Technische Universität Graz
Institut für Grundlagen und Theorie der Elektrotechnik

Diplomarbeit

Quellenfeldberechnung für komplexe Leitergeometrien

Betreuer:
Univ.-Prof. Dipl.-Ing. Dr.techn. Oszkar Biro
Dipl.-Ing. Gebhard Wallinger, BSc

Verfasser:
Wolfgang Schütz

November 10, 2016

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....
(Unterschrift)

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Contents

1 Abstract

1.1 Deutsch

In der numerischen Feldberechnung wird zur Berechnung des stationären Magnetfeldes für gewöhnlich der Ansatz des Reduzierten-Skalarpotentials herangezogen. Das Quellenfeld, welches man dafür benötigt, wird meist über die Biot-Savart'sche-Methode berechnet.
In dieser Arbeit wird eine Alternative zur Biot-Savart'schen-Methode beschrieben und untersucht. Weiters wird eine mögliche MATLAB- Implementierung vorgestellt.

1.2 Englisch

In numerical field calculation, the approach of the reduced-scalar potential formulation is used to calculate the stationary magnetic field. The source field is usually obtained by exploiting the Biot-Savart method.
In this thesis, an alternative approach to the Biot-Savart method is described and investigated. Furthermore, a possible MATLAB implementation is presented.

2 Einleitung

In der numerischen Feldberechnung ist einer der gängigen Ansätze zur Berechnung des stationären Magnetfeldes die Verwendung eines reduzierten Skalarpotentials [2].
Bei dieser Beschreibung wird die magnetische Erregung als Kombination eines Vektorpotentials und dem Gradienten eines Skalarpotentials dargestellt. Diese Beschreibung gründet darauf, dass die magnetische Erregung $\boldsymbol{H}$ in einem stromlosen Gebiet (im statischen Fall) durch ein Gradientenfeld beschreibbar ist. Dies kann durch die vierte Maxwell'sche Gleichung [3] gezeigt werden. Für Gebiete mit vorhandener Stromdichte $\boldsymbol{J}$ wird angenommen, dass die gesamte magnetische Erregung $\boldsymbol{H}$ aus der Summe von $\boldsymbol{T}$ und $\boldsymbol{H_R}$ besteht.

$\boldsymbol{H_R}$ oder reduzierte *Magnetische Erregung*, ist jener Anteil von $\boldsymbol{H}$, welcher durch den Gradienten eines Skalarfeldes darstellbar ist.
Das Quellenfeld $\boldsymbol{T}$ übernimmt die Darstellung des Wirbelanteils von $\boldsymbol{H}$, also jenen Anteil, welcher von Strömen verursacht wird (rechte Seite der vierten Maxwell-Gleichung). Somit muss die Rotation von $\boldsymbol{T}$ die Ströme darstellen.

Bei der Lösung des Problems durch die Methode der finiten Elemente unter Verwendung von Edge-Elementen (Elementen mit Kantenformfunktionen) wird klassisch der Biot-Savart'sche-Ansatz verwendet (Kapitel 3.5). Dieser Ansatz hat den Nachteil, dass für jede Kante des Systems das Biot-Savart-Feld berechnet werden muss. Dies ist aus später noch genauer erläuterten Gründen überaus zeitaufwendig.
Aus diesem Grund wird in dieser Arbeit ein anderer Weg zur Berechnung des Quellenfeldes $\boldsymbol{T}$ untersucht und implementiert. Dieser Ansatz berechnet das $\boldsymbol{T}$-Feld direkt über die Stromdichte und wird im folgenden deshalb **Direkter-Ansatz** genannt.

2.1 Aufgabenstellung

Das Ziel dieser Diplomarbeit besteht darin, eine Evaluierungsplattform für den direkten Ansatz zu generieren. Um die Machbarkeit bzw. den nötigen Arbeitsaufwand abzuschätzen, wurde eine geeignete Implementierungsplattform gesucht. Schnell fiel die Wahl auf MATLAB, da das MATLAB-Framework alle nötigen und gewünschten Features unterstützt und keine zusätzliche Einarbeitungszeit meinerseits benötigt wurde.
Den ersten Arbeitsschritt stellte die Generierung eines Datenflusskonzeptes dar. Dieses Datenflusskonzept gliedert und koordiniert den Datenaustausch zwischen den verschiedenen Programmteilen.
Außerdem wurde evaluiert, welche Teile der bereits bestehenden ELEFANT3D-Infrastruktur wiederverwendet werden können. Aufbauend auf diese Evaluierung wurde eine ELEFANT3D-Schnittstelle entworfen und implementiert.

3 Methoden

3.1 Ritz-Galerkin-Methode

Die Ritz-Galerkin-Methode oder auch das Ritz-Galerkin-Verfahren dient zur Lösung von partiellen-DGL, welche als Operator-Gleichung der Form $\Psi y - f = 0$ vorliegen, wobei Ψ einen symmetrischen, positiven Operator darstellt.
Hat man die partiellen-DGL erst in eine Operatorgleichung umgeschrieben, kann man über $\int_\Omega h\,(\Psi y - f)\;d\Omega = 0$ eine Lösung bestimmen. Wobei h die homogenen und y die inhomogenen Dirichlet-Randbedingungen erfüllt. Die Methode selbst kommt aus der Variationsrechnung. Ausgehend davon, dass die gesuchte Funktion $y(x)$ eine Näherung der Form

$$\hat{y}(x) = y_D(x) + \sum_{i=1}^{n} a_i \cdot y_i(x)$$

besitzt, für welches das Funktional

$$I(a_1, a_2, ..., a_n) = \int_\Omega F(x, y, \dot{y})\, d\Omega$$

ein Extremum besitzt, formuliert Ritz, dass bei gegebenen y_D und y_i die Konstanten a_i wie folgt bestimmt werden können.

$$\frac{\partial I(a_1, a_2, ..., a_n)}{\partial\,[a_1, a_2, ..., a_n]} = \mathbf{0}$$

Für das Funktional kann der Satz über das Minimum eines quadratischen Funktionals herangezogen werden. Dieser Satz besagt, dass eine Operatorgleichung der Form $\Psi y - f = 0$ mit dem linearen, symmetrischen und positiven Operator Ψ folgendes Funktional minimiert.

$$I(u) = \frac{1}{2}\int_\Omega y\Psi y\, d\Omega - \int_\Omega yf\, d\Omega$$

Beim Verfahren der gewichteten Residuen wird der Fehler der Approximation, das Residuum, welches durch $\Psi\hat{y} - f = r$ gegeben ist, mit einer Gewichtsfunktion h gewichtet und über das Definitionsgebiet auf 0 setzt. Galerkin verwendete zur Gewichtung des Residuums die Ansatzfunktionen y_i, wodurch er folgende Gleichung erhielt.

$$\int_\Omega \sum_{j=1}^{n} y_j \left(\Psi \left(y_D(x) + \sum_{i=1}^{n} a_i \cdot y_i(x) \right) - f \right) d\Omega = 0 \tag{1}$$

$$\int_\Omega y\,(\Psi\,(y_D(x) + y) - f)\;d\Omega = 0 \tag{2}$$

3.2 Skalarpotenialbeschreibung des Stationären-Elektrischen-Strömungsfeldes

Die zur Skalarpotentialbeschreibung des Stationären-Elektrischen-Strömungsfeldes benötigte Differenzialgleichung lässt sich div$\boldsymbol{J} = 0$ und dem Ansatz $\boldsymbol{E} = -\text{grad}\,(V)$ mithilfe der Materialgleichung $\boldsymbol{J} = \sigma \boldsymbol{E}$ hergeleitet werden.

$$-\text{div}\sigma\text{grad}\,(V) = 0 \tag{3}$$

Mit den Randbedingungen:

$$V = V_D \qquad \text{auf } \Gamma_D \tag{4}$$

$$\sigma\text{grad}\,(V) \cdot \boldsymbol{n} = \sigma\frac{\partial V}{\partial n} = 0 \qquad \text{auf } \Gamma_N \tag{5}$$

Wobei Γ_D den Dirichlet'schen-Rand (Elektroden) und Γ_N den Neumann'schen-Rand (Grenze zum nicht leitfähigen Gebiet) darstellt.

Daraus lässt sich folgende Operatorgleichung ableiten, welche die Neumann'sche-Randbedingung erfüllt.

$$-\text{div}\sigma\text{grad}\,(V) + \delta_{\Gamma_N}\sigma\frac{\partial V}{\partial n} = 0 \tag{6}$$

$$\int_\Omega \delta_{\Gamma_N} f \, d\Omega = \int_{\Gamma_N} f \, d\Gamma \tag{7}$$

Eingesetzt in die Ritz-Galerkin'sche-Gleichung (2) erhält man:

$$\int_\Omega f_i \left(-\text{div}\sigma\text{grad}\,(V) + \delta_{\Gamma_N}\sigma\frac{\partial V}{\partial n}\right) d\Omega = 0$$

$$\int_\Omega \text{grad}\,(f_i)\,\sigma\text{grad}\,(V)\; d\Omega = 0 \tag{8}$$

Mit dem Ansatz $V = V_D + \sum_{j=1}^{N} a_j \cdot f_j$ kommt man auf die finale Form:

$$\sum_{j=1}^{N} a_j \int_\Omega \text{grad}\,(f_i)\,\sigma\text{grad}\,(f_j)\; d\Omega = -\int_\Omega \text{grad}\,(f_i)\,\sigma\text{grad}\,(V_D)\; d\Omega \tag{9}$$

Dies kann als lineares Gleichungssystem der Form $\boldsymbol{M}\boldsymbol{a} = \boldsymbol{b}$ angeschrieben werden, wobei:

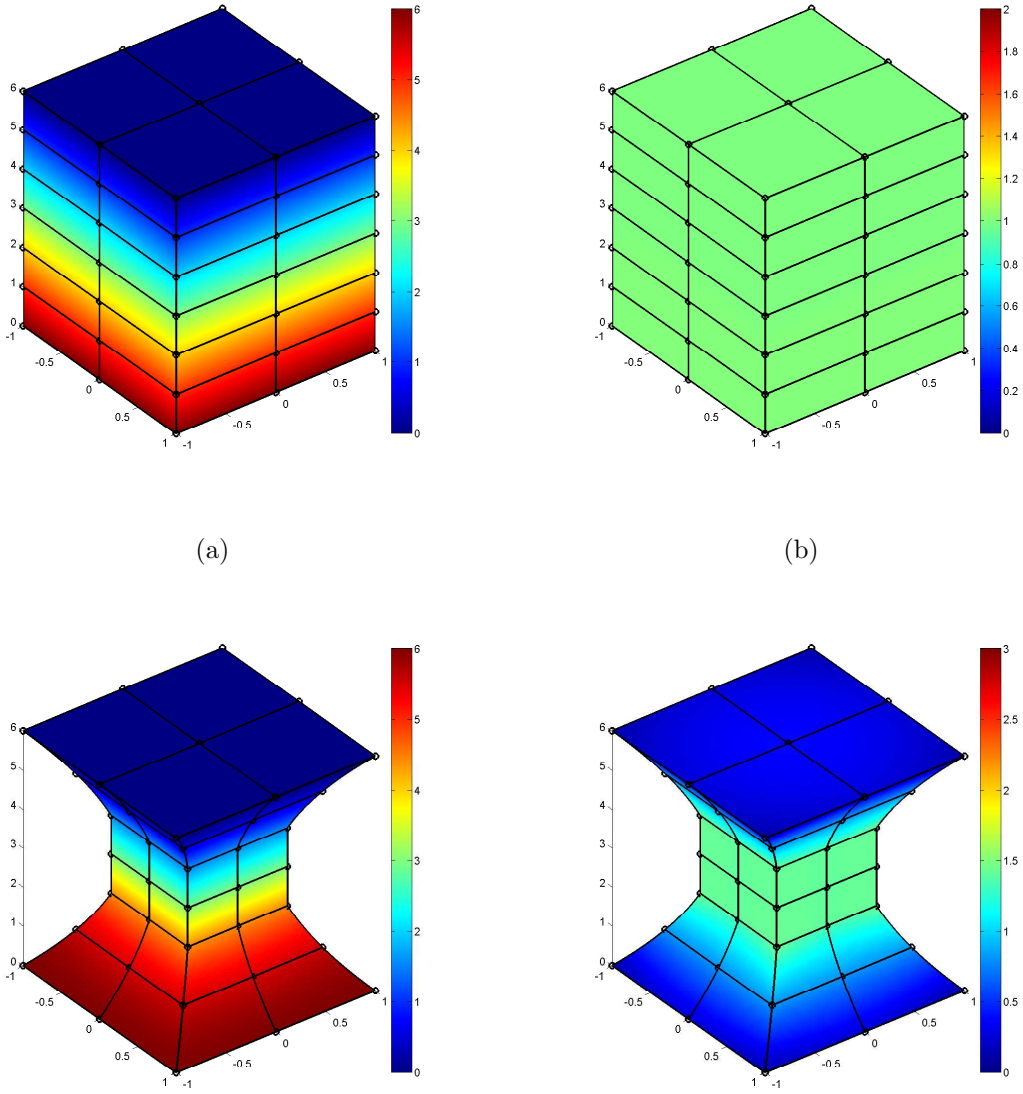
$$\boldsymbol{M}_{i,j} = \int_\Omega \text{grad}\,(f_i)\,\sigma\text{grad}\,(f_j)\; d\Omega \tag{10}$$

$$\boldsymbol{a}_i = a_i \tag{11}$$

$$\boldsymbol{b}_i = -\int_\Omega \text{grad}\,(f_i)\,\sigma\text{grad}\,(V_D) \tag{12}$$

Bei der Realisierung mittels finiter Elemente, kommen Knotenformfunktionen zum Einsatz. Für die Erstellung des Gleichungssystems erhalten alle Knoten (und die entsprechende Formfunktion) eine globale Nummerierung. a_i und f_i stellen somit das Gewicht (Wert) bzw. die Formfunktion des i-ten Knotens dar. Von der Nummerierung ausgenommen werden jene Knoten, bei denen das Potential bereits bekannt ist, zum Beispiel durch eine Potentialvorgabe.

Nehmen wir an, man wollte die Stromdichte durch einen quaderförmigen Leiter ($2 \times 2 \times 6$ m) mit einer spezifischen Leitfähigkeit $\sigma = 1\frac{A}{V\,m}$ berechnen. An den Enden des Leiters wird eine Spannung von 10 V angelegt (Dirichlet'sche Randbedingung). Zuerst wird die Geometrie des Leiters durch finite Elemente approximiert. Anschließend wird das Gleichungssystem erstellt und gelöst. Aus der Lösung des Gleichungssystems werden nun die Gewichtungs-Matrizen der Formfunktionen für die finiten Elemente extrahiert und benutzt, um die Lösung zu approximieren. Das Ergebnis ist in den Abbildungen 1(a) und 1(b) zu sehen. Diese Beispiel wurde leicht modifiziert und erneut gelöst, die Lösung ist in den Abbildungen 1(c) und 1(d) zu sehen.



(a) (b)

(c) (d)

Figure 1: Graphische Darstellung der Lösungen

a) Verlauf des Skalarpotentials V — Farbskala in Volt
b) Verlauf der Stromdichte $|J|$ — Farbskala in Ampere
c) Verlauf des Skalarpotentials V — Farbskala in Volt
d) Verlauf der Stromdichte $|J|$ — Farbskala in Ampere

3.3 Vektorpotenialbeschreibung des Stationären-Elektrischen-Strömungsfeldes

Aus den Maxwell-Gleichungen im stationären Fall geht hervor, dass die Divergenz der Stromdichte gleich 0 ist ($\mathrm{div}\boldsymbol{J} = 0$). Dies ist erfüllt, wenn die Stromdichte $\boldsymbol{J}$ durch die Rotation eines Vektorpotentials beschrieben wird ($\boldsymbol{J} = \mathrm{rot}\boldsymbol{T}$). Aus diesem Ansatz geht mithilfe der Maxwell-Gleichung $\mathrm{rot}\boldsymbol{E} = \boldsymbol{0}$ und dem Materialzusammenhang $\boldsymbol{E} = \rho\boldsymbol{J}$ folgende Differenzialgleichung hervor.

$$\mathrm{rot}\rho\,\mathrm{rot}\boldsymbol{T} = \boldsymbol{0} \tag{13}$$

Mit den Randbedingungen:

$$\boldsymbol{n} \times \boldsymbol{T} = \tau \qquad \text{auf } \Gamma_D \tag{14}$$

$$\rho\,\mathrm{rot}\boldsymbol{T} \times \boldsymbol{n} = \boldsymbol{0} \qquad \text{auf } \Gamma_N \tag{15}$$

Die Dirichlet'sche Randbedingung stellt die Normalkomponente der Stromdichte dar. An den Grenzen zum nicht leitfähigen Gebiet muss τ die Bedingung $\mathrm{div}\tau = 0$ erfüllen. Die Neumann'sche Randbedingung stellt sicher, dass das $\boldsymbol{E}$-Feld auf den Elektroden nur eine Normalkomponente besitzt. Dieses Randwertproblem ist nicht eindeutig, da man zum Vektorpotential $\boldsymbol{T}$ den Gradienten eines beliebigen Skalarpotentials addieren kann, ohne die Differenzialgleichung oder Neumann'sche Randbedingung zu verletzen. Deshalb wird sie auch als ungeeichte Vektorpotentialbeschreibung bezeichnet.

