet in Dreiecke trianguliert. Dann werden Basisfunktionen auf diesen Dreiecken definiert, die sogenannten globalen Formfunktionen. Aus diesen ist die gesuchte Funktion zusammengesetzt und kann damit berechnet werden. Die hier beschriebene Theorie richtet sich nach der Vorlesung „Numerik partieller Differentialgleichungen“gelesen von Dr. F. Wübbeling [7].

Es ist ein rechteckiges Gebiet in 2D gegeben. O.b.d.A. $\Omega = [0, a] \times [0, b]$. Auf dieses Gebiet legen wir ein äquidistantes Gitter G_h.

$$G_h := \left\{ (ih_1, jh_2) | i = 0, \cdots, \frac{a}{h_1}, j = 0, \cdots, \frac{b}{h_2} \right\}$$

$h = (h_1, h_2)$ ist die Schrittweite mit $a = (n+1)h_1$ und $b = (m+1)h_2$, $n+1$ die Anzahl der Stützpunkte in x-Richtung und $m+1$ die Anzahl der Stützpunkte in y-Richtung. Um ein sinnvolles Gitter zu erhalten, sollten h_1 und h_2 recht nahe beieinander gewählt werden. Nun wird durch die Gitterpunkte die Triangulierung gelegt. Diese nennen wir E_k und ist in Abbildung 3.1 dargestellt.

Um das Referenzelement aufstellen zu können, werden dreieckig lineare Lagrange Elemente genutzt. Bei diesen sind die Funktionsauswertungen auf den Ecken der Dreiecke

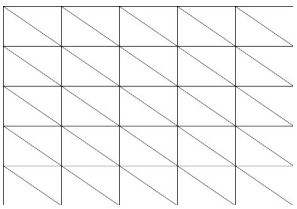


Abbildung 3.1.: Triangulierung eines rechteckigen Gebietes

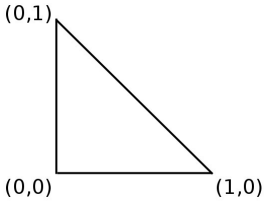


Abbildung 3.2.: Referenzdreieck

gegeben. Das Finite Element ist gegeben durch (E, P, Ψ), wobei E das Referenzdreieck 3.2 ist, $P = \mathcal{P}_1$, Polynome auf $\mathbb{R}^2$ vom Grad 1 mit Basis $\{p_1, p_2, p_3\}$,

$$p_1(x, y) := 1, \qquad p_2(x, y) := x, \qquad p_3(x, y) := y$$

und $\Psi := \{\varphi_0, \varphi_1, \varphi_2\}$ Funktionale auf P und damit eine Basis von P^* sind. φ_i nennt man lokale Formfunktionen und es muss gelten: $\varphi_i(p_j) = \delta_{ij}, i, j \in \{1, 2, 3\}$. Dabei ist δ_{ij} das Kronecker-Delta. Außerdem soll gelten $\varphi_i(p_j) = p_j(a_i)$, wobei a_i eine Auswertung in einer Ecke des Dreiecks ist. Daraus ergibt sich, dass

$$\varphi_1 = 1 - x - y, \qquad \varphi_2 = x, \qquad \varphi_3 = y \tag{3.7}$$

Jedes Finite Element (E_k, P_k, Ψ_k) lässt sich mit der affin linearen Transformation

$$\begin{array}{rccc} T: & \mathbb{R}^2 & \to & \mathbb{R}^2 \\ & \begin{pmatrix} x \\ y \end{pmatrix} & \mapsto & \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \pm \begin{pmatrix} h_1 x \\ h_2 y \end{pmatrix} \end{array}$$

durch das Referenzelement darstellen. Dabei entspricht $(a_1, a_2)^t$ dem Eckpunkt mit dem 90° Winkel des Dreiecks und $(h_1, h_2)^t$ ist die Höhe bzw. Breite des Dreiecks. Mit dem Transformationssatz können alle Berechnungen auf dem Referenzelement ausgeführt werden und dann auf das transformierte Element übertragen werden. Durch die Transformation muss zu allen Integralen der Term $|\det DT(x, y)|^{-1}$ multipliziert werden.

Das ergibt

$$|\det \mathrm{D}\ T(x,y)|^{-1} = |\det \begin{pmatrix} h_1 & 0 \\ 0 & h_2 \end{pmatrix}|^{-1} = \frac{1}{h_1 h_2}$$

Die Familie $\{(E_k, P_k, \Psi_k)\}$ von Finiten Elementen, die durch unsere Triangulierung hervorgegangen ist, ist verträglich. Deshalb können die globalen Formfunktionen aufgestellt werden, die auf dem gesamten Gebiet Ω definiert sind. Die globale Formfunktion T_j ist 1 auf dem Gitterpunkt j und 0 sonst.

Für die Berechnung von linearen Funktionen auf dreieckig linearen Lagrange Elementen brauchen wird teilweise eine explizite Darstellung benötigt. Durch die Triangulierung ergeben sich zwei Arten von Dreiecken. Dabei entspricht a_i dem Wert der Funktion $a : \mathbb{R}^2 \to \mathbb{R}$ an dem Eckpunkt i.

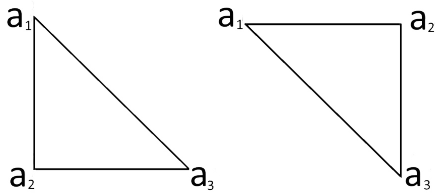


Abbildung 3.3.: Gerade (links) und ungerade Dreiecke mit den Werten von a

$a(x,y)$ wird auf dem linken Dreieck von Abbildung 3.3 dargestellt durch

$$a(x,y) = (a_3 - a_2)x + (a_1 - a_2)y + a_2 \quad \text{mit} \quad \nabla a(x,y) = \begin{pmatrix} a_3 - a_2 \\ a_1 - a_2 \end{pmatrix} \tag{3.8}$$

und auf dem rechten Dreieck von 3.3 wird $a(x,y)$ dargestellt durch

$$a(x,y) = (a_1 - a_2)x + (a_3 - a_2)y + a_2 \quad \text{mit} \quad \nabla a(x,y) = \begin{pmatrix} a_1 - a_2 \\ a_3 - a_2 \end{pmatrix} \tag{3.9}$$

3.4. Semidifferenzierbare Newton Methode

Semiglatte Newton Methoden werden gebraucht, um Nullstellen von nicht differenzierbaren Funktionen numerisch zu berechnen. Die Rissentstehung ist ein nicht differenzierbares Problem. Um die Idee der Newton Methoden zu verstehen, werden zunächst die einfache Newton Methode ohne Nebenbedingung und dann solche mit einfachen Ne-

benbedingungen eingeführt. Um diese realisieren zu können, wird der Begriff der Semidifferenzierbarkeit benötigt. Das ist eine Mengenwertige Ableitung, mit der auch nicht differenzierbare, aber stetige Punkte in einer Funktion abgeleitet werden können. Damit kann die semidifferenzierbare Newton Methode eingeführt werden, von der wir auch die Konvergenz betrachten werden. Dieses Kapitel richtet sich nach "Optimazation with PDE Constraints"von M. Hinze u.a. [3, S. 115 ff].

3.4.1. Newton Methode ohne Nebenbedingung

Als Erstes leiten wir uns zum Verständnis der semidifferenzierbaren Newton Methode die einfache Newton Methode her. Dazu betrachten wir wie vorher das Minimierungsproblem

$$\min_{w\in\mathbb{R}^n} f(w) \qquad f:\mathbb{R}^n\to\mathbb{R} \tag{3.10}$$

Die Optimalbedingung zu diesem Problem lautet $\nabla f(w)=0$. Zweitens wollen wir ein numerisches Verfahren für dieses Problem entwickeln. Dazu setzen wir $G := \nabla f$. Da wir ein diskretes Verfahren wollen, setzten wir $w_0, w_1, \dots$ in G ein. Wir erhalten

$$G(w_{k+1}) = 0$$

Um ein iteratives Verfahren zu erhalten taylorn wir G in w_k. Das ergibt

Theorem 3.4.1 (einfaches Newtonverfahren)**.** *Der Algorithmus 1 löst das Optimierungsproblem* (3.10)*. Es konvergiert superlinear falls $G \in C^1$ und G' invertierbar ist.*

Data*: w^0 (möglichst nah an der Lösung $\overline{w}$)*
for $k = 0, 1, \dots$ **do**
  Löse $G'(w^k)s^k = -G(w^k)$;
  $w^{k+1} = w^k + s^k$;
end

Algorithm 1: einfache Newton Methode

3.4.2. Konvergenz der generalisierten Newton Methode

Betrachten wir ein unendlichdimensionales Optimierungsproblem

$$G(x) = 0 \tag{3.11}$$

mit $G : X \to Y$, wobei X, Y Banachräume sind. Sei $\overline{x}$ die Lösung der Gleichung.

Um eine numerische Lösung von (3.11) zu erhalten, benutzen wir einen ähnlichen Algorithmus, wie den für das einfache Newtonverfahren, nur allgemeiner:

Data: $x^0 \in X$ (möglichst nah an der Lösung $\overline{x}$)
for $k = 0, 1, \ldots$ **do**
  Wähle invertierbaren Operator $M_k \in L(X, Y)$;
  Erhalte s_k *beim Lösen von* $M_k s^k = -G(x^k)$;
  $x^{k+1} = x^k + s^k$;
end

Algorithm 2: Generalisierte Newton Methode

Bis jetzt war der Operator M_k die Ableitung von G. Dies ist jedoch nicht möglich, wenn G nicht differenzierbar ist. Wie der Operator M_k in diesem Fall sinnvoll zu wählen ist, wird später bestimmt.

Nun untersuchen wir die durch diesen Algorithmus gewonnene Folge x^k in einer Umgebung von $\overline{x}$. Sei $d^{k+1} = x^{k+1} - \overline{x}$ der Abstand zwischen dem Iterationsschritt und der Lösung. Dann gilt

$$\begin{aligned} M_k d^{k+1} &= M_k(x^{k+1} - \overline{x}) = M_k(x^k + s^k - \overline{x}) = M_k d^k - G(x^k) \\ &= G(\overline{x}) + M_k d^k - G(x^k) \end{aligned}$$

Wir erhalten

Theorem 3.4.2 (Konvergenz der generalisierten Newton Methode)**.** *Betrachte* (3.11) *mit der Lösung* $\overline{x}$*. Sei* x^k *die Folge, die durch den Generalisierten Newton Algorithmus 2 erzeugt wurde. Sei* x^0 *nah genug an* $\overline{x}$ *gewählt*

1. *Falls* $\exists \gamma \in (0, 1)$ *mit*

$$\begin{aligned} &\|d^{k+1}\|_X = \|M_k^{-1}\left(G(\overline{x} + d^k) - G(\overline{x}) - M_k d^k\right)\|_X \leq \gamma \|d^k\|_X \\ &\forall k \text{ mit } \|d_k\|_X \text{ klein genug} \end{aligned}$$

   gilt, dann konvergiert $x^k \to \overline{x}$ *linear mit Konstante* γ*.*

2. *Falls* $\forall \eta \in (0, 1) \quad \exists \delta_\eta > 0$*, sodass*

$$\begin{aligned} &\|d^{k+1}\|_X = \|M_k^{-1}\left(G(\overline{x} + d^k) - G(\overline{x}) - M_k d^k\right)\|_X \leq \eta \|d^{k+1}\|_X \\ &\text{für } \|d_k\|_X < \delta_\eta \end{aligned}$$

gilt, dann konvergiert $x^k \to \overline{x}$ super linear.

Beweis. Der Beweis ist in [3, S. 118] zu finden. □

Oft teilt man diese Kleinheitsannahmen in zwei Teile auf:
Definition 3.4.3 (Regularitätsannahme)**.** *Sei $M_k \in L(X,Y)$, wobei X und Y Banachräume sind. Dann ist die Regularitätsannahme gegeben durch:*

$$\|M_k^{-1}\|_{Y\to X} \leq C \quad \forall k \geq 0$$

Bemerkung (Operatornorm)*.* Die Notation für die Operatornorm von einem linearen Operator $f : X \to Y$, wobei X, Y normierte Vektorräume sind lautet:

$$\|f\|_{X\to Y} := \sup_{\|x\|_X=1} \|f(x)\|_Y$$

Definition 3.4.4 (Approximationsannahme)**.** *Sei $M_k \in L(X,Y)$, wobei X,Y Banachräume sind, $\overline{x}$ die Lösung von $G(x) = 0$ und $d^k := x^k - \overline{x}$. Sei $\alpha + 1 > 1$ Dann ist die Approximationsannahme gegeben durch:*

$$\|G(\overline{x} + d^k) - G(\overline{x}) - M_k d^k\|_Y = o(\|d^k\|_X) \text{ für } \|d_k\|_X \to 0$$

3.4.3. Semidifferential

Die geeingnete Wahl von M_k ist das sogenannte Semidifferential
Definition 3.4.5 (verallgemeinerte Differentiale)**.** *Seien X und Y Banachräume und $G : X \to Y$ ein stetiger Operator. Dann ist die Menge der verallgemeinerten Differentiale definiert als*

$$\partial G : X \rightrightarrows L(X,Y)$$

Dabei meint $X \rightrightarrows L(X,Y)$, dass ein Punkt $x \in X$ auf eine Menge von linearen Operatoren abgebildet wird.

Nun können wir, um in unser Newtonverfahren das Semidifferential zu verwenden, $M_k \in \partial G(x^k)$ wählen. Damit unser Verfahren aber super linear konvergiert, muss

gelten

$$\sup_{M \in \partial G(\overline{x}+d)} \|G(\overline{x}+d^k) - G(\overline{x}) - M_k d\|_Y = o\left(\|d\|_X\right) \text{ für } \|d\|_X \to 0$$

Dieses nennt sich semidifferenzierbar.

Definition 3.4.6 (semidifferenzierbar)**.** *Sei $G : X \to Y$ ein stetiger Operator zwischen Banachräumen. Sei $\partial G : X \rightrightarrows L(X,Y)$ mit nicht leeren Bildern gegeben wie oben. G heißt ∂G semidifferenzierbar in $x \in X$, falls*

$$\sup_{M \in \partial G(x+d)} \|G(x+d^k) - G(x) - M_k d\|_Y = o\left(\|d\|_X\right) \text{ für } \|d\|_X \to 0 \tag{3.12}$$

Lemma 3.4.7. *Sei $G : X \to Y$ ein Operator zwischen Banachräumen und stetig Fréchet differenzierbar in einer Umgebung von x. Dann ist G $\{G'\}$-semidifferenzierbar in x.*

$\{G'\}$ beschreibt den Operator $\{G'\} : X \rightrightarrows L(X,Y)$ mit $\{G'\}(x) = \{G'(x)\}$

Beweis.

$$\begin{aligned}
&\|G(x+d^k) - G(x) - G'(x+d)d\|_Y \\
&\leq \|G(x+d^k) - G(x) - G'(x)d\|_Y + \|G'(x)d - G'(x+d)d\|_Y \\
&\leq o\left(\|d\|_X\right) + \|G'(x) - G'(x+d)\|_{X\to Y}\|d\|_X = o\left(\|d\|_X\right)
\end{aligned}$$

Der zweite Teil des Beweises erfolgt analog, siehe [3, S. 121] □

Theorem 3.4.8 (Rechenregeln semidifferenzierbare Funktionen)**.** *Seien X, Y, Z, X_i, Y_i Banachräume.*

1. *Falls die Operatoren $G_i : X_i \to Y_i$ ∂G_i-semidifferenzierbar in x sind, dann ist (G_1, G_2) $(\partial G_1, \partial G_2)$-semidifferenzierbar in x.*
2. *Falls die Operatoren $G_i : X \to Y$ ∂G_i-semidifferenzierbar in x sind, dann ist $G_1 + G_2$ $(\partial G_1 + \partial G_2)$-semidifferenzierbar in x.*
3. *Seien $G_1 : Y \to Z$ und $G_2 : X \to Y$ ∂G_i-semidifferenzierbar in $G_2(x)$ und in x. Sei außerdem ∂G_1 beschränkt in einer Umgebung von $x = G_2(x)$ und G_2 Lipschitzstetig in einer Umgebung von x. Dann ist $G = G_1 \circ G_2$ ∂G-semidifferenzierbar*

mit

$$\partial G(x) = \{M_1 M_2 | M_1 \in \partial G_1 \left(\partial G_2(x)\right), \quad M_2 \in \partial G_2(x)\}$$

Beweis. Der Beweis ist in [3, S. 122] zu finden. □

3.4.4. Semidifferenzierbare Newton Methode

Mit dem Semidifferential können wir nun die semidifferenzierbare Newton Methode definieren.

Data: $x^0 \in X$ (möglichst nah an der Lösung $\overline{x}$)
for $k = 0, 1, \cdots$ **do**
  Wähle $M_k \in \partial G(x^k)$;
  Erhalte s_k *beim Lösen von* $M_k s^k = -G(x^k)$;
  $x^{k+1} = x^k + s^k$;
end

Algorithm 3: semidifferenzierbare Newton Methode

Damit diese konvergiert, muss die Approximations- und Regularitätsannahme erfüllt sein. Die Approximationsannahme ist durch die Semidifferenzierbarkeit gegeben. Es fehlt noch die Regularitätsannahme.

Definition 3.4.9 (Regularitätsannahme für das semidifferenzierbare Newton Verfahren)**.** *Betrachte* (3.11) *mit der Lösung* $\overline{x}$*. Dann lautet die Regularitätsannahme*

$$\exists C > 0, \quad \exists \delta > 0 : \|M^{-1}\|_{X \to Y} \leq C \quad \forall M \in \partial G(x) \quad \forall x \in X, \quad \|x - \overline{x}\|_X < \delta \tag{3.13}$$

Theorem 3.4.10 (Konvergenz das semidifferenzierbare Newton Verfahrens)**.** *Sei das Problem* (3.11) *gegeben mit der Lösung* $\overline{x}$*. Seien* X, Y *Banachräume,* $G : X \to Y$ *stetig und* ∂G *semidifferenzierbar und die Regularitätsannahme* (3.13) *sei erfüllt. Dann existiert* $\delta > 0$*, sodass für alle* $x^0 \in X$ *mit* $\|x^0 - \overline{x}\|_X < \delta$ *die semidifferenzierbar Newton Methode super linear gegen* $\overline{x}$ *konvergiert.*

Beweis. Nach 3.4.2 können wir aus einem Newtonverfahren der Form 2, d.h. es gilt $M_k \in \mathcal{L}(X, Y)$, M_k ist invertierbar und

$$\|M_k^{-1} \left(G(\overline{x} + d^k) - G(\overline{x}) - M_k d^k\right)\|_X = o(\|d^k\|_X)$$

gilt,folgern, dass das Newtonverfahren super linear konvergiert. Da $M_k \in \partial G$, ist $M_k \in \mathcal{L}(X, Y)$. M_k ist invertierbar, da die Regularitätsannahme gilt. Außerdem gilt mit der Regularitätsannahme und der Semidifferenzierbarkeit

$$\begin{aligned}
&\|M_k^{-1}\left(G(\overline{x}+d^k) - G(\overline{x}) - M_k d^k\right)\|_X \\
&\leq \|M_k^{-1}\|_X\| \left(G(\overline{x}+d^k) - G(\overline{x}) - M_k d^k\right)\|_X \\
&\leq Co(\|d^k\|_X) = o(\|d^k\|_X)
\end{aligned}$$

Also ist 3.4.2 anwendbar. □

Damit haben wir Bedingungen für die Konvergenz der semidifferenzierbaren Newton Methode gefunden. Diese können wir für den Beweis der Konvergenz bei unserer Newton Methode anwenden.

4. Anwendung auf das Phasenfeldmodell für Rissentstehung

Nachdem wir die mathematischen Grundlagen für die Betrachtung eines Optimierungsproblems kennengelernt haben, wollen wir diese anwenden. Das Problem war gegeben durch

$$\begin{aligned}&\min_{u\in H^1(\Omega)^2, v\in H^1(\Omega)} \int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 + \nu\left(\epsilon_2|\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\right)dx\\ &\text{s.d. } 0\le v\le v_0\\ &u = u_0 \text{ auf } \Gamma_1\cup\Gamma_2\end{aligned}$$

Zunächst teilen das Optimierungsproblem in zwei voneinander unabhängige Optimierungen auf: Die Optimierung nach u und die Optimierung nach v. Im Anschluss betrachten wir beide Optimierungen genauer, indem wir sie umschreiben und numerische Verfahren zur Lösung entwickeln. Zum Schluss fusionieren wir beide Verfahren.

4.1. Erste Betrachtung des Modells

Beim genaueren Betrachten bemerkt man, dass die Ungleichungsnebenbedingung nur von v und die Randbedingung nur von u abhängt. Dies bietet die Möglichkeit das

Optimierungsproblem in zwei Teilprobleme aufzuteilen.

$$\begin{aligned}
&\min_{u\in H^1(\Omega)^2} \int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 + \nu\left(\epsilon_2|\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\right)\mathrm{d}x \\
&u = u_0 \text{ auf } \Gamma_1 \cup \Gamma_2 \\
&\min_{v\in H^1(\Omega)} \int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 + \nu\left(\epsilon_2|\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\right)\mathrm{d}x \\
&\text{s.d. } 0 \le v \le v_0
\end{aligned}$$

Wenn man beide Probleme implementiert, löst man zunächst die Optimierung nach u und setzt die Lösung dann in die Optimierung nach v ein. Danach setzt man die Lösung von v in die Optimierung nach u ein. Dieses Vorgehen wird mit der semidifferenzierbaren Newton Methode wiederholt. Betrachten wir zuerst die Optimierung nach u.

