TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Diplomarbeit

Numerische Integration der Landau-Lifschitz-Gilbert-Gleichung

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

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Math. Dr.techn. Dirk Praetorius,
Dr.rer.nat. Marcus Page, M.Sc. und
Dott.mag. Michele Ruggeri

durch
Josef Friedrich Kemetmüller
Marktplatz 13
4501 Neuhofen an der Krems

Abstract

Die Landau-Lifschitz-Gilbert-Gleichung beschreibt das dynamische Verhalten eines Ferromagneten unter Einfluss eines elektromagnetischen Feldes. Sie zeichnet sich dabei durch Ihre starke Nichtlinearität und nichtkonvexe Nebenbedingung aus und stellt damit eine Herausforderung für die Lösung mittels numerischer Methoden dar. In dieser Arbeit wird dafür ein Finite-Elemente-Schema entworfen. Dieses erweitert ein bekanntes Verfahren um die physikalischen Beiträge der Anisotropie-, Streufeld- und Zeeman-Energie auf das volle magnetische Feld. Der Term der Austauschenergie wird dabei implizit behandelt, während die anderen Terme explizit in das Schema eingehen. Damit kann auf etablierte Methoden zur Berechnung des Streufeldes zurückgegriffen werden. Für das vorgestellte Verfahren wird der Beweis der bedingten Konvergenz einer Teilfolge gegen eine schwache Lösung erbracht. Es werden zwei Iterationen vorgestellt, die zur Lösung verwendet werden können, anschließend deren MATLAB-Implementierung präsentiert und damit Standardbeispiele gelöst.

Inhaltsverzeichnis

1 Einleitung

Ferromagnetismus wird üblicherweise mittels sogenannten Elementarmagneten modelliert. Dabei handelt es sich um Magneten kleinster Größe, die jeweils ein eigenes Magnetfeld induzieren. Die Stärke der Magnetisierung eines Körpers ergibt sich dann als Summe der einzelnen induzierten Magnetfelder. Eine starke Magnetisierung ergibt sich demnach, wenn alle Elementarmagneten gleich ausgerichtet sind. Die von der Temperatur abhängige stärkstmögliche Magnetisierung wird dabei als Sättigungsmagnetisierung M_s bezeichnet. Die in Ampere pro Meter [A/m] gegebene Magnetisierung eines Körpers Ω konstanter Temperatur bis zu einem Zeitpunkt T lässt sich dann also als Funktion

$$\mathbf{M} : (0,T) \times \Omega \to \{\mathbf{x} \in \mathbb{R}^3 : |\mathbf{x}| = \mathrm{M}_s\}$$

darstellen. Um die Dynamik dieser Magnetisierung zu beschreiben wurde von Lew Dawidowitsch Landau und Jewgeni Michailowitsch Lifschitz [LL35] eine phänomenologische Gleichung beschrieben. Diese wurde von Thomas Lewis Gilbert [Gil55] modifiziert, um für Materialien großer Dämpfung realistische Ergebnisse zu erhalten. In ihrer einfachsten, dimensionslosen Form sieht die Gilbert-Formulierung dieses Problems dann wie folgt aus.

Für ein Gebiet Ω und einen Endzeitpunkt T definieren wir $\Omega_T := (0,T) \times \Omega$ und suchen eine Magnetisierung $\mathbf{m} : \Omega_T \to \mathbb{S}^2 := \{\mathbf{x} \in \mathbb{R}^3 : |\mathbf{x}| = 1\}$, welche die Gleichung

$$\mathbf{m}_t = \alpha \mathbf{m} \times \mathbf{m}_t - \mathbf{m} \times \mathbf{h}_{\text{eff}} \tag{1.1}$$

mit gegebenen Anfangs- und Randbedingungen $\mathbf{m}(0) = \mathbf{m}^0 \in \mathbf{H}^1(\Omega; \mathbb{S}^2)$ und $\partial_n \mathbf{m} = 0$ auf $(0,T) \times \partial\Omega$ erfüllt. Der Term $\mathbf{m}_t$ ist dabei die Zeitableitung von $\mathbf{m}$, $\alpha \geq 0$ der *Gilbert-Dämpfungsparameter* und das *effektive Feld* $\mathbf{h}_{\text{eff}}$ ein von $\mathbf{m}$ abhängiges Vektorfeld, das gewisse physikalische Effekte zusammenfasst.

In dieser Arbeit werden wir ein Verfahren zur Lösung des soeben genannten Problems vorstellen, für das wir schwache Subkonvergenz gegen eine schwache Lösung der Gleichung erhalten. Das vorgestellte Verfahren basiert dabei auf einer Arbeit von Sören Bartels und Andreas Prohl [BP06]. In die Analysis gehen dabei auch Argumente aus [Pag13] mit ein. Schwierigkeiten bei der numerischen Behandlung dieses Problems ergeben sich aus der nichtlinearen Natur der Gleichung, der Nebenbedingung $|\mathbf{m}| = 1$ und dem nichtlokalen Beitrag im effektiven Feld. Zur Berechnung des Randintegraloperators, der für die Berechnung des Streufeldbeitrags nötig ist, wird auf [May13] zurückgegriffen.

Der Rest dieser Arbeit ist wie folgt strukturiert

- In Kapitel 2 stellen wir kurz die physikalische Formulierung der LLG-Gleichung vor, gehen dann auf die dimensionslose Formulierung über und definieren schließlich den in der Arbeit verwendeten schwachen Lösungsbegriff.
- In Kapitel 3 werden die grundsätzlichen Begriffe für das verwendete Finite-Elemente-Schema eingeführt. Insbesondere wird der Raum der Ansatzfunktionen definiert und darauf ein modifiziertes Skalarprodukt eingeführt, das für die Längenerhaltung des Lösungsverfahrens entscheidend ist.
- Kapitel 4 widmet sich der Diskretisierung der LLG-Gleichung. Dabei werden zuerst die Voraussetzungen an die Feldbeiträge gestellt, die für die Konvergenzanalyse benötigt werden. Anschließend wird beschrieben wie die Feldbeiträge diskretisiert werden müssen um dem darauffolgend beschriebenen Lösungsverfahren zu genügen.
- Im Kapitel 5, dem Hauptteil dieser Arbeit, findet sich eine rigorose Analyse des Verfahrens aus Kapitel 4. Dabei wird einerseits die bedingte eindeutige Lösbarkeit präsentiert und andererseits die schwache Subkonvergenz dieser Lösungen gegen eine schwache Lösung des kontinuierlichen Problems gezeigt. Für die numerische Bearbeitung des Verfahrens wird außerdem eine Fixpunktiteration vorgestellt.
- Die entscheidenden Teile der Implementierung finden sich in Kapitel 6. Diese umfasst die Fixpunktiteration aus Kapitel 5, sowie eine Newton-Iteration, die eine höhere Konvergenzordnung aufweist.
- Zur Überprüfung der Implementierung werden in Kapitel 7 einige numerische Experimente durchgeführt. Dabei werden Standardprobleme aufgegriffen, die sich zur Verifikation von Lösern der LLG-Gleichung etabliert haben.
- Im Anhang finden sich sowohl wichtige Begebenheiten, die in die Analysis eingehen, als auch die übrigen MATLAB-Routinen, die im Zuge der Implementierung entwickelt wurden.

Danksagung

An dieser Stelle möchte ich noch ein Danke aussprechen an alle, die mich bei der Schaffung dieser Diplomarbeit unterstützt haben. Professor Praetorius danke ich für die gute Betreuung und die Einbringung entscheidender Ideen, die in die Analysis eingeflossen sind.

Mille grazie auch an meine Zweitbetreuer und magnetischen Bürokollegen Marcus Page und Michele Ruggeri für Rede, Antwort und Humor.

Das größte Danke aber widme ich meiner Familie, meinen Eltern, meinem Bruder. Danke für das Vertrauen und die Unterstützung die ich von Euch stets bedingungslos erhalten habe.

Finanziell wurde diese Arbeit durch das WWTF-Projekt *Micromagnetic Simulation and Computational Design of Future Devices (WWTF Project MA09-029)* und das FWF-Projekt *Adaptive Boundary Element Method (FWF Project P21732)* unterstützt.

2 Die LLG-Gleichung

2.1. Aus physikalischer Sicht

Da wir uns für eine physikalische Größe interessieren, werden wir hier die LLG-Gleichung in einer Form präsentieren, in der Sie auch in physikalischer Literatur gefunden werden kann. Siehe dazu beispielsweise [HS98, Abschnitt 3.2.7]. Für ein ferromagnetisches Material konstanter Temperatur, weit unter dessen Curie-Temperatur wird die Magnetisierung als Vektorfeld $\mathbf{M} : (0,T) \times \Omega \to \mathbb{R}^3$ konstanter Länge $|\mathbf{M}| = \mathrm{M}_s$ mit der Sättigungsmagnetisierung M_s dargestellt. Die LLG-Gleichung in der Gilbert-Form lautet dann

$$\mathbf{M}_t = \frac{\alpha}{\mathrm{M}_s}\mathbf{M} \times \mathbf{M}_t - \gamma\mu_0 \mathbf{M} \times \mathbf{H}_{\text{eff}} \quad \text{auf } \Omega_T, \tag{2.1}$$

mit dem gyromagnetischen Verhältnis für das Elektron $\gamma = 1,76 \cdot 10^{11}\,[(Ts)^{-1}]$ und der Vakuumpermeabilität $\mu_0 = 4\pi \cdot 10^{-7}\,[N/A^2]$. Die Konstante α ist hier ein dimensionsloser empirischer Parameter, der den Dämpfungswert des Materials beschreibt. Das *effektive Feld* $\mathbf{H}_{\text{eff}} : \Omega_T \to \mathbb{R}^3\,[A/m]$ hängt von der Magnetisierung ab und ist proportional zur negativen Variation der *Gibbs-Energie* $\mathcal{E}(\mathbf{M})$, genauer

$$\mu_0 \mathbf{H}_{\text{eff}}(\mathbf{M}) = -\frac{\partial \mathcal{E}(\mathbf{M})}{\partial \mathbf{M}}. \tag{2.2}$$

Dieses Energiefunktional ist gegeben durch

$$\mathcal{E}(\mathbf{M}) = \frac{A}{\mu_0 \mathrm{M}_s^2}\int_\Omega |\nabla \mathbf{M}|^2\,d\mathbf{x} + K\int_\Omega \Phi(\mathbf{M}/\mathrm{M}_s)\,d\mathbf{x} + \frac{\mu_0}{2}\int_{\mathbb{R}^3} |\nabla u|^2\,d\mathbf{x} - \mu_0 \int_\Omega \mathbf{H}_{\text{ext}} \cdot \mathbf{M}\,d\mathbf{x}. \tag{2.3}$$

Die einzelnen Beiträge werden Austauschenergie, Anisotropieenergie, Streufeldenergie und äußere oder Zeeman-Energie genannt. Entsprechend wird $A > 0\,[J/m]$ Austauschkonstante und $K > 0\,[J/m^3]$ Anisotropiekonstante genannt. Die glatte Funktion $\Phi : \mathbb{S}^2 \to \mathbb{R}$ beschreibt die Anisotropie des Materials. Ein externes Feld wird durch das Vektorfeld $\mathbf{H}_{\text{ext}}\,[A/m]$ dargestellt, und $u : \mathbb{R}^3 \to \mathbb{R}$ beschreibt das magnetostatische Potential als Lösung der magnetostatischen Maxwell-Gleichungen

$$\begin{aligned}
\Delta u &= \operatorname{div} \mathbf{M} && \text{auf } \Omega,\\
\Delta u &= 0 && \text{auf } \mathbb{R}^3 \setminus \overline{\Omega},\\
[u] &= 0 && \text{auf } \partial\Omega,\\
[\partial_{\mathbf{n}} u] &= -\mathbf{M} \cdot \mathbf{n} && \text{auf } \partial\Omega,\\
u(\mathbf{x}) &= \mathcal{O}\big(1/|\mathbf{x}|\big) && \text{für } |\mathbf{x}| \to \infty.
\end{aligned} \tag{2.4}$$

Als Kombination der Gleichungen (2.2) und (2.3) erhält man den folgenden Ausdruck für das effektive Feld

$$\mathbf{H}_{\text{eff}} = \frac{2A}{\mu_0 \mathrm{M}_s^2}\Delta\mathbf{M} - \frac{K}{\mu_0 \mathrm{M}_s} D\Phi(\mathbf{M}/\mathrm{M}_s) + \mathbf{H}_{\text{s}} + \mathbf{H}_{\text{ext}},$$

wobei $\mathbf{H}_{\text{s}} = -\nabla u\,[A/m]$ nun das Streufeld bezeichnet.

2.2. Aus mathematischer Sicht

Um für die mathematische Behandlung des Problems eine einfachere Basis zu schaffen, gehen wir nun zu einer dimensionslosen Formulierung des Problems über.

2.2.1. Starke Formulierung

Wir substituieren $t' = \gamma\mu_0 \mathrm{M}_s t$, um eine *reduzierte Zeit* zu erhalten, entsprechend setzen wir $T' = \gamma\mu_0 \mathrm{M}_s T$. Die Ortskomponenten ersetzen wir durch $\mathbf{x}' = \mathbf{x}/L$ mit einer Länge L in $[m]$. Hier kann beispielsweise die sogenannte intrinsische Länge $L = \sqrt{2A/(\mu_0 \mathrm{M}_s^2)}$ gewählt werden. Um die Notation zu vereinfachen, werden wir in weiterer Folge dennoch $t, T, \mathbf{x}$ und Ω anstatt $t', T', \mathbf{x}'$ bzw. Ω/L schreiben. Die dimensionslose Magnetisierung definieren wir durch $\mathbf{m} = \mathbf{M}/\mathrm{M}_s$ und erhalten $|\mathbf{m}| = 1$. Weiters setzen wir $\mathbf{h}_{\text{eff}} = \mathbf{H}_{\text{eff}}/\mathrm{M}_s$, $\mathbf{f} = \mathbf{H}_{\text{ext}}/\mathrm{M}_s$, $\mathbf{h}_{\text{s}} = \mathbf{H}_{\text{s}}/\mathrm{M}_s$. Mit diesen Substitutionen erhalten wir die dimensionslose Formulierung von (2.1) als

$$\mathbf{m}_t = \alpha\mathbf{m} \times \mathbf{m}_t - \mathbf{m} \times \mathbf{h}_{\text{eff}},$$

wobei das effektive Feld gegeben ist durch

$$\mathbf{h}_{\text{eff}} = C_{\text{ex}}\Delta\mathbf{m} - C_{\text{ani}} D\phi(\mathbf{m}) + \mathbf{h}_{\text{s}}(\mathbf{m}) + \mathbf{f},$$

mit den Konstanten $C_{\text{ex}} = 2A/(\mu_0 L^2 \mathrm{M}_s^2)$ und $C_{\text{ani}} = K/(\mu_0 \mathrm{M}_s^2)$.

Für die Konvergenzanalyse der nachfolgenden Kapitel werden wir diese Formulierung noch etwas vereinfachen. Wir betrachten dazu ein effektives Feld

$$\mathbf{h}_{\text{eff}} = C_{\text{ex}}\Delta\mathbf{m} + \Pi\mathbf{m} + \mathbf{f},$$

wobei $\Pi : \mathbf{L}^2(\Omega) \to \mathbf{L}^2(\Omega)$ ein allgemeiner zeitunabhängiger Feldbeitrag ist. Bei der Implementierung wird wieder die Wahl $\Pi\mathbf{m} = -C_{\text{ani}} D\phi(\mathbf{m}) + \mathbf{h}_{\text{s}}(\mathbf{m})$ auftauchen.

2.2.2. Schwache Formulierung

Da wir nun auf die schwache Formulierung des Problems übergehen, stellen wir noch einige Definitionen voran, die in den darauffolgenden Kapiteln die Notationen bestimmen werden. Es werden dabei lediglich die klassischen eindimensionalen Definitionen kanonisch auf unseren Fall der Abbildungen nach $\mathbb{R}^3$ erweitert.

Definition 2.2.1. *Sei $\Omega \subseteq \mathbb{R}^n$ ein Gebiet. Sei weiters $f \in L^1_{loc}(\Omega) := \{w : \Omega \to \mathbb{R}$ messbar $\,|\, w \in L^1(K)$ für alle $K \subset \Omega$ kompakt $\}$. Erfüllt eine Funktion $g \in L^1_{loc}(\Omega)$ die Gleichung*

$$\int_\Omega \phi g \, dx = -\int_\Omega \frac{\partial \phi}{\partial x_j} f \, dx \text{ für alle } \phi \in C_c^\infty(\Omega), \tag{2.5}$$

dann nennen wir g die schwache partielle Ableitung von f nach x_j. Wir schreiben dann $\frac{\partial f}{\partial x_j} := g$.

Definition 2.2.2. *Sei mit $\Omega \subseteq \mathbb{R}^k$ offen der Raum $\mathbf{H}^m(\Omega;\mathbb{R}^n)$ die Menge aller Funktionen $\mathbf{u} \in \mathbf{L}^2(\Omega;\mathbb{R}^n)$, deren Komponenten $\mathbf{u}_j$ schwache partielle Ableitungen $\partial^\alpha \mathbf{u}_j$ für alle $|\alpha| \leq m$ und $j = 1,\dots,n$ besitzen. Auf $\mathbf{H}^m(\Omega;\mathbb{R}^n)$ wird durch*

$$(\mathbf{u},\mathbf{v})_{\mathbf{H}^m} := \sum_{|\alpha|\leq m} (\partial^\alpha \mathbf{u}, \partial^\alpha \mathbf{v})_{\mathbf{L}^2}$$

ein Skalarprodukt erklärt. Die davon induzierte Norm bezeichnen wir mit $\|\cdot\|_{\mathbf{H}^m}$. Außerdem definieren wir die Seminorm

$$|\mathbf{u}|_{\mathbf{H}^m} := \sqrt{\sum_{|\alpha|=m} \|\partial^\alpha \mathbf{u}\|^2_{\mathbf{L}^2}}.$$

Definition 2.2.3. *Der Gradient einer vektorwertigen Funktion $\mathbf{u} \in \mathbf{H}^1(\Omega;\mathbb{R}^3)$ mit $\Omega \subseteq \mathbb{R}^3$ sei definiert als die Jacobi-Matrix von $\mathbf{u}$*

$$\nabla \mathbf{u}(\mathbf{x}) := \begin{pmatrix} \frac{\partial}{\partial x_1}\mathbf{u}(\mathbf{x}) & \frac{\partial}{\partial x_2}\mathbf{u}(\mathbf{x}) & \frac{\partial}{\partial x_3}\mathbf{u}(\mathbf{x}) \end{pmatrix} = \left(\frac{\partial \mathbf{u}_i}{\partial x_j}(\mathbf{x})\right)_{i,j\in\{1,2,3\}}$$

Definition 2.2.4. *Die Divergenz einer matrixwertigen Funktion $M : \Omega \subseteq \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ sei definiert als der Vektor der zeilenweisen Divergenzen*

$$\operatorname{div} M := \begin{pmatrix} \partial_1 M_{11} + \partial_2 M_{12} + \partial_3 M_{13} \\ \partial_1 M_{21} + \partial_2 M_{22} + \partial_3 M_{23} \\ \partial_1 M_{31} + \partial_2 M_{32} + \partial_3 M_{33} \end{pmatrix} = \begin{pmatrix} \operatorname{div}(M_{11}\ M_{12}\ M_{13}) \\ \operatorname{div}(M_{21}\ M_{22}\ M_{23}) \\ \operatorname{div}(M_{31}\ M_{32}\ M_{33}) \end{pmatrix}.$$

Definition 2.2.5. *Das $\mathbf{L}^2$-Skalarprodukt von matrix- oder tensorwertigen Funktionen $f, g : \Omega \to \mathbb{R}^{d_1\times\dots\times d_m}$ definieren wir durch die kanonische Identifikation des $\mathbb{R}^{d_1\times\dots\times d_m}$ mit $\mathbb{R}^{\prod_{j=1}^m d_j}$ als*

$$(f,g)_{\mathbf{L}^2(\Omega)} = \sum_{\substack{I=(i_1,\dots,i_m)\\ i_j\in\{1,\dots,d_j\}}} (f_I, g_I)_{\mathbf{L}^2(\Omega)}.$$

Bemerkung 2.2.6. *Beachte, dass im Sinne von Definition 2.2.5 die Abbildung*

$$\mathbf{u} \mapsto \|D^j \mathbf{u}\|_{\mathbf{L}^2(\Omega)} = \sum_{|\alpha|=m} \binom{m}{\boldsymbol{\alpha}} \|\partial^\alpha \mathbf{u}\|_{\mathbf{L}^2(\Omega)}$$

eine zu $|\mathbf{u}|_{\mathbf{H}^m}$ äquivalente Seminorm darstellt, wobei wir hier unter $\binom{k}{\boldsymbol{\alpha}}$ den Multinomialkoeffizienten $\binom{k}{\alpha_1,\dots,\alpha_n}$ verstehen. Gleiches gilt für die Abbildung

$$\mathbf{u} \mapsto \sum_{j=0}^m \|D^j \mathbf{u}\|_{\mathbf{L}^2(\Omega)} = \sum_{|\alpha|\leq m} \binom{m}{\boldsymbol{\alpha}} \|\partial^\alpha \mathbf{u}\|_{\mathbf{L}^2(\Omega)}$$

und die Norm $\|\mathbf{u}\|_{\mathbf{H}^m}$.

Wir schließen diesen Abschnitt mit unserem schwachen Lösungsbegriff. Dieser besteht aus der Nebenbedingung der Normierung, den Anfangswerten der Magnetisierung und der klassischen schwachen Formulierung, die man durch Multiplikation mit einer Testfunktion und anschließendem Integrieren erhält. Außerdem fordern wir eine Beschränktheit der Austauschenergie in einem gewissen Sinne. Diese entspricht nicht ganz der gewünschten physikalischen Energieabschätzung. Sie ist aber bereits in dieser abgeschwächten Form für die weitere Analysis hilfreich. Ob zusätzliche Annahmen an die Feldbeiträge Π und $\mathbf{f}$ eine bessere Abschätzung der Gesamtenergie liefern ist im Zuge dieser Arbeit nicht untersucht worden.

Definition 2.2.7. *Sei* $\mathbf{m}^0 \in \mathbf{H}^1(\Omega;\mathbb{S}^2)$*, dann nennen wir* $\mathbf{m} \in \mathbf{H}^1(\Omega_T;\mathbb{R}^3)$ *eine schwache Lösung der LLG-Gleichung, wenn gilt*

1. $|\mathbf{m}| = 1$ *fast überall auf* Ω_T *mit* $\mathbf{m}(0) = \mathbf{m}^0$ *im Spursinn;*
2. *Für alle Testfunktionen* $\boldsymbol{\phi} \in C^\infty(\overline{\Omega}_T;\mathbb{R}^3)$ *gilt*

$$\begin{aligned}\int_{\Omega_T} \langle \mathbf{m}_t, \boldsymbol{\phi}\rangle d\mathbf{x}dt = \alpha \int_{\Omega_T} \langle \mathbf{m}\times\mathbf{m}_t, \boldsymbol{\phi}\rangle d\mathbf{x}dt + C_{\text{ex}} \int_{\Omega_T} \langle \mathbf{m}\times\nabla\mathbf{m}, \nabla\boldsymbol{\phi}\rangle d\mathbf{x}dt \\ - \int_{\Omega_T} \langle \mathbf{m}\times\Pi\mathbf{m}, \boldsymbol{\phi}\rangle d\mathbf{x}dt \\ - \int_{\Omega_T} \langle \mathbf{m}\times\mathbf{f}, \boldsymbol{\phi}\rangle d\mathbf{x}dt\end{aligned}$$

3. *für fast alle* $t' \in (0,T)$ *gilt*

$$\frac{C_{\text{ex}}}{2}\int_\Omega |\nabla\mathbf{m}(t')|^2 d\mathbf{x} + \alpha\int_{(0,t')\times\Omega} |\mathbf{m}_t|^2 d\mathbf{x}dt \le C_{\text{energy}}$$

mit einem $C_{\text{energy}} > 0$*, das nur von* $\alpha, \mathbf{m}^0, \Omega, T, \Pi$ *und* $\mathbf{f}$ *aber nicht von* t' *abhängt.*

3 Grundlagen zur Diskretisierung

Wir stellen nun die typischen Begriffe für die Lösung mittels Finiter-Elemente-Methode vor.

3.1. Triangulierungen

Wir beginnen mit einigen Definitionen zur Diskretisierung des Simulationsgebietes.

Definition 3.1.1. *Wir nennen eine Menge $T \subset \mathbb{R}^3$ einen nicht-entarteten Tetraeder, wenn es Knoten $v_1, v_2, v_3, v_4 \in \mathbb{R}^3$ gibt, sodass gilt $T = \mathrm{Conv}\{v_1, v_2, v_3, v_4\}$ und das Volumen $|T| \neq 0$ erfüllt. Es bezeichnet*

$$\mathcal{N}_T := \{v_i : 1 \leq i \leq 4\}$$

die Menge der Knoten (nodes),

$$\mathcal{E}_T := \{\mathrm{Conv}\{v_i, v_j\} : 1 \leq i < j \leq 4\}$$

die Menge der Kanten (edges) und

$$\mathcal{F}_T := \{\mathrm{Conv}\{v_i, v_j, v_k\} : 1 \leq i < j < k \leq 4\}$$

die Menge der Flächen (faces) des gegebenen Tetraeders.

Definition 3.1.2. *Wir nennen eine Teilmenge $\mathcal{T} \subseteq \mathcal{P}(\Omega)$ der Potenzmenge von Ω eine Triangulierung des Gebietes Ω, wenn sie die folgenden Eigenschaften erfüllt:*

- *$\mathcal{T}$ ist eine endliche Menge von nicht-entarteten Tetraedern,*
- *$\overline{\Omega} = \bigcup_{T \in \mathcal{T}} T$,*
- *der Schnitt zweier Tetraeder hat Volumen $|T \cap T'| = 0$ für $T \neq T'$.*

Zu einer gegebenen Triangulierung $\mathcal{T}$ definieren wir die Menge $\mathcal{N} := \bigcup_{T \in \mathcal{T}} \mathcal{N}_T$ ihrer Knoten.

Dieser Triangulierungsbegriff ist für unsere Zwecke allerdings noch nicht stark genug. Wir wollen erreichen, dass sich Flächen von benachbarten Tetraedern nicht überlappen und damit sogenannte hängende Knoten vermeiden. Deswegen definieren wir, wie folgt:

Definition 3.1.3. *Eine Triangulierung von Ω heißt regulär, wenn für je zwei Tetraeder $T, T' \in \mathcal{T}$ mit $T \neq T'$ gilt, dass der Schnitt $T \cap T'$*

- *entweder leer,*
- *oder ein gemeinsamer Knoten $v \in \mathcal{N}_{T'} \cap \mathcal{N}_T$,*
- *oder eine gemeinsame Kante $E \in \mathcal{E}_{T'} \cap \mathcal{E}_T$,*
- *oder eine gemeinsame Fläche $F \in \mathcal{F}_{T'} \cap \mathcal{F}_T$ ist.*

Definition 3.1.4. *Wir notieren den Durchmesser eines Tetraeders T mit $h_T := \operatorname{diam}(T)$ und mit ρ_T den Radius der größten in T eingeschriebenen Sphäre. Eine Triangulierung werden wir mit $\mathcal{T}_h$ bezeichnen, falls für alle $T \in \mathcal{T}_h$ gilt, dass $h_T \leq h$.*

Definition 3.1.5. *Sei $(\mathcal{T}_h)_h$ eine Familie von Triangulierungen von Ω. Wir nennen die Familie* quasi-uniform*, falls es ein $C_{\text{form}} > 0$ gibt, sodass*

$$C_{\text{form}} \geq \frac{\max_{T \in \mathcal{T}_h} h_T}{\min_{T \in \mathcal{T}_h} \rho_T} \quad \text{für alle } h > 0.$$

3.2. Ansatzraum

Den Funktionenraum, in dem wir unsere FEM-Approximation suchen definieren wir wie folgt.

Definition 3.2.1. *Zu einer gegebenen Triangulierung $\mathcal{T}_h$ von Ω definieren wir die Menge der stückweise affinen Funktionen $S^1(\mathcal{T}_h) : \Omega \to \mathbb{R}^3$ als*

$$S^1(\mathcal{T}_h) := \{\boldsymbol{\phi}_h \in C(\overline{\Omega}; \mathbb{R}^3) : \boldsymbol{\phi}_h\big|_T \in \mathcal{P}^1(T) \text{ für alle } T \in \mathcal{T}_h\}.$$

Definition 3.2.2. *Für $\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)$ definieren wir die nodale Basisfunktion $\varphi_{\mathbf{x}_h} \in S^1(\mathcal{T}_h; \mathbb{R})$ durch $\varphi_{\mathbf{x}_h}(\mathbf{x}_h) = 1$ sowie $\varphi_{\mathbf{x}_h}(\mathbf{z}_h) = 0$ für alle $\mathbf{z}_h \in \mathcal{N}(\mathcal{T}_h) \setminus \{\mathbf{x}_h\}$.*

Definition 3.2.3. *Für eine Triangulierung $\mathcal{T}_h$ definieren wir den nodalen Interpolationsoperator $\mathcal{I}_h : C(\overline{\Omega}; \mathbb{R}^3) \to S^1(\mathcal{T}_h)$ durch die Eigenschaft*

$$\mathcal{I}_h \boldsymbol{\phi}(\mathbf{x}_h) = \boldsymbol{\phi}(\mathbf{x}_h) \quad \text{für alle } \mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h),$$

also

$$\mathcal{I}_h \boldsymbol{\phi} = \sum_{\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)} \boldsymbol{\phi}(\mathbf{x}_h) \varphi_{\mathbf{x}_h}.$$

An einigen Stellen werden wir die folgende sogenannte *reduzierte Integration* im Skalarprodukt benutzen. Dies wird in der Literatur häufig als *mass-lumping* bezeichnet.

Definition 3.2.4. *Für eine Triangulierung $\mathcal{T}_h$ definieren wir die Funktion $(\mathbf{u}, \mathbf{v})_h$ durch*

$$(\mathbf{u}, \mathbf{v})_h := \int_\Omega \mathcal{I}_h(\langle \mathbf{u}, \mathbf{v} \rangle) d\lambda.$$

Lemma 3.2.5. *Die Funktion $(\cdot, \cdot)_h$ aus Definition 3.2.4 ist ein Skalarprodukt auf $S^1(\mathcal{T}_h)$. Die davon induzierte Norm bezeichnen wir mit $\|\cdot\|_h$.*

Beweis. Die Bilinearität folgt sofort aus der Linearität des Integrals und der nodalen Interpolation sowie der Bilinearität des euklidischen Skalarprodukts auf $\mathbb{R}^n$. Entsprechende Argumente gelten auch für die Symmetrie und Definitheit. Wir zeigen die Definitheit. Sei also $(\boldsymbol{\phi}_h, \boldsymbol{\phi}_h)_h = 0$ und damit

$$0 = (\boldsymbol{\phi}_h, \boldsymbol{\phi}_h)_h = \int_\Omega \mathcal{I}_h(\langle \boldsymbol{\phi}_h, \boldsymbol{\phi}_h \rangle) d\lambda = \sum_{T \in \mathcal{T}_h} \int_T \mathcal{I}_h(|\boldsymbol{\phi}_h|^2) d\lambda.$$

Auf Grund der Nichtnegativität der Integranden und der elementweisen Stetigkeit erhalten wir die Bedingung

$$\mathcal{I}_h(|\boldsymbol{\phi}_h|^2)(\mathbf{x}) = 0 \quad \forall T \in \mathcal{T}_h \; \forall \mathbf{x} \in T.$$

Woraus insbesondere $\mathcal{I}_h(|\boldsymbol{\phi}_h|^2)(\mathbf{x}_h) = 0$ für alle $\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)$ folgt. Wir erhalten, dass für alle Knoten $\mathbf{x}_h$ die Gleichheit $\boldsymbol{\phi}_h(\mathbf{x}_h) = 0$ gelten muss, was uns für die Funktion $\boldsymbol{\phi}_h \in S^1(\mathcal{T}_h)$ die Identität $\boldsymbol{\phi}_h \equiv 0$ liefert. Für jede reguläre Triangulierung $\mathcal{T}_h$ erhalten wir also ein Skalarprodukt auf Ω. □

Bemerkung 3.2.6. *Das Skalarprodukt $(\mathbf{u}_h, \mathbf{v}_h)_h$ aus Definition 3.2.4 lässt sich auf $S^1(\mathcal{T}_h)$ darstellen durch*

$$(\mathbf{u}_h, \mathbf{v}_h)_h = \sum_{\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)} \beta_{\mathbf{x}_h} \langle \mathbf{u}_h(\mathbf{x}_h), \mathbf{v}_h(\mathbf{x}_h) \rangle \text{ mit } \beta_{\mathbf{x}_h} := \int_\Omega \varphi_{\mathbf{x}_h} d\mathbf{x}.$$

Weiters gilt für alle $\mathbf{u}, \mathbf{v} \in C(\Omega)$ die Gleichheit

$$(\mathbf{u}, \mathbf{v})_h = (\mathcal{I}_h(\mathbf{u}), \mathcal{I}_h(\mathbf{v}))_h.$$

Lemma 3.2.7. *Die Norm $\|\cdot\|_h$ auf $S^1(\mathcal{T}_h)$ aus Definition 3.2.4 ist äquivalent zu $\|\cdot\|_{\mathbf{L}^2(\Omega)}$ und deren Normäquivalenzkonstanten hängen nicht von der Triangulierung ab. Es existieren also feste $C_{\mathbf{L}^2 \leq h}, C_{h \leq \mathbf{L}^2} > 0$, sodass für alle Triangulierungen $\mathcal{T}_h$ von Ω und $\mathbf{u}_h \in S^1(\mathcal{T}_h)$ gilt*

$$C_{\mathbf{L}^2 \leq h}^{-1} \|\mathbf{u}_h\|_{\mathbf{L}^2(\Omega)} \leq \|\mathbf{u}_h\|_h \leq C_{h \leq \mathbf{L}^2} \|\mathbf{u}_h\|_{\mathbf{L}^2(\Omega)}.$$

Die Konstanten können dabei als $C_{\mathbf{L}^2 \leq h} = 1$ und $C_{h \leq \mathbf{L}^2} = \sqrt{5}$ gewählt werden.

Beweis. Wir benützen ein Skalierungsargument und verwenden die Normäquivalenz auf einem Referenzelement. Zuerst spalten wir das Integral in die Beiträge der einzelnen Elemente auf.

$$\|\mathbf{u}_h\|_{\mathbf{L}^2(\Omega)}^2 = \sum_{T \in \mathcal{T}_h} \int_T |\mathbf{u}_h|^2 d\lambda. \tag{3.1}$$

Mit geeigneten affinen Transformationen Φ_T auf ein Referenztetraeder T_{ref} gilt aufgrund des Transformationssatzes

$$\begin{aligned} \|\mathbf{u}_h\|_{\mathbf{L}^2(\Omega)}^2 &= \sum_{T \in \mathcal{T}_h} \int_{T_{\text{ref}}} |\mathbf{u}_h|^2 \circ \Phi_T \cdot \underbrace{|\det d\Phi_T|}_{=:\alpha_T} d\lambda \\ &= \sum_{T \in \mathcal{T}_h} \alpha_T \|\mathbf{u}_h \circ \Phi_T\|_{\mathbf{L}^2(T_{\text{ref}})}^2. \end{aligned} \tag{3.2}$$

Dieselben Transformationen Φ_T verwenden wir nun um obige Umformungsschritte auf $\|\mathbf{u}_h\|_h^2$ anzuwenden.

$$\begin{aligned}\|\mathbf{u}_h\|_h^2 &= \sum_{T\in\mathcal{T}_h}\int_T \mathcal{I}_h(|\mathbf{u}_h|^2)d\lambda \\ &= \sum_{T\in\mathcal{T}_h}\int_{T_{\text{ref}}} \mathcal{I}_h(|\mathbf{u}_h|^2)\circ\Phi_T\cdot\underbrace{|\det d\Phi_T|}_{=:\alpha_T}d\lambda \\ &= \sum_{T\in\mathcal{T}_h}\alpha_T\|\mathbf{u}_h\circ\Phi_T\|_{h,T_{\text{ref}}}^2\end{aligned} \tag{3.3}$$

Für den endlichdimensionalen Vektorraum $\mathcal{P}^1(T_{\text{ref}})$ erhalten wir die Normäquivalenz

$$C_{\mathbf{L}^2\le h}^{-2}\|\mathbf{u}_h\circ\Phi_T\|_{\mathbf{L}^2(T_{\text{ref}})}^2 \le \|\mathbf{u}_h\circ\Phi_T\|_{h,T_{\text{ref}}}^2 \le C_{h\le\mathbf{L}^2}^2\|\mathbf{u}_h\circ\Phi_T\|_{\mathbf{L}^2(T_{\text{ref}})}^2.$$

Wir erhalten also durch Einsetzen in Gleichungen (3.2)–(3.3) sowohl

$$\|\mathbf{u}_h\|_h^2 \le C_{h\le\mathbf{L}^2}^2\sum_{T\in\mathcal{T}_h}\alpha_T\|\mathbf{u}_h\circ\Phi_T\|_{\mathbf{L}^2(T_{\text{ref}})}^2 = C_{h\le\mathbf{L}^2}^2\|\mathbf{u}_h\|_{\mathbf{L}^2(\Omega)}^2,$$

als auch die andere Richtung

$$\|\mathbf{u}_h\|_h^2 \ge C_{\mathbf{L}^2\le h}^{-2}\sum_{T\in\mathcal{T}_h}\alpha_T\|\mathbf{u}_h\circ\Phi_T\|_{\mathbf{L}^2(T_{\text{ref}})}^2 = C_{\mathbf{L}^2\le h}^{-2}\|\mathbf{u}_h\|_{\mathbf{L}^2(\Omega)}^2.$$

Dies liefert uns die gewünschte Normäquivalenz. Für den Beweis der genauen Werte der Abschätzung siehe [Gol12, Lemma 2.2.3]. □

4 Diskretisierung der LLG-Gleichung

In diesem Abschnitt stellen wir unser diskretes Lösungsverfahren vor. Im Wesentlichen wird dabei in der Zeit ein Finite-Differenzen Schema durch Mittelpunktsauswertung und im Ort ein Finite-Elemente Ansatz gewählt. Wie bereits erwähnt, beruht dieses maßgeblich auf dem in [BP06] vorgestellten Verfahren. In die Berechnungen gehen aber zusätzlich zum Austauschterm noch die anderen Feldbeiträge ein. Ein positiver Aspekt der vorgestellten Methode liegt in der natürlich Erhaltung der Nebenbedingung der Längenerhaltung auf diskreter Ebene, wofür nicht zuletzt das zuvor vorgestellte reduzierte Skalarprodukt verantwortlich ist.

Für die nachfolgenden Kapitel werden wir immer einige Dinge voraussetzen.

4.1. Voraussetzungen

Das Gebiet $\Omega \subseteq \mathbb{R}^3$ sei triangulierbar und $T > 0$ sei der Endzeitpunkt der Simulation. Desweiteren sei $(\mathcal{T}_h)_h$ eine quasi-uniforme Familie von Triangulierungen von Ω. Wir werden also insbesondere für jeden Zeitpunkt $t \in [0,T]$ die gleiche Familie $(\mathcal{T}_h)_h$ verwenden. Für die Zeitschrittweite k setzen wir die Schreibweisen $0 = t_0 < t_1 < \ldots < t_N = T$ mit $k = k_j := t_j - t_{j-1}$ für alle $j = 1, \ldots, N$ fest.

Als Anfangswert sei für jedes $\mathcal{T}_h$ die Funktion $\mathbf{m}_h^0$ gegeben. Wir werden im weiteren Verlauf an einigen Stellen folgende Begebenheit verwenden und setzen diese deshalb als gegeben voraus.