Daraus lässt sich folgende Operatorgleichung ableiten:

$$\mathrm{rot}\rho\,\mathrm{rot}\boldsymbol{T} + \delta_{\Gamma_N} \cdot \rho\,\mathrm{rot}\boldsymbol{T} \times \boldsymbol{n} = 0 \tag{16}$$

Eingesetzt in die Ritz-Galerkin'sche-Gleichung (2) erhält man:

$$\int_\Omega \boldsymbol{f}_i \left(\mathrm{rot}\rho\,\mathrm{rot}\boldsymbol{T} + \delta_{\Gamma_N} \cdot (\rho\,\mathrm{rot}\boldsymbol{T} \times \boldsymbol{n})\right) d\Omega = 0$$

$$\int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho\,\mathrm{rot}\boldsymbol{T}\, d\Omega = 0 \tag{17}$$

Was mit dem Ansatz $\boldsymbol{T} = \boldsymbol{T}_D + \sum_{i=1}^{n} a_i \cdot \boldsymbol{f}_i$ zur folgenden Form führt.

$$\sum_{f=1}^{n} a_j \left(\int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho\,\mathrm{rot}\boldsymbol{f}_j\, d\Omega\right) = -\int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho\,\mathrm{rot}\boldsymbol{T}_D\, d\Omega \tag{18}$$

Dies kann wieder als lineares Gleichungssystem der Form $\boldsymbol{M} \cdot \boldsymbol{a} = \boldsymbol{b}$ angeschrieben werden, wobei:

$$\boldsymbol{M} \cdot \boldsymbol{a} = \boldsymbol{b} \tag{19}$$

$$M_{i\,j} = \int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho\,\mathrm{rot}\boldsymbol{f}_j\, d\Omega \tag{20}$$

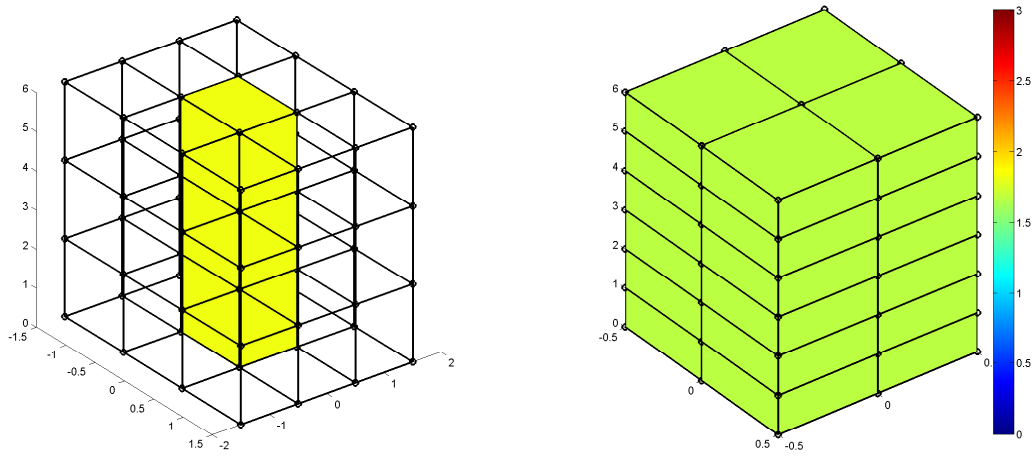
$$\boldsymbol{a}_i = a_i \tag{21}$$

$$\boldsymbol{b}_i = -\int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho\,\mathrm{rot}\boldsymbol{T}_D\, d\Omega \tag{22}$$

Dieses Gleichungssystems kann nicht eindeutig gelöst werden, da die Determinante von $\boldsymbol{M}$ ($\det(\boldsymbol{M})$) gleich null ist. Jedoch liefert das numerische Lösungsverfahren der Konjungierten-Gradienten eine (von ∞ vielen) Lösungen.

Bei der Erstellung des Gleichungssystems kommt rot$\boldsymbol{T}_D$ zum Einsatz. Auf dem Dirichlet'schen Rand (Grenze zum nicht leitfähigen Gebiet) wissen wir, dass die Tangentialkomponente von $\boldsymbol{T}$ gleich null ist ($\boldsymbol{n} \times \boldsymbol{T} = 0$). Zur Veranschaulichung ein kleines Minimalbeispiel:

Betrachten wir die in Abbildung 2(a) gezeigte Geometrie. In den gelben Elementen prägen wir eine konstante Stromdichte $\boldsymbol{J} = J_z = 1$ Ampere ein (Lösung des ersten Beispiels aus Punkt 3.2). In allen anderen Elementen ist die Stromdichte gleich 0. Nach dem Aufstellen und Lösen des Gleichungssystem erhalten wir das Vektorpotential $\boldsymbol{T}$. Um die Lösung zu überprüfen, berechnen wir die Stromdichte der zentralen Elemente (Abbildung 2(b)). Den Gesamtstrom durch eine beliebige Fläche können wir aufgrund des Satzes von Stokes über das geschlossene Linienintegral von $\boldsymbol{T}$ über den Rand der Fläche berechnen. Auch diesen Test besteht die Lösung einwandfrei.



(a) (b)

Figure 2: Graphische Darstellung der Lösungen
a) Geometrie des Beispiels
b) Verlauf der Stromdichte $|J|$ — Farbskala in Ampere

3.4 Reduzierte Skalar-Potential-Beschreibung des stationären Magnetfeldes

Verwendet man ein reduziertes Skalarpotential zur Berechnung des stationären Magnetfeldes, wird man für die magnetische Erregung $\boldsymbol{H}$ mit hoher Wahrscheinlichkeit einen Ansatz der folgenden Form verwenden.

$$\boldsymbol{H} = \boldsymbol{T} - \underbrace{\operatorname{grad}(\Psi)}_{\boldsymbol{H_R}} \tag{23}$$

Bei diesem Ansatz wird die magnetische Erregung $\boldsymbol{H}$ in zwei Komponenten zerlegt. $\boldsymbol{H_R}$ stellt den Gradienten eines Skalarpotentials dar und modelliert den nicht-Quellen-abhängigen Teil der magnetischen Erregung. Das Strömungsvektorpotential $\boldsymbol{T}$ hingegen bildet den Quellen-abhängigen Teil des Feldes dar.
Zur Bestimmung des Quellenfeldes wird normalerweise ein Biot-Savart'scher-Ansatz verwendet. Dieser wird in Kapitel 3.5 genauer beschrieben. Vorwegnehmend kann gesagt werden, dass dieser Ansatz jedoch für komplexere Leiteranordnungen überaus aufwendig ist.

Deshalb erfolgt in Kapitel 3.6 die Beschreibung des Direkten Ansatzes. Bei diesem Ansatz wird die Stromdichte $\boldsymbol{J}$ direkt zur Berechnung des Quellenfeldes $\boldsymbol{T}$ verwendet. Die hohe Skalierbarkeit dieser Methode macht sie zu einer vielversprechenden Alternative zur klassischen Biot-Savart'schen-Methode.

Aber betrachten wir das Problem nun zuerst einmal etwas genauer.
Wir suchen ein Vektorfeld, dessen Rotation gleich der Stromdichte $\boldsymbol{J}$ ist, siehe Gleichung (24). Daraus folgt ebenfalls, dass an den Grenzen zum nicht leitfähigen Gebiet der Normalanteil der Stromdichte gleich Null ist (25).

$$\operatorname{rot}\boldsymbol{T} = \boldsymbol{J} \tag{24}$$

$$\operatorname{rot}\boldsymbol{T} \cdot \boldsymbol{n} = 0 \qquad \text{auf } \boldsymbol{\Gamma}_E \tag{25}$$

Zur Lösung des Problems wird die Finite-Elemente-Methode verwendet. Bei den verwendeten Elementen handelt es sich um 36-kantige Edge-Elemente (Abbildung 3).
Edge-Elemente eignen sich besonders zur Beschreibung von Vektorpotentialen. Bei der Beschreibung eines Vektorpotentials ist es nötig, Informationen über dessen Rotation bzw. dessen Divergenz zu formulieren. Die Kantenformfunktionen haben deshalb folgende Eigenschaften: Die i-te Kantenformfunktion $\boldsymbol{F}_i$ verfügt über die Eigenschaft, dass ein Linienintegral über die i-te Kante 1 ergibt und über alle anderen Kanten 0. Außerdem ist es möglich, mit den K Kantenformfunktionen des Elementes den Gradienten eines beliebigen Skalarfeldes darzustellen.

$$\int_{\boldsymbol{l}_j} \boldsymbol{F}_i(\boldsymbol{l}_j) d\boldsymbol{l} = 1, \quad i = j \tag{26}$$

$$\int_{\boldsymbol{l}_j} \boldsymbol{F}_i(\boldsymbol{l}_j) d\boldsymbol{l} = 0, \quad i \neq j \tag{27}$$

$$\operatorname{grad}(\Phi)_{\boldsymbol{\xi}} = \sum_{i=1}^{K} a_i \cdot \boldsymbol{F}_i \tag{28}$$

$\boldsymbol{F}_i$ Kantenformfunktion der i-ten Kante
$\boldsymbol{l}_j$ j-te Kante des Elementes

Die Kantenformfunktionen werden in den lokalen Koordinaten $\boldsymbol{\xi}$ definiert. Sie bestehen aus zwei Teilen: dem Richtungs- und dem Gewichtungsteil. Der Richtungsteil $\boldsymbol{d}$ der i-ten Kantenfunktion besteht aus einem Vektor, welcher in die lokale Richtung der i-ten Kante zeigt. Diese ist über die Jacobi-Matrix mit der globen Richtung verbunden. Der Gewichtungsteil besteht aus gewöhnlichen skalaren Polynomfunktionen. Die Polynomfunktion wird über das Vektorprodukt aus einem Koeffizientenvektor ($\boldsymbol{m_i}$) und einem Potenzenvektor ($\boldsymbol{L}(\boldsymbol{\xi})$) dargestellt. Somit kann die Formfunktion wie in (29) angeschrieben werden.

$$\boldsymbol{F}_i = \boldsymbol{m_i} \cdot \boldsymbol{L}(\boldsymbol{\xi}) \cdot \boldsymbol{d_i} \tag{29}$$

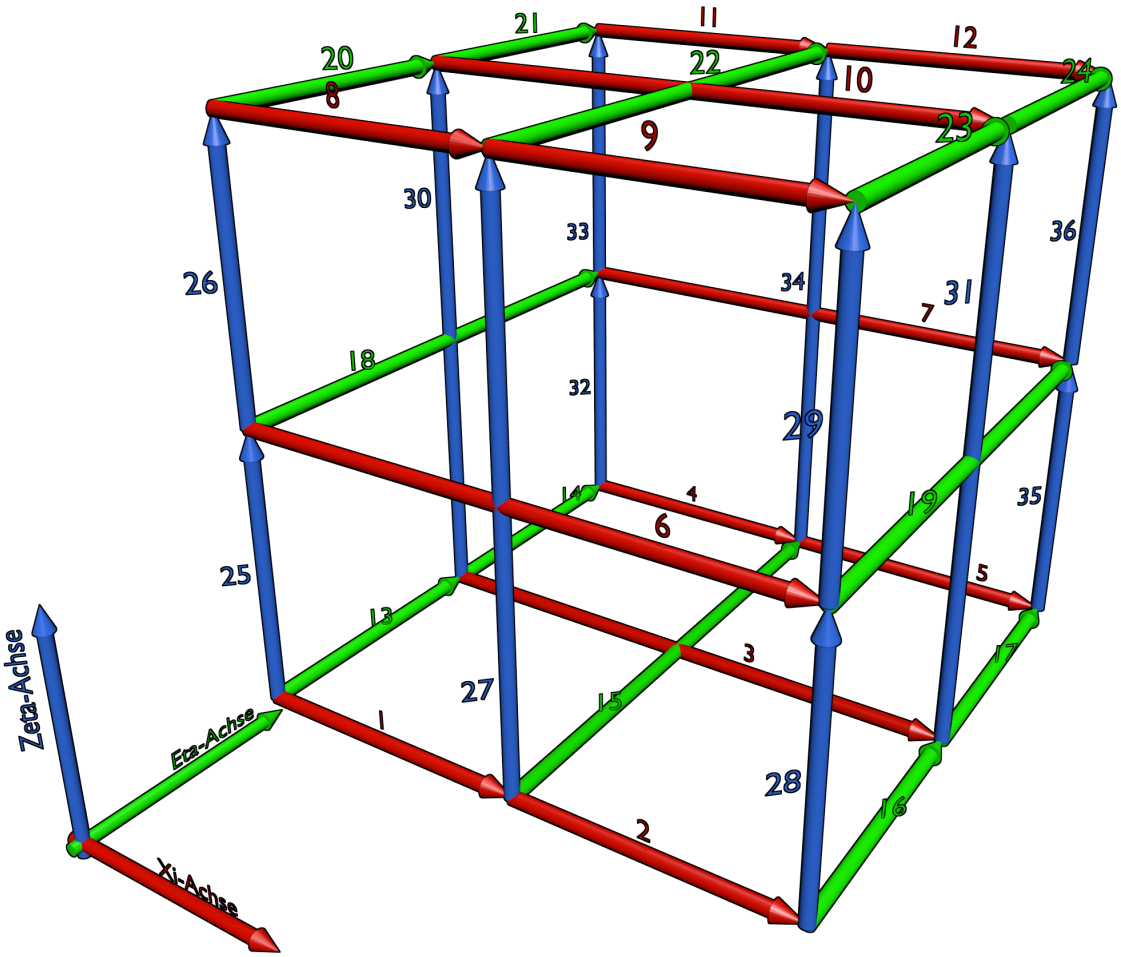


Figure 3: Kantennummerierung eines 36-kantigen Edge-Elementes

3.5 Biot–Savartsche–Methode

Beim klassischen Biot-Savart'schen-Ansatz (beschrieben in [2]) wird das Biot-Savart'sche-Gesetz (30) verwendet, um das Quellenfeld $\boldsymbol{T}$ zu bestimmen.

$$\boldsymbol{H_B}(\boldsymbol{r}) = \frac{I}{4\pi} \oint_{\Omega} \frac{d\boldsymbol{r}' \times \left(\boldsymbol{r} - \boldsymbol{r}'\right)}{|\boldsymbol{r} - \boldsymbol{r}'|^3} \tag{30}$$

Das Quellenfeld $\boldsymbol{T}$ wird durch die Kantenformfunktionen wie folgt beschrieben:

$$\boldsymbol{T} = \sum_{k=1}^{K} g_k \cdot \boldsymbol{F_k} \tag{31}$$

K steht hierbei für die Gesamtanzahl der unterschiedlichen Kanten des Finiten-Element-Gitters.

Zur Berechnung des Quellenfeldes wird zuerst das Problem inklusive Leitergeometrie durch finite Elemente approximiert. Die Modellierung der Ströme erfolgt durch Fadenströme, welche innerhalb des leitfähigen Gebietes verlaufen. Die Bestimmung der Kantengewichtungen g_k für jene Kanten, welche sich außerhalb oder auf den Grenzen des leitfähigen Gebietes befinden, erfolgt nun über die Auswertung des Biot-Savart'schen-Gesetzes (30). Obwohl sich das Volumsintegral über den Fadenstrom auf ein Linienintegral reduziert, muss zur Berechnung der Kantengewichtung von Kante k (g_k) immer noch folgendes Doppelintegral gelöst werden (32).

$$\begin{aligned} g_k &= \int_{\boldsymbol{l}_k} \boldsymbol{H_B}\left(\boldsymbol{x}\right)) \, d\boldsymbol{x} \\ g_k &= \sum_{m=1}^{M} \frac{I_m}{4\pi} \int_{\boldsymbol{l}_k} \oint_{\boldsymbol{l}_{c_m}} \frac{d\boldsymbol{y} \times (\boldsymbol{y} - \boldsymbol{x})}{|\left(\boldsymbol{y} - \boldsymbol{x}\right)|^3} \mathrm{d}\boldsymbol{x} \end{aligned} \tag{32}$$

Der Verlauf des m-ten von M Stromfäden bzw. dessen Stromstärke werden durch $\boldsymbol{l}_{c_m}$ bzw. I_m beschreiben. Die Geometrie (den Verlauf) der k-ten Kante des Finiten-Elemente-Gitters beschreibt $\boldsymbol{l}_k$.
Die restlichen Kanten (Kanten innerhalb des leitfähigen Gebietes) werden nun durch die Vektorfeldbeschreibung des elektischen-Strömungsfeldes ermittelt [1]. Abschließend wird das Gleichungssystem, welches in Gleichung (33) dargestellt ist, aufgestellt und gelöst.

$$\begin{aligned} \boldsymbol{M} \cdot \boldsymbol{g} &= \boldsymbol{b} \\ M_{i,j} &= \int_{\Omega} \mathrm{rot}\boldsymbol{f}_i \cdot \rho \cdot \mathrm{rot}\boldsymbol{f}_j \mathrm{d}\Omega \\ b_i &= -\int_{\Omega} \mathrm{rot}\boldsymbol{f}_i \cdot \rho \cdot \mathrm{rot}\boldsymbol{T_D} \mathrm{d}\Omega \end{aligned} \tag{33}$$

$\boldsymbol{T_D}$ ist jener Teil des Feldes, welcher die Dirichlet'schen Randbedingungen erfüllt. Sprich jene Kanten, welche sich auf der Grenze zum nicht leitfähigen Gebiet befinden. Diese wurden bereits durch die Auswertung der Biot-Savart'schen-Beziehung ermittelt. Somit ist nach dem Lösen des Gleichungssystems ein Ansatz für $\boldsymbol{T}$ gefunden, welcher die Bedingungen (24) und (25) erfüllt.