4.2. Optimierung nach u

Das Kapitel ist in zwei Unterkapitel aufgeteilt: Die analytische und numerische Betrachtung.

Im analytischen Teil formulieren wir das Minimierungsproblem zu einem Problem ohne Nebenbedingung um. Dieses lässt sich dann als partielle Differentialgleichung schreiben. Die Existenz und Eindeutigkeit der schwachen Lösung wird am Schluss des ersten Teils bewiesen.

Für die numerische Betrachtung schreiben wir das Problem nochmal um und wenden dann Finite Elemente und den Galerkin Ansatz an. Das Resultat ist ein Gleichungssystem, das sich einfach lösen lässt.

4.2.1. Analytische Betrachtung

Erinnern wir uns an das Optimierungsproblem, das die Verschiebung des Körpers bei der Entstehung von einem Riss beschreibt.

$$\min_{u\in H^1(\Omega)^2} \int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 + \nu\left(\epsilon_2|\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\right)\mathrm{d}x$$
$$u = u_0 \text{ auf } \Gamma_1 \cup \Gamma_2$$

Es ist leichter ein Problem ohne Nebenbedingung zu betrachten, also nehmen wir die Nebenbedingung mit in dem Raum auf, über den wir optimieren. Also suchen wir statt $u \in H^1(\Omega)^2$

$$u \in u_0 + H_0^1(\Omega)^2 := u_0 + \{u \in H^1(\Omega)^2 | u = 0 \text{ auf } \Gamma_1 \cup \Gamma_2\}$$

Das Problem hat dann folgende Form

$$\min_{u\in u_0+H_0^1(\Omega)^2} \int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 + \nu\left(\epsilon_2|\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\right)\mathrm{d}x \tag{4.1}$$

Da $u : \Omega \to \mathbb{R}^2$, müssen wir nach u_1 und nach u_2 minimieren. Es tauchen keine Mischungen aus den Termen u_1 und u_2 auf, das heißt, dass wir die Optimierungen trennen können. Beide sind identisch, es müssen später nur unterschiedliche Werte eingesetzt werden. Betrachten wir o.B.d.A. die Optimierung nach u_1.

Theorem 4.2.1 (Bedingung für ein Minimum)**.** *Sei das Minimierungsproblem* (4.1) *gegeben und* $\tilde{u_1}$ *nimmt das Minimum an. Dann gilt*

$$\int_\Omega 2\left(v^2+\epsilon_1\right)\nabla\tilde{u_1}\nabla\psi\,\mathrm{d}x = 0 \quad \forall\psi \in u_0 + H_0^1(\Omega) \tag{4.2}$$

Beweis. Nach 3.1.4 muss nur überprüft werden, ob die Gâteaux-Ableitung von

$$\begin{array}{rccl} J: & u_0 + H_0^1(\Omega)^2 & \to & \mathbb{R} \\ & u & \mapsto & \int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 + \epsilon_2|\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\,\mathrm{d}x \end{array}$$

von der Form (4.2) ist. Leiten wir J ab:

$$\begin{aligned}
\partial J(u)(\psi) &= \lim_{t\to 0}\tfrac{1}{t}\Big(\quad J(u_1+t\psi,u_2)-J(u_1,u_2)\Big)\\
&= \lim_{t\to 0}\tfrac{1}{t}\Big(\int_\Omega (v^2+\epsilon_1)\,|\nabla(u_1+t\psi)|^2+|\nabla u_2|^2+\epsilon_2|\nabla v|^2\\
&\qquad +\tfrac{1}{\epsilon_2}(1-v)^2\,\mathrm{d}x\\
&\qquad -\int_\Omega (v^2+\epsilon_1)\,|\nabla u_1|^2 \qquad\qquad +|\nabla u_2|^2+\epsilon_2|\nabla v|^2\\
&\qquad +\tfrac{1}{\epsilon_2}(1-v)^2\,\mathrm{d}x\Big)\\
&= \lim_{t\to 0}\tfrac{1}{t}\int_\Omega (v^2+\epsilon_1)\,(|\nabla(u_1+t\psi)|^2-|\nabla u_1|^2)\,\mathrm{d}x\\
&= \lim_{t\to 0}\tfrac{1}{t}\int_\Omega (v^2+\epsilon_1)\,(|\nabla u_1+t\nabla\psi|^2-|\nabla u_1|^2)\,\mathrm{d}x\\
&= \lim_{t\to 0}\tfrac{1}{t}\int_\Omega (v^2+\epsilon_1)\,(|\nabla u_1|^2+2t\nabla u_1\nabla\psi+t^2|\nabla\psi|^2-|\nabla u_1|^2)\,\mathrm{d}x\\
&= \lim_{t\to 0}\tfrac{1}{t}\int_\Omega (v^2+\epsilon_1)\,(2t\nabla u_1\nabla\psi+t^2|\nabla\psi|^2)\,\mathrm{d}x\\
&= \int_\Omega 2\,(v^2+\epsilon_1)\,\nabla u_1\nabla\psi\,\mathrm{d}x
\end{aligned}$$

Damit dies eine Gâteaux Ableitung ist, muss die Abbildung $J'(u_1):\psi\mapsto\partial J(u_1,\psi)\in\mathbb{R}$ linear und beschränkt sein. Linearität ist einfach nachzurechnen. Beschränktheit lässt sich durch Cauchy-Schwarz zeigen. □

Somit lautet unser analytisches Problem: Finde $u_1\in u_0+H_0^1(\Omega)$, sodass für alle $\psi\in u_0+H_0^1(\Omega)$ gilt

$$0=\int_\Omega 2\left(v^2+\epsilon_1\right)\nabla u_1\nabla\psi\,\mathrm{d}x \tag{4.3}$$

Nun ist noch interessant, ob eine Lösung existiert und ob sie eindeutig ist. Dieses hängt von u_0 und v_0 ab.

Theorem 4.2.2 (Existenz und Eindeutigkeit)**.** *Sei $u_0\in H^1(\Omega)^2,,v_0\in H^1(\Omega)$. Die schwache Lösung $u\in u_0+H_0^1(\Omega)$ von (4.3) existiert und ist eindeutig.*

Beweis. Wir wenden 3.2.4 an. Dazu müssen wir die Bilinearform aufstellen und alle Annahmen aus 3.2.2 zeigen. Die Bilinearform lautet

$$\begin{aligned}
B(u_1,\psi):\ \left(u_0+H^1(\Omega)0^2\right)^2 &\to \mathbb{R}\\
(u_1,\psi) &\mapsto \int_\Omega (v^2+\epsilon_1)\,2\nabla u_1\nabla\psi\,\mathrm{d}x
\end{aligned}$$

Mit den Bezeichnungen aus Kapitel 3.2 ist $g = u_0,\ f, b, c = 0$ und

$$A(x) := \begin{pmatrix} v^2(x) + \epsilon_1 & 0 \\ 0 & v^2(x) + \epsilon_1 \end{pmatrix}$$

Aus 3.2.2 sind 3 und 4 bereits erfüllt, da $b, c = 0$ gilt. Beweisen wir 1.

Sei $\xi \in \mathbb{R}^n$. Dann gilt:

$$\begin{aligned} \xi^T A(x)\xi &= \xi^T \begin{pmatrix} v^2(x) + \epsilon_1 & 0 \\ 0 & v^2(x) + \epsilon_1 \end{pmatrix} \xi \\ &= (v^2 + \epsilon_1)\xi \cdot \xi \\ &\geq \epsilon_1 |\xi|^2 \end{aligned}$$

Damit ist Annahme 1 erfüllt mit $\lambda = \epsilon_1$. Für Annahme 2 gilt

$$|\xi^T A(x)\zeta| = (v^2 + \epsilon_1)\xi \cdot \zeta \leq (v_0^2 + \epsilon_2)\xi \cdot \zeta \leq (\sup(v_0)^2 + \epsilon_2)|\xi||\zeta|$$

mit $\Lambda = \sup(v_0)^2 + \epsilon$. □

4.2.2. Numerische Betrachtung

Das vorherige Kapitel hat das Minimierungsproblem zu folgender Nullstellensuchen vereinfacht:

Finde $u_1 \in u_0 + H_0^1(\Omega)$, sodass

$$0 = \int_\Omega \left(v^2 + \epsilon_1\right) \nabla u_1 \nabla \psi \, \mathrm{d}x \quad \forall \psi \in u_0 + H_0^1(\Omega)$$

Da die Nullstelle im Raum $H_0^1(\Omega)$ einfacher zu finden ist, als im Raum $u_0 + H_0^1(\Omega)$, stellen wir das Problem um. Dazu definieren wir $\tilde{u}_0 \in u_0 + H_0^1(\Omega)$, sodass $\tilde{u}_0$ auf dem Rand $\Gamma_1 \cup \Gamma_2$ u_{0_1} entspricht und sonst 0 ist. Definiere zusätzlich $\tilde{u} \in H_0^1(\Omega)$, sodass $u_1 = \tilde{u} + \tilde{u}_0$. Damit lässt sich das Problem umschreiben zu

Finde $\tilde{u} \in H_0^1(\Omega)$, sodass $\forall \psi \in H_0^1(\Omega)$

$$-\int_\Omega \left(v^2 + \epsilon_1\right) \nabla \tilde{u}_0 \nabla \psi \, \mathrm{d}x = \int_\Omega \left(v^2 + \epsilon_1\right) \nabla \tilde{u} \nabla \psi \, \mathrm{d}x$$

Zur numerischen Betrachtung bieten sich Finite Elemente, insbesondere die dreieckig linearen Lagrange Elemente an. Dafür triangulieren wir das Gebiet, wie in 3.3 dargestellt. Nun nutzen wir den Galerkin Ansatz. Dafür gilt ab jetzt $k := (m+1)(n+1)$

$$\tilde{u}(x,y) := \sum_{i=1}^{k} u_i^h T_i(x,y)$$

Dabei sind $T_i(x,y)$ die globalen Formfunktionen und u_i^h die gesuchten Konstanten. Setzt man die Definition von $\tilde{u}$ ein und ersetzt $\psi \in H_0^1(\Omega)$ durch die Basis von P^*, also den globalen Formfunktionen T_i, so gilt $\forall i \in \{1, \cdots, k\}$

$$\begin{aligned}
&-\int_\Omega \left(v^2+\epsilon_1\right)\nabla\tilde{u}_0\nabla\psi\,\mathrm{d}x = \int_\Omega \left(v^2+\epsilon_1\right)\nabla\tilde{u}\nabla\psi\,\mathrm{d}x \\
\Leftrightarrow &-\int_\Omega \left(v^2+\epsilon_1\right)\sum_{i=1}^{k} u_{0\,i}^h\nabla T_i\nabla T_i\,\mathrm{d}x = \int_\Omega \left(v^2+\epsilon_1\right)\sum_{i=1}^{k} u_i^h\nabla T_i\nabla T_j\,\mathrm{d}x \\
\Leftrightarrow &-\sum_{i=1}^{k} u_{0\,i}^h\int_\Omega \left(v^2+\epsilon_1\right)\nabla T_j\nabla T_i\,\mathrm{d}x = \sum_{i=1}^{k} u_i^h\int_\Omega \left(v^2+\epsilon_1\right)\nabla T_i\nabla T_j\,\mathrm{d}x \\
\Leftrightarrow &L * u_0^h = L * u^h
\end{aligned}$$

wobei $u^h := (u_1^h, \cdots u_k^h)^T$, $u_0^h := (u_{0\,1}^h, \cdots u_{0\,k}^h)^T$ und $L := \left(\int_\Omega \left(v^2+\epsilon\right)\nabla T_i\nabla T_j\,\mathrm{d}x\right)_{ij}$

Also müssen wir L berechnen und dann das Gleichungssystem $L * u_0^h = L * u^h$ lösen.

Berechnung des u Integrals

Als erste Vereinfacherung betrachten wir nicht mehr das Integral über Ω, sondern über die einzelnen Dreiecke der Triangulierung. Desweiteren ist T_i linear, also ∇T_i konstant. Es gilt

$$\begin{aligned}
\int_\Omega \left(v^2+\epsilon_1\right)\nabla T_i\nabla T_j\,\mathrm{d}x &= \sum_{\tilde{E}\in E_k}\int_{\tilde{E}} \left(v^2+\epsilon_1\right)\nabla T_i\nabla T_j\,\mathrm{d}x \\
&= \sum_{\tilde{E}\in E_k}\nabla T_i\nabla T_j\int_{\tilde{E}} \left(v^2+\epsilon_1\right)\mathrm{d}x
\end{aligned}$$

Wir kennen $\nabla T_i\nabla T_j$ auf jedem Dreieck. Also muss nur noch $\int_E v^2+\epsilon\,\mathrm{d}x$ berechnet werden. Es darf über das Referenzdreieck integriert werden, da durch den Transfor-

mationssatz das Integral über das transformierte Element gewonnen werden kann. Es gilt:

$$\int_E v^2 + \epsilon_1 \,\mathrm{d}x = \int_E v^2 \,\mathrm{d}x + \frac{1}{2}\epsilon_1$$

Da v bereits numerisch berechnet wurde, haben wir nur Funktionsauswertungen von v an den Ecken des Dreieckes gegeben und wir wissen, dass $v \in \mathcal{P}_1$. Also ist v eindeutig bestimmt und kann berechnet werden. Die Darstellung von v ist in (3.8) und (3.9) zu finden.

Die Berechnung von $\int_E v^2 \,\mathrm{d}x$ sieht wie folgt aus:

$$\begin{aligned}\int_E v(x,y)^2 \,\mathrm{d}x\,\mathrm{d}y &= \int_0^1 \int_0^{1-y} \left((v_3 - v_1)x + (v_2 - v_1)y + v_1\right)^2 \mathrm{d}x\,\mathrm{d}y \\ &= \frac{1}{12}(v_1^2 + v_2^2 + v_3^2 + v_1v_2 + v_1v_3 + v_2v_3)\end{aligned}$$

Da die Berechnung über das transformierte Element durchgeführt wurde, muss noch der Multiplikator $\frac{1}{h_1 h_2}$ eingefügt werden.

Berechnen wir nun

$$L_{i,j} = \sum_{\tilde{E} \in E_k} \nabla T_i \nabla T_j \int_{\tilde{E}} \left(v^2 + \epsilon\right) \mathrm{d}x$$

wobei T_i auf dem Gitterpunkt i den Wert 1 hat und sonst den Wert 0 annimmt. Das heißt genauer, dass T_i nur auf sechs Dreiecken ungleich 0 ist. Um das Integral zu bestimmen, braucht man also maximal sechs Dreiecke. Falls der Gitterpunkt am Rand liegen sollte, betrachtet man nur drei Dreiecke, da die Dreiecke hinter dem Rand 0 gesetzt werden. An den Ecken berechnet man aus den selben Grund entweder ein oder zwei Dreiecke.

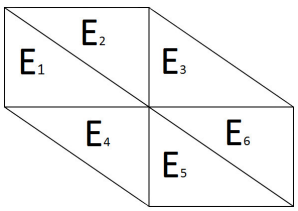


Abbildung 4.1.: Triangulierung im Inneren

Jetzt können wir L_{ij} für festes i, j berechnen. Falls i und j nicht adjazent sind, ist $L_{ij} = 0$, da $T_i T_j = 0$ gilt. Seien nun T_I und T_j adjazent. Hier haben wir vier Fälle:

- i liegt auf j, also $j = i$
- j liegt rechts neben i also $j = i + 1$
- j liegt direkt unter i, $j = i + n + 1$
- j liegt schräg unter i, also $j = i + n + 2$

Betrachten wir für die einzelnen Berechnungen Abbildung 4.1. T_i ist immer der Mittelpunkt dieser Zeichnung, T_j ist entsprechend des jeweiligen j positioniert. In den Berechnungen stimmen die Nummerierungen der Dreiecke mit den Nummerierungen in der Abbildung 4.1 überein und φ_k mit $k \in \{1, 2, 3\}$ entspricht den φ_k in (3.7).

Betrachten wir nun die vier Fälle.

i und j sind gleich

$$\begin{aligned} L_{i,i} &= \sum_{E \in E_k} \nabla T_i \nabla T_i \int_E \left(v^2 + \epsilon\right) \mathrm{d}x \\ &= \nabla\varphi_1 \nabla\varphi_1 \int_{E_1} \left(v^2 + \epsilon\right) \mathrm{d}x + \nabla\varphi_2 \nabla\varphi_2 \int_{E_2} \left(v^2 + \epsilon\right) \mathrm{d}x \\ &+ \nabla\varphi_0 \nabla\varphi_0 \int_{E_3} \left(v^2 + \epsilon\right) \mathrm{d}x + \nabla\varphi_0 \nabla\varphi_0 \int_{E_4} \left(v^2 + \epsilon\right) \mathrm{d}x \\ &+ \nabla\varphi_2 \nabla\varphi_2 \int_{E_5} \left(v^2 + \epsilon\right) \mathrm{d}x + \nabla\varphi_1 \nabla\varphi_1 \int_{E_6} \left(v^2 + \epsilon\right) \mathrm{d}x \end{aligned}$$

j liegt rechts neben i

$$\begin{aligned} L_{i,i+1} &= \sum_{E \in E_k} \nabla T_i \nabla T_{i+1} \int_E \left(v^2 + \epsilon\right) \mathrm{d}x \\ &= \nabla\varphi_0 \nabla\varphi_1 \int_{E_3} \left(v^2 + \epsilon\right) \mathrm{d}x + \nabla\varphi_0 \nabla\varphi_1 \int_{E_6} \left(v^2 + \epsilon\right) \mathrm{d}x \end{aligned}$$

j liegt unter i

$$\begin{aligned} L_{i,i+1+n} &= \sum_{E\in E_k} \nabla T_i \nabla T_{i+1+n} \int_E \left(v^2+\epsilon\right) \mathrm{d}x \\ &= \nabla\varphi_0\nabla\varphi_1 \int_{E_4} \left(v^2+\epsilon\right) \mathrm{d}x + \nabla\varphi_0\nabla\varphi_1 \int_{E_5} \left(v^2+\epsilon\right) \mathrm{d}x \end{aligned}$$

j liegt schräg unter i

$$\begin{aligned} L_{i,i+2+n} &= \sum_{E\in E_k} \nabla T_i \nabla T_{i+2+n} \int_E \left(v^2+\epsilon\right) \mathrm{d}x \\ &= \nabla\varphi_1\nabla\varphi_2 \int_{E_5} \left(v^2+\epsilon\right) \mathrm{d}x + \nabla\varphi_1\nabla\varphi_2 \int_{E_6} \left(v^2+\epsilon\right) \mathrm{d}x \\ &= 0 \end{aligned}$$

Zusammenfassung Mit diesen Werten können wir die Matrix $\left(\int_\Omega (v^2+\epsilon)\nabla T_i \nabla T_j\right)_{ij}$ aufstellen:

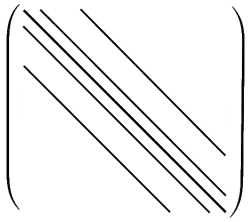


Abbildung 4.2.: Darstellung der Form der Matrix $\left(\int_\Omega (v^2+\epsilon)\nabla T_i \nabla T_j\right)_{ij}$

Dabei sind auf der Diagonalen die Einträge $L_{i,i}$, auf der Nebendiagonalen die Einträge $L_{i,i+1}$ und auf der anderen Diagonale die Einträge $L_{i,i+n+1}$. Wir haben bis jetzt immer nur über das Referenzdreieck integriert. Da wir aber eigentlich über die transformierten Dreiecke integrieren, müssen wir die Matrix mit dem Faktor $1/(h_1 h_2)$ multiplizieren.

4.2.3. Aggregation

Nun haben wir die Matrix L und den Vektor u_0^h gegeben, um das Gleichungssystem $Lu^h = Lu_0^h$ zu berechnen. Allerdings haben wir noch nicht eingebracht, dass auf $\Gamma_1 \cup \Gamma_2$ $u = 0$ gilt. Eigentlich würden wir das im Vektor u mit aufnehmen, also die Zeilen 0

setzen, die den Rand repräsentieren und dann das Gleichungssystem lösen. Dies geht numerisch jedoch nicht so einfach. Die Information muss in L und in Lu_0^h codiert sein. Dazu setzt man die Zeilen in L gleich 0, die zum Rand gehören. Die zugehörigen Diagonaleinträge werden 1 gesetzt. Diese Matrix nennen wir $\tilde{L}$. Die zugehörige Einträge in Lu_0^h, also die Einträge, die in der selben Zeile sind, setzt man 0. Den neuen Vektor nennen wir $\tilde{Lu_0^h}$ Dadurch erhält man, dass u^h an dieser Stelle 0 wird.

Damit haben wir beide Seiten diskretisiert und können das Gleichungssystem implementieren. Wir wollen

$$\frac{1}{h_1 h_2}\tilde{L}u = \frac{1}{h_1 h_2}\tilde{Lu_0^h} \Leftrightarrow \tilde{L}u = \tilde{Lu_0^h}$$

berechnen. Der Code dazu hat folgende Form

1. Berechne Matrix $\tilde{L}$
2. Berechne Vektor $\tilde{Lu_0^h}$
3. $u = \tilde{Lu_0^h} \backslash \tilde{L}$

Algorithm 4: Berechnung von u

Da sowohl $\tilde{L}$ als auch $\tilde{Lu_0^h}$ aus fast nur Nullen besteht, verwende ich in Matlab Sparse Matrizen, also dünn besetzte Matrizen, mit denen Matlab sehr effizient rechnen kann. Dies führt zu einer wesentlich besseren Laufzeit.