Die Folge der $\mathbf{m}_h^0$ konvergiere schwach in $\mathbf{H}^1(\Omega)$ gegen eine Funktion $\mathbf{m}^0$, in Zeichen

$$\mathbf{m}_h^0 \rightharpoonup \mathbf{m}^0 \text{ in } \mathbf{H}^1(\Omega). \tag{4.1}$$

Das liefert uns nach dem Satz von Banach-Steinhaus insbesondere Beschränktheit der Folge $\|\mathbf{m}_h^0\|_{\mathbf{H}^1(\Omega)}$ für $h \to 0$. Mit dem Gedanken an ein nicht exakt berechenbares Streufeld verwenden wir in der weiteren Analyse den Feldbeitrag $\Pi_h : \mathbf{L}^2(\Omega) \to S^1(\mathcal{T}_h)$, von dem wir fordern, dass für beliebiges $\mathbf{u}$ in $\mathbf{L}^2(\Omega)$ die Funktion $\Pi_h \mathbf{u}$ schwach in $\mathbf{L}^2(\Omega)$ gegen $\Pi\mathbf{u}$ konvergiert, in Zeichen

$$\Pi_h \mathbf{u} \rightharpoonup \Pi\mathbf{u} \text{ in } \mathbf{L}^2(\Omega).$$

Für das externe Feld fordern wir

$$\mathbf{f} \in C([0,T]; \mathbf{L}^2(\Omega; \mathbb{R}^3)).$$

Für die soeben genannten Feldbeiträge erhalten wir also Konstanten C_Π und $C_\mathbf{f}$, sodass gilt

$$\begin{aligned} \|\Pi_h \mathbf{m}_h^j\|_{\mathbf{L}^2} &\leq C_\Pi \text{ und} \\ \|\mathbf{f}(t)\|_{\mathbf{L}^2} &\leq C_\mathbf{f}. \end{aligned} \tag{4.2}$$

4.2. Diskretisierung der Feldbeiträge

Definition 4.2.1. *Wir definieren als diskretes Pendant zum Laplace-Operator den Operator* $\tilde{\Delta}_h : \mathbf{H}^1(\Omega;\mathbb{R}^3) \to S^1(\mathcal{T}_h)$*, sowie zu* Π_h *den Operator* $\widetilde{\Pi}_h : S^1(\mathcal{T}_h) \to S^1(\mathcal{T}_h)$ *und das diskrete externe Feld* $\tilde{\mathbf{f}}_h \in C([0,T]; S^1(\mathcal{T}_h))$ *durch die Eigenschaften*

$$\begin{aligned}(\tilde{\Delta}_h \mathbf{u}, \mathbf{v}_h)_h &= -(\nabla \mathbf{u}, \nabla \mathbf{v}_h),\\ (\widetilde{\Pi}_h \mathbf{u}_h, \mathbf{v}_h)_h &= (\Pi_h \mathbf{u}_h, \mathbf{v}_h),\\ (\tilde{\mathbf{f}}_h(t), \mathbf{v}_h)_h &= (\mathbf{f}(t), \mathbf{v}_h)\end{aligned}$$

für alle $\mathbf{v}_h \in S^1(\mathcal{T}_h)$ *und* $t \in [0,T]$*. Man beachte dabei, dass das Lemma von Lax-Milgram A.4.7 die Existenz und Eindeutigkeit dieser Operatoren sichert.*

Lemma 4.2.2. *Für alle* $\boldsymbol{\phi}_h \in S^1(\mathcal{T}_h)$ *gilt*

$$\|\tilde{\Delta}_h \boldsymbol{\phi}_h\|_h \leq C_{A.5.4} h^{-1} \|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^2} \tag{4.3}$$

und

$$\begin{aligned}\|\tilde{\Delta}_h \boldsymbol{\phi}_h\|_h &\leq C_{A.5.4}^2 h^{-2} \|\boldsymbol{\phi}_h\|_{\mathbf{L}^2},\\ \|\tilde{\Delta}_h \boldsymbol{\phi}_h\|_h &\leq C_{A.5.4}^2 h^{-2} \|\boldsymbol{\phi}_h\|_h.\end{aligned} \tag{4.4}$$

Beweis. Wir verwenden die Definition des diskreten Laplace-Operators aus Definition 4.2.1 und wenden die Cauchy-Schwarzsche Ungleichung darauf an.

$$\begin{aligned}\|\tilde{\Delta}_h \boldsymbol{\phi}_h\|_h^2 &= -(\nabla \boldsymbol{\phi}_h, \nabla \tilde{\Delta}_h \boldsymbol{\phi}_h)\\ &\leq \|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^2} \|\nabla \tilde{\Delta}_h \boldsymbol{\phi}_h\|_{\mathbf{L}^2}\\ &\leq C_{A.5.4} h^{-1} \|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^2} \|\tilde{\Delta}_h \boldsymbol{\phi}_h\|_{\mathbf{L}^2}\\ &\leq C_{A.5.4} h^{-1} \|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^2} C_{\mathbf{L}^2 \leq h} \|\tilde{\Delta}_h \boldsymbol{\phi}_h\|_h.\end{aligned}$$

Im letzten Schritt haben wir einmal das Korollar der inversen Abschätzung A.5.4 angewandt. Dies zeigt mittels $C_{\mathbf{L}^2 \leq h} = 1$ die Ungleichung (4.3). Durch erneute Anwendung erhalten wir mit

$$\|\tilde{\Delta}_h \boldsymbol{\phi}_h\|_h \leq C_{A.5.4} h^{-1} \|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^2} \leq C_{A.5.4}^2 h^{-2} \|\boldsymbol{\phi}_h\|_{\mathbf{L}^2}$$

und der Normäquivalenzkonstante $C_{\mathbf{L}^2 \leq h} = 1$ auch (4.4). □

Lemma 4.2.3. *Sei* $(\mathcal{T}_h)_h$ *eine quasi-uniforme Familie von Triangulierungen von* Ω*. Dann gibt es ein* $C_{4.2.3} > 0$*, sodass für alle* $h > 0$ *und* $\boldsymbol{\phi}_h \in S^1(\mathcal{T}_h)$ *und* $\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)$ *gilt*

$$|\tilde{\Delta}_h \boldsymbol{\phi}_h(\mathbf{x}_h)| \leq C_{4.2.3} h^{-2} \|\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty}.$$

Beweis. Wir beginnen mit der Gleichheit $(\tilde{\Delta}_h \boldsymbol{\phi}_h, \varphi_{\mathbf{x}_h} \tilde{\Delta}_h \boldsymbol{\phi}_h(\mathbf{x}_h))_h = \beta_{\mathbf{x}_h} |\tilde{\Delta}_h \boldsymbol{\phi}_h(\mathbf{x}_h)|^2$ aus Bemerkung 3.2.6. Nun formen wir um und stellen $\boldsymbol{\phi}_h$ bezüglich der Basisfunktionen $\varphi_{\mathbf{x}_h}$ dar.

$$\begin{aligned}\beta_{\mathbf{x}_h} |\tilde{\Delta}_h \boldsymbol{\phi}_h(\mathbf{x}_h)|^2 &= (\tilde{\Delta}_h \boldsymbol{\phi}_h, \varphi_{\mathbf{x}_h} \tilde{\Delta}_h \boldsymbol{\phi}_h(\mathbf{x}_h))_h\\ &= (\nabla \boldsymbol{\phi}_h, \nabla \varphi_{\mathbf{x}_h} \tilde{\Delta}_h \boldsymbol{\phi}_h(\mathbf{x}_h))\\ &= (\nabla \sum_{\mathbf{z}_h \in \mathcal{N}(\mathcal{T}_h)} \varphi_{\mathbf{z}_h} \boldsymbol{\phi}_h(\mathbf{z}_h), \nabla \varphi_{\mathbf{x}_h} \tilde{\Delta}_h \boldsymbol{\phi}_h(\mathbf{x}_h))\\ &= \sum_{\mathbf{z}_h \in \mathcal{N}(\mathcal{T}_h)} \langle \boldsymbol{\phi}_h(\mathbf{z}_h), \tilde{\Delta}_h \boldsymbol{\phi}_h(\mathbf{x}_h) \rangle (\nabla \varphi_{\mathbf{z}_h}, \nabla \varphi_{\mathbf{x}_h}).\end{aligned}$$

Wir wenden die Cauchy-Schwarzsche Ungleichung an und erhalten unter Beachtung des Trägers von $\varphi_{\mathbf{x}_h}$ und $\varphi_{\mathbf{z}_h}$ die Ungleichung

$$\begin{aligned}\beta_{\mathbf{x}_h}|\tilde{\Delta}_h\boldsymbol{\phi}_h(\mathbf{x}_h)|^2 &\leq \sum_{\mathbf{z}_h\in\mathcal{N}(\mathcal{T}_h)} |\boldsymbol{\phi}_h(\mathbf{z}_h)||\tilde{\Delta}_h\boldsymbol{\phi}_h(\mathbf{x}_h)|(\nabla\varphi_{\mathbf{z}_h},\nabla\varphi_{\mathbf{x}_h})\\ &\leq \|\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty}|\tilde{\Delta}_h\boldsymbol{\phi}_h(\mathbf{x}_h)| \sum_{[\mathbf{x}_h,\mathbf{z}_h]\in\mathcal{E}(\mathcal{T}_h)} \|\nabla\varphi_{\mathbf{z}_h}\|_{\mathbf{L}^2}\|\nabla\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^2}.\end{aligned}$$

Nun benötigen wir die Formregularität der Triangulierungen. Diese liefert uns nämlich eine Schranke für die Anzahl der Nachbarknoten von $\mathbf{x}_h$. Außerdem garantiert sie, dass die Volumina zweier benachbarter Tetraeder sich nur um höchstens einen festen Faktor unterscheiden können. Damit ist also $\|\nabla\varphi_{\mathbf{z}_h}\|_{\mathbf{L}^2}$ von derselben Ordnung wie $\|\nabla\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^2}$ und wir können weiter abschätzen.

$$\begin{aligned}\beta_{\mathbf{x}_h}|\tilde{\Delta}_h\boldsymbol{\phi}_h(\mathbf{x}_h)| &\lesssim \|\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty}\|\nabla\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^2}^2\\ &\lesssim \|\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty}h^{-2}\|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^2}^2.\end{aligned}$$

Im zweiten Schritt haben wir die inverse Abschätzung angewandt. Insgesamt folgt

$$|\tilde{\Delta}_h\boldsymbol{\phi}_h(\mathbf{x}_h)| \lesssim h^{-2}\|\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty}\|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^2}^2\beta_{\mathbf{x}_h}^{-1}.$$

Da wir mit der Hölderschen Ungleichung

$$\|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^2}^2 = \|\varphi_{\mathbf{x}_h}\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^1} \leq \|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^1}\|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^\infty} = \|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^1} = \beta_{\mathbf{x}_h}$$

erhalten, können wir die Ungleichung noch vereinfachen zu

$$|\tilde{\Delta}_h\boldsymbol{\phi}_h(\mathbf{x}_h)| \lesssim h^{-2}\|\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty}$$

Womit wir die gewünschte Beziehung erhalten. □

Lemma 4.2.4. *Folgende Abschätzungen vererben sich auch auf die diskreten Versionen der Feldbeiträge*

$$\begin{aligned}\|\widetilde{\Pi}_h\mathbf{m}_h^j\|_h &\leq \|\Pi_h\mathbf{m}_h^j\|_{\mathbf{L}^2} \leq C_\Pi\\ \|\widetilde{\Pi}_h\mathbf{m}_h^j\|_{\mathbf{L}^\infty} &\leq C_{4.2.4}h^{-3/2}\\ \|\tilde{\mathbf{f}}_h(t)\|_h &\leq \|\mathbf{f}(t)\|_{\mathbf{L}^2} \leq C_{\mathbf{f}}\\ \|\tilde{\mathbf{f}}_h(t)\|_{\mathbf{L}^\infty} &\leq C_{4.2.4}h^{-3/2}\end{aligned}$$

Die Konstante $C_{4.2.4}$ hängt dabei nur von der Formregularitätskonstante $C_{\text{form}}, C_{\mathbf{f}}$ und C_Π ab.

Beweis. Die folgende Ungleichungskette zeigt die erste Behauptung

$$\begin{aligned}\|\widetilde{\Pi}_h\mathbf{m}_h^j\|_h^2 = (\widetilde{\Pi}_h\mathbf{m}_h^j,\widetilde{\Pi}_h\mathbf{m}_h^j)_h &= (\widetilde{\Pi}_h\mathbf{m}_h^j,\Pi_h\mathbf{m}_h^j)\\ &\leq \|\widetilde{\Pi}_h\mathbf{m}_h^j\|_{\mathbf{L}^2}\|\Pi_h\mathbf{m}_h^j\|_{\mathbf{L}^2}\\ &= \|\widetilde{\Pi}_h\mathbf{m}_h^j\|_h\|\Pi_h\mathbf{m}_h^j\|_{\mathbf{L}^2}\end{aligned}$$

Für die zweite Behauptung testen wir die Funktion $\widetilde{\Pi}_h\mathbf{m}_h^j$ mit $\boldsymbol{\phi}_h := \varphi_{\mathbf{x}_h}\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h)$ und erhalten

$$\begin{aligned}\beta_{\mathbf{x}_h}|\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h)|^2 = (\widetilde{\Pi}_h\mathbf{m}_h^j,\varphi_{\mathbf{x}_h}\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h))_h &= (\Pi_h\mathbf{m}_h^j,\varphi_{\mathbf{x}_h}\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h))_{\mathbf{L}^2}\\ &\leq \|\Pi_h\mathbf{m}_h^j\|_{\mathbf{L}^2}\|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^2}|\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h)|\end{aligned}$$

Da wir mit der Hölderschen Ungleichung die Eigenschaft

$$\|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^2}^2 = \|\varphi_{\mathbf{x}_h}\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^1} \leq \|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^1}\|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^\infty} = \|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^1} = \beta_{\mathbf{x}_h}$$

erhalten, bekommen wir die Ungleichung

$$\beta_{\mathbf{x}_h}|\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h)|^2 \leq \|\Pi_h\mathbf{m}_h^j\|_{\mathbf{L}^2}\beta_{\mathbf{x}_h}^{1/2}|\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h)|.$$

Ingesamt folgt

$$|\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h)| \leq \|\Pi_h\mathbf{m}_h^j\|_{\mathbf{L}^2}\beta_{\mathbf{x}_h}^{-1/2} \leq C_\Pi\beta_{\mathbf{x}_h}^{-1/2}. \tag{4.5}$$

Der Transformationssatz liefert uns

$$\begin{aligned}\beta_{\mathbf{x}_h} = \|\varphi_{\mathbf{x}_h}\|_{\mathbf{L}^1} &= \sum_{T\in\mathcal{T}(\mathbf{x}_h)}\int_{T_{\mathrm{ref}}} |\varphi_{\mathbf{x}_h}|\circ\Phi_T\cdot|\det D\Phi_T|d\lambda.\\ &\lesssim h^3\int_{T_{\mathrm{ref}}} 1\,d\lambda = O(h^3),\end{aligned}$$

wobei wir verwendet haben, dass für quasi-uniforme Triangulierungen die Anzahl der Tetraeder im Knotenpatch beschränkt ist und die Volumina der Tetraeder mit der Ordnung h^3 fallen. Beachte, dass durch die Forderung der Formregularität auch $|\det D\Phi_T|$ nach unten durch h^3 abgeschätzt werden kann. Wir erhalten insbesondere sogar $\beta_{\mathbf{x}_h} = \Theta(h^3)$. Die Abschätzung $\beta_{\mathbf{x}_h} = \Theta(h^3)$ setzen wir nun in (4.5) ein und erhalten die zweite Behauptung. Der Beweis der beiden anderen Behauptungen läuft gänzlich analog. Die folgende Ungleichungskette zeigt dabei die dritte Behauptung

$$\begin{aligned}\|\tilde{\mathbf{f}}_h(t)\|_h^2 = (\tilde{\mathbf{f}}_h(t),\tilde{\mathbf{f}}_h(t))_h &= (\tilde{\mathbf{f}}_h(t),\mathbf{f}(t))\\ &\leq \|\tilde{\mathbf{f}}_h(t)\|_{\mathbf{L}^2}\|\mathbf{f}(t)\|_{\mathbf{L}^2}\\ &= \|\tilde{\mathbf{f}}_h(t)\|_h\|\mathbf{f}(t)\|_{\mathbf{L}^2}\end{aligned}$$

und auf gleichem Weg wie oben erhalten wir

$$|\tilde{\mathbf{f}}_h(\mathbf{x}_h)| \leq \|\mathbf{f}(t)\|_{\mathbf{L}^2}\beta_{\mathbf{x}_h}^{-1/2} \leq C_{\mathbf{f}}\beta_{\mathbf{x}_h}^{-1/2},$$

und somit die letzte Aussage dieses Lemmas. □

4.3. Das Verfahren

Wir werden die Magnetisierung auch in der Zeit stückweise linear approximieren. Deshalb beginnen wir mit folgender Definition.

Definition 4.3.1. *Für eine Folge $(\varphi^j)_{j\geq 0}$ definieren wir die Notationen für die diskrete Zeitableitung d_t als*

$$d_t\varphi^j := k^{-1}(\varphi^j - \varphi^{j-1})$$

und die Mittelpunktsauswertung $\overline{\varphi}^{j+1/2}$ als

$$\overline{\varphi}^{j+1/2} := \frac{1}{2}(\varphi^j + \varphi^{j+1}).$$

Wir stellen nun ein Verfahren vor, das auf [BP06] beruht und die Feldbeiträge auf das gesamte effektive Feld erweitert. Die Terme niederer Ordnung berechnen wir dabei explizit, den Austauschterm implizit.

Algorithmus 4.3.2. *Sei* $\mathbf{m}_h^0 \in S^1(\mathcal{T}_h)$. *Finde Funktionen* $\mathbf{m}_h^j \in S^1(\mathcal{T}_h)$, *welche für* $j = 1, \dots, N$ *und für alle* $\boldsymbol{\phi}_h \in S^1(\mathcal{T}_h)$ *folgende Gleichung erfüllen*

$$(d_t \mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h = \alpha(\overline{\mathbf{m}}_h^{j+1/2} \times d_t \mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h - (\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, \boldsymbol{\phi}_h)_h \tag{4.6}$$

wobei wir das effektive Feld diskretisieren durch

$$\tilde{\mathbf{h}}_{\text{eff}}^{j+1/2} := C_{\text{ex}} \tilde{\Delta}_h \overline{\mathbf{m}}_h^{j+1/2} + \widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2} \ \textit{mit} \ \tilde{\mathbf{f}}_h^{j+1/2} := \tilde{\mathbf{f}}_h(t_{j+1/2}).$$

Unter welchen Voraussetzungen Gleichung (4.6) eine eindeutige Lösung hat, werden wir später in Satz 5.3.6 sehen.

5

Analyse des Verfahrens

5.1. Diskrete Energie und Längenerhaltung

Wir werden in weiterer Folge meist die zu (4.6) äquivalente Gleichung

$$(d_t\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h = \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h - (\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, \boldsymbol{\phi}_h)_h \tag{5.1}$$

verwenden, die sich aus der Gleichheit $\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1} = \overline{\mathbf{m}}_h^{j+1/2} \times d_t\mathbf{m}_h^{j+1}$ ergibt. Folgendes Lemma liefert ein diskretes Gegenstück zur Bedingung 3 aus der Definition des schwachen Lösungsbegriffs 2.2.7.

Lemma 5.1.1. *Die aus dem Algorithmus 4.3.2 hervorgehende Folge* $\{\mathbf{m}_h^j\}_{j\geq 0}$ *erfüllt für alle* $j \geq 0$

$$\frac{C_{\text{ex}}}{2} d_t \|\nabla \mathbf{m}_h^{j+1}\|_{\mathbf{L}^2}^2 + \alpha \|d_t\mathbf{m}_h^{j+1}\|_h^2 = (d_t\mathbf{m}_h^{j+1}, \widetilde{\Pi}_h(\mathbf{m}_h^j) + \tilde{\mathbf{f}}_h^{j+1/2})_h.$$

Beweis. Wir setzen $\boldsymbol{\phi}_h := d_t\mathbf{m}_h^{j+1} \in S^1(\mathcal{T}_h)$ in Algorithmus 4.3.2 ein und erhalten

$$(d_t\mathbf{m}_h^{j+1}, d_t\mathbf{m}_h^{j+1})_h = \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}, d_t\mathbf{m}_h^{j+1})_h - (\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, d_t\mathbf{m}_h^{j+1})_h.$$

Der erste Summand der rechten Seite lässt sich nach Bemerkung 3.2.6 durch den Term

$$\alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}, d_t\mathbf{m}_h^{j+1})_h = \alpha \sum_{\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)} \beta_{\mathbf{x}_h} \langle \mathbf{m}_h^j(\mathbf{x}_h) \times d_t\mathbf{m}_h^{j+1}(\mathbf{x}_h), d_t\mathbf{m}_h^{j+1}(\mathbf{x}_h)\rangle$$

beschreiben und ist aufgrund der Form $\langle a \times b, b\rangle = 0$ der einzelnen Summanden gleich Null. Damit erhalten wir

$$\|d_t\mathbf{m}_h^{j+1}\|_h^2 = -(\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, d_t\mathbf{m}_h^{j+1})_h.$$

Oder unter Anwendung der alternierenden Eigenschaft $\langle a \times b, c\rangle = -\langle a \times c, b\rangle$ des Spatproduktes

$$\|d_t\mathbf{m}_h^{j+1}\|_h^2 = (\overline{\mathbf{m}}_h^{j+1/2} \times d_t\mathbf{m}_h^{j+1}, \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2})_h.$$

Wir benützen, dass $\overline{\mathbf{m}}_h^{j+1/2} \times d_t\mathbf{m}_h^{j+1} = \frac{1}{2}(\mathbf{m}_h^j + \mathbf{m}_h^{j+1}) \times \frac{1}{k}(\mathbf{m}_h^{j+1} - \mathbf{m}_h^j) = \mathbf{m}_h^j \times \frac{1}{k}\mathbf{m}_h^{j+1} = \mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}$ gilt. Damit wird obige Gleichung zu

$$\|d_t\mathbf{m}_h^{j+1}\|_h^2 = (\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}, \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2})_h. \tag{5.2}$$

Nun setzen wir erneut in Algorithmus 4.3.2 ein, diesmal mit der Wahl unserer Testfunktion $\boldsymbol{\phi}_h := \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2} := C_{\text{ex}}\tilde{\Delta}_h\overline{\mathbf{m}}_h^{j+1/2} + \widetilde{\Pi}_h\mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}$, und sehen

$$(d_t\mathbf{m}_h^{j+1}, \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2})_h = \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}, \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2})_h - (\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2})_h.$$

Der letzte Beitrag der rechten Seite ist wieder von der Form $(a \times b, b)_h$ und deshalb Null. Es gilt also

$$(d_t\mathbf{m}_h^{j+1}, \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2})_h = \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}, \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2})_h. \tag{5.3}$$

Wir setzen nun Gleichungen (5.2) und (5.3) zusammen und erhalten

$$(d_t\mathbf{m}_h^{j+1}, C_{\text{ex}}\tilde{\Delta}_h\overline{\mathbf{m}}_h^{j+1/2} + \widetilde{\Pi}_h\mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2})_h = \alpha\|d_t\mathbf{m}_h^{j+1}\|_h^2. \tag{5.4}$$

Weiters lässt sich der Laplace-Term vereinfachen zu

$$\begin{aligned}(d_t\mathbf{m}_h^{j+1}, C_{\text{ex}}\tilde{\Delta}_h\overline{\mathbf{m}}_h^{j+1/2})_h &= -C_{\text{ex}}(\nabla d_t\mathbf{m}_h^{j+1}, \nabla\overline{\mathbf{m}}_h^{j+1/2}) \\ &= -\frac{C_{\text{ex}}}{2k}(\nabla\mathbf{m}_h^{j+1} - \nabla\mathbf{m}_h^j, \nabla\mathbf{m}_h^{j+1} + \nabla\mathbf{m}_h^j) \\ &= -\frac{C_{\text{ex}}}{2k}\big(\|\nabla\mathbf{m}_h^{j+1}\|_{\mathbf{L}^2(\Omega)}^2 - \|\nabla\mathbf{m}_h^j\|_{\mathbf{L}^2(\Omega)}^2\big) \\ &= -\frac{C_{\text{ex}}}{2}d_t\|\nabla\mathbf{m}_h^{j+1}\|_{\mathbf{L}^2(\Omega)}^2.\end{aligned} \tag{5.5}$$

Man beachte, dass im ersten Schritt durch die Definition von $\tilde{\Delta}_h$ auf das $\mathbf{L}^2$-Skalarprodukt übergegangen wird. Aus Gleichungen (5.3) und (5.5) erhält man die gewünschte Gleichheit. □

Satz 5.1.2. *Der Output von Algorithmus 4.3.2 erfüllt für alle $j = 1, \dots, N$*

$$|\mathbf{m}_h^j(\mathbf{x})| \leq 1 \quad \text{für alle } \mathbf{x} \in \overline{\Omega}, \tag{5.6}$$

sowie

$$|\mathbf{m}_h^j(\mathbf{x}_h)| = 1 \quad \text{für alle } \mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h). \tag{5.7}$$

Insbesondere gilt für alle $\mathbf{m}_h^j$ die Gleichheit $\|\mathbf{m}_h^j\|_{\mathbf{L}^\infty(\Omega)} = 1$.

Für den Beweis verwenden wir das nachfolgende Lemma.

Lemma 5.1.3. *Der Output von Algorithmus 4.3.2 erfüllt die Beziehung*

$$|\mathbf{m}_h^j(\mathbf{x}_h)| = |\mathbf{m}_h^{j+1}(\mathbf{x}_h)| \text{ für alle } j \geq 0 \text{ und alle } \mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h).$$

Beweis. Wir setzen dazu in Algorithmus 4.3.2 die Testfunktion $\boldsymbol{\phi}_h := \varphi_{\mathbf{x}_h}\overline{\mathbf{m}}_h^{j+1/2}(\mathbf{x}_h)$ ein und bringen alle Ausdrücke auf eine Seite.

$$\begin{aligned}0 = (d_t\mathbf{m}_h^{j+1}, \varphi_{\mathbf{x}_h}\overline{\mathbf{m}}_h^{j+1/2}(\mathbf{x}_h))_h &- \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}, \varphi_{\mathbf{x}_h}\overline{\mathbf{m}}_h^{j+1/2}(\mathbf{x}_h))_h \\ &+ (\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, \varphi_{\mathbf{x}_h}\overline{\mathbf{m}}_h^{j+1/2}(\mathbf{x}_h))_h.\end{aligned} \tag{5.8}$$

Wir bemerken, dass $\varphi_{\mathbf{x}_h}(\mathbf{z}_h) = 0$ für alle $\mathbf{x}_h \neq \mathbf{z}_h \in \mathcal{N}(\mathcal{T}_h)$ gilt, und erhalten damit nach Bemerkung 3.2.6

$$(\boldsymbol{\phi}_h, \varphi_{\mathbf{x}_h}\boldsymbol{\xi})_h = \beta_{\mathbf{x}_h}\langle\boldsymbol{\phi}_h(\mathbf{x}_h), \boldsymbol{\xi}(\mathbf{x}_h)\rangle.$$

Dies benützen wir um Gleichung (5.8) weiter umzuformen.

$$0 = \beta_{\mathbf{x}_h}\langle d_t\mathbf{m}_h^{j+1}(\mathbf{x}_h), \overline{\mathbf{m}}_h^{j+1/2}(\mathbf{x}_h)\rangle - \alpha\beta_{\mathbf{x}_h}\langle(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1})(\mathbf{x}_h), \overline{\mathbf{m}}_h^{j+1/2}(\mathbf{x}_h)\rangle$$
$$\beta_{\mathbf{x}_h}\underbrace{\langle(\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2})(\mathbf{x}_h), \overline{\mathbf{m}}_h^{j+1/2}(\mathbf{x}_h)\rangle}_{=0}.$$

Das letzte Skalarprodukt ist von der Form $\langle a \times b, a\rangle$ und deswegen Null. Nach Definitionen von $d_t\mathbf{m}_h^{j+1}$ und $\overline{\mathbf{m}}_h^{j+1/2}$ erhalten wir weiters:

$$\begin{aligned}0 =&\beta_{\mathbf{x}_h}\frac{1}{2k}\langle\mathbf{m}_h^{j+1}(\mathbf{x}_h) - \mathbf{m}_h^j(\mathbf{x}_h), \mathbf{m}_h^{j+1}(\mathbf{x}_h) + \mathbf{m}_h^j(\mathbf{x}_h)\rangle\\ &- \alpha\beta_{\mathbf{x}_h}\frac{1}{2k}\langle\mathbf{m}_h^j \times (\mathbf{m}_h^{j+1}(\mathbf{x}_h) - \mathbf{m}_h^j(\mathbf{x}_h)), \mathbf{m}_h^{j+1}(\mathbf{x}_h) + \mathbf{m}_h^j(\mathbf{x}_h)\rangle\end{aligned}$$

Unter Benützung der Bilinearität und Symmetrie des Skalarproduktes gilt

$$\begin{aligned}0 =&\beta_{\mathbf{x}_h}\frac{1}{2k}\left(\langle\mathbf{m}_h^{j+1}(\mathbf{x}_h), \mathbf{m}_h^{j+1}(\mathbf{x}_h)\rangle - \langle\mathbf{m}_h^j(\mathbf{x}_h), \mathbf{m}_h^j(\mathbf{x}_h)\rangle\right)\\ &- \alpha\beta_{\mathbf{x}_h}\frac{1}{2k}\underbrace{\langle\mathbf{m}_h^j(\mathbf{x}_h) \times (\mathbf{m}_h^{j+1}(\mathbf{x}_h) - \mathbf{m}_h^j(\mathbf{x}_h)), \mathbf{m}_h^{j+1}(\mathbf{x}_h)\rangle}_{=\langle\mathbf{m}_h^j(\mathbf{x}_h)\times\mathbf{m}_h^{j+1}(\mathbf{x}_h), \mathbf{m}_h^{j+1}(\mathbf{x}_h)\rangle=0}\end{aligned}$$

und damit

$$0 = \left(\langle\mathbf{m}_h^{j+1}(\mathbf{x}_h), \mathbf{m}_h^{j+1}(\mathbf{x}_h)\rangle - \langle\mathbf{m}_h^j(\mathbf{x}_h), \mathbf{m}_h^j(\mathbf{x}_h)\rangle\right).$$

Hieraus folgt schließlich

$$|\mathbf{m}_h^j(\mathbf{x}_h)| = |\mathbf{m}_h^{j+1}(\mathbf{x}_h)| \quad \text{für alle } \mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)$$

und damit die gewünschte Eigenschaft. □

Beweis von Satz 5.1.2. Wir zeigen Gleichung (5.6). Da die Funktionswerte $\mathbf{m}_h^j(\mathbf{x})$ für alle $\mathbf{x} \in \Omega$ lediglich Konvexkombinationen der entsprechenden Knotenwerte $\mathbf{m}_h^j(\mathbf{x}_h)$ darstellen, folgt die Eigenschaft $|\mathbf{m}_h^j(\mathbf{x})| \leq 1$ aus der Konvexität der Einheitskugel $B^3 := \{\mathbf{x} \in \mathbb{R}^3 : |\mathbf{x}| \leq 1\} \supseteq \mathbb{S}^2 \supseteq \{\mathbf{m}_h^j(\mathbf{x}_h) : \mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)\}$. Das sieht man wie folgt. Sei $\mathcal{N}(T) := \{\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h) : \mathbf{x}_h \in T\}$. Für alle $T \in \mathcal{T}_h$ und $\mathbf{x} \in T$ gibt es baryzentrische Koordinaten $a_{T,\mathbf{x}_h} \in [0,1]$ mit $\sum_{\mathbf{x}_h\in\mathcal{N}(T)} a_{T,\mathbf{x}_h} = 1$, sodass gilt $\mathbf{x} = \sum_{\mathbf{x}_h\in\mathcal{N}(T)} a_{T,\mathbf{x}_h}\mathbf{x}_h$. Für die affine Funktion $\mathbf{m}_h^j\big|_T(\mathbf{x})$ gilt also:

$$|\mathbf{m}_h^j\big|_T(\mathbf{x})| = |\mathbf{m}_h^j\Big(\sum_{\mathbf{x}_h\in\mathcal{N}(T)} a_{T,\mathbf{x}_h}\mathbf{x}_h\Big)| = |\sum_{\mathbf{x}_h\in\mathcal{N}(T)} a_{T,\mathbf{x}_h}\mathbf{m}_h^j(\mathbf{x}_h)| \leq \sum_{\mathbf{x}_h\in\mathcal{N}(T)} a_{T,\mathbf{x}_h}\underbrace{|\mathbf{m}_h^j(\mathbf{x}_h)|}_{=1} = 1.$$

Gleichung (5.7) folgt aus Lemma 5.1.3 in Kombination mit $|\mathbf{m}_h^0(\mathbf{x}_h)| = 1$. Für die Funktionen $\mathbf{m}_h^j \in C(\overline{\Omega})$ gilt aufgrund der Stetigkeit und weil in den Knoten das Maximum angenommen wird

$$\|\mathbf{m}_h^j\|_{\mathbf{L}^\infty(\overline{\Omega})} = \max_{\mathbf{x}\in\overline{\Omega}}|\mathbf{m}_h^j(\mathbf{x})| = |\mathbf{m}_h^j(\mathbf{x}_h)| = 1.$$

□

Definition 5.1.4. *Für $\mathbf{x} \in \Omega$ und $t \in [t_j, t_{j+1})$ definieren wir*

$$\begin{aligned}
\mathbf{m}_{hk}(t,\mathbf{x}) &:= \frac{t-t_j}{k}\mathbf{m}_h^{j+1}(\mathbf{x}) + \frac{t_{j+1}-t}{k}\mathbf{m}_h^{j}(\mathbf{x}) \\
\mathbf{m}_{hk}^{-}(t,\mathbf{x}) &:= \mathbf{m}_h^{j}(\mathbf{x}) \\
\mathbf{m}_{hk}^{+}(t,\mathbf{x}) &:= \mathbf{m}_h^{j+1}(\mathbf{x}) \\
\overline{\mathbf{m}}_{hk}(t,\mathbf{x}) &:= \frac{1}{2}\big(\mathbf{m}_h^{j}(\mathbf{x}) + \mathbf{m}_h^{j+1}(\mathbf{x})\big)
\end{aligned}$$

Lemma 5.1.5. *Die Funktionenfolgen aus Definition 5.1.4 bleiben beschränkt. Insbesondere gilt für alle $h, k > 0$*

$$\|\mathbf{m}_{hk}\|_{\mathbf{L}^\infty(\Omega_T)} = \|\mathbf{m}_{hk}^{+}\|_{\mathbf{L}^\infty(\Omega_T)} = \|\mathbf{m}_{hk}^{-}\|_{\mathbf{L}^\infty(\Omega_T)} = 1$$

und

$$\|\overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^\infty(\Omega_T)} \leq 1.$$

Beweis. Für die Funktionen $\mathbf{m}_{hk}^{+}$ und $\mathbf{m}_{hk}^{-}$ folgt dies sofort aus Satz 5.1.2, da die gewünschte Eigenschaft hier sogar punktweise gilt. Die beiden Funktionen $\mathbf{m}_{hk}$ und $\overline{\mathbf{m}}_{hk}$ erfüllen als Konvexkombinationen der $\mathbf{m}_h^j$ sicherlich

$$\|\mathbf{m}_{hk}\|_{\mathbf{L}^\infty(\Omega_T)} \leq 1 \text{ und } \|\overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^\infty(\Omega_T)} \leq 1.$$

Wegen der Zeitstetigkeit von $\|\mathbf{m}_{hk}(t,.)\|_{\mathbf{L}^\infty(\Omega)}$ gilt sogar für alle $j = 0, ..., N$

$$\|\mathbf{m}_{hk}\|_{\mathbf{L}^\infty(\Omega_T)} = \max_{t\in[0,T]} \|\mathbf{m}_{hk}(t,\mathbf{x})\|_{\mathbf{L}^\infty(\Omega)} \geq \max_j \|\mathbf{m}_h^j\|_{\mathbf{L}^\infty(\Omega)} = 1.$$

Damit gilt die Aussage auch für $\mathbf{m}_{hk}$. □

5.2. Konvergenzanalyse

Lemma 5.2.1. *Für alle $t' \in [0,T]$ und $\varepsilon > 0$ gilt die Energieabschätzung*

$$\frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_{hk}^{+}(t')\|_{\mathbf{L}^2}^2 + (\alpha - \varepsilon)\int_0^{t'} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt \leq \frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_{hk}(0)\|_{\mathbf{L}^2}^2 + \frac{T}{2\varepsilon}(C_\Pi^2 + C_{\mathbf{f}}^2)$$

Beweis. Wir verwenden dazu Lemma 5.1.1. Es gilt also

$$\frac{C_{\text{ex}}}{2} d_t \|\nabla \mathbf{m}_h^{j+1}\|_{\mathbf{L}^2}^2 + \alpha \|d_t \mathbf{m}_h^{j+1}\|_h^2 = (d_t \mathbf{m}_h^{j+1}, \widetilde{\Pi}_h(\mathbf{m}_h^j) + \tilde{\mathbf{f}}_h^{j+1/2})_h.$$

Wir wählen $n := \min\{j : t_j > t'\}$. Wir summieren die Zeitschritte von $j = 0, \ldots, n-1$ und erhalten

$$\frac{C_{\text{ex}}}{2}\sum_{j=0}^{n-1} d_t \|\nabla \mathbf{m}_h^{j+1}\|_{\mathbf{L}^2}^2 + \alpha \sum_{j=0}^{n-1} \|d_t \mathbf{m}_h^{j+1}\|_h^2 = \sum_{j=0}^{n-1} (d_t \mathbf{m}_h^{j+1}, \widetilde{\Pi}_h(\mathbf{m}_h^j) + \tilde{\mathbf{f}}_h^{j+1/2})_h.$$

Durch die Gleichheit $d_t \|\nabla \mathbf{m}_h^{j+1}\|_{\mathbf{L}^2}^2 = k^{-1}(\|\nabla \mathbf{m}_h^{j+1}\|_{\mathbf{L}^2}^2 - \|\nabla \mathbf{m}_h^{j}\|_{\mathbf{L}^2}^2)$ wird die erste Summe zur Teleskopsumme und wir erhalten

$$\frac{C_{\text{ex}}}{2k}(\|\nabla \mathbf{m}_h^{n}\|_{\mathbf{L}^2}^2 - \|\nabla \mathbf{m}_h^{0}\|_{\mathbf{L}^2}^2) + \alpha \sum_{j=0}^{n-1} \|d_t \mathbf{m}_h^{j+1}\|_h^2 = \sum_{j=0}^{n-1} (d_t \mathbf{m}_h^{j+1}, \widetilde{\Pi}_h(\mathbf{m}_h^j) + \tilde{\mathbf{f}}_h^{j+1/2})_h.$$

Nach Multiplikation mit k gilt

$$\frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_h^n\|_{\mathbf{L}^2}^2 + \alpha \sum_{j=0}^{n-1} k\|d_t \mathbf{m}_h^{j+1}\|_h^2 = \frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_h^0\|_{\mathbf{L}^2}^2 + \sum_{j=0}^{n-1} k(d_t \mathbf{m}_h^{j+1}, \widetilde{\Pi}_h(\mathbf{m}_h^j) + \tilde{\mathbf{f}}_h^{j+1/2})_h.$$

Im nächsten Schritt nutzen wir aus, dass $\partial_t \mathbf{m}_{hk}$ konstant in jedem Zeitintervall ist, d.h. es gilt $k\|d_t \mathbf{m}_h^{j+1}\|_h^2 = \int_{t_j}^{t_{j+1}} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt$ und damit

$$\frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_h^n\|_{\mathbf{L}^2}^2 + \alpha \sum_{j=0}^{n-1} \int_{t_j}^{t_{j+1}} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt = \frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_h^0\|_{\mathbf{L}^2}^2 + \sum_{j=0}^{n-1} k(d_t \mathbf{m}_h^{j+1}, \widetilde{\Pi}_h(\mathbf{m}_h^j) + \tilde{\mathbf{f}}_h^{j+1/2})_h,$$

also etwas umgeschrieben

$$\frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_h^n\|_{\mathbf{L}^2}^2 + \alpha \int_{t_0}^{t_n} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt = \frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_{hk}(0)\|_{\mathbf{L}^2}^2 + \sum_{j=0}^{n-1} k(d_t \mathbf{m}_h^{j+1}, \widetilde{\Pi}_h(\mathbf{m}_h^j) + \tilde{\mathbf{f}}_h^{j+1/2})_h.$$

Nun schätzen wir den rechten Term mit der Cauchy-Schwarzschen Ungleichung ab.

$$\begin{aligned}\frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_h^n\|_{\mathbf{L}^2}^2 + \alpha \int_0^{t_n} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt \leq{}& \frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_{hk}(0)\|_{\mathbf{L}^2}^2 \\ &+ \sum_{j=0}^{n-1} k\|d_t \mathbf{m}_h^{j+1}\|_h (\|\widetilde{\Pi}_h(\mathbf{m}_h^j)\|_h + \|\tilde{\mathbf{f}}_h^{j+1/2}\|_h).\end{aligned}$$

Mit der Young-Ungleichung aus Lemma A.3.5 für $\varepsilon/2$ erhalten wir weiters

$$\begin{aligned}\frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_h^n\|_{\mathbf{L}^2}^2 + \alpha \int_0^{t_n} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt \leq{}& \frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_{hk}(0)\|_{\mathbf{L}^2}^2 + 2\sum_{j=0}^{n-1} \frac{\varepsilon}{2} \int_{t_j}^{t_{j+1}} \|\partial_t \mathbf{m}_h^j\|_h^2 dt \\ &+ \sum_{j=0}^{n-1} \frac{1}{2\varepsilon} k\|\widetilde{\Pi}_h(\mathbf{m}_h^j)\|_h^2 + \sum_{j=0}^{n-1} \frac{1}{2\varepsilon} k\|\tilde{\mathbf{f}}_h^{j+1/2}\|_h^2.\end{aligned}$$

Wir bringen nun den Zeitableitungsterm auf die linke Seite und erhalten

$$\frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_h^n\|_{\mathbf{L}^2}^2 + (\alpha - \varepsilon) \int_0^{t_n} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt \leq \frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_{hk}(0)\|_{\mathbf{L}^2}^2 + \sum_{j=0}^{n-1} \frac{1}{2\varepsilon} k(\|\widetilde{\Pi}_h(\mathbf{m}_h^j)\|_h^2 + \|\tilde{\mathbf{f}}_h^{j+1/2}\|_h^2).$$

Mit $t' \in [t_{n-1}, t_n)$ gilt dann einerseits $\nabla \mathbf{m}_h^n = \nabla \mathbf{m}_{hk}^+(t')$ und andererseits $\int_0^{t'} \|\cdot\| dt \leq \int_0^{t_n} \|\cdot\| dt$. Damit können wir obige Gleichung also nach unten abschätzen und erhalten nach Lemma 4.2.4

$$\frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_{hk}^+(t')\|_{\mathbf{L}^2}^2 + (\alpha - \varepsilon) \int_0^{t'} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt \leq \frac{C_{\text{ex}}}{2}\|\nabla \mathbf{m}_{hk}(0)\|_{\mathbf{L}^2}^2 + \frac{T}{2\varepsilon}(C_\Pi^2 + C_{\mathbf{f}}^2)$$

die gesuchte Energieabschätzung. □

Lemma 5.2.2. *Es existiert ein $C_{5.2.2} > 0$, sodass für alle $h, k > 0$ gilt*

$$\begin{aligned}\|\nabla \mathbf{m}_{hk}^{\pm}\|_{\mathbf{L}^2(\Omega_T)}^2 &\leq C_{5.2.2}, \\ \|\nabla \overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^2(\Omega_T)}^2 &\leq C_{5.2.2}, \\ \|\mathbf{m}_{hk}\|_{\mathbf{H}^1(\Omega_T)}^2 &\leq C_{5.2.2}.\end{aligned}$$

Dieses $C_{5.2.2}$ hängt dabei lediglich von den Daten $\alpha > 0$, $\sup_{h>0}\|\nabla \mathbf{m}_h^0\|_{\mathbf{L}^2}$, $|\Omega_T|$, T, C_{ex}, C_Π und $C_{\mathbf{f}}$ ab.

Beweis. Wir bemerken zuerst, dass es wegen Lemma 5.2.1 ein festes c gibt, sodass für alle $h, k > 0$ und alle $t \in [0, T]$ die Abschätzungen

$$\|\nabla \mathbf{m}_{hk}^{+}(t)\|_{\mathbf{L}^2(\Omega)}^2 \leq c \tag{5.9}$$

$$\int_0^t \|\partial_t \mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega)}^2 dt \leq c \tag{5.10}$$

gelten. Wir erhalten also die Beschränktheit der Folge $(\nabla \mathbf{m}_{hk}^{+})$ mittels

$$\|\nabla \mathbf{m}_{hk}^{+}\|_{\mathbf{L}^2(\Omega_T)}^2 = \int_0^T \underbrace{\|\nabla \mathbf{m}_{hk}^{+}(t)\|_{\mathbf{L}^2(\Omega)}^2}_{\leq c} dt \leq Tc.$$

Für $\|\nabla \mathbf{m}_{hk}^{-}\|_{\mathbf{L}^2(\Omega_T)}^2$ bemühen wir genauso den Satz von Fubini und nützen aus, dass $\nabla \mathbf{m}_{hk}^{-}$ und $\nabla \mathbf{m}_{hk}^{+}$ auf geeignet gewählten Zeitintervallen übereinstimmen.

$$\begin{aligned}
\|\nabla \mathbf{m}_{hk}^{-}\|_{\mathbf{L}^2(\Omega_T)}^2 &= \int_0^T \|\nabla \mathbf{m}_{hk}^{-}\|_{\mathbf{L}^2(\Omega)}^2 dt \\
&= \sum_{i=0}^{n-1} \int_{t_j}^{t_{j+1}} \|\nabla \mathbf{m}_{hk}^{-}\|_{\mathbf{L}^2(\Omega)}^2 dt \\
&= \int_0^{t_1} \|\nabla \mathbf{m}_h^0\|_{\mathbf{L}^2(\Omega)}^2 dt + \sum_{j=1}^{n-1} \int_{t_j}^{t_{j+1}} \|\nabla \mathbf{m}_h^j\|_{\mathbf{L}^2(\Omega)}^2 dt \\
&= k\|\nabla \mathbf{m}_h^0\|_{\mathbf{L}^2(\Omega)}^2 + k \sum_{j=1}^{n-1} \|\nabla \mathbf{m}_h^j\|_{\mathbf{L}^2(\Omega)}^2 \\
&= k\|\nabla \mathbf{m}_h^0\|_{\mathbf{L}^2(\Omega)}^2 + k \sum_{j=0}^{n-2} \|\nabla \mathbf{m}_h^{j+1}\|_{\mathbf{L}^2(\Omega)}^2 \\
&\leq \int_0^{t_1} \|\nabla \mathbf{m}_{hk}^{-}\|_{\mathbf{L}^2(\Omega)}^2 dt + \|\nabla \mathbf{m}_{hk}^{+}\|_{\mathbf{L}^2(\Omega_T)}^2 \\
&\leq k\|\nabla \mathbf{m}_h^0\|_{\mathbf{L}^2(\Omega)}^2 + Tc
\end{aligned}$$

Da nun $\mathbf{m}_h^0$, wie in (4.1) vorausgesetzt, gegen $\mathbf{m}^0$ in $\mathbf{H}^1(\Omega)$ konvergiert, bleibt insbesondere $\|\nabla \mathbf{m}_h^0\|_{\mathbf{L}^2(\Omega)}^2$ beschränkt. Damit erhalten wir die gleichmäßige Beschränktheit von $\nabla \mathbf{m}_{hk}^{-}$ in $\mathbf{L}^2(\Omega_T)$.