Der Nachteil dieser Methode liegt im hohen Berechnungsaufwand, welcher nötig ist, um das Biot-Savart-Feld zu berechnen.

3.6 Direkter Ansatz

Verwendet man die Materialgleichung (34) und die dritte Maxwell-Gleichung (35), um die Bedingung (24), welche an $\boldsymbol{T}$ gestellt wird, umzuformulieren, erhält man für $\boldsymbol{T}$, im statischen Fall, folgende Differenzialgleichung:

$$\boldsymbol{E} = \rho \cdot \boldsymbol{J} \tag{34}$$

$$\mathrm{rot}\boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \tag{35}$$

$$\mathrm{rot}\rho\mathrm{rot}\boldsymbol{T} = 0 \tag{36}$$

Dieses Problem ist dasselbe, welches für die Vektorpotential-Beschreibung des elektrischen Strömungsfeldes gelöst werden muss. Verwendet man zum Lösen dieses Problems das Galerkin-Verfahren (17), erhält man ein Gleichungssystem folgender Form:

$$\boldsymbol{M} \cdot \boldsymbol{g} = \boldsymbol{b} \tag{37}$$

$$M_{i,j} = \int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho \cdot \mathrm{rot}\boldsymbol{f}_j \mathrm{d}\Omega$$

$$b_i = -\int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho \cdot \mathrm{rot}\boldsymbol{T_D} \mathrm{d}\Omega$$

Die Indizes i und j stehen wie üblich für die Position der Kante in der globalen Kantennummerierung. Dies ist prinzipiell das gleiche Gleichungssystem, welches bereits bei der Biot-Savart'schen-Methode erwähnt wurde.
Jedoch setzen wir für $\mathrm{rot}\boldsymbol{T_D}$ nun direkt die Stromdichte $\boldsymbol{J}$ ein. Dies ist möglich, da (24) auch für $\boldsymbol{T_D}$ gelten muss. Dadurch erhalten wir für das Gleichungssystem folgende Form:

$$\boldsymbol{M} \cdot \boldsymbol{g} = -\boldsymbol{b} \tag{38}$$

$$M_{i,j} = \int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho \cdot \mathrm{rot}\boldsymbol{f}_j \mathrm{d}\Omega$$

$$b_i = \int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho \cdot \boldsymbol{J} \mathrm{d}\Omega$$

Möchte man dieses Gleichungssystem aufstellen, stößt man auf ein Problem mit der Materialkonstante ρ. Außerhalb des leitfähigen Gebietes ist die Leitfähigkeit naturgemäß sehr klein, idealerweise $\rho = 0$. Das Einsetzen von $\rho = 0$ führt dazu, dass beide Seiten des Gleichungssystems gleich null wären, was zu einem nicht sinnvollen Gleichungssystem führen würde. Da uns jedoch das $\boldsymbol{E}$-Feld nicht interessiert, können wir annehmen, dass im gesamten Gebiet dieselbe Materialkonstante gilt (vorzugsweise $\rho = 1$). Dies hat zur Folge, dass wir zwar ein lösbares Gleichungssystem erhalten, jedoch muss beachtet werden, dass $\boldsymbol{T}$ nicht verwendet werden kann, um den Verlauf der elektrischen Feldlinien $\boldsymbol{E}$ zu modellieren, da das verwendete ρ nicht der physikalischen Materialkonstante entspricht. Dies ist für unsere Zwecke jedoch nicht von Belang.

Nun bleibt nur noch zu klären, woher man die Stromdichte $\boldsymbol{J}$ erhält.
Ideal wäre, wenn $\boldsymbol{J}$ als analytische-Funktion der Form $\boldsymbol{J} = f(\boldsymbol{x})$ vorliegen würde, wobei $\boldsymbol{x}$ die Koordinaten im globalen Koordinatensystem beschreibt. Dies ist aus verständlichen Gründen speziell für umfangreichere Leitergeometrien nicht einfach möglich.
Wir verwenden hierfür ein spezielles Finite-Elemente-Gitter, das $\boldsymbol{Q}$-Grid, es besteht aus spezialisierten Elementen, welche die Aufgabe haben, die Stromdichte abzubilden, die Conductor-Elemente.

3.6.1 Das Conductor-Element

Bei dem hier verwendeten Conductor-Element handelt es sich um ein klassisches finites Element, nur wird mit dessen Hilfe nun der Verlauf der Stromdichte $\boldsymbol{J}$ modelliert. Das Conductor-Element dient zur Modellierung eines Leiterteilstücks mit über dem Querschnitt gleichmäßig verteilter Stromdichte. Die Vorgabe der Geometrie erfolgt wie üblich über die Definition der Eckpunkte. Abschließend wird dem Element noch sein Strom eingeprägt.
Es wird angenommen, dass die eingeprägte Stromdichte $\boldsymbol{J}$ in Richtung der positiven lokalen ζ-Achse (Zeta-Achse) verläuft. Der konkrete Wert der Stromdichte $\boldsymbol{J}$ am Punkt $\boldsymbol{x}$ wird aus dem eingeprägten Gesamtstrom I und dem globalen Flächeninhalt der lokalen ζ-konstant-Fläche A_ζ berechnet. Hierfür muss zuerst der globale Punkt $\boldsymbol{x}$ in das lokale Koordinatensystem transformiert werden. Sein lokales Gegenstück $\boldsymbol{\xi}_0$ wird nun verwendet, um den globalen Flächeninhalt A_ζ der ζ-konstant-Fläche zu berechnen. Nachdem mithilfe des Stromes I und des Flächeninhalts A_ζ der Betrag der Stromdichte $\boldsymbol{J}$ berechnet wurde, wird dessen Richtung noch auf die globale Richtung der lokalen ζ-Achse gesetzt.

$$
\begin{aligned}
\boldsymbol{x} &= g(\boldsymbol{\xi}) \\
\boldsymbol{\xi}_0 &= g^{-1}(\boldsymbol{x}) \rightarrow \zeta_0 \\
\boldsymbol{\xi}_\zeta^T &= \begin{bmatrix} \xi & \eta & \zeta_0 \end{bmatrix} \\
\boldsymbol{A}_\zeta(\boldsymbol{\xi}) &= \int_{\xi=-1}^{1} \int_{\eta=-1}^{1} \frac{\partial g(\boldsymbol{\xi}_\zeta)}{\partial \xi} \times \frac{\partial g(\boldsymbol{\xi}_\zeta)}{\partial \eta} d\eta d\xi \\
A_\zeta(\boldsymbol{\xi}) &= |\boldsymbol{A}_\zeta(\boldsymbol{\xi})| \\
\boldsymbol{J}(\boldsymbol{\xi}) &= \frac{I}{|A_\zeta(\boldsymbol{\xi})|} \cdot \frac{\frac{\partial g(\boldsymbol{\xi}_0)}{\partial \zeta}}{\left|\frac{\partial g(\boldsymbol{\xi}_0)}{\partial \zeta}\right|}
\end{aligned}
\tag{39}
$$

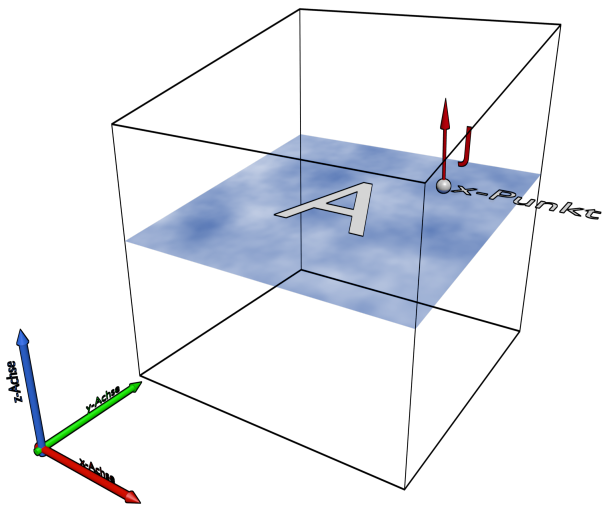


Figure 4: Skizze eines Conductor-Elements im globalen Koordinatensystem

4 Implementierung

Die Implementierung wurde in MATLAB vorgenommen. Wobei die objektorientierte Implementierung besonderes fokussiert wurde. Diese macht den Code nicht nur übersichtlicher und wartbarer, sonder erlaubt es Programmteile unabhängig voneinander zu optimieren und gegebenenfalls zu parallelisieren.

4.1 Informationsfluss

Die Arbeitsschritte, welche zur Lösung einer Problemstellung nötig sind, werden im Informationsflussdiagram (Abbildung 5) graphisch dargestellt.

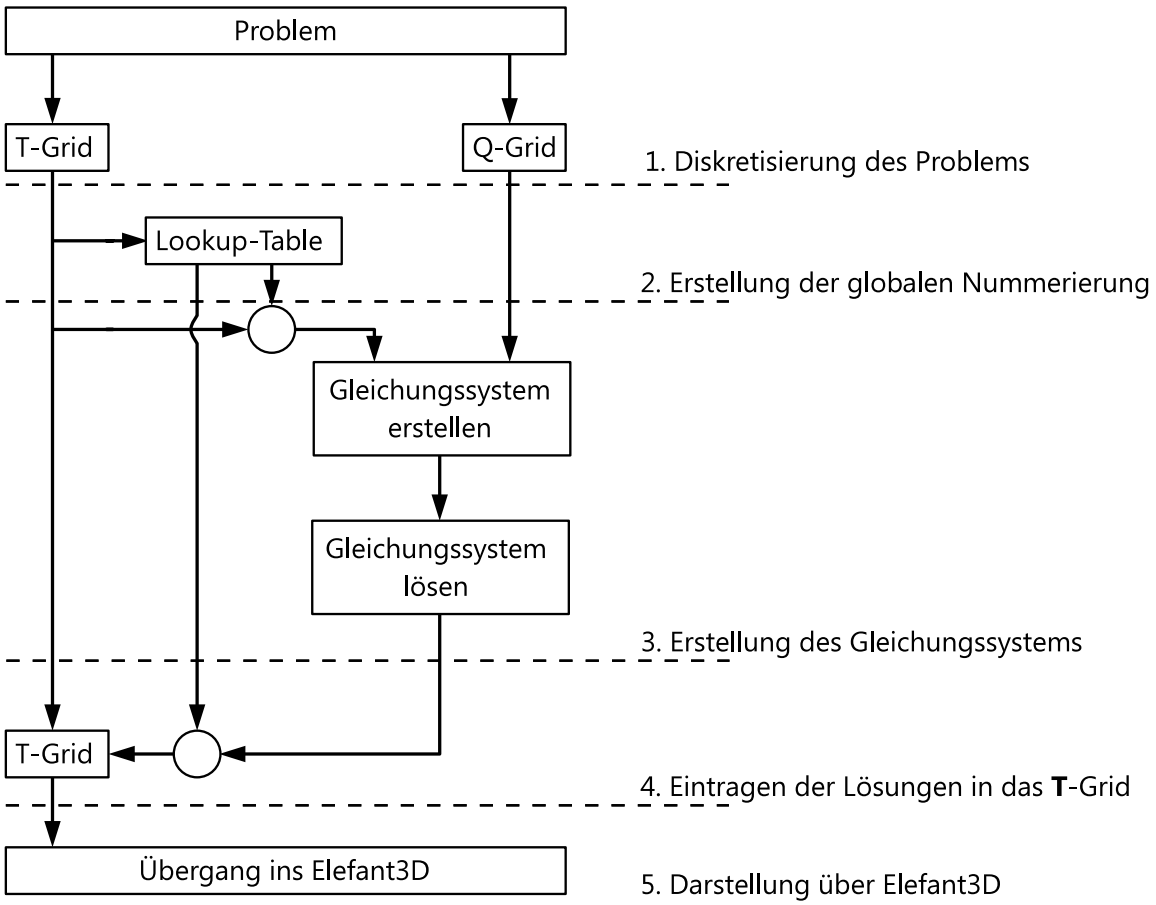


Figure 5: Informationsflussdiagramm

4.1.1 Diskretisierung des Problems

Der erste Schritt zur Lösung eines Problems besteht in der Diskretisierung desselbigen. Dabei wird zuerst die Leitergeometrie durch Conductor-Elemente genähert. Die Conductor-Elemente bilden anschließend das Q-Grid. Hierfür werden sie alle in einer Instanz des Grid-Containers zusammengefasst. Parallel dazu muss noch das gesamte Problemgebiet mit finiten Elementen abgebildet werden. Diese Elemente bilden nun das $\boldsymbol{T}$-Grid und werden ebenfalls in einer Instanz des Grid-Containers zusammengefasst.

Bei der Modellierung des Leiters sollte darauf geachtet werden, dass die angenommene Stromdichte in Richtung der lokalen ζ-Achse des Conductor-Elementes verläuft. Dies ist besonders relevant

bei Rundungen in der Leiterstruktur. Es hat sich gezeigt, dass eine 90°-Biegung durch mindestens zwei 12-knotige Element modelliert werden sollte, um eine gute Approximation zu erhalten.

Für die Erstellung des $\boldsymbol{T}$-Grids gilt Ähnliches. Die besten Ergebnisse werden erzielt, wenn das $\boldsymbol{T}$-Grid alle Kanten des $\boldsymbol{Q}$-Grids enthält (matched Grid). Außerdem sollten die Kanten dem erwarteten Feldverlauf folgen, dies die Qualität der Approximation erhöht.

4.1.2 Erstellung einer globalen Nummerierung

Die globale Kantennummerierung ist notwendig, um die einzelnen Elemente zu einem zusammenhängenden Netz zu verbinden. Ohne globale Nummerierung haben wird nur eine Sammlung an unabhängigen Elementen. Bei der globalen Nummerierung wir jeder Kante der globalen Netzstruktur eine individuelle Nummer zugewiesen. Zu beachten ist, dass zwei aneinander grenzende Elemente sich mehrere Kanten teilen. Die geteilten Kanten haben denselben räumlichen Verlauf und bekommen deshalb jeweils dieselbe globale Nummer zugewiesen. Dies ist nicht weiter verwunderlich, es handelt sich ja um dieselbe Kante in der globalen Netzstruktur.
Um den Zusammenhang zwischen lokalen und globaler Nummerierung herzustellen, wird ein Lookup-Table verwendet. Der Lookup-Table ist folgendermaßen aufgebaut:
Jede Spalte gehört zu einem Element, jede Zeile zu einer lokalen Kantennummer. In die Zellen wird die dazugehörige globale Kantennummer geschrieben. Die Struktur wird in Tabelle 1 dargestellt. Angenommen die erste Kante des ersten Elementes erhält die globale Nummer x, dann wird in die Zelle 1×1 x eingetragen. Diese Kante überschneidet sich jetzt mit der dritten Kante des zweiten Elements (beispielsweise) so erhält nun auch diese Zelle (3×2) den Eintrag x.

		Element Nummer							
		1	2	3	4	5	6	7	...
Lokale Kantennummer	1	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	...
	2	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	...
	3	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	...
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
	36	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	...

Table 1: Lookup-Table

Die Vorgehensweise beim Erstellen der Tabelle ohne a-priori Wissen über den Netzaufbau erfolgt wie in Pseudocode 1 beschrieben. Der größte Nachteil dieses Algorithmus ist vor allem die relativ lange Ausführungszeit $\mathcal{O}(n^2)$, welche durch das ständige Durchsuchen der "alten" Einträge zustande kommt. Kann man jedoch auf Vorwissen über den Netzaufbau zurückgreifen, kann man einiges an Rechenzeit einsparen bis zu einer Ordnung von $\mathcal{O}(n)$.

```
edgeCount = 1;                                  ... Initialisierung des Edge-Counters
LookUp = zeros(36, numel(Elemente));            ... Initialisierung des Lookup-Table
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:numel(Elemente), ... Schleife i alle Elemente
    for j = 1:36,          ... Schleife j alle Kanten (hier 36)
                           ...
        for ii = 1:i-1,    ...
            for jj = 1:36, ... Durchsuchen aller bereits nummerierten Elemente
                           ...
                if Element{i}.Edge(j) == Element{ii}.Edge(jj),
                           ... Sind diese Kanten ident?
                           ... z.B. gleicher Start und Endpunkt
                    LookUp(i,j) = LookUp(ii,jj); ... Wenn ja,
                                 ... Trage die bereits vorhande Nummerierung ein
                    break;     ... und brich die Suche ab
                end
            end
            if LookUp(i,j) ~= 0, ... Globale Kantennummer bereits vorhanden?
                break;           ... Wenn ja, die Suche abbrechen
            end
        end
        if LookUp(i,j) == 0,             ... Neue Kante?
            LookUp(i,j) = edgeCount;     ... Wenn ja, neue Kantennummer zuweisen
            edgeCount = edgeCount +1;    ... Edge-Counter erhoehen
        end
    end
end
```

Pseudocode 1: Erstellung des Lookup-Table ohne Vorwissen

4.1.3 Erstellung des Gleichungssystems

Da jedes Element des $\boldsymbol{T}$-Grids ein Member der Finiten-Element-Klasse ist und diese Klasse, die Methode "AssKantenGlei(QGrid)" zur Verfügung stellt, sieht dieser Abschnitt relativ unspektakulär aus.
Jedoch erkennt man gut, warum die Verwendung von Lookup-Tables sinnvoll ist, denn mit ihrer Hilfe können die elementweise generierten Teile des Gleichungssystems schnell zu einem globalen Gesamtsystem zusammengeführt werden.
Denn durch den Aufruf der "AssKantenGlei(QGrid)" Methode wird unter Verwendung des $\boldsymbol{Q}$-Grids für jedes Element des $\boldsymbol{T}$-Grids eine quadratische Martrix (linke Seite) und eine Vektor (rechte Seite) erzeugt. Diese Matrix bzw. dieser Vektor werden mithilfe der Lookup-Tables zu ihren globalen Gegenstücken addiert, siehe Pseudocode 2.

