Inhaltsverzeichnis

1 Einleitung

1.1 Zielsetzung

Ziel des Projektes ist es, die Kernverluste in induktiven Bauelementen besser vorhersagen zu können als mit den bisherigen Methoden. Das typische Vorgehen bei der Abschätzung dieser Verluste basiert derzeit meist auf der Steinmetz-Formel bzw. auf ihren Erweiterungen. Diese beschreibt rein phänomenologisch die Verluste abhängig von Flussdichte, Frequenz und Temperatur. Dabei wird jedoch weder die Strom- noch die Kernform einbezogen.

1.2 Hysteresemodelle

Um hier eine Verbesserung erzielen zu können, ist es notwendig, die Hysterese transient zu beschreiben und mit diesem Modell die Verluste zu berechnen. Die Wissenschaft hat hierfür einige Modelle hervorgebracht:

Komplexe Permeabilität	Lineares Modell: Der Realteil der Permeabilität beschreibt die „normale" Permeabilität, der Imaginärteil stellt die Kernverluste dar.
Jiles-Atherton [1]	Weit verbreitetes, physikalisch motiviertes Modell, das die Hysteresekurve mit fünf Parametern beschreibt. Trotz der physikalischen Grundlage eher phänomenologisch, lässt unphysikalische Hysteresekurven zu.
Preisach [2]	Mathematisches Modell: Eine Hysteresekurve wird in eine Summe von *Hysteronen* entwickelt – Stufenfunktionen mit Hysterese.
Coleman-Hodgdon	Nichtlineare, differentielle Formulierung, enthält zwei Funktionen, die den Charakter der Hysterese bestimmen.
Maxwell-Slip	Modell für Reibung, jedoch auch auf andere Hysereseprozesse übertragbar
VINCH [3]	Thermodynamisch begründetes Modell mit einer beliebigen Anzahl von internen Modellvariablen. Reproduziert *minor loops* relativ gut.

Die verschiedenen Modelle unterscheiden sich insbesondere in der Anzahl der Modellparameter. Während das Jiles-Atherton-Modell mit fünf Parametern auskommt, können etwa im Preisach- oder VINCH-Modell beliebig viele Parameter eingeführt werden – dies führt dazu, dass gemessene Hysteresekurven genauer angenähert werden können, jedoch der Modellcharakter verloren geht.
Ein entscheidendes Problem bei fast allen erwähnten Modellen ist die Parameteridentifikation – die Modellparameter müssen an Messkurven angeglichen werden. Bei dem Modell der komplexen Permeabilität fällt dies noch recht leicht, da hier eine einfache Leistungsmessung ausreicht (noch dazu ist das Modell linear). Für die anderen Modelle, die alle nichtlinear sind, gestaltet sich dies jedoch deutlich schwieriger.

1.3 Parameteridentifikation

Das übliche Vorgehen ist, eine Kostenfunktion aufzustellen, die ihr Minimum dann annimmt, wenn Messdaten und Simulationsdaten am besten übereinstimmen. Diese Kostenfunktion muss anschließend mit einem geeigneten Algorithmus minimiert werden. Für nichtlineare Modelle wie die vorliegenden gestaltet sich dies jedoch oft sehr schwierig – die meisten Algorithmen benötigen für eine schnelle Konvergenz die Jacobi- oder sogar auch die Hesse-Matrix der zu minimierenden Kostenfunktion. Diese sind analytisch jedoch nicht, bzw. nur mit exorbitantem Aufwand zugänglich. Ein alternativer Ansatz ist, gradientenfreie Algorithmen zu verwenden (genetische Algorithmen, linear Programming etc.). Dies wurde in der Literatur auch wiederholt vorgenommen – die Konvergenz kann hierbei jedoch nicht garantiert werden, noch dazu sind diese Algorithmen recht langsam.

1.4 Automatisches Differenzieren

Ein Ausweg aus diesen Problemen versprechen die neuen Möglichkeiten aus dem Gebiet des automatischen Differenzierens. Durch die stetige Erhöhung der Rechenleistung von Computern und die neuen Anwendungsmöglichkeiten im Bereich der künstlichen Intelligenz konnten hier in den letzten Jahren viele Verbesserungen erreicht werden.
Mit dem automatischen Differenzieren ist es möglich, Computerprogramme abzuleiten, also Jacobi-, Hessematrix etc. für Programme zu bestimmen. Die Idee hinter diesem Projekt ist es nun, die Fortschritte in diesem Bereich zu nutzen und

auf das dargestellte Problem der Parameteridentifikation von Hysteresemodellen anzuwenden.