Für $\nabla \overline{\mathbf{m}}_{hk}$ benützen wir die Linearität des Gradienten und erhalten

$$\|\nabla \overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^2(\Omega_T)} = \frac{1}{2}\|\nabla \mathbf{m}_{hk}^{-} + \nabla \mathbf{m}_{hk}^{+}\|_{\mathbf{L}^2(\Omega_T)} \leq \frac{1}{2}(\|\nabla \mathbf{m}_{hk}^{-}\|_{\mathbf{L}^2(\Omega_T)} + \|\nabla \mathbf{m}_{hk}^{+}\|_{\mathbf{L}^2(\Omega_T)}).$$

Also folgt die Beschränktheit von $\nabla \overline{\mathbf{m}}_{hk}$ aus der Beschränktheit von $\nabla \mathbf{m}_{hk}^{-}$ und $\nabla \mathbf{m}_{hk}^{+}$.

Nun fehlt noch zu zeigen, dass $\|\mathbf{m}_{hk}\|_{\mathbf{H}^1(\Omega_T)}^2$ beschränkt bleibt. Dazu spalten wir die Norm definitionsgemäß in die einzelnen Beiträge auf:

$$\|\mathbf{m}_{hk}\|_{\mathbf{H}^1(\Omega_T)}^2 = \|\mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega_T)}^2 + \|\partial_t \mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega_T)}^2 + \|\nabla \mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega_T)}^2$$

Die Abschätzung des ersten Summanden folgt aus der Beschränktheit des Gebietes, sowie der Beschränktheit von $\mathbf{m}_{hk}$

$$\|\mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega_T)} \leq \|\mathbf{m}_{hk}\|^2_{\mathbf{L}^\infty(\Omega_T)}|\Omega_T| = |\Omega_T| < \infty.$$

Nach Voraussetzung (5.10) gilt weiters

$$\|\partial_t \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega_T)} = \int_0^T \|\partial_t \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega)} dt \leq c.$$

Die Beschränktheit von $\nabla \mathbf{m}_{hk}$ in $\mathbf{L}^2(\Omega_T)$ ergibt sich erneut aus dem Zusammenhang mit $\nabla \mathbf{m}^-_{hk}$ und $\nabla \mathbf{m}^+_{hk}$.

$$\begin{aligned}
\|\nabla \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega_T)} &= \int_0^T \|\nabla \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega)} dt \\
&= \sum_{i=0}^{n-1} \int_{t_j}^{t_{j+1}} \|\nabla \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega)} dt \\
&= \sum_{i=0}^{n-1} \int_{t_j}^{t_{j+1}} \|\frac{t-t_j}{k} \nabla \mathbf{m}_h^{j+1} + \frac{t_{j+1}-t}{k} \nabla \mathbf{m}_h^j\|^2_{\mathbf{L}^2(\Omega)} dt.
\end{aligned}$$

Wir benützen nun die Ungleichung $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$ und erhalten

$$\begin{aligned}
\|\nabla \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega_T)} &\leq \sum_{i=0}^{n-1} \int_{t_j}^{t_{j+1}} 2\frac{t-t_j}{k} \|\nabla \mathbf{m}_h^{j+1}\|^2_{\mathbf{L}^2(\Omega)} + 2\frac{t_{j+1}-t}{k} \|\nabla \mathbf{m}_h^j\|^2_{\mathbf{L}^2(\Omega)} dt \\
&\leq \sum_{i=0}^{n-1} \int_{t_j}^{t_{j+1}} 2\|\nabla \mathbf{m}_h^{j+1}\|^2_{\mathbf{L}^2(\Omega)} + 2\|\nabla \mathbf{m}_h^j\|^2_{\mathbf{L}^2(\Omega)} dt \\
&= 2\|\nabla \mathbf{m}^+_{hk}\|^2_{\mathbf{L}^2(\Omega_T)} + 2\|\nabla \mathbf{m}^-_{hk}\|^2_{\mathbf{L}^2(\Omega_T)}.
\end{aligned}$$

Damit bleibt auch $\|\mathbf{m}_{hk}\|^2_{\mathbf{H}^1(\Omega_T)}$ beschränkt und, um den Beweis zu vervollständigen, sei abschließend $C_{5.2.2}$ als Summe der Schranken gewählt. □

Lemma 5.2.3. *Es existiert ein* $\mathbf{m} \in \mathbf{H}^1(\Omega_T)$*, sodass*

$$\mathbf{m}_{hk} \xrightarrow{sub} \mathbf{m} \text{ in } \mathbf{H}^1(\Omega_T), \tag{5.11}$$

$$\mathbf{m}_{hk}, \mathbf{m}^-_{hk}, \mathbf{m}^+_{hk}, \overline{\mathbf{m}}_{hk} \xrightarrow{sub} \mathbf{m} \text{ in } \mathbf{L}^2(\Omega_T), \tag{5.12}$$

$$\nabla \mathbf{m}^-_{hk}, \nabla \mathbf{m}^+_{hk}, \nabla \overline{\mathbf{m}}_{hk} \xrightarrow{sub} \nabla \mathbf{m} \text{ in } \mathbf{L}^2(\Omega_T). \tag{5.13}$$

Hierbei gibt es sogar eine Teilfolge, für die alle obigen Konvergenzen erfüllt sind. Für dieses $\mathbf{m}$ *gilt* $|\mathbf{m}(t,\mathbf{x})| = 1$ *fast überall, d.h.* $\|\mathbf{m}\|_{\mathbf{L}^\infty(\Omega_T)} = 1$.

Beweis. Die Existenz von $\mathbf{m} \in \mathbf{H}^1(\Omega_T)$ und die schwache Konvergenz aus (5.11) erhalten wir direkt aus Satz A.4.1 und Lemma 5.2.2. Wir verwenden den Satz von Rellich A.4.5 und erhalten

$\mathbf{m}_{hk} \xrightarrow{\text{sub}} \mathbf{m}$ in $\mathbf{L}^2(\Omega_T)$. Nun betrachten wir $\|\mathbf{m}_{hk} - \mathbf{m}_{hk}^-\|^2_{\mathbf{L}^2(\Omega_T)}$. Mittels Satz von Fubini gilt

$$\begin{aligned}
\|\mathbf{m}_{hk} - \mathbf{m}_{hk}^-\|^2_{\mathbf{L}^2(\Omega_T)} &= \sum_{j=0}^{N-1} \int_{t_j}^{t_{j+1}} \|\mathbf{m}_h^{j+1} \frac{t-t_j}{k} + \mathbf{m}_h^j \frac{t_{j+1}-k-t}{k}\|^2_{\mathbf{L}^2(\Omega)} dt \\
&= \sum_{j=0}^{N-1} \int_{t_j}^{t_{j+1}} \|(t-t_j) d_t \mathbf{m}_h^{j+1}\|^2_{\mathbf{L}^2(\Omega)} dt \\
&= \sum_{j=0}^{N-1} \|d_t \mathbf{m}_h^{j+1}\|^2_{\mathbf{L}^2(\Omega)} \int_{t_j}^{t_{j+1}} (t-t_j)^2 dt \\
&= \frac{k^3}{3} \sum_{j=0}^{N-1} \|d_t \mathbf{m}_h^{j+1}\|^2_{\mathbf{L}^2(\Omega)} \\
&= \frac{k^2}{3} \sum_{j=0}^{N-1} \int_{t_j}^{t_{j+1}} \|d_t \mathbf{m}_h^{j+1}\|^2_{\mathbf{L}^2(\Omega)} \\
&= \frac{k^2}{3} \|\partial_t \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega_T)}
\end{aligned}$$

Und da $\|\partial_t \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega_T)}$ wegen der Energieabschätzung 5.2.1 kombiniert mit der Normäquivalenz 3.2.7 beschränkt bleibt, geht $\frac{k^2}{3}\|\partial_t \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega_T)}$ gegen 0. Haben wir nun eine Teilfolge $\mathbf{m}_{h_\ell k_\ell}$ für die gilt $\mathbf{m}_{h_\ell k_\ell} \to \mathbf{m}$ in $\mathbf{L}^2(\Omega_T)$ dann erhalten wir also auch $\|\mathbf{m}_{h_\ell k_\ell} - \mathbf{m}_{h_\ell k_\ell}^-\|^2_{\mathbf{L}^2(\Omega_T)} \to 0$. Mittels Dreiecksungleichung gilt

$$\|\mathbf{m} - \mathbf{m}_{h_\ell k_\ell}^-\|_{\mathbf{L}^2(\Omega_T)} \leq \|\mathbf{m} - \mathbf{m}_{h_\ell k_\ell}\|_{\mathbf{L}^2(\Omega_T)} + \|\mathbf{m}_{h_\ell k_\ell} - \mathbf{m}_{h_\ell k_\ell}^-\|_{\mathbf{L}^2(\Omega_T)} \to 0.$$

Die Konvergenz von $\mathbf{m}_{h_\ell k_\ell}^+$ erhalten wir mit folgendem Argument. Die Funktion $\mathbf{m}_{hk}$ ist stückweise affin und erfüllt deshalb $\mathbf{m}_{hk}(t_j + t, \mathbf{x}) - \mathbf{m}_{hk}(t_j, \mathbf{x}) = \mathbf{m}_{hk}(t_j + t + \alpha, \mathbf{x}) - \mathbf{m}_{hk}(t_j + \alpha, \mathbf{x})$ für $0 < t < k$. Wir wählen $\alpha := t_{j+1} - t_j - t$ und erhalten

$$\begin{aligned}
\mathbf{m}_{hk}(t_j + t, \mathbf{x}) - \mathbf{m}_{hk}^-(t_j + t, \mathbf{x}) &= \mathbf{m}_{hk}(t_j + t, \mathbf{x}) - \mathbf{m}_{hk}(t_j, \mathbf{x}) \\
&= \mathbf{m}_{hk}(t_j + t + \alpha, \mathbf{x}) - \mathbf{m}_{hk}(t_j + \alpha, \mathbf{x}) \\
&= \mathbf{m}_{hk}(t_{j+1}, \mathbf{x}) - \mathbf{m}_{hk}(t_{j+1} - t, \mathbf{x}) \\
&= -(\mathbf{m}_{hk}(t_{j+1} - t, \mathbf{x}) - \mathbf{m}_{hk}^+(t_{j+1} - t, \mathbf{x})).
\end{aligned}$$

Aufgrund dessen lässt sich durch stückweises Umdrehen der Integrationsrichtung die Gleichheit der Normen

$$\|\mathbf{m}_{hk} - \mathbf{m}_{hk}^-\|_{\mathbf{L}^2(\Omega_T)} = \|\mathbf{m}_{hk} - \mathbf{m}_{hk}^+\|_{\mathbf{L}^2(\Omega_T)}$$

und somit die Konvergenz folgern. Wir erhalten also die Konvergenzen $\mathbf{m}_{h_\ell k_\ell}^+ \to \mathbf{m}$ und $\mathbf{m}_{h_\ell k_\ell}^- \to \mathbf{m}$ in $\mathbf{L}^2(\Omega_T)$. Für den Mittelwert $\overline{\mathbf{m}}_{hk} := \frac{1}{2}(\mathbf{m}_{hk}^+ + \mathbf{m}_{hk}^-)$ gilt damit sofort

$$\|\mathbf{m} - \overline{\mathbf{m}}_{h_\ell k_\ell}\|_{\mathbf{L}^2(\Omega_T)} \leq \frac{1}{2}\|\mathbf{m} - \mathbf{m}_{h_\ell k_\ell}^+\|_{\mathbf{L}^2(\Omega_T)} + \frac{1}{2}\|\mathbf{m} - \mathbf{m}_{h_\ell k_\ell}^-\|_{\mathbf{L}^2(\Omega_T)} \to 0.$$

Um (5.13) zu zeigen, wählen wir nun eine beliebige Testfunktion $\boldsymbol{\phi} \in C_c^\infty(\Omega_T;\mathbb{R}^{3\times 3})$.

$$\begin{aligned}\lim_{\ell\to\infty}(\nabla \mathbf{m}^-_{h_\ell k_\ell}, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)} &= -\lim_{\ell\to\infty}(\mathbf{m}^-_{h_\ell k_\ell}, \underbrace{\operatorname{div}\boldsymbol{\phi}}_{\in C_c^\infty(\Omega_T;\mathbb{R}^3)})_{\mathbf{L}^2(\Omega_T)}\\ &= -(\mathbf{m}, \operatorname{div}\boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}\\ &= (\nabla\mathbf{m}, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}.\end{aligned}$$

Dabei gilt die erste Gleichheit indem wir zeilenweise partielle Integration A.7.3 anwenden. Um die matrixwertige Gleichheit kompakt auszudrücken werden hier die Definitionen 2.2.3 und 2.2.4 verwendet. Die zweite Gleichheit gilt wegen der starken Konvergenz von $\mathbf{m}^-_{h_\ell k_\ell}$ und die dritte, weil $\mathbf{m}$ in $\mathbf{H}^1(\Omega_T)$ liegt und wir erneut Korollar A.7.3 anwenden können. Die Aussagen für $\nabla\mathbf{m}^+_{hk}$ und $\nabla\overline{\mathbf{m}}_{hk}$ folgen analog.

Nun zeigen wir die letzte Aussage dieses Lemmas. Diese erhalten wir durch Anwendung der Dreiecksungleichung und des verallgemeinerten Mittelwertsatzes der Differentialrechnung für beliebige $t \in [0,T]$, $\mathbf{x} \in \Omega$ und $\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)$

$$\begin{aligned}\big||\mathbf{m}^-_{hk}(t,\mathbf{x})| - 1\big| = \big||\mathbf{m}^-_{hk}(t,\mathbf{x})| - |\mathbf{m}^-_{hk}(t,\mathbf{x}_h)|\big| &\le |\mathbf{m}^-_{hk}(t,\mathbf{x}) - \mathbf{m}^-_{hk}(t,\mathbf{x}_h)|\\ &\le \max_{\mathbf{y}\in\overline{\mathbf{x}_h\mathbf{x}}} |\nabla\mathbf{m}^-_{hk}(t,\mathbf{y})||\mathbf{x}-\mathbf{x}_h|.\end{aligned}$$

Für jedes $\mathbf{x}$ wählen wir nun einen Knoten $\mathbf{x}_h$, sodass $\mathbf{x}$ und $\mathbf{x}_h$ im gleichen Element $T_{\mathbf{x}}$ liegen. Da $\nabla\mathbf{m}^-_{hk}$ elementweise konstant ist, können wir obige Gleichung mit der Schrittweitenfunktion h durch

$$\big||\mathbf{m}^-_{hk}(t,\mathbf{x})| - 1\big| \le h\big|\nabla\mathbf{m}^-_{hk}|_{T_{\mathbf{x}}}\big|$$

endgültig nach oben abschätzen. Wir integrieren und erhalten

$$\||\mathbf{m}^-_{hk}| - 1\|_{\mathbf{L}^2(\Omega_T)} \le h\|\nabla\mathbf{m}^-_{hk}\|_{\mathbf{L}^2(\Omega_T)}.$$

Also die Konvergenz $|\mathbf{m}^-_{hk}| \to 1$ in $\mathbf{L}^2(\Omega_T;\mathbb{R}^3)$. Womit mit der Dreiecksungleichung abschließend gilt

$$\||\mathbf{m}| - 1\|_{\mathbf{L}^2(\Omega_T)} \le \|\mathbf{m} - \mathbf{m}^-_{hk}\|_{\mathbf{L}^2(\Omega_T)} + \||\mathbf{m}^-_{hk}| - 1\|_{\mathbf{L}^2(\Omega_T)} \to 0.$$

Dies impliziert weiters $|\mathbf{m}| = 1$ fast überall. □

Wenn wir nun in weiterer Folge von Konvergenz sprechen, dann sei damit die Konvergenz der Teilfolge aus dem vorangehenden Lemma gemeint.

Lemma 5.2.4. *Für den Grenzwert* $\mathbf{m}$ *aus dem vorangegangenen Lemma gilt* $\mathbf{m}(0) = \mathbf{m}^0$ *im Spursinn.*

Beweis. Nach Voraussetzung konvergiert $\mathbf{m}^0_h \rightharpoonup \mathbf{m}^0$ in $\mathbf{H}^1(\Omega)$. Außerdem konvergiert mittels $\mathbf{m}_{hk} \xrightarrow{\text{sub}} \mathbf{m}$ in $\mathbf{H}^1(\Omega_T)$ auch die Spur $\gamma_0\mathbf{m}_{hk} \xrightarrow{\text{sub}} \gamma_0\mathbf{m}$ in $\mathbf{L}^2(\Omega)$ wegen der Stetigkeit des Spuroperators. Da aber wegen der Zeitstetigkeit von $\mathbf{m}_{hk}$ auch $\gamma_0\mathbf{m}_{hk} = \mathbf{m}_{hk}(0) = \mathbf{m}^0_h$ gilt, erhalten wir die Konvergenzaussage

$$\mathbf{m}^0_h \rightharpoonup \gamma_0\mathbf{m} = \mathbf{m}(0) \quad \text{in } \mathbf{L}^2(\Omega). \tag{5.14}$$

Nun impliziert die Voraussetzung $\mathbf{m}^0_h \rightharpoonup \mathbf{m}^0$ in $\mathbf{H}^1(\Omega)$ aber insbesondere

$$\mathbf{m}^0_h \rightharpoonup \mathbf{m}^0 \quad \text{in } \mathbf{L}^2(\Omega). \tag{5.15}$$

Aus (5.14) und (5.15) erhalten wir durch die Eindeutigkeit des Grenzwertes das gewünschte Resultat $\mathbf{m}(0) = \mathbf{m}^0$. □

Nun haben wir für die Funktion $\mathbf{m}$ Bedingung 1 aus der Definition des schwachen Lösungsbegriffs 2.2.7 gezeigt. Folgendes Lemma dient als Basis um die Bedingung 2 zu zeigen.

Lemma 5.2.5. *Sei für ein beliebiges $\boldsymbol{\phi} \in C^\infty(\Omega_T)$ die Funktion $\boldsymbol{\phi}_h$ gewählt als $\boldsymbol{\phi}_h(t) := \mathcal{I}_h(\boldsymbol{\phi}(t,.))$, dann gilt*

$$\begin{aligned}\int_0^T (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h dt = \alpha \int_0^T (\mathbf{m}_{hk}^- \times \partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h dt \\ - \int_0^T (\overline{\mathbf{m}}_{hk} \times (C_{\text{ex}} \tilde{\Delta}_h \overline{\mathbf{m}}_{hk} + \widetilde{\Pi}_h \mathbf{m}_{hk}^- + \overline{\mathbf{f}}_{hk}), \boldsymbol{\phi}_h)_h dt.\end{aligned}$$

Für $t \in [t_j, t_{j+1})$ definieren wir dabei $\overline{\mathbf{f}}_{hk}(t) := \tilde{\mathbf{f}}_h^{j+1/2}$.

Beweis. Für alle Zeitpunkte t ist $\boldsymbol{\phi}_h(t)$ in $S^1(\mathcal{T}_h)$. Damit gilt nach Algorithmus 4.3.2 für alle $j = 0, ..., N-1$

$$(d_t \mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h = \alpha(\mathbf{m}_h^j \times d_t \mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h - (\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, \boldsymbol{\phi}_h)_h. \tag{5.16}$$

Indem wir Gleichung (5.16) aufintegrieren erhalten wir die gewünschte Gleichheit. □

Wir zeigen nun für die Funktion $\mathbf{m}$ Bedingung 2 aus der Definition des schwachen Lösungsbegriffs 2.2.7.

Lemma 5.2.6. *Die Funktion $\mathbf{m}$ erfüllt*

$$\begin{aligned}\int_{\Omega_T} \langle \mathbf{m}_t, \boldsymbol{\phi} \rangle d\mathbf{x}dt = \alpha \int_{\Omega_T} \langle \mathbf{m} \times \mathbf{m}_t, \boldsymbol{\phi} \rangle d\mathbf{x}dt + C_{\text{ex}} \int_{\Omega_T} \langle \mathbf{m} \times \nabla \mathbf{m}, \nabla \boldsymbol{\phi} \rangle d\mathbf{x}dt \\ - \int_{\Omega_T} \langle \mathbf{m} \times \Pi \mathbf{m}, \boldsymbol{\phi} \rangle d\mathbf{x}dt \\ - \int_{\Omega_T} \langle \mathbf{m} \times \mathbf{f}, \boldsymbol{\phi} \rangle d\mathbf{x}dt\end{aligned}$$

für alle Testfunktionen $\boldsymbol{\phi} \in C^\infty(\overline{\Omega}_T; \mathbb{R}^3)$, wobei wir hier das Kreuzprodukt $\mathbf{m} \times \nabla \mathbf{m}$ spaltenweise verstehen.

Beweis. Wir führen diesen Beweis in mehreren Schritten, indem wir für die einzelnen Terme der diskreten Version aus Lemma 5.2.5 die Konvergenz beweisen. Zu beachten ist, dass falls wir hier von Konvergenz sprechen, immer die Konvergenz der entsprechenden Teilfolge aus Lemma 5.2.3 gemeint ist. Unter der Verwendung von $\boldsymbol{\phi}_h(t) := \mathcal{I}_h(\boldsymbol{\phi}(t,.))$ zeigen wir konkret

$$\int_0^T (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h dt \to (\mathbf{m}_t, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}, \tag{5.17}$$

$$\int_0^T (\mathbf{m}_{hk}^- \times \partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h dt \to (\mathbf{m} \times \mathbf{m}_t, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}, \tag{5.18}$$

$$\int_0^T (\overline{\mathbf{m}}_{hk} \times \tilde{\Delta}_h \overline{\mathbf{m}}_{hk}, \boldsymbol{\phi}_h)_h dt \to -(\mathbf{m} \times \nabla \mathbf{m}, \nabla \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}, \tag{5.19}$$

$$\int_0^T (\overline{\mathbf{m}}_{hk} \times \widetilde{\Pi}_h \mathbf{m}_{hk}^-, \boldsymbol{\phi}_h)_h dt \to (\mathbf{m} \times \Pi \mathbf{m}, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}, \tag{5.20}$$

$$\int_0^T (\overline{\mathbf{m}}_{hk} \times \overline{\mathbf{f}}_{hk}, \boldsymbol{\phi}_h)_h dt \to (\mathbf{m} \times \mathbf{f}, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}. \tag{5.21}$$

1. Teil: Wir beginnen also mit der Konvergenz aus (5.17). Dazu verwenden wir den Satz von Fubini

$$|\int_0^T (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h dt - (\mathbf{m}_t, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}| = |\int_0^T (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h - (\mathbf{m}_t, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega)} dt|$$

und wenden dann die Dreiecksungleichung an

$$\begin{aligned}
&|\int_0^T (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h dt - (\mathbf{m}_t, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}| \\
&\qquad \leq |\int_0^T (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h - (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_{\mathbf{L}^2(\Omega)} dt| + |\int_0^T (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_{\mathbf{L}^2(\Omega)} - (\mathbf{m}_t, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega)} dt|.
\end{aligned} \tag{5.22}$$

Den Integranden des ersten Summanden schätzen wir mit Satz A.5.2 ab und erhalten

$$|\int_0^T (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h - (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_{\mathbf{L}^2(\Omega)}| \leq \int_0^T Ch \|\partial_t \mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega)} \|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega)} dt.$$

Dass $\|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega)}$ beschränkt bleibt, sieht man mittels Approximationssatz A.5.1 durch Anwendung der Dreiecksungleichung

$$\|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega)} \leq \|\nabla \boldsymbol{\phi}_h - \nabla \boldsymbol{\phi}\|_{\mathbf{L}^2(\Omega)} + \|\nabla \boldsymbol{\phi}\|_{\mathbf{L}^2(\Omega)}.$$

Hier geht $\|\nabla \boldsymbol{\phi}_h - \nabla \boldsymbol{\phi}\|_{\mathbf{L}^2(\Omega)}$ wegen Ungleichung A.5 gegen Null, und $\|\nabla \boldsymbol{\phi}\|_{\mathbf{L}^2(\Omega)}$ ist beschränkt wegen $\boldsymbol{\phi} \in C^\infty(\overline{\Omega}_T)$. Also gilt

$$\begin{aligned}
|\int_0^T (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h dt - (\partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_{\mathbf{L}^2(\Omega_T)}| &\leq \tilde{C}h \int_0^T \|\partial_t \mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega)} dt \\
&\leq \tilde{C}h|T|^{1/2} \|\partial_t \mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega_T)} \to 0.
\end{aligned}$$

Hierbei folgt die zweite Abschätzung aus der Hölderschen Ungleichung A.6.3, indem man die Indikatorfunktion $\mathbb{1}_T(t)$ in das Integral einschiebt. Hier ist $\|\partial_t \mathbf{m}_{hk}(t)\|_{\mathbf{L}^2(\Omega_T)}$ beschränkt, wegen der Konvergenz $\mathbf{m}_{hk} \rightharpoonup \mathbf{m}$ in $\mathbf{H}^1(\Omega_T)$. Wobei wir auch hier unter Konvergenz die Konvergenz der Teilfolge aus Lemma 5.2.3 verstehen.

Für den zweiten Summanden von (5.22) bemerken wir, dass $\partial_t \mathbf{m}_{hk} \rightharpoonup \mathbf{m}_t$ in $\mathbf{L}^2(\Omega_T)$ und wegen dem Approximationssatz $\boldsymbol{\phi}_h \to \boldsymbol{\phi}$ in $\mathbf{L}^2(\Omega_T)$ und wenden Lemma A.4.4 an. Damit gilt (5.17).

2. Teil: Für (5.18) beginnen wir mit

$$\begin{aligned}
&|\int_0^T (\mathbf{m}_{hk}^- \times \partial_t \mathbf{m}_{hk}, \boldsymbol{\phi}_h)_h - (\mathbf{m} \times \mathbf{m}_t, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega)} dt| \\
&\qquad \leq |\int_0^T (\partial_t \mathbf{m}_{hk}, \mathcal{I}_h(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h))_h - (\partial_t \mathbf{m}_{hk}, \mathcal{I}_h(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h))_{\mathbf{L}^2(\Omega)} dt| \\
&\qquad + |\int_0^T (\partial_t \mathbf{m}_{hk}, \mathcal{I}_h(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h))_{\mathbf{L}^2(\Omega)} - (\partial_t \mathbf{m}_{hk}, \mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)_{\mathbf{L}^2(\Omega)} dt| \\
&\qquad + |\int_0^T (\partial_t \mathbf{m}_{hk}, \mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)_{\mathbf{L}^2(\Omega)} - (\mathbf{m}_t, \mathbf{m} \times \boldsymbol{\phi})_{\mathbf{L}^2(\Omega)} dt|.
\end{aligned}$$

Den ersten Summanden können wir mittels Satz A.5.2 durch den Ausdruck

$$\begin{aligned}|\int_0^T (\partial_t \mathbf{m}_{hk}, \mathcal{I}_h(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h))_h - (\partial_t \mathbf{m}_{hk}, \mathcal{I}_h(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h))_{\mathbf{L}^2(\Omega)} dt| \\ \leq \int_0^T C_{A.5.2} h \|\partial \mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega)} \|\nabla \mathcal{I}_h(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega)} dt\end{aligned}$$

abschätzen. Wir betrachten den Term $\|\nabla \mathcal{I}_h(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega)}$, schätzen ihn mittels Dreiecksungleichung ab und wenden darauf den Approximationssatz A.5.1 und Lemma A.3.4 an.

$$\begin{aligned}\|\nabla \mathcal{I}_h(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega)} &\leq \|\nabla(\mathcal{I}_h - \mathrm{Id})(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega)} + \|\nabla \mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega)} \\ &\leq C_{\mathrm{approx}} C_{\mathrm{form}} \|h D^2(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega)} + \|\nabla \mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega)} \\ &\leq h C_{\mathrm{approx}} C_{\mathrm{form}} C_{A.3.4} \|\nabla \mathbf{m}_{hk}^-\|_{\mathbf{L}^2} \|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^\infty} + \|\nabla \mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega)}.\end{aligned}$$

Mit der eben gezeigten Beschränktheit folgt nun die Konvergenz des ersten Summanden gegen Null.

Als Schranke für den zweiten Term erhalten wir mittels der Cauchy-Schwarzschen Ungleichung

$$\begin{aligned}|\int_0^T (\partial_t \mathbf{m}_{hk}, \mathcal{I}_h(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h))_{\mathbf{L}^2(\Omega)} - (\partial_t \mathbf{m}_{hk}, \mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)_{\mathbf{L}^2(\Omega)} dt| \\ \leq \int_0^T \|\partial \mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega)} \|(\mathrm{Id} - \mathcal{I}_h)(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega)} dt.\end{aligned}$$

Dabei ist $\|\partial \mathbf{m}_{hk}\|_{\mathbf{L}^2(\Omega)}$ beschränkt und $\|(\mathrm{Id} - \mathcal{I}_h)(\mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega)}$ konvergiert wegen Approximationssatz A.5.1 und Lemma A.3.4 wie beim ersten Term gegen Null. Für den dritten Term $|\int_0^T (\partial_t \mathbf{m}_{hk}, \mathbf{m}_{hk}^- \times \boldsymbol{\phi}_h)_{\mathbf{L}^2(\Omega)} - (\mathbf{m}_t, \mathbf{m} \times \boldsymbol{\phi})_{\mathbf{L}^2(\Omega)} dt|$ folgt die Konvergenz aus Korollar A.3.3 und Lemma A.4.4.

3. Teil: Für (5.19) beginnen wir mit folgenden Umformungsschritten aus der Definition des diskreten Laplace-Operators $\tilde{\Delta}_h$

$$\begin{aligned}&|(\overline{\mathbf{m}}_{hk} \times \tilde{\Delta}_h \overline{\mathbf{m}}_{hk}, \boldsymbol{\phi}_h)_h - (\mathbf{m} \times \nabla \boldsymbol{\phi}, \nabla \mathbf{m})_{\mathbf{L}^2(\Omega)}| \\ &\quad = |-(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h, \tilde{\Delta}_h \overline{\mathbf{m}}_{hk})_h - (\mathbf{m} \times \nabla \boldsymbol{\phi}, \nabla \mathbf{m})_{\mathbf{L}^2(\Omega)}| \\ &\quad = |-\big(\mathcal{I}_h(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \tilde{\Delta}_h \overline{\mathbf{m}}_{hk}\big)_h - (\mathbf{m} \times \nabla \boldsymbol{\phi}, \nabla \mathbf{m})_{\mathbf{L}^2(\Omega)}| \\ &\quad = |\big(\nabla \mathcal{I}_h(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \nabla \overline{\mathbf{m}}_{hk}\big)_{\mathbf{L}^2(\Omega)} - (\mathbf{m} \times \nabla \boldsymbol{\phi}, \nabla \mathbf{m})_{\mathbf{L}^2(\Omega)}| \\ &\quad = |\big(\nabla(\mathcal{I}_h - \mathrm{Id})(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \nabla \overline{\mathbf{m}}_{hk}\big)_{\mathbf{L}^2(\Omega)} \\ &\qquad + \big(\nabla(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \nabla \overline{\mathbf{m}}_{hk}\big)_{\mathbf{L}^2(\Omega)} - (\mathbf{m} \times \nabla \boldsymbol{\phi}, \nabla \mathbf{m})_{\mathbf{L}^2(\Omega)}|.\end{aligned}$$

Mittels Dreiecksungleichung erhalten wir also

$$\begin{aligned}&|(\overline{\mathbf{m}}_{hk} \times \tilde{\Delta}_h \overline{\mathbf{m}}_{hk}, \boldsymbol{\phi}_h)_h - (\mathbf{m} \times \nabla \boldsymbol{\phi}_h, \nabla \mathbf{m})_{\mathbf{L}^2(\Omega)}| \\ &\qquad \leq |(\nabla(\mathcal{I}_h - \mathrm{Id})(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \nabla \overline{\mathbf{m}}_{hk})_{\mathbf{L}^2(\Omega)}| \\ &\qquad\quad + |(\nabla(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \nabla \overline{\mathbf{m}}_{hk})_{\mathbf{L}^2(\Omega)} - (\mathbf{m} \times \nabla \boldsymbol{\phi}, \nabla \mathbf{m})_{\mathbf{L}^2(\Omega)}|. \quad (5.23)\end{aligned}$$

Die beiden Summanden betrachten wir nun wieder getrennt. Der erste lässt sich mit der Cauchy-Schwarzschen Ungleichung abschätzen durch

$$|(\nabla(\mathcal{I}_h - \mathrm{Id})(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \nabla \overline{\mathbf{m}}_{hk})| \leq \|\nabla(\mathrm{Id} - \mathcal{I}_h)(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega)} \|\nabla \overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^2(\Omega)}.$$

Wir wenden nun die zweite Aussage (A.5) des Approximationssatzes A.5.1 an.

$$|(\nabla(\mathcal{I}_h - \mathrm{Id})(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \nabla\overline{\mathbf{m}}_{hk})| \leq hC_{\mathrm{approx}}C_{\mathrm{form}}\|D^2(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega)}\|\nabla\overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^2(\Omega)}$$

Für den Term $\|D^2(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h)\|^2_{\mathbf{L}^2(\Omega)}$ können wir Lemma A.3.4 anwenden und erhalten die Abschätzung

$$|((\mathcal{I}_h - \mathrm{Id})(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \tilde{\Delta}_h\overline{\mathbf{m}}_{hk})_h| \leq hC_{\mathrm{approx}}C_{\mathrm{form}}C_{A.3.4}\|\nabla\overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^2(\Omega)}\|\nabla\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty(\Omega)}\|\nabla\overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^2(\Omega)}$$

Nun sind $\|\nabla\overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^2(\Omega)}$, $\|\nabla\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty(\Omega)}$ und $\|\nabla\overline{\mathbf{m}}_{hk}\|_{\mathbf{L}^2(\Omega)}$ gleichmäßig beschränkt, und damit konvergiert das Integral $|\int_0^T((\mathcal{I}_h - \mathrm{Id})(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \tilde{\Delta}_h\overline{\mathbf{m}}_{hk})_h dt|$ dieses ersten Summanden gegen Null.

Der zweite Term von Gleichung (5.23) ist schneller abgearbeitet. Es gilt nach Lemma A.1.2 die Gleichheit $(\nabla\boldsymbol{\xi}, \nabla(\boldsymbol{\xi}\times\boldsymbol{\eta})) = (\nabla\boldsymbol{\xi}, \boldsymbol{\xi}\times\nabla\boldsymbol{\eta})$ für alle $\boldsymbol{\xi}, \boldsymbol{\eta} \in \mathbf{H}^1(\Omega;\mathbb{R}^3)$, weswegen wir die Gleichheit

$$\begin{aligned}|(\nabla(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \nabla\overline{\mathbf{m}}_{hk})_{\mathbf{L}^2(\Omega_T)} &- (\mathbf{m} \times \nabla\boldsymbol{\phi}, \nabla\mathbf{m})_{\mathbf{L}^2(\Omega_T)}| \\ &= |(\overline{\mathbf{m}}_{hk} \times \nabla\boldsymbol{\phi}_h, \nabla\overline{\mathbf{m}}_{hk})_{\mathbf{L}^2(\Omega_T)} - (\mathbf{m} \times \nabla\boldsymbol{\phi}, \nabla\mathbf{m})_{\mathbf{L}^2(\Omega_T)}|\end{aligned}$$

erhalten und mit Lemma A.4.4 die Konvergenz folgt.

4. Teil: Nun behandeln wir die Konvergenz aus (5.20). In erster Instanz bemühen wir wieder die alternierende Eigenschaft des Spatproduktes.

$$\begin{aligned}|\int_0^T (\overline{\mathbf{m}}_{hk} \times \widetilde{\Pi}_h\mathbf{m}^-_{hk}, \boldsymbol{\phi}_h)_h dt &- (\mathbf{m} \times \Pi\mathbf{m}, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}| \\ &= |\int_0^T (\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h, \widetilde{\Pi}_h\mathbf{m}^-_{hk})_h dt - (\mathbf{m} \times \boldsymbol{\phi}, \Pi\mathbf{m})_{\mathbf{L}^2(\Omega_T)}|\end{aligned}$$

Wir beachten, dass das h-Skalarprodukt nur von den Knotenwerten abhängt und schieben den Term $\pm(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h, \Pi_h\mathbf{m}^-_{hk})_{\mathbf{L}^2(\Omega_T)}$ ein. Die Dreiecksungleichung liefert das Folgende.

$$\begin{aligned}|\int_0^T (\overline{\mathbf{m}}_{hk} \times \widetilde{\Pi}_h\mathbf{m}^-_{hk}, \boldsymbol{\phi}_h)_h dt &- (\mathbf{m} \times \Pi\mathbf{m}, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}| \\ &= |\int_0^T (\mathcal{I}_h(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \widetilde{\Pi}_h\mathbf{m}^-_{hk})_h dt - (\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h, \Pi_h\mathbf{m}^-_{hk})_{\mathbf{L}^2(\Omega_T)} \\ &\quad + (\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h, \Pi_h\mathbf{m}^-_{hk})_{\mathbf{L}^2(\Omega_T)} - (\mathbf{m} \times \boldsymbol{\phi}, \Pi\mathbf{m})_{\mathbf{L}^2(\Omega_T)}| \\ &\leq |((\mathcal{I}_h - \mathrm{Id})(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h), \Pi_h\mathbf{m}^-_{hk})_{\mathbf{L}^2(\Omega_T)}| \\ &+ |(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h, \Pi_h\mathbf{m}^-_{hk})_{\mathbf{L}^2(\Omega_T)} - (\mathbf{m} \times \boldsymbol{\phi}, \Pi\mathbf{m})_{\mathbf{L}^2(\Omega_T)}|.\end{aligned}$$

Mit der Cauchy-Schwarzschen Ungleichung erhalten wir

$$\begin{aligned}|\int_0^T (\overline{\mathbf{m}}_{hk} \times \widetilde{\Pi}_h\mathbf{m}^-_{hk}, \boldsymbol{\phi}_h)_h dt &- (\mathbf{m} \times \Pi\mathbf{m}, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}| \\ &\leq \|(\mathcal{I}_h - \mathrm{Id})(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega_T)}\|\Pi_h\mathbf{m}^-_{hk}\|_{\mathbf{L}^2(\Omega_T)} \\ &+ |(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h, \Pi_h\mathbf{m}^-_{hk})_{\mathbf{L}^2(\Omega_T)} - (\mathbf{m} \times \boldsymbol{\phi}_h, \Pi\mathbf{m})_{\mathbf{L}^2(\Omega_T)}|.\end{aligned}$$

Der erste Summand geht nun gegen Null, weil auf den ersten Faktor der Approximationssatz A.5.1 angewandt werden kann und $\|\Pi_h\mathbf{m}^j_h\|_{\mathbf{L}^2(\Omega)}$ beschränkt bleibt. Für die Konvergenz des zweiten Summanden wenden wir Lemma A.4.4 und Korollar A.3.3 an.

5. Teil: Gänzlich analog zum 4. Teil des Beweises erhalten wir die Ungleichung

$$\begin{aligned}|\int_0^T (\overline{\mathbf{m}}_{hk} \times \tilde{\mathbf{f}}_h, \boldsymbol{\phi}_h)_h dt - (\mathbf{m} \times \mathbf{f}, \boldsymbol{\phi})_{\mathbf{L}^2(\Omega_T)}| \\ \leq \|(\mathcal{I}_h - \mathrm{Id})(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h)\|_{\mathbf{L}^2(\Omega_T)} \|\mathbf{f}\|_{\mathbf{L}^2(\Omega_T)} \\ + |(\overline{\mathbf{m}}_{hk} \times \boldsymbol{\phi}_h, \mathbf{f})_{\mathbf{L}^2(\Omega_T)} - (\mathbf{m} \times \boldsymbol{\phi}_h, \mathbf{f})_{\mathbf{L}^2(\Omega_T)}|.\end{aligned}$$

Und mit der Beschränktheit von $\|\mathbf{f}\|_{\mathbf{L}^2(\Omega_T)}$ folgt wie oben die Konvergenz (5.21).

Die Gleichungen (5.17)–(5.21) kombinieren wir nun zum gewünschten Resultat. □

Nun fehlt uns nur noch ein Beweis zur Bedingung 3 des schwachen Lösungsbegriffs.

Lemma 5.2.7. *Die Funktion* $\mathbf{m}$ *erfüllt für fast alle* $t' > 0$

$$\frac{C_{\mathrm{ex}}}{2} \int_\Omega |\nabla \mathbf{m}(t')|^2 d\mathbf{x} + \alpha \int_{(0,t')\times\Omega} |\mathbf{m}_t|^2 d\mathbf{x} dt \leq C_{\mathrm{energy}},$$

für eine von t' *unabhängige Konstante* $C_{\mathrm{energy}} > 0$.