Dieser Algorithmus ist sehr gut parallelisierbar. Da auf die Lookup-Tables und das $\boldsymbol{Q}$-Grid nur lesend zugegriffen wird, können sie für diesen Abschnitt als konstant und gegeben betrachtet werden. Die Elemente des $\boldsymbol{T}$-Grids können ebenfalls in beliebig viel Teilsysteme zerlegt werden, diese Teile können dann separat berechnet werden (Abbildung 6).
Dadurch spielt die Rechenzeit für diesen Abschnitt bei der Gesamtperformance eine untergeordnete Rolle. Ein Schema für eine mögliche Parallelisierung ist in Abbildung 6 dargestellt. Die Lösung des Gleichungssystems erfolgt mittels des Konjugierten-Gradienten-Verfahrens.

```
edgeCount                                          ... Gesamtanzahl der Kanten

M = sparse(EdgeCount,EdgeCount);                   ... Initialisierung der linken Seite
b = sparse(EdgeCount,1);                           ... Initialisierung der rechten Seite
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:numel(TGrid.Elemente)                    ... Schleife durch alle Elemente des
                                                   ... T-Grids
[lhs, rhs ~] = TGrid.Elemente{i}.AssKantenGlei(QGrid);
                                ... Jedes Element erstellt seine lokalen Gleichungen

M(LookUp(:,i),LookUp(:,i)) = M(LookUp(:,i),LookUp(:,i)) + lhs;
b(LookUp(:,i))             = b(LookUp(:,i)) + rhs;
                            ... Die lokalen Gleichungen werden zum globalen
                            ... Gleichungssystem einsortiert
end

val = bicg(M,-b,1e-12,10000);    ... Loesen des Gleichungssystems
```

Pseudocode 2: Erstellung des Gleichungssystems

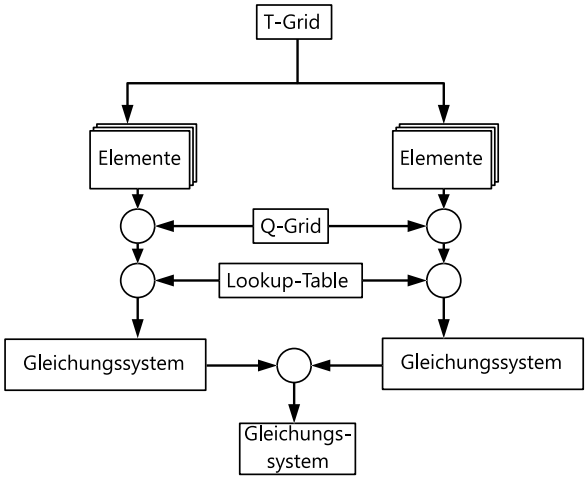


Figure 6: Mögliche Parallelisierung bei der Erstellung des Gleichungssystems

4.1.4 Eintragen der Lösung in das T-Grid

Zum Abschluss werden die Lösungen noch in das $\boldsymbol{T}$-Grid eingetragen. Die Zuordnung zwischen globaler Kantennummer und Element-bezogener-lokaler-Kantennummerierung erfolgt wieder über den Lookup-Table. Durch die MATLAB-Syntax gestaltet sich diese Zuweisung besonders einfach, siehe Pseudocode 3.

```
val                                    ... Loesungsvektor des Gleichungssystems
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:numel(TGrid.Elemente)        ... Schleife durch alle Elemente des
                                       ... T-Grids
TGrid.Elemente{i}.A_E = val(LookUp(:,i))';
                                       ... Kantengewichte in Elementliste eintragen
end
```

Pseudocode 3: Eintragen der Lösungen

4.2 Die FiniteElement-Klasse

Die FiniteElemente-Klasse stellt den Grundstock der Implementierung dar. Sie soll ein finites Element symbolisieren. Bei der Erstellung dieser Klasse wurde vor allem auf die Parallelisierung der Abfragen geachtet. Dies war wichtig, um die Rechenzeit gering zu halten. Nachfolgend werden die wichtigsten Methoden erklärt.

4.2.1 CalcL26

Die Berechnung des $\boldsymbol{L_{26}}$-Vektors ist überaus wichtig, da er den Grundstock für viele Berechnungen darstellt. Die Formfunktionen setzen sich aus einer Linearkombination der lokalen Koordinaten und deren Potenzen bzw. Kombinationen aus selbigen zusammen. CalcL26 ist eine Funktion, die aus einem lokalen Koordinaten Tripel $\boldsymbol{\xi} = [\xi, \eta, \zeta]^T$ alle benötigten Potenzen und deren Kombinationen berechnet und in einen Vektor einsortiert, dem $\boldsymbol{L_{26}}$-Vektor.
Dieser Vektor wird später unter anderem verwendet, um die Formfunktionen auszuwerten. Bei der von mir implementierten Variante können die $\boldsymbol{L_{26}}$-Vektoren für mehrere lokale Positionen gleichzeitig berechnet werden. Dafür müssen die lokalen Koordinaten nur in einer Matrix übergeben werden. Eine Spalte der Matrix stellt hierbei ein lokales Koordinaten-Tripel dar. Man erhält daraufhin eine Matrix, deren Spalten die einzelnen $\boldsymbol{L_{26}}$-Vektoren darstellen. Pseudocode 4 zeigt die Vorgehensweise beim Erstellen des Vektors.

```
1  xi            ... Matrix mit lokalen Koordinaten - Koordinaten-Tupel spalteweise
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  function value = CalcL26(Xi)
4      x = Xi(1,:);            ... Berechnung einiger Gruppen, welche mehrmals
5      y = Xi(2,:);            ... verwendet werden
6      z = Xi(3,:);
7      xy= x .* y;
8      xz= x .* z;
9      yz= y .* z;
10     xx= x .* x;
11     yy= y .* y;
12     zz= z .* z;
13
14     ... Erstellung des L_26 Vektors (Matrix)
15     value = [...                          Eintrag
16             ones(size(x));               ... 1
17             x;                           ... 2
18             y;                           ... 3
19             z;                           ... 4
20             ...
21             ];
22 end
```

Pseudocode 4: Berechnung des $\boldsymbol{L_{26}}$-Vektor(s)

4.2.2 GetGlobalPos

Durch diese Methode wird die Geometrie-Formfunktion $\boldsymbol{X} = \boldsymbol{X_0} \cdot \boldsymbol{M} \cdot \boldsymbol{L_{26}}(\boldsymbol{\xi})$ ausgewertet. Diese dient zur Umrechnung der lokalen Koordinaten in die globalen. Eine Mehrfachabfrage ("gleichzeitige"Umrechnung mehrerer Punkte) ist möglich, hierfür übergibt man der Funktion anstatt des Koordinaten-Vektors eine Koordinaten-Matrix. Jede Spalte dieser Matrix stellt ein lokales Koordinaten-Tripel dar. Der Rückgabewert der Funktion liefert die globalen Koordinaten im selben Format.

4.2.3 CalcNablaL26

Genauso essentiell wie der $\boldsymbol{L_{26}}$-Vektor ist auch sein Gradient $\boldsymbol{\nabla L_{26}}$. Da der Gradient eines Vektors jedoch eine Matrix ist, stellt die Parallelisierung (Ermöglichung von Mehrfachabfragen) eine Herausforderung dar. Bei der von mir gewählten Implementierung, beschrieben in Pseudocode 5, ist der Rückgabewert ein Tensor dritter Stufe mit der Dimension $(3 \times 3 \times N)$ wobei N die Anzahl der Koordinaten-Tupel darstellt. Einer der Vorteile dieser Implementierung liegt darin, dass der Zugriff auf die einzelnen Gradienten sehr kompakt und schnell erfolgen kann.

```
xi            ... Matrix mit lokalen Koordinaten - Koordinaten-Tupel spalteweise
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function value = CalcNablaL26(xi)
    x = xi(1,:);    y = xi(2,:);    z = xi(3,:);
    x2 = x*2;       y2 = y*2;       z2 = z*2;
    xx = x.^2;      yy = y.^2;      zz = z.^2;
    xy = x.*y;      xz = x.*z;      yz = y.*z;
    ichi = ones(size(x));   zero = zeros(size(x));

    dx =    [... nach xi
        zero;       ...1
        ichi;       ...2
        zero;       ...3
        ...
               ];
    dy =    [... nach eta
        zero;       ...1
        zero;       ...2
        ichi;       ...3
        ...
                ];
    dz =    [... nach zeta
        zero;       ...1
        zero;       ...2
        zero;       ...3
        ...
        ];

        ... Bilden des Tensors durch Reshaping
    value = [...
            reshape(dx,[26,size(x)]),...
            reshape(dy,[26,size(y)]),...
            reshape(dz,[26,size(z)])...
            ];
end
```

Pseudocode 5: Erstellung des $\boldsymbol{\nabla L_{26}}$-Tensors

4.2.4 GetJacobi

Mit dieser Methode wird die Jacobi-Matrix der Geometrieformfunktion berechnet. Genau wie bei CalcNablaL26 wird auch hier ein Tensor verwendet, um Mehrfachabfragen zu behandeln. Da MATLAB jedoch keine Tensormultiplikation in der von mir gewünschten Form unterstützt, muss hier über eine Schleife gearbeitet werden. Diese For-Schleife könnte zur Verbesserung der Performance noch parallelisiert werden (M Punkte - N CPUs, jede CPU berechnet $\frac{M}{N}$ der Punkte).

4.2.5 GetLokalPos

Die Umrechnung der globalen in die lokalen Koordinaten erfolgt über den Gauß-Newton Algorithmus, wie er in [4] beschrieben wird. Die Vorgehensweise dieses Algorithmus kann wie folgt zusammengefasst werden:
Von einer lokalen Startposition ausgehend wird die entsprechende globale Position ermittelt. Die Differenz der gesuchten Position und der berechneten, stellt die neue globale Suchrichtung dar, diese wird genutzt, um die neue lokale Suchrichtung zu bestimmen. Die Umrechnung zwischen der globalen und der lokalen Suchrichtung erfolgt über die Jacobi-Matrix bzw. in diesem Fall ihrer Inversen. Anschließend wird mithilfe der lokalen Suchrichtung die neue lokale Position ermittelt und so wird iterativ die lokale Position gesucht (siehe Pseudocode 6).

```
x           ... Matrix mit globalen Koordinaten - Koordinaten-Tupel spaltenweise
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function value = GetLokalPos(obj, x)
    value = zeros(size(x));
    for i = 1:size(x,2)     ... Zur Bearbeitung von Mehrfachabfragen
        tempX = x(:,i);     ... Auswahl des aktuellen Punktes
        l = [0;0;0];        ... Startwert fuer die Iteration

        for ii = 1:obj.MaxInter     ... Beginnn Iteration
            diff = tempX - obj.GetGlobalPos(l); ... Bestimmung der Differenz
            if norm(diff)<= obj.GlobalEps,      ... Ueberpruefe Abbruchkriterium
                value(:,i) = l;
                break;
            end
        J = obj.GetJacobi(l);          ... Berechnung der Jacobi-Matrix
        l = l +  J \ diff;             ... und des naechsten Iterationsschrittes
        end

        if ii >= obj.MaxInter,
            value(:,i) = NaN * ones(3,1);
            ... Setzt den Rueckgabewert auf  "Not a Number",
            ... wenn der Algorithmus nicht konvergiert
        end
    end
end
```

Pseudocode 6: Implementierung des Algorithmus zur Berechnung der lokalen Position

4.2.6 CalcBetween

Wie in der Beschreibung erwähnt, berechnet diese Methode, ob sich ein Punkt innerhalb eines Quaders befindet. Verwendung findet diese Funktion bei der Überprüfung, ob sich ein Punkt in einem Element befindet, oder nicht. Dies ist durch die Tatasche möglich, dass ein Element im lokalen Koordinatensystem immer einen Würfel darstellt, welcher sich vom Punkt $[-1,-1,-1]$ zum Punkt $[1,1,1]$ erstreckt. Pseudocode 7 zeigt die Implementierung.

```
Values    ... Zu ueberpruefende Punkte --- Spalte ist ein Tupel
Min       ... unterer Begrenzungspunkt
Max       ... oberer Begrenzungspunkt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function value = CalcBetween(Min,Max,Values)
    value = sum(...
        repmat(Min,[1,size(Values,2)]) <= Values & ...
        Values <= repmat(Max,[1,size(Values,2)])   ...
        ,1) == 3;
end
```

Pseudocode 7: Implementierung der CalcBetween-Methode

4.2.7 IsInside

Die Performance dieser Funktion hat sich als überaus wichtig herausgestellt, da sie verhältnismäßig häufig aufgerufen wird. Bei der Implementierung wurde besonderes Augenmerk auf die Effizienz bei Mehrfachabfragen gelegt. Der Ablauf kann wie folgt zusammen gefasst werden: Zuerst wird überprüft, ob sich die Punkte in einem Quader befinden, welcher das Element umschließt. Für alle Punkte innerhalb des umschließenden Quaders werden nun die lokalen Koordinaten berechnet. Zum Abschluss wird überprüft, ob sich die lokalen Koordinaten innerhalb des Elementes befinden. Natürlich würde es ohne Vorauswahl durch den umschließenden Quader auch funktionieren, aber zur Berechnung der lokalen Koordinaten wären viel mehr Iterationen nötig, welche die Performance senken (Pseudocode 8).

```
x     ... Zu ueberpruefende Punkte --- Spalte ist ein Tupel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [flag, xi] = IsInside(obj, x)
    flag = false(1,size(x,2));      ... Initialisierung der Inside-Flags
    xi = NaN(size(x));              ... Initialisierung der lokalen Koordinaten
                                    ... fuer die Rueckgabe
    ... Befindet sich der Punkt im umschliessenden Quader?
    routh = obj.CalcBetween(obj.box(:,1),obj.box(:,2),x);
    ... Auswahl der Punkte innerhalb des Quaders
    xi(:,routh) = obj.GetLokalPos(x(:,routh));
    ... Befinden sich diese Punkte innerhalb des Elementes?
    flag(routh) = obj.CalcBetween(-ones(3,1)-FiniteElement.GlobalEps,...
                                  ones(3,1)+FiniteElement.GlobalEps,...
                                  xi(:,routh));
    xi(:,~flag) = NaN; ... Zuweisung des Rueckgabewertes fuer ...
end                    ... alle Punkte ausserhalb des Elements
```

Pseudocode 8: Implementierung der IsInside-Methode

4.2.8 CrossSection

Die Berechnung der Querschnittfläche erfolgt über ein Flächenintegral. Dieses Integral bestimmt den Flächeninhalt der ζ-konstant-Fläche des Elements am Punkt $\boldsymbol{\xi}$. Zur Auswertung des Integrals wird die Gauß-Quadratur verwendet (siehe Pseudocode 9).

```
Xi    ... Zu ueberpruefende Punkte --- Spalte ist ein Tupel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function value = CrossSection(obj, Xi)
    value = zeros(size(Xi));     ... Initialisierung der Werte
    for ii = 1:size(Xi,2),       ... Schleife durch alle Punkte zur Mehrfachabfrage
        tempXi = Xi(:,ii);       ... Auswahl eines temporaeren Punktes

                              ... Bestimmung der Eckpunkte und Seitenmittelpunkte
                              ... der Zeta-Konstant-Flaeche
        l = FiniteElement.LokalNodePositions(:, [13:16 22:25]) +...
                             repmat([0;0;tempXi(3)],1,8);
        A = obj.GetGlobalPos(l);

                             ... Vorbereitung zur Gaussintegration
        gau = FiniteElement.Gaus3_2;
        L = obj.CalcNablaL26(gau(1:3,:));
        Fak = reshape(gau(4,:),1,1,9);

                              ... Gaussintegration
        erg = zeros(1,1,9);
        for i = 1:size(Fak,3),
            JJ = A * FiniteElement.M2_8 * L(:,1:2,i);
            erg(:,1,i)  = Fak(:,:,i) * sqrt(det(JJ'*JJ));
        end
        temp = sum(erg,3);

                              ... Bestimmung der globalen Zeta-Richtung
        dir = obj.GetJacobi(tempXi);
        dir = dir(:,3);
                             ... Erzeugung des Ausgangswertes
        value(:,ii) = temp * dir/norm(dir);
    end
end
```