4.3. Optimierung nach v

Bei der Optimierung nach v geht es um die Fortsetzung des Risses. Für dieses Optimierungsproblem mit Ungleichungsnebenbedingung sicheren wir zunächst die Existenz und Eindeutigkeit der Lösung. Danach stellen wir die Optimalitätsbedingungen auf. Das resultiert in einem Karush Kuhn Tucker System. Dieses wollen wir mittels semidifferenzierbarer Newton Methode lösen. Dazu müssen zunächst alle Funktionen des KKT Systems differenziert und danach diskretisiert werden. Dies geschieht wieder mit Finiten Elementen. Am Schluss führen wir beide Optimierungsprobleme zu einem Verfahren zusammen.

4.3.1. Analytische Betrachtung

Die Optimierung nach v hat folgende Form:

$$\begin{aligned} &\min_{v\in H^1(\Omega)} \int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 + \nu\left(\epsilon_2|\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\right)\mathrm{d}x \\ &\text{s.d. } 0 \le v \le v_0 \end{aligned} \tag{4.4}$$

Das lässt sich allgemein als Optimierungsproblem mit Ungleichungsnebenbedingungen darstellen

$$\min_{w\in W} J(w) \quad \text{s.d.} \quad w \in C$$

wobei W ein Banachraum, $J : W \to \mathbb{R}$ Gâteaux-differenzierbar und $C \subset W$. In diesem Fall bedeutet das:

$$\begin{aligned} J: \quad H^1(\Omega) &\to \mathbb{R} \\ v &\mapsto \textstyle\int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 + \nu\left(\epsilon_2|\nabla v|^2 + \frac{1}{\epsilon_2}(1-v)^2\right)\mathrm{d}x \\ C := \{v \in H^1(\Omega) | 0 \le v \le v_0\} & \end{aligned}$$

Zunächst wollen wir die Existenz und Eindeutigkeit der Lösung zeigen. Dafür brauchen wir die Gâteaux differenzierbarkein von J.

Lemma 4.3.1. *J ist Gâteaux differenzierbar mit*

$$\begin{aligned} J'(v): \quad \overline{H^1}(\Omega) &\to \mathbb{R} \\ \psi_1 &\mapsto \int\limits_\Omega 2\psi_1 v|\nabla u|^2 + \nu\left(\epsilon_2 2\nabla v\nabla\psi_1 - \tfrac{2}{\epsilon_2}(1-v)\psi_1\right)\mathrm{d}x \end{aligned}$$

Beweis. Zunächst müssen wir die Richtungsableitung bestimmen.

$$
\begin{aligned}
\partial J(v)(\psi_1) &= \lim_{t\to 0} \tfrac{1}{t}\Big(\quad J(v+t\psi_1) - J(v)\Big) \\
&= \lim_{t\to 0} \tfrac{1}{t}\Big(\int_\Omega \left((v+t\psi_1)^2+\epsilon_1\right)|\nabla(u)|^2 \\
&\qquad +\nu\left(\epsilon_2|\nabla(v+t\psi_1)|^2 + \tfrac{1}{\epsilon_2}\left(1-(v+t\psi_1)\right)^2 \mathrm{d}x\right) \\
&\qquad -\int_\Omega \left(v^2+\epsilon_1\right)|\nabla u|^2 \\
&\qquad +\nu\left(\epsilon_2|\nabla v|^2 \qquad\qquad + \tfrac{1}{\epsilon_2}\left(1-v\right)^2\right)\mathrm{d}x\Big) \\
&= \lim_{t\to 0} \tfrac{1}{t}\Big(\int_\Omega \left(\left((v+t\psi_1)^2+\epsilon_1\right)-\left(v^2+\epsilon_1\right)\right)|\nabla(u)|^2 \\
&\qquad\quad +\nu\left(\epsilon_2\left(|\nabla(v+t\psi_1)|^2-|\nabla v|^2\right)\right. \\
&\qquad\quad \left.+\tfrac{1}{\epsilon_2}\left((1-v-t\psi_1)^2-(1-v)^2\right)\right)\mathrm{d}x\Big) \\
&= \lim_{t\to 0} \tfrac{1}{t}\Big(\int_\Omega \left(v^2+2vt\psi_1+t^2\psi_1^2-v^2\right)|\nabla u|^2 \\
&\qquad\quad +\nu\left(\epsilon_2\left(|\nabla v|^2+2t\nabla v\nabla\psi_1+t^2|\nabla\psi_1|^2-|\nabla v|^2\right)\right. \\
&\qquad\quad \left.+\tfrac{1}{\epsilon_2}\left((1-v)^2-2(1-v)t\psi_1+t^2\psi_1^2-(1-v)^2\right)\right)\Big) \\
&= \lim_{t\to 0}\Big(\int_\Omega \left(2v\psi_1+t\psi_1^2\right)|\nabla u|^2+\nu\left(\epsilon_2\left(2\nabla v\nabla\psi_1+t|\nabla\psi_1|^2\right)\right. \\
&\qquad\quad \left.-\tfrac{1}{\epsilon_2}\left(2(1-v)\psi_1+t\psi_1^2\right)\right)\mathrm{d}x\Big) \\
&= \int_\Omega 2\psi_1 v|\nabla u|^2+\nu\left(\epsilon_2 2\nabla v\nabla\psi_1-\tfrac{2}{\epsilon_2}(1-v)\psi_1\right)\mathrm{d}x \\
&= \int_\Omega 2v|\nabla u|^2\psi_1-\nu\left(\epsilon_2 2\Delta v\psi_1-\tfrac{2}{\epsilon_2}(1-v)\psi_1\right)\mathrm{d}x \\
&\qquad +\int_{\partial\Omega} 2\nu\epsilon_2\nabla v\nu\psi_1\,\mathrm{d}x
\end{aligned}
$$

Damit es auch eine Gâteaux Ableitung ist, muss sie beschränkt und linear sein. Dies ist einfach zu sehen. □

Theorem 4.3.2. *Das Problem* (4.4) *besitzt genau eine Lösung, falls v_0 stetig ist.*

Beweis. Wir wollen Theorem 3.1.6 anwenden. Zunächst müssen wir alle Voraussetzungen prüfen.

1. $W = H^1(\Omega)$ ist ein Hilbertraum, also auch ein reflexiver Banachraum.

2. Nun muss gezeigt werden, dass C nichtleer, abgeschlossen und konvex ist. C ist nichtleer, da $0 \in C$.

   Sei v_n eine konvergente Folge in C. Dann gilt $0 \leq v_n \leq v_0$ für alle $n \in \mathbb{N}$. Es gilt auch $0 \leq \lim_{n\to\infty} u_n \leq v_0$. Also ist C abgeschlossen.

Um Konvexität von C zu zeigen sei $0 < \lambda < 1$ und $v, w \in C$. Dann gilt $0 \leq \lambda v + (1-\lambda)w$, da $\lambda > 0$. Außerdem gilt $\lambda v + (1-\lambda)w \leq \lambda v_0 + (1-\lambda)v_0 = v_0$. Also ist jede Konvexkombination in C enthalten, C ist konvex.

3. J ist strikt konvex. Der Beweis dazu kann durch einfaches Nachrechnen geführt werden. Für Stetigkeit gilt dasselbe.

4. J ist Gâteaux differenzierbar nach (4.3.1)

5. Sei $w \in C$ mit $\|v\|_{H^1(\Omega)} \to \infty$. Dann gilt

$$\begin{aligned}
J(v) &= \int_\Omega (v^2 + \epsilon_1) |\nabla u|^2 + \nu \left(\epsilon_2 |\nabla v|^2 + \tfrac{1}{\epsilon_2} (1-v)^2 \right) \mathrm{d}x \\
&= \int_\Omega v^2 |\nabla u|^2 + \epsilon_1 |\nabla u|^2 + \nu \left(\epsilon_2 |\nabla v|^2 + \tfrac{1}{\epsilon_2} (1 - 2v + v^2) \right) \mathrm{d}x \\
&= \int_\Omega v^2 |\nabla u|^2 - \nu \left(\tfrac{2}{\epsilon_2} v + \tfrac{1}{\epsilon_2} v^2 \right) \mathrm{d}x + \int_\Omega \epsilon_1 |\nabla u|^2 + \nu \tfrac{1}{\epsilon_2} \, \mathrm{d}x \\
&\quad + \int_\Omega \nu \epsilon_2 |\nabla v|^2 \, \mathrm{d}x \\
&\leq \int_\Omega v^2 \left(|\nabla u|^2 + \nu \tfrac{1}{\epsilon_2} \right) \mathrm{d}x + c + \epsilon_2 \|\nabla v\|^2_{L^2(\Omega)} \\
&\leq c' \|v\|^2_{L^2(\Omega)} + c + \epsilon_2 \|\nabla v\|^2_{L^2(\Omega)} \\
&\leq c'' \left(\|v\|^2_{L^2(\Omega)} + \|\nabla v\|^2_{L^2(\Omega)} \right) + c \\
&\leq c'' \|v\|^2_{H^1(\Omega)} + c \\
&\to \infty
\end{aligned}$$

mit $c, c', c'' > 0$ passende Konstanten.

Alle Voraussetzungen aus Theorem 3.1.6 sind erfüllt, also existiert genau eine Lösung des Optimierungsproblems. □

Nachdem wir nun wissen, dass die Lösung existiert und eindeutig ist, wollen wir das Minimum finden. Dazu brauchen wir Optimalitätsbedingungen. Diese stellt das folgende Theorem auf:

Theorem 4.3.3 (Optimalitätsbedingungen)**.** *Sei* $a := \inf\{J(w) | H(w) \leq_P 0\}$. *Dann gilt:*

$$a = \inf_{v \in H^1(\Omega)} J(v) + \langle H(v), \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \rangle_{H^1(\Omega), H^{-1}(\Omega)}$$

mit

$$H : H^1(\Omega) \to H^1(\Omega)$$
$$v \mapsto \begin{pmatrix} -v \\ v - v_0 \end{pmatrix}$$

Beweis. Die Bedingungen aus 3.1.7 müssen gelten. Sei

$$P := \{(v, w) \in H^1(\Omega) \times H^1(\Omega) | v \geq 0 \text{ und } w \geq 0\} \subset H^1(\Omega) \times H^1(\Omega)$$

$\mathring{P} \neq \emptyset$, da $H^1(\Omega)$ nur stetige Funktionen enthält. Also ist P ein positiver Kegel.

$J : H^1(\Omega) \to \mathbb{R}$ sei wie oben definiert. H ist linear, also konvex.

Das Bild von J enthält ein $\hat{v}$, sodass $H(\hat{v}) <_P 0$ gilt, da es ein $v \in H^1(\Omega)$ geben muss, das echt zwischen 0 und v_0 liegt.

Außerdem ist $a := \inf\{J(w) | H(w) \leq_P 0\} < \infty$, da J stetig und beschränkt ist.

Also kann Theorem 3.1.7 angewendet werden. Damit existiert $(\mu, \lambda) \in H^{-1}(\Omega) \times H^{-1}(\Omega)$ mit $(\mu, \lambda) \geq 0$ komponentenweise, sodass

$$a = \inf_{v \in H^1(\Omega)} J(v) + \langle H(v), \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \rangle_{H^1(\Omega), H^{-1}(\Omega)}$$

□

Damit müssen wir nur noch das Minimum der Lagrangefunktion suchen. Dies funktioniert, indem wir die Ableitung bestimmen und 0 setzen. Wir leiten die Lagragefunktion ab und erhalten $\nabla J(v) + \lambda - \mu = 0$. Ausformuliert lautet das Problem:

$$2v|\nabla u|^2 - \epsilon_2 2\Delta v - \frac{2}{\epsilon_2}(1 - v) + \lambda - \mu = 0 \qquad \text{auf } \Omega$$
$$2\epsilon_2 \nabla v \nu = 0 \qquad \text{auf } \partial\Omega$$

Nun sind alle Voraussetzungen erfüllt, damit das KKT System aufgestellt werden kann.

$$\begin{aligned}
&2\overline{v}|\nabla u|^2 - \epsilon_2 2\Delta\overline{v} - \tfrac{2}{\epsilon_2}(1-\overline{v}) + \lambda - \mu = 0 \text{ auf } \Omega \\
&2\epsilon_2 \nabla\overline{v}\nu = 0 \text{ auf } \partial\Omega \\
&\overline{v} \geq a \quad \mu \geq 0 \quad \mu\overline{v} = 0 \\
&\overline{v} \leq b \quad \lambda \geq 0 \quad \lambda(v_0 - \overline{v}) = 0
\end{aligned}$$

Die Projektion für die Nebenbedingung lautet nach (3.5):

$$\mu - \lambda = \max\{0, \mu - \lambda + c(\overline{v} - v_0)\} + \min\{0, \mu - \lambda + c\overline{v}\} \forall c > 0$$

Daraus ergibt sich eine starke und schwache Formulierung. Die Starke lautet: Suche $v \in H^1$, sodass

$$\begin{aligned}
&2\overline{v}|\nabla u|^2 - \epsilon_2 2\Delta\overline{v} - \tfrac{2}{\epsilon_2}(1-\overline{v}) + \eta = 0 \text{ auf } \Omega \\
&2\epsilon_2 \nabla\overline{v}\nu = 0 \text{ auf } \partial\Omega \\
&\eta = \max\{0, \eta + c(\overline{v} - v_0)\} + \min\{0, \eta + c\overline{v}\} \ \forall c > 0
\end{aligned}$$

Die schwache Formulierung ist dann

$$\begin{aligned}
&\int_\Omega 2\psi_1 v|\nabla u|^2 + \epsilon_2 2\nabla v \nabla\psi_1 - \frac{2}{\epsilon_2}(1-v)\psi_1 + \eta\psi_1 \,\mathrm{d}x = 0 \\
&\int_\Omega \left(\eta - \max\{0, \eta + c(\overline{v} - v_0)\} - \min\{0, \eta + c\overline{v}\}\right)\psi_1 \,\mathrm{d}x = 0
\end{aligned}$$

$\forall c > 0, \forall \psi_1 \in H^1(\Omega)$ mit $\eta = \mu - \lambda$

4.3.2. Anwendung auf semidifferenzierbare Newton Methode

Unser Ziel ist es, eine Methode zu finden, mit der das KKT System lösen können. Betrachten wir also

$$\begin{aligned}
&G : H^1(\Omega) \times H^1(\Omega) \to H^{-1}(\Omega)^2 \\
&(v, \eta) \mapsto \begin{pmatrix} \int_\Omega 2\psi_1 v|\nabla u|^2 + \nu\left(\epsilon_2 2\nabla v\nabla\psi_1 - \frac{2}{\epsilon_2}(1-v)\psi_1 + \eta\psi_1\right)\mathrm{d}x \\ \int_\Omega \left(\eta - \max\{0, \eta + c(v - v_0)\} - \min\{0, \eta + cv\}\right)\psi_2 \,\mathrm{d}x \end{pmatrix}
\end{aligned}$$

Wir wollen (v, η) finden, sodass $G = 0$ ist. Direkt kann diese Formel nicht gelöst werden, da wir, um G_2 zu berechnen, (v, η) benötigen. Dies ist nicht gegeben. Also lösen wir das Problem mit einer Newton Methode. Für jeden Iterationsschritt ist (v, η) durch den vorherigen gegeben. Für die Newton Methode wird die Ableitung von G benötigt. Da G_2 offensichtlich keine Gâteaux-Ableitung hat, muss das Semidifferenzial berechnet werden. Daraus folgt, dass die semidifferenzierbare Newton Methode angewendet werden muss.

Sehen wir uns zunächst die $\partial G_1(v, \eta)(h)$ an. Es gilt:
Theorem 4.3.4. *$G_1(v, \eta)$ ist semidifferenzierbar mit*

$$\partial G_{1v}(v, \eta)(\phi_1) = \int_\Omega 2\psi_1\phi_1|\nabla u|^2 + \nu\left(\epsilon_2 2\nabla\phi_1\nabla\psi_1 + \frac{2}{\epsilon_2}\phi_1\psi_1\right)\mathrm{d}x$$
$$\partial G_{1\eta}(v, \eta)(\phi_2) = \int_\Omega \nu\phi_2\psi_1\,\mathrm{d}x$$

falls u fest gewählt ist, oder $u \in L^\infty$.

Beweis. Nach Lemma 3.4.7 ist G_1 ∂G_1 semidifferenzierbar, falls G_1 stetig Fréchet differenzierbar ist. Bestimmen wir zunächst die Richtungsableitung in Richtung ϕ_1 bzw ϕ_2. Diese ist gegeben durch

$$\partial G_{1v}(v, \eta)(\psi_1, \phi_1) = \int_\Omega 2\psi_1\phi_1|\nabla u|^2 + \nu\left(\epsilon_2 2\nabla\phi_1\nabla\psi_1 + \frac{2}{\epsilon_2}\phi_1\psi_1\right)\mathrm{d}x$$
$$G_{1\eta}(v, \eta)(\phi_2) = \int_\Omega \nu\phi_2\psi_1\,\mathrm{d}x$$

Die Berechnung wird hier nicht weiter ausgeführt. Die Richtungsableitungen müssen linear und beschränkt sein. Linearität ist einfach zu sehen. Für Beschränktheit von $\partial G_{1v}(v, \eta)(\psi_1, \phi_1)$ gilt:

$$\begin{aligned}\|\partial G_{1v}(v, \eta)(\psi_1, \phi_1)\|_{H^{-1}} \leq &= \int_\Omega 2\psi_1\phi_1|\nabla u|^2 + \nu\left(\epsilon_2 2\nabla\phi_1\nabla\psi_1 + \frac{2}{\epsilon_2}\phi_1\psi_1\right)\mathrm{d}x \\ &\leq 2\|\nabla u\|^2|(\psi_1, \phi_1)_{H^1}| + \nu\max\{2\epsilon_2, \frac{2}{\epsilon_2}\}|(\psi_1, \phi_1)_{H^1}| \\ &\leq \left(C + \tilde{C}\|\nabla u\|^2\right)\|\psi_1\|_{H^1}\|\phi_1\|_{H^1}\end{aligned}$$

Da $u \in L^\infty$, ist $\|\nabla u\|^2$ beschränkt. Damit ist G_{1v} beschränkt. Für Fréchet Differenzier-

barkeit muss eine Abschätzung überprüft werden:

$$
\begin{aligned}
&\|G_1(v+h,\eta) - G_1(v,\eta) - \partial G_{1v}(v,\eta)(h)\| \\
&= \| \int_\Omega 2\psi_1(v+h)|\nabla u|^2 + \nu\left(\epsilon_2 2\nabla(v+h)\nabla\psi_1 - \frac{2}{\epsilon_2}(1-(v+h))\psi_1 + \eta\psi_1\right)\mathrm{d}x \\
&\quad - \int_\Omega 2\psi_1 v|\nabla u|^2 + \nu\left(\epsilon_2 2\nabla v\nabla\psi_1 - \frac{2}{\epsilon_2}(1-v)\psi_1 + \eta\psi_1\right)\mathrm{d}x \\
&\quad - \int_\Omega 2\psi_1 h|\nabla u|^2 + \nu\left(\epsilon_2 2\nabla h\nabla\psi_1 + \frac{2}{\epsilon_2}h\psi_1\right)\mathrm{d}x\| \\
&= \| \int_\Omega 2\psi_1 h|\nabla u|^2 + \nu\left(\epsilon_2 2\nabla h\nabla\psi_1 + \frac{2}{\epsilon_2}h\psi_1\right)\mathrm{d}x - \int_\Omega 2\psi_1 h|\nabla u|^2 \\
&\quad + \nu\left(\epsilon_2 2\nabla h\nabla\psi_1 + \frac{2}{\epsilon_2}h\psi_1\right)\mathrm{d}x\| \\
&= 0
\end{aligned}
$$

Da G_{1v} beschränkt und linear ist, ist G_{1v} auch stetig. □

Für das Semidifferenzial von G_2 beweisen wir zunächst ein Lemma.

Lemma 4.3.5. *Betrachte $f : H^1(\Omega)^2 \to H^{-1}(\Omega)$ mit*

$$(\eta, v) \mapsto \eta - \max\{0, \eta + c(v - v_0)\} - \min\{0, \eta + cv\} \tag{4.5}$$

Dann ist f semidifferenzierbar mit

$$\frac{\partial f}{\partial \eta} = \begin{cases} \{0\} & \textit{falls } -c(v-v_0) < \eta \textit{ oder } \eta < -cv \\ \{1\} & \textit{falls } -cv < \eta < -c(v-v_0) \\ [0,1] & \textit{falls } -c(v-v_0) = \eta \textit{ oder } \eta = -cv \end{cases}$$

und

$$\frac{\partial f}{\partial v} = \begin{cases} \{-c\} & \textit{falls } -c(v-v_0) < \eta \textit{ oder } \eta < -cv \\ \{0\} & \textit{falls } -cv < \eta < -c(v-v_0) \\ [-c,0] & \textit{falls } -c(v-v_0) = \eta \textit{ oder } \eta = -cv \end{cases}$$

Beweis. f kann in einer anderen Form dargestellt werden

$$f(v,\eta) = \begin{cases} -c(v-v_0) & \text{falls } -c(v-v_0) \leq \eta \\ \eta & \text{falls } -cv < \eta < -c(v-v_0) \\ -cv & \text{falls } \eta \leq -cv \end{cases}$$

Die Äquivalenz dieser Form von f und (4.5), kann einfach nachgerechnet werden. Betrachten wir zunächst die Ableitung nach η. Es reicht, die Semidifferenzierbarkeit der einzelnen Abschnitte zu betrachten. Falls jeder Abschnitt semidifferenzierbar ist und die Übergänge auch, so ist f semidifferenzierbar.