Beweis. Wir verwenden hier Lemma 5.2.1. Da wir bereits über die Beschränktheit Bescheid wissen, verstecken wir den ε-Term in folgender Manier auf der rechten Seite:

$$\frac{C_{\mathrm{ex}}}{2} \int_I \|\nabla \mathbf{m}_{hk}^+(t')\|^2_{\mathbf{L}^2(\Omega)} dt' + \alpha \int_I \int_0^{t'} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt\, dt' \leq \int_I C, \tag{5.24}$$

für beliebige messbare $I \subseteq [0,T]$. Wir wenden hier Lemma A.4.6 für die schwach in $\mathbf{L}^2(\Omega_T)$ konvergenten Folgen $\mathbf{m}_{hk}^+ \rightharpoonup \mathbf{m}$ sowie $\partial_t \mathbf{m}_{hk} \rightharpoonup \mathbf{m}_t$ und die konvexen Funktionen $f_1(x) := \|x\|^2_{\mathbf{L}^2(\Omega\times I)}$ und $f_2(x) := \int_I \int_0^{t'} \|x\|^2_{\mathbf{L}^2(\Omega)} dt\, dt'$ an. Man erhält

$$\int_I \|\nabla \mathbf{m}(t')\|^2_{\mathbf{L}^2(\Omega)} dt' \leq \liminf_{h,k} \|\nabla \mathbf{m}_{hk}^+\|^2_{\mathbf{L}^2(\Omega\times I)}.$$

Da wir wissen, dass die Normäquivalenzkonstante aus Lemma 3.2.7 gleich eins ist, können wir nach unten abschätzen $\|\partial_t \mathbf{m}_{hk}\|_h^2 \geq \|\partial_t \mathbf{m}_{hk}\|^2_{\mathbf{L}^2}$. Also gilt weiters mit Lemma A.4.6

$$\int_I \int_0^{t'} \|\mathbf{m}_t\|^2_{\mathbf{L}^2(\Omega)} dt\, dt' \leq \liminf_{h,k} \int_I \int_0^{t'} \|\partial_t \mathbf{m}_{hk}\|^2_{\mathbf{L}^2(\Omega)} dt\, dt' \leq \liminf_{h,k} \int_I \int_0^{t'} \|\partial_t \mathbf{m}_{hk}\|_h^2 dt\, dt'$$

Da die rechte Seite in (5.24) unabhängig von h, k ist, können wir mit diesen beiden Ungleichungen die Ungleichung (5.24) nach unten abschätzen durch

$$\frac{C_{\mathrm{ex}}}{2} \int_I \|\nabla \mathbf{m}(t')\|^2_{\mathbf{L}^2(\Omega)} dt' + \alpha \int_I \int_0^{t'} \|\mathbf{m}_t\|^2_{\mathbf{L}^2(\Omega)} dt\, dt' \leq \int_I C.$$

Unter Anwendung von Satz A.6.2 erhalten wir das gewünschte Resultat. □

Hauptsatz 5.2.8. *Sei* $\mathbf{m}^0 \in \mathbf{H}^1(\Omega; \mathbb{S}^2)$ *und die Familie* $\mathbf{m}_h^0 \in S^1(\mathcal{T}_h)$ *so gewählt, dass diese schwach in* $\mathbf{H}^1(\Omega; \mathbb{R}^3)$ *gegen* $\mathbf{m}^0$ *konvergiere. Weiters existieren für die Wahlen der* h *und* k *alle Funktionen* $(\mathbf{m}_h^j)_{j=1}^N$*, welche Algorithmus 4.3.2 genügen. Dann konvergiert jede schwach in* $\mathbf{H}^1(\Omega; \mathbb{R}^3)$ *konvergente Teilfolge der* $\mathbf{m}_{hk}$ *gegen eine schwache Lösung der LLG-Gleichung im Sinne von Definition 2.2.7.*

Man beachte, dass wir in diesem Abschnitt immer vorausgesetzt haben, dass wir bereits Lösungen des Verfahrens 4.3.2 in der Hand haben. Da diese allerdings nicht ohne weiteres vorliegen, werden wir im nächsten Abschnitt sehen, dass wir für die Existenz und Eindeutigkeit einer Lösung des Zeitschrittverfahrens noch die Bedingung $k = o(h^2)$ einführen müssen.

5.3. Fixpunktiteration zur Lösung der Diskretisierung

In diesem Abschnitt werden wir näher darauf eingehen, wie wir die Folge der $\mathbf{m}_h^j$ aus Algorithmus 4.3.2 erhalten. Bis jetzt hatten wir diese Teillösungen nicht konkret in der Hand, sondern wussten nur, falls wir Funktionen fänden, die der gewünschten Eigenschaft genügen, konvergieren diese gegen eine schwache Lösung. Wir werden zuerst die in [BP06] dargestellte Iteration herleiten und dann deren bedingte Konvergenz zeigen.

5.3.1. Iteration

Herleitung 5.3.1. *Wir formen die Bedingung aus Algorithmus 4.3.2 um.*

$$\begin{aligned} 0 = (d_t\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h - \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h + C_{\mathrm{ex}}(\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\Delta}_h\overline{\mathbf{m}}_h^{j+1/2}, \boldsymbol{\phi}_h)_h \\ + (\overline{\mathbf{m}}_h^{j+1/2} \times \widetilde{\Pi}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h \\ + (\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{f}}_h^{j+1/2}, \boldsymbol{\phi}_h)_h. \end{aligned}$$

Und erhalten durch Anwenden der Definitionen

$$\begin{aligned} 0 = \frac{1}{k}(\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h - \frac{1}{k}(\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{\alpha}{k}(\mathbf{m}_h^j \times \mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h + \frac{\alpha}{k}(\mathbf{m}_h^j \times \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h \\ + \frac{C_{\mathrm{ex}}}{4}\Big((\mathbf{m}_h^{j+1} \times \tilde{\Delta}_h\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h + (\mathbf{m}_h^j \times \tilde{\Delta}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h \\ + (\mathbf{m}_h^{j+1} \times \tilde{\Delta}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h + (\mathbf{m}_h^j \times \tilde{\Delta}_h\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h\Big) \\ + \frac{1}{2}\Big((\mathbf{m}_h^{j+1} \times (\widetilde{\Pi}_h\mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h + (\mathbf{m}_h^j \times (\widetilde{\Pi}_h\mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h\Big). \end{aligned}$$

Hier fällt der Term $(\mathbf{m}_h^j \times \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h$ weg, und wir erhalten die äquivalente Gleichung

$$\begin{aligned} \frac{1}{k}(\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{1}{2}(\mathbf{m}_h^j \times (\widetilde{\Pi}_h\mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h \\ = \frac{1}{k}(\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h - \frac{\alpha}{k}(\mathbf{m}_h^j \times \mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h + \frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^{j+1} \times \tilde{\Delta}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h \\ + \frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h + \frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^{j+1} \times \tilde{\Delta}_h\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h \\ + \frac{1}{2}(\mathbf{m}_h^{j+1} \times (\widetilde{\Pi}_h\mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h \end{aligned} \tag{5.25}$$

Wenn wir nun die obige Gleichung unter dem Gesichtspunkt eines bekannten $\mathbf{m}_h^j$ und eines gesuchten $\mathbf{m}_h^{j+1}$ betrachten, dann fällt folgendes auf: Die linke Seite ist konstant bezüglich $\mathbf{m}_h^{j+1}$ und die rechte Seite nahezu linear. Lediglich der vorletzte Term $(\mathbf{m}_h^{j+1} \times \tilde{\Delta}_h\mathbf{m}_h^{j+1}, \boldsymbol{\phi}_h)_h$ ist nichtlinear. Wie auch in [BP06] wird im nächsten Algorithmus eine durch Linearisierung dieses Terms gewonnene implizite Fixpunktiteration vorgestellt. Wir werden anschließend zeigen, dass diese unter gewissen Voraussetzungen Lösungen für Algorithmus 4.3.2 generiert. Dies wird dann insbesondere zeigen, dass (5.25) *bzw.* (4.6) *eine eindeutige Lösung besitzt, sodass Algorithmus 4.3.2 wohldefiniert ist, vgl. Satz 5.3.6*

Algorithmus 5.3.2. *Setze $\mathbf{m}_h^{j+1,0} := \mathbf{m}_h^j$ und $\ell := 1$*

(i) Berechne $\mathbf{m}_h^{j+1,\ell} \in S^1(\mathcal{T}_h)$*, sodass für alle* $\boldsymbol{\phi}_h \in S^1(\mathcal{T}_h)$ *folgende Gleichung gilt*

$$\begin{aligned}
&\frac{1}{k}(\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{1}{2}(\mathbf{m}_h^j \times (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h \\
&\qquad = \frac{1}{k}(\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h - \frac{\alpha}{k}(\mathbf{m}_h^j \times \mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h \\
&\qquad + \frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h + \frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h \\
&\qquad + \frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell-1}, \boldsymbol{\phi}_h)_h \\
&\qquad + \frac{1}{2}(\mathbf{m}_h^{j+1,\ell} \times (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h.
\end{aligned} \tag{5.26}$$

(ii) Falls $\|\mathbf{m}_h^{j+1,\ell} - \mathbf{m}_h^{j+1,\ell-1}\|_h \leq \varepsilon$ *dann setze* $\mathbf{m}_h^{j+1} := \mathbf{m}_h^{j+1,\ell}$.

(iii) Ansonsten setze $\ell := \ell + 1$ *und gehe zu* (i).

Vorerst bleibt natürlich offen inwiefern dieses Verfahren eindeutig definiert und korrekt ist. Die nächsten Lemmata werden uns die bedingte Konvergenz gegen eine Funktion $\mathbf{m}_h^{j+1,\infty}$ aus $S^1(\mathcal{T}_h)$ liefern – die Wohldefiniertheit des Algorithmus. Anschließend wird gezeigt, dass diese Grenzfunktion genau die gesuchte Magnetisierung $\mathbf{m}_h^{j+1}$ aus Algorithmus 4.3.2 ist.

Wir beginnen damit, dass falls die Zeitschrittweite k klein genug gewählt wird, Schritt (i) aus Algorithmus 5.3.2 wohldefiniert ist.

5.3.2. Wohldefiniertheit

Im folgenden Abschnitt werden wir die Fixpunktiteration genauer betrachten und ein Kriterium dafür finden, wann diese konvergiert. Dies wird uns $k = o(h^2)$ als hinreichende Bedingung an die Zeit- und Ortsschrittweite liefern. Die Konstanten, die diese Bedingungen widerspiegeln werden wir hier voranstellen.

Definition 5.3.3. *Für feste* h, k *definieren wir die Konstante* γ*, die wir für die Wohldefiniertheit der Iteration benötigen durch*

$$\gamma := \frac{C_{\mathrm{ex}}}{4} C_{A.5.4}^2 h^{-2} k. \tag{5.27}$$

Weiters definieren wir die Konstante Θ*, die wir für die Kontraktionseigenschaft und somit Termination der Fixpunktiteration benötigen durch*

$$\Theta := \frac{1 + C_{\mathrm{ex}} C_{4.2.3} k h^{-2}/4 + C_{4.2.4} k h^{-3/2}}{1 - C_{\mathrm{ex}} C_{4.2.3} k h^{-2}/4}. \tag{5.28}$$

Lemma 5.3.4. *Für* $\gamma < 1$ *und gegebenes* $\mathbf{m}_h^{j+1,\ell-1} \in S^1(\mathcal{T}_h)$ *hat Gleichung* (5.26) *eine eindeutige Lösung* $\mathbf{m}_h^{j+1,\ell}$*. In diesem Fall bezeichnen wir die Abbildung, die uns zu* $\mathbf{m}_h^{j+1,\ell-1}$ *die Funktion* $\mathbf{m}_h^{j+1,\ell}$ *liefert mit* $K : S^1(\mathcal{T}_h) \to S^1(\mathcal{T}_h)$*.*

Beweis. Wir zeigen, dass Gleichung (5.26) eine eindeutige Lösung für $\mathbf{m}_h^{j+1,\ell}$ hat. Dazu verwenden wir das Lemma von Lax-Milgram A.4.7. Wir definieren nun schrittweise für $\ell = 1$ die

Bilinearform $B_\ell(\boldsymbol{\psi}_h, \boldsymbol{\phi}_h)$ auf $S^1(\mathcal{T}_h) \times S^1(\mathcal{T}_h)$ durch

$$\begin{aligned} B_\ell(\boldsymbol{\psi}_h, \boldsymbol{\phi}_h) := \frac{1}{k}(\boldsymbol{\psi}_h, \boldsymbol{\phi}_h)_h - \frac{\alpha}{k}(\mathbf{m}_h^j \times \boldsymbol{\psi}_h, \boldsymbol{\phi}_h)_h \\ + \frac{C_{\text{ex}}}{4}(\boldsymbol{\psi}_h \times \tilde{\Delta}_h \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h + \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \boldsymbol{\psi}_h, \boldsymbol{\phi}_h)_h \\ + \frac{C_{\text{ex}}}{4}(\boldsymbol{\psi}_h \times \tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell-1}, \boldsymbol{\phi}_h)_h \\ + \frac{1}{2}(\boldsymbol{\psi}_h \times (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h. \end{aligned} \tag{5.29}$$

Mit $L(\boldsymbol{\phi}_h) := \frac{1}{k}(\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{1}{2}(\mathbf{m}_h^j \times (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h$ stellt dann

$$B_\ell(\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h) = L(\boldsymbol{\phi}_h)$$

genau die Gleichung (5.26) dar. Für ein beliebiges $\boldsymbol{\phi}_h \in S^1(\mathcal{T}_h)$ ergibt sich

$$\begin{aligned} B_\ell(\boldsymbol{\phi}_h, \boldsymbol{\phi}_h) &= \frac{1}{k}(\boldsymbol{\phi}_h, \boldsymbol{\phi}_h)_h - \frac{\alpha}{k}(\mathbf{m}_h^j \times \boldsymbol{\phi}_h, \boldsymbol{\phi}_h)_h \\ &\quad + \frac{C_{\text{ex}}}{4}(\boldsymbol{\phi}_h \times \tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell-1}, \boldsymbol{\phi}_h)_h \\ &\quad + \frac{C_{\text{ex}}}{4}(\boldsymbol{\phi}_h \times \tilde{\Delta}_h \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h + \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \boldsymbol{\phi}_h, \boldsymbol{\phi}_h)_h \\ &\quad + \frac{1}{2}(\boldsymbol{\phi}_h \times (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h \\ &= \frac{1}{k}\|\boldsymbol{\phi}_h\|_h^2 + \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \boldsymbol{\phi}_h, \boldsymbol{\phi}_h)_h. \end{aligned}$$

Mit der Cauchy-Schwarzschen Ungleichung erhalten wir daraus

$$B_\ell(\boldsymbol{\phi}_h, \boldsymbol{\phi}_h) \geq \frac{1}{k}\|\boldsymbol{\phi}_h\|_h^2 - \frac{C_{\text{ex}}}{4}\|\mathbf{m}_h^j \times \tilde{\Delta}_h \boldsymbol{\phi}_h\|_h \|\boldsymbol{\phi}_h\|_h.$$

Im nächsten Schritt wenden wir die Abschätzung aus Lemma A.3.1 an.

$$B_\ell(\boldsymbol{\phi}_h, \boldsymbol{\phi}_h) \geq \frac{1}{k}\|\boldsymbol{\phi}_h\|_h^2 - \frac{C_{\text{ex}}}{4}\|\mathbf{m}_h^j\|_{\mathbf{L}^\infty} \|\tilde{\Delta}_h \boldsymbol{\phi}_h\|_h \|\boldsymbol{\phi}_h\|_h.$$

Der Laplace-Term lässt sich mithilfe von Korollar 4.4 abschätzen.

$$\begin{aligned} B_\ell(\boldsymbol{\phi}_h, \boldsymbol{\phi}_h) &\geq \frac{1}{k}\|\boldsymbol{\phi}_h\|_h^2 - \frac{C_{\text{ex}}}{4} C_{A.5.4}^2 h^{-2} \|\mathbf{m}_h^j\|_{\mathbf{L}^\infty} \|\boldsymbol{\phi}_h\|_h \|\boldsymbol{\phi}_h\|_h \\ &\geq \|\boldsymbol{\phi}_h\|_h^2 \big(\frac{1}{k} - \frac{C_{\text{ex}}}{4} C_{A.5.4}^2 h^{-2}\big). \end{aligned}$$

Falls nun der Klammernausdruck positiv ist – das gilt, falls $\gamma := \frac{C_{\text{ex}}}{4} C_{A.5.4}^2 h^{-2} k < 1$ ist – haben wir eine koerzitive Bilinearform. Diese ist aufgrund der endlichen Dimension des Raumes $S^1(\mathcal{T}_h)$ stetig. Das Lemma von Lax-Milgram A.4.7 sichert uns deshalb eine eindeutige Lösung $\mathbf{m}_h^{j+1,\ell} \in S^1(\mathcal{T}_h)$ der Gleichung

$$B_\ell(\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h) = L(\boldsymbol{\phi}_h) \quad \text{ für alle } \boldsymbol{\phi}_h \in S^1(\mathcal{T}_h).$$

Das ist aber genau Gleichung (5.26). Für gegebenes $\mathbf{m}_h^{j+1,\ell-1}$ lässt sich also Gleichung (5.26), eindeutig nach $\mathbf{m}_h^{j+1,\ell}$ lösen. □

Wir wissen nun, in welchem Fall wir die Berechnungen sinnvoll durchführen können. Um zu sehen, dass das Verfahren terminiert, zeigen wir das nächste Lemma.

Lemma 5.3.5. *Sei $\gamma < 1$ für die Konstante aus Definition 5.3.3. Dann gilt*

$$\|\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}\|_h \leq \Theta \frac{\gamma}{1-\gamma} \|\mathbf{m}_h^{j+1,\ell} - \mathbf{m}_h^{j+1,\ell-1}\|_h.$$

Beweis. Wir wählen ein $\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)$, sodass gilt $|\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| = \|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty}$. Nun setzen wir die Testfunktion $\boldsymbol{\phi}_h := \varphi_{\mathbf{x}_h} \mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)$, mit $\varphi_{\mathbf{x}_h}$ der nodalen Basisfunktion aus Definition 3.2.2, in Gleichung (5.26) ein. Hierbei fallen die Terme $(\mathbf{m}_h^{j+1,\ell} \times \ldots, \varphi_{\mathbf{x}_h} \mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h))_h$ weg. Es ergibt sich

$$\begin{aligned}\frac{1}{k}(\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{1}{2}(\mathbf{m}_h^j \times (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h \\ = \frac{1}{k}(\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h + \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h.\end{aligned}$$

Nun muss noch geeignet umgeformt werden. Wir vereinfachen nach Bemerkung 3.2.6. Auf beiden Seiten lässt sich durch $\beta_{\mathbf{x}_h}$ kürzen.

$$\begin{aligned}\frac{1}{k}|\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)|^2 &= \frac{1}{k}\langle \mathbf{m}_h^j(\mathbf{x}_h), \mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)\rangle \\ &\quad - \frac{C_{\text{ex}}}{4}\langle \mathbf{m}_h^j(\mathbf{x}_h) \times \tilde{\Delta}_h \mathbf{m}_h^j(\mathbf{x}_h), \mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)\rangle \\ &\quad - \frac{C_{\text{ex}}}{4}\langle \mathbf{m}_h^j(\mathbf{x}_h) \times \tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h), \mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)\rangle \\ &\quad - \frac{1}{2}\langle \mathbf{m}_h^j(\mathbf{x}_h) \times (\widetilde{\Pi}_h \mathbf{m}_h^j(\mathbf{x}_h) + \tilde{\mathbf{f}}_h^{j+1/2}(\mathbf{x}_h)), \mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)\rangle.\end{aligned}$$

Wir wenden die Cauchy-Schwarzsche Ungleichung an.

$$\begin{aligned}\frac{1}{k}|\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)|^2 &\leq \frac{1}{k}|\mathbf{m}_h^j(\mathbf{x}_h)||\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \\ &\quad + \frac{C_{\text{ex}}}{4}|\mathbf{m}_h^j(\mathbf{x}_h)||\tilde{\Delta}_h \mathbf{m}_h^j(\mathbf{x}_h)||\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \\ &\quad + \frac{C_{\text{ex}}}{4}|\mathbf{m}_h^j(\mathbf{x}_h)||\tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)||\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \\ &\quad + \frac{1}{2}|\mathbf{m}_h^j(\mathbf{x}_h)||\widetilde{\Pi}_h \mathbf{m}_h^j(\mathbf{x}_h)||\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \\ &\quad + \frac{1}{2}|\mathbf{m}_h^j(\mathbf{x}_h)||\tilde{\mathbf{f}}_h^{j+1/2}(\mathbf{x}_h)||\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)|.\end{aligned}$$

Der Ausdruck lässt sich mittels $|\mathbf{m}_h^j(\mathbf{x}_h)| = 1$ vereinfachen.

$$\begin{aligned}\frac{1}{k}|\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)|^2 &\leq \frac{1}{k}|\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \\ &\quad + \frac{C_{\text{ex}}}{4}|\tilde{\Delta}_h \mathbf{m}_h^j(\mathbf{x}_h)||\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \\ &\quad + \frac{C_{\text{ex}}}{4}|\tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)||\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \\ &\quad + \frac{1}{2}|\widetilde{\Pi}_h \mathbf{m}_h^j(\mathbf{x}_h)||\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \\ &\quad + \frac{1}{2}|\tilde{\mathbf{f}}_h^{j+1/2}(\mathbf{x}_h)||\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)|.\end{aligned}$$

Division durch $|\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)|$ und Anwenden von $|\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| = \|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty}$ liefert

$$\begin{aligned}\frac{1}{k}\|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty} \leq \frac{1}{k} + \frac{C_{\text{ex}}}{4}|\tilde{\Delta}_h\mathbf{m}_h^j(\mathbf{x}_h)| + \frac{C_{\text{ex}}}{4}|\tilde{\Delta}_h\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \\ + \frac{1}{2}|\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h)| + \frac{1}{2}|\tilde{\mathbf{f}}_h^{j+1/2}(\mathbf{x}_h)|.\end{aligned} \tag{5.30}$$

Mit Lemma 4.2.3 angewandt auf die beiden Laplace-Terme gelten die Ungleichungen

$$\begin{aligned}|\tilde{\Delta}_h\mathbf{m}_h^{j+1,\ell}(\mathbf{x}_h)| \leq C_{4.2.3}h^{-2}\|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty}, \\ |\tilde{\Delta}_h\mathbf{m}_h^{j}(\mathbf{x}_h)| \leq C_{4.2.3}h^{-2}\|\mathbf{m}_h^{j}\|_{\mathbf{L}^\infty}.\end{aligned}$$

Wegen Korollar 4.2.4 erhalten wir auch die Abschätzungen

$$\begin{aligned}|\widetilde{\Pi}_h\mathbf{m}_h^j(\mathbf{x}_h)| \leq C_{4.2.4}h^{-3/2}, \\ |\tilde{\mathbf{f}}_h^{j+1/2}(\mathbf{x}_h)| \leq C_{4.2.4}h^{-3/2}.\end{aligned}$$

Diese Abschätzungen können wir nun auf (5.30) anwenden und erhalten

$$\begin{aligned}\frac{1}{k}\|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty} \leq \frac{1}{k} + \frac{C_{\text{ex}}}{4}C_{4.2.3}h^{-2}\|\mathbf{m}_h^{j}\|_{\mathbf{L}^\infty} + \frac{C_{\text{ex}}}{4}C_{4.2.3}h^{-2}\|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty} \\ + \frac{1}{2}C_{4.2.4}h^{-3/2} + \frac{1}{2}C_{4.2.4}h^{-3/2}.\end{aligned}$$

Woraus man mittels Umformen und $\|\mathbf{m}_h^j\|_{\mathbf{L}^\infty} = 1$ die Ungleichung

$$\|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty} \leq \frac{1 + C_{\text{ex}}C_{4.2.3}kh^{-2}/4 + C_{4.2.4}kh^{-3/2}}{1 - C_{\text{ex}}C_{4.2.3}kh^{-2}/4} = \Theta \tag{5.31}$$

gewinnen kann. Nun betrachten wir die Differenz zweier aufeinanderfolgender Gleichungen der Iteration.

$$\begin{aligned}0 = \frac{1}{k}(\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h - \frac{\alpha}{k}(\mathbf{m}_h^j \times (\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}), \boldsymbol{\phi}_h)_h \\ + \frac{C_{\text{ex}}}{4}((\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}) \times \tilde{\Delta}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h + \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^{j+1,\ell+1} \times \tilde{\Delta}_h\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h \\ - \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h\mathbf{m}_h^{j+1,\ell-1}, \boldsymbol{\phi}_h)_h + \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h(\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}), \boldsymbol{\phi}_h)_h \\ + \frac{1}{2}((\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}) \times (\widetilde{\Pi}_h\mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h.\end{aligned}$$

Wir addieren den Term $\frac{C_{\text{ex}}}{4}(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h - \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h = 0$ und fassen geeignet zusammen.

$$\begin{aligned}0 = \frac{1}{k}\big(\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h\big)_h - \frac{\alpha}{k}\big(\mathbf{m}_h^j \times (\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}), \boldsymbol{\phi}_h\big)_h \\ + \frac{C_{\text{ex}}}{4}\big((\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}) \times \tilde{\Delta}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h\big)_h + \frac{C_{\text{ex}}}{4}\big((\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}) \times \tilde{\Delta}_h\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h\big)_h \\ + \frac{C_{\text{ex}}}{4}\big(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h(\mathbf{m}_h^{j+1,\ell} - \mathbf{m}_h^{j+1,\ell-1}), \boldsymbol{\phi}_h\big)_h + \frac{C_{\text{ex}}}{4}\big(\mathbf{m}_h^j \times \tilde{\Delta}_h(\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}), \boldsymbol{\phi}_h\big)_h \\ + \frac{1}{2}((\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}) \times (\widetilde{\Pi}_h\mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h.\end{aligned}$$

Wir wählen $\boldsymbol{\phi}_h := \mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}$. Damit sind der zweite, dritte, vierte und siebte Term von der Form $(a \times b, b)_h$ und fallen weg. Es bleibt

$$\begin{aligned}
&\frac{1}{k}\big(\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}, \mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}\big)_h \\
&\quad = -\frac{C_{\mathrm{ex}}}{4}\big(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h(\mathbf{m}_h^{j+1,\ell} - \mathbf{m}_h^{j+1,\ell-1}), \mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}\big)_h \\
&\quad - \frac{C_{\mathrm{ex}}}{4}\big(\mathbf{m}_h^{j} \times \tilde{\Delta}_h(\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}), \mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}\big)_h.
\end{aligned}$$

Nun wenden wir die Cauchy-Schwarzsche Ungleichung und das Lemma A.3.1 an und dividieren dann durch $\|\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}\|_h$.

$$\begin{aligned}
\frac{1}{k}\|\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}\|_h \leq{}& \frac{C_{\mathrm{ex}}}{4}\|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty}\|\tilde{\Delta}_h(\mathbf{m}_h^{j+1,\ell} - \mathbf{m}_h^{j+1,\ell-1})\|_h \\
&+ \frac{C_{\mathrm{ex}}}{4}\|\mathbf{m}_h^{j}\|_{\mathbf{L}^\infty}\|\tilde{\Delta}_h(\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell})\|_h.
\end{aligned}$$

Auf $\|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty}$ wenden wir (5.31) und auf die beiden Laplace-Terme Korollar 4.4 an. Außerdem gilt $\|\mathbf{m}_h^{j}\|_{L^\infty} = 1$. Das liefert uns

$$\begin{aligned}
\|\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}\|_h \leq{}& k\frac{C_{\mathrm{ex}}}{4}\Theta C_{A.5.4}^2 h^{-2}\|\mathbf{m}_h^{j+1,\ell} - \mathbf{m}_h^{j+1,\ell-1}\|_h \\
&+ k\frac{C_{\mathrm{ex}}}{4}C_{A.5.4}^2 h^{-2}\|\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}\|_h.
\end{aligned}$$

Erneutes Umformen liefert die Ungleichung

$$\|\mathbf{m}_h^{j+1,\ell+1} - \mathbf{m}_h^{j+1,\ell}\|_h \leq \Theta\frac{C_{\mathrm{ex}}C_{A.5.4}^2 h^{-2}k/4}{1 - C_{\mathrm{ex}}C_{A.5.4}^2 h^{-2}k/4}\|\mathbf{m}_h^{j+1,\ell} - \mathbf{m}_h^{j+1,\ell-1}\|_h.$$

Dies ist das gewünschte Ergebnis unseres Beweises. □

Nun müssen wir noch feststellen, wie h und k gewählt werden müssen, damit wir aus vorigem Lemma eine Kontraktion und damit Konvergenz erhalten.

Satz 5.3.6. *Falls gilt $\Theta\frac{\gamma}{1-\gamma} < 1$ für die Konstanten aus Definition 5.3.3. Dann hat Algorithmus 4.3.2 für jedes j eine eindeutige Lösung. Diese lässt sich durch die Fixpunktiteration aus Algorithmus 5.3.2 mittels $\mathbf{m}_h^{j+1,\infty} := \lim_{\ell\to\infty}\mathbf{m}_h^{j+1,\ell} \in S^1(\mathcal{T}_h)$ berechnen.*

Beweis. Laut vorigem Lemma stellt die Abbildung K aus Lemma 5.3.4 für $\Theta\frac{\gamma}{1-\gamma} < 1$ eine Kontraktion dar. Anwendung des Banachschen Fixpunktsatzes garantiert uns somit einen eindeutigen Fixpunkt $\mathbf{m}_h^{j+1,\infty} := \lim_{\ell\to\infty}\mathbf{m}_h^{j+1,\ell}$ von K, gegen den Algorithmus 5.3.2 konvergiert.

Beachte nun mittels Gleichung (5.26), dass $\mathbf{m}_h^{j+1,\infty}$ genau dann ein Fixpunkt von K ist, falls $\mathbf{m}_h^{j+1,\infty}$ Gleichung (5.25) löst. Diese ist aber äquivalent zur Bedingung (4.6) aus Algorithmus 4.3.2. □

Bemerkung 5.3.7. *Betrachten wir den Ausdruck $\Theta\frac{\gamma}{1-\gamma}$ in Abhängigkeit des Produktes $x := h^{-2}k$, dann ist dieser bis auf die beiden Polstellen stetig in x und hat eine Nullstelle für $x = 0$. Das bedeutet insbesondere, dass $\Theta\frac{\gamma}{1-\gamma} < 1$ gilt, falls x klein genug gewählt wird. Dies beschert uns also die Voraussetzung $k = o(h^2)$ für die Konvergenz des Verfahrens.*

5.3.3. Abbruchkriterium

In diesem Abschnitt zeigen wir nun noch eine Abschätzung für das Residuum der Fixpunktiteration, um das Abbruchkriterium zu rechtfertigen.

Lemma 5.3.8. *Sei $\gamma < 1$ für die Konstante aus Definition 5.3.3. Dann gilt*

$$\begin{aligned}|(d_t\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h - \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h + C_{\mathrm{ex}}(\overline{\mathbf{m}}_h^{j+1/2,\ell} \times \tilde{\Delta}_h\overline{\mathbf{m}}_h^{j+1/2,\ell}, \boldsymbol{\phi}_h)_h \\ + (\overline{\mathbf{m}}_h^{j+1/2,\ell} \times \widetilde{\Pi}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h \\ + (\overline{\mathbf{m}}_h^{j+1/2,\ell} \times \tilde{\mathbf{f}}_h^{j+1/2}, \boldsymbol{\phi}_h)_h| \\ \leq \Theta\frac{C_{\mathrm{ex}}}{4}C_{A.5.4}^2 h^{-2}\|\mathbf{m}_h^{j+1,\ell} - \mathbf{m}_h^{j+1,\ell-1}\|_h\|\boldsymbol{\phi}_h\|_h,\end{aligned}$$

wobei $d_t\mathbf{m}_h^{j+1,\ell} := k^{-1}(\mathbf{m}_h^{j+1,\ell} - \mathbf{m}_h^j)$ und $\overline{\mathbf{m}}_h^{j+1/2,\ell} := \frac{1}{2}(\mathbf{m}_h^{j+1,\ell} + \mathbf{m}_h^j)$ sind.

Beweis. Wir benutzen in erster Instanz die Gleichung (5.26), die wir durch Linearisierung des quadratischen Laplace-Terms erhalten haben. Übrig bleibt deshalb genau der Linearisierungsfehler.

$$\begin{aligned}|(d_t\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h - \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h + C_{\mathrm{ex}}(\overline{\mathbf{m}}_h^{j+1/2,\ell} \times \tilde{\Delta}_h\overline{\mathbf{m}}_h^{j+1/2,\ell}, \boldsymbol{\phi}_h)_h \\ + (\overline{\mathbf{m}}_h^{j+1/2,\ell} \times \widetilde{\Pi}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h + (\overline{\mathbf{m}}_h^{j+1/2,\ell} \times \tilde{\mathbf{f}}_h^{j+1/2}, \boldsymbol{\phi}_h)_h| \\ = \big|\frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h\mathbf{m}_h^{j+1,\ell-1}, \boldsymbol{\phi}_h)_h - \frac{C_{\mathrm{ex}}}{4}(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h\big| \\ = \frac{C_{\mathrm{ex}}}{4}\big|(\mathbf{m}_h^{j+1,\ell} \times \tilde{\Delta}_h(\mathbf{m}_h^{j+1,\ell-1} - \mathbf{m}_h^{j+1,\ell}), \boldsymbol{\phi}_h)_h\big|.\end{aligned}$$

Für den nächsten Schritt bemühen wir die Cauchy-Schwarzsche Ungleichung und Lemma A.3.1.

$$\begin{aligned}|(d_t\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h - \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1,\ell}, \boldsymbol{\phi}_h)_h + C_{\mathrm{ex}}(\overline{\mathbf{m}}_h^{j+1/2,\ell} \times \tilde{\Delta}_h\overline{\mathbf{m}}_h^{j+1/2,\ell}, \boldsymbol{\phi}_h)_h \\ + (\overline{\mathbf{m}}_h^{j+1/2,\ell} \times \widetilde{\Pi}_h\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h + (\overline{\mathbf{m}}_h^{j+1/2,\ell} \times \tilde{\mathbf{f}}_h^{j+1/2}, \boldsymbol{\phi}_h)_h| \\ \leq \frac{C_{\mathrm{ex}}}{4}\|\mathbf{m}_h^{j+1,\ell}\|_{\mathbf{L}^\infty}\|\tilde{\Delta}_h(\mathbf{m}_h^{j+1,\ell-1} - \mathbf{m}_h^{j+1,\ell})\|_h\|\boldsymbol{\phi}_h\|_h.\end{aligned}$$

Unter Benützung von Korollar 4.4 und (5.31) erhalten wir die gewünschte Abschätzung. □

6 Implementierung

In diesem Abschnitt werden wir die entscheidenden Teile der Implementierung besprechen. Neben der bereits vorgestellten Fixpunktiteration stellen wir auch eine Newton-Iteration vor. Diese werden wir allerdings nur aus praktischer Sicht betrachten und Konvergenzbeweise vernachlässigen. Die numerischen Experimente bestätigen dabei die Erwartung einer schnelleren Konvergenz.

6.1. Fixpunktiteration

Wir werden hier nun die wichtigsten Codes für die Implementierung der Bilinearform (5.29) besprechen. Die weiteren Codes finden sich im Anhang. Wir wollen also die folgende Bilinearform auf $S^1(\mathcal{T}_h) \times S^1(\mathcal{T}_h)$ implementieren.

$$\begin{aligned} B_\ell(\boldsymbol{\psi}_h, \boldsymbol{\phi}_h) := \frac{1}{k}(\boldsymbol{\psi}_h, \boldsymbol{\phi}_h)_h - \frac{\alpha}{k}(\mathbf{m}_h^j \times \boldsymbol{\psi}_h, \boldsymbol{\phi}_h)_h \\ + \frac{C_{\text{ex}}}{4}(\boldsymbol{\psi}_h \times \tilde{\Delta}_h \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h + \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \boldsymbol{\psi}_h, \boldsymbol{\phi}_h)_h \\ + \frac{C_{\text{ex}}}{4}(\boldsymbol{\psi}_h \times \tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell-1}, \boldsymbol{\phi}_h)_h \\ + \frac{1}{2}(\boldsymbol{\psi}_h \times (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h. \end{aligned}$$

Dazu wählen wir zuerst eine Basis von $S^1(\mathcal{T}_h)$ mit einer Reihenfolge der Basisvektoren. Die Knoten der Triangulierung seien mittels `coordinates` als ein $N \times 3$-Array gegeben. Um eine 3×3-Blockstruktur zu erhalten verwenden wir die dadurch gegebene Nummerierung $\mathbf{x}_i =$ `coordinates(i,:)` für die nodalen Basisfunktionen $\boldsymbol{\varphi}_\ell \in S^1(\mathcal{T}_h)$ in folgender Weise:

Definition 6.1.1. *Wir definieren für $i \in \{1, \dots, N\}, d \in \{1, 2, 3\}$*

$$\boldsymbol{\varphi}_{i+N(d-1)} := \varphi_i \mathbf{e}_d.$$

Weiters definieren wir $[i, d] := i + N(d-1)$, womit $\boldsymbol{\varphi}_{[i,d]} = \varphi_i \mathbf{e}_d$ gilt. Für $\boldsymbol{\varphi}_m$ schreiben wir damit auch $\boldsymbol{\varphi}_{[i_m, d_m]}$.

Dann bilden wir die Matrix B, für die gilt $B_{m,n} = B_\ell(\boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m)$ für $m, n = 1, \dots, 3N$. Beachte dabei die Reihenfolge von m und n.

Wir bringen nun die einzelnen Beiträge der Matrix auf eine vereinfachte Form. Wir nutzen dabei wieder die alternierende Eigenschaft des Skalarproduktes aus.

$$
\begin{aligned}
B_\ell(\boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m) &= \frac{1}{k}(\boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m)_h - \frac{\alpha}{k}(\mathbf{m}_h^j \times \boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m)_h \\
&\quad + \frac{C_{\text{ex}}}{4}(\boldsymbol{\varphi}_n \times \tilde{\Delta}_h \mathbf{m}_h^j, \boldsymbol{\varphi}_m)_h + \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m)_h \\
&\quad + \frac{C_{\text{ex}}}{4}(\boldsymbol{\varphi}_n \times \tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell-1}, \boldsymbol{\varphi}_m)_h \\
&\quad + \frac{1}{2}(\boldsymbol{\varphi}_n \times (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\varphi}_m)_h \\
&= \frac{1}{k}(\boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m)_h + \frac{\alpha}{k}(\boldsymbol{\varphi}_m \times \boldsymbol{\varphi}_n, \mathbf{m}_h^j)_h \\
&\quad + \frac{C_{\text{ex}}}{4}(\boldsymbol{\varphi}_m \times \boldsymbol{\varphi}_n, \tilde{\Delta}_h \mathbf{m}_h^j)_h - \frac{C_{\text{ex}}}{4}(\boldsymbol{\varphi}_m \times \tilde{\Delta}_h \boldsymbol{\varphi}_n, \mathbf{m}_h^j)_h \\
&\quad + \frac{C_{\text{ex}}}{4}(\boldsymbol{\varphi}_m \times \boldsymbol{\varphi}_n, \tilde{\Delta}_h \mathbf{m}_h^{j+1,\ell-1})_h \\
&\quad + \frac{1}{2}(\boldsymbol{\varphi}_m \times \boldsymbol{\varphi}_n, (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}))_h \\
&= \frac{1}{k}(\boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m)_h - \frac{C_{\text{ex}}}{4}(\boldsymbol{\varphi}_m \times \tilde{\Delta}_h \boldsymbol{\varphi}_n, \mathbf{m}_h^j)_h \\
&\quad + (\boldsymbol{\varphi}_m \times \boldsymbol{\varphi}_n, \frac{\alpha}{k}\mathbf{m}_h^j + \frac{1}{2}(C_{\text{ex}} \tilde{\Delta}_h \frac{\mathbf{m}_h^j + \mathbf{m}_h^{j+1,\ell-1}}{2} + \widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}))_h
\end{aligned}
$$

Es gilt also neben der Linearform

- $L(\boldsymbol{\phi}_h) := \frac{1}{k}(\mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{C_{\text{ex}}}{4}(\mathbf{m}_h^j \times \tilde{\Delta}_h \mathbf{m}_h^j, \boldsymbol{\phi}_h)_h - \frac{1}{2}(\mathbf{m}_h^j \times (\widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2}), \boldsymbol{\phi}_h)_h$

auch die drei Matrizen

- $(\boldsymbol{\varphi}_m, \boldsymbol{\varphi}_n)_h$,
- $(\boldsymbol{\varphi}_m \times \tilde{\Delta}_h \boldsymbol{\varphi}_n, \mathbf{m}_h^j)_h$ und
- $(\boldsymbol{\varphi}_m \times \boldsymbol{\varphi}_n, \boldsymbol{\eta}_h)_h$

für ein $\boldsymbol{\eta}_h \in S^1(\mathcal{T}_h)$ zu implementieren, welches dann nach obiger Rechnung als der Term $\boldsymbol{\eta}_h := \frac{\alpha}{k}\mathbf{m}_h^j + \frac{1}{2}(C_{\text{ex}} \tilde{\Delta}_h \frac{\mathbf{m}_h^j + \mathbf{m}_h^{j+1,\ell-1}}{2} + \widetilde{\Pi}_h \mathbf{m}_h^j + \tilde{\mathbf{f}}_h^{j+1/2})$ gewählt wird.

6.1.1. Matrix $\mathbf{A}_{mn} = (\boldsymbol{\varphi}_m, \boldsymbol{\varphi}_n)_h$

Nach Bemerkung 3.2.6 gilt

$$
(\boldsymbol{\varphi}_m, \boldsymbol{\varphi}_n)_h = \sum_{\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)} \beta_{\mathbf{x}_h} \langle \boldsymbol{\varphi}_m(\mathbf{x}_h), \boldsymbol{\varphi}_n(\mathbf{x}_h) \rangle = \beta_{\mathbf{x}_{i_m}} \langle \boldsymbol{\varphi}_m(\mathbf{x}_{i_m}), \boldsymbol{\varphi}_n(\mathbf{x}_{i_m}) \rangle = \begin{cases} \beta_{\mathbf{x}_{i_m}} & \text{falls } m = n, \\ 0 & \text{sonst.} \end{cases}
$$

Die Matrix $(\boldsymbol{\varphi}_m, \boldsymbol{\varphi}_n)_h$ ist also eine $3N \times 3N$-Diagonalmatrix und lässt sich leicht implementieren. Für die Implementierung dieser Matrix siehe Listing 6.1.