Pseudocode 9: ζ-Konstant-Flächenquerschnittberechnung

4.2.9 AssKantenGlei

In dieser Funktion wird ein Teil des Gleichungssystems aufgestellt. Die zu implementierenden Gleichungen sind für die linke Seite:

$$M_{ij} = \int_\Omega \operatorname{rot} \boldsymbol{f}_i \cdot \rho \operatorname{rot} \boldsymbol{f}_j \, d\Omega$$

und für die rechte Seite:

$$b_i = \int_\Omega \operatorname{rot} \boldsymbol{f}_i \cdot \rho \, \boldsymbol{J} \, d\Omega$$

für alle $i, j \in \{1, 2, 3, \cdots 36\}$
Zur Erstellung der rechten Seite benötigen wir die Stromdichte $\boldsymbol{J}$ an jedem Gaußpunkt. Diese wird vom $\boldsymbol{Q}$-Grid über die Methode "**GetDensity**" (Unterpunkt 4.3) zur Verfügung gestellt. Pseudocode 10 zeigt die konkrete Implementierung unter Ausnützung der Symmetrie-Eigenschaften.

```
Source  ... Q-Grid zur Berechung des rechten Seite
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [lhs, rhs, UsedGausPoints] = AssKantenGlei(obj, Source)
    lhs = zeros(36);     ... Initialisierung der Returnvalues
    rhs = zeros(36,1);

    ... Laden der vorberechneten Gausspunkte und deren Gewichte
    gaus = obj.Gaus{count}.Gaus;

    %% Vorbereitung der Integration
    J = obj.GetJacobi(gaus(1:3,:));
        ... Abfrage der Stromdichte aus dem Q-Grid
    S = reshape(Source.GetDensity(obj.GetGlobalPos(gaus(1:3,:))),...
            1,3,size(gaus,2));

    R = obj.GetGlobalRotKantenFormfunktion(gaus(1:3,:));
    Fak = reshape(gaus(4,:),1,1,size(gaus,2));
        ... Berechnung der gewichteten Jacobi-Determinante
    for i = 1:size(J,3),
        Fak(:,:,i)  = Fak(:,:,i) * det(J(:,:,i));
    end
        ... Integration --- unter Ausnuetzung der Symmetrieeigenschaft
    for i = 1:36,
        for j = i:36,
            lhs(i,j) = sum(dot(R(i,:,:),obj.Mat*R(j,:,:)).*Fak,3);
        end
        rhs(i) = -sum(dot(R(i,:,:),obj.Mat*S(:,:,:)).*Fak,3);
    end

    for i = 1:36-1,
        for j = i+1:36,
            lhs(j,i) = lhs(i,j);
        end
    end
end
```

Pseudocode 10: Aufstellung des Gleichungssystems

4.3 Die Conductor-Klasse

Die Conductor-Klasse dient zum Modellieren eines Leiters. Sie leitet sich direkt von der Finiten-Elemente-Klasse ab und erweitert diese um die Methode "**GetDensity**". Diese Methode ermöglicht es die Stromdichte $\boldsymbol{J}$ an einem oder mehreren Punkten im globalen Koordinatensystem abzufragen. Hierfür wird zuerst überprüft, ob sich der Punkt überhaupt im Element befindet. Wenn ja, wird über den gegeben Gesamtstrom und die ζ-konstant-Querschnittfläche die Stromdichte berechnet (Pseudocode 11)

```
x  ... globale Koordinaten des Punktes / der Punkte
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [found, value] = GetDensity(obj, x)
    value = zeros(size(x)); ... Initialisierung

    [found, xi] = obj.IsInside(x);   ... Ist der Punkt im Element, wenn ja -- wo
    area = obj.CrossSection(xi(:,found));    ... Berechnung der Flaeche

        ... Berchnung von J
    value(:,found) = repmat(...
                        repmat(obj.Current,1,size(area,2)) ./ sum(area.^2,1)...
                        ,3,1) .* area;
end
```

Pseudocode 11: Abfrage der Stromdichte

4.4 Der Grid-Container

Wie der Name schon verrät, stellt diese Klasse einen Container dar. Die Aufgabe dieses Containers ist es, die Elemente der Grids (***T*** und ***Q***) aufzubewahren und einige Methoden-Aufrufe an alle Elemente weiterzugeben.
Die einzig komplexere Methode ist "**GetDensityGrid**" (Pseudocode 12), welche die Stromdichte auswertet. Sie fragt beim ersten Conductor-Element nach, ob dieses für den betreffenden Punkt zuständig ist. Wenn ja, bekommt es die Stromdichte geliefert. Befindet sich der Punkt außerhalb des betroffenen Elements, wird das nächste Element befragt. Haben sich alle Elemente für nicht zuständig erklärt, liegt der Punkt außerhalb des Leiters und hat somit keine Stromdichte.

```
x  ... globale Koordinaten des Punktes / der Punkte
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [value] = GetDensityGrid(obj, x)
    value = zeros(size(x)); ... Initialisierung
    N = size(x,2);          ... Wieviele Punkte werden gesucht
    found = false(N,1);     ... Welche wurden bereits gefunden
        ... Start Suche durch alle Elemente
    for i = 1:obj.Count,
                ... Befindet sich der Punkt im i-ten Element?
        [f, v] = obj.elements{i}.GetDensity(x(:,~found));
                ... Auswahl der noch zu suchenden Punkte
        temp = false(N,1);
        temp(~found) = f;
        value(:,temp) = v(:,f);
        found(~found) = f;

                ... Wenn alle Punkt gefunden wurden, Suche abbrechen
        if sum(~found) == 0
            break;
        end
    end
end
```

Pseudocode 12: Abfrage der Stromdichte aus dem Q-Grid

4.5 Schnittstelle zu Elefant3D

Um die Ergebnisse graphisch darzustellen, wird der Elefant3D "**FieldView**" verwendet. Zur Kommunikation werden die "**InputFiles**" von Elefant genutzt. Um beispielsweise die Geometrie eines Grids anzuzeigen, muss man die Geometrieinformationen in das "minp01.inp"-File schreiben. Die Geometrieinformationen werden als ASCII-Zeichenfolge in folgender Kodierung ins File geschrieben:

```
#ElementAnzahl      ... Anzahl der Makro-Elemente des gesamten Gitters
[x] [y] [z]         ... Koordinaten-Tupel von Element 1 Knoten 1
[x] [y] [z]         ... Koordinaten-Tupel von Element 1 Knoten 2
...                 ... Die Koordinaten der restlichen Knoten des Elements
[Nx] [Ny] [Nz]      ... Die Teilung von Makroelementes 1
[x] [y] [z]         ... Koordinaten-Tupel von Element 2 Knoten 1
[x] [y] [z]         ... Koordinaten-Tupel von Element 2 Knoten 2
...                 ... Die Koordinaten der restlichen Knoten des Elements
[Nx] [Ny] [Nz]      ... Die Teilung von Makroelement 2
...                 ... Restlichen Elemente des Grids
XX                  ... Sonstige Eingaben bezueglich des Problems
0                   ... Ending Zeros
```

Im Anschluss wird die "minp01.exe" ausgeführt, wodurch die vom FieldView benötigten Files geschrieben werden. Startet man nun FieldView, kann man die Geometrie betrachten. Möchte man das berechnete $\boldsymbol{T}$-Grid graphisch darstellen, ist es nötig die Gewichte der Kantenformfunktionen zu übertragen.

Zum Übertragen der Kantenformfunktionsgewichte, sprich der Lösung des Gleichungssystems, dient das "cint.dat"-File. Dies ist ein Binärfile, in dem neben einer Prüfsumme noch die Gewichte der einzelnen Kantenformfunktionen geschrieben werden. Zusätzlich muss noch das Einstellungsfile "Setup.ini"-File richtig konfiguriert werden. Dies erfolgt jedoch nur durch das Überschreiben mit einer Vorlage.

Um die Erstellung der Input-Files für den Benutzer komfortabel zu gestalten, wird vom Grid-Container eine Methode zur Verfügung gestellt, welche das File generiert. Diese Methode schreibt die Informationen der einzelnen Grid-Elemente in der richtigen Reihenfolge in das jeweilige File.

5 Anwendung

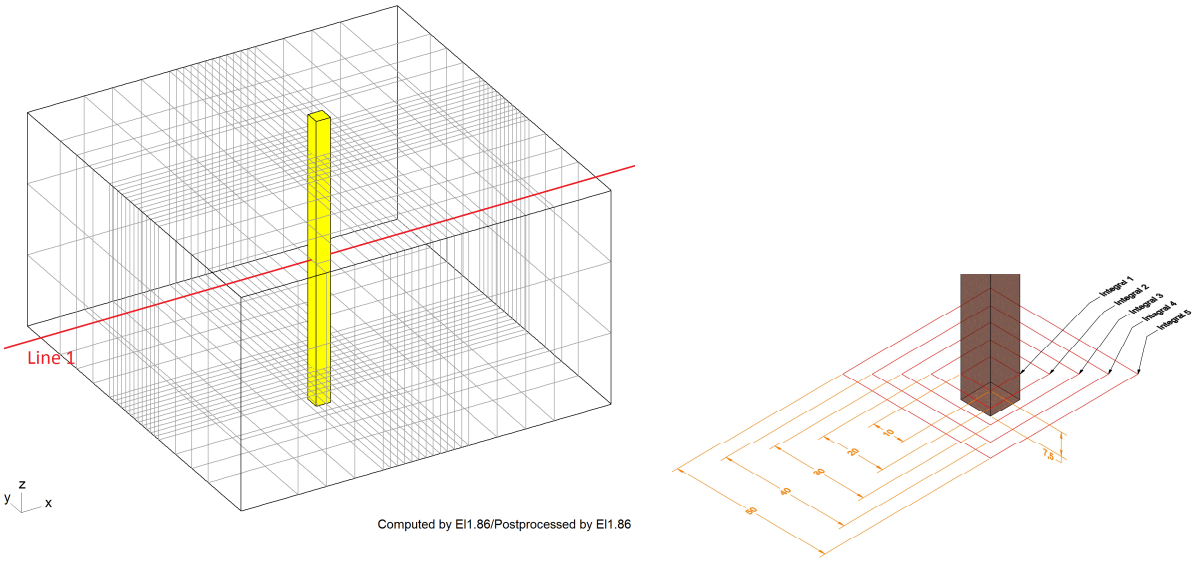
In diesem Schritt erfolgt die Erprobung des Verfahrens anhand einiger Beispiele. Um die Qualität der Lösung beurteilen zu können, werden neben den "FieldView"-Plots noch einige Linienintegrale um den Leiterquerschnitt berechnet. Diese sollten aufgrund des Satzes von Stoke [3] dem durch den Leiter fließenden Strom entsprechen. Der durch das Integrale berechnete Strom wird auf den vorgegebenen Strom bezogenen (40). Der prozentualen Fehler wird in einem Diagramm dargestellt.

$$e_i = \left| \frac{I_{\text{Soll}} - I_{\text{Integral}_i}}{I_{\text{Soll}}} \right| \cdot 100\% \tag{40}$$

Weiter wird $|\boldsymbol{J}|$ entlang einer Linie ausgewertet und ebenfalls in einem Diagramm dargestellt. Bei diesen Diagrammen erkennt man gut die Position der Leiter entlang der Linie.

5.1 Gerader Leiter mit Matched Meshes

Dieses Beispiel soll die grundsätzliche Funktionsfähigkeit der Methode zeigen. Deshalb wurde eine triviale Geometrie gewählt, deren Lösung bekannt ist.
Das Problem besteht aus einem geraden Leiter quadratischen Querschnitts. Die Seitenlänge des Leiters beträgt 1 cm daraus folgt eine Querschnittsfläche von 0.0001 m^2
Prägen wir dem Leiter einen Strom von 1 A ein, erhalten wir eine Stromdichte von $10000\frac{\text{A}}{\text{m}^2}$.
Das Gitter, in dem das $\boldsymbol{T}$-Grid modelliert werden soll, wurde der Leitergeometrie angepasst ("**Matched Mesh**"). Das Mesh besteht aus $20\times20\times3$ Elementen. Eine graphische Darstellung der Anordnung inklusive Mesh ist in Abbildung 7 zu sehen.

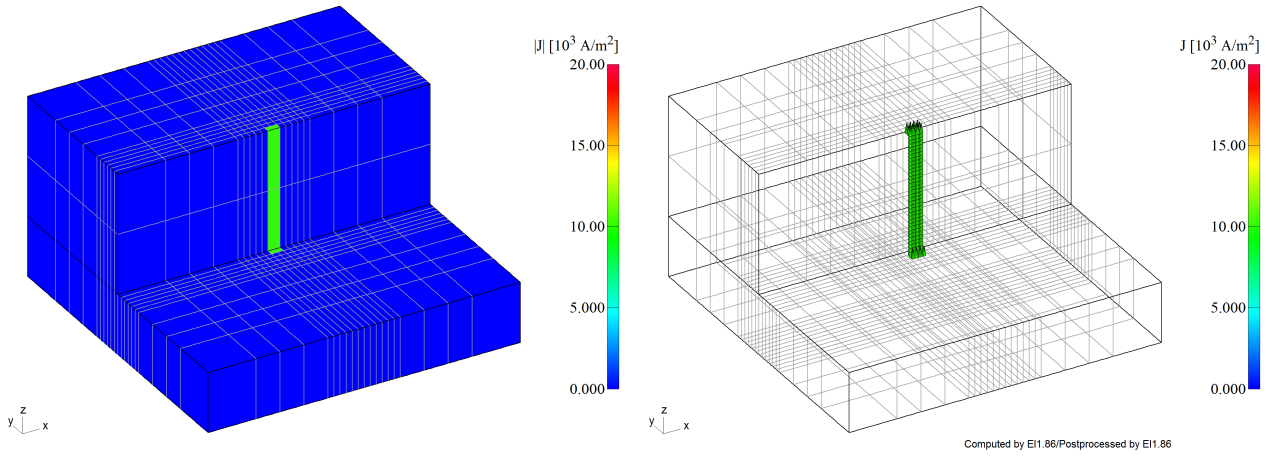


(a) (b)

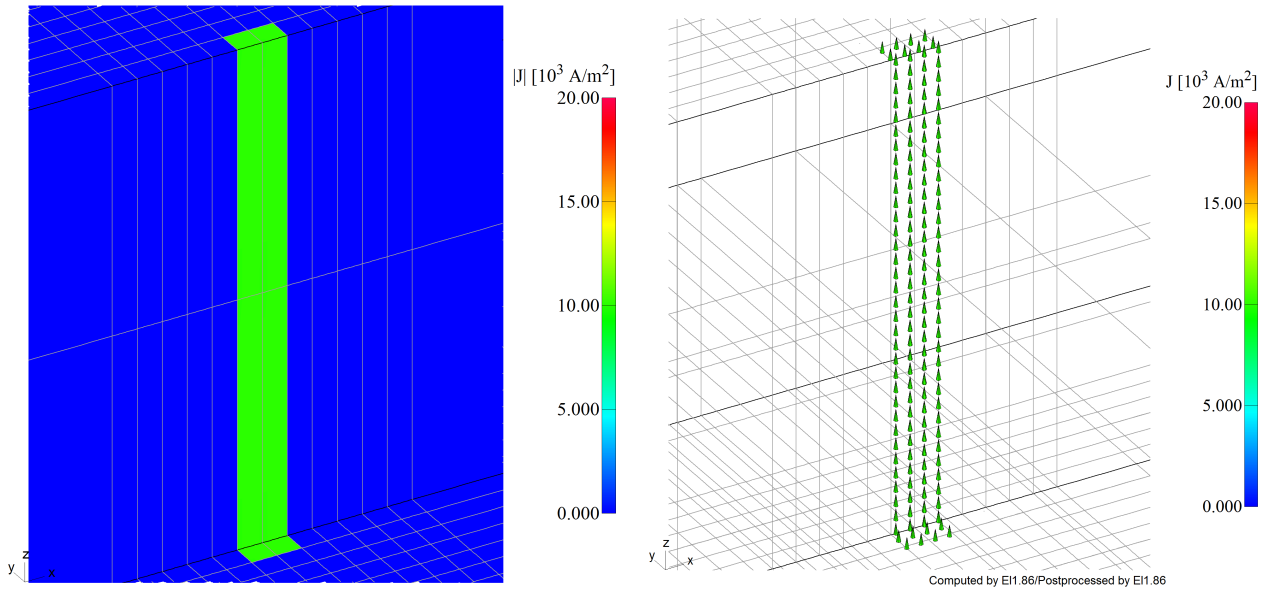
Figure 7: Problemgeometrie
a) Leiter und Mesh
b) Bemaßung und Integralpfade

Die graphische Auswertung wird in Abbildung 8 gezeigt. Bereits in ihr erkennt man die hohe Qualität dieser Berechnung. Die Auswertung der Linienintegrale, zu sehen in Abbildung

9(b), eliminiert die letzten Zweifel. Man beachte, dass ein Fehler von $10^{-12}\%$ die numerische Auflösung des IEEE-64 Bit-Fließkomma-Typs darstellt, da diese bei ca. 16 signifikanten Stellen liegt.