Sei dazu $-c(v-v_0) < \eta$ oder $\eta < -cv$. Mit Lemma 3.4.7 gilt die ∂f Semidifferenzierbarkeit von f, falls f stetig Fréchet Differenzierbar ist. Um Fréchet Differenzierbarkeit zu zeigen, bestimmen wir zunächst die Richtungsableitung. Diese ist offensichtlich 0. Dadurch folgt sofort die Fréchet Differenzierbarkeit.

Sei nun $-cv < \eta < -c(v-v_0)$. Durch Lemma 3.4.7 müssen wir wieder die Fréchet Differenzierbarkeit überprüfen. Offensichtlich ist die Identität Fréchet differenzierbar. Das Differenzial ist hier 1.

Sei $\eta = -c(v-v_0)$. Sei zunächst $d > 0$. Die Abschätzung (3.12) muss gelten. Hier ist $\partial f(\eta + d, v) = \{0\}$ und damit

$$\begin{aligned} &\sup_{M \in \partial f(\eta+d,v)} \|f(\eta+d,v) - f(\eta) - Md\|_{H^{-1}(\Omega)} \\ &\quad = \| - c(v-v_0) + c(v-v_0)\|_{H^{-1}(\Omega)} = 0 = o\left(\|d\|_{H^1(\Omega)}\right) \text{ für } \|d\|_{H^1(\Omega)} \to 0 \end{aligned}$$

Sei nun $d < 0$. Da d nahe an 0 ist, gilt $d > -cv_0$ mit $v_0 > 0$. Es ist $\partial G_2^\eta(\eta + d) = \{1\}$ und damit

$$\begin{aligned} &\sup_{M \in \partial f(\eta+d,v)} \|f(\eta+d,v) - f(\eta,v) - Md\|_{H^{-1}(\Omega)} \\ &\quad = \| - c(v-v_0) + d + c(v-v_0) - d\|_{H^{-1}(\Omega)} = 0 = o\left(\|d\|_{H^1(\Omega)}\right) \end{aligned}$$

für $\|d\|_{H^1(\Omega)} \to 0$ Es fehlt noch $\eta = -cv$. Sei zunächst $d > 0$. Da d nahe an 0 ist, gilt auch $d < cv_0$. Es gilt $\partial f(\eta + d, v) = \{1\}$ und damit

$$\begin{aligned} &\sup_{M \in \partial G_2^\eta(\eta+d)} \|f(\eta+d,v) - f(\eta) - Md\|_{H^{-1}(\Omega)} \\ &\quad = \| - cv + d + cv - d\|_{H^{-1}(\Omega)} = 0 = o\left(\|d\|_{H^1(\Omega)}\right) \text{ für } \|d\|_{H^1(\Omega)} \to 0 \end{aligned}$$

Sei nun $d < 0$. Es gilt: Es gilt $\partial G_2^\eta(\eta + d) = \{0\}$ und damit

$$\begin{aligned}&\sup_{M \in \partial f(\eta+d,v)} \|f(\eta + d, v) - f(\eta, v) - Md\|_{H^{-1}(\Omega)} \\ &= \| - cv + cv\|_{H^{-1}(\Omega)} = 0 = o\left(\|d\|_{H^1(\Omega)}\right)\end{aligned}$$

für $\|d\|_{H^1(\Omega)} \to 0$. Damit ist f semidifferenzierbar nach η. Die Semidifferenzierbarkeit von v berechnet man analog. □

Das eigentliche Ziel war, das Semidifferenzial von G_2 zu finden. Dieses können wir nun tun.

Theorem 4.3.6. *$G_2 : H^1(\Omega)^2 \to H^{-1}(\Omega)$ mit*

$$(v, \eta) \mapsto \int_\Omega \left(\eta - \max\{0, \eta + c(v - v_0)\} - \min\{0, \eta + cv\}\right) \psi_2 \,\mathrm{d}x$$

ist semidifferenzierbar mit

$$\partial G_{2\eta}(\eta, v)(\psi_2, \phi_2) = \int_\Omega \frac{\partial f}{\partial \eta} \psi_2 \phi_2 \,\mathrm{d}x$$

$$\partial G_{2v}(\eta, v)(\psi_2, \phi_1) = \int_\Omega \frac{\partial f}{\partial v} \psi_2 \phi_1 \,\mathrm{d}x$$

Beweis. Es gilt

$$\begin{aligned}&\int_\Omega \left(\eta - \max\{0, \eta + c(v - v_0)\} - \min\{0, \eta + cv\}\right) \psi_2 \,\mathrm{d}x \\ &= \int_\Omega \eta\psi_2 \,\mathrm{d}x - \int_\Omega \max\{0, \eta + c(v - v_0)\}\psi_2 \,\mathrm{d}x - \int_\Omega \min\{0, \eta + cv\}\psi_2 \,\mathrm{d}x \\ &= F_1(v, \eta)(\psi_2) - F_2(v, \eta)(\psi_2) - F_3(v, \eta)(\psi_2)\end{aligned}$$

Wir können nach 3.4.8 F_1, F_2, F_3 getrennt ableiten. Betrachten wir die Ableitung nach η. Die Ableitung von F_1 ist einfach

$$\partial F_{1\eta}(\psi_2, \phi_2) = \int_\Omega \psi_2 \phi_2 \,\mathrm{d}x$$

F_2 und F_3 lassen sich mithilfe der Kettenregel aus 3.4.8 ableiten

$$\begin{aligned}
\partial F_{2\eta}(\psi_2, \phi_2) &= \int_\Omega \frac{\partial}{\partial \eta + c(v - v_0)} \max\{0, \eta + c(v - v_0)\} \frac{\partial \eta}{\partial \eta} \psi_2 \, \mathrm{d}x \\
&= \int_\Omega \frac{\partial f_2}{\partial \eta} \psi_2 \phi_2 \, \mathrm{d}x \\
\partial F_{3\eta}(\psi_2, \phi_2) &= \int_\Omega \frac{\partial}{\partial \eta + cv} \min\{0, \eta + cv\} \frac{\partial \eta}{\partial \eta} \psi_2 \, \mathrm{d}x \\
&= \int_\Omega \frac{\partial f_3}{\partial \eta} \psi_2 \phi_2 \, \mathrm{d}x
\end{aligned}$$

mit f_1 und f_2 der min bzw. der max Term. Die Ableitungen kann man davon einfach berechnen. Es ergibt sich:

$$\begin{aligned}
\partial G_{2\eta}(\eta, v)(\psi_2, \phi_2) &= \partial F_1(\eta, v)(\psi_2, \phi_2) + \partial F_2(\eta, v)(\psi_2, \phi_2) + \partial F_3(\eta, v)(\psi_2, \phi_2) \\
&= \int_\Omega \psi_2 \phi_2 \, \mathrm{d}x + \int_\Omega \frac{\partial f_2}{\partial \eta} \psi_2 \phi_2 \, \mathrm{d}x \int_\Omega \frac{\partial f_3}{\partial \eta} \psi_2 \phi_2 \, \mathrm{d}x \\
&= \int_\Omega (id + \frac{\partial f_2}{\partial \eta} + \frac{\partial f_3}{\partial \eta}) \phi_2 \psi_2 \, \mathrm{d}x \\
&= \int_\Omega \frac{\partial f}{\partial \eta} \psi_2 \phi_2 \, \mathrm{d}x
\end{aligned}$$

Es fügen sich die Ableitungen von f_2, f_3 und η wieder zu $\frac{\partial f}{\partial \eta}$ zusammen. □

Damit ergibt sich als Ableitung

$$G'(v, \eta) = \begin{pmatrix} G_{1v} & G_{1\eta} \\ G_{2v} & G_{2\eta} \end{pmatrix}$$

Also lautet das Gleichungssystem, das für das Newtonverfahren nach s gelöst werden muss

$$-\begin{pmatrix} G_1 \\ G_2 \end{pmatrix} = \begin{pmatrix} G_{1v} & G_{1\eta} \\ G_{2v} & G_{2\eta} \end{pmatrix} \begin{pmatrix} s^1 \\ s^2 \end{pmatrix}$$

Stellen wir dazu die Newton Methode auf:

Data: v^0, η^0 möglichst nah an der Lösung $\overline{v}, \overline{\eta}$
for $k = 0, 1, \cdots$ **do**
$\quad$ *Erhalte* s_1^k *beim Lösen von* $-\begin{pmatrix} G_1 \\ G_2 \end{pmatrix} = \begin{pmatrix} G_{1v} & G_{1\eta} \\ G_{2v} & G_{2\eta} \end{pmatrix} \begin{pmatrix} s^1 \\ s^2 \end{pmatrix}$;
$\quad v^{k+1} = v^k + s_1^k \quad \eta^{k+1} = \eta^k + s_2^k$;
end

Algorithm 5: semidifferenzeirbare Newton Methode

4.3.3. Numerische Betrachtung

Alle Funktionen aus dem Newtonsystem müssen numerisch dargestellt werden.

Für die Diskretisierung wird das selbe Gitter und die selben Elemente genommen wie bei der Optimierung nach u. Auch hier werden wir wieder mit dem Galerkin Ansatz arbeiten, d.h.

$$v = \sum_{i=1}^{k} v_i^h T_i \qquad \eta = \sum_{i=1}^{k} \eta_i^h T_i \qquad v_0 = \sum_{i=1}^{k} {v_0}_i^h T_i$$

wobei die T_i wieder die globalen Formfunktionen sind.

Da u gegeben ist, ist u ein Vektor mit den Auswertungen an den Ecken der Dreiecke. Die Darstellung ist die gleiche wie in (3.9) und (3.9). Also gilt

$$|\nabla u|^2 = (u_{31} - u_{21})^2 + (u_{11} - u_{21})^2 + (u_{32} - u_{22})^2 + (u_{12} - u_{22})^2 =: u^{dis}$$

Numerische Darstellung von G_1

Betrachten wir

$$G_1(v, \eta) = \int_\Omega 2\psi_1 v |\nabla u|^2 + \nu \left(\epsilon_2 2 \nabla v \nabla \psi_1 - \frac{2}{\epsilon_2}(1 - v)\psi_1 + \right) \eta \psi_1 \, \mathrm{d}x$$

Die Diskretisierung lautet:

$$
\begin{aligned}
&\int_\Omega 2\psi_1 v|\nabla u|^2 + \nu\left(2\epsilon_2\nabla v\nabla\psi_1 - \frac{2}{\epsilon_2}(1-v)\psi_1\right) + \eta\psi_1\,\mathrm{d}x \\
&= \int_\Omega 2T_j\sum_{i=1}^{k} v_i^h T_i u^{dis} + \nu\left(\epsilon_2 2\sum_{i=1}^{k} v_i^h\nabla T_i\nabla T_j - \frac{2}{\epsilon_2}(1-\sum_{i=1}^{k} v_i^h T_i)T_j\right) \\
&\quad + \sum_{i=1}^{k}\eta_i^h T_i T_j\,\mathrm{d}x \\
&= \sum_{i=1}^{k} v_i^h\left(2\int_\Omega u^{dis}T_iT_j\,\mathrm{d}x + 2\epsilon_2\nu\int_\Omega\nabla T_i\nabla T_j\,\mathrm{d}x + \nu\frac{2}{\epsilon_2}\int_\Omega T_iT_j\,\mathrm{d}x\right) \\
&- \nu\frac{2}{\epsilon_2}\sum_{i=1}^{k}\int_\Omega T_j\,\mathrm{d}x + \sum_{i=1}^{k}\eta_i^h\int_\Omega T_iT_j\,\mathrm{d}x \\
&= (2A + \nu 2\epsilon_2 B + \nu\frac{2}{\epsilon_2}D)v^h - \nu\frac{2}{\epsilon_2}D*e + D\eta^h
\end{aligned}
$$

mit $e := (1,\cdots,1)^T v^h := (v_1^h,\cdots v_k^h)^T$, $\eta^h := (\eta_1^h,\cdots\eta_k^h)^T$, $A_{ij} = \int_\Omega u^{dis}T_iT_j\,\mathrm{d}x$, $B_{ij} := \int_\Omega \nabla T_i\nabla T_j\,\mathrm{d}x$, $c_j := \int_\Omega T_j\,\mathrm{d}x$ und $D_{ij} := \int_\Omega T_iT_j\,\mathrm{d}x$.

Die Berechnung der Matrizen A, B, D erfolgt im Anhang A.1. Es ergibt sich

$$(2A + 2\nu\epsilon_2 B + \frac{2\nu}{\epsilon_2}D)v^h - \frac{2\nu}{\epsilon_2}De + D\eta^h =$$

$$
\left(2\left(\begin{array}{ccccc} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{array}\right) + 2\epsilon\left(\begin{array}{ccccc} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{array}\right) + \frac{2}{\epsilon}\left(\begin{array}{ccccc} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{array}\right)\right)v^h - \frac{2}{\epsilon}\left(\begin{array}{ccccc} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{array}\right)\left(\begin{array}{c}1\\1\\1\\1\\1\end{array}\right) + \left(\begin{array}{ccccc} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{array}\right)\eta^h
$$

wobei bei A auf der Hauptdiagonalen $\frac{1}{12}(\sum_{i=1}^{6} u_{E_i}^{dis})$ steht, auf der Nebendiagonalen steht $\frac{1}{24}(u_{E_3}^{dis} + u_{E_6}^{dis})$, auf der zweiten Nebendiagonalen $\frac{1}{24}(u_{E_4}^{dis} + u_{E_5}^{dis})$ und auf der dritten Nebendiagonalen $\frac{1}{24}(u_{E_5}^{dis} + u_{E_6}^{dis})$.

Bei B steht auf der Hauptdiagonalen 4, auf der Nebendiagonalen und der zweiten Nebendiagonalen -1.

Bei D steht auf der Hauptdiagonalen $\frac{1}{2}$, auf der ersten, zweiten und dritten Nebendiagonalen $\frac{1}{12}$.

Durch die noch ausstehende Transformation der Dreiecke, muss der gesamte Term mit $1/h_1h_2$ multipliziert werden.

Numerische Darstellung von G_2

$$\int_\Omega (\eta - \max\{0, \eta + c(v - v_0)\} - \min\{0, \eta + cv\})\, \psi_2 \,\mathrm{d}x$$

Um dieses Funktional numerisch darzustellen, benutzen wir den Galerkin Ansatz mit

$$v = \sum_{i=1}^{k} v_i^h T_i(x,y) \qquad \eta = \sum_{i=1}^{k} \eta_i^h T_i(x,y) \qquad v_0 = \sum_{i=1}^{k} {v_0}_i^h T_i(x,y)$$

Daraus ergibt sich:

$$\begin{aligned}
&\int_\Omega (\eta - \max\{0, \eta + c(v - v_0)\} - \min\{0, \eta + cv\})\, \psi_2 \,\mathrm{d}x \\
&= \int_\Omega \left(\sum_{i=1}^{k} \eta_i^h T_i - \max\left\{0, \sum_{i=1}^{k} \eta_i^h T_i + c(\sum_{i=1}^{k} v_i^h T_i - \sum_{i=1}^{k} {v_0}_i^h T_i)\right\} \right. \\
&\qquad \left. - \min\left\{0, \sum_{i=1}^{k} \eta_i^h T_i + c\sum_{i=1}^{k} v_i^h T_i\right\} \right) T_j \,\mathrm{d}x \\
&= \int_\Omega \left(\sum_{i=1}^{k} \eta_i^h T_i - \sum_{i=1}^{k} \max\left\{0, \eta_i^h + c(v_i^h - {v_0}_i^h)\right\} T_i \right. \\
&\qquad \left. - \sum_{i=1}^{k} \min\left\{0, \eta_i^h + cv_i^h\right\} T_i \right) T_j \,\mathrm{d}x \\
&= \int_\Omega \sum_{i=1}^{k} \left(\eta_i^h - \max\left\{0, \eta_i^h + c(v_i^h - {v_0}_i^h)\right\} - \min\left\{0, \eta_i^h + cv_i^h\right\}\right) T_i T_j \,\mathrm{d}x \\
&= \left(\sum_{i=1}^{k} \eta_i^h - \max\left\{0, \eta_i^h + c(v_i^h - {v_0}_i^h)\right\} - \min\left\{0, \eta_i^h + cv_i^h\right\} \right) \int_\Omega T_i T_j \,\mathrm{d}x \\
&= Dw_{v\eta}
\end{aligned}$$

mit D aus der numerischen Darstellung von G_1 und
$(w_{v\eta})_i := \eta_i^h - \max\left\{0, \eta_i^h + c(v_i^h - {v_0}_i^h)\right\} - \min\left\{0, \eta_i^h + cv_i^h\right\}$. Die Summe und T_i dürfen aus dem max bzw. min herausgezogen werden, da T_i immer nur an einem Punkt un-

gleich 0 ist. Dadurch kommt niemals zustande, dass mehr als ein Term der Summe ungleich 0 ist. $w_{v\eta}$ kann auch explizit dargestellt werden:

$$(w_{v\eta})_i = \begin{cases} -c(v_i^h - {v_0}_i^h) & \text{falls } -c(v_i^h - {v_0}_i^h) \leq \eta_i^h \\ \eta_i^h & \text{falls } -cv_i^h < \eta < -c(v_i^h - {v_0}_i^h) \\ -cv_i^h & \text{falls } \eta_i^h \leq -cv_i^h \end{cases}$$

Auch hier muss das Integral wieder transformiert werden, wodurch der Faktor $1/h_1h_2$ multipliziert wird.

Numerische Darstellung von G_{1v}

$$G_{1v}(v,\eta) = \int_\Omega 2\psi_1\phi_1|\nabla u|^2 + \nu 2\epsilon_2\nabla\phi_1\nabla\psi_1 + \frac{2\nu}{\epsilon_2}\phi_1\psi_1 \,\mathrm{d}x$$

wird nun diskretisieren:

$$\begin{aligned} &\int_\Omega 2\psi_1\phi_1|\nabla u|^2 + 2\nu\epsilon_2\nabla\phi_1\nabla\psi_1 + \frac{2\nu}{\epsilon_2}\phi_1\psi_1 \,\mathrm{d}x \\ &= \int_\Omega 2T_jT_iu^{dis} + 2\nu\epsilon_2\nabla T_i\nabla T_j + \frac{2\nu}{\epsilon_2}T_iT_j \,\mathrm{d}x \\ &= 2A + 2\nu\epsilon_2 B + \frac{2\nu}{\epsilon_2}D \end{aligned}$$

Wir benutzen die gleichen Notationen, wie bei der numerischen Darstellung von G_1 und die gleiche Transformation.

Numerische Darstellung von $G_{1\eta}$

$$G_{1\eta}(v,\eta) = \int_\Omega \phi_2\psi_1 \,\mathrm{d}x$$

wird nun diskretisieren:

$$\int_\Omega \phi_2\psi_1 \,\mathrm{d}x = \int_\Omega T_jT_i = D$$

Auch hier muss wieder transformiert werden.

Numerische Darstellung von G_{2v}

Es soll

$$\partial G_{2v}(\eta, v)(\psi_2, \phi_1) = \int_\Omega \frac{\partial f}{\partial v}\psi_2\phi_1 \,\mathrm{d}x$$

mit

$$\frac{\partial f}{\partial v} = \begin{cases} \{-c\} & \text{falls } -c(v-v_0) < \eta \text{ oder } \eta < -cv \\ \{0\} & \text{falls } -cv < \eta < -c(v-v_0) \\ [-c, 0] & \text{falls } -c(v-v_0) = \eta \text{ oder } \eta = -cv \end{cases} .$$

numerisch dargestellt werden. Statt $\frac{\partial f}{\partial v}$ implementieren wir eine Vereinfachung, die nicht mehr Mengenwertig ist. Dazu wählen wir statt $[-c, 0]$ einen Punkt aus dem Intervall, z.B. $-c/2$. Nun kann $\frac{\partial f}{\partial v}$ diskretisiert werden zu f^h. Dies ist einfach die Funktion ausgewertet an den Gitterpunkten.

Nun wird $\partial G_{2v}(\eta, v)(\psi_2, \phi_1)$ diskretisiert. Hier wird wie immer ψ_2, ϕ_1 durch die globalen Formfunktionen T_i ersetzt und Ω durch die Vereinigung aller Dreiecke. Nun kann für jedes Dreieck $\int_E \frac{\partial f}{\partial v}T_iT_j \,\mathrm{d}x$ berechnet werden. Dies erfolgt im Anhang A.1.

Wir erhalten wieder eine Matrix folgender Form:

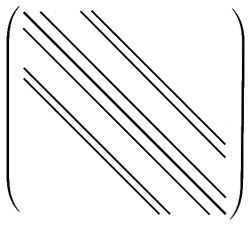


Abbildung 4.3.: Darstellung der Matrix $\left(\int_\Omega (v^2+\epsilon)\nabla T_i \nabla T_j\right)_{ij}$

Jeder Eintrag, der nicht 0 ist, besteht aus der Summe von unterschiedlichen Auswertungen der Funktion f auf den Dreiecken, über denen integriert wurde.

Numerische Darstellung von $G_{2\eta}$

Die numerische Darstellung ist genau die gleiche, wie bei G_{2v}, nur dass die Funktionsauswertungen von f andere sind.

4.3.4. Aggregation

Nun sind alle Funktionen diskretisiert und das Problem kann implementiert werden. Stellen wir die genaue Newton Methode auf.