Listing 6.1: h-Skalarprodukt zweier stückweise linearer Funktionen

```
1 function S = inner_P1_P1_h(mesh, phi, psi, dimPhi)
2 %INNER_P1_P1_H    h-inner product of two piecewise linear globally continous
```

```
%functions.
%   S = INNER_P1_P1_H(MESH, PHI, PSI) returns reduced inner_product (.,.)_h
%   defined by $(phi, psi)_h = \int I_h(<phi ,psi>) dx$. ['mass lumping']
%
%   V = INNER_P1_P1_H(MESH, PHI) returns the vector V of PHI tested with
%   all hat functions V(i) == INNER_P1_P1_H(MESH, PHI, HAT(i))
%
%   M = INNER_P1_P1_H(MESH) or M = INNER_P1_P1_H(MESH, [], [], DIMPHI)
%   returns the matrix M(i,j) = INNER_P1_P1_H(MESH, HAT(i), HAT(j))
%   [We could call this the 'reduced mass matrix using mass lumping'].
%
%   Author: Josef Kemetmueller - 16.12.2013

if ~exist('phi','var')
    S = inner_P1_P1_h_mat(mesh, 1);
elseif ~exist('psi','var')
    S = inner_P1_P1_h(mesh, phi, []);
elseif isempty(phi) && isempty(psi)
    S = inner_P1_P1_h_mat(mesh, dimPhi);
elseif isempty(phi)
    S = inner_P1_P1_h_mat(mesh, 1)*psi;
elseif isempty(psi)
    S = inner_P1_P1_h_mat(mesh, 1)*phi;
elseif exist('dimPhi','var') && strcmpi(dimPhi, 'elementwise')
    S = dot(phi, inner_P1_P1_h_mat(mesh, 1)*psi, 2);
else
    S = sum(dot(phi, inner_P1_P1_h_mat(mesh, 1)*psi, 2));
end
end

function S = inner_P1_P1_h_mat(mesh,dimPhi)
nC = size(mesh.coordinates,1);
S = spdiags(repmat(mesh.betas(:),dimPhi,1),0,dimPhi*nC,dimPhi*nC);
end
```

6.1.2. Matrix $\mathbf{A}_{mn} = (\boldsymbol{\varphi}_m \times \tilde{\Delta}_h \boldsymbol{\varphi}_n, \mathbf{m}_h^j)_h$

Für diese Matrix müssen wir auf die Definition des diskreten Laplace-Operators zurückgreifen. Wir formen wieder entsprechend Bemerkung 3.2.6 um.

$$\begin{aligned}\mathbf{A}_{mn} = (\boldsymbol{\varphi}_{[i_m,d_m]} \times \tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}, \mathbf{m}_h^j)_h &= \beta_{\mathbf{x}_{i_m}} \langle \boldsymbol{\varphi}_{[i_m,d_m]}(\mathbf{x}_{i_m}) \times \tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}(\mathbf{x}_{i_m}), \mathbf{m}_h^j(\mathbf{x}_{i_m})\rangle \\ &= \beta_{\mathbf{x}_{i_m}} \langle \mathbf{e}_{d_m} \times \tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}(\mathbf{x}_{i_m}), \mathbf{m}_h^j(\mathbf{x}_{i_m})\rangle \\ &= \beta_{\mathbf{x}_{i_m}} \langle \tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}(\mathbf{x}_{i_m}), \mathbf{e}_{d_n}\rangle \langle \mathbf{e}_{d_m} \times \mathbf{e}_{d_n}, \mathbf{m}_h^j(\mathbf{x}_{i_m})\rangle.\end{aligned}$$

Für die weitere Vereinfachung des obigen Ausdrucks benötigen wir die Gleichheit der Terme $\langle \tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}(\mathbf{x}_{i_m}), \mathbf{e}_{d_n}\rangle = -\beta_{\mathbf{x}_{i_m}}^{-1} (\nabla \varphi_{i_n}, \nabla \varphi_{i_m})_{\mathbf{L}^2}$, die wir mittels

$$\begin{aligned}-(\nabla \varphi_{i_n}, \nabla \varphi_{i_m})_{\mathbf{L}^2} = -(\nabla \boldsymbol{\varphi}_{[i_n,d_n]}, \nabla \boldsymbol{\varphi}_{[i_m,d_n]})_{\mathbf{L}^2} &= (\tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}, \boldsymbol{\varphi}_{[i_m,d_n]})_h \\ &= \sum_{\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)} \beta_{\mathbf{x}_h} \langle \tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}(\mathbf{x}_h), \boldsymbol{\varphi}_{[i_m,d_n]}(\mathbf{x}_h)\rangle \\ &= \beta_{\mathbf{x}_{i_m}} \langle \tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}(\mathbf{x}_{i_m}), \boldsymbol{\varphi}_{[i_m,d_n]}(\mathbf{x}_{i_m})\rangle \\ &= \beta_{\mathbf{x}_{i_m}} \langle \tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}(\mathbf{x}_{i_m}), \mathbf{e}_{d_n}\rangle\end{aligned}$$

erhalten. Es gilt also

$$\mathbf{A}_{mn} = (\boldsymbol{\varphi}_{[i_m,d_m]} \times \tilde{\Delta}_h \boldsymbol{\varphi}_{[i_n,d_n]}, \mathbf{m}_h^j)_h = -(\nabla\varphi_{i_n}, \nabla\varphi_{i_m})_{\mathbf{L}^2} \langle \mathbf{e}_{d_m} \times \mathbf{e}_{d_n}, \mathbf{m}_h^j(\mathbf{x}_{i_m})\rangle.$$

Damit hat die Matrix die folgende Blockstruktur

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & +\mathbf{A}^3 & -\mathbf{A}^2 \\ -\mathbf{A}^3 & \mathbf{0} & +\mathbf{A}^1 \\ +\mathbf{A}^2 & -\mathbf{A}^1 & \mathbf{0} \end{pmatrix}$$

mit Blöcken der Größe $N \times N$ und den Einträgen

$$\mathbf{A}^{\ell}_{ii'} = -(\nabla\varphi_{i'}, \nabla\varphi_i)_{\mathbf{L}^2} \langle \mathbf{e}_{\ell}, \mathbf{m}_h^j(\mathbf{x}_i)\rangle.$$

Für die Implementierung dieser Matrix siehe Listing 6.2.

Listing 6.2: KREUZPRODUKTMATRIX $\mathbf{A}_{mn} = (\boldsymbol{\varphi}_m \times \tilde{\Delta}_h \boldsymbol{\varphi}_n, \mathbf{m}_h^j)_h$

```
function M = inner_HatIxlaphHatJ_P1_h(mesh, P1)
%INNER_HATIXLAPHHATJ_P1_h    Laplace-Cross product matrix using mass lumping.
%   M = INNER_HATIXLAPHHATJ_P1_h(MESH, P1) returns the matrix defined by:
%   M(i,j) = (hat(i) x lap_h(hat(j)) , P1)_h, where the hat functions are
%   numbered by hat(nC*(dim-1)+node), dim=1...3, node=1...nC.
%
%   Author: Josef Kemetmueller - 16.12.2013
assert(all(size(P1) == size(mesh.coordinates)),'Input not nC-by-3 function.');
%%
nE = size(mesh.elements,1);   % number of elements
nC = size(mesh.coordinates,1);   % number of nodes
dim = 3;
dimMesh = size(mesh.elements,2)-1;
indLocal2Global = @(d,nodes) bsxfun(@plus,nC*(d-1),reshape(nodes,[],1));
% Yields the indices to the global block-matrix: <e(i) x e(j) , P1>
ILocal2Global = @(nodes) indLocal2Global([1 1 2 2 3 3],nodes);
JLocal2Global = @(nodes) indLocal2Global([2 3 1 3 1 2],nodes);
% Yields the entries of the block-matrix: <e(i) x e(j) , P1>
crossMatLocal2Global = @(X) [         X(:,3) -X(:,2) ...
                            -X(:,3)          X(:,1) ...
                             X(:,2) -X(:,1)        ];
[S,I,J] = deal(zeros(nE,6,dimMesh+1,dimMesh+1));

for i = 1:dimMesh+1
    phiElI = P1(mesh.elements(:,i),:);
    for j = 1:dimMesh+1
        S(:,:,i,j) = bsxfun(@times,-crossMatLocal2Global(phiElI), ...
                                   mesh.inner_gradHatI_gradHatJ{i,j});
        I(:,:,i,j) = ILocal2Global(mesh.elements(:,i));
        J(:,:,i,j) = JLocal2Global(mesh.elements(:,j));
    end
end
M = sparse(I(:),J(:),S(:),dim*nC,dim*nC);
end
```

6.1.3. Matrix $\mathbf{A}_{mn} = (\boldsymbol{\varphi}_m \times \boldsymbol{\varphi}_n, \boldsymbol{\eta}_h)_h$

Sei erneut $\boldsymbol{\eta}_h \in S^1(\mathcal{T}_h)$ eine feste Funktion. Auch hier formen wir entsprechend Bemerkung 3.2.6 um.

$$\mathbf{A}_{mn} = (\boldsymbol{\varphi}_m \times \boldsymbol{\varphi}_n, \boldsymbol{\eta}_h)_h = \beta_{\mathbf{x}_{i_m}} \langle \boldsymbol{\varphi}_m(\mathbf{x}_{i_m}) \times \boldsymbol{\varphi}_n(\mathbf{x}_{i_m}), \boldsymbol{\eta}_h(\mathbf{x}_{i_m}) \rangle = \delta_{i_m i_n} \beta_{\mathbf{x}_{i_m}} \langle \mathbf{e}_{d_m} \times \mathbf{e}_{d_n}, \boldsymbol{\eta}_h(\mathbf{x}_{i_m}) \rangle$$

Damit ergibt sich wieder eine Matrix von der Blockstruktur

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & +\mathbf{A}^3 & -\mathbf{A}^2 \\ -\mathbf{A}^3 & \mathbf{0} & +\mathbf{A}^1 \\ +\mathbf{A}^2 & -\mathbf{A}^1 & \mathbf{0} \end{pmatrix}$$

mit Diagonalmatrixblöcken der Größe $N \times N$. Die Einträge der Diagonalmatrizen sind dabei durch

$$\mathbf{A}^{\ell}_{ii} = \beta_{\mathbf{x}_i} \langle \mathbf{e}_\ell, \boldsymbol{\eta}_h(\mathbf{x}_i) \rangle$$

gegeben. Die Implementierung dieser Matrix findet sich in Listing 6.3.

Listing 6.3: KREUZPRODUKTMATRIX $\mathbf{A}_{mn} = (\boldsymbol{\varphi}_m \times \boldsymbol{\varphi}_n, \boldsymbol{\eta}_h)_h$

```
function M = inner_HatIxHatJ_P1_h(mesh, P1)
%INNER_HATIXHATJ_P1_H    Cross product matrix using mass lumping.
%   M = INNER_HATIXHATJ_P1_H(MESH, P1) returns the matrix defined by:
%   M(i,j) = (hat(i) x hat(j) , P1)_h, where the hat functions are numbered
%   by hat(nC*(dim-1)+node), dim=1...3, node=1...nC.
%   This results in the following block matrix, which is skew-symmetric.
%
%       [(e1xe1,m(n))_h, (e1xe2,m(n))_h, (e1xe3,m(n))_h ]
%   M = [(e2xe1,m(n))_h, (e2xe2,m(n))_h, (e2xe3,m(n))_h ]
%       [(e3xe1,m(n))_h, (e3xe2,m(n))_h, (e3xe3,m(n))_h ]
%
%   Author: Josef Kemetmueller - 16.12.2013
assert(all(size(P1) == size(mesh.coordinates)),'Input not nC-by-3 function.');
%%
nC = size(mesh.coordinates,1);
dim = 3;
%%
indLocal2Global = @(d,nodes) bsxfun(@plus,nC*(d-1),reshape(nodes,[],1));
% Yields the indices to the global block-matrix: <e(i) x e(j) , phi>
ILocal2Global = @(nodes) indLocal2Global([1 1 2 2 3 3],nodes);
JLocal2Global = @(nodes) indLocal2Global([2 3 1 3 1 2],nodes);
% Yields the entries of the block-matrix: <e(i) x e(j) , phi>
crossMatLocal2Global = @(phi) [           phi(:,3) -phi(:,2) ...
                              -phi(:,3)            phi(:,1) ...
                               phi(:,2) -phi(:,1)          ];
S = crossMatLocal2Global(inner_P1_P1_h(mesh,P1));
I = ILocal2Global(1:nC);
J = JLocal2Global(1:nC);

M = sparse(I,J,S,dim*nC,dim*nC);
end
```

Die genannten Programmteile werden nun in Listing 6.4 zur Fixpunktiteration zusammengesetzt.

Listing 6.4: Fixpunktiteration zur Lösung der nichtlinearen Gleichung

```
function mhjpe = midpoint_fixedPoint(mesh, alpha, C_ex, Piht, fht, mhj, tj, tjpe)
%MIDPOINT_NEWTON    Computes the next magnetization using the midpoint
%scheme and a fixed point-iteration to solve the nonlinear system.
%   MHJPE = MIDPOINT_NEWTON(MESH, ALPHA, C_EX, PIHT, FHT, MHJ, TJ, TJPE)
%   returns the magnetization m_{j+1} for the given input
%       mesh         Geometry of the problem
%       alpha        damping factor
%       C_ex         Exchange constant
%       @(X) Piht(X) Discretized Pi-operator
%       @(t) fht(X)  Discretized external field
%       mhj          Magnetization at previous timestep tj
%       tj           Previous time
%       tjpe         Next time
%
%   Author: Josef Kemetmueller - 16.12.2013
maxIt = 40;
k = tjpe-tj;
%%
h = norm(diff(mesh.coordinates(mesh.elements(1,1:2),:),1));
epsilon = 1e-9*k*(h^2);
%%
mhjpeellm1 = mhj;
%%
h_low = @(mhj) Piht(mhj) + fht((tjpe+tj)/2);
h_eff = @(mhj, mhjpe) C_ex*laplace_hP1(mesh,(mhjpe+mhj)/2) ...
                      + Piht(mhj) + fht((tjpe+tj)/2);

L = inner_P1_P1_h(mesh, 1/k*mhj ...
                         -cross(mhj, 1/2*h_low(mhj) ...
                                   + C_ex/4*laplace_hP1(mesh,mhj),2));
change(1) = NaN;
for l=1:maxIt
    B_l = 1/k*inner_P1_P1_h(mesh,[],[],3) ...
          -C_ex/4*inner_HatIxlaphHatJ_P1_h(mesh,mhj) ...
          +inner_HatIxHatJ_P1_h(mesh,alpha/k*mhj + 1/2*h_eff(mhj,mhjpeellm1));
    % SOLVE
    mhjpeell = reshape(B_l\L(:),[],3);
    %% Stopping criterion
    change(l+1) = norm_P1_h(mesh, mhjpeellm1-mhjpeell);
    if  (change(l+1)<epsilon)
        fprintf('Timestep completed after %d steps.\n',l);
        break;
    elseif change(l+1)*(1+1e-7) > change(l)
        warning(['Tolerance not reached, but the iteration didn''t', ...
                 'change anymore.\n We would have wished for a tolerance ', ...
                 'of eps=%g, but only reached eps=%g'],epsilon, change(end));
        fprintf('Here is a table of the changes in norm: \n');
        fprintf('Step %d: %g\n', [1:length(change); change]);
        break;
    else % Continue iterating.
        mhjpeellm1 = mhjpeell;
    end
end
assert(l<maxIt,'Maximum number of iterations reached.');
mhjpe = mhjpeell;
end
```

6.2. Newton-Iteration

Wir können die nichtlineare Gleichung (5.1) auch mit einem Standardverfahren wie einer Newton-Iteration lösen. Dazu formulieren wir sie auf eine mehrdimensionale Funktion um

$$F\big(\mathbf{m}_h^{j+1}\big)_m := (d_t\mathbf{m}_h^{j+1}, \boldsymbol{\varphi}_m)_h - \alpha(\mathbf{m}_h^j \times d_t\mathbf{m}_h^{j+1}, \boldsymbol{\varphi}_m)_h + (\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, \boldsymbol{\varphi}_m)_h,$$

deren Nullstelle $F(\mathbf{m}_h^{j+1}) = \mathbf{0}$ die gesuchte Magnetisierung darstellt. Wir leiten ab und erhalten die Jacobi-Matrix

$$\begin{aligned}(dF)_{i,j} = \frac{1}{k}(\boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m)_h - \frac{\alpha}{k}(\mathbf{m}_h^j \times \boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m)_h + (\frac{1}{2}\boldsymbol{\varphi}_n \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, \boldsymbol{\varphi}_m)_h \\ + (\overline{\mathbf{m}}_h^{j+1/2} \times \frac{C_{\text{ex}}}{2}\tilde{\Delta}_h\boldsymbol{\varphi}_n, \boldsymbol{\varphi}_m)_h.\end{aligned}$$

Wobei beim Term $(\overline{\mathbf{m}}_h^{j+1/2} \times \tilde{\mathbf{h}}_{\text{eff}}^{j+1/2}, \boldsymbol{\varphi}_m)_h$ die Produktregel zum Einsatz gekommen ist. Es zeigt sich also, dass wir für die Implementierung wieder die gleichen Matrizen wie bei der Fixpunktiteration aus dem vorigen Abschnitt benötigen. Die Newton-Iteration

$$\mathbf{m}_h^{j+1,\ell} := \mathbf{m}_h^{j+1,\ell-1} - (dF)^{-1}F\big(\mathbf{m}_h^{j+1,\ell-1}\big)$$

findet sich in Listing 6.6.

Listing 6.5: Newton-Iteration zur Lösung der nichtlinearen Gleichung

```
function mhjpe = midpoint_newton(mesh, alpha, C_ex, Piht, fht, mhj, tj, tjpe)
%MIDPOINT_NEWTON    Computes the next magnetization using the midpoint
%scheme and a newton-iteration to solve the nonlinear system.
%   MHJPE = MIDPOINT_NEWTON(MESH, ALPHA, C_EX, PIHT, FHT, MHJ, TJ, TJPE)
%   returns the magnetization m_{j+1} for the given input
%       mesh          Geometry of the problem
%       alpha         damping factor
%       C_ex          Exchange constant
%       @(X) Piht(X)  Discretized Pi-operator
%       @(t) fht(X)   Discretized external field
%       mhj           Magnetization at previous timestep tj
%       tj            Previous time
%       tjpe          Next time
%
%   Author: Josef Kemetmueller - 16.12.2013
maxIt = 40;
k = tjpe-tj;
%%
h = norm(diff(mesh.coordinates(mesh.elements(1,1:2),:),1));
epsilon = 1e-9*k*(h^2);
%%
mhjpeellm1 = mhj;
%%
h_eff = @(mhj, mhjpe) C_ex*laplace_hP1(mesh,(mhjpe+mhj)/2) ...
                      + Piht(mhj) + fht((tjpe+tj)/2);
F = @(mhj, mhjpe) inner_P1_P1_h(mesh, [],...
                  1/k*(mhjpe - mhj) ...
                  - alpha*cross(mhj, 1/k*(mhjpe - mhj),2) ...
                  + cross((mhjpe+mhj)/2, h_eff(mhj,mhjpe),2));
change(1) = NaN;
for l = 1:maxIt
```

```
32      dF = 1/k*inner_P1_P1_h(mesh,[],[],3) ...
33          + inner_P1xHatJ_HatI_h(mesh, -alpha/k*mhj ...
34                                       -1/2*(h_eff(mhj,mhjpeellm1))) ...
35          + C_ex/2*inner_P1xlaphHatJ_HatI_h(mesh,(mhj+mhjpeellm1)/2);
36      % SOLVE
37      mhjpeell = mhjpeellm1 - reshape(dF\reshape(F(mhj,mhjpeellm1),[],1),[],3);
38      %% Stopping criterion
39      change(l+1) = norm_P1_h(mesh, mhjpeellm1-mhjpeell);
40      if  (change(l+1)<epsilon)
41          fprintf('Timestep completed after %d steps.\n',l);
42          break;
43      elseif change(l+1)*(1+1e-8) > change(l)
44          warning(['Newton tolerance not reached, but the iteration didn''t', ...
45                   'change anymore.\n We would have wished for a tolerance ', ...
46                   'of eps=%g, but only reached eps=%g'], epsilon, change(end));
47          fprintf('Here is a table of the changes in norm: \n');
48          fprintf('Step %d: %g\n', [1:length(change); change]);
49          break;
50      else % Continue iterating.
51          mhjpeellm1 = mhjpeell;
52      end
53  end
54  assert(l<maxIt,'Maximum number of iterations reached.');
55  mhjpe = mhjpeell;
56  end
```

6.3. Ein Beispielskript

Hier sei nun noch ein Skript angeführt, das die Daten lädt und die Simulation startet. Dieses Beispiel wird im Abschnitt 7.1 behandelt.

Listing 6.6: μMAG STANDARDPROBLEM 3

```
1  %% Load geometry files
2  geometry = 'cube24000';
3  coordinates = dlmread(['coordinates_', geometry, '.txt']);
4  elements = dlmread(['elements_', geometry, '.txt']);
5  %% Define material properties
6  material = struct(...
7      'A',  1e-11,...        % [J/m]
8      'Js', 1,...            % [T]
9      'K',  39788.7346,...   % [J/m^3]
10     'easyaxis', [0,0,1]);
11 %% Precompute values often used in the iteration
12 mesh = precomputeMeshValues(coordinates,elements,material);
13 %% Compute nondimensional time and other constants
14 t0 = 0; % [ns]
15 tEnd = 1e-9; % [ns]
16 [C_ex, C_ani, tau0, tauEnd, mesh.material.Ms] = ...
17                 nondimensionalization(material, t0, tEnd);
18 %% Define initial magnetziation
19 nC = size(coordinates,1);
20 mh0 = [zeros(nC,1),zeros(nC,1),ones(nC,1)];
21 %% Define external field
22 f = @(X) zeros(size(X));
23 %% Combine simulation data
```

```
ExampleData = struct(...
    'mesh',   mesh, ...
    'alpha',  1, ...
    'C_ex',   C_ex, ...
    'Piht',   @(m) hTildeP0(mesh,strayfield(mesh, m)) ...
                  -C_ani*hTildeP1(mesh,anisotropy(m,material)), ...
    'fht',    constantInTimeExternalFieldTilde(mesh, f), ...
    'mh0',    mh0, ...
    'tau0',   tau0, ...
    'tauEnd', tauEnd, ...
    'steps',  2500);
%% Start simulation while saving and plotting the results
name = 'standard_problem_3_40nm';
resultFolder = [thisDir,'/results'];
timeStepping(ExampleData, @midpoint_newton, ...
           plotAndSavePostprocessor(mesh, resultFolder, name));
```

7 Numerische Experimente

Zur numerischen Verifikation des Verfahrens wurden drei Simulationen durchgeführt, deren Problemstellung vom U.S.-amerikanischen *National Institute of Standards and Technology* definiert wurden. Die Abteilung CTCMS definiert dabei vier mikromagnetische Problemstellungen (siehe `http://www.ctcms.nist.gov/~rdm/mumag.org.html`), die zum Gütevergleich von Mikromagnetismus-Simulatoren verwendet werden können.

7.1. μmag Standardproblem 3

Ziel dieser Simulation ist es die *Eindomänengrenze* eines würfelförmigen magnetischen Partikels festzustellen. Diese stellt die Kantenlänge dar, bei der zwei verschieden geartete Zustände minimaler Energie die gleiche Gesamtenergie beinhalten. Bei den zwei Zuständen handelt es sich einerseits um den sogenannten *Flower-Zustand* und andererseits um den *Vortex-Zustand*.

Nun zur genauen Problemstellung: Gegeben sei ein Würfel mit uniaxialer Anisotropie parallel zu einer der Achsen des Würfels und Anisotropiekonstante $K = 3{,}9788736 \cdot 10^5\,[J/m^3]$, Sättigungsmagnetisierung $\mathrm{M}_s = 1/(4\pi 10^{-7})\,[A/m]$ und Austauschkonstante $A = 10^{-11}\,[J/m]$. Wir simulieren für verschiedene Kantenlängen ℓ und Startmagnetisierungen einen Zeitraum von $1[ns]$. Nach diesem Zeitraum bestimmen wir die einzelnen Energiebeiträge und verwenden diese Werte um die Eindomänengrenze zu schätzen. Für die Simulation, die im Flower-Zustand enden soll wählen wir eine homogene Magnetisierung entlang der Anisotropieachse. Um den Vortex-Zustand zu erhalten kehren wir die Magnetisierung auf einem halbierenden Schnitt durch die Anisotropieachse um. In Abbildung 7.1, 7.2 und 7.3 werden nun für ausgewählte Zeitpunkte die Magnetisierungen dargestellt. Wir verwenden eine regelmäßige Triangulierung mit 16^3 Würfeln bestehend aus 6 Tetraedern. Das entspricht in Summe 24.576 Tetraedern. Die Simulationszeit von $1[ns]$ wird in 2500 Zeitschritte unterteilt. Das entspricht damit Orts- und Zeitschrittweiten von $h = \ell\sqrt{3}/16 \cdot 10^{-9} \approx \ell \cdot 10^{-10}[m]$ bzw. $k = 4 \cdot 10^{-13}[s]$ bei einer Kantenlänge von $\ell[nm]$. Wir rechnen mit vollem Streufeld und verwenden einen Dämpfungsparameter von $\alpha = 1$.

Für die inhomogene Startmagnetisierung pendelt sich der wirbelförmige Vortex-Zustand als Zustand geringster Energie ein. Für den homogen magnetisierten Würfel entstehen abhängig von der Kantenlänge verschiedene Ausprägungen des Flower-Zustands. Bei kleineren Kantenlängen ein symmetrischer, bei Kantenlängen ab $42[nm]$ beginnt sich dieser symmetrische Zustand noch etwas einzudrehen und resultiert im getwisteten Flower-Zustand.

In Abbildung 7.4 werden zu den durchgeführten Simulationen die mittlere Energie pro Volumen relativ zur magnetostatischen Energiedichte $K_m = 1/2\mu_0\mathrm{M}_s^2\ [J/m^3]$ auf die Kantenlänge

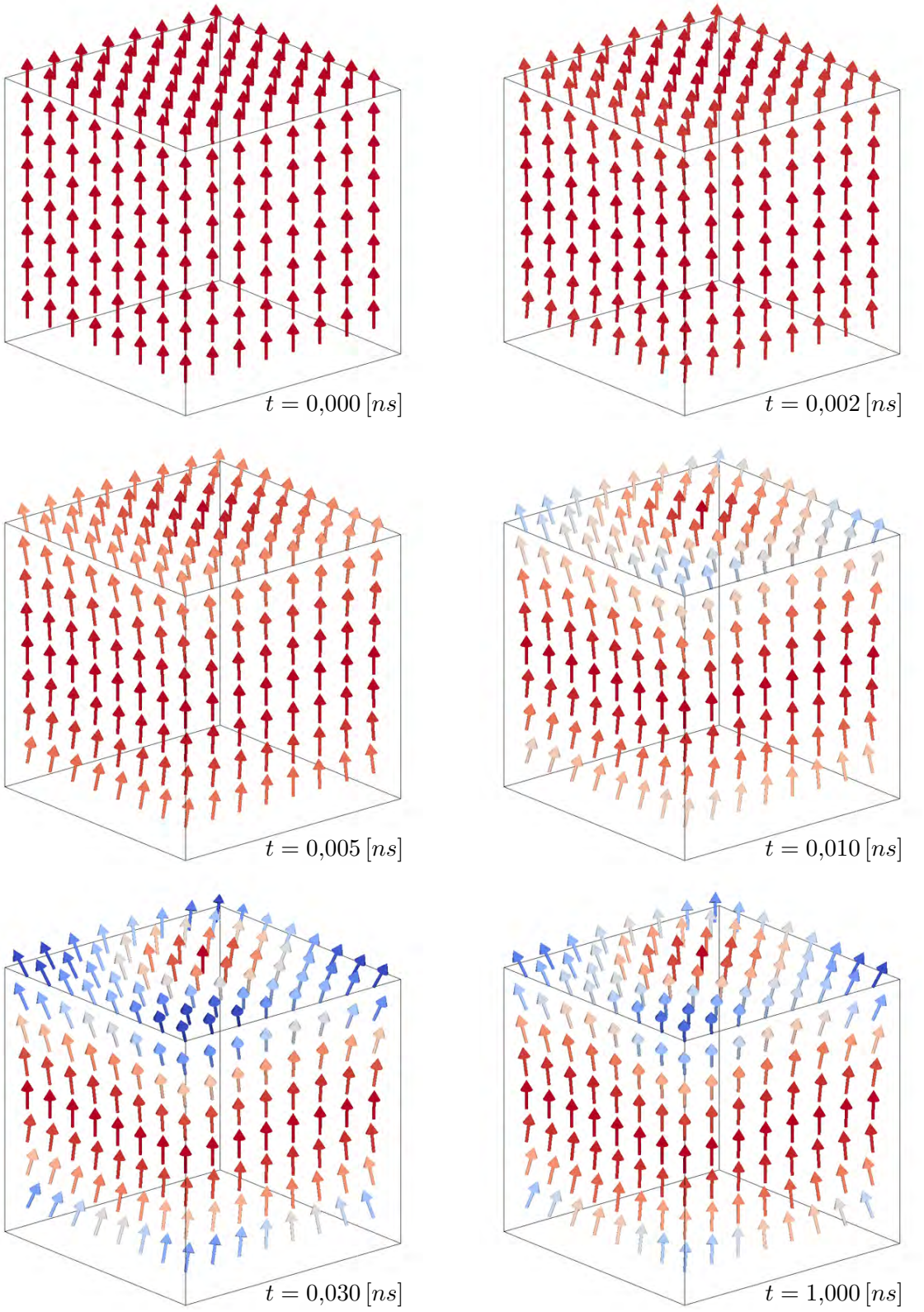

Abbildung 7.1.: Simulation eines Würfels mit Kantenlänge $39\,[nm]$ und homogener Startmagnetisierung. Es stellt sich der symmetrische *Flower-Zustand* ein. Die Färbung entspricht der z-Koordinate der Magnetisierung.

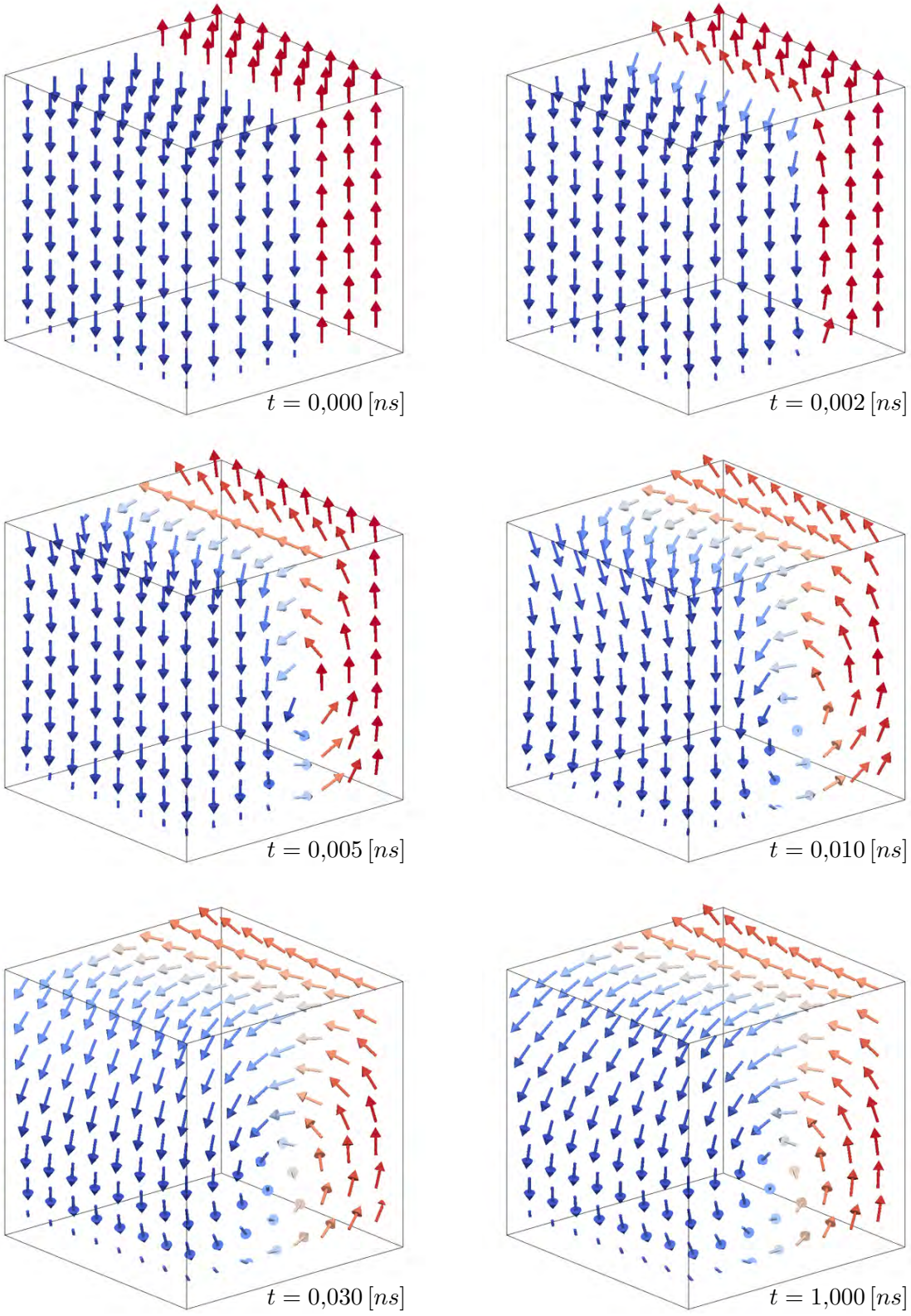

Abbildung 7.2.: Simulation eines Würfels mit Kantenlänge $39\,[nm]$ und inhomogener Startmagnetisierung. Es stellt sich der *Vortex-Zustand* ein. Die Färbung entspricht der z-Koordinate der Magnetisierung.

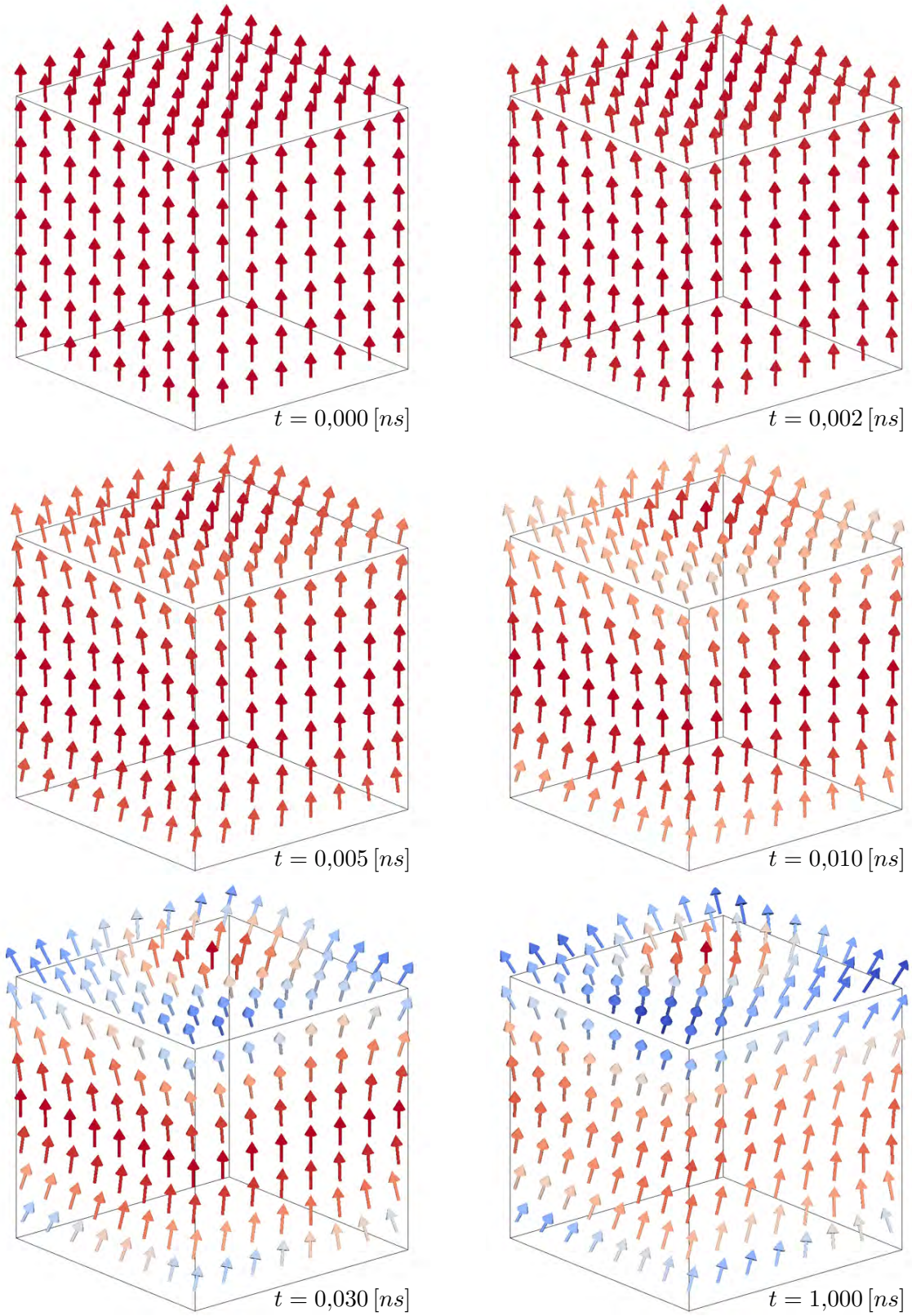

Abbildung 7.3.: Simulation eines Würfels mit Kantenlänge 43 $[nm]$ und homogener Startmagnetisierung. Es stellt sich der *getwistete Flower-Zustand* ein. Die Färbung entspricht der z-Koordinate der Magnetisierung.

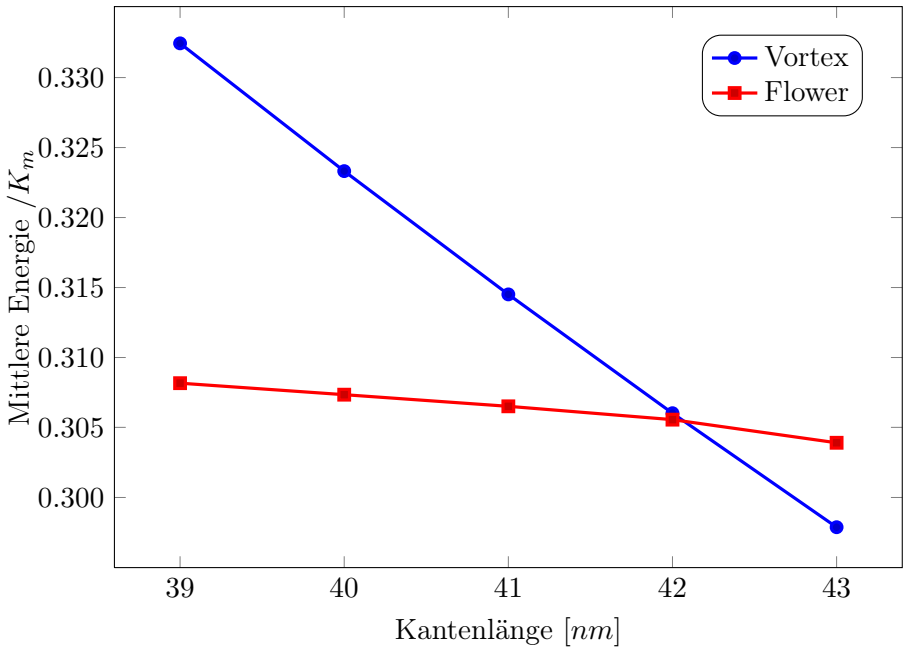

Abbildung 7.4.: Energie pro Volumen der einzelnen Simulationen nach einer Nanosekunde. Die Werte sind dabei dimensionslos relativ zur magnetostatischen Energiedichte $K_m = 1/2\mu_0 \mathrm{M}_s^2$ $[J/m^3]$ gegeben.

aufgetragen. Dort kann nun die berechnete Eindomänengrenze von etwas mehr als $42[nm]$ entnommen werden. Die auf der Webseite des NIST veröffentlichten Ergebnisse bewegen sich dabei auch zwischen 42 und $43[nm]$. Entgegen den veröffentlichten Daten wurden in diesem Fall die Energien nach $1[ns]$ gemessen und nicht bis zum Gleichgewichtszustand extrapoliert. Dies führt dazu, dass die hier berechnete Eindomänengrenze etwas niedriger ausfällt. Die Implementierung besteht diesen Test damit.

7.2. μmag Standardproblem 4

Das nun vorgestellte Problem beinhaltet mehr die dynamischen Aspekte der mikromagnetischen Simulation. Berechnet wird nun die zeitliche Entwicklung der Magnetisierung eines Quaders, der einer Permalloy-Legierung ähnelt. Ein achsenparalleler Film mit einer Dicke von $3[nm]$ und einer Breite und Länge von $125[nm]$ bzw. $500[nm]$ befindet sich im Gleichgewichtszustand eines sogenannten *S-States*. Wie in Abbildung 7.5 ersichtlich ist auch hier wieder die Form namensgebend. Dieser pendelt sich etwa ein, wenn man entlang der Achse $[1, 1, 1]$ ein Feld ausreichender Stärke anlegt und dieses dann langsam abklingen lässt. In diesem Beispiel wurde dafür ein Feld mit der Stärke $\mu_0 \mathrm{M}_s[T]$ angelegt und für die Dauer von $1[ns]$ linear verschwinden gelassen. Um einen Gleichgewichtszustand zu erreichen, wurde anschließend eine weitere Nanosekunde ohne angelegtes Feld gerechnet. Die Austauschkonstante wählen wir als $A = 1{,}3 \cdot 10^{-11}\,[J/m]$, die Sättigungsmagnetisierung als $\mathrm{M}_s = 8{,}0 \cdot 10^5 [A/m]$ und die Anisotropiekonstante als $K = 0{,}0\,[J/m^3]$. Die Gilbert-Dämpfungskonstante wird als $\alpha = 0{,}02$ gewählt. Als Zeitschrittweite verwenden wir $k = 0{,}0002\,[ns]$ für eine Simulationsdauer von einer Nanosekunde. Als Triangulierung verwenden wir ein Netz mit 29.483 Tetraedern und daraus resultierend eine Ortsschrittweite von

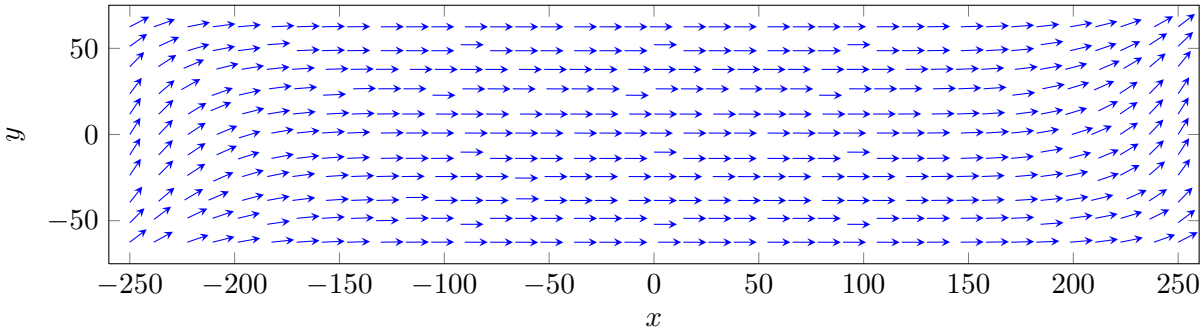

Abbildung 7.5.: *S-State*, der durch ein langsam abklingendes Feld entlang der Achse $[1, 1, 1]$ entsteht.

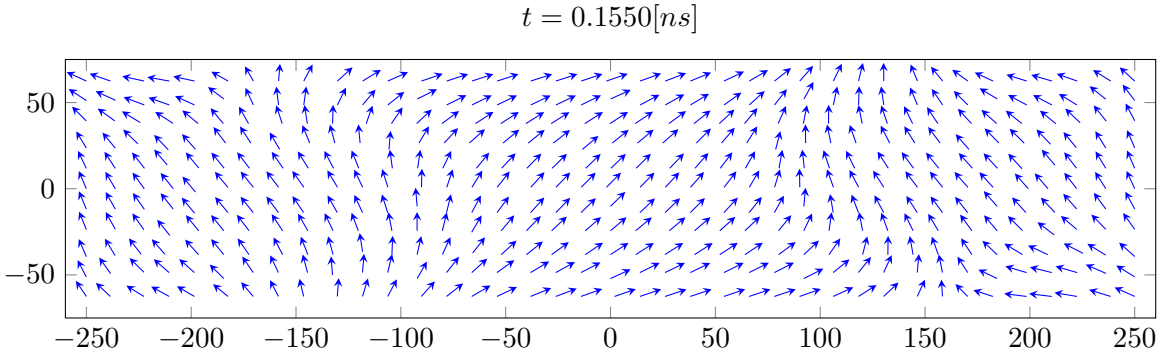

Abbildung 7.6.: Zeitpunkt, zu dem die durchschnittliche Magnetisierung aus Experiment 1 in x-Richtung zum ersten Mal den Wert 0 erreicht.

$h \approx 4 \cdot 10^{-9}[nm]$. Es werden nun zwei verschiedene Felder angelegt.