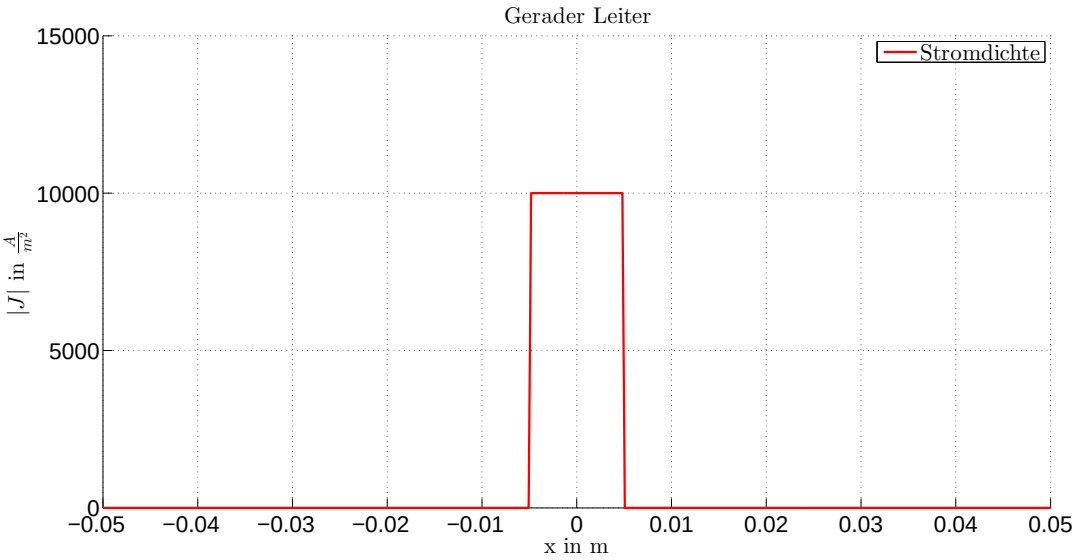


(a) (b)

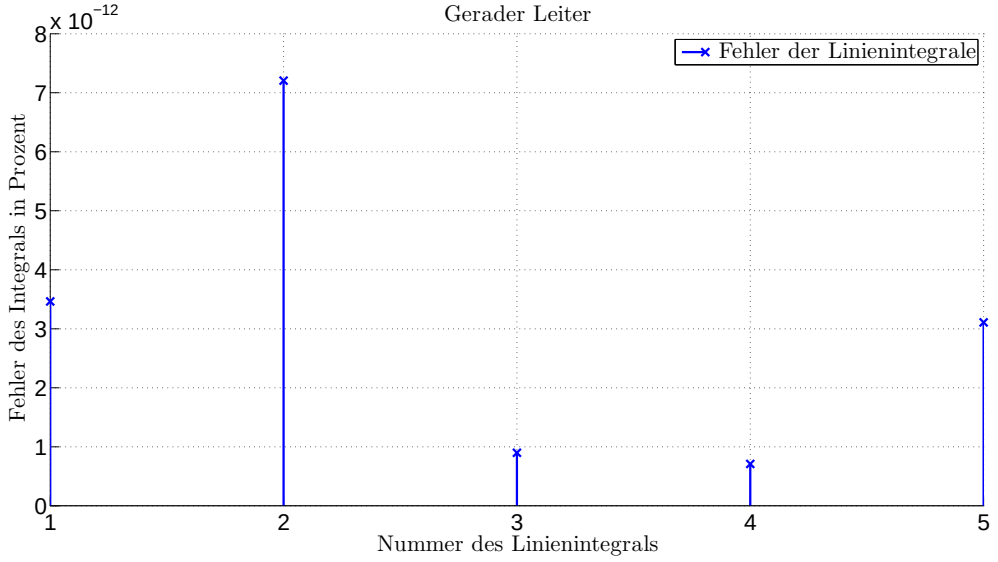


(c) (d)

Figure 8: Graphische Darstellung der Stromdichte $\boldsymbol{J}$
- **a)** Skalarplot von $|\boldsymbol{J}|$
- **b)** Vektorpfeilansicht von $\boldsymbol{J}$
- **c)** Detailansicht des Skalarplots
- **d)** Detailansicht der Vektorpfeilansicht



(a)

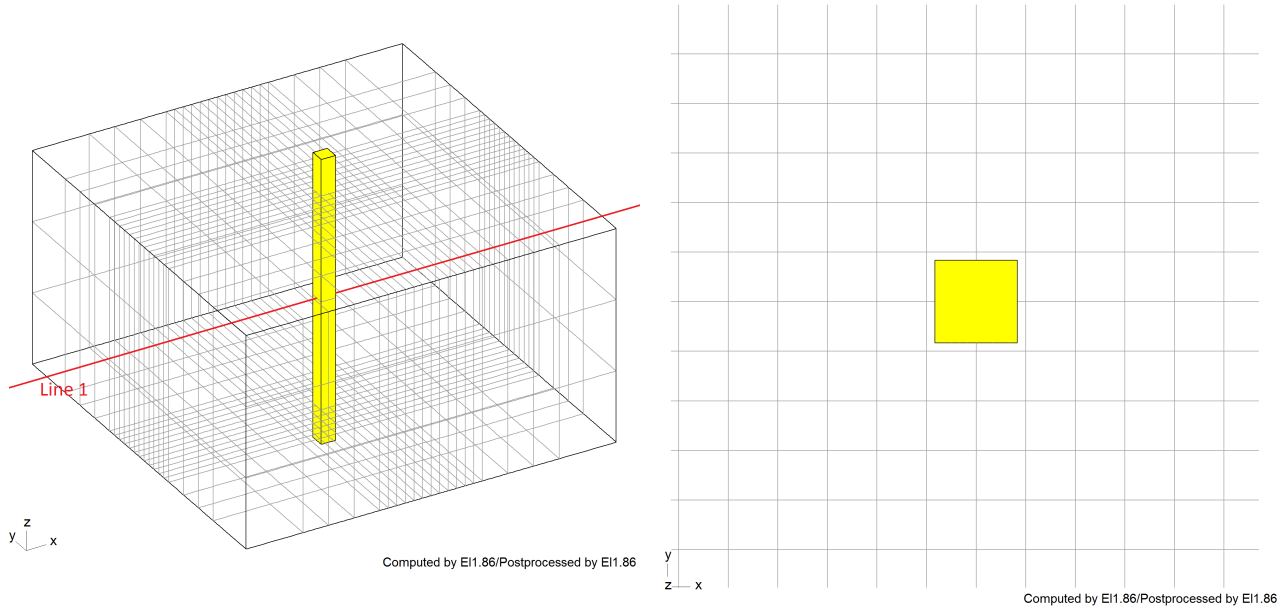


(b)

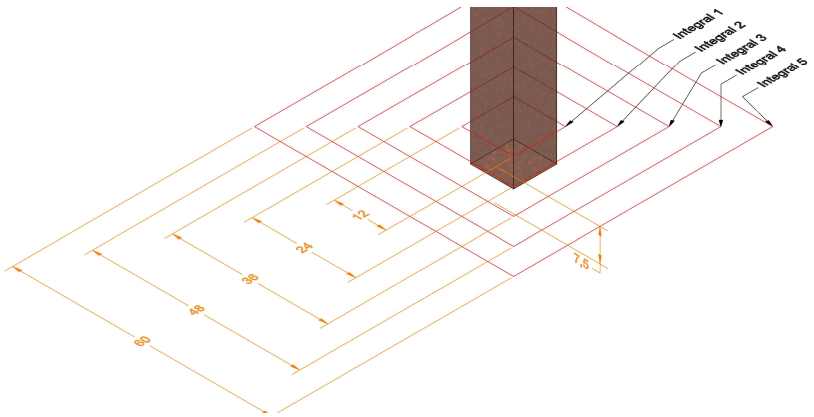
Figure 9: Graphische Darstellung der Stromdichte $\boldsymbol{J}$
a) Linienplot – Auswertung von $|\boldsymbol{J}|$ entlang der Geraden *Line 1*
b) Fehlerberechnung des Integrals in Prozent

5.2 Gerader Leiter ohne Matched Mesh

In diesem Beispiel wird ein gerader Leiter mit quadratischen Querschnitt (Seitenlänge 1 cm) von einem konstanten Strom mit 1 Ampere durchflossen. Die daraus resultierende Stromdichte beträgt $10000\frac{\mathrm{A}}{\mathrm{m}^2}$. Das Mesh, in welchem das $\boldsymbol{T}$-Grid angezeigt wird, wurde nun nicht dem Problem angepasst. Die Meshes des $\boldsymbol{Q}$- (Modellierung des Leiters) und des $\boldsymbol{T}$-Grids (für $\boldsymbol{T}$-Feld) besitzen nun nicht übereinstimmende Kanten. Für dieses Beispiel wurde ein sehr ungünstiger Fall angenommen. Der Leiter liegt nun innerhalb der vier Zentralelemente einer Modellierungsebene (siehe Abbildung 10(b)). Wir wenden nun denselben Algorithmus an, wie in Beispiel 5.1, um die Lösung zu berechnen.



(a) (b)



(c)

Figure 10: Problemgeometrie
a) Leiter und Mesh
b) Detailansicht
c) Bemaßung und Integralpfade

Die graphische Auswertung der Skalar- und Vektorpfeilansicht sehen auf den ersten Blick recht vielversprechend aus. Doch die vielen Artefakte der Vektorpfeilansicht stellen eine Warnung dar. Denn betrachtet man die Auswertung der Integrale, erkennt man, dass der Integrationsfehler

des Linienintegrals knapp unter 25% liegt. Betrachtet man Abbildung 12(a) wird erkenntlich, dass die Stromdichte im Zentrum des Leiters die größte Abweichung aufweist. Der Grund dafür liegt daran ist dass innerhalb eines Elementes sich die Stromdichte nur stetig ändern kann (Eigenschaft der Formfunktionen). Der Strom den wir modellieren ändert sich jedoch sprunghaft und kann somit nur bedingt Abgebildet werden.

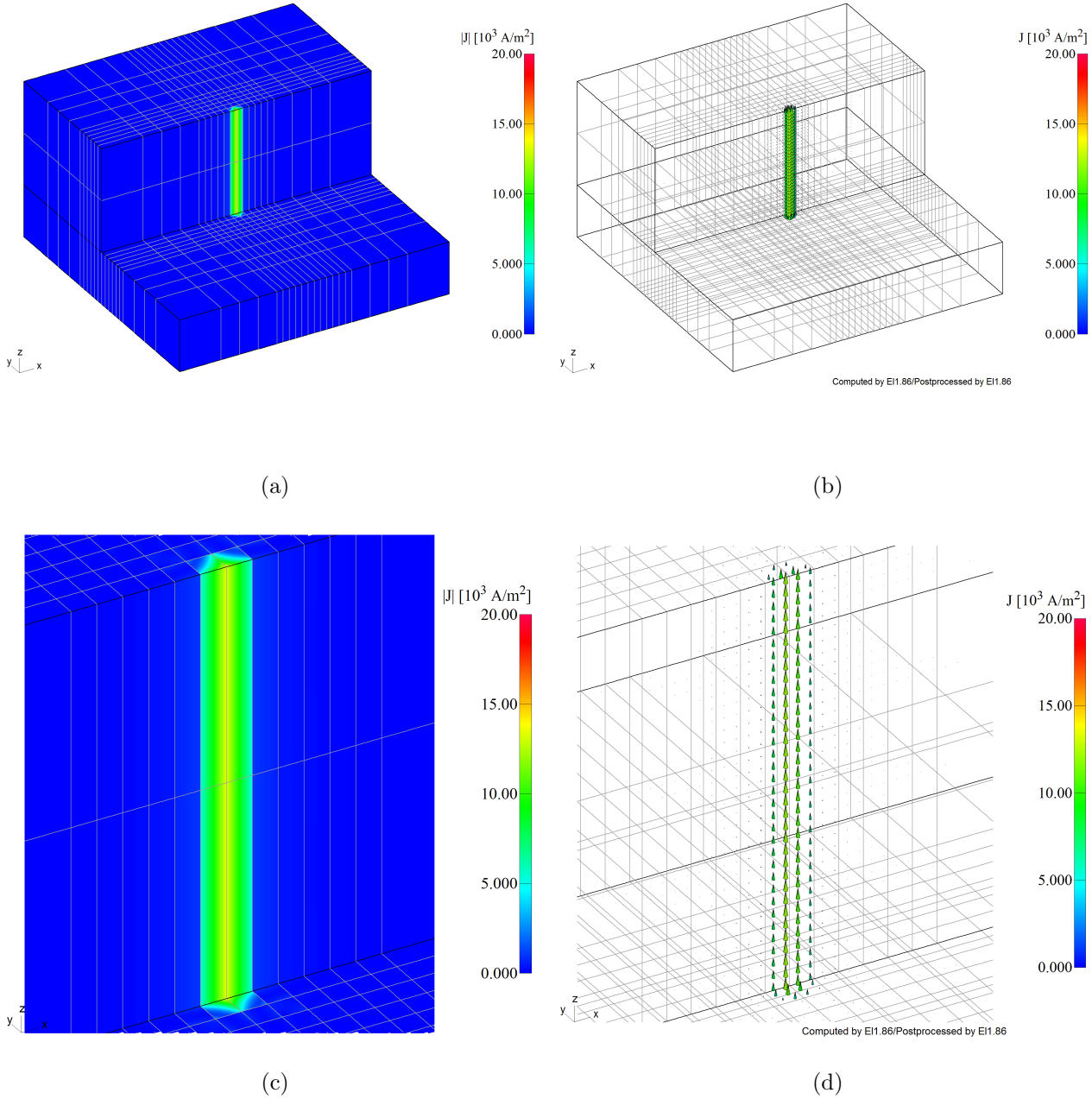


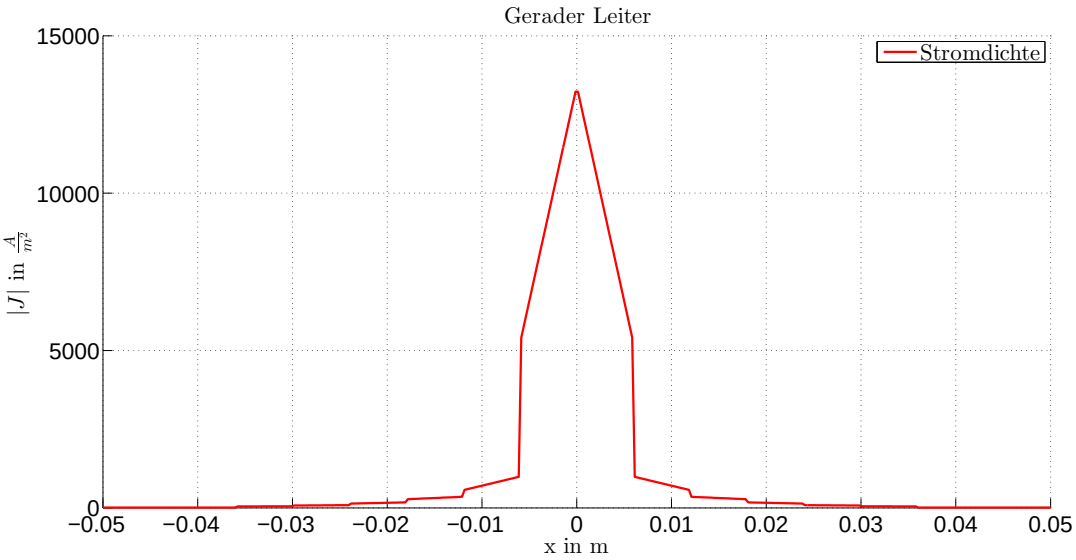
Figure 11: Graphische Darstellung der Stromdichte $\boldsymbol{J}$}

a) Skalarplot von $|\boldsymbol{J}|$

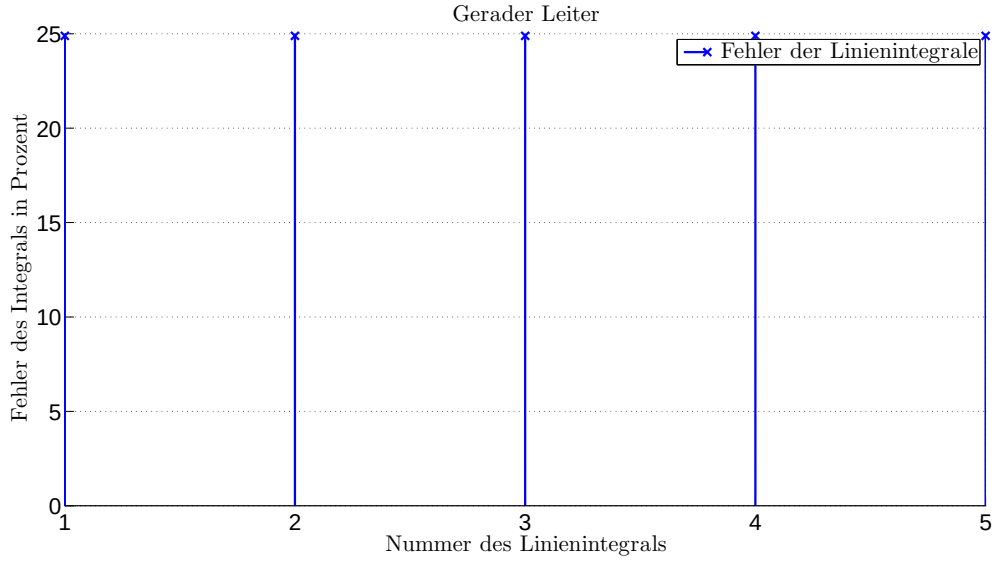
b) Vektorpfeilansicht von $\boldsymbol{J}$

c) Detailansicht des Skalarplots

d) Detailansicht der Vektorpfeilansicht



(a)



(b)

Figure 12: Graphische Darstellung der Stromdichte $\boldsymbol{J}$
a) Linienplot – Auswertung von $|\boldsymbol{J}|$ entlang der Geraden *Line 1*
b) Fehlerberechnung des Integrals in Prozent

5.2.1 Fehlersuche

Ein so hoher Fehler kann in den meisten Anwendungsgebieten nicht akzeptiert werden.
Eine der möglichen Fehleruhrsachen besteht in der Erstellung der rechten Seite des Gleichungssystems.

$$b_i = \int_\Omega \mathrm{rot}\boldsymbol{f}_i \cdot \rho\, \boldsymbol{J}\, d\Omega$$

Bei der numerischen Integration mittels Gaußquadratur können die Kantenformfunktionen $\boldsymbol{f}_i$ bzw. deren Rotation $\mathrm{rot}\boldsymbol{f}_i$ exakt genähert werden. Die Stromdichte $\boldsymbol{J}$ hingegen kann nur schlecht angenähert werden, da sie sich in diesem Bereich gänzlich anderen Formfunktion unterwirft. Dadurch ergibt sich ein Integrationsfehler.