1.5 Vorgehen

In Abbildung 1 wird das Vorgehen schematisch dargestellt, zumindest der erste Teil es Projektes, die Lösung der Aufgabe im nicht-ortsaufgelösten Fall. Der Projektplan für den zweiten Teil ist im Ausblick, Abbildung 12 bildlich dargestellt. Auf die verschiedenen Bestandteile soll nun im Folgenden eingegangen werden.

Jiles-Atherton-Modell → Differentialquotienten Als Beispielmodell wird das Jiles-Atherton-Modell verwendet, da es relativ populär ist und somit viele Erfahrungswerte in der wissenschaftlichen Literatur vorhanden sind. Außerdem hat es eine recht kleine Parameteranzahl, was einige Vereinfachungen mit sich bringt. Das Modell wird in Abschnitt 2 genauer dargestellt.
Die Methodik, mit der die Parameter im Rahmen dieses Projektes ermittelt werden, lässt sich jedoch problemlos auf andere Hysteresemodelle übertragen. Man muss dafür natürlich die Algorithmen neu implementieren, das Grundkonzept bleibt jedoch das gleiche.

Skalares Vorwärtsmodell Der nächste Schritt ist, das Vorwärtsmodell für den skalaren Fall (keine Ortsauflösung von Feld und Hysterese) zu entwerfen. Dieses Vorwärtsmodell ist effektiv eine Abbildung vom Parameterraum in die Menge der Hysteresekurven – für jede Parameterkombination erhält man eine (zeitlich diskretisierte) B-H-Kurve.

Integration Dies wird durch die Integration der differentiellen Magnetisierung, die das Jiles-Atherton-Modell beschreibt, erreicht. Da die verfügbaren AD-Werkzeuge nicht mit den integrierten Algorithmen der üblichen Mathematik-Pakete umgehen können, muss diese Integration manuell implementiert werden. Als Algorithmus wird ein Runge-Kutta-Verfahren gewählt, welches in einem Fachartikel [4] als geeignet für das Jiles-Atherton-Modell beschrieben wurde. Eine kurze Übersicht zum Thema der numerischen Integration und den Runge-Kutta-Verfahren ist in Abschnitt 3 gegeben.