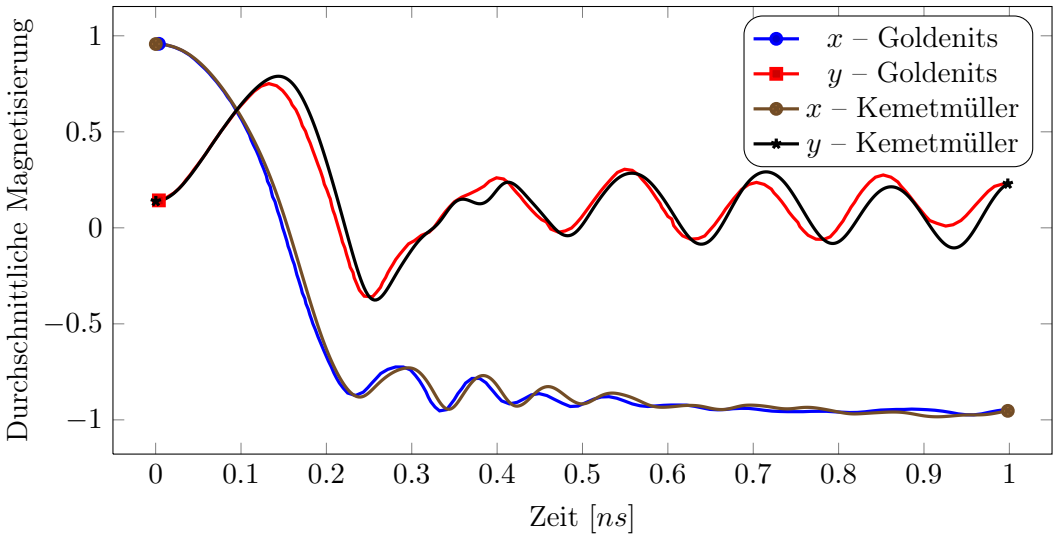

Abbildung 7.7.: Durchschnittliche Magnetisierung beim Anlegen des Feldes aus Experiment 1.

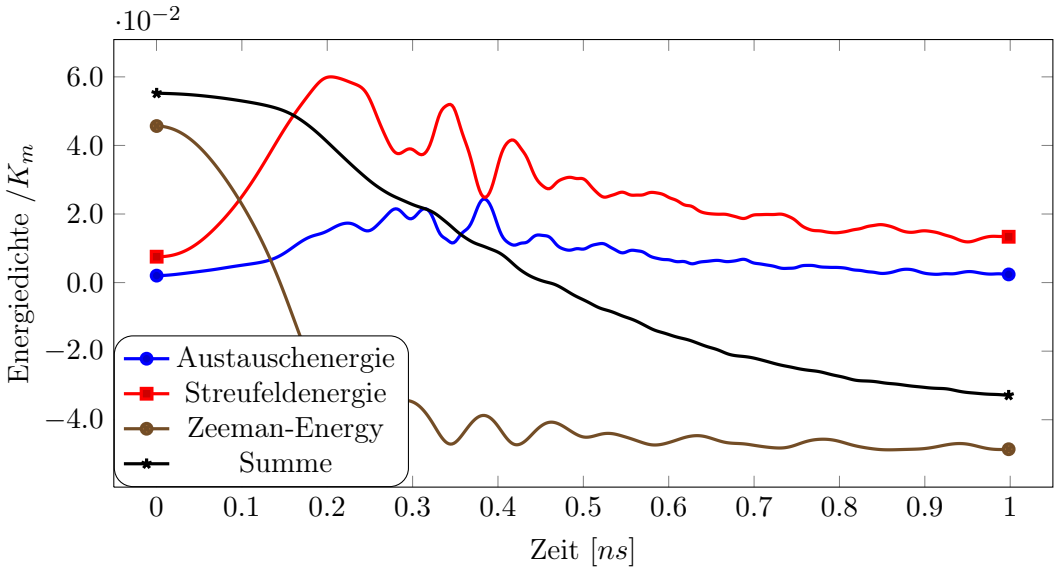

Abbildung 7.8.: Entwicklung der Energiebeiträge bei Anlegen des Feldes aus Experiment 1.

Experiment 1

Das erste Feld sei durch die Komponenten $\mu_0 F_x = -24{,}6[mT]$, $\mu_0 F_y = 4{,}3\,[mT]$ und $\mu_0 F_z = 0{,}0\,[mT]$ gegeben. Der Permalloy-Film ist achsenparallel mit der Länge in x-Richtung und der Breite in y-Richtung ausgerichtet. Das Feld wird ab $t = 0\,[ns]$ angelegt. Zum Vergleich mit den Ergebnissen auf der Webseite des NIST ist in Abbildung 7.6 die Magnetisierung visualisiert, zum Zeitpunkt an dem die durchschnittliche Magnetisierung in Richtung der x-Achse zum ersten Mal den Wert 0 erreicht. Die Komponenten der durchschnittlichen Magnetisierung sind in Abbildung 7.7 zu sehen. Wir vergleichen hier die Ergebnisse mit den numerischen Resultaten aus [Gol12]. Abbildung 7.8 zeigt den Verlauf der einzelnen Energiebeiträge. Diese sind dabei wieder relativ zur magnetostatischen Energiedichte $K_m = 1/2\mu_0 \mathrm{M}_s^2$ $[J/m^3]$ angegeben. Die Abbildungen 7.9 und 7.10 zeigen den gesamten Verlauf der Magnetisierung für ausgewählte Zeitpunkte.

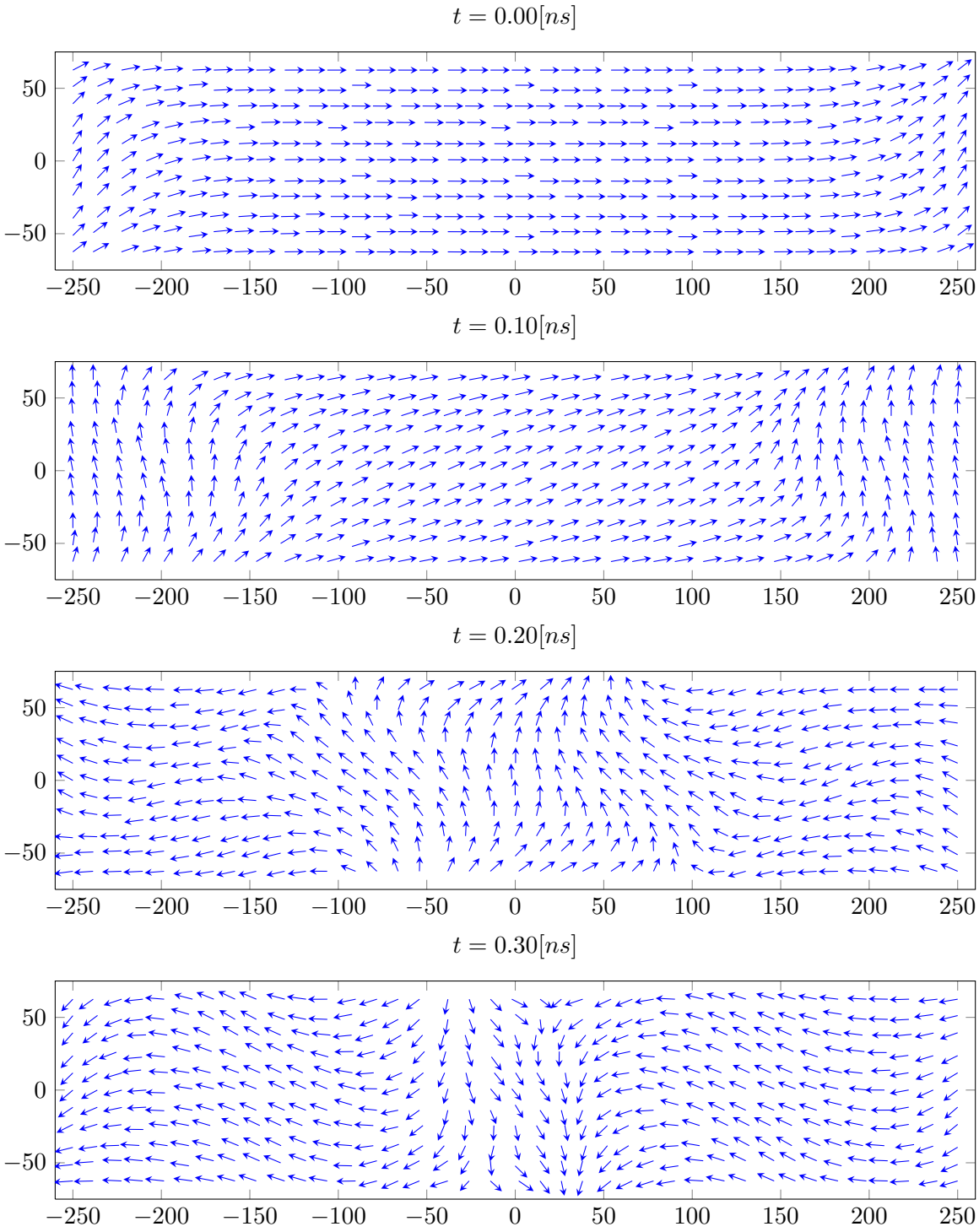

Abbildung 7.9.: Ausgewählte Zeitpunkte aus Experiment 1 zu denen jeweils die Magnetisierung visualisiert wird.

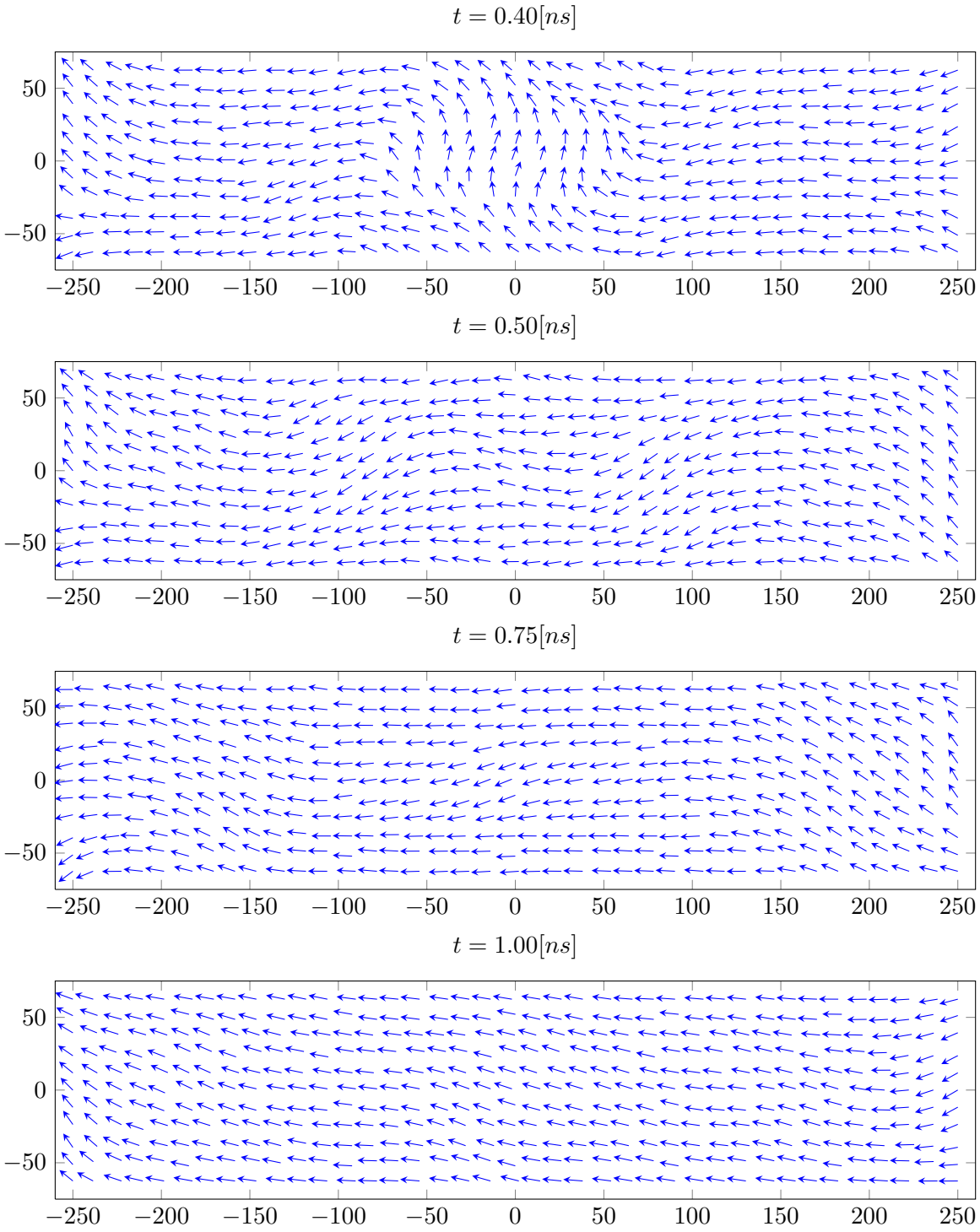

Abbildung 7.10.: Fortsetzung ausgewählter Zeitpunkte aus Experiment 1 zu denen jeweils die Magnetisierung visualisiert wird.

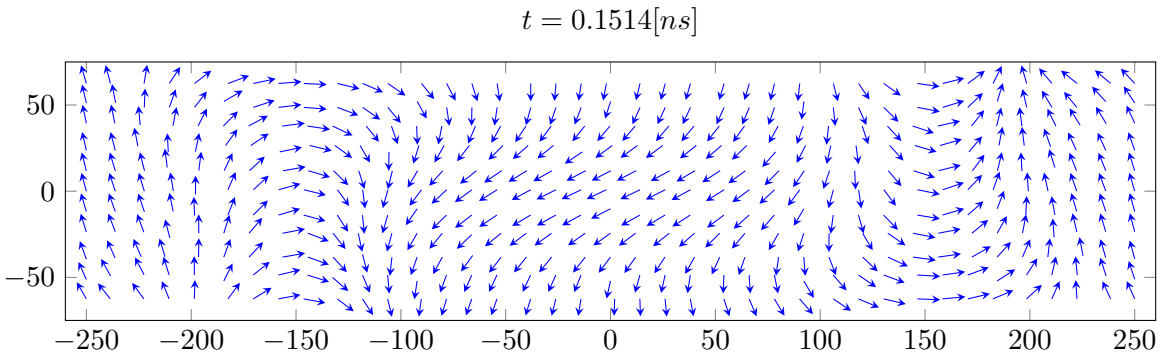

Abbildung 7.11.: Zeitpunkt, zu dem die durchschnittliche Magnetisierung aus Experiment 2 in x-Richtung zum ersten Mal den Wert 0 erreicht.

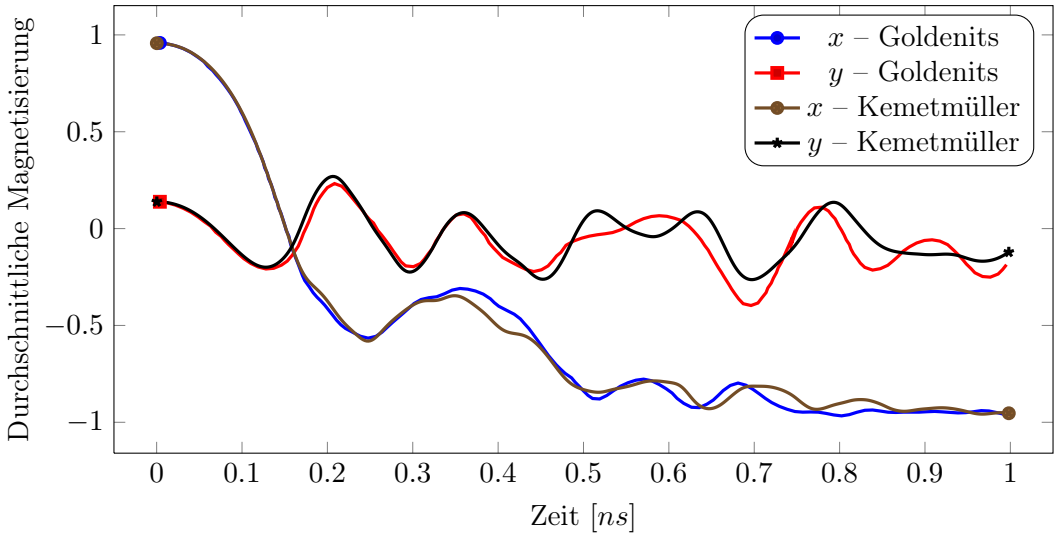

Abbildung 7.12.: Durchschnittliche Magnetisierung beim Anlegen des Feldes aus Experiment 2.

Experiment 2

Die Voraussetzungen für die zweite Simulation bleiben dieselben. Lediglich das angelegte Feld ist geändert durch $\mu_0 F_x = -35{,}5[mT]$, $\mu_0 F_y = -6{,}3[mT]$ und $\mu_0 F_z = 0{,}0[mT]$ gegeben. In Abbildung 7.11 ist erneut ein Bild der Magnetisierung gegeben, wenn deren x-Komponente durchschnittlich zum ersten Mal null wird. Die durchschnittliche Magnetisierung kann Abbildung 7.12 entnommen werden. Auch hier vergleichen wir die Ergebnisse mit den numerischen Resultaten aus [Gol12]. Abbildung 7.13 zeigt den Verlauf der einzelnen Energiebeiträge über die Dauer der Simulation. Abschließend sind in den Abbildungen 7.14 und 7.15 wieder ausgewählte Zeitpunkte visualisiert.

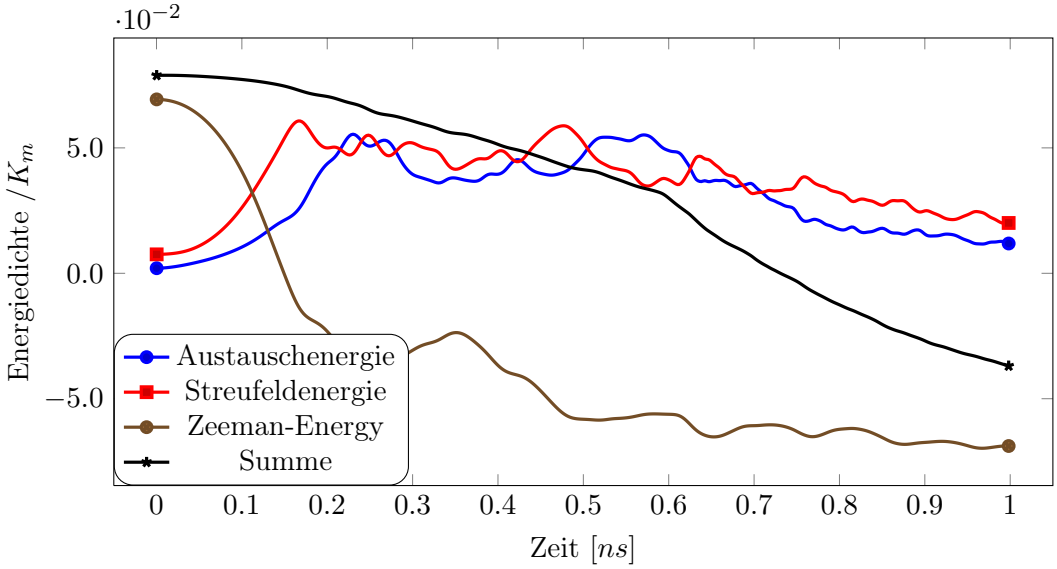

Abbildung 7.13.: Entwicklung der Energiebeiträge bei Anlegen des Feldes aus Experiment 2.

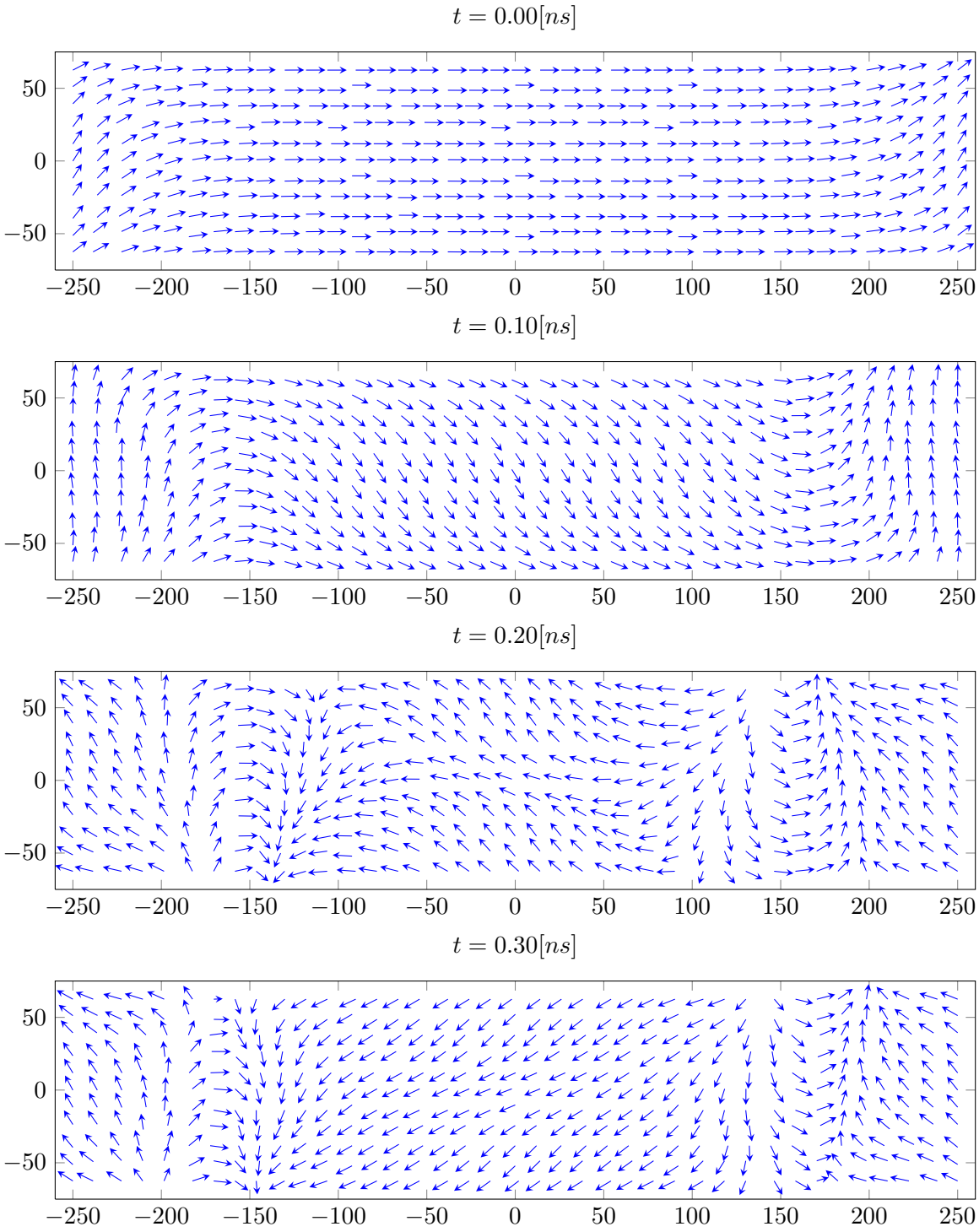

Abbildung 7.14.: Ausgewählte Zeitpunkte aus Experiment 2 zu denen jeweils die Magnetisierung visualisiert wird.

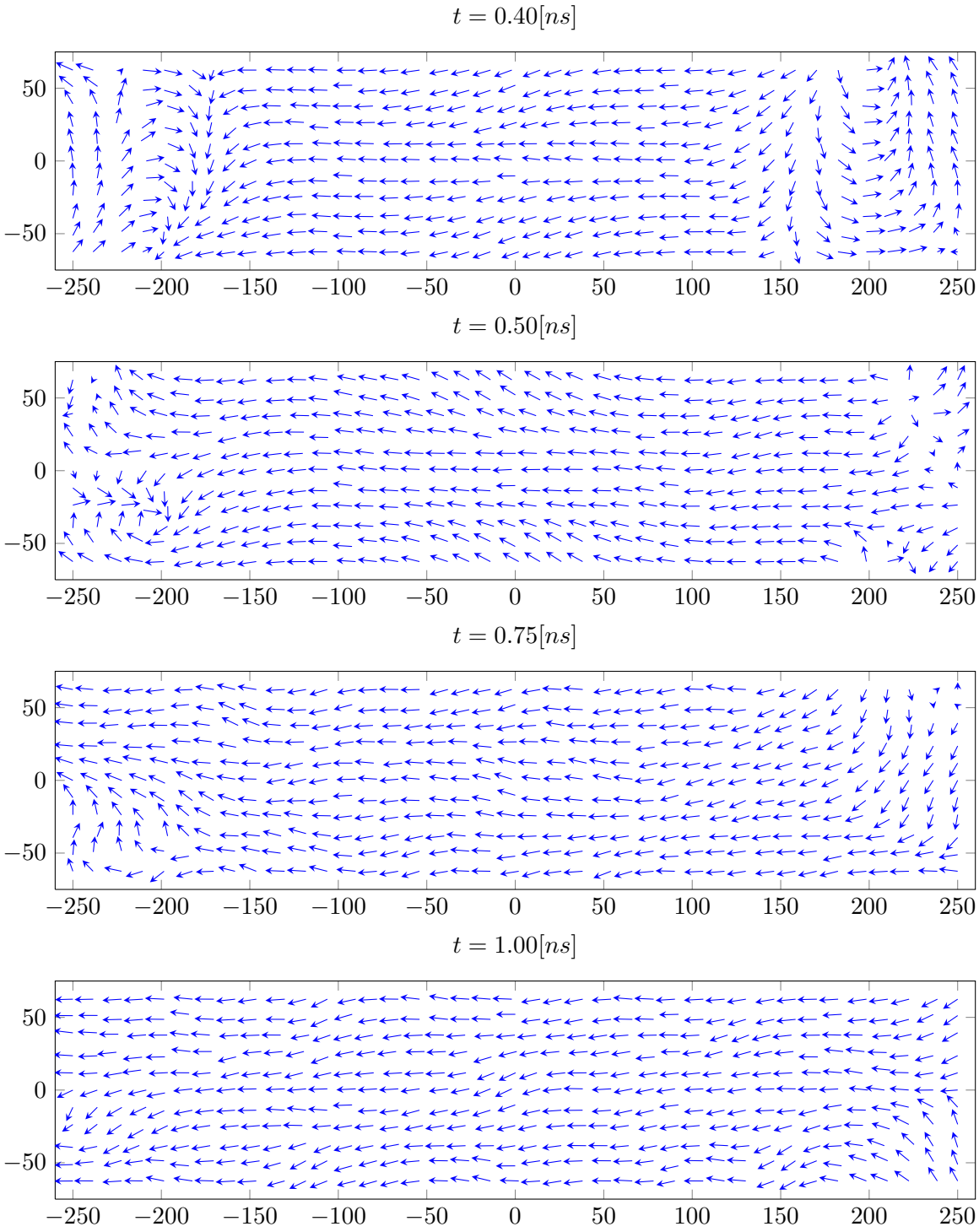

Abbildung 7.15.: Fortsetzung ausgewählter Zeitpunkte aus Experiment 2 zu denen jeweils die Magnetisierung visualisiert wird.

A

Wahrheiten

A.1. Diverse Rechenregeln

Lemma A.1.1. *Folgende Eigenschaften des Kreuzproduktes gelten für $a, b, c \in \mathbb{R}^3$:*

1. *Graßmann-Identität:*

$$a \times (b \times c) = \langle a, c\rangle b - \langle a, b\rangle c$$

2. *Die alternierende Eigenschaft des Spatproduktes*

$$\langle a \times b, c\rangle = \det(a, b, c) = -\langle a \times c, b\rangle = \langle a, b \times c\rangle$$

3. *Die Orthogonalitätseigenschaft*

$$\langle a, a \times b\rangle = 0 \text{ und } \langle b, a \times b\rangle = 0.$$

4. *Die Lagrange-Identität*

$$\langle a \times b, c \times d\rangle = \langle a, c\rangle\langle b, d\rangle - \langle b, c\rangle\langle a, d\rangle$$

5. *Eine Abschätzung die direkt aus der Lagrange-Identität folgt*

$$|a \times b| \leq |a||b|$$

6. *Die Produktregel für die Ableitung des Kreuzproduktes*

$$\frac{\partial}{\partial x_j}(\mathbf{u} \times \mathbf{v}) = \big(\frac{\partial}{\partial x_j}\mathbf{u}\big) \times \mathbf{v} + \mathbf{u} \times \big(\frac{\partial}{\partial x_j}\mathbf{v}\big)$$

Lemma A.1.2. *Für alle $\boldsymbol{\xi}, \boldsymbol{\eta} \in \mathbf{H}^1(\Omega; \mathbb{R}^3)$ gilt*

$$(\nabla\boldsymbol{\xi}, \nabla(\boldsymbol{\xi} \times \boldsymbol{\eta})) = (\nabla\boldsymbol{\xi}, \boldsymbol{\xi} \times \nabla\boldsymbol{\eta}).$$

Beweis. Diese Eigenschaft folgt unmittelbar aus der spaltenweisen Anwendung der Produktregel für das Kreuzprodukt.

$$(\nabla\boldsymbol{\xi}, \nabla(\boldsymbol{\xi} \times \boldsymbol{\eta})) = (\nabla\boldsymbol{\xi}, \nabla\boldsymbol{\xi} \times \boldsymbol{\eta}) + (\nabla\boldsymbol{\xi}, \boldsymbol{\xi} \times \nabla\boldsymbol{\eta}) = (\nabla\boldsymbol{\xi}, \boldsymbol{\xi} \times \nabla\boldsymbol{\eta}).$$

Hier fällt der Summand $(\nabla\boldsymbol{\xi}, \nabla\boldsymbol{\xi} \times \boldsymbol{\eta})$ weg, weil er die Struktur $\langle a, a \times b\rangle$ aufweist. □

A.2. LLG-Gleichung

Lemma A.2.1. *Folgende Formulierungen der LLG-Gleichung sind äquivalent*

$$\mathbf{m}_t = -\frac{1}{1+\alpha^2}\mathbf{m}\times\mathbf{h}_{\text{eff}} - \frac{\alpha}{1+\alpha^2}\mathbf{m}\times(\mathbf{m}\times\mathbf{h}_{\text{eff}}), \tag{A.1}$$

$$\mathbf{m}_t = \alpha\mathbf{m}\times\mathbf{m}_t - \mathbf{m}\times\mathbf{h}_{\text{eff}}. \tag{A.2}$$

Beweis. Wir beginnen mit Gleichung (A.1) und bilden das Kreuzprodukt mit $\alpha\mathbf{m}$,

$$\mathbf{m}_t\times\alpha\mathbf{m} = -(\frac{1}{1+\alpha^2}\mathbf{m}\times\mathbf{h}_{\text{eff}})\times\alpha\mathbf{m} - (\frac{\alpha}{1+\alpha^2}\mathbf{m}\times(\mathbf{m}\times\mathbf{h}_{\text{eff}}))\times\alpha\mathbf{m}.$$

Im nächsten Schritt benutzen wir, dass das Kreuzprodukt alternierend ist: $a\times b = -b\times a$,

$$\begin{aligned}-\alpha\mathbf{m}\times\mathbf{m}_t &= \alpha\mathbf{m}\times(\frac{1}{1+\alpha^2}\mathbf{m}\times\mathbf{h}_{\text{eff}}) + \alpha\mathbf{m}\times(\frac{\alpha}{1+\alpha^2}\mathbf{m}\times(\mathbf{m}\times\mathbf{h}_{\text{eff}}))\\ &= \frac{\alpha}{1+\alpha^2}\mathbf{m}\times(\mathbf{m}\times\mathbf{h}_{\text{eff}}) + \frac{\alpha^2}{1+\alpha^2}\mathbf{m}\times(\mathbf{m}\times(\mathbf{m}\times\mathbf{h}_{\text{eff}})).\end{aligned}$$

Nun benötigen wir die Graßmann-Identität $a\times(b\times c) = \langle a,c\rangle b - \langle a,b\rangle c$ um weiter umzuformen,

$$\begin{aligned}-\alpha\mathbf{m}\times\mathbf{m}_t &= \frac{\alpha}{1+\alpha^2}\mathbf{m}\times(\mathbf{m}\times\mathbf{h}_{\text{eff}}) + \frac{\alpha^2}{1+\alpha^2}\mathbf{m}\times(\langle\mathbf{m},\mathbf{h}_{\text{eff}}\rangle\mathbf{m} - \langle\mathbf{m},\mathbf{m}\rangle\mathbf{h}_{\text{eff}})\\ &= \frac{\alpha}{1+\alpha^2}\mathbf{m}\times(\mathbf{m}\times\mathbf{h}_{\text{eff}}) + \frac{\alpha^2}{1+\alpha^2}\langle\mathbf{m},\mathbf{h}_{\text{eff}}\rangle\underbrace{\mathbf{m}\times\mathbf{m}}_{=0} - \frac{\alpha^2}{1+\alpha^2}\underbrace{\langle\mathbf{m},\mathbf{m}\rangle}_{=1}\mathbf{m}\times\mathbf{h}_{\text{eff}}.\end{aligned}$$

Damit ergibt sich

$$-\alpha\mathbf{m}\times\mathbf{m}_t = -\frac{\alpha^2}{1+\alpha^2}\mathbf{m}\times\mathbf{h}_{\text{eff}} + \frac{\alpha}{1+\alpha^2}\mathbf{m}\times(\mathbf{m}\times\mathbf{h}_{\text{eff}}). \tag{A.3}$$

Nun addieren wir Gleichung (A.3) auf Gleichung (A.1) und erhalten

$$\mathbf{m}_t - \alpha\mathbf{m}\times\mathbf{m}_t = \frac{-1-\alpha^2}{1+\alpha^2}\mathbf{m}\times\mathbf{h}_{\text{eff}},$$

was mit einem letzten Umformungsschritt

$$\mathbf{m}_t = \alpha\mathbf{m}\times\mathbf{m}_t - \mathbf{m}\times\mathbf{h}_{\text{eff}}$$

genau die gewünschte Identität aus Gleichung (A.2) darstellt. □

A.3. Abschätzungen

Lemma A.3.1. *Die h-Norm erfüllt für alle* $\boldsymbol{\phi}_h, \boldsymbol{\psi}_h \in S^1(\mathcal{T}_h)$

$$\|\boldsymbol{\phi}_h\times\boldsymbol{\psi}_h\|_h \le \|\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty}\|\boldsymbol{\psi}_h\|_h.$$

Beweis. Wir verwenden hier als ersten Schritt die Zerlegung aus Bemerkung 3.2.6, sowie die Abschätzung aus Bemerkung A.1.1 Punkt 5.

$$\begin{aligned}\|\boldsymbol{\phi}_h \times \boldsymbol{\psi}_h\|_h^2 &= \sum_{\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)} \beta_{\mathbf{x}_h} |\boldsymbol{\phi}_h(\mathbf{x}_h) \times \boldsymbol{\psi}_h(\mathbf{x}_h)|^2 \\ &\leq \sum_{\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)} \beta_{\mathbf{x}_h} |\boldsymbol{\phi}_h(\mathbf{x}_h)|^2 |\boldsymbol{\psi}_h(\mathbf{x}_h)|^2\end{aligned}$$

Nun schätzen wir für $\boldsymbol{\phi}_h(\mathbf{x}_h)$ durch das Supremum ab und fassen die Terme geeignet zusammen.

$$\begin{aligned}\|\boldsymbol{\phi}_h \times \boldsymbol{\psi}_h\|_h^2 &\leq \sup_{\mathbf{x}_h \in \Omega} |\boldsymbol{\phi}_h(\mathbf{x}_h)|^2 \sum_{\mathbf{x}_h \in \mathcal{N}(\mathcal{T}_h)} \beta_{\mathbf{x}_h} |\boldsymbol{\psi}_h(\mathbf{x}_h)|^2 \\ &= \|\boldsymbol{\phi}_h\|_{\mathbf{L}^\infty}^2 \|\boldsymbol{\psi}_h\|_h^2,\end{aligned}$$

was die gewünschte Abschätzung zeigt. □

Obige Abschätzungen lassen sich auch für die $\mathbf{L}^2$-Norm zeigen.

Lemma A.3.2. *Sei $|X| < \infty$ dann erfüllt die $\mathbf{L}^2$-Norm für alle $\boldsymbol{\phi} \in \mathbf{L}^\infty(X; \mathbb{R}^3)$ und $\boldsymbol{\psi} \in \mathbf{L}^2(X; \mathbb{R}^3)$*

$$\|\boldsymbol{\phi} \times \boldsymbol{\psi}\|_{\mathbf{L}^2} \leq \|\boldsymbol{\phi}\|_{\mathbf{L}^\infty} \|\boldsymbol{\psi}\|_{\mathbf{L}^2}.$$

Beweis. Wir verwenden wieder die Abschätzung aus Bemerkung A.1.1 Punkt 5. Dann liefert

$$\begin{aligned}\|\boldsymbol{\phi} \times \boldsymbol{\psi}\|_{\mathbf{L}^2}^2 &= \int_X |\boldsymbol{\phi}(\mathbf{x}) \times \boldsymbol{\psi}(\mathbf{x})|^2 d\mathbf{x} \\ &\leq \int_X |\boldsymbol{\phi}(\mathbf{x})|^2 |\boldsymbol{\psi}(\mathbf{x})|^2 d\mathbf{x} \\ &\leq \big\| |\boldsymbol{\phi}(\mathbf{x})|^2 \big\|_{L^\infty} \big\|\boldsymbol{\psi}\big\|_{\mathbf{L}^2}^2 = \|\boldsymbol{\phi}\|_{\mathbf{L}^\infty}^2 \|\boldsymbol{\psi}\|_{\mathbf{L}^2}^2,\end{aligned}$$

die gewünschte Abschätzung. □

Korollar A.3.3. *Sei $|X| < \infty$ und die Funktionen $\boldsymbol{\phi}_h, \boldsymbol{\psi}_h \in \mathbf{L}^2(X; \mathbb{R}^3)$ konvergent gegen $\boldsymbol{\phi}_h \to \boldsymbol{\phi}$ sowie $\boldsymbol{\psi}_h \to \boldsymbol{\psi}$ in $\mathbf{L}^2$. Falls gilt $\boldsymbol{\phi} \in \mathbf{L}^\infty$ und die Folge der $\boldsymbol{\psi}_h$ in $\mathbf{L}^\infty$ beschränkt bleibt, dann gilt*

$$\boldsymbol{\phi}_h \times \boldsymbol{\psi}_h \to \boldsymbol{\phi} \times \boldsymbol{\psi} \quad \text{in } \mathbf{L}^2.$$

Beweis.

$$\begin{aligned}\|\boldsymbol{\phi}_h \times \boldsymbol{\psi}_h - \boldsymbol{\phi} \times \boldsymbol{\psi}\|_{\mathbf{L}^2} &\leq \|(\boldsymbol{\phi}_h - \boldsymbol{\phi}) \times \boldsymbol{\psi}_h + \boldsymbol{\phi} \times (\boldsymbol{\psi}_h - \boldsymbol{\psi})\|_{\mathbf{L}^2} \\ &\leq \|(\boldsymbol{\phi}_h - \boldsymbol{\phi}) \times \boldsymbol{\psi}_h\|_{\mathbf{L}^2} + \|\boldsymbol{\phi} \times (\boldsymbol{\psi}_h - \boldsymbol{\psi})\|_{\mathbf{L}^2} \\ &\leq \|\boldsymbol{\phi}_h - \boldsymbol{\phi}\|_{\mathbf{L}^2} \|\boldsymbol{\psi}_h\|_{\mathbf{L}^\infty} + \|\boldsymbol{\psi}_h - \boldsymbol{\psi}\|_{\mathbf{L}^2} \|\boldsymbol{\phi}\|_{\mathbf{L}^\infty}.\end{aligned}$$

Hier haben wir lediglich die Dreiecksungleichung und Lemma A.3.2 verwendet. Die Konvergenz ergibt sich nun aus den geforderten Eigenschaften der Funktionen. □

Lemma A.3.4. *Es gibt ein $C_{A.3.4}$ sodass für alle $\boldsymbol{\phi}_h, \boldsymbol{\psi}_h \in S^1(\Omega)$ gilt*

$$\|D^2(\boldsymbol{\phi}_h \times \boldsymbol{\psi}_h)\|_{\mathbf{L}^2} \leq C_{A.3.4} \|\nabla \boldsymbol{\phi}_h\|_{\mathbf{L}^2} \|\nabla \boldsymbol{\psi}_h\|_{\mathbf{L}^\infty}.$$

Beweis. Wir benützen einerseits die Äquivalenz zur Norm aus Bemerkung 2.2.6 und andererseits die Produktregel, die sich auf $\times$ als bilineare Funktion anwenden lässt.

$$\begin{aligned}\|D^2(\boldsymbol{\phi}_h \times \boldsymbol{\psi}_h)\|_{\mathbf{L}^2}^2 &\lesssim \sum_{\substack{|\alpha|=2\\ \alpha\in\mathbb{N}_0^3}} \|\partial^\alpha(\boldsymbol{\phi}_h \times \boldsymbol{\psi}_h)\|_{\mathbf{L}^2}^2 \\ &= \sum_{\substack{|\alpha|=2\\ \alpha\in\mathbb{N}_0^3}} \Big\|\sum_{\beta\le\alpha} \binom{\alpha}{\beta} \partial^{\alpha-\beta}\boldsymbol{\phi}_h \times \partial^\beta \boldsymbol{\psi}_h\Big\|_{\mathbf{L}^2}^2.\end{aligned}$$

Hier definieren wir $\binom{\alpha}{\beta} := \prod_{i=1}^n \binom{\alpha_i}{\beta_i}$ und $\alpha \le \beta$ falls $\alpha_i \le \beta_i$ für alle $i = 1, \dots, n$ gilt. Nun gilt aber für alle $\mathbf{u}_h \in S^1(\Omega)$, dass $\partial^\gamma \mathbf{u}_h = 0$ für $|\gamma| > 1$. Also erhalten wir

$$\begin{aligned}\|D^2(\boldsymbol{\phi}_h \times \boldsymbol{\psi}_h)\|_{\mathbf{L}^2}^2 \lesssim \|&\partial^x\boldsymbol{\phi}_h \times \partial^x\boldsymbol{\psi}_h + \partial^x\boldsymbol{\phi}_h \times \partial^y\boldsymbol{\psi}_h + \partial^x\boldsymbol{\phi}_h \times \partial^z\boldsymbol{\psi}_h \\ &+\partial^y\boldsymbol{\phi}_h \times \partial^x\boldsymbol{\psi}_h + \partial^y\boldsymbol{\phi}_h \times \partial^y\boldsymbol{\psi}_h + \partial^y\boldsymbol{\phi}_h \times \partial^z\boldsymbol{\psi}_h \\ &+\partial^z\boldsymbol{\phi}_h \times \partial^x\boldsymbol{\psi}_h + \partial^z\boldsymbol{\phi}_h \times \partial^y\boldsymbol{\psi}_h + \partial^z\boldsymbol{\phi}_h \times \partial^z\boldsymbol{\psi}_h\|_{\mathbf{L}^2}^2\end{aligned}$$

Und somit das gewünschte Ergebnis

$$\begin{aligned}\|D^2(\boldsymbol{\phi}_h \times \boldsymbol{\psi}_h)\|_{\mathbf{L}^2}^2 &\lesssim \|(\partial^x\boldsymbol{\phi}_h + \partial^y\boldsymbol{\phi}_h + \partial^z\boldsymbol{\phi}_h) \times (\partial^x\boldsymbol{\psi}_h + \partial^y\boldsymbol{\psi}_h + \partial^z\boldsymbol{\psi}_h)\|_{\mathbf{L}^2}^2 \\ &\lesssim \|\partial^x\boldsymbol{\phi}_h + \partial^y\boldsymbol{\phi}_h + \partial^z\boldsymbol{\phi}_h\|_{\mathbf{L}^2}^2 \|\partial^x\boldsymbol{\psi}_h + \partial^y\boldsymbol{\psi}_h + \partial^z\boldsymbol{\psi}_h\|_{\mathbf{L}^\infty}^2 \\ &\lesssim (\|\partial^x\boldsymbol{\phi}_h\|_{\mathbf{L}^2}^2 + \|\partial^y\boldsymbol{\phi}_h\|_{\mathbf{L}^2}^2 + \|\partial^z\boldsymbol{\phi}_h\|_{\mathbf{L}^2}^2) \\ &\qquad \max(\|\partial^x\boldsymbol{\psi}_h\|_{\mathbf{L}^\infty}, \|\partial^y\boldsymbol{\psi}_h\|_{\mathbf{L}^\infty}, \|\partial^z\boldsymbol{\psi}_h\|_{\mathbf{L}^\infty})^2 \\ &\lesssim \|\nabla\boldsymbol{\phi}_h\|_{\mathbf{L}^2}^2 \|\nabla\boldsymbol{\psi}_h\|_{\mathbf{L}^\infty}^2.\end{aligned}$$

Wobei wir hier neben diversen Normabschätzungen auch Lemma A.3.2 verwendet haben. □

Lemma A.3.5 (Youngsche Ungleichung mit ε)**.** *Für $a, b, \varepsilon > 0$ gilt*

$$ab \le \varepsilon a^2 + \frac{b^2}{4\varepsilon}$$

Beweis. Wende die Youngsche Ungleichung $xy \le \frac{x^2}{2} + \frac{y^2}{2}$ auf

$$ab = ((2\varepsilon)^{1/2}a)\big(\frac{b}{(2\varepsilon)^{1/2}}\big)$$

an. □

A.4. Funktionalanalysis

Satz A.4.1. *[Wer00, Theorem III.3.7] In einem reflexiven Raum besitzt jede beschränkte Folge eine schwach konvergente Teilfolge.*

Korollar A.4.2. *Da jeder Hilbertraum reflexiv ist, besitzt also jede beschränkte Folge in einem Hilbertraum eine schwach konvergente Teilfolge. Insbesondere gilt das für $\mathbf{H}^1(\Omega_T)$, sowie $\mathbf{L}^2(\Omega_T)$ und $\mathbf{L}^2([0,T], \mathbf{H}^1(\Omega))$.*

Das folgende Lemma folgt aus dem Satz von Banach-Steinhaus.