Zur Behebung ergeben sich mehrere Vorgehensweisen:

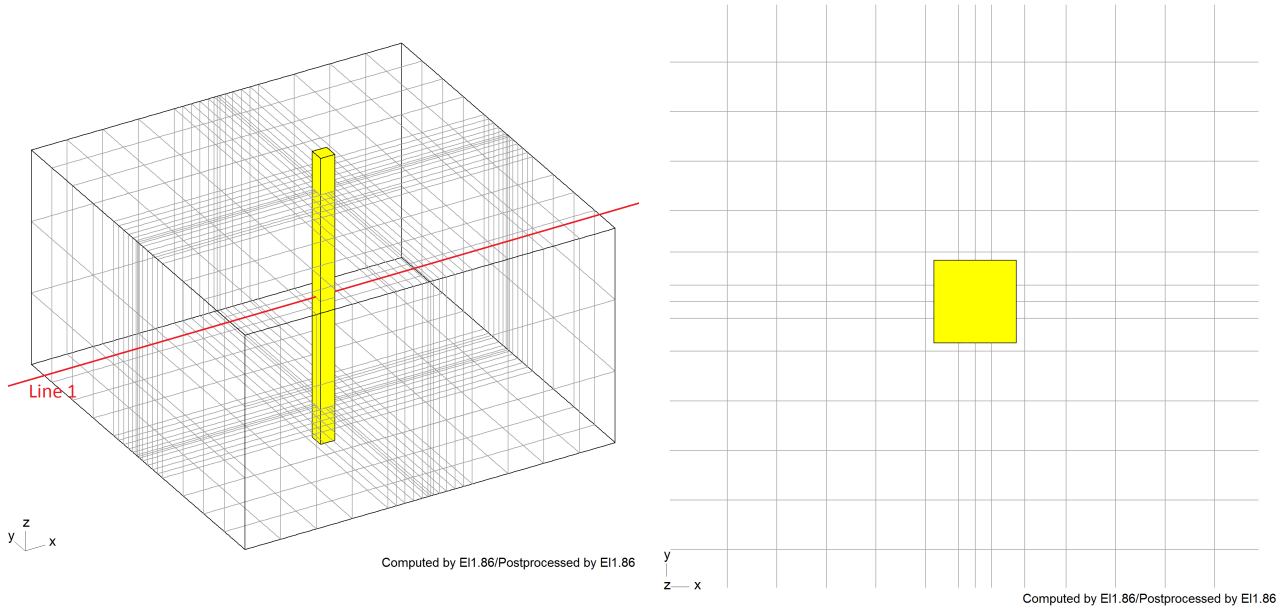
- **Gitteranpassung**
  Anpassung des $\boldsymbol{T}$-Grids, sodass die Kanten übereinstimmen, damit sich die Stromdichte denselben Formfunktionen unterwirft. Damit erhalten wir dieselbe Ausgangslage wie in Punkt 5.1 und auch dasselbe Ergebnis.

- **Gitterverfeinerung**
  Um den Verlauf der Stromdichte besser nähern zu können, könnte ich die Anzahl der finiten Elemente in diesem Bereich erhöhen und so eine besser Näherung erzielen. Dies kann das Ergebnis verbessern. Jedoch bedeutet dies eine neue Generierung des Gitters — Wenn ich schon ein neues Gitter erzeugen muss, könnte ich auch gleich ein Mached-Grid generieren und so lande ich wieder beim Punkt Gitteranpassung.

- **Verbesserung der Integrationsmethode**
  Die Genauigkeit des Integrals ergibt sich aus der Anzahl der Stützstellen. 27 Stützstellen sind für die von uns gewählten Kantenformfunktionen ausreichend. Jedoch hält sich $\boldsymbol{J}$ nicht an diese und so könnte man die Anzahl der Gaußpunkte erhöhen und so versuchen die Lösung zu verbessern.

5.3 Verfeinerung des Gitters

Zur Reduzierung des Fehlers aus Beispiel 5.2 wurde nun das Mesh verfeinert. Der Leiter Befindet sich nun in dem zentralen 4×4 Block einer Modellierungsebene, wieder wurde absichtlich eine schlechte Aufteilung gewählt (Abbildung 13). Wie in Abbildung 15(b) unschwer zu erkennen ist konnte das Ergebnis deutlich verbessert werden. Wieder ist in der Stromdichte die markante Überhöhung erkennbar.



(a) (b)

Figure 13: Problemgeometrie
a) Leiter und Mesh
b) Detailansicht

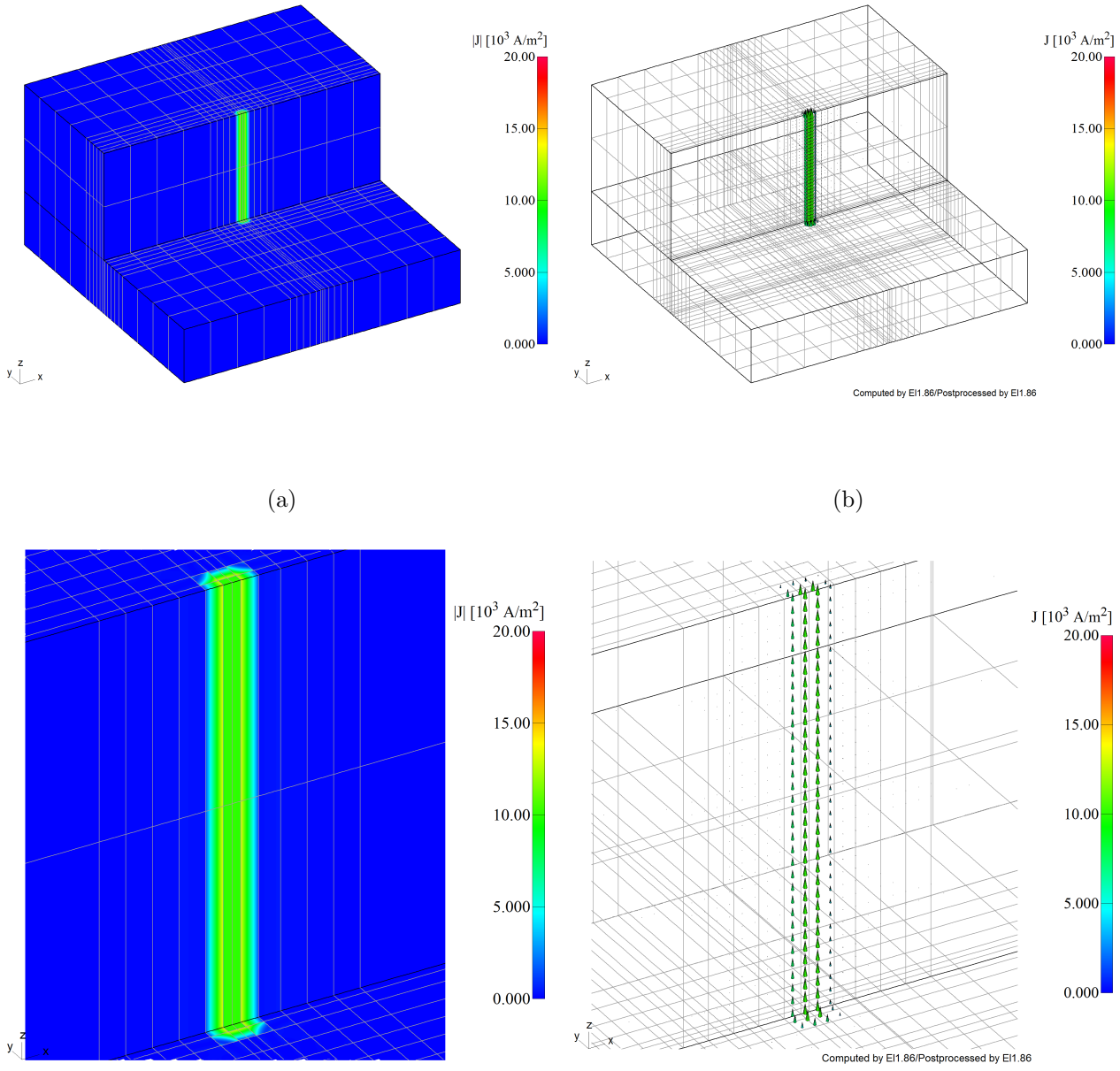


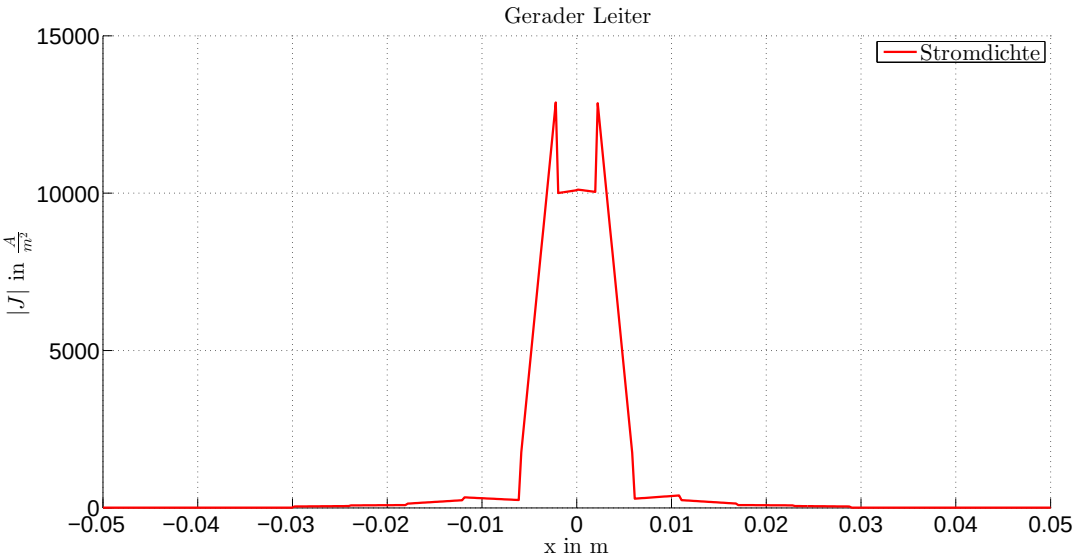
Figure 14: Graphische Darstellung der Stromdichte $\boldsymbol{J}$}

a) Skalarplot von $|\boldsymbol{J}|$

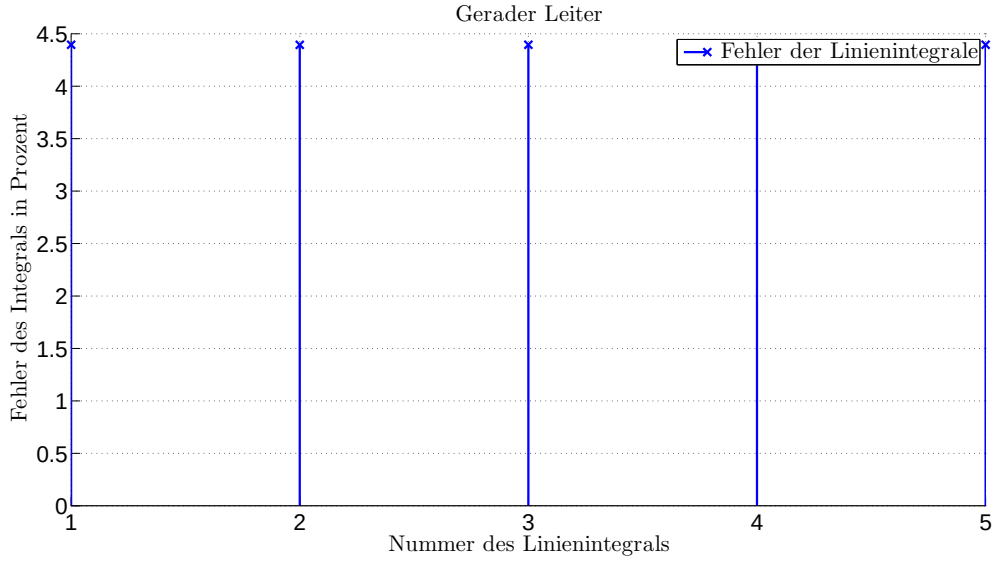
b) Vektorpfeilansicht von $\boldsymbol{J}$

c) Detailansicht des Skalarplots

d) Detailansicht der Vektorpfeilansicht



(a)

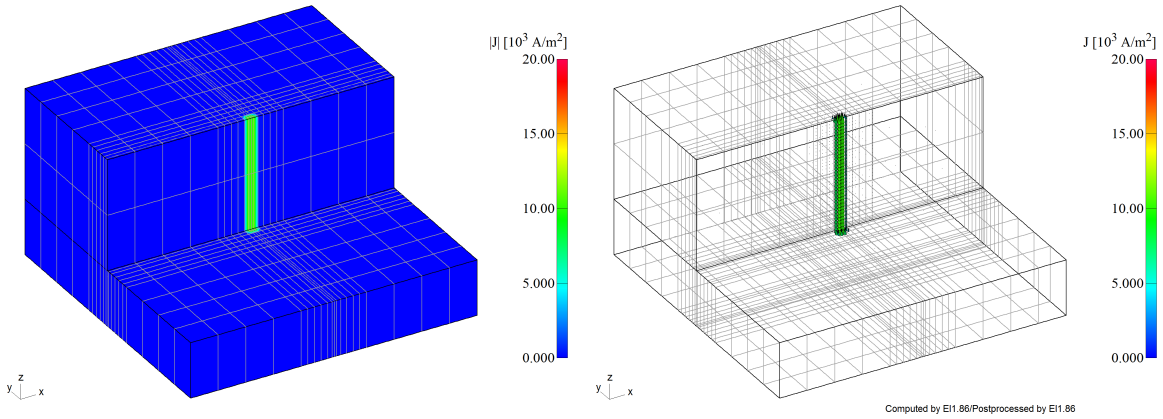


(b)

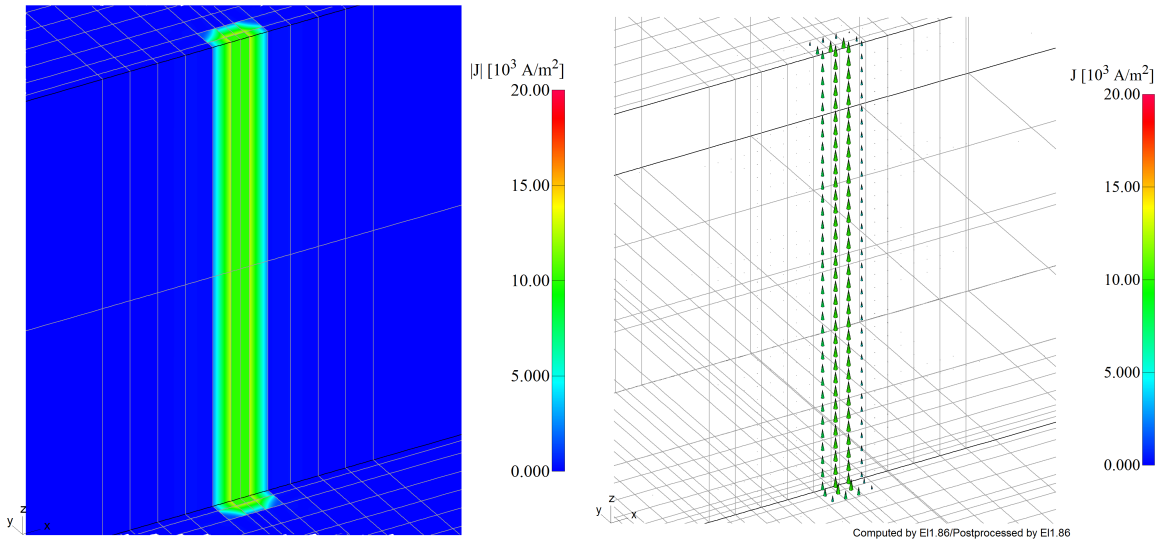
Figure 15: Graphische Darstellung der Stromdichte $\boldsymbol{J}$
a) Linienplot – Auswertung von $|\boldsymbol{J}|$ entlang der Geraden *Line 1*
b) Fehlerberechnung des Integrals in Prozent

5.3.1 Neuberechnung mit erhöhter Gaußpunktanzahl

Zur Verbesserung des Ergebnisses wurde nun die Anzahl der Gaußpunkte in jenen Elementen erhöht, in denen ein Leiter zugegen ist.
Bereits die graphische Auswertung (Abbildung 16) zeigt, dass im Vektorplot weniger Artefakte vorhanden sind und die Auswertung der Linienintegrale (Abbildung 17(b)) bestätigt dies. Der Fehler konnte von ca. 4% auf unter 1% gesenkt werden.