Data: v^0, η^h (möglichst nah an der Lösung $(\overline{v}, \overline{\eta})$)
for $k = 0, 1, \cdots$ **do**
  Löse das Gleichungssytem $Lu^k = Lu_0^k$ *nach* u^k ;
  Erhalte s_k *beim Lösen von*
  $$-\begin{pmatrix} G_1(v^k, \eta^k) \\ G_2(v^k, \eta^k) \end{pmatrix} = \begin{pmatrix} \partial G_{1v}(v^k, \eta^k)(T_i) & \partial G_{1\eta}(v^k, \eta^k)(T_i) \\ \partial G_{2v}(v^k, \eta^k)(T_i) & \partial G_{2\eta}(v^k, \eta^k)(T_i) \end{pmatrix} \begin{pmatrix} s_1^k \\ s_2^k \end{pmatrix};$$
  $v^{k+1} = v^k + s_1^k \quad \eta^{k+1} = \eta^k + s_2^k;$
end

Algorithm 6: semidifferenzierbare Newton Methode

mit

$$\begin{aligned}
L &= \int_\Omega (v^2 + \epsilon_1) \nabla T_i \nabla T_j \, \mathrm{d}x \\
G_1 &= \left(2(\int_\Omega u^{dis} T_i T_j \, \mathrm{d}x)_{ij} + 2\nu\epsilon_2 (\int_\Omega \nabla T_i \nabla T_j \, \mathrm{d}x)_{ij} + \tfrac{2\nu}{\epsilon_2} (\int_\Omega T_i T_j \, \mathrm{d}x)_{ij} \right) v^k \\
&\quad -\tfrac{2\nu}{\epsilon_2} (\int_\Omega T_j \, \mathrm{d}x)_j + (\int_\Omega T_i T_j \, \mathrm{d}x)_{ij} \eta^k \\
G_2 &= (\int_\Omega T_i T_j \, \mathrm{d}x)_{ij} (\eta_i^h - \max\left\{0, \eta_i^h + c(v_i^h - {v_0}_i^h)\right\} - \min\left\{0, \eta_i^h + c v_i^h\right\})_i \\
\partial G_{1v} &= 2(\int_\Omega u^{dis} T_i T_j \, \mathrm{d}x)_{ij} + 2\nu\epsilon_2 (\int_\Omega \nabla T_i \nabla T_j \, \mathrm{d}x)_{ij} + \tfrac{2\nu}{\epsilon_2} (\int_\Omega T_i T_j \, \mathrm{d}x)_{ij} \\
\partial G_{1\eta} &= (\int_\Omega T_i T_j \, \mathrm{d}x)_{ij} \\
\partial G_{2v} &= (\int_\Omega \tfrac{\partial f}{\partial v} T_i T_j)_{ij} \\
\partial G_{2\eta} &= (\int_\Omega \tfrac{\partial f}{\partial \eta} T_i T_j)_{ij} \\
f &= \eta - \max\{0, \eta + c(v - v_0)\} - \min\{0, \eta + cv\}
\end{aligned}$$

5. Numerische Resultate

Zuerst betrachten wir die Implementierung der Optimierung. Wir werden darauf eingehen, wie man die Parameter im Code A.2 wählen muss. Danach werde ich einige Beispiele für unterschiedliche Gitterweiten, verschiedene Risse und Möglichkeiten des Einspannens des Materials betrachten.

5.1. Justieren der Parameter im Code

Am Anfang ist zu sagen, dass wir nur Risse betrachten, die mindestens zwei Gitterpunkte breit sind. Wenn ein Riss nur einen Gitterpunkt breit ist, wird v nicht konstant 0, was dazu führt, dass in der Implementation nie ein vollständiger Riss vorhanden ist.

Außerdem muss u_0 an beiden Rändern unterschiedlich sein. Wenn u_0 an beiden Rändern gleich ist, ist die Verschiebung des Materials überall gleich, d.h. dass nur eine Translation stattgefunden hat. Selbst wenn wir einen Riss einfügen, werden wir diesen nicht sehen.

Zur Wahl von der Konstanten c, die durch den Lagrangemultiplikator η hinzugekommen ist: Wählt man c zu groß und betrachtet

$$f(v,\eta) = \eta - \max\{0, \eta + c(v - v_0)\} - \min\{0, \eta + cv\}$$

werden das Maximum und das Minimum 0, da $v - v_0$ negativ ist und v positiv ist. Es gilt $f(v,\eta) = \eta$. Nach dem Newtonverfahren ist $-G_2 = \partial G_{2v} s_1^k + \partial G_{2\eta} s_2^k$ mit

$$G_2 = (\int_\Omega T_i T_j \,\mathrm{d}_x)_{ij}\eta^h \qquad \partial G_{2v} = 0 \qquad \partial G_{2\eta} = (\int_\Omega T_i T_j \,\mathrm{d}x)_{ij}$$

Daraus folgt $\eta^h = -s^k$. Außerdem gilt $\eta^{k+1} = \eta^k + s^k = 0$. Dadurch fallen nach dem ersten Iterationsschritt alle Terme mit η weg. Also berücksichtigen wir die vorgegebene

Bedingung von $0 < v < v_0$ nicht mehr. Diese soll aber explizit berücksichtigt werden. Also sollte man c nicht zu groß wählen. Auch wenn c zu klein gewählt wird, beachten wir die Schranke nicht, d.h. sinnvollerweise sollte man ungefähr $c = 1$ wählen.

Die richtige Wahl von η am Anfang zu finden, ist gar nicht so einfach. η kann positiv und negativ sein, da $\eta = \lambda - \mu$ ist, wobei $\lambda > 0, \mu > 0$ Lagrangemultiplikatoren sind. Experimentell findet man heraus, dass $\eta = 1$ kein Ergebnis liefert, bei $\eta \geq 2$ fast kein Übergang zwischen Riss und nicht Riss ist, bei $0,4 < \eta < 1$ ist das Papier nicht richtig gerissen, also v wird nie ganz 0 und/oder das Papier ist nie ganz heile, d.h. 1 wird nie angenommen. Die Wahl von $\eta = 0,34$ ist für manche Probleme sehr gut, für andere nicht. Erst im negativen Bereich erhalten wir für alle Probleme gute Werte. Hier muss aber $\eta > -1$ gelten.

Kommen wir zur Wahl von v_0. Es sollte Sinn ergeben, v_0 konstant in einer Umgebung von 1 zu wählen, da $v = 1$ schon bedeutet, dass kein Riss vorhanden ist. Kleiner sollte es nicht gewählt werden, da sonst nicht zugelassen ist, dass das Gebiet nicht gerissen ist. Dies ist leider in der Implementation nicht zu sehen. Bei $v_0 > 0,7$ erhalten wir Ergebnisse, die mit den erwarteten Ergebnissen übereinstimmen. Damit ist auch $v > v_0$, was im Widerspruch zu der Nebenbedingung $0 < v < v_0$ steht. Außerdem verändern auch große Werte von v_0 nichts am Ergebnis. Dieses Verhalten lässt überlegen, ob es überhaupt sinnvoll ist, diese Bedingung mit in das Problem zu nehmen. Als weiteren Ansatz könnte die Rissentstehung auch ohne Ungleichungsnebenbedingungen getestet werden.

5.2. Beispiele

Mit dem implementierten Algorithmus können wir nun einige Beispiele betrachten. Dazu betrachten wir zwei unterschiedliche Grafiken, z.B. 5.2 und 5.1. In einer Grafik ist die Verschiebung des Gebietes dargestellt, in der anderen, ob und wie stark ein Riss vorhanden ist. Bei beiden Grafiken ist jeweils das gesamte Gebiet, hier ein rechteckiges Gebiet in 2D auf der x und y Koordinate dargestellt. Die z Koordinate gibt bei der Verschiebung an, wie stark das Gebiet verschoben ist und bei dem Riss, wie stark der jeweilige Gitterpunkt eingerissen ist. Ein Wert in der Nähe von 1 bedeutet, dass das Gebiet nicht gerissen ist und 0, dass es vollständig gerissen ist.

Bei beiden Grafiken sind je 8 Bilder zu sehen. Diese geben an, wie viele Iterationsschritte der Newton Algorithmus durchgelaufen ist. Das erste Bild (also links oben) gibt das

Ergebnis nach der ersten Iteration an. Das zweite Bild, rechts neben dem ersten Bild, gibt das Ergebnis nach 2 Schritten aus. Dann folgen 5, 10, 20, 50, 100 und zum Schluss, rechts unten, 200 Iterationsschritte.

5.2.1. Der Riss im Material ist durchgängig

Beim ersten Beispiel ist das Gebiet an der rechten und linken Seite konstant eingespannt. Hier ist u am linken Rand immer 1 und am rechten Rand immer 2. Am Anfang wurde ein Riss durch das gesamte Gebiet eingefügt, der parallel zum eingespannten Rand verläuft. Dieser befindet sich genau in der Mitte des Gebietes. Dieses sieht man auch in dem ersten Bild von 5.2.

Betrachtet man den Riss im Verlauf der Iterationen, sieht man, dass der Riss an der gleichen Stelle bleibt und geglättet wird. Manchmal schwankt das Gebiet, das nicht gerissen ist, zwischen $0,9$ und $1,1$. Die Verschiebung des Gebietes 5.1 sieht ähnlich aus. An der Stelle des Risses, ist eine Unstetigkeitsstelle zu finden. Diese zieht sich durch alle Bilder. Die Veränderung im Verlauf der Iterationen ist zum einen, dass die Unstetigkeit geglättet wird. Zum anderen ist die Verschiebung, die nicht beim Riss ist, linear und mit steigender Iterationszahl fast konstant.

Die Glättung des Risses, und damit auch die Glättung der Unstetigkeitsstelle bei der Verschiebung im Gebiet, entsteht durch den $|\nabla v|^2$ Term im Optimierungsproblem, wie am Anfang erläutert. Ebenfalls ist die zunächst lineare und dann annähernd konstante Verschiebung hierauf zurückzuführen. Der Riss breitet sich nicht weiter aus, da das Gebiet nicht weiter verschoben wird. Dieses Szenario liefert realitätsnahe Ergebnisse.

Betrachten wir das gleiche Szenario bei einem 10×10 Gitter wie in 5.3 und 5.4. Hier gibt es andere Ergebnisse. Der Riss verschiebt sich zum Rand, an dem $v = 1$ gilt. Vermutlich liegt die Verschiebung des Risses daran, dass der Riss sehr groß im Vergleich zu dem Gitter ist. Dadurch wird das gesamte Papier gerissen, was niedriger eingespannt ist.

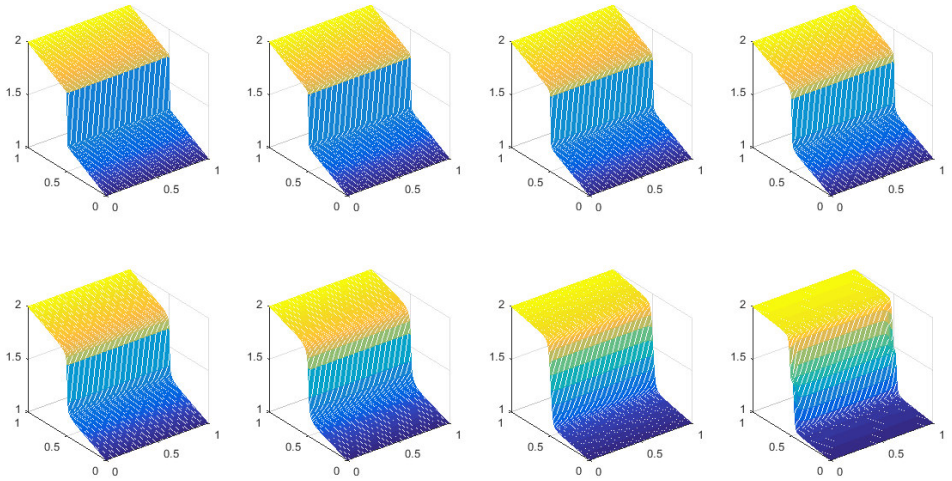


Abbildung 5.1.: Darstellung der Verschiebung des Gebietes bei konstantem u_0, einem Riss in der Mitte und 100×100 Gitterpunkten

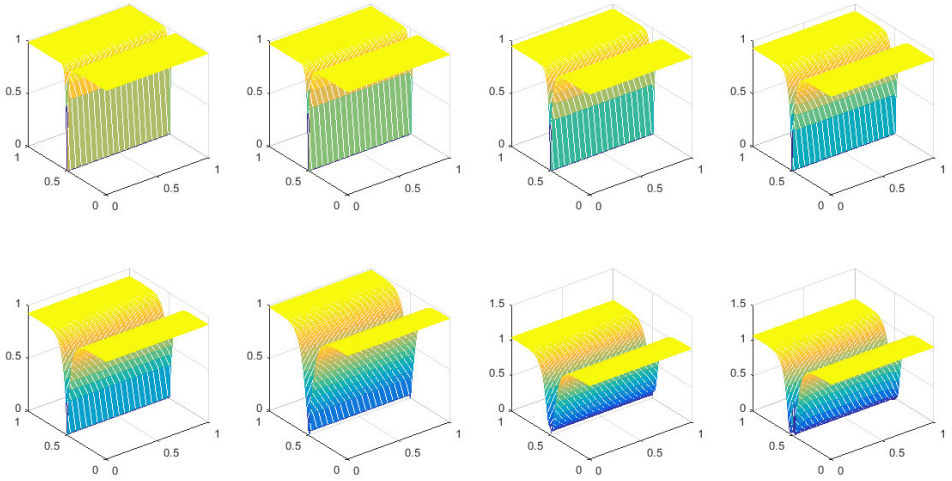


Abbildung 5.2.: Darstellung des Risses bei konstantem u_0, einem Riss in der Mitte und 100×100 Gitterpunkten

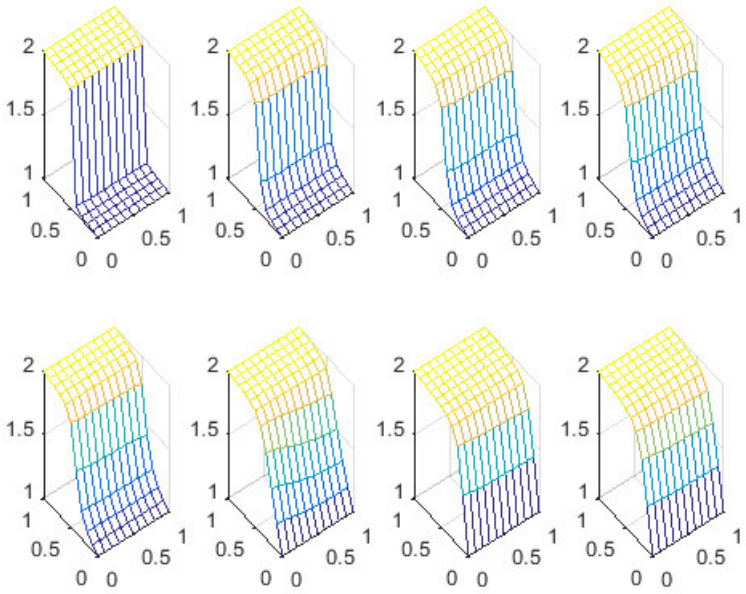


Abbildung 5.3.: Darstellung der Verschiebung des Gebietes bei konstantem u_0, einem Riss in der Mitte und 10×10 Gitterpunkten

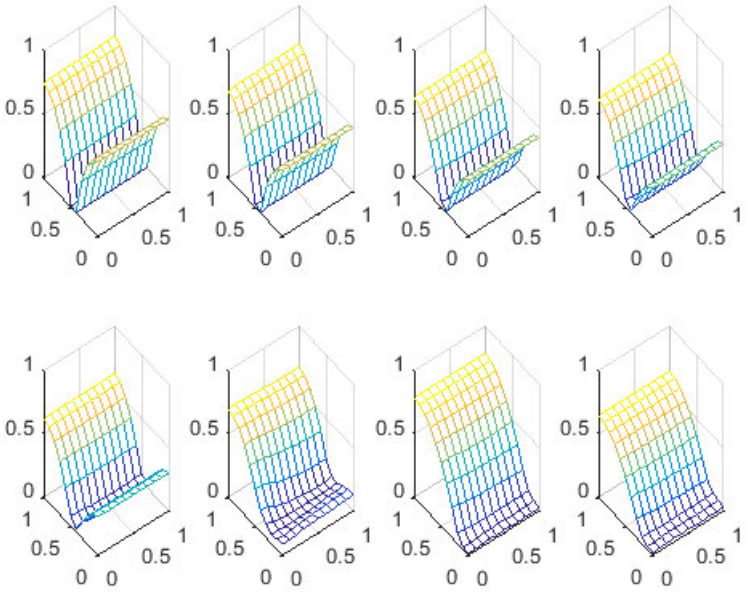


Abbildung 5.4.: Darstellung des Risses bei konstantem u_0, einem Riss in der Mitte und 10×10 Gitterpunkten

5.2.2. Das Material ist am Rand angerissen

Beim nächsten Experiment betrachten wir das gleiche u wie vorhin, aber wir reißen das Papier nur an einer Seite ein wenig an. Dieser Riss ist 10 Gitterpunkte lang auf einem 100×100 Gitter. Abbildung 5.5 und 5.6 zeigen dieses Szenario.

Betrachten wir zunächst die Verschiebung des Gebietes. Hier sieht man nur dort eine Unstetigkeitsstelle, wo am Anfang der Riss ist. Diese Unstetigkeitsstelle wird länger und breitet sich parallel zur Befestigung aus. Am Ende, bei 200 Iterationen, ist im gesamten Papier der Riss zu sehen. Die Unstetigkeitsstelle wird wieder glatter, je mehr Iterationen das Programm durchgeführt hat.

Auch die Ausbreitung des Risses verläuft fast identisch. Während der ersten 20 Iterationen ist kaum eine Veränderung festzustellen. Bei der 50. Iteration liegt der Wert von v bei 0,7 dort, wo das Extremum nicht angenommen wird. Bei späteren Iterationen zieht sich der Riss wie erwartet durch die gesamte Oberfläche.

Im Groben zeigt der Algorithmus das erwartete Verhalten, d.h. der Riss breitet sich parallel zur Befestigung aus. Die Anomalie bei Iteration 50 könnte eine Ungenauigkeit in meinem Algorithmus oder der zu Grunde liegenden Modellierung sein.

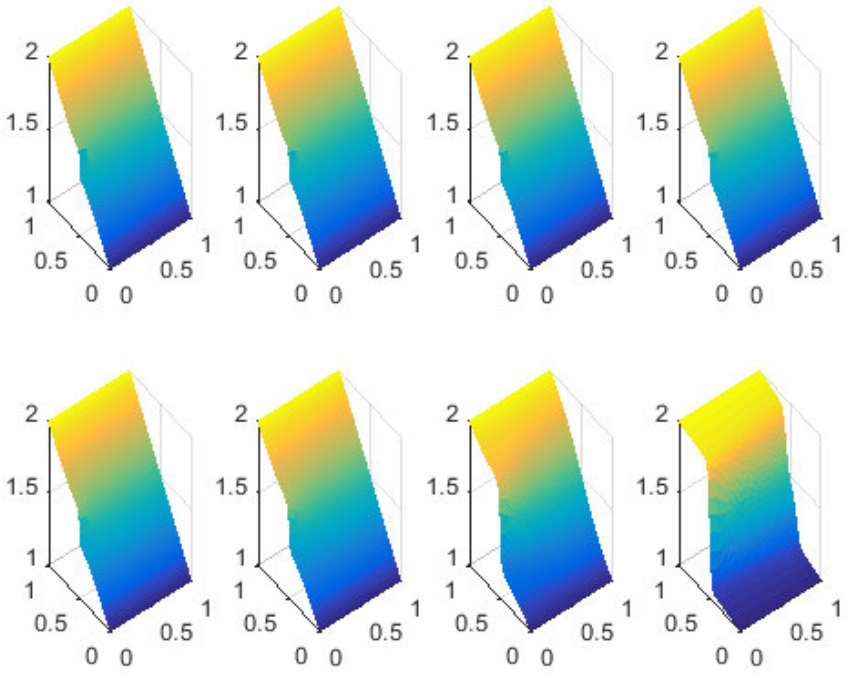


Abbildung 5.5.: Verschiebung des Gebietes bei konstantem u_0, einem kleinen Riss am Rand und 100×100 Gitterpunkten

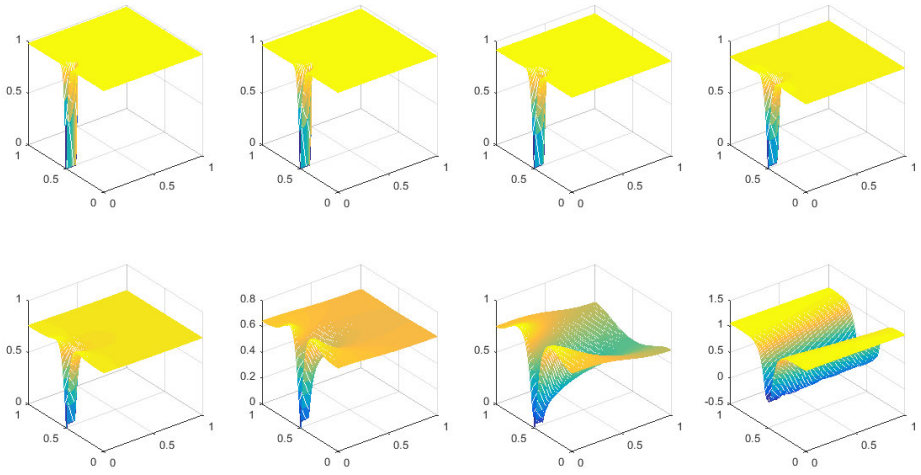


Abbildung 5.6.: Darstellung des Risses bei konstantem u_0, einem kleinen Riss am Rand und 100×100 Gitterpunkten

5.2.3. Das Material ist an beiden Seiten eingerissen

Befestigen wir nun das Material wieder wie vorhin und reißen es an beiden Seiten an, sodass die Risse aufeinander zulaufen. Dies stellen Abbildung 5.7 und 5.8 dar.