Lemma A.4.3. *[Wer00, Korollar IV.2.3] Schwach konvergente Folgen sind beschränkt.*

Lemma A.4.4. *Seien f_n und g_n Folgen in einem Hilbertraum X mit $f_n \rightharpoonup f$ und $g_n \to g$ in X, dann gilt $(f_n, g_n) \to (f, g)$.*

Beweis. Durch geeignetes Umformen erhalt man mittels folgender Ungleichungskette die Konvergenz.

$$\begin{aligned} |(f_n, g_n) - (f, g)| &= |(f_n, g - g_n) - (f_n - f, g)| \\ &\leq |(f_n, g - g_n)| + |(f_n - f, g)| \\ &\leq \|f_n\|\|g - g_n\| + |(f_n - f, g)| \to 0 \end{aligned}$$

Dabei geht der erste Summand gegen Null, weil f_n als schwach konvergente Folge normbeschränkt ist. Der zweite Summand geht wegen der schwachen Konvergenz von $f_n - f$ gegen Null. □

Satz A.4.5 (Rellichscher Einbettungssatz)**.** *[McL00, Theorem 3.27] Sei $\Omega \subseteq \mathbb{R}^n$ offen und beschränkt mit Lipschitzrand. Dann ist die Abbildung*

$$\mathrm{Id} : H^1(\Omega) \hookrightarrow L^2(\Omega)$$

ein linearer kompakter Operator.

Lemma A.4.6. *[Wer00, III.5.9] Sei X ein normierter Raum, $f : X \to \mathbb{R}$ konvex, stetig und $x_n \rightharpoonup x$ in X, dann gilt*

$$f(x) \leq \liminf_{n\to\infty} f(x_n).$$

Lemma A.4.7 (Lemma von Lax-Milgram)**.** *[Wer00, V.6.15] Sei H ein komplexer Hilbertraum mit Skalarprodukt $\langle ., . \rangle$ und sei $B : H \times H \to \mathbb{C}$ sequilinear. Falls B stetig ist, existiert $T \in L(H)$ mit*

$$B(x, y) = \langle Tx, y \rangle \quad \forall x, y \in H.$$

Ist B zusätzlich koerzitiv, d.h.

$$\exists C > 0 \, \forall x \in H \quad B(x, x) \geq C\|x\|^2,$$

so ist T invertierbar mit $\|T^{-1}\| \leq 1/C$.

A.5. FEM

Satz A.5.1 (Approximationssatz)**.** *[Bra07, II.6.4] Sei $\mathcal{T}_h$ eine quasi-uniforme (dreidimensionale) Triangulierung. Für eine Funktion $u \in H^2(\Omega) \subseteq C(\overline{\Omega})$ und deren nodalen Interpolanten $u_h := \mathcal{I}_h(u)$ gilt*

$$\|h^\alpha (u - u_h)\|_{\mathbf{L}^2(\Omega)} \leq C_{\mathrm{approx}} \|h^{2+\alpha} D^2 u\|_{\mathbf{L}^2(\Omega)} \tag{A.4}$$

und

$$\|h^\alpha \nabla(u - u_h)\|_{\mathbf{L}^2(\Omega)} \leq C_{\mathrm{approx}} C_{\mathrm{form}} \|h^{1+\alpha} D^2 u\|_{\mathbf{L}^2(\Omega)} \tag{A.5}$$

mit einer von u, $\mathcal{T}_h$ und Ω unabhängigen Konstante $C_{\mathrm{approx}} > 0$ und der Formregularitätskonstante C_{form}.

Satz A.5.2. *[Gol12, Lemma 2.2.5] Es gilt*

$$\left| \int_\Omega \mathcal{I}_h(\langle \boldsymbol{\phi}_h, \boldsymbol{\psi}_h \rangle) d\mathbf{x} - \int_\Omega \langle \boldsymbol{\phi}_h, \boldsymbol{\psi}_h \rangle d\mathbf{x} \right| \leq C_{A.5.2} h \|\boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega)} \|\nabla \boldsymbol{\psi}_h\|_{\mathbf{L}^2(\Omega)}$$

für alle Funktionen $\boldsymbol{\phi}_h, \boldsymbol{\psi}_h \in S^1(\mathcal{T}_h)$

Satz A.5.3 (Inverse Abschätzung)**.** *[BS02, Theorem 4.5.11] Sei* $(\mathcal{T}_h)_h$, $0 < h \leq 1$, *eine quasi-uniforme Familie von regulären Triangulierungen eines Gebietes* $\Omega \subseteq \mathbb{R}^n$. *Sei weiters* $0 \leq m \leq \ell$ *und* $k \in \mathbb{N}$. *Dann gibt es ein* $C_{A.5.3} > 0$, *sodass für alle* h *gilt*

$$\|u\|_{H^\ell(\Omega)} \leq C_{A.5.3} h^{m-\ell} \|u\|_{H^m(\Omega)} \text{ für alle } u \in S^k(\mathcal{T}_h; \mathbb{R}).$$

Korollar A.5.4. *Sei* $(\mathcal{T}_h)_h$, $0 < h \leq 1$, *eine quasi-uniforme Familie von regulären Triangulierungen eines Gebietes* $\Omega \subseteq \mathbb{R}^n$. *Dann gibt es ein* $C_{A.5.4} > 0$, *sodass für alle* h *gilt*

$$\|D\boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega;\mathbb{R}^{n\times d})} \leq C_{A.5.4} h^{-1} \|\boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega;\mathbb{R}^d)} \text{ für alle } \boldsymbol{\phi}_h \in S^k(\mathcal{T}_h; \mathbb{R}^d)$$

Beweis. Wir wenden Satz A.5.3 mit $m = 0$ und $\ell = 1$ komponentenweise an schätzen die $\mathbf{H}^1$-Norm nach unten durch die H^1-Seminorm ab. Durch die Äquivalenz der $\mathbf{H}^1$-Seminorm zu $\|D\boldsymbol{\phi}_h\|_{\mathbf{L}^2(\Omega;\mathbb{R}^{n\times d})}$ erhalten wir das gewünschte Resultat. □

Korollar A.5.5. *Sei* $(\mathcal{T}_h)_h$, $0 < h \leq 1$, *eine quasi-uniforme Familie von regulären Triangulierungen eines Gebietes* $\Omega \subseteq \mathbb{R}^n$. *Dann gibt es ein* $C_{A.5.4} > 0$, *sodass für alle* h *gilt*

$$\|D^k\boldsymbol{\phi}_h\|_{\mathbf{L}^2} \leq C_{A.5.4}^k h^{-k} \|\boldsymbol{\phi}_h\|_{\mathbf{L}^2} \text{ für alle } \boldsymbol{\phi}_h \in S^k(\mathcal{T}_h; \mathbb{R}^n)$$

Beweis. Induktive Anwendung von Korollar A.5.4 liefert das gewünschte Resultat. □

A.6. Maßtheorie

Satz A.6.1 (Satz von der majorisierten Konvergenz)**.** *[Els96, Satz 5.2] Seien* $f, f_n : X \to \mathbb{R} \cup \{\pm\infty\}$ *messbare Funktionen, und es gelte* $\lim_{n\to\infty} f_n = f$ *fast überall. Ferner gebe es eine integrierbare, messbare nichtnegative Funktion* g, *so dass für alle* $n \in \mathbb{N}$ *gilt* $|f_n| \leq g$ *fast überall. Dann sind* f *und alle* f_n *integrierbar, und es gilt*

$$\lim_{n\to\infty} \int_X f_n d\mathbf{x} = \int_X f d\mathbf{x} \tag{A.6}$$

und

$$\lim_{n\to\infty} \int_X |f_n - f| d\mathbf{x} = 0. \tag{A.7}$$

Satz A.6.2. *[Els96, Satz 4.4] Sind* $f, g : X \to \mathbb{R}$ *integrierbar und*

$$\int_B f \, d\mathbf{x} \leq \int_B g \, d\mathbf{x} \quad \text{für alle messbaren } B,$$

so ist $f \leq g$ *fast überall. Gilt insbesondere Gleichheit, so ist* $f = g$ *fast überall.*

Satz A.6.3 (Höldersche Ungleichung)**.** *[Els96, 1.5, S. 222] Es seien* $1 \leq p, q \leq \infty$, $\frac{1}{p} + \frac{1}{q} = 1$, *wobei* $1/\infty := 0$, *und* $f, g : X \to \hat{\mathbb{K}}$ *messbar. Dann gilt:*

$$\|fg\|_{L^1} \leq \|f\|_{L^p} \|g\|_{L^q} \tag{A.8}$$

A.7. Integralsätze

Satz A.7.1 (Gaußscher Integralsatz)**.** *Sei* $\boldsymbol{\phi} : \overline{\Omega} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ *stetig differenzierbar auf dem beschränkten Gebiet* Ω *mit stückweise glattem Rand* Γ *und* $\mathbf{n}$ *die nach außen gerichtete Normale. Dann gilt*

$$\int_\Omega \operatorname{div} \boldsymbol{\phi}\, d\mathbf{x} = \int_\Gamma \langle \boldsymbol{\phi}, \mathbf{n} \rangle d\mu.$$

Satz A.7.2 (Partielle Integration im $\mathbb{R}^n$)**.** *Sei* $\Omega \subseteq \mathbb{R}^n$ *ein beschränktes Gebiet mit stückweise glattem Rand* Γ *und* $\mathbf{n}$ *die nach außen gerichtete Normale. Seien weiters* $u : \Omega \to \mathbb{R}$ *und* $\boldsymbol{\psi} : \Omega \to \mathbb{R}^n$ *stetig differenzierbar. Dann gilt*

$$\int_\Omega u \operatorname{div} \boldsymbol{\psi}\, d\mathbf{x} = \int_\Gamma u \langle \boldsymbol{\psi}, \mathbf{n} \rangle d\mu - \int_\Omega \langle \nabla u, \boldsymbol{\psi} \rangle d\mathbf{x}.$$

Beweis. Dies ist ein Spezialfall das Gaußschen Integralsatzes A.7.1 für die Wahl $\boldsymbol{\phi} := u\boldsymbol{\psi}$, dessen Darstellung man mit der komponentenweisen Anwendung der Produktregel erhält. □

Unmittelbar durch die Definition der schwachen Ableitung ergibt sich auch eine Formel zur partiellen Integration von Funktionen aus $H^1(\Omega; \mathbb{R})$ analog zu Satz A.7.2

Korollar A.7.3. *Sei* $\Omega \subseteq \mathbb{R}^n$ *ein Gebiet. Seien weiters* $u \in H^1(\Omega; \mathbb{R})$ *und* $\boldsymbol{\phi} \in C_c^\infty(\Omega; \mathbb{R}^3)$*. Dann gilt*

$$\int_\Omega u \operatorname{div} \boldsymbol{\phi}\, d\mathbf{x} = -\int_\Omega \langle \nabla u, \boldsymbol{\phi} \rangle d\mathbf{x}.$$

Beweis.

$$\begin{aligned}\int_\Omega u \operatorname{div} \boldsymbol{\phi}\, d\mathbf{x} = \int_\Omega u \sum_{j=1}^n \frac{\partial \boldsymbol{\phi}_j}{\partial x_j}\, d\mathbf{x} &= \sum_{j=1}^n \int_\Omega u \frac{\partial \boldsymbol{\phi}_j}{\partial x_j}\, d\mathbf{x} \\ &= \sum_{j=1}^n -\int_\Omega \frac{\partial u}{\partial x_j} \boldsymbol{\phi}_j\, d\mathbf{x} = -\int_\Omega \sum_{j=1}^n \frac{\partial u}{\partial x_j} \boldsymbol{\phi}_j\, d\mathbf{x} = -\int_\Omega \langle \nabla u, \boldsymbol{\phi} \rangle d\mathbf{x}.\end{aligned}$$

Wobei wir lediglich die Defintion der schwachen Ableitung 2.2.1 verwendet haben. □

B Weitere Codes

In diesem Abschnitt sind alle weiteren Codes angehängt, die zur Simulation der LLG-Gleichung entwickelt wurden. Für die Berechnung des Randintegraloperators, der für die Berechnung des Streufelds nötig ist wird auf die Diplomarbeit von Markus Mayr[May13] zurückgegriffen.

B.1. Integration and Differentiation

Listing B.1: Gradienten des Netzes

```
function hatGrads = getHatGrads(mesh)
%GETHATGRADS    Gradients of the hat functions.
%   HATGRADS = GETHATGRADS(MESH) returns the gradients of the hat functions
%   on the simplicial mesh MESH, which is a structure containing the
%   arrays ELEMENTS and COORDINATES. These represent the mesh by standard
%   simplex-vertex format. HATGRADS is a (DIM+1)-cell array, so that
%   HATGRADS{j}(i,:) is the gradient of the hat function corresponding to
%   the node MESH.ELEMENTS(i,j) on the element MESH.ELEMENTS(i,:).
%
%   Author: Josef Kemetmueller - 16.12.2013
if isfield(mesh,'hatGrads')
    hatGrads = mesh.hatGrads;
    return;
end
dimMesh = size(mesh.elements,2)-1;
dimSpace = size(mesh.coordinates,2);

assert(dimMesh<=dimSpace, ...
                ['Your %dD-simplices are lying in %dD-space.' ...
                'I''m not sure there even is a ''gradient''.'], ...
                dimMesh, dimSpace);
assert(dimSpace<=3, ...
                'Dimension %d is larger than the implementation limit 3.', ...
               dimSpace);

[X,hatGrads] = deal(cell(1,dimMesh+1));
for d = 1:dimMesh+1
    X{d} = mesh.coordinates(mesh.elements(:,d),:);
end
% "Modulo operation" m(i)=^=mod(i,dimMesh+1) for small i,
```

```
%  with 0 =^= dimMesh+1.
mo = [1:dimMesh+1, 1:dimMesh+1];
switch dimMesh
    case 1
        divideByLengthSquared = @(X) bsxfun(@rdivide,X,dot(X,X,2));
        for i = 1:dimMesh+1
            hatGrads{i} = divideByLengthSquared(X{mo(i+1)}-X{mo(i)});
        end
    case 2
        switch dimSpace
            % orth(Xa,Xb) generates a vector orthogonal to the edge
            % [Xa,Xb] lying inside the triangle.
            case 2 %2D mesh embedded in R^2
                orth = @(Xa,Xb) [-Xa(:,2)+Xb(:,2), Xa(:,1)-Xb(:,1)];
            case 3 %2D mesh embedded in R^3
                N_3D = cross(X{3}-X{1},X{2}-X{1},2);
                orth = @(Xa,Xb) cross(N_3D,Xa-Xb,2);
        end
        for i = 1:dimMesh+1
            Ni = orth(X{mo(i+2)},X{mo(i+1)});
            hatGrads{i} = bsxfun(@rdivide,Ni,dot(Ni,X{i}-X{mo(i+1)},2));
        end
    case 3
        % Gradient has the same direction as normal vector of opposite
        % face. The length is 1/(length point to opposite face).
        for i = 1:dimMesh+1
            Ni = cross(X{mo(i+2)}-X{mo(i+1)}, X{mo(i+3)}-X{mo(i+1)},2);
            hatGrads{i} = bsxfun(@rdivide,Ni,dot(Ni,X{i}-X{mo(i+1)},2));
        end
end
end
```

Listing B.2: Gradient einer stückweise linearen Funktion

```
function P0 = gradP1(mesh, P1)
%GRADP1    Gradient of elementwise linear, globally continuous function.
%   P0 = GRADP1(mesh, P1) returns the gradient of the function P1
%   represented via the nodal basis as a nC-by-dimP1 matrix, where nC is
%   the number of nodes in the mesh and dimP1 is the dimension of the
%   codomain of the function P1. P0 is a nE-by-dimSpace*dimP1 array containing
%   the gradients on all elements.
%
%   JacTr = reshape(P0(i,:), dimMesh, dimP1) is the transposed
%   jacobian-matrix on the i-th element.
%
%   Author: Josef Kemetmueller - 16.12.2013
nC = size(mesh.coordinates,1);
nE = size(mesh.elements,1);
dimMesh = size(mesh.elements,2)-1;
dimP1 = size(P1,2);
dimSpace = size(mesh.coordinates,2);
%
assert(size(P1,1)==nC, 'Function must be given as a P1 function');
%
P0 = zeros(nE,dimSpace,dimP1);
%%
hatGrads = getHatGrads(mesh);
for i = 1:dimP1
```

```
25      for j = 1:dimMesh+1
26          % P0(T,:,i) is the gradient of the i-th component of P1 on element T.
27          P0(:,:,i) = P0(:,:,i) + bsxfun(@times, hatGrads{j}, ...
28                                                 P1(mesh.elements(:,j),i));
29      end
30  end
31  P0 = reshape(P0,nE,[]);
32  end
```

Listing B.3: Integrationsroutine für allgemeine Funktionen

```
1   function integrals = integrate(mesh, f, quadDeg, varargin)
2   %INTEGRATE    Integrates a general function given as a function handle.
3   %   INTEGRAL = INTEGRATE(MESH, f, quadDeg) returns the integral of the
4   %   function f which is given by a function handle over the given mesh,
5   %   which is a structure containing the arrays ELEMENTS and COORDINATES.
6   %   These represent the mesh by standard simplex-vertex format. quadDeg is
7   %   an integer representing the number of quadrature points used in the
8   %   underlying gaussian quadrature.
9   %
10  %   INTEGRALS = INTEGRATE(MESH, f, quadDeg, 'elementwise') returns a
11  %   nE-by-dimf array containing the elementwise integrals.
12  %
13  %   INTEGRALS = INTEGRATE(mesh, f, quadDeg, 'patchwise') returns
14  %   nC-by-dim array containing the patchwise integrals.
15  %
16  %   WARNING: The function f must be given in a manner, that n points can
17  %   be evaluated simultaneously given as a n-by-dimSpace matrix.
18  %   If you don't have a vectorized version of the function f it can be
19  %   integrated using the [slower] option 'for':
20  %
21  %   INTEGRALS = INTEGRATE(..., 'for')
22  %
23  %   Author: Josef Kemetmueller - 16.12.2013
24
25  dimf = size(f(mesh.coordinates(1,:)),2);
26  assert(size(f(mesh.coordinates(1,:)),1)==1,'f values must be row vectors.');
27
28  if any(strcmpi('for',varargin))
29      f = @(X) manualVectorization(f,X,dimf);
30  else
31      assert(size(f([mesh.coordinates(1,:);mesh.coordinates(1,:)]),1) == 2, ...
32                              'Function not vectorized. Use ''for'' option.');
33  end
34
35  dimMesh = size(mesh.elements,2)-1;
36  nE = size(mesh.elements,1);
37  nC = size(mesh.coordinates,1);
38  %%
39  integral_elements = zeros(nE,dimf);
40  [W, V] = simplexQuadratureRule(dimMesh, quadDeg);
41  volumes = getElementVolumes(mesh);
42  nQ = length(W);
43  for j = 1:nQ
44      XYZ = barycentricToCartesians(mesh,V(j,:));
45      fXYZ = f(XYZ);
46      for d = 1:dimf
47          integral_elements(:,d) = integral_elements(:,d) ...
```

```
                                        + W(j)*volumes.*fXYZ(:,d);
    end
end

if (isempty(varargin) || strcmpi(varargin{1},'for'))
    integrals = sum(integral_elements,1);
else
    switch lower(varargin{1})
        case {'elementwise'}
            integrals = integral_elements;
        case {'patchwise'}
            integrals = zeros(nC,dimf);
            for d = 1:dimf
                for node = 1:dimMesh+1
                    integrals(:,d) = integrals(:,d) + ...
                                     accumarray(mesh.elements(:,node), ...
                                                integral_elements(:,d),[nC,1]);
                end
            end
        otherwise
            error(['Invalid optional argument: ''', ...
                    varargin{1},'''.']);
    end
end
end

function Y = manualVectorization(f,X,dimf)
% If f is not vectorized, we use a loop to calculate the values.
Y = zeros(size(X,1), dimf);
for i = 1:size(X,1)
    Y(i,:) = f(X(i,:));
end
end
```

Listing B.4: Integrationsroutine für stückweise konstante Funktionen

```
function integrals = integrateP0(mesh, P0, varargin)
%INTEGRATEP0    Integrates a piecewise constant function
%   INTEGRAL = INTEGRATEP0(MESH, P0) returns the integral of the function
%   P0 which is given by a nE-by-dimP0 matrix over the given mesh,
%   which is a structure containing the arrays ELEMENTS and COORDINATES.
%   These represent the mesh by standard simplex-vertex format.
%
%   INTEGRALS = INTEGRATEP0(MESH, P0, 'elementwise') returns a nE-by-dimP0
%   array containing the elementwise integrals.
%
%   INTEGRALS = INTEGRATEP0(mesh, P0, 'patchwise') returns a nC-by-dimP0
%   array containing the patchwise integrals.
%
%   Author: Josef Kemetmueller - 16.12.2013

dimP0 = size(P0,2);
dimMesh = size(mesh.elements,2)-1;
nE = size(mesh.elements,1);
nC = size(mesh.coordinates,1);
assert(size(P0,1)==nE, ['Integrand must be given as a P^0 function. ',...
                        '[size(elements,1)==size(P0,1)]']);
if ~isfield(mesh,'volumes')
```

```
23      mesh.volumes = getElementVolumes(mesh);
24  end
25
26  if (isempty(varargin))
27      integrals = mesh.volumes'*P0;
28  else
29      switch lower(varargin{1})
30          case {'elementwise'}
31              integrals = bsxfun(@times, mesh.volumes, P0);
32          case {'patchwise'}
33              integrals = zeros(nC,dimP0);
34              for d = 1:dimP0
35                  for node = 1:dimMesh+1
36                      integrals(:,d) = integrals(:,d) + ...
37                          accumarray(mesh.elements(:,node), ...
38                                      bsxfun(@times,mesh.volumes,P0(:,d)),[nC,1]);
39                  end
40              end
41          otherwise
42              error(['Invalid optional argument: ''', ...
43                      varargin{1},'''.']);
44      end
45  end
46  end
```

Listing B.5: Integrationsroutine für stückweise lineare Funktionen

```
1  function integrals = integrateP1(mesh, P1, varargin)
2  %INTEGRATEP1    Integrates a piecewise linear function
3  %   INTEGRAL = INTEGRATEP1(MESH, P1) returns the integral of the function
4  %   P1 which is given by a nC-by-dimP1 matrix over the given mesh,
5  %   which is a structure containing the arrays ELEMENTS and COORDINATES.
6  %   These represent the mesh by standard simplex-vertex format.
7  %
8  %   INTEGRALS = INTEGRATEP1(MESH, P1, 'elementwise') returns a nE-by-dimP1
9  %   array containing the elementwise integrals.
10 %
11 %   INTEGRALS = INTEGRATEP1(mesh, P1, 'patchwise') returns a nC-by-dimP1
12 %   array containing the patchwise integrals.
13 %
14 %   Author: Josef Kemetmueller - 16.12.2013
15 nC = size(mesh.coordinates,1);
16 assert(size(P1,1)==nC, ['Integrand must be given as a P^1 function. ', ...
17                         '[size(coordinates,1) == size(P1,1)]']);
18 %% Reuse the available functions
19 integrals = integrateP0(mesh, L2ProjectP1ToP0(mesh, P1), varargin{:});
```

Listing B.6: Quadraturregeln per Tensor-Gauss-Quadratur

```
1  function [W, V] = simplexQuadratureRule(dim, n)
2  %SIMPLEXQUADRATURERULE    Quadrature points for simplices.
3  %   [WEIGHTS, POINTS] = SIMPLEXQUADRATURERULE(DIM, QUADDEG) returns
4  %   quadrature weights and points in barycentric coordinates for a simplex
5  %   of dimension DIM<=3 using QUADDEG^DIM quadrature points.
6  %   WEIGHTS is a row vector of quadrature weights corresponding to the rows
7  %   of the matrix POINTS. Which are the quadrature points given in
```

```
%   barycentric coordinates.
%   If a function f can evaluate multiple points at once you can simply use
%       integral = volume*W*f(V)
%   meaning $\int f dx = volume \sum_i w_i f(V_i)$.
%
%   Otherwise use the following code
%       integral = 0;
%       for i = 1:length(W)
%           integral = integral + volume*W(i)*f(V(i,:))
%       end
%   The quadrature rule is exact for polynomials of degree 2*QUADDEG-DIM.
%
%   See also:
%   BARYCENTRICTOCARTESIANS
%
%   Author: Josef Kemetmueller - 16.12.2013

assert((dim>=1 && dim<=3), ['Only quadrature rules for',...
                            '1 <= DIM <= 3 are implemented.']);
% One-point quadrature.
if (isempty(n) || (n == 1))
    V = 1/(dim+1)*ones(1,dim+1);
    W = 1;
    return;
end
% We use tensor quadrature based on 1D-gaussian quadrature and
% integration by substitution via the affine transformation TRAFO and its
% determinant TRAFODET.
[X_1D,W_1D] = gauss1D(n,0,1);
switch dim
    case 1
        W = W_1D;
        V = [X_1D, 1-X_1D];
    case 2
        trafo = @(x,y) [x,(1-x).*y];
        trafodet = @(x,y) (1-x);
        % Tensor quadrature
        X = repmat(X_1D',[n,1]);
        W = repmat(W_1D,[n,1]);
        Y = X';
        W = W.*W';
        % Affine transformation of quadrature rule
        XY = trafo(X(:),Y(:));
        V = [XY,1-sum(XY,2)];
        W = W.*abs(trafodet(X,Y));
    case 3
        trafo = @(x,y,z) [x,(1-x).*y,(1-x).*(1-y).*z];
        trafodet = @(x,y,z) (1-x).*(1-x).*(1-y);
        % Tensor quadrature
        X = repmat(X_1D',[n,1,n]);
        W = repmat(W_1D,[n,1,n]);
        Y = shiftdim(X,1);
        Z = shiftdim(X,2);
        W = W.*shiftdim(W,1).*shiftdim(W,2);
        % Affine transformation of quadrature rule
        XYZ = trafo(X(:),Y(:),Z(:));
        V = [XYZ,1-sum(XYZ,2)];
        W = W.*abs(trafodet(X,Y,Z));
end
```

```
% As the above yields a quadrature rule for the n-dimensional simplex, it
% holds sum(W) = 1/factorial(dim) [= volumeOf(nDimSimplex)]. But as we want
% to use the function via integral = volume*W*f(V) we have to scale so
% that sum(W) == 1.
W = reshape(factorial(dim)*W,1,[]);
end

function [nodes,weights] = gauss1D(n,varargin)
beta = (1:n-1)./sqrt((2*(1:n-1)).^2-1);
A = diag(beta,-1)+diag(beta,1);
[eigenvector,nodes] = eig(A);
[nodes,idx] = sort(diag(nodes));
weights = 2*eigenvector(1,idx).^2;

if nargin >= 3
    a = varargin{1};
    b = varargin{2};
    weights = 0.5*abs(b-a)*weights;
    nodes = 0.5*( a+b + nodes*(b-a) );
end
end
```

B.2. Geometrische Berechnungen

Listing B.7: Baryzentrische Koordinaten in Kartesische Koordinaten umwandeln

```
function cartesians = barycentricToCartesians(mesh, barycentric)
%BARYCENTRICTOCARTESIANS    Single barycentric coordinate to multiple
%cartesian coordinates
%   CARTESIANS = BARYCENTRICTOCARTESIANS(MESH, BARYCENTRIC) yields a
%   nE-by-dimSpace array containing the points obtained using the given
%   barycentric coordinate as local coordinate on all the elements.
%
%   Author: Josef Kemetmueller - 16.12.2013

assert(length(barycentric)==numel(barycentric), ...
                'Barycentric coordinate must be given as a single vector');
dimMesh = size(mesh.elements,2)-1;
dimSpace = size(mesh.coordinates,2);
nE = size(mesh.elements,1);
cartesians = zeros(nE, dimSpace);
for d = 1:dimMesh+1
    cartesians = cartesians + barycentric(d)*mesh.coordinates(mesh.elements(:,d),:);
end
```

Listing B.8: Überflüssige Punkte aus Netz entfernen

```
function [mesh, old2new, new2old] = cleanMesh(mesh)
%CLEANMESH    Removes duplicate and unused points from mesh
%   [CLEANEDMESH, OLD2NEW, NEW2OLD] = ...
%                       CLEANMESH(MESH)
%   The arrays old2new and new2old are used for reindexing in the following
%   way:
%   cleanedmesh.elements == old2new(mesh.elements)
```

```
%   mesh.elements == new2old(cleanedmesh.elements)
%       and
%   cleanedmesh.coordinates == mesh.coordinates(new2old,:)
%
% Author: Josef Kemetmueller - 16.12.13

% Compute remaining elements
remaining = unique(mesh.elements(:));
nRe = size(remaining,1);
nE = size(mesh.elements,1);

old2new = zeros(nE,1);
new2old = zeros(nRe,1);

% Compute new elements
new2old(1:nRe) = remaining;
old2new(remaining) = 1:nRe;

mesh.coordinates = mesh.coordinates(new2old,:);
mesh.elements = reshape(old2new(mesh.elements),nE,[]);

end
```

Listing B.9: Topologischen Rand eines Netzes finden

```
function boundary = getBoundary(mesh, varargin)
%GETBOUNDARY    Returns the topological boundary of the mesh.
%   BOUNDARY = GETBOUNDARY(MESH) returns the topological boundary.
%
%   BOUNDARY = GETBOUNDARY(MESH, 'outwards') or
%   BOUNDARY = GETBOUNDARY(MESH, 'inwards') returns the boundary in a way
%   that the normals are oriented outwards resp. inwards.
%
% Author: Josef Kemetmueller - 16.12.13
% Note: freeBoundary(TR) should do the same - but without orientation.
% TODO: The orientation property should be programmed in a way, that it
% doesn't use the oriented volume, so it would work e.g. for 2D meshes in 3D.

nE = size(mesh.elements,1);
dimMesh = size(mesh.elements,2) - 1;
faceorder = reshape(repmat((dimMesh+1):-1:1,dimMesh,1),dimMesh+1,dimMesh);
% Reorder, so that faceorder contains the correct orientation!
faceorder(1:2:end,:) = faceorder(1:2:end,end:-1:1);

if (nE==0)
    boundary = [];
    return;
end

%% Correct orientation
if (nargin>1)
    assert((size(mesh.coordinates,2) == dimMesh), ...
                        'Dimensions unsuitable for orientation property.');
    switch varargin{1}
        case 'inwards'
            correct = @(X) X<0;
        case 'outwards'
            correct = @(X) X>0;
```

```
        otherwise
            error(['Unknown orientation option. ', ...
                  'Only valid: ''inwards'' or ''outwards'''']);
    end
    signedVolumes = getSignedElementVolumes(mesh);
    orientcorrect = correct(signedVolumes);
    % Change orientation
    mesh.elements(~orientcorrect,[1,2]) = mesh.elements(~orientcorrect,[2,1]);
end

%% Compute hyperfaces of the mesh
hyperfaces = reshape(mesh.elements(:,faceorder),[],dimMesh);
sortedfaces = sort(hyperfaces,2);

[foo,I,J] = unique(sortedfaces,'rows');
boundary = hyperfaces(I(accumarray(J,1)==1),:);
```

Listing B.10: Volumina der Simplices bestimmen

```
function volumes = getElementVolumes(mesh)
%GETELEMENTVOLUMES    Volumes(/areas/lengths) of the simplices in the mesh.
%   VOLUMES = GETELEMENTVOLUMES(MESH) returns a vector containing the
%   volumes(/areas/lengths) of each simplex.
%
%   Author: Josef Kemetmueller - 16.12.2013

nE = size(mesh.elements, 1);
dimMesh = size(mesh.elements,2)-1;
dimSpace = size(mesh.coordinates,2);

assert(dimMesh<=dimSpace,['Your %dD-simplices are lying in %dD-space.\n',...
                          'The only correct ''volume'' is zero.'],...
                                                  dimMesh, dimSpace);

if (dimMesh==dimSpace)

    volumes = abs(getSignedElementVolumes(mesh));
else % Our mesh is a manifold in some higher dimensional space.
    X = @(d) mesh.coordinates(mesh.elements(:,d),:);
    lengths = @(X) sqrt(sum(X.^2,2));
    %%
    if (dimMesh==1) % Lengths are easy
        volumes = lengths(X(2)-X(1));
    elseif(dimMesh==2 && dimSpace==3) % Areas in 3D using cross product.
        volumes = lengths(cross(X(2)-X(1),X(3)-X(1),2))/2;
    else % Using general rule.
        volumes = zeros(nE,1);
        for i = 1:nE
            nodes = mesh.elements(i,:);
            MT = [ones(1,dimMesh+1); mesh.coordinates(nodes,:)'];
            volumes(i) = 1/factorial(dimMesh)*sqrt(det(MT'*MT));
        end
    end
end
```

Listing B.11: Volumina der Knotenpatches bestimmen

```
function patchVols = getPatchVolumes(mesh)
%GETPATCHVOLUMES    Volumes of the node patches.
%   PATCHVOLS = GETPATCHVOLUMES(MESH) returns the vector of the sums of the
%   volumes of the simplices adjacent to each node.
%
%   Author: Josef Kemetmueller - 16.12.2013
nE = size(mesh.elements,1);
% Patch integrals of the constant 1-function.
patchVols = integrateP0(mesh,ones(nE,1),'patchwise');
```

Listing B.12: Orientiertes Volumen bestimmen

```
function volumes = getSignedElementVolumes(mesh)
%GETSIGNEDELEMENTVOLUMES    Signed volumes(/areas/lengths) of the simplices
% in the given mesh.
%   VOLUMES = GETSIGNEDELEMENTVOLUMES(MESH) returns a vector containing the
%   signed volumes(/areas/lengths) of each simplex.
%
%   For this to be well defined the mesh dimension must be the same as the
%   space dimension.
%
%   Author: Josef Kemetmueller - 16.12.2013

nE = size(mesh.elements, 1);
dimMesh = size(mesh.elements,2)-1;
dimSpace = size(mesh.coordinates,2);

assert(dimMesh==dimSpace,['Your %dD-simplices are lying in %dD-space.',...
                          'There is no signed volume.'],dimMesh,dimSpace);

X = cell(1,dimMesh+1);
for d = 1:dimMesh+1
    X{d} = mesh.coordinates(mesh.elements(:,d),:);
end
switch dimMesh
    case 1 % Edge lengths are easy
        volumes = X{2}-X{1};
    case 2 % Manually compute 2D-determinants
        d21 = X{2}-X{1};
        d31 = X{3}-X{1};
        volumes = 1/2*(d21(:,1).*d31(:,2)-d21(:,2).*d31(:,1));
    case 3 % The cross product can be used to compute the oriented volume.
        N{1} = -cross(X{3}-X{2},X{4}-X{2},2);
        volumes = (1/6)*dot(N{1},X{1}-X{2},2);
    otherwise % Generalized simplex volume.
        volumes = zeros(nE,1);
        for i = 1:nE
            nodes = mesh.elements(i,:);
            MT = [ones(1,dimMesh+1); mesh.coordinates(nodes,:)'];
            volumes(i) = 1/factorial(dimMesh)*det(MT);
        end
end
```

B.3. Skalarprodukte

Listing B.13: Steifigkeitsmatrix berechnen

```
function S = inner_gradP1_gradP1_L2(mesh, phi, psi, dimPhi)
%INNER_GRADP1_GRADP1_L2 L2-inner product of the gradients of two
%elementwise linear, globally continuous functions. ['Stiffness matrix']
%   S = inner_gradP1_gradP1_L2(MESH, PHI, PSI) returns the inner product
%   (grad PHI, gradPSI)_L2.
%
%   V = inner_gradP1_gradP1_L2(MESH, PHI) returns the vector V of grad PHI
%   tested with all gradients of hat functions
%   V(i) == inner_gradP1_gradP1_L2(MESH, grad PHI, grad HAT(i))
%
%   M = inner_gradP1_gradP1_L2(MESH) or
%   M = inner_gradP1_gradP1_L2(MESH, [], [], DIMPHI)
%   returns the matrix M(i,j) = inner_gradP1_gradP1_L2(MESH, HAT(i), HAT(j))
%
%   Author: Josef Kemetmueller - 16.12.2013
if ~isfield(mesh,'volumes')
    mesh.volumes = getElementVolumes(mesh);
end

if ~exist('phi','var')
    S = inner_gradP1_gradP1_L2_mat(mesh, 1);
elseif ~exist('psi','var')
    S = inner_gradP1_gradP1_L2(mesh, phi, []);
elseif isempty(phi) && isempty(psi)
    S = inner_gradP1_gradP1_L2_mat(mesh, dimPhi);
elseif isempty(phi)
    S = inner_gradP1_gradP1_L2_mat(mesh, 1)*psi;
elseif isempty(psi)
    S = inner_gradP1_gradP1_L2_mat(mesh, 1)*phi;
elseif exist('dimPhi','var') && strcmpi(dimPhi, 'elementwise')
    S = dot(phi, inner_gradP1_gradP1_L2_mat(mesh, 1)*psi, 2);
else
    S = sum(dot(phi, inner_gradP1_gradP1_L2_mat(mesh, 1)*psi, 2));
end
end


function S = inner_gradP1_gradP1_L2_mat(mesh, dim)
% So called 'stiffness matrix'.
if dim==1 && isfield(mesh,'stiffness')
    S = mesh.stiffness;
    return;
end
nC = size(mesh.coordinates,1);
nE = size(mesh.elements,1);
dimMesh = size(mesh.elements,2)-1;
hatGrads = getHatGrads(mesh);
% (dimMesh+1) hat-functions per Element
%  => nEx(dimMesh+1)x(dimMesh+1) contributions.
[S,I,J] = deal(zeros(nE,dimMesh+1,dimMesh+1));
for i = 1:dimMesh+1
    for j = 1:dimMesh+1
        S(:,i,j) = inner_P0_P0_L2(mesh,hatGrads{i},hatGrads{j},'elementwise');
        I(:,i,j) = mesh.elements(:,i);
        J(:,i,j) = mesh.elements(:,j);
    end
end
```

```
58 %% Compute the matrix
59 if dim==1
60     S = sparse(I(:),J(:),S(:),nC,nC);
61 else
62     S = sparseBlockDiag(I,J,S,nC,nC,dim);
63 end
64 end
```

Listing B.14: $\mathbf{L}^2$-Skalarprodukt einer allgemeinen und einer stückweise linearen Funktion

```
1  function S = inner_L2_P1_L2(mesh, L2, P1, quaddeg, option)
2  %INNER_L2_P1_L2    Approximate L2-inner product of a L2 function and an
3  %elementwise linear, globally continuous function.
4  %   L2 must be given as a function handle, P1 using the nodal basis.
5  %
6  %   S = INNER_L2_P1_L2(MESH, L2, P1) returns the inner product (L2,P1)_L2.
7  %
8  %   V = INNER_L2_P1_L2(MESH, L2) returns the vector V of L2 tested with
9  %   all hat functions V(i) == INNER_L2_P1_L2(MESH, L2, HAT(i))
10 %
11 %   Author: Josef Kemetmueller - 16.12.2013
12 if ~isfield(mesh,'volumes')
13     mesh.volumes = getElementVolumes(mesh);
14 end
15
16 if ~exist('P1','var')
17     S = inner_L2_P1_L2_vec(mesh, L2, 3);
18 elseif isempty(P1)
19     S = inner_L2_P1_L2_vec(mesh,L2,quaddeg);
20 elseif ~exist('quaddeg','var')
21     S = inner_L2_P1_L2(mesh, L2, P1, 3);
22 elseif exist('option','var') && strcmpi(option, 'elementwise')
23     S = dot(inner_L2_P1_L2_vec(mesh, L2,quaddeg),P1, 2);
24 else
25     S = sum(dot(inner_L2_P1_L2_vec(mesh,L2,quaddeg),P1, 2));
26 end
27 end
28
29 function SPs = inner_L2_P1_L2_vec(mesh,L2,quaddeg)
30
31 dimL2 = size(L2(mesh.coordinates(1,:)),2);
32 dimMesh = size(mesh.elements,2)-1;
33 nC = size(mesh.coordinates,1);
34
35 [W, V] = simplexQuadratureRule(dimMesh, quaddeg);
36 nQ = size(V,1);
37 SPs = zeros(nC,dimL2);
38
39 for j = 1:nQ
40     XYZ = barycentricToCartesians(mesh,V(j,:));
41     L2XYZ = L2(XYZ);
42     for d = 1:dimL2
43         for node = 1:dimMesh+1
44             SPs(:,d) = SPs(:,d) + accumarray(mesh.elements(:,node), ...
45                                 W(j)*V(j,node)*mesh.volumes.*L2XYZ(:,d),[nC,1]);
46         end
```

```
47      end
48  end
49
50  end
```