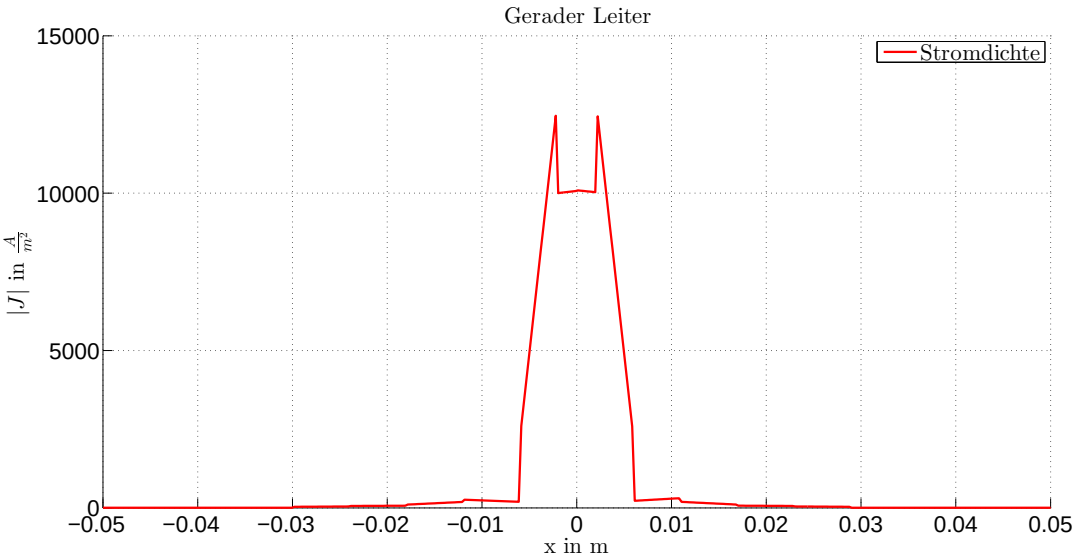


(a) (b)

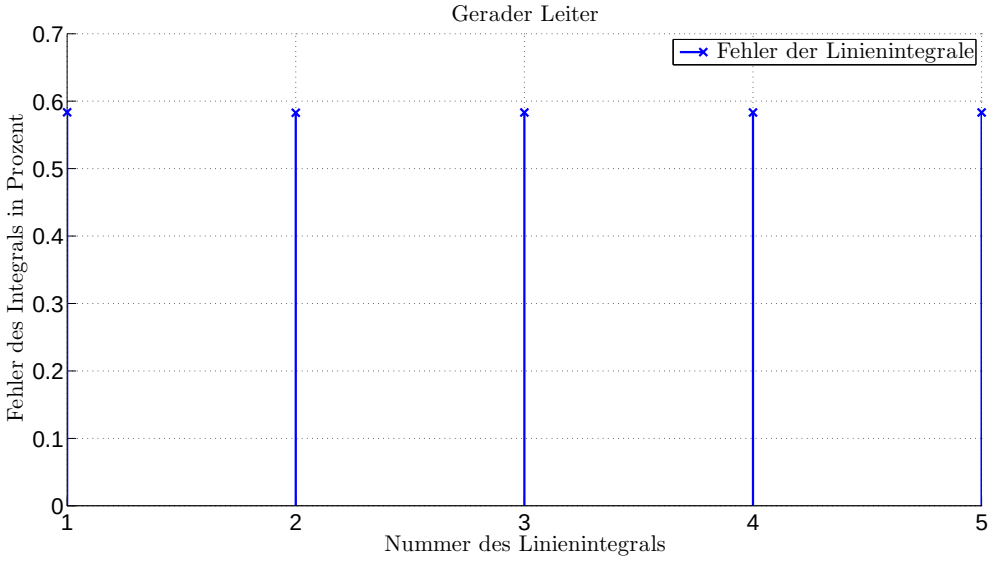


(c) (d)

Figure 16: Graphische Darstellung der Stromdichte $\boldsymbol{J}$
- **a)** Skalarplot von $|\boldsymbol{J}|$
- **b)** Vektorpfeilansicht von $\boldsymbol{J}$
- **c)** Detailansicht des Skalarplots
- **d)** Detailansicht der Vektorpfeilansicht



(a)

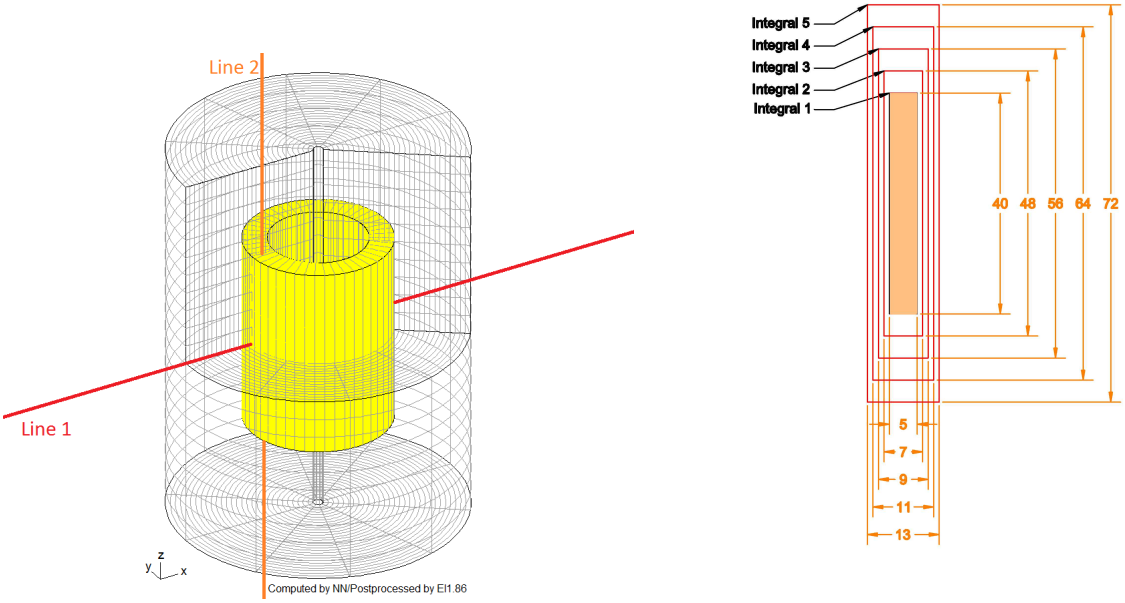


(b)

Figure 17: Graphische Darstellung der Stromdichte $\boldsymbol{J}$
a) Linienplot – Auswertung von $|\boldsymbol{J}|$ entlang der Geraden *Line 1*
b) Fehlerberechnung des Integrals in Prozent

5.4 Zylinderspule unmatched Meshes

Das Quellenfeld ($\boldsymbol{T}$-Feld) einer Zylinderspule mit einem Innendurchmesser von 2 cm, einem Außendurchmesser von 3 cm und einer Höhe von 4 cm soll modelliert werden. Wie in Abbildung 18 zu erkennen ist, sind die Gitter unangepasst. Der Leiter ist mit 47 Elementen sehr fein modelliert. Das T-Grid ist mit nur 10 Unterteilungen pro Umdrehung im Vergleich eher grob. Das Ergebnis hingegen ist überaus klar, sowohl in der graphischen Auswertung als auch in der Auswertung der Integale sind die Ergebnisse ausreichend gut. Dies liegt vor allem daran, dass die Gitter zwar nicht gleich, aber sehr ähnlich sind.



(a) (b)

Figure 18: Graphische Darstellung der Problemgeometrie
a) Leiter und Mesh
b) Bemaßung und Integralpfade

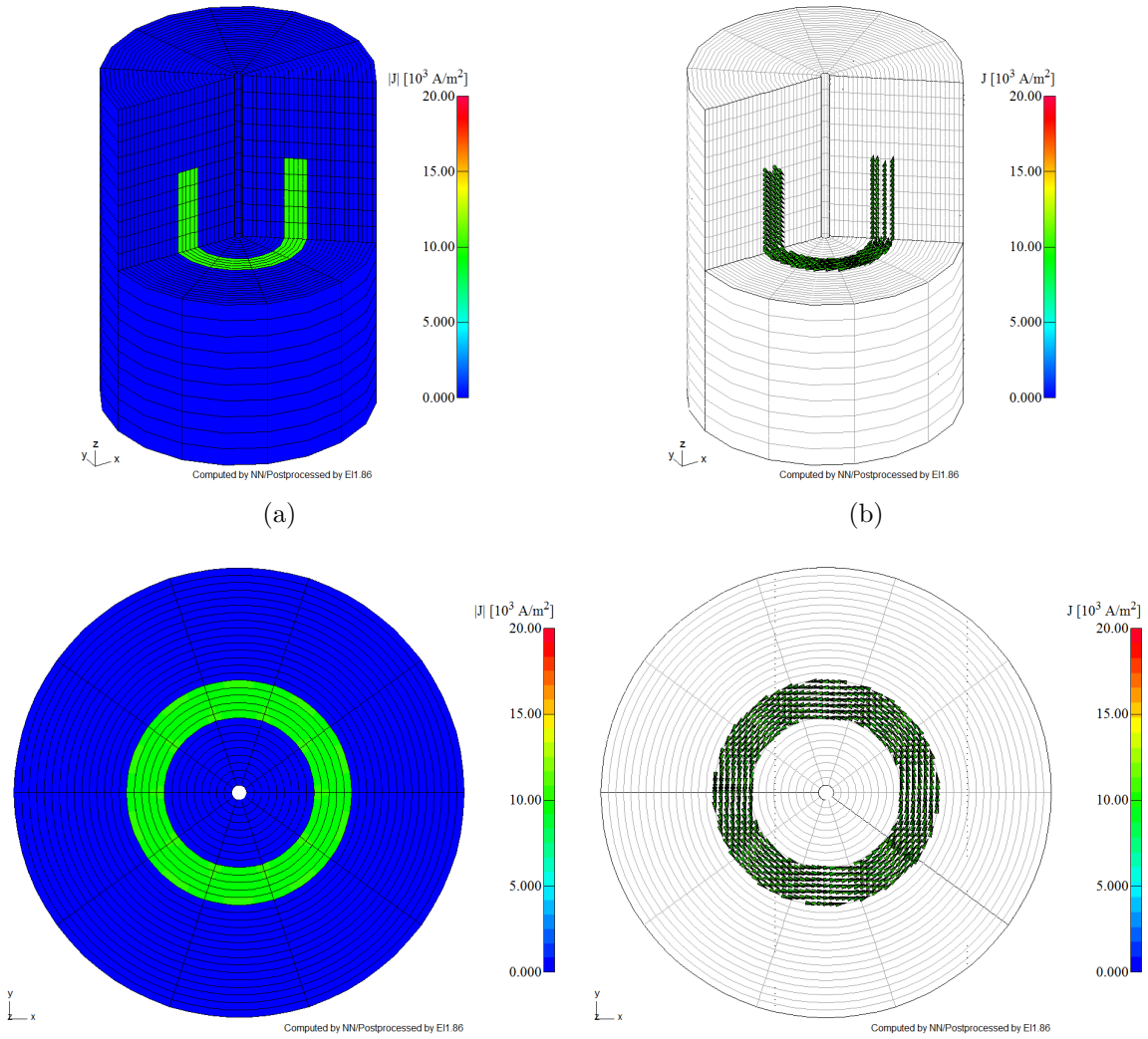


Figure 19: Graphische Darstellung der Stromdichte $\boldsymbol{J}$

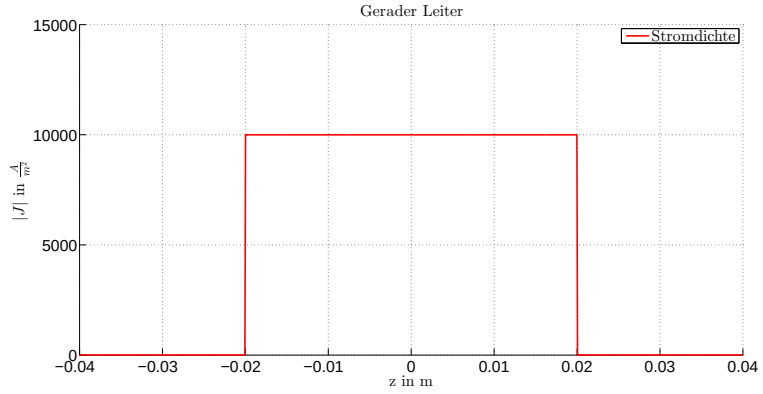
a) Skalarplot von $|\boldsymbol{J}|$

b) Vektorpfeilansicht von $\boldsymbol{J}$

c) Draufsicht eines Schnittes durch die Spule im Skalarplot

d) Draufsicht eines Schnittes durch die Spule in der Vektorpfeilansicht

(a)



(b)

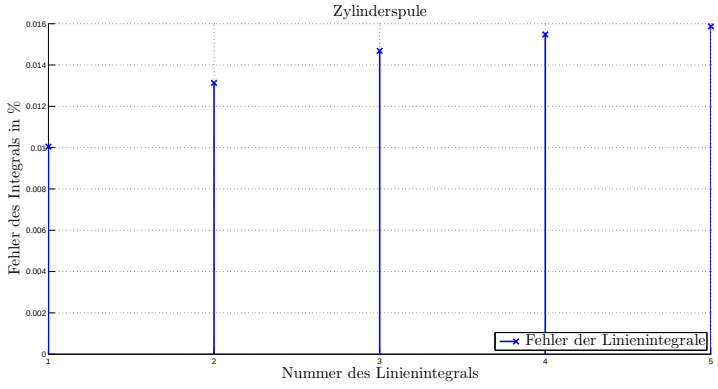


Figure 20: Graphische Darstellung der Stromdichte $\boldsymbol{J}$
a) Linienplot – Auswertung von $|\boldsymbol{J}|$ entlang der Geraden *Line 1*
b) Linienplot – Auswertung von $|\boldsymbol{J}|$ entlang der Geraden *Line 2*
c) Fehlerberechnung des Integrals in Prozent

6 Diskussion

In dieser Arbeit wurde das direkte Verfahren zur Berechnung des Quellenfeldes beschrieben. Die Vorteile dieses Verfahrens liegen darin, dass zur Berechnung des $\boldsymbol{T}$-Feldes nicht das Biot-Savart-Feld im gesamten Gebiet berechnet werden muss. Weiters kann man unter Umständen auch auf das Modellieren des Leiters verzichten. Sofern das $\boldsymbol{T}$-Grid nur fein genug ist. "Fein" deshalb, damit der Integrationsfehler beim Erstellen der rechten Seite des Gleichungssystems nicht zu groß wird.

Denn wie in den Beispielen gezeigt wird, führt eine ungenaue oder unzureichende Auflösung des $\boldsymbol{T}$-Grids im Bereich der Leiter zu Integrationsfehlern beim Erstellen der rechten Seite des Gleichungssystems. Zu den Symptomen dieser Fehler gehören:

- Wirbel im $\boldsymbol{T}$-Feld außerhalb des Leiters
- Erhöhte Stromdichte an Elementgrenze innerhalb des Leiters
- Schlechte Approximation des Leitergesamtstromes.

Diese Fehler treten auf da die Formfunktionen nur stetige Stromänderungen darstellen können. Die Stromdichte welche approximiert werden soll kann sich jedoch beliebig ändern. Durch Erhöhung der Gaußpunkte kann die Approximation einer nicht stetigen Änderung verbessert werden, jedoch steigt der Berechnungsaufwand mit $\mathcal{O}(n^3)$. Zielführender ist ein gezielte Verfeinung des Gitters in den Problemgebieten.

In Zukunft müsste noch untersucht werden, durch welche Verfahren der Integrationsfehler auf der rechten Seite minimiert werden könnte.
Zum Beispiel würde es sich anbieten Formfunktionen einzuführen welche eine nicht stetige Änderung der Stromdichte innerhalb des Elementes erlauben und so eine bessere Approximation gewährleisten könnten.

References

[1] O. Biro and K. Preis. An edge finite element eddy current formulation using a reduced magnetic and a current vector potential. *IEEE Transactions on Magnetics*, 36(5):3128–3130, Sept 2000.

[2] O. Biro, K. Preis, G. Vrisk, K. R. Richter, and I. Ticar. Computation of 3-d magnetostatic fields using a reduced scalar potential. *IEEE Transactions on Magnetics*, 29(2):1329–1332, Mar 1993.

[3] David J. Griffiths. *Elektrodynamik*. Pearson, 2011.

[4] Rolf Isermann and Marco Münchhof. *Identification of Dynamic Systems*. Springer, 2011.