Zunächst wird der Riss bei beiden Bildern nur geglättet. Später setzt er sich fort, sodass beide Risse zu einem Riss werden. Dadurch erhält man einen Riss, der quer durch das gesamte Material verläuft.

Dieses ist auch das erwartete Verhalten des Risses.

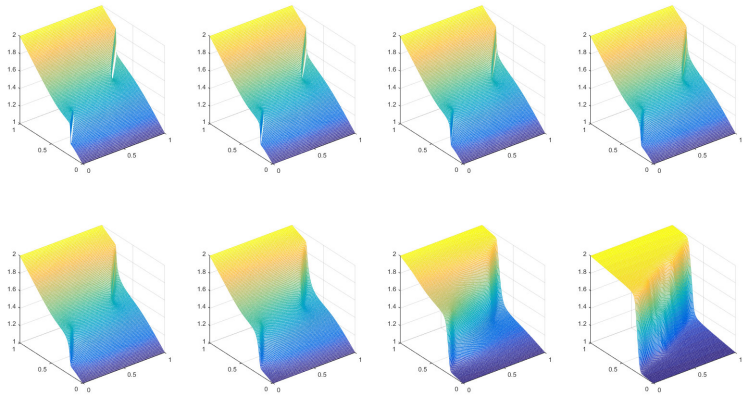


Abbildung 5.7.: Verschiebung des Gebietes bei konstantem u_0, einem kleinen Riss an beiden Rändern und 100×100 Gitterpunkten

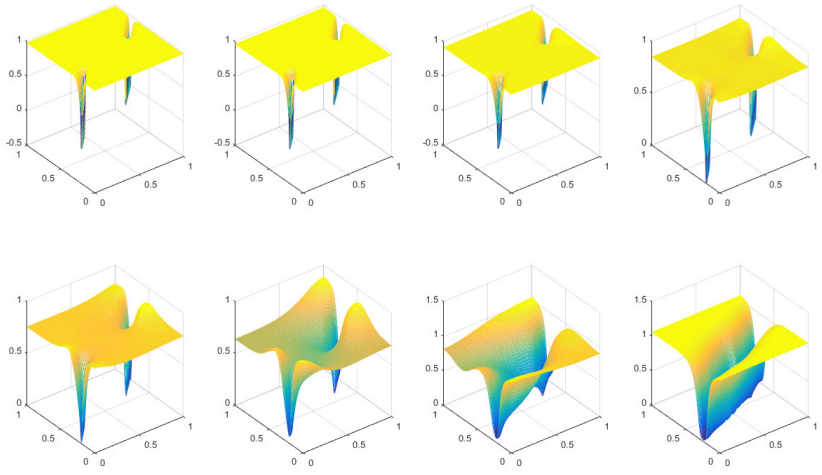


Abbildung 5.8.: Darstellung des Risses bei konstantem u_0, einem kleinen Riss an beiden Rändern und 100×100 Gitterpunkten

6. Fazit und Ausblick

Insgesamt lässt sich feststellen, dass die Implementation bei einer größeren Anzahl an Gitterpunkten bei allen vorgestellten Beispielen realitätsnahe Ergebnisse liefert. Bei weniger Gitterpunkten erhalten wir nicht das erwartete Ergebnis, wie z.B. bei Abbildung 5.3 und 5.4. Hier reißt das Material am Rand und nicht in der Mitte. Eine mögliche Erklärung ist, dass der Riss sehr groß im Vergleich zu dem Material ist und jegliche Fortsetzung des Risses schon dazu führt, dass das Material auch am Rand gerissen wird.

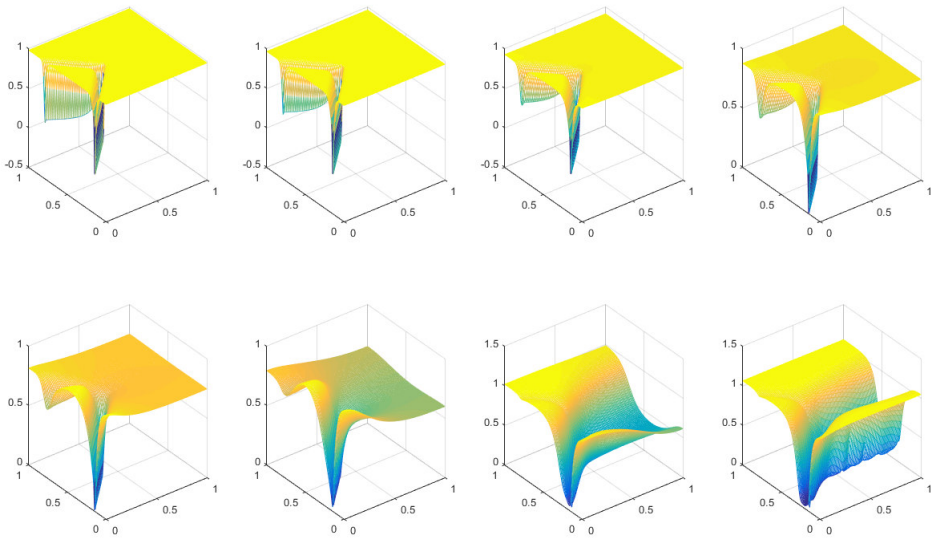


Abbildung 6.1.: Darstellung des Risses bei konstantem u_0, zwei kleinen Riss an einem Rand und $100x100$ Gitterpunkten

Ein anderes, nicht funktionierendes Beispiel, zeigt die Modellierung von zwei Rissen, die in einem Material auf der gleichen Seite sind und aufeinander zu laufen. Ein Riss verschwindet komplett bei vielen Iterationen, wie in Abbildung 6.1 dargestellt. Dies dürfte nicht sein. Wenn das Material einmal gerissen ist, sollte der Riss immer dort bleiben. Weiterführend könnte betrachtet werden, warum dieser Riss verschwindet. Vielleicht

ist dies ein Problem in der Modellierung, da in dieser Modellierung die Oberfläche des Risses klein gehalten wird, wenn die Verschiebung des Gebietes nicht zu groß wird. Die Verschiebung des Materials wird klein, wenn schon ein anderer Riss vorhanden ist. Dadurch könnte der Riss verschwinden.

Betrachten wir nun die Laufzeit der Implementation. Diese liegt bei $O(n^2)$, wobei n die Breite bzw. Höhe des Gitters ist. Dieses ist in 6.2 zu sehen. Eigentlich sollte es in $O(n)$ möglich sein, das Problem zu implementieren, da nur Matrizen aufgestellt werden und mehrere Matrixmultiplikationen mit Sparse-Matrizen stattfinden, die sehr effizient sind. Beim genaueren Betrachten des Codes fällt auf, dass meine Implementation sehr viel Zeit bei dem erstellen der Matrizen braucht. Dieses könnte vermutlich sehr viel effizienter berechnet werden.

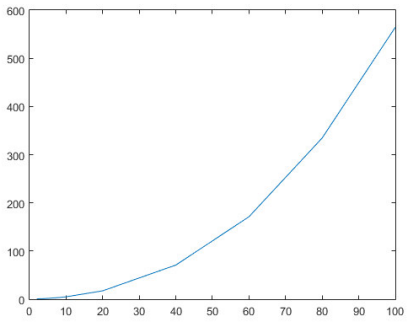


Abbildung 6.2.: Laufzeit der Implemention. Auf der x-Achse ist die Gitterweite, auf der y-Achse die Zeit in Sekunden angegeben

In der analytischen Betrachtung könnte man untersuchen, ob weniger Voraussetzungen an u_0 und v_0 gestellt werden müssen. Stetigkeit von u_0 sollte ausreichend sein, ist aber auch notwendig. In der Praxis kann man Material nicht unstetig einspannen.

Eine wichtige Untersuchung wäre die Konvergenz der Newton Methode. Diese zu beweisen erweist sich als sehr schwierig, da nicht klar ist, ob die Regularitätsannahme erfüllt ist. Im Diskreten konvergiert die Methode für alle untersuchten Beispiele, was natürlich keine Aussage darüber trifft, ob sie auch im analytischen Sinn konvergiert.

Nun haben wir eine Methode gefunden, mit der das Phasenfeld Modell für Rissentstehung berechnet werden kann.

A. Allgemeine Informationen

A.1. Rechnungen

In diesem Kapitel werden alle Rechnungen vorgestellt, die zur numerischen Darstellung der Optimierung nach v notwendig sind.

A.1.1. Numerische Darstellung von G_1

Die Formel

$$\begin{aligned} G_1(v^h, \eta^h) = &\left(2(\int_\Omega u^{dis} T_i T_j \,\mathrm{d}x)_{ij} + 2\epsilon_2 (\int_\Omega \nabla T_i \nabla T_j \,\mathrm{d}x)_{ij} + \frac{2}{\epsilon_3}(\int_\Omega T_i T_j \,\mathrm{d}x)_{ij} \right) v^k \\ &- \frac{2}{\epsilon_3}(\int_\Omega T_j \,\mathrm{d}x)_j + (\int_\Omega T_i T_j \,\mathrm{d}x)_{ij} \eta^k \\ &= (2A + 2\epsilon_2 B + \frac{2}{\epsilon_3} D) v^h - \frac{2}{\epsilon_3} c + D\eta^h \end{aligned}$$

soll berechnet werden.

Um A, B, D zu berechnen, brauchen wir $\int_E \varphi_i \varphi_j$ bzw. $\int_E \nabla\varphi_i \nabla\varphi_j$, wobei E das Einheitsdreieck ist.

	$\int_E \varphi_i \varphi_j$	$\int_E \nabla\varphi_i \nabla\varphi_j$
$\varphi_0\varphi_0$	$\frac{1}{12}$	1
$\varphi_1\varphi_1$	$\frac{1}{12}$	$\frac{1}{2}$
$\varphi_2\varphi_2$	$\frac{1}{12}$	$\frac{1}{2}$
$\varphi_0\varphi_1$	$\frac{1}{24}$	$-\frac{1}{2}$
$\varphi_0\varphi_2$	$\frac{1}{24}$	$-\frac{1}{2}$
$\varphi_1\varphi_2$	$\frac{1}{24}$	0

Nun können wir die einzelnen Matrizen berechnen. Die Berechnung erfolgt analog zur Optimierung nach u. Bei den Matrizen gibt es immer die Fälle, dass i und j gleich sind, j rechts neben i ist, j direkt unter i liegt und j rechts unter i liegt. Für alle anderen i und j ist der Matrixeintrag immer 0. Die Bezeichnungen sind die Gleichen, wie bei u.

Berechnung der Matrix A

$$\begin{aligned}
A_{i,i} &= \int_{\Omega} u^{dis} T_i T_i \,\mathrm{d}x \\
&= \int_{E_1} u^{dis}_{E_1} \varphi_1 \varphi_1 \,\mathrm{d}x + \int_{E_2} u^{dis}_{E_2} \varphi_2 \varphi_2 \,\mathrm{d}x + \int_{E_3} u^{dis}_{E_3} \varphi_0 \varphi_0 \,\mathrm{d}x \\
&\quad + \int_{E_4} u^{dis}_{E_4} \varphi_0 \varphi_0 \,\mathrm{d}x + \int_{E_5} u^{dis}_{E_5} \varphi_2 \varphi_2 \,\mathrm{d}x + \int_{E_6} u^{dis}_{E_6} \varphi_1 \varphi_1 \,\mathrm{d}x \\
&= \frac{1}{12} u^{dis}_{E_1} + \frac{1}{12} u^{dis}_{E_2} + \frac{1}{12} u^{dis}_{E_3} + \frac{1}{12} u^{dis}_{E_4} + \frac{1}{12} u^{dis}_{E_5} + \frac{1}{12} u^{dis}_{E_6} \\
&= \frac{1}{12} \left(\sum_{i=1}^{6} u^{dis}_{E_i} \right)
\end{aligned}$$

$$\begin{aligned}
A_{i,i+1} &= \int_{\Omega} u^{dis} T_i T_{i+1} \,\mathrm{d}x = \int_{E_3} u^{dis}_{E_3} \varphi_0 \varphi_1 \,\mathrm{d}x + \int_{E_6} u^{dis}_{E_6} \varphi_0 \varphi_1 \,\mathrm{d}x \\
&= \frac{1}{24} u^{dis}_{E_3} + \frac{1}{24} u^{dis}_{E_6} = \frac{1}{24} \left(u^{dis}_{E_3} + u^{dis}_{E_6} \right)
\end{aligned}$$

$$\begin{aligned}
A_{i,i+1+n} &= \int_{\Omega} u^{dis} T_i T_{i+1+n} \,\mathrm{d}x = \int_{E_4} u^{dis}_{E_4} \varphi_0 \varphi_1 \,\mathrm{d}x + \int_{E_5} u^{dis}_{E_5} \varphi_0 \varphi_1 \,\mathrm{d}x \\
&= \frac{1}{24} u^{dis}_{E_4} + \frac{1}{24} u^{dis}_{E_5} = \frac{1}{24} \left(u^{dis}_{E_4} + u^{dis}_{E_5} \right)
\end{aligned}$$

$$\begin{aligned}
A_{i,i+n+2} &= \int_{\Omega} u^{dis} T_i T_{i+2+n} \,\mathrm{d}x = \int_{E_5} u^{dis}_{E_5} \varphi_1 \varphi_2 \,\mathrm{d}x + \int_{E_6} u^{dis}_{E_6} \varphi_1 \varphi_2 \,\mathrm{d}x \\
&= \frac{1}{24} u^{dis}_{E_5} + \frac{1}{24} u^{dis}_{E_6} = \frac{1}{24} \left(u^{dis}_{E_5} + u^{dis}_{E_6} \right)
\end{aligned}$$

Berechnung der Matrix B

$$
\begin{aligned}
B_{i,i} &= \int_{\Omega} \nabla T_i \nabla T_i \,\mathrm{d}x \\
&= \int_{E_1} \nabla\varphi_1 \nabla\varphi_1 \,\mathrm{d}x + \int_{E_2} \nabla\varphi_2 \nabla\varphi_2 \,\mathrm{d}x + \int_{E_3} \nabla\varphi_0 \nabla\varphi_0 \,\mathrm{d}x \\
&+ \int_{E_4} \nabla\varphi_0 \nabla\varphi_0 \,\mathrm{d}x + \int_{E_5} \nabla\varphi_2 \nabla\varphi_2 \,\mathrm{d}x + \int_{E_6} \nabla\varphi_1 \nabla\varphi_1 \,\mathrm{d}x \\
&= \frac{1}{2} + \frac{1}{2} + 1 + 1 + \frac{1}{2} + \frac{1}{2} = 4
\end{aligned}
$$

$$
\begin{aligned}
B_{i,i+1} &= \int_{\Omega} \nabla T_i \nabla T_{i+1} \,\mathrm{d}x = \int_{E_3} \nabla\varphi_0 \nabla\varphi_1 \,\mathrm{d}x + \int_{E_6} \nabla\varphi_0 \nabla\varphi_1 \,\mathrm{d}x \\
&= -\frac{1}{2} - \frac{1}{2} = -1
\end{aligned}
$$

$$
\begin{aligned}
B_{i,i+n+1} &= \int_{\Omega} \nabla T_i \nabla T_{i+1+n} \,\mathrm{d}x = \int_{E_4} \nabla\varphi_0 \nabla\varphi_1 \,\mathrm{d}x + \int_{E_5} \nabla\varphi_0 \nabla\varphi_1 \,\mathrm{d}x \\
&= -\frac{1}{2} - \frac{1}{2} = -1
\end{aligned}
$$

$$
\begin{aligned}
B_{i,i+n+2} &= \int_{\Omega} \nabla T_i \nabla T_{i+2+n} \,\mathrm{d}x \\
&= \int_{E_5} \nabla\varphi_1 \nabla\varphi_2 \,\mathrm{d}x + \int_{E_6} \nabla\varphi_1 \nabla\varphi_2 \,\mathrm{d}x = 0
\end{aligned}
$$

Berechnung der Matrix D

$$\begin{aligned} D_{i,i} &= \int_\Omega T_i T_i \,\mathrm{d}x \\ &= \int_{E_1} \varphi_1\varphi_1 \,\mathrm{d}x + \int_{E_2} \varphi_2\varphi_2 \,\mathrm{d}x + \int_{E_3} \varphi_0\varphi_0 \,\mathrm{d}x \\ &\quad + \int_{E_4} \varphi_0\varphi_0 \,\mathrm{d}x + \int_{E_5} \varphi_2\varphi_2 \,\mathrm{d}x + \int_{E_6} \varphi_1\varphi_1 \,\mathrm{d}x \\ &= \frac{1}{12} + \frac{1}{12} + \frac{1}{12} + \frac{1}{12} + \frac{1}{12} + \frac{1}{12} = \frac{1}{2} \end{aligned}$$

$$\begin{aligned} D_{i,i+1} &= \int_\Omega T_i T_{i+1} \,\mathrm{d}x = \int_{E_3} \varphi_0\varphi_1 \,\mathrm{d}x + \int_{E_6} \varphi_0\varphi_1 \,\mathrm{d}x \\ &= \frac{1}{24} + \frac{1}{24} = \frac{1}{12} \end{aligned}$$

$$\begin{aligned} D_{i,i+n+1} &= \int_\Omega T_i T_{i+1+n} \,\mathrm{d}x = \int_{E_4} \varphi_0\varphi_1 \,\mathrm{d}x + \int_{E_5} \varphi_0\varphi_1 \,\mathrm{d}x \\ &= \frac{1}{24} + \frac{1}{24} = \frac{1}{12} \end{aligned}$$

$$\begin{aligned} D_{i,i+n+2} &= \int_\Omega T_i T_{i+2+n} \,\mathrm{d}x = \int_{E_5} \varphi_1\varphi_2 \,\mathrm{d}x + \int_{E_6} \varphi_1\varphi_2 \,\mathrm{d}x \\ &= \frac{1}{24} + \frac{1}{24} = \frac{1}{12} \end{aligned}$$

A.1.2. Berechnung von G_{2v}

Es muss $\int_\Omega \frac{\partial f}{\partial v} T_i T_j \,\mathrm{d}x$ berechnet werden. Dazu integriert man statt über Ω wieder über die einzelnen Dreiecke. Dabei ist zu beachten, dass für gerade und ungerade Dreiecke andere Ergebnisse zustande kommen:

i	j	$\int_E \frac{\partial f}{\partial v} T_i T_j \,\mathrm{d}x$ gerades Dreieck	$\int_E \frac{\partial f}{\partial v} T_i T_j \,\mathrm{d}x$ ungerades Dreieck
0	0	$\frac{1}{60}(f_1^h + 3f_2^h + f_3^h)$	$\frac{1}{60}(f_1^h + 3f_2^h + f_3^h)$
0	1	$\frac{1}{120}(f_1^h + 2f_2^h + 2f_3^h)$	$\frac{1}{120}(2f_1^h + 2f_2^h + f_3^h)$
0	2	$\frac{1}{120}(2f_1^h + 2f_2^h + f_3^h)$	$\frac{1}{120}(f_1^h + 2f_2^h + 2f_3^h)$
1	1	$\frac{1}{60}(f_1^h + f_2^h + 3f_3^h)$	$\frac{1}{60}(3f_1^h + f_2^h + f_3^h)$
1	2	$\frac{1}{120}(2f_1^h + f_2^h + 2f_3^h)$	$\frac{1}{120}(2f_1^h + f_2^h + 2f_3^h)$
2	2	$\frac{1}{60}(3f_1^h + f_2^h + f_3^h)$	$\frac{1}{60}(f_1^h + f_2^h + 3f_3^h)$

Dabei ist f_1^h bei einem geraden Dreieck die Auswertung von f^h an der oberen linken Ecke des Dreiecks. Die anderen Bezeichnungen sind darauf aufbauend.