Listing B.15: $\mathbf{L}^2$-Skalarprodukt zweier stückweise konstanter Funktionen

```
1   function S = inner_P0_P0_L2(mesh, P0_1, P0_2, dimPhi)
2   %INNER_P0_P0_L2    L2-inner product of two piecewise constant functions.
3   %   S = INNER_P0_P0_L2(MESH, PHI, PSI) returns inner_product (PHI,PSI)_L2.
4   %
5   %   V = INNER_P0_P0_L2(MESH, PHI) returns the vector V of PHI tested with
6   %   all basis functions V(i) == INNER_P0_P0_L2(MESH, PHI, CHI(i)), where
7   %   CHI(i) is the basis function which is one on the element(i,:) and zero
8   %   everywhere else.
9   %
10  %   M = INNER_P0_P0_L2(MESH) or M = INNER_P0_P0_L2(MESH, [], [], DIMPHI)
11  %   returns the matrix M(i,j) = INNER_P0_P0_L2(MESH, CHI(i), CHI(j)), where
12  %   CHI(i) is again the basis function which is one on the element(i,:) and
13  %   zero everywhere else.
14  %
15  %   Author: Josef Kemetmueller - 16.12.2013
16  if ~isfield(mesh,'volumes')
17      mesh.volumes = getElementVolumes(mesh);
18  end
19
20  if ~exist('P0_1','var')
21      S = inner_P0_P0_L2_mat(mesh, 1);
22  elseif ~exist('P0_2','var')
23      S = inner_P0_P0_L2(mesh, P0_1, []);
24  elseif isempty(P0_1) && isempty(P0_2)
25      S = inner_P0_P0_L2_mat(mesh, dimPhi);
26  elseif isempty(P0_1)
27      S = inner_P0_P0_L2_mat(mesh, 1)*P0_2;
28  elseif isempty(P0_2)
29      S = inner_P0_P0_L2_mat(mesh, 1)*P0_1;
30  elseif exist('dimPhi','var') && strcmpi(dimPhi, 'elementwise')
31      S = dot(P0_1, inner_P0_P0_L2_mat(mesh, 1)*P0_2, 2);
32  else
33      S = sum(dot(P0_1, inner_P0_P0_L2_mat(mesh, 1)*P0_2, 2));
34  end
35  end
36
37  function S = inner_P0_P0_L2_mat(mesh, dim)
38  nE = size(mesh.elements,1);
39  S = spdiags(repmat(mesh.volumes(:),dim,1),0,dim*nE,dim*nE);
40  end
```

Listing B.16: $\mathbf{L}^2$-Skalarprodukt einer stückweise konstanten und einer stückweise linearen Funktion

```
1   function S = inner_P0_P1_L2(mesh, P0, P1, dimPhi)
2   %INNER_P0_P1_L2    L2-inner product of an elementwise constant and an
3   %elementwise linear, globally continuous function.
4   %   S = INNER_P0_P1_L2(MESH, P0, P1) returns the inner product (P0,P1)_L2.
5   %
```

```
6  %   V = INNER_P0_P1_L2(MESH, P0) returns the vector V of P0 tested with
7  %   all hat functions V(i) == INNER_P0_P1_L2(MESH, P0, HAT(i))
8  %
9  %   V = INNER_P0_P1_L2(MESH, [], P1) returns the vector V of P1 tested with
10 %   all basis functions CHI(i): V(i) == INNER_P0_P1_L2(MESH, CHI(i), P1),
11 %   where CHI(i) is the basis function which is one on the element(i,:) and
12 %   zero everywhere else.
13 %
14 %   M = INNER_P0_P1_L2(MESH) or M = INNER_P0_P1_L2(MESH, [], [], DIMPHI)
15 %   returns the matrix M(i,j) = INNER_P0_P1_L2(MESH, CHI(i), HAT(j)).
16 %
17 %   Author: Josef Kemetmueller - 16.12.2013
18 if ~isfield(mesh,'volumes')
19     mesh.volumes = getElementVolumes(mesh);
20 end
21
22 if ~exist('P0','var')
23     S = inner_P0_P1_L2_mat(mesh, 1);
24 elseif ~exist('P1','var')
25     S = inner_P0_P1_L2(mesh, P0, []);
26 elseif isempty(P0) && isempty(P1)
27     S = inner_P0_P1_L2_mat(mesh, dimPhi);
28 elseif isempty(P0)
29     S = inner_P0_P1_L2_mat(mesh, 1)*P1;
30 elseif isempty(P1)
31     S = inner_P0_P1_L2_mat(mesh, 1)'*P0;
32 elseif exist('dimPhi','var') && strcmpi(dimPhi, 'elementwise')
33     S = dot(P0, inner_P0_P1_L2_mat(mesh, 1)*P1, 2);
34 else
35     S = sum(dot(P0, inner_P0_P1_L2_mat(mesh, 1)*P1, 2));
36 end
37 end
38
39 function S = inner_P0_P1_L2_mat(mesh, dim)
40
41 nE = size(mesh.elements,1);
42 nC = size(mesh.coordinates,1);
43
44 dimMesh = size(mesh.elements,2)-1;
45
46 S = repmat(mesh.volumes/(dimMesh+1), 1, dimMesh+1);
47 I = repmat(1:nE, 1, dimMesh+1);
48 J = mesh.elements;
49
50 if dim==1
51     S = sparse(I(:),J(:),S(:),nE,nC);
52 else
53     S = sparseBlockDiag(I,J,S,nE,nC,dim);
54 end
55
56
57 end
```

Listing B.17: L^2-Skalarprodukt zweier stückweise linearer Funktionen

```
1 function S = inner_P1_P1_L2(mesh, phi, psi, dimPhi)
2 %INNER_P1_P1_L2    L2-inner product of two elementwise linear, globally
3 %continuous functions.
```

```
4  %   S = INNER_P1_P1_L2(MESH, PHI, PSI) returns the inner product (PHI,PSI)_L2.
5  %
6  %   V = INNER_P1_P1_L2(MESH, PHI) returns the vector V of PHI tested with
7  %   all hat functions V(i) == INNER_P1_P1_L2(MESH, PHI, HAT(i))
8  %
9  %   M = INNER_P1_P1_L2(MESH) or M = INNER_P1_P1_L2(MESH, [], [], DIMPHI)
10 %   returns the matrix M(i,j) = INNER_P1_P1_L2(MESH, HAT(i), HAT(j))
11 %   [The so called 'mass matrix'].
12 %
13 %   Author: Josef Kemetmueller - 16.12.2013
14 if ~isfield(mesh,'volumes')
15     mesh.volumes = getElementVolumes(mesh);
16 end
17
18 if ~exist('phi','var')
19     S = inner_P1_P1_L2_mat(mesh, 1);
20 elseif ~exist('psi','var')
21     S = inner_P1_P1_L2(mesh, phi, []);
22 elseif isempty(phi) && isempty(psi)
23     S = inner_P1_P1_L2_mat(mesh, dimPhi);
24 elseif isempty(phi)
25     S = inner_P1_P1_L2_mat(mesh, 1)*psi;
26 elseif isempty(psi)
27     S = inner_P1_P1_L2_mat(mesh, 1)*phi;
28 elseif exist('dimPhi','var') && strcmpi(dimPhi, 'elementwise')
29     S = dot(phi, inner_P1_P1_L2_mat(mesh, 1)*psi, 2);
30 else
31     S = sum(dot(phi, inner_P1_P1_L2_mat(mesh, 1)*psi, 2));
32 end
33 end
34
35 function S = inner_P1_P1_L2_mat(mesh,dim)
36 nC = size(mesh.coordinates,1);
37 nE = size(mesh.elements,1);
38 dimMesh = size(mesh.elements,2)-1;
39
40 % (dimMesh+1) hat-functions per Element
41 %  => nEx(dimMesh+1)x(dimMesh+1) contributions.
42 [S,I,J] = deal(zeros(nE,dimMesh+1,dimMesh+1));
43
44 inner_HatI_HatJ_ref = factorial(dimMesh)/factorial(dimMesh+2)* ...
45                                         (eye(dimMesh+1)+ones(dimMesh+1));
46 for i = 1:dimMesh+1
47     for j = 1:dimMesh+1
48         S(:,i,j) = mesh.volumes*inner_HatI_HatJ_ref(i,j);
49         I(:,i,j) = mesh.elements(:,i);
50         J(:,i,j) = mesh.elements(:,j);
51     end
52 end
53
54 %% Compute the matrix
55 if dim==1
56     S = sparse(I(:),J(:),S(:),nC,nC);
57 else
58     S = sparseBlockDiag(I,J,S,nC,nC,dim);
59 end
60
61 end
```

Listing B.18: Bestimmung der Koeffizienten β für das h-Skalarprodukt

```
function betas = getBetas(mesh)
%GETBETAS    Factors for the computation of the reduced (mass lumped) inner
%product (.,.)_h.
%   Usage:
%   BETAS = GETBETAS(MESH)
%
%   It holds BETAS(i)==INTEGRATEP1(mesh, HAT(i)) and using the vector BETAS
%   the reduced inner-product can be computed by
%   $(PHI,PSI)_h = \sum_i betas(i) <PHI(i), PSI(i)>$
%
%   Author: Josef Kemetmueller - 16.12.2013

dimMesh = size(mesh.elements,2)-1;
% The elementwise integral of one hat function is the element volume
% divided by the number of nodes in the element. (For the elements adjacent
% to the node; for the others it is zero)
betas = 1/(dimMesh+1)*getPatchVolumes(mesh);
end
```

Listing B.19: h-Norm

```
function val = norm_P1_h(mesh, P1)
%NORM_P1_H    h-Norm of a elementwise linear globally continuous function.
%   NORM = NORM_P1_H(MESH, P1) returns norm of the function P1 induced by
%   the reduced inner_product (.,.)_h defined by
%   $(phi, psi)_h = \int I_h(<phi,psi>) dx$.
%
%   Author: Josef Kemetmueller - 16.12.2013
val = sqrt(inner_P1_P1_h(mesh, P1, P1));
end
```

B.4. Operatoren

Listing B.20: Diskretisierung einer allgemeinen Funktion

```
function P1 = hTildeL2(mesh, L2, quadDeg)
%HTILDEL2    'Isometry' from ({P1}, (.,{P1})_L2) to ({P1}, (.,{P1})_h)
%   P1 = hTildeL2(mesh, L2) returns the elementwise linear, globally
%   continuous function P1, that satisfies:
%   $(P1, P1s)_h = (L2, P1s)_L2$ for all elementwise linear, globally
%   continuous P1s.
%
%   Author: Josef Kemetmueller - 16.12.2013
if ~exist('quaddeg','var');
    quadDeg = 3;
end
assert(size(L2(mesh.coordinates(1,:)),1)==1,'L2 values must be row vectors.');

SPs = inner_L2_P1_L2(mesh, L2, [], quadDeg);
P1 = bsxfun(@rdivide, SPs, getBetas(mesh));

% This would be an alternative to compute it, but since its a diagonal
% matrix, we do it explicitly:
```

```
%P1 = inner_P1_P1_h(mesh)\inner_L2_P1_L2(mesh, L2, [], quadDeg);
```

Listing B.21: Diskretisierung einer stückweise konstanten Funktion

```
function P1 = hTildeP0(mesh, P0)
%HTILDEP0    'Isometry' from ({P0}, (.,{P1})_L2) to ({P1}, (.,{P1})_h)
%   P1 = hTildeP0(mesh, P0) returns the elementwise linear, globally
%   continuous function P1, that satisfies:
%   $(P1, P1s)_h = (P0, P1s)_L2$ for all elementwise linear, globally
%   continuous P1s.
%
%   Author: Josef Kemetmueller - 16.12.2013

SPs = inner_P0_P1_L2(mesh, P0);
P1 = bsxfun(@rdivide, SPs, getBetas(mesh));

% This would be an alternative to compute it, but since its a diagonal
% matrix, we do it explicitly:
%P1 = inner_P1_P1_h(mesh)\inner_P0_P1_L2(mesh, P0);
```

Listing B.22: Diskretisierung einer stückweise linearen Funktion

```
function P1_h = hTildeP1(mesh, P1_L2)
%HTILDEP1    'Isometry' from ({P1}, (.,{P1})_L2) to ({P1}, (.,{P1})_h)
%   P1_h = hTildeP1(mesh, P1_L2) returns the elementwise linear, globally
%   continuous function P1_h, that satisfies:
%   $(P1_h, P1s)_h = (P1_L2, P1s)_L2$ for all elementwise linear, globally
%   continuous P1s.
%
%   Author: Josef Kemetmueller - 16.12.2013

P1_h = bsxfun(@rdivide, inner_P1_P1_L2(mesh, P1_L2), getBetas(mesh));

% This would be an alternative to compute it, but since its a diagonal
% matrix, we do it explicitly:
%P1 = inner_P1_P1_h(mesh)\inner_P1_P1_L2(mesh, P1_L2);

end
```

Listing B.23: Clement-Interpolation einer allgemeinen Funktion

```
function P1 = interpolateClement(mesh, f, quaddeg)
%INTERPOLATECLEMENT    Clement interpolation of a function handle.
%   P1 = INTERPOLATECLEMENT(MESH, f) returns the elementwise linear,
%   globally continuous function P1, whose nodal-values are the mean values
%   of the corresponding node-patches.
%
%   Author: Josef Kemetmueller - 16.12.2013
if ~exist('quaddeg','var');
    quaddeg = 3;
end
P1 = bsxfun(@rdivide, integrate(mesh,f,quaddeg,'patchwise'), ...
                      getPatchVolumes(mesh));
end
```

Listing B.24: Clement-Interpolation einer stückweise konstanten Funktion

```
function P1 = interpolateClementP0(mesh,P0)
%INTERPOLATECLEMENTP0 Clement interpolation of a P0 function.
%   P1 = INTERPOLATECLEMENTP0(MESH, P0) returns the elementwise linear,
%   globally continuous function P1, whose nodal-values are the mean values
%   of the corresponding node-patches.
%
%   Author: Josef Kemetmueller - 16.12.2013

% Integral mean of the patch
P1 = bsxfun(@rdivide, integrateP0(mesh,P0,'patchwise'), getPatchVolumes(mesh));
end
```

Listing B.25: $\mathbf{L}^2$-Projektion einer allgemeinen auf die stückweise linearen Funktionen

```
function P1 = L2ProjectL2ToP1(mesh, L2)
%L2PROJECTL2TOP1 L2-Projection from P1 to P0.
%   P1 = L2PROJECTP1TOP1(MESH, L2) returns the L2-projection onto the space
%   of elementwise linear, globally continuous functions.
%
%   Author: Josef Kemetmueller - 16.12.2013
P1 = inner_P1_P1_L2(mesh)\inner_L2_P1_L2(mesh, L2);
```

Listing B.26: $\mathbf{L}^2$-Projektion einer stückweise konstanten auf die stückweise linearen Funktionen

```
function P0 = L2ProjectP1ToP0(mesh, P1)
%L2PROJECTP1TOP0 L2-Projection from P1 to P0.
%   P0 = L2PROJECTP1TOP0(MESH, P1) returns the L2-projection onto the space
%   of elementwise constant functions.
%
%   Author: Josef Kemetmueller - 16.12.2013
nE = size(mesh.elements,1);
dimMesh = size(mesh.elements,2)-1;

% As this projection is just the elementwise mean, we compute it explicitly
P0 = sum(reshape(P1(mesh.elements,:)',[],nE,dimMesh+1),3)'/(dimMesh+1);

% This would be an alternative to compute it, but since its a diagonal
% matrix, we do it explicitly:
%P0 = inner_P0_P0_L2(mesh)\inner_P0_P1_L2(mesh, [], P1);
```

Listing B.27: Diskretisierung des Laplace-Operators

```
function laph = laplace_hP1(mesh, P1)
%LAPLACE_hP1 Discrete version of the laplacian.
%   LAPH = LAPLACE_hP1(MESH, P1) returns the discrete (mass lumped)
%   laplacian of the piecewise linear, globally continuous function P1.
%   The result satisfies:
%   (LAPH, P1s)_h = -(grad P1, grad P1s)_L2 for all piecewise linear, globally
%   continuous P1s.
%
%   Author: Josef Kemetmueller - 16.12.2013
```

```
laph = -bsxfun(@rdivide, mesh.stiffness*P1, mesh.betas);
% This would be an alternative to compute it, but since its a diagonal
% matrix, we do it explicitly:
%laph = -inner_P1_P1_h(mesh)\inner_gradP1_gradP1_L2(mesh,P1);
end
```

Listing B.28: Kreuzproduktmatrix $M_{ij} = (\boldsymbol{\phi}_h \times \varphi_j, \varphi_i)_h$

```
function M = inner_P1xHatJ_HatI_h(mesh, P1)
%INNER_P1XHATJ_HATI_H    Cross product matrix using mass lumping.
%   M = INNER_P1XHATJ_HATI_H(MESH, P1) returns the matrix defined by:
%   M(i,j) = (P1 x hat(j) , hat(i))_h, where the hat functions are numbered
%   by hat(nC*(dim-1)+node), dim=1...3, node=1...nC.
%   This results in the following block matrix, which is skew-symmetric.
%
%       [(m(n)xe1,e1)_h, (m(n)xe2,e1)_h, (m(n)xe3,e1)_h ]
%   M = [(m(n)xe1,e2)_h, (m(n)xe2,e2)_h, (m(n)xe3,e2)_h ]
%       [(m(n)xe1,e3)_h, (m(n)xe2,e3)_h, (m(n)xe3,e3)_h ]
%
%   Author: Josef Kemetmueller - 16.12.2013
M = -inner_HatIxHatJ_P1_h(mesh, P1);
```

Listing B.29: Kreuzproduktmatrix $M_{ij} = (\boldsymbol{\phi}_h \times \tilde{\Delta}_h \varphi_j, \varphi_i)_h$

```
function M = inner_P1xlaphHatJ_HatI_h(mesh, P1)
%INNER_P1XLAPHHATJ_HATI_h    Laplace-Cross product matrix using mass lumping.
%   M = INNER_P1XLAPHHATJ_HATI_h(MESH, P1) returns the matrix defined by:
%   M(i,j) = (P1 x lap_h(hat(j)) , hat(i))_h, where the hat functions are
%   numbered by hat(nC*(dim-1)+node), dim=1...3, node=1...nC.
%
%   Author: Josef Kemetmueller - 16.12.2013
M = -inner_HatIxlaphHatJ_P1_h(mesh, P1);
```

Listing B.30: Vektor $(\nabla \varphi_i, \boldsymbol{\phi}_h)_{\mathbf{L}^2}$

```
function P1TGradHat = inner_GradHatI_P1_L2(mesh, P1)
%INNER_GRADHATI_P1_L2    Elementwise linear function tested by hat gradients.
%   V = INNER_GRADHATI_P1_L2(mesh, P1) returns the vector of P1 tested by
%   all gradients of hat functions
%   V(i) == (P1, grad hat(i))_L2
%
%   Author: Josef Kemetmueller - 16.12.2013
dimMesh = size(mesh.elements,2)-1;
nC = size(mesh.coordinates,1);
P1TGradHat = zeros(nC,1);
hatGrads = getHatGrads(mesh);
for node = 1:dimMesh+1
    P1TGradHat = P1TGradHat + accumarray(mesh.elements(:,node), ...
        inner_P0_P1_L2(mesh, hatGrads{node}, P1, 'elementwise'), [nC,1]);
end
```

Listing B.31: Blockdiagonalmatrix aus dünnbesetzten Matrizen

```
function S = sparseBlockDiag(I,J,S,m,n,repnum)
%SPARSEBLOCKDIAG Sparse block diagonal matrix.
%   S = SPARSEBLOCKDIAG(I,J,S,m,n,repnum) produces a block diagonal matrix,
%   with repnum diagonal blocks consisting of the matrix SPARSE(I,J,S,m,n)
%
%   Author: Josef Kemetmueller - 16.12.2013
ILocal2Global = @(d,nodes) bsxfun(@plus,m*(d-1),reshape(nodes,[],1));
JLocal2Global = @(d,nodes) bsxfun(@plus,n*(d-1),reshape(nodes,[],1));

S = repmat(S(:),1,repnum);
I = ILocal2Global(1:repnum,I);
J = JLocal2Global(1:repnum,J);

S = sparse(I,J,S,m*repnum,n*repnum);

end
```

B.5. Effektives Feld

Listing B.32: Berechnung des Anisotropie-Feldbeitrags

```
function ani = anisotropy(m,material)
%ANISOTROPY    Computes the anisotropy DPhi(m).
%   ANI = ANISOTROPY(M, MATERIAL) returns the anisotropy DPhi(M) = -2(e,x)e
%   for given magnetization m and struct MATERIAL containing the easyaxis
%   MATERIAL.easyaxis
%
%   Author: Josef Kemetmueller - 16.12.2013
e = reshape(material.easyaxis,3,[]);
ani = -2*bsxfun(@times,e',m*e);
```

Listing B.33: Erstellung eines diskreten externen Feldes, konstant in der Zeit

```
function f = constantInTimeExternalFieldTilde(mesh, f_Space, quadDeg)
%CONSTANTINTIMEEXTERNALFIELDTILDE    Generate function handle which is
%constant in time.
%   F = CONSTANTINTIMEEXTERNALFIELDTILDE(MESH, F_SPACE, QUADDEG) returns a
%   function handle f(t), which yields a discrete P1 field for each
%   time t. The input should be a function handle F_SPACE(X) which can be
%   evaluated using an array whose rows are points in R^3.
%
%   Author: Josef Kemetmueller - 16.12.2013

if ~exist('quadDeg','var')
    quadDeg = 3;
end
timeConstant = hTildeL2(mesh, f_Space, quadDeg);
f = @(t) timeConstant;
end
```

Listing B.34: Berechnung des Streufeld-Feldbeitrags

```
function minusGRADu = strayfield(mesh, m)
%STRAYFIELD    Computes the strayfield.
%   MINUSGRADU = STRAYFIELD(MESH, M) returns the strayfield computed via
%   the Fredkin-Koehler FEM-BEM-coupling ansatz. The result is a P0
%   function, namely minus the gradient of the magnetostatic potential U.
%   [-gradP1(MESH, U)]
%
%   Author: Josef Kemetmueller - 16.12.2013
nC = size(mesh.coordinates,1);
dimMesh = size(mesh.elements,2)-1; % Should always be 3...
h = mesh.volumes(1)^(1/dimMesh);
%% STEP 1: compute (\nabla u1, \nabla v_h) = (M,\nabla v_h)
%  ------  for all v_h in S^1_*(T_h)
% compute P1-FEM approx. of Neumann problem by solving extended system,
% where the extra row and column results in a lagrange multiplier, so for
% the solution holds \int_(T_h) x1 = 0.

% We scale the lagrange multiplier, so the matrix-condition is ok.
EA = [mesh.stiffness, h^(-2)*mesh.betas; h^(-2)*mesh.betas' 0];
% assembe right-hand side via (grad(V_j),m)_L2
Eb = [inner_GradHatI_P1_L2(mesh, m); 0];
Ex1 = EA \ Eb;

x1 = Ex1(1:end-1,1);
%% STEP 2: compute Dirichlet data g_h = J_h(K - 1/2)u1|_Gamma
%  ------
u1 = x1(mesh.bd.nodesOf3DMesh);
% use 'double layer' potential operator K-1/2
gh_P0 = mesh.bd.DLPot*u1;
% get P1 approximation by Clement-interpolation on boundary
g_h = interpolateClementP0(mesh.bd,gh_P0);

%% STEP 3: compute solution of (\nabla u2h, \nabla vh) = 0
%  ------  with u2h|_Gamma = g_h

% prescribe values at boundary
x2 = zeros(nC,1);
x2(mesh.bd.nodesOf3DMesh) = g_h;
% assemble right-hand side
b = -mesh.stiffness*x2;
% compute P1-FEM solution
freenodes = setdiff(1:nC, mesh.bd.nodesOf3DMesh);
x2(freenodes) = mesh.stiffness(freenodes, freenodes)\b(freenodes);

%% STEP 4: obtain discrete solution and demagnetization field
%  ------
% compute demagnetization field
% PhM = \grad x1 + \grad x2 on elements

minusGRADu = -gradP1(mesh, x1+x2);
```

Listing B.35: Erstellung eines diskreten externen Feldes, variabel in der Zeit

```
function f = variableInTimeExternalFieldTilde(mesh, f_TimeSpace, quadDeg)
%VARIABLEINTIMEEXTERNALFIELDTILDE    Generate function handle which is
%variable in time.
%   F = VARIABLEINTIMEEXTERNALFIELDTILDE(MESH, F_L2, QUADDEG) returns a
%   function handle f(t), which yields a discrete P1 field for each
```

```
%   time t. The input should be a function handle F_L2(t,X) which for every
%   time t can be evaluated using an array whose rows are points in R^3.
%
%   Author: Josef Kemetmueller - 16.12.2013

f = @(t) hTildeL2(mesh, @(X) f_TimeSpace(t,X), quadDeg);
end
```

B.6. Dimensionsbehaftete Berechnungen

Listing B.36: Berechnung der Anisotropie-Energie

```
function energyAni = energyAnisotropy(mesh, m)
%ENERGYANISOTROPY    Compute the anisotropy energy
%   ENERGYANISOTROPY(MESH, M) returns the anisotropy energy of the
%   magnetziation M.
%
%   Author: Josef Kemetmueller - 16.12.2013
energyAni = mesh.material.K*inner_P1_P1_L2(mesh, 1-m*mesh.material.easyaxis, ...
                                                 1+m*mesh.material.easyaxis);
```

Listing B.37: Berechnung der Austausch-Energie

```
function energyEx = energyExchange(mesh, m)
%ENERGYEXCHANGE    Compute the exchange energy
%   ENERGYEXCHANGE(MESH, M) returns the exchange energy of the
%   magnetziation M.
%
%   Author: Josef Kemetmueller - 16.12.2013
energyEx = mesh.material.A*inner_gradP1_gradP1_L2(mesh, m, m);
% Alternatively:
% energyEx = mesh.material.A*inner_P0_P0_L2(mesh,gradP1(mesh,m),gradP1(mesh, m));
```

Listing B.38: Berechnung der Zeeman-Energie

```
function energyExt = energyExternal(mesh, m, f_t, quadDeg)
%ENERGYEXTERNAL    Compute the external [Zeeman] energy
%   ENERGYEXTERNAL(MESH, M, F_T, QUADDEG) returns the external energy of
%   the magnetziation M. F_T(X) is a function handle for the external field
%   at some time T and QUADDEG is the quadrature degree.
%
%   Author: Josef Kemetmueller - 16.12.2013

[~, ~, ~, ~, mesh.material.Ms] = nondimensionalization(mesh.material, 0, 0);
mu0 = 4*pi*1e-7; % vacuum permeability in [T*m/A]
energyExt = -mu0*mesh.material.Ms^2*inner_L2_P1_L2(mesh, f_t, m, quadDeg);
```

Listing B.39: Berechnung der Streufeld-Energie

```
function energyStray = energyStrayfield(mesh, m)
%ENERGYSTRAYFIELD    Compute the strayfield energy
```

```
%   ENERGYSTRAYFIELD(MESH, M) returns the strayfield energy of the
%   magnetziation M.
%
%   Author: Josef Kemetmueller - 16.12.2013
[~, ~, ~, ~, mesh.material.Ms] = nondimensionalization(mesh.material, 0, 0);


mu0 = 4*pi*1e-7;    % vacuum permeability in [T*m/A]
SF = strayfield(mesh, m);
energyStray = -mu0*mesh.material.Ms^2/2*inner_P0_P1_L2(mesh,SF,m);
```

Listing B.40: Berechnung der konstanten zur Entdimensionalisierung der Gleichung

```
function [C_ex, C_ani, tau0, tauEnd, Ms, mu0] = ...
                                        nondimensionalization(material, t0, tEnd)
%NONDIMENSIONALIZATION    Computes the constants necessary for the
%nondimensional formulation of the LLG.
%
%   Author: Josef Kemetmueller - 16.12.2013

% physical constants
gamma_0 = 2.210173e5;    % gyromagnetic ratio in [m/(As)]
mu0 = 4*pi*1e-7;    % permeability of vacuum in [T*m/A]
L = 1;% [m]
assert(isfield(material,'Js')||isfield(material,'Ms'), ...
                                          'Either Js or Ms must be given.');

% saturation magnetization Ms in [A/m]
if ~isfield(material,'Ms')
    Ms = material.Js/mu0;
else
    Ms = material.Ms;
end
C_ex  = 2*material.A/(mu0*Ms*Ms*L*L);
C_ani = material.K/(mu0*Ms*Ms);
tau0 = gamma_0*Ms*t0;
tauEnd = gamma_0*Ms*tEnd;
```

B.7. Postprocessing

Listing B.41: Plottet und speichert eine Magnetisierung

```
function plotAndSaveMagnetization(mesh, path, name, mhj, j, variant)
%PLOTANDSAVEMAGNETIZATION    Plots and saves the magnetization.
%   See also:
%   PLOTANDSAVEPOSTPROCESSOR
%   PLOTPOSTPROCESSOR
%   SAVEPOSTPROCESSOR
%   ENERGYPLOTPOSTPROCESSOR
%
%   Author: Josef Kemetmueller - 16.12.2013

plotMagnetization(mesh,mhj,j);
```

```
12 saveMagnetization(mesh,path,name,mhj,j,variant);
```

Listing B.42: Plottet die Energiebeiträge

```
1  function plotEnergies(mesh, mhj, j, f_t)
2  %PLOTENERGIES    Plots the energies.
3  %   See also:
4  %   PLOTANDSAVEPOSTPROCESSOR
5  %   PLOTPOSTPROCESSOR
6  %   SAVEPOSTPROCESSOR
7  %   ENERGYPLOTPOSTPROCESSOR
8  %
9  %   Author: Josef Kemetmueller - 16.12.2013
10 persistent energies;
11 if j==1
12     energies = struct('Exchange',[],...
13                       'Anisotropy', [], ...
14                       'Strayfield', [], ...
15                       'External',[]);
16 end
17
18 energyExch = energyExchange(mesh, mhj);
19 energyAni = energyAnisotropy(mesh, mhj);
20 energyStray = energyStrayfield(mesh, mhj);
21 energyExt = energyExternal(mesh, mhj, f_t, 2);
22
23 energies.Exchange(end+1) = energyExch;
24 energies.Anisotropy(end+1) = energyAni;
25 energies.Strayfield(end+1) = energyStray;
26 energies.External(end+1) = energyExt;
27 Esum = energies.Exchange+...
28        energies.Anisotropy+...
29        energies.Strayfield+...
30        energies.External;
31
32 % if mod(j,numSteps)~=0
33 %     return;
34 % end
35 figure(1);
36 plotMagnetization(mesh,mhj,j)
37 figure(2);
38 clf;
39 plot([energies.Exchange; energies.Anisotropy; ...
40       energies.Strayfield; energies.External; Esum]');
41 legend('Exchange', 'Anisotropy', 'Strayfield', 'External', 'Sum of energies');
42 drawnow;
43
44 end
```

Listing B.43: Plottet die Magnetisierung

```
1  function plotMagnetization(mesh,mhj,j)
2  %PLOTMAGNETIZATION    Plots the magnetization.
3  %   See also:
4  %   PLOTANDSAVEPOSTPROCESSOR
5  %   PLOTPOSTPROCESSOR
```

```
%   SAVEPOSTPROCESSOR
%   ENERGYPLOTPOSTPROCESSOR
%
%   Author: Josef Kemetmueller - 16.12.2013
scalePlot = 1e-9;
whatToPlot = 'everything';
switch whatToPlot
    case 'boundary'
        frontBd = any(mesh.coordinates>(0.9)*scalePlot/2,2);
        X = @(i) 1/scalePlot*mesh.coordinates(frontBd,i);
        mjhplot = @(i) mhj(frontBd,i);
        quiver3(X(1),X(2),X(3), ...
                mjhplot(1),mjhplot(2),mjhplot(3));
    case 'everything'
        X = @(i) 1/scalePlot*mesh.coordinates(:,i);
        quiver3(X(1),X(2),X(3), ...
                mhj(:,1),mhj(:,2),mhj(:,3));
        xlabel('x');
        ylabel('y');
        zlabel('z');
end

title(sprintf('%d. step',j));
axis vis3d;
axis equal;
drawnow;
```

Listing B.44: Speichert die Magnetisierung

```
function saveMagnetization(mesh,path,name,mhj,j,variant) %#ok<*INUSL>
%SAVEMAGNETIZATION    Saves the magnetization.
%   See also:
%   PLOTANDSAVEPOSTPROCESSOR
%   PLOTPOSTPROCESSOR
%   SAVEPOSTPROCESSOR
%   ENERGYPLOTPOSTPROCESSOR
%
%   Author: Josef Kemetmueller - 16.12.2013
filename = sprintf('%s/m_%s_%d',path,name,j);
switch variant
    case 'txt'
        dlmwrite([filename,'.txt'], mhj);
    case 'mat'
        save(filename,'mhj');
    otherwise
        error('Unknown option %s', variant);
end
coo = 1e9*mesh.coordinates; % Rescale from nanometers
fileout = [filename, '.vtu'];
vtkquiver(coo(:,1), coo(:,2), coo(:,3), ...
          mhj(:,1), mhj(:,2), mhj(:,3), 'magnetization', fileout);
%%
bdN = mesh.bd.nodesOf3DMesh;
fileout = [filename, '_bd.vtu'];
vtkquiver(coo(bdN,1), coo(bdN,2), coo(bdN,3), ...
          mhj(bdN,1), mhj(bdN,2), mhj(bdN,3), 'magnetization', fileout);
```

Listing B.45: Postprozessor um die Energy zu plotten

```
function postProcessor = energyPlotPostprocessor(mesh, f)
%ENERGYPLOTPOSTPROCESSOR    Constructs a postprocessor that plots the energies.
%   POSTPROCESSOR = ENERGYPLOTPOSTPROCESSOR(MESH, F) plots the energies
%   using the function handle F(t,x) as external field.
%
%   See also:
%   PLOTANDSAVEPOSTPROCESSOR
%   PLOTPOSTPROCESSOR
%   SAVEPOSTPROCESSOR
%
%   Author: Josef Kemetmueller - 16.12.2013
postProcessor = @(mhj,j,tj) plotEnergies(mesh, mhj, j, @(x) f(tj,x));
```

Listing B.46: Postprozessor um die Magnetisierung zu plotten und zu speichern

```
function postProcessor = plotAndSavePostprocessor(mesh, path, name)
%PLOTPOSTPROCESSOR    Constructs a postprocessor that plots and saves.
%   POSTPROCESSOR = PLOTPOSTPROCESSOR(MESH, PATH, NAME) plots the
%   magnetization at the given mesh and saves in PATH under NAME
%
%   See also:
%   PLOTPOSTPROCESSOR
%   SAVEPOSTPROCESSOR
%   ENERGYPLOTPOSTPROCESSOR
%
%   Author: Josef Kemetmueller - 16.12.2013
postProcessor = @(mhj,j,tj) plotAndSaveMagnetization(mesh,path,name,mhj,j,'mat');
```

Listing B.47: Postprozessor um die Magnetisierung zu plotten

```
function postProcessor = plotPostprocessor(mesh)
%PLOTPOSTPROCESSOR    Constructs a postprocessor that plots.
%   POSTPROCESSOR = PLOTPOSTPROCESSOR(mesh) plots the magnetization at the
%   given mesh.
%
%   See also:
%   PLOTANDSAVEPOSTPROCESSOR
%   SAVEPOSTPROCESSOR
%   ENERGYPLOTPOSTPROCESSOR
%
%   Author: Josef Kemetmueller - 16.12.2013
postProcessor = @(mhj,j,tj) plotMagnetization(mesh,mhj,j);
```

Listing B.48: Postprozessor um die Magnetisierung zu speichern

```
function postProcessor = savePostprocessor(mesh, path, name)
%SAVEPOSTPROCESSOR    Constructs a postprocessor that saves
%   POSTPROCESSOR = SAVEPOSTPROCESSOR(MESH, PATH, NAME) saves the magnetization
%   in the path PATH using the filename NAME.
%
%   See also:
%   PLOTANDSAVEPOSTPROCESSOR
%   PLOTPOSTPROCESSOR
```

```
9  %   ENERGYPLOTPOSTPROCESSOR
10 %
11 %   Author: Josef Kemetmueller - 16.12.2013
12 postProcessor = @(mhj,j,tj) saveMagnetization(mesh,path,name,mhj,j,'mat');
```

B.8. Lösung

Listing B.49: Uniformes Zeitschrittverfahren zur Lösung der LLG-Gleichung

```
1  function timeStepping(ExampleData, timestepFunction, postprocessMagnetization)
2  %TIMESTEPPING    Solves the LLG-equation using uniform timestepping.
3  %   TIMESTEPPING(EXAMPLEDATA, TIMESTEPFUNCTION, POSTPROCESSOR) performs
4  %   uniform timestepping to solve the nondimensional LLG-equation:
5  %   ------------------------------------------------------
6  %       m_t - alpha*cross(m,m_t) = -cross(m,h_eff)
7  %           where h_eff is defined by:
8  %       h_eff := C_ex*laplace(m) + Pi(m) + f(t)
9  %       Pi(m) := -C_ani*DPhi(m) + strayfield(m)
10 %   ------------------------------------------------------
11 %   EXAMPLEDATA is a struct, that contains the following data:
12 %       mesh         Geometry of the problem
13 %       alpha        damping factor
14 %       C_ex         Exchange constant
15 %       @(X) Piht(X) Discretized Pi-operator
16 %       @(t) fht(X)  Discretized external field
17 %       tau0         Start time
18 %       tauEnd       End time
19 %       steps        Number of timesteps to perform
20 %
21 %   TIMESTEPFUNCTION is a function handle to a solver that computes the
22 %   next timestep. At the moment the options are
23 %       @midpoint_newton      A newton iteration of the midpoint scheme
24 %       @midpoint_fixedPoint  A fixedpoint iteration of the midpoint scheme
25 %
26 %   POSTPROCESSOR is function handle, that specifies what to do with each
27 %   simulation. At the moment the options are:
28 %       @savePostprocessor(path, name)
29 %       @plotPostprocessor(mesh)
30 %       @plotAndSavePostprocessor(mesh, path, name)
31 %       @energyPlotPostprocessor(mesh, f)
32 %
33 %   Author: Josef Kemetmueller - 16.12.2013
34
35 mhj = ExampleData.mh0;
36
37 t = linspace(ExampleData.tau0, ExampleData.tauEnd, ExampleData.steps+1);
38 for j = 1:length(t)-1
39     mhj = timestepFunction(ExampleData.mesh, ExampleData.alpha, ...
40                            ExampleData.C_ex, ExampleData.Piht, ...
41                            ExampleData.fht, mhj, t(j), t(j+1));
42     postprocessMagnetization(mhj,j,t(j));
43 end
```

Listing B.50: Vorberechnung häufig verwendeter Werte

```
function mesh = precomputeMeshValues(coordinates, elements, material)
%PRECOMPUTEMESHVALUES    Precomputes data which is often needed in the simulation
%   MESH = PRECOMPUTEMESHVALUES(COORDINATES, ELEMENTS, MATERIAL) returns
%   a mesh structure containing often used data of the mesh.
%   COORDINATES and ELEMENTS represent the mesh by standard simplex-vertex
%   format. MATERIAL is a struct containing the material parameters
%       'A'  [J/m]      used for exchange energy computation
%       'Ms' [A/m]      used for external and strayfield energy computation
%       'K'  [J/m^3]    used for anisotropy energy computation
%       'easyaxis' []   used for anisotropy energy computation and anistropy.
%
%   Author: Josef Kemetmueller - 16.12.2013
dimMesh = size(elements,2)-1;
nC = size(coordinates,1);
dimSpace = size(coordinates,2);
%% Meshdata
mesh.coordinates = coordinates;
mesh.elements = elements;
mesh.volumes = getElementVolumes(mesh);
mesh.betas = getBetas(mesh);
mesh.hatGrads = getHatGrads(mesh);
%%
lengths = @(X) sum(X.^2);
if exist('material','var')
    material.easyaxis = reshape(material.easyaxis,3,[]);
    assert(abs(lengths(material.easyaxis)-1)<1e2*eps, 'Easyaxis not normed');
    mesh.material = material;
    [~, ~, ~, ~, mesh.material.Ms] = nondimensionalization(material, 0, 0);
end
%% inner_gradHatI_gradHatJ
% Used in: inner_HatIxlaphHatJ_P1_h
mesh.inner_gradHatI_gradHatJ = cell(dimMesh+1,dimMesh+1);
for i = 1:dimMesh+1
    for j = 1:dimMesh+1
        mesh.inner_gradHatI_gradHatJ{i,j} = ...
            inner_P0_P0_L2(mesh, ...
                           mesh.hatGrads{i},mesh.hatGrads{j},...
                           'elementwise');
    end
end
%% Stiffness matrix
mesh.stiffness = inner_gradP1_gradP1_L2(mesh);
%% Boundary-data and Double-Layer-Potential
% outwards orientation of the boundary-faces is important for buildK.
mesh.bd = struct('elements', getBoundary(mesh,'outwards'), ...
                 'coordinates', coordinates);
% Remove unused nodes from boundary-mesh and renumber.
[mesh.bd, ~, mesh.bd.nodesOf3DMesh] = cleanMesh(mesh.bd);
mesh.bd.volumes = getElementVolumes(mesh.bd); % volumes = areas
% volumes are actually the face areas, but we call them volumes to be
% consistent with the mesh-data-structure.
if dimMesh==3
    mesh.bd.DLPot = bsxfun(@rdivide, -buildK(mesh.bd.coordinates, ...
                                        mesh.bd.elements,mesh.bd.volumes,0.5),...
                       mesh.bd.volumes);
% The above computation yields the same DLPot independent of the scaling of
% mesh.bd.coordinates. (If you divide mesh.bd.coordinates by 10 and the
```

```
58 % areas by 100 it will return the same matrix)
59
60 else
61     % Also outwards oriented boundary doesn't work with getBoundary.
62     mesh.coordinates = [coordinates, zeros(nC,3-dimSpace)];
63     warning('No strayfield computation is available for %dD',dimMesh);
64     mesh.bd.DLPot = zeros([size(mesh.bd.elements,1),size(mesh.bd.coordinates,1)]);
65 end
66 end
```

Literaturverzeichnis

[BP06] Bartels, Sören und Andreas Prohl: *Convergence of an Implicit Finite Element Method for the Landau-Lifshitz-Gilbert Equation.* SIAM J. Numerical Analysis, 44(4):1405–1419, 2006.

[Bra07] Braess, Dietrich: *Finite Elemente.* Springer-Verlag, Berlin, 4. Auflage, 2007.

[BS02] Brenner, Susanne C. und L. Ridgway Scott: *The Mathematical Theory of Finite Element Methods.* Texts in Applied Mathematics. Springer-Verlag, New York, 2. Auflage, 2002.

[Els96] Elstrodt, Jürgen: *Maß- und Integrationstheorie.* Springer-Verlag, Berlin, 1996.

[Gil55] Gilbert, Thomas L.: *A Lagrangian formulation of the gyromagnetic equation of the magnetic field.* Physical Review, 100:1243–1255, 1955.

[Gol12] Goldenits, Petra: *Konvergente numerische Integration der Landau-Lifshitz-Gilbert Gleichung.* Doktorarbeit, Technische Universität Wien, Mai 2012.

[HS98] Hubert, Alex und Rudolf Schäfer: *Magnetic Domains: The Analysis of Magnetic Microstructures.* Springer-Verlag, Berlin, 1998.

[LL35] Landau, Lev Davidovich und Evgeny Lifshitz: *On the theory of the dispersion of magnetic permeability in ferromagnetic bodies.* Phys. Z. Sowjetunion, 8:153–169, 1935.

[May13] Mayr, Markus: *Millg - ein Löser für die Landau-Lifshitz-Gilbert-Gleichung.* Diploma Thesis, Institut für Analysis and Scientific Computing, Technische Universität Wien, Wien, Austria, 2013.

[McL00] McLean, William: *Strongly elliptic systems and boundary integral equations.* Cambridge University Press, 2000.

[Pag13] Page, Marcus: *On Dynamical Micromagnetism.* Doktorarbeit, Technische Universität Wien, Oktober 2013.

[Wer00] Werner, Dirk: *Funktionalanalysis.* Springer-Verlag, Berlin, 3. Auflage, 2000.