Damit können wir ∂G_{2v} diskretisieren. Wir nennen die Diskretisierung F_{ij}. Hier hat man wieder die vier Fälle:

$$
\begin{aligned}
F_{i,i} &= \int_\Omega f^h T_i T_i \,\mathrm{d}x \\
&= \int_{E_1} f^h_{E_1} \varphi_1 \varphi_1 \,\mathrm{d}x + \int_{E_2} f^h_{E_2} \varphi_2 \varphi_2 \,\mathrm{d}x + \int_{E_3} f^h_{E_3} \varphi_0 \varphi_0 \,\mathrm{d}x \\
&\quad + \int_{E_4} f^h_{E_4} \varphi_0 \varphi_0 \,\mathrm{d}x + \int_{E_5} f^h_{E_5} \varphi_2 \varphi_2 \,\mathrm{d}x + \int_{E_6} f^h_{E_6} \varphi_1 \varphi_1 \,\mathrm{d}x \\
&= \frac{1}{60} \Big(\big(f_1^h + f_2^h + 3f_3^h\big)_{E_1} + \big(f_1^h + f_2^h + 3f_3^h\big)_{E_2} + \big(f_1^h + 3f_2^h + f_3^h\big)_{E_3} \\
&+ \big(f_1^h + 3f_2^h + f_3^h\big)_{E_4} + \big(3f_1^h + f_2^h + f_3^h\big)_{E_5} + \big(3f_1^h + f_2^h + f_3^h\big)_{E_6} \Big)
\end{aligned}
$$

$$
\begin{aligned}
A_{i,i+1} &= \int_\Omega f^h T_i T_{i+1} \,\mathrm{d}x = \int_{E_3} f^h_{E_3} \varphi_0 \varphi_1 \,\mathrm{d}x + \int_{E_6} f^h_{E_6} \varphi_0 \varphi_1 \,\mathrm{d}x \\
&= \frac{1}{120} \Big(\big(f_1^h + 2f_2^h + 2f_3^h\big)_{E_3} + \big(2f_1^h + 2f_2^h + f_3^h\big)_{E_6} \Big)
\end{aligned}
$$

$$
\begin{aligned}
A_{i,i+1+n} &= \int_\Omega f^h T_i T_{i+1+n} \,\mathrm{d}x = \int_{E_4} f^h_{E_4} \varphi_0 \varphi_1 \,\mathrm{d}x + \int_{E_5} f^h_{E_5} \varphi_0 \varphi_1 \,\mathrm{d}x \\
&= \frac{1}{120} \Big(\big(2f_1^h + 2f_2^h + f_3^h\big)_{E_4} + \big(f_1^h + 2f_2^h + 2f_3^h\big)_{E_5} \Big)
\end{aligned}
$$

$$A_{i,i+n+2} = \int_{\Omega} f^h T_i T_{i+2+n} \,\mathrm{d}x = \int_{E_5} f^h_{E_5} \varphi_1 \varphi_2 \,\mathrm{d}x + \int_{E_6} f^h_{E_6} \varphi_1 \varphi_2 \,\mathrm{d}x$$
$$= \frac{1}{120} \left(\left(2f_1^h + f_2^h + 2f_3^h\right)_{E_5} + \left(2f_1^h + f_2^h + 2f_3^h\right)_{E_6} \right)$$

hierbei bedeutet $\left(f_1^h + f_2^h + f_3^h\right)_{E_j}$, dass f_i^h f auf dem i-ten Gitterpunkt des Dreieck E_j ausgewertet wird. Die Transformation mit $1/h_1h_2$ wird auch hier am Schluss ausgeführt.

A.2. Code

```
%OPTIMAZATION berechnet die Optimierung von \int_\Omega (v^2 + eps_1)
%|\nabla u|^2 + eps_2 |\nabla v|^2 + 1/eps_3
% (1-v)^2 dx
%   Dabei wurde die Optimierung in zwei Teile geteilt: einmal die
%   Optimierung nach u und die Optimierung nach v.
%
%   Optimierung nach u:
%   Die Optimierung nach u kann direkt gelöst werden mittels
%   \int_\Omega (v^2 + eps_1) \nabla u_1
%   \nabla T_i dx \forall i \in N, wobei N die Anzahl der Stützstellen ist.
%
%   Optimierung nach v:
%   Hier wird eine Semiglatte Newton Methode verwendet, da keine direkte
%   Lösung möglich ist.
%
%   Insgesamt sieht das Verfahren wie folgt aus:
%   Für k=1,2,...
%   1. Berechne u durch lösen eines Gleichungssystems
%   2. Löse das gleichungssystem
%         -(G_1(v^k,eta^k)) = (G_1v  G_1eta) (s_1^k)
%          (G_2(v^k,eta^k))   (G_2v  G_2eta) (s_2^k)
%      nach s^k
%   3. v^k+1   = s_1^k + v^k
%      eta^k+1 = s_2^k + eta^k
%   4. wiederhole

%   Dabei ist G gegeben durch
%   G(v^k,eta^k)_1 = \int_\Omega T_i v |\nabla u|^2 + eps_2 \nabla v \nabla
%   T_i - 2/ eps_2 (1-v)T_i + eta T_i dx \forall i
%   G(v^k,eta^k)_2 = \int_\Omega (eta - max{0,eta + c(v-v0)}
%                         - min{0,eta+cv}) T_i dx \forall i
%   wobei v_0 eine geg. Schranke ist und c ein Konstante.
%   Die genauen Ableitungen stehen an der Stelle, an der wir sie später
%   brauchen.


% Deklaration der Variablen

% n+1 ist die Anzahl der Stützstellen nach rechts
% m+1 ist die Anzahl der Stützstellen nach unten
n=100;
```

```
m=100;
nu = .1;
% Deklaration aller epsilons
epsilon = [0.01;nu* 5/(n+1) ;(5/(n+1) )*(1/nu)];

% setzten von v für Anfangsdaten. Hier liegt ein Riss in der Mitte des
% Gebietes vor.
v = ones((m+1)*(n+1),1);
for i=0:20
    v(i*(n+1)+ n/2) = 0;
    v(i*(n+1)+ n/2+1) = 0;
end

% eta ist der Lagrangemultiplikator. Er muss auch vorab gesetzt werden.
eta = -.34*ones((n+1)*(m+1),1);

%v0 ist die Schranke. Im Problem gilt, dass 0<v<v0
v0 =.9*ones((n+1)*(m+1),1);

% die Konstante kommt durch den Lagrangemultiplikator rein.
const = 1;

%Anzahl der Newtonschritte die durchgeführt werden sollen
k=200;
%für plots
l=1;
% u0 sind die Randdaten von u. Hier wird u am rechten und am linken Rand
% eingespannt, sodass an einem Rand u 1 und und am anderen 2.
u0=zeros((n+1)*(m+1), 1);

for i=0:m
     u0(i*(n+1)+1) = 1;
     u0(i*(n+1)+n+1) = 2;
end

% Da alles mit dreieckig linearen Lagrangeelementen implementiert ist, muss
% das Gebiet zunächst Triangulisiert werden.
% edges ist eine n*m*2 x 3 Matrix und berechnet alle Ecken der Dreiecke
% allSurroundingTriangles ist eine (m+1)*(n+1) x 6 Matrix und beinhaltet in
% der i ten Zeile alle indizes der Dreiecke, die um den Punkt i liegen.
[edges, allSurroundingTriangles] = triangulasation(m,n);

for i=1:k
```

```
% u1 wird berechnet.
u1 = calculate_u(edges, allSurroundingTriangles, u0, epsilon(1), v,...
    m, n);
u = [u1,u1] ;

% als nächstes soll das Gleichungssystem gelöst werden, Dazu muss G
% berechnte werden. Alle Matrizen, die gebraucht werden, werden hier
% berechnet.
% numerisch dargestellt ist G1 nichts anderes als
% (A+2eps_2B+2/eps_3D)v-2/eps_3*D*ones() +D*eta
% mit A = \int\Omega |\nabla u|^2 T_i T_j dx
%     B = \int\Omega \nabla T_i \nabla T_j dx
%     D = \int\Omega T_i T_j dx
[A, B, D] = Matrixes_for_G(u, edges, allSurroundingTriangles, m, n );
D=1/(n*m)*D;
% Berechnung von G1
G1 = (2*A+2*epsilon(2)*B+ (2/epsilon(3))*D)*v - ...
    (2/epsilon(3))*D*ones((n+1)*(m+1),1) + D*eta;

% Berechnung von G1v. Die Ableitung ist recht einfach zu sehen
G1v = m*A + 2*epsilon(2)*B + (2/epsilon(3))*D;

% Berechnung von G1eta. Die Ableitung ist recht einfach zu sehen
G1eta = D;

% Berechnung von G2 und den Ableitungen
[G2, G2v, G2eta] = G2_and_deratives(v, eta, const, v0, D, edges,...
    allSurroundingTriangles, m, n);

% Die Ableitungen müssen noch zu einer großen Matrix, wie oben
% beschrieben zusammengefügt werden. Hier bieten sich sparse Matrizen
% an, da es viele Nullen als Einträge gibt.
gradG = sparse ((m+1)*(n+1)*2 ,(m+1)*(n+1)*2 ) ;

% G1v steht oben links
gradG(1:(m+1)*(n+1),1:(m+1)*(n+1)) = G1v;
% G1eta steht oben rechts
gradG(1:(m+1)*(n+1),(m+1)*(n+1)+1:(m+1)*(n+1)*2) = G1eta;
% G2v steht unten links
gradG((m+1)*(n+1)+1:(m+1)*(n+1)*2,1:(m+1)*(n+1)) = G2v;
% G2eta steht unten rechts
gradG((m+1)*(n+1)+1:(m+1)*(n+1)*2,(m+1)*(n+1)+1:(m+1)*(n+1)*2) = G2eta;

% G1 und G2 müssen als Vektor dargestllt werden.
```

```
    G = sparse ((m+1)*(n+1)*2,1) ;
    % G1 steht oben
    G(1:(m+1)*(n+1),1) = - G1;
    % G2 steht unten
    G((m+1)*(n+1)+1:(m+1)*(n+1)*2,1) = - G2;

    % Berechnet die Schrittweite s, indem das Gleichungssystem G = G's
    % gelöst wird.
    s = gradG\G ;
    % berechnet das neue v
    v = s(1:(m+1)*(n+1)) + v;
    % berechnet das neue eta
    eta = s((m+1)*(n+1)+ 1:(m+1)*(n+1)*2) + eta;

    if i==1 || i==2 || i==5 || i==10 || i==20 || i==50 || i==100 || i==200

        resultu = reshape(u(:,1),n+1,m+1);
        resultv = reshape(v,n+1,m+1);
        resulteta = reshape(eta,n+1,m+1);
        [X,Y] = meshgrid(0:1/n:1);
        figure(3)
            subplot(2,4,l);
            mesh(X,Y,resultu);
        figure(4)
            subplot(2,4,l);
            mesh(X,Y,resultv);
        l=l+1
    end
end


function result = calculate_u(edges, allSurroundingTriangles, u0, ...
    epsilon, v, m, n)
% CALCULATE_U berechnet u
%   Ziel ist es
%   - \int_Omega (v^2 + eps) \nabla u0 \nabla T_j
%   = \int_Omega (v^2+eps) \nabla u \nabla T_i \nabla T_j
%   für alle j =1,... (m+1)(n+1) zu berechnen
%   Dafür muss die linke und rechte Seite numerisch berechnet werden.

    % Da für die Berechnung des Integrals für alle Dreiecke das Integral
    % von v^2+eps gebraucht wird, berechnen wir es hier.
    integral_v = integral_of_v_total(v, edges, epsilon,n,m);
```

```
%   rechte Seite
%   die rechte Seite ist 0 für alle Werte am Rand und sonst die Werte des
%   Integrals.
    % rechts muss an den rändern 0 gesetzt werden. Dadurch werden die 0
    % Randwerte eingebracht. Da die Matrix sonst singulär wird, muss der
    % Diagonaleintrag 1 gesetzt werden.
    rechts = integrate_u(allSurroundingTriangles,integral_v, m, n);
         for i=0:m
            rechts(i*(n+1)+1,: ) = 0;
            rechts(i*(n+1)+1+n,:) = 0;
            rechts(i*(n+1)+1,i*(n+1)+1) = 1;
            rechts(i*(n+1)+1+n,i*(n+1)+1+n ) = 1;
         end

%   linke Seite: - \int_Omega (v^2 + eps) \nabla u0 \nabla T_j
%   = - \int_Omega (v^2 + eps) \nabla T_i \nabla T_j * u_0

    links = - rechts * u0;
    % links muss an den Rändern 0 gesetzt werden. Dadurch werden die
    % 0 Randwerte eingebracht.
    for i=0:m
       links(i*(n+1)+1) = 0;
       links(i*(n+1)+1+n) = 0;
    end

    % hier wird x aus  Ax=b berechnet.

    result = rechts\links;

    result = result + u0;

end


function result = integrate_u(allSurroundingTriangles, integral_v, m, n)
%integrate_u berechnet das Integral über u, also  \int_Omega (v^2+eps)
%\nabla T_i \nabla T_j dx


    % result ist eine Sparse Matrix, die nur auf der Diagonalen, der ersten
    % Nebendiagonalen und der n+1 ten Nebendiagonalen Einträge hat.

    % berechnet die Diagonale von u. Hier werden Einträge der Form
    % \int_Omega (v^2+eps) \nabla T_i \nabla T_i dx berechnet
```

```
    diag = diagonal_u(allSurroundingTriangles,integral_v, n, m);
    % berechnet die erste Nebendiagonale von u. Also Einträge der Form
    % \int_Omega (v^2+eps) \nabla T_i \nabla T_i+1 dx berechnet
    secDiag = second_diagonal_u(allSurroundingTriangles,integral_v, n, m);
    % berechnet die n+1 te Nebendiagonale von u, also Einträge der Form
    % \int_Omega (v^2+eps) \nabla T_i \nabla T_i+1+n dx berechnet
    farDiag = far_diagonal_u(allSurroundingTriangles,integral_v, n, m);
    % damit spDiags auf die richtigen Elemente zugreift, muss der erste bzw
    % die ersten n+1 Elemente 0 gesetzt werden.
    secDiag2 =  [0; secDiag(1:(n+1)*(m+1)-1)];
    farDiag2 = [zeros(n+1,1) ; farDiag(1:(n+1)*(m+1)-(n+1))] ;
    % schreibt alle Diagonalen in eine Sparse Matrix
    result = spdiags([farDiag secDiag diag secDiag2 farDiag2 ]...
        ,[-n-1 -1 0 1 n+1],(n+1)*(m+1),(n+1)*(m+1) );
end


function result = diagonal_u(allSurroundingTriangles, integral_v, n, m)
%DIAGONAL_U berechnet \int_Omega (v^2+eps) \nabla T_i \nabla T_i dx

    result = zeros((n+1)*(m+1),1) ;

    for i=1:(m+1)*(n+1)
        % berechnet die umliegenden Dreiecke vom Gitterpunkt i
        surrounders = allSurroundingTriangles(i,:);
        % berechnet das \int_Omega (v^2+eps) dx für die umliegenden
        % Dreiecke
        intvsurround = integral_of_v_surroundings(surrounders, integral_v);
        % rechnet die Integrale mit den richtigen Vorfaktoren zusammen. Die
        % Berechnung der Vorfaktoren kann in der Bachelorarbeit nachgelesen
        % werden.
        result(i) =  intvsurround(1) + intvsurround(2) ...
            + 2 * intvsurround(3) + 2 * intvsurround(4) ...
            + intvsurround(5) + intvsurround(6) ;
    end
end


function result = second_diagonal_u(allSurroundingTriangles,integral_v,...
    n, m)
%SECOND_DIAGONAL_U berechnet\int_Omega (v^2+eps) \nabla T_i \nabla T_i+1 dx

    result = zeros((n+1)*(m+1),1);
```

```
    for i=1:(n+1)*(m+1)
        % berechnet die umliegenden Dreiecke vom Gitterpunkt i
        surrounders = allSurroundingTriangles(i,:);
        % berechnet das \int_Omega (v^2+eps) dx für die umliegenden
        % Dreiecke
        intvsurround = integral_of_v_surroundings(surrounders, integral_v);
        % rechnet die Integrale mit den richtigen Vorfaktoren zusammen. Die
        % Berechnung der Vorfaktoren kann in der Bachelorarbeit nachgelesen
        % werden.
        result(i) = - intvsurround(3) - intvsurround(6);
    end
end


function result = far_diagonal_u(allSurroundingTriangles,integral_v, n, m)
%FAR_DIAGONAL_U berechnet \int_Omega (v^2+eps) \nabla T_i \nabla T_i+1 dx

    result = zeros((n+1)*(m+1),1);

    for i=1:(n+1)*(m+1)
        % berechnet die umliegenden Dreiecke vom Gitterpunkt i
        surrounders = allSurroundingTriangles(i,:);
        % berechnet das \int_Omega (v^2+eps) dx für die umliegenden
        % Dreiecke
        intvsurround = integral_of_v_surroundings(surrounders, integral_v);
        % rechnet die Integrale mit den richtigen Vorfaktoren zusammen. Die
        % Berechnung der Vorfaktoren kann in der Bachelorarbeit nachgelesen
        % werden.
        result(i) = - intvsurround(4) - intvsurround(5);
    end
end


function integrate = integral_of_v_total(v, edges, epsilon,n,m)
%INTEGRAL_OF_V_TOTAL berechnet \int_\Omega v^2 + eps_1 dx

    integrate = zeros(n*m*2,1);
    for i=1:n*m*2
        integrate(i) = integral_of_v(v(edges(i,:)),epsilon);
    end

end

function integral  = integral_of_v(v, epsilon)
```

```
%INTEGRAL_OF_V_SQUARE berechnet \int_E_i v^2 + eps_1 dx
%   dabei gilt:
%   eps ist der kleine Parameter Epsilon
%   vi ist der Wert von v an einem Eckpunkt
%   Die Formel für das Integral wurde in der zugehörigen Bachelorarbeit
%   berechnet

    integral= 1/12*(v(1)^2 + v(2)^2 + v(3)^2 + v(1)*v(2) + v(1)*v(3) ...
        + v(2)*v(3)) + 1/2*epsilon;

end


function result = integral_of_v_surroundings(surroundings,integral_v )
    % gibt alle v^2+eps für anliegende Dreiecke aus
    result = zeros(1,6);
    for i=1:6
        if (surroundings(i)~=0)
            result(i)=integral_v(surroundings(i));
        end
    end
end


function [A, B, D] =  Matrixes_for_G(u, edges, allSurroundingTriangles,...
    m, n )
%Berechnet alle Matrizen und Vektoren, die für die Berechnung von G
%gebraucht werden, außer e.

    % grad_u_total berechnet |\nabla u|^2 für alle Dreiecke der
    % Triangulierung.
    gradUTotal = grad_u_total(u, edges);

    % A = \int\Omega |\nabla u|^2 T_i T_j dx
    A = integrate_u_ti_tj(gradUTotal, allSurroundingTriangles, m, n);

    % B = \int\Omega \nabla T_i \nabla T_j dx
    B = integrate_gti_gtj(allSurroundingTriangles, m, n);

    % D = \int\Omega T_i T_j dx
    D = integrate_ti_tj(allSurroundingTriangles, m, n);
end

function gradU = grad_u_total(u, edges)
```

```
%GRAD_U_TOTAL berechnet |\nabla u|^2 für alle Dreiecke der Triangulierung.
% Dabei ist  |\nabla u|^2 = (u31-u21)^2 + (u31-u21)^2 + (u32-u22)^2 +
% (u32-u22)^2 Die Formel kann in der Bachelorarbeit nachgeschlagen werden.
    gradU = ( u(edges(:,3),1) - u(edges(:,2),1) ).^2 ...
        + ( u(edges(:,1),1) - u(edges(:,2),1) ).^2 ...
        + ( u(edges(:,3),2) - u(edges(:,2),2) ).^2 ...
        + ( u(edges(:,1),2) - u(edges(:,2),2) ).^2;
end

function result = integrate_u_ti_tj(gradUTotal, allSurroundingTriangles,...
    m, n)
%integrate_u_ti_tj berechnet int |nabla u|^2 T_i T_j.

    % komplette Matrix integrate_u_ti_tj
    result = zeros((m+1)*(n+1));

    %Diagonale der Matrix also die Einträge T_i T_i
    diag =zeros((m+1)*(n+1),1);

    % erste Nebendiagonale von u, also die Einträge zu T_i T_i+1
    secDiag = zeros((m+1)*(n+1),1);

    % n+1 te Nebendiagonale von u, die Einträge zu T_i T_i+n+1
    farDiag1 = zeros((m+1)*(n+1),1);

    %n+2 te Nebendiagonale von u, die Einträge zu T_i, T_i+n+2
    farDiag2 = zeros((m+1)*(n+1),1);

    for i=1:(m+1)*(n+1)
        % alle umliegenden Dreiecke zum Gitterpunkt i
        surroundings = allSurroundingTriangles(i,:);
        % gibt gradU von den umliegenden Dreiecken an.
        gradUSurroundings = zeros(6,1);
        for j=1:6
            if (surroundings(j)~=0)
                gradUSurroundings(j)=gradUTotal(surroundings(j));
            end
        end

        % berechnet die diagonalen. Formeln wieder in der Bachelorarbeit.
        diag(i)=(1/12) * sum(gradUSurroundings);
        secDiag(i) = (1/24) * (gradUSurroundings(3) ...
            + gradUSurroundings(6) ) ;
        farDiag1(i) = (1/24) * (gradUSurroundings(4) ...
```

```
            + gradUSurroundings(5) ) ;
        farDiag2(i) = (1/24) * (gradUSurroundings(5) ...
            + gradUSurroundings(6) ) ;
    end

    % wie vorhin ist die gewünschte Matrix symmetrisch und die Diagonalen
    % müssen auf beiden Seiten stehen.
    secDiag2 =  [0; secDiag(1:(n+1)*(m+1)-1)];
    farDiag12 = [zeros(n+1,1) ; farDiag1(1:(n+1)*(m+1)-(n+1))] ;
    farDiag22 = [zeros(n+2,1) ; farDiag2(1:(n+1)*(m+1)-(n+2))] ;
    % schreibt alle Diagonalen an der richtigen Stelle in eine Sparse
    % Matrix
    result = spdiags([farDiag2 farDiag1 secDiag diag secDiag2 farDiag12 ...
        farDiag22 ],[-n-2 -n-1 -1 0 1 n+1 n+2],(n+1)*(m+1),(n+1)*(m+1) );
end

function result = integrate_gti_gtj( allSurroundingTriangles, m, n)
%integrate_gti_gtj berechnet \int_\Omega nabla Ti nabla Tj dx
% Diese Matrix heißt später B

    % komplette Matrix integrate_u_ti_tj
    result = zeros((m+1)*(n+1));

    %Diagonale der Matrix
    diag =zeros((m+1)*(n+1),1);

    % erste Nebendiagonale von u
    secDiag = zeros((m+1)*(n+1),1);

    % n+1 te Nebendiagonale von u
    farDiag = zeros((m+1)*(n+1),1);

    for i=1:(m+1)*(n+1)
        % berechnet alle umliegenden Dreiecke des Gitterpunktes i
        sur = allSurroundingTriangles(i,:);
        % alle Einträge werden 1 gesetzt, die vorher einen Wert hatten, da
        % für die Berechnung der Diagonalen für jeden Wert nur 1 eingesetzt
        % wird. So werden die nicht vorhandenen Dreiecke ausgelassen und
        % die anderen Berechnet.
        for j=1:6
            if (sur(j)~=0)
                sur(j)=1;
            end
        end
```

```
        % Diagonalen werden berechnet. Formel wieder in der Bachelorarbeit
        diag(i)=(1/2) * sur(1) + (1/2) * sur(2) + sur(3) + sur(4) ...
            + (1/2) * sur(5) + (1/2) * sur(6);
        secDiag(i) = -(1/2) * sur(3) - (1/2) * sur(6);
        farDiag(i) =  -(1/2) * sur(4) - (1/2) * sur(5);
    end
    % Da Matrix symmetrisch ist, müssen die Diagonalen auf beiden Seiten
    % der Matrix stehen.
    secDiag2 =  [0; secDiag(1:(n+1)*(m+1)-1)];
    farDiag2 = [zeros(n+1,1) ; farDiag(1:(n+1)*(m+1)-(n+1))] ;
    % Ergebnisse werden in der Sparsematrix geschrieben.
    result = spdiags([farDiag secDiag diag secDiag2 farDiag2 ]...
        ,[-n-1 -1 0 1 n+1],(n+1)*(m+1),(n+1)*(m+1) );
end

function result = integrate_ti_tj(allSurroundingTriangles, m, n)
%integrate_ti_tj berechnet integral T_i T_j dx
% Diese Matrix heißt D in der Ausarbeitung

    % komplette Matrix integrate_u_ti_tj
    result = zeros((m+1)*(n+1));

    %Diagonale der Matrix
    diag =zeros((m+1)*(n+1),1);

    % erste Nebendiagonale von u
    secDiag = zeros((m+1)*(n+1),1);

    % n+1 te Nebendiagonale von u
    farDiag1 = zeros((m+1)*(n+1),1);

    % n+2 te Nebendiagonale von u
    farDiag2 = zeros((m+1)*(n+1),1);

    for i=1:(m+1)*(n+1)
        % berechnet die umliegenden Dreiecke des Gitterpunktes i
        surroundings = allSurroundingTriangles(i,:);
        % alle Einträge werden 1 gesetzt, die vorher einen Wert hatten, da
        % für die Berechnung der Diagonalen für jeden Wert nur 1 eingesetzt
        % wird. So werden die nicht vorhandenen Dreiecke ausgelassen und
        % die anderen Berechnet.
        for j=1:6
            if (surroundings(j)~=0)
                surroundings(j)=1;
```

```
            end
        end
        % berechnet die Diagonalen. Formel in der Bachelorarbeit.
        diag(i)=(1/12) * sum(surroundings);
        secDiag(i) = (1/24) * surroundings(3) + (1/24) * surroundings(6);
        farDiag1(i) =  (1/24) * surroundings(4) + (1/24) * surroundings(5);
        farDiag2(i) =  (1/24) * surroundings(5) + (1/24) * surroundings(6);
    end
    % Fast alle Diagonalen müssen zweimal eingetragen werden.
    secDiag2 =  [0; secDiag(1:(n+1)*(m+1)-1)];
    farDiag12 = [zeros(n+1,1) ; farDiag1(1:(n+1)*(m+1)-(n+1))] ;
    farDiag22 = [zeros(n+2,1) ; farDiag2(1:(n+1)*(m+1)-(n+2))] ;
    % Diagonalen werden in der Sparsematrix geschrieben.
    result = spdiags([farDiag2 farDiag1 secDiag diag secDiag2 farDiag12 ...
        farDiag22 ],[-n-2 -n-1 -1 0 1 n+1 n+2],(n+1)*(m+1),(n+1)*(m+1) );
end


function [ G2, G2v, G2eta ] = G2_and_deratives(v, eta, const, v0, D, ...
    edges, allSurroundingTriangles, m, n)
%G2_AND_DERATIVES berechnet G2 und die Ableitung nach v und eta.
%   G2(v, eta) = \int_\Omega f(v,eta) T_i dx \forall i
%   mit f(v,eta) = eta - max{0,eta + c(v-v0)} - min{0,eta+cv}
%   oder anders geschrieben
%   f(v,eta) = -c(v-v0) falls -c(v-v0) <= eta
%              eta      falls -cv < eta <  -c(v-v0)
%              -cv      falls  eta <=  -cv
%
%    f_v(v,eta) =  -c       falls -c(v-v0) < eta oder eta < -cv
%                   0       falls -cv < eta <  -c(v-v0)
%                 [-c,0]    falls -c(v-v0) = eta oder eta = -cv
%
%    f_eta(v,eta)= 0        falls -c(v-v0) < eta oder eta < -cv
%                  1        falls -cv < eta <  -c(v-v0)
%                 [0,1]     falls -c(v-v0) = eta oder eta = -cv
%   damit ist die gesamte Ableitung
%   G_2v(v, eta)   = \int_\Omega f_v   T_i T_j dx
%   G_2eta(v, eta) = \int_\Omega f_eta T_i T_j dx

    % Berechnet G2
    G2 = calculate_G2(v, eta, const, v0,D, m, n);
    % Berechnet die ableitungen von f
    [f_v, f_eta] = calculate_f_v_eta(v, eta, const, v0, m, n);
    % berechnet die gesamten Ableitungen
```

```
    G2v = calculate_G2div(f_v, edges, allSurroundingTriangles, m, n);
    G2eta = calculate_G2div(f_eta, edges, allSurroundingTriangles, m, n);
end

function G2 = calculate_G2(v, eta, const, v0, D, m, n)

    % initialisierung
    w = zeros((n+1)*(m+1),1);
    % alle Fallunterscheidungen in der Formel sind realisiert
    for i=1:(n+1)*(m+1)
        if ( -const*(v(i)-v0(i)) <= eta(i) )
            w(i) = -const*(v(i)-v0(i)) ;
        elseif ( - const * v(i) < eta(i) && eta(i) <  -const*(v(i)-v0(i)) )
            w(i) = eta(i);
        elseif ( eta(i) <= -const* v(i) )
            w(i) = - const * v(i) ;
        end
    end
    % alle Vorfaktoren aus dem min und max können aus dem integral gezogen
    % werden. Damit bleibt nur die Matrix D = \int_\Omega T_i T_j zu
    % berechnen, die vorher schon berechnet wurde.
    G2 = D * w;

end

function [f_v, f_eta] = calculate_f_v_eta(v, eta, const, v0, m, n)

    f_v = zeros((n+1)*(m+1),1);
    f_eta = zeros((n+1)*(m+1),1);
    % hier werden einfach wiede die Fallunterscheidungen aufgenommen, die
    % oben bereits erklärt wurden.
    for i=1:(n+1)*(m+1)
        if ( -const*(v(i)-v0(i)) < eta(i) || eta(i) < -const* v(i))
            f_v(i) = -const ;
            f_eta(i) = 0 ;
        elseif ( - const * v(i) < eta(i) && eta(i) <  -const*(v(i)-v0(i)) )
            f_v(i) = 0;
            f_eta(i) = 1;
        elseif ( -const*(v(i)-v0(i)) == eta(i) || eta(i) == -const* v(i) )
              % es f_v und f_eta aus dem intervall gewählt werden. Der
              % Einfachheit halber habe ich jeweils 0 gewählt.
              f_v(i) = 0;
              f_eta(i) = 0;
        end
```

```
    end
end

function G2div = calculate_G2div(f, edges, allSurroundingTriangles, m, n)
% CALCULATE_G2DIV berechnet die Ableitung von G_2.
%   Es ist egal, ob die ableitung nach v oder eta betrachet wird, da die
%   unterschiede nur in f liegen. f wurde bereits berechnet.

    % initialisierung
    diag = zeros((n+1)*(m+1),1);
    secdiag = zeros((n+1)*(m+1),1);
    thirddiag = zeros((n+1)*(m+1),1);
    forthdiag = zeros((n+1)*(m+1),1);
    for i=1:(n+1)*(m+1)
       % umliegende Dreieck des Gitterpunktes i berechnen
       surrounders = allSurroundingTriangles(i,:);
       % alle Ecken der umliegenden Dreiecke berechnen. Fallunterscheidung
       % ist notwendig, da 0 als Index nicht zugelassen ist.
       alledges = zeros(6,3);
       for j=1:6
           if surrounders(j) == 0
               alledges(j,:) = 0;
           else
                alledges(j,:) = edges(surrounders(j),:);
           end
       end
       % hier will ich wieder mit Sparsematrizen arbeiten. Also betrachte
       % ich nur die Diagonalen, auf denen Werte stehen. Die Herleitung
       % dieser Formeln findet sich in der zugehörigen Bachelorarbeit.
       % fn0 wird benötigt, da alledges auch 0 werden kann. Null ist als
       % Index nicht zugelassen.
       % diag ist die Hauptdiagonale. Sie beinhaltet die Werte für T_i T_i
       diag(i) =  1/60 * (fn0(f,alledges(1,1)) + fn0(f,alledges(1,2)) ...
           + 3 * fn0(f,alledges(1,3)) + fn0(f,alledges(2,1)) ...
           + fn0(f,alledges(2,2)) + 3 * fn0(f,alledges(2,3)) ...
           + fn0(f,alledges(3,1)) + 3 * fn0(f,alledges(3,2)) ...
           + fn0(f,alledges(3,3)) + fn0(f,alledges(4,1)) ...
           + 3 * fn0(f,alledges(4,2)) + fn0(f,alledges(4,3)) ...
           + 3 * fn0(f,alledges(5,1)) + fn0(f,alledges(5,2)) ...
           + fn0(f,alledges(5,3)) + 3 * fn0(f,alledges(6,1)) ...
           + fn0(f,alledges(6,2)) + fn0(f,alledges(6,3)));
       % secdiag ist die erste Nebendiagonale. Sie gibt an, wenn die
       % Gitterpunkte i und j direkt nebeneinander liegen, also T_i T_i+1
       secdiag(i) =   1/120 * (fn0(f,alledges(3,1)) ...
```

```
            + 2 * fn0(f,alledges(3,2)) + 2 * fn0(f,alledges(3,3)) ...
            + 2 * fn0(f,alledges(6,1)) + 2 * fn0(f,alledges(6,2)) ...
            + fn0(f,alledges(6,3)) ) ;
        % thirddiag ist die nächste Nebendiagonale. Sie liegt weiter weg, da
        % sie angibt, wenn j direkt unter dem Gitterpunkt i ist, also
        % j=i+n+1
        thirddiag(i) = 1/120 * (2*fn0(f,alledges(4,1)) ...
            + 2 * fn0(f,alledges(4,2)) + fn0(f,alledges(4,3)) ...
            + fn0(f,alledges(5,1)) + 2 * fn0(f,alledges(5,2)) ...
            + 2*fn0(f,alledges(5,3)) ) ;
        % forthdiag ist die letzte Nebendiagonale und gibt an, wenn der
        % Gitterpunkt j schräg unter dem Gitterpunkt i liegt, also T_i
        % T_i+n+2
        forthdiag(i) = 1/120 * (2 * fn0(f,alledges(5,1)) ...
            + fn0(f,alledges(5,2)) + 2 * fn0(f,alledges(5,3)) ...
            + 2 * fn0(f,alledges(6,1)) + fn0(f,alledges(6,2)) ...
            + 2 * fn0(f,alledges(6,3)) ) ;
    end
    % G2div ist symmetrisch. Also muss jede Diagonale auf jeder Seite der
    % Matrix sein. Damit das möglich wird, definiere ich mir ...diag2.
    % diese muss auf den ersten Stellen 0 gesetzt werden, damit spDiags
    % auch auf die richtigen Elemente zugreift.
    secdiag2 =  [0; secdiag(1:(n+1)*(m+1)-1)];
    thirddiag2 = [zeros(n+1,1) ; thirddiag(1:(n+1)*(m+1)-(n+1))] ;
    forthdiag2 = [zeros(n+2,1) ; forthdiag(1:(n+1)*(m+1)-(n+2))] ;
    % hier werden nun alle Diagonalen zu einer Sparsematrix
    % zusammengesetzt.
    G2div = spdiags([forthdiag thirddiag secdiag diag secdiag2 ...
        thirddiag2 forthdiag2],[-n-2 -n-1 -1 0 1 n+1 n+2]...
        ,(n+1)*(m+1),(n+1)*(m+1) );
end

function result = fn0 (f,int)
% FN0 gibt den Wert f zurück, falls int nicht 0 ist und gibt sonst 0
% zurück.
    if int==0
        result = 0;
    else
        result = f(int);
    end
end


function result = integrate_eta_ti_total(edges, sur, eta, m, n )
```

```
%INTEGRATE_ETA_TI_TOTAL  berechnet den Vektor \int_\Omega eta T_i dx für
% alle i
% sur sind alle umgebenden dreiecke also eine n+1*m+1 x 6 matrix

    % berechnet für jedes einzelne Dreieck das Integral
    intEtaTi = integrate_eta_ti_triangles(eta, edges, m, n );

    result = zeros((m+1)*(n+1),1);
    % summert alle Integrale der umliegenden Dreiecke des Gitterpunktes i.
    for i=1:(m+1)*(n+1)
        result(i) = surroundings_not_null(intEtaTi,sur,i,1,3) ...
            + surroundings_not_null(intEtaTi,sur,i,2,3)...
            + surroundings_not_null(intEtaTi,sur,i,3,2) ...
            + surroundings_not_null(intEtaTi,sur,i,4,2) ...
            + surroundings_not_null(intEtaTi,sur,i,5,1) ...
            + surroundings_not_null(intEtaTi,sur,i,6,1);
    end
end

function result = integrate_eta_ti_triangles(eta, edges, m, n )
%INTERATE_ETA_TI_TRIANGLES berechnet für jedes Dreieck \int_\E eta t_i dx
%f+r T_0,T_1 und T_2
% Daraus ergibt sich, dass wir eine 2*n*m x 3 Matrix haben

    result = zeros(2*n*m,3) ;

    result(:,1) = 1/24 * (2 * eta(edges(:,1)) + eta(edges(:,2)) ...
        + eta(edges(:,3))) ;
    result(:,2) = 1/24 * (eta(edges(:,1)) + 2 * eta(edges(:,2)) ...
        + eta(edges(:,3))) ;
    result(:,3) = 1/24 * (eta(edges(:,1)) + eta(edges(:,2)) ...
        + 2 * eta(edges(:,3)));
end

function result = surroundings_not_null(intEtaTi, sur, i, j, k)
%SURROUNDINGS_NOT_NULL ist eine funktion, die den wert von intEtaTi
%ausgibt, falls sur(i,j) nicht 0 ist.
        if (sur(i,j)==0)
            result = 0;
        else
            result=intEtaTi(sur(i,j),k);
        end

end
```

```
function [edges, E] = triangulasation(m,n)
% TRIANGULASATION gibt die Triangulierung von dem Gebiet Omega an.
%   edges ist eine n*m*2 x 3 Matrix und berechnet alle Ecken der Dreiecke
%   allSurroundingTriangles ist eine (m+1)*(n+1) x 6 Matrix und beinhaltet
%   in der i ten Zeile alle indizes der Dreiecke, die um den Punkt i
%   liegen.

    edges = edges_of_triangles(m,n);
    E = surrounding_triangles(m,n);
end

function edges = edges_of_triangles(m,n)
% EDGES_OF_TRIANGLES berechnet die Ecken jedes Dreiecks.

%   Nun möchte ich von meinem Gitter die Eckpunkt des jeweiligen Dreiecks
%   bestimmen.
%   - - - - - -
%   |\ 2|\ 4|\ 6|
%   |1\ |3\ |5\ |
%   |--\|--\|--\|
%   |\ 8|\10|\12|
%   |7\ |9\ |1\ |
%   |--\|--\|--\|
%
%   hier gilt n=3, m=2
%   Zunächst berechnet man für alle Dreiecke den linken obersten Randpunkt.
%   Nun unterscheidet man zwischen geraden und ungeraden Dreiecken.
%   Gerade Dreiecke:
%   Die nächste Ecke liegt einfach rechts neben der schon berechneten Ecke,
%   d.h. man rechnet den Wert +1
%   Die untere Ecke liegt direkt unter der vorherigen Ecke, also +n+1
%   Ungerade Dreiecke:
%   Die nächste Ecke liegt einfach unter der schon berechneten Ecke, d.h.
%   man rechnet den Wert +n+1
%   Die untere Ecke liegt direkt neben der vorherigen Ecke, also +1

edges=zeros(n*m*2,3);

for i=0:n*m*2-1
    edges(i+1,1)=floor(i/2)+floor(i/(2*n))+1;
    if (mod(i,2)~=0)
        edges(i+1,2)=edges(i+1,1)+1;
        edges(i+1,3)=edges(i+1,2)+n+1;
    else
```

```
        edges(i+1,2)=edges(i+1,1)+n+1;
        edges(i+1,3)=edges(i+1,2)+1;
    end
end

end


function E = surrounding_triangles(m,n)
%SURROUNDING_TRIANGLES gibt alle anliegenden Dreiecke von jedem Gitterpunkt
%an.

%   Insgesamt brauchen wir die 6 Dreiecke, die um den Gitterpunkt j liegen.
%   Sobald wir das erste Dreieck (oben links) gefunden haben, können wir
%   die anderen daraus berechnen.
%
%   Berechnung der Lage des ersten Dreieckes:
%   Diese Formel scheint aus dem Himmel zu fallen, funktioniert aber.

% initiierung von E
E = zeros((n+1)*(m+1),6);

for j=1:(n+1)*(m+1)
    if (j>n+1 && j<=(n+1)*m && mod(j,n+1)~=0 && mod(j,n+1)~=1)
        % Gitterpunkte in der Mitte
        E(j,1) = 2*j-(2*n+5)-2*floor((j-n-2)/(n+1));
        E(j,2)=E(j,1)+1;
        E(j,3)=E(j,1)+2;
        E(j,4)=E(j,1)+2*n+1;
        E(j,5)=E(j,4)+1;
        E(j,6)=E(j,4)+2;
    elseif (j==1)
        % Gitterpunkt in der oberen linken Ecke
        E(j,5)=1;
        E(j,6)=2;
    elseif (j<n+1 && j>1)
        % oberer Rand
        E(j,4)= 2*j-(2*n+5)-2*floor((j-n-2)/(n+1)) + 2*n + 1;
        E(j,5)=E(j,4)+1;
        E(j,6)=E(j,4)+2;
    elseif (j==n+1)
        % Gitterpunkt in der unteren linken Ecke
        E(j,4)= 2*n;
    elseif (mod(j,n+1)==0 && j~= (n+1)*(m+1))
```

```
        % Gitterpunkte am rechten Rand
        E(j,1) = 2*j-(2*n+5)-2*floor((j-n-2)/(n+1));
        E(j,2)=E(j,1)+1;
        E(j,4)=E(j,1)+2*n+1;
    elseif (j== (n+1)*(m+1))
        % Gitterpunkt an der rechten unteren Ecke
        E(j,1) = 2*j-(2*n+5)-2*floor((j-n-2)/(n+1));
        E(j,2)=E(j,1)+1;
    elseif( mod(j,n+1)==1 && j~= (n+1)*m+1)
        % Gitterpunkte am linken Rand
        E(j,3)= 2*j-(2*n+5)-2*floor((j-n-2)/(n+1))+2;
        E(j,5)=E(j,3)+2*n;
        E(j,6)=E(j,5)+1;
    elseif (j==(n+1)*m+1)
        % Gitterpunkt in der rechten oberen Ecke
        E(j,3) = 2*j-(2*n+5)-2*floor((j-n-2)/(n+1)) +2;
    elseif (j>(n+1)*m+1 && j< (n+1)*(m+1))
        % unterer Rand
        E(j,1) = 2*j-(2*n+5)-2*floor((j-n-2)/(n+1));
        E(j,2)=E(j,1)+1;
        E(j,3)=E(j,1)+2;
    end
end


end
```

Literaturverzeichnis

[1] Dietrich Braess. *Finite Elemente*. Springer-Verlag, 1992. 3

[2] Andrea Braides. Γ-*convergence for Beginners*. Oxford University Press, 2005. 4

[3] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimazation with PDE Constraints*. Springer, 2009. 6, 8, 13, 15, 16, 17

[4] Karl Kunisch. Semi-smooth newton methods for non-differentiable optimazation problems. In *Lipschitz Lectures*, Universität Graz, 2008. http://math.uni-graz.at/kunisch/papers/lipschitzlectureskunisch.pdf. 9

[5] Prof. Benedikt Wirth. Vorlesung optimierung ii. 6, 7, 8

[6] Prof. Benedikt Wirth. Vorlesung partielle differentialgleichungen. 9, 10

[7] Dr. Frank Wübbeling. Vorlesung numerik partieller differentialgleichungen. 10