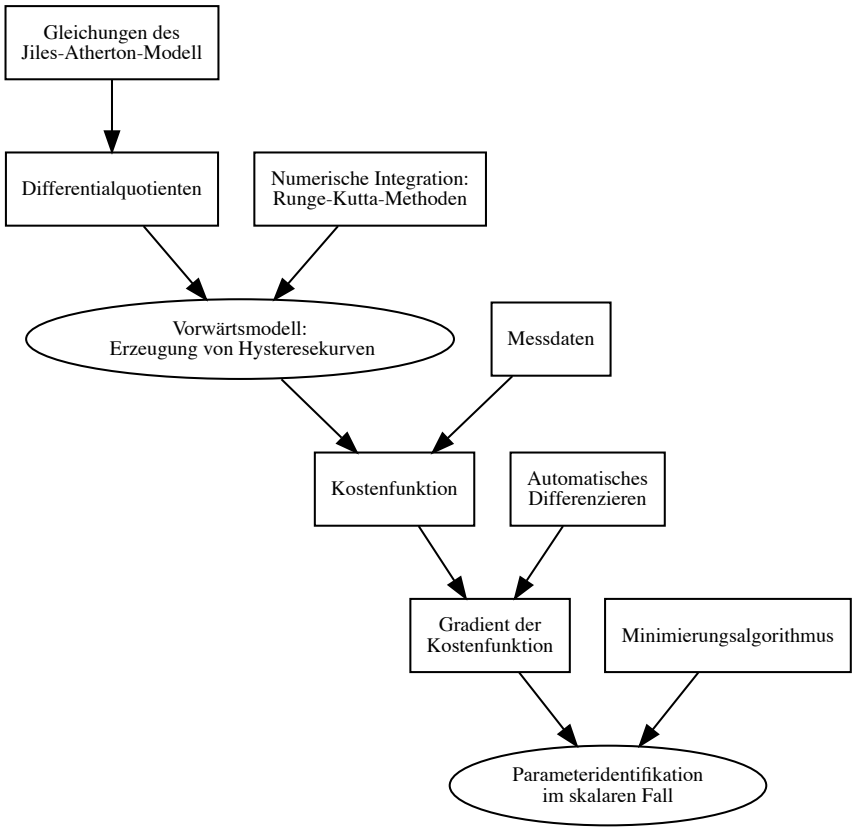

Abbildung 1: Vorgehen zur Parameteridentifikation im Fall ohne Ortsauflösung.
Ausgangspunkt sind die konstituierenden Gleichungen des Jiles-Atherton-Modells, aus denen man die Differentialquotienten gewinnen kann, die man für die numerische Integration benötigt. Mit dieser können Hysteresekurven zu gegebenen Parametern erzeugt werden. Die Differenz von erzeugten Hysteresekurven und den Messdaten führen zur Kostenfunktion, die mit der Methode des automatischen Differenzierens nach den Parametern abgeleitet werden kann.
Mit einem Minimierungsalgorithmus kann man nun das Optimierungsproblem lösen und so die Parameter identifizieren.

Messdaten, Kostenfunktion Aus der quadratischen Abweichung zwischen modellierten und gemessen Hysteresekurven kann man nun die Kostenfunktion für das Minimierungsproblem aufstellen. Im Rahmen dieses Projektes wird nicht auf tatsächliche Messdaten zurückgegriffen, sondern auf simulierte. Diese kann man leicht erzeugen und z.B. mit Rauschen etc. versehen, um den Algorithmus zu testen.

Inverses Problem: Automatisches Differenzieren, Parameteridentifikation Der nächste Schritt ist der Entwurf des inversen Problems: Gesucht ist nun das Urbild einer Hysteresekurve (Messung) im Parameterraum. Dazu wird zuerst eine Kostenfunktion aufgestellt, von der anschließend mit den Methoden des automatischen Differenzierens Jacobi- und Hessematrix ermittelt werden. Daraufhin wird die Kostenfunktion mit einem geeigneten Algorithmus minimiert. Dadurch erhält man die optimierten Parameter zur gegebenen Messkurve.
Mit diesem ersten Teil kann ein *proof-of-concept* erreicht werden. Man kann überprüfen, ob das automatische Differenzieren und somit das Gewinnen von Ableitungen der Kostenfunktion für den Optimierungsalgorithmus gelingt und ob die Zahl der Funktionsaufrufe dadurch tatsächlich im Vergleich zu einem gradientenfreien Verfahren fällt.
Im nächsten Schritt sollen dann die Methoden und Prinzipen auf das ortsaufgelöste Modell übertragen werden:

Ortsaufgelöstes Vorwärtsmodell Die bisherigen Berechnungen galten stets für ein skalares Modell, der Kern wurde also betrachtet als würde in jedem Punkt das gleiche Feld herrschen. Dies ist jedoch natürlich nur eine Näherung. Daher soll im nächsten Schritt das Jiles-Atherton-Modell auf einen ausgedehnten Kern übertragen werden. Die Zeitdiskretisierung wird genauso wie vorher mit finiten Differenzen (Runge-Kutta-Methode) vorgenommen, die Ortsdiskretisierung basiert auf der Finite-Elemente-Methode. Mit dem ortsaufgelösten Vorwärtsmodell kann nun die Hysterese im gesamten Kern ortsaufgelöst und unter Berücksichtigung der Maxwell-Gleichungen modelliert werden.

Inverses ortsaufgelöstes Problem Wie beim skalaren Modell gilt es nun, das inverse Problem zu implementieren. Hierfür wird abermals auf das Automatische Differenzieren zurückgegriffen – in diesem Fall ist bei einem einzelnen Zeitschritt jedoch nicht eine einzelne Gleichung zu lösen, sondern ein ganzes FEM-Problem. Mit der Software *dolfin-adjoint* kann jedoch die Differenziation

von FEM-Modellen abstrahiert und so stark vereinfach werden. Die Kostenfunktion wird nun zu einem Kostenfunktional, also einer Funktion der Lösung der einzelnen FEM-Probleme aus den einzelnen Zeitschritten. Mit einem geeigneten Optimierungssalgorithmus kann nun abermals aus Messwerten auf die Parameter geschlossen werden.

2 Jiles-Atherton-Modell

Das Jiles-Atherton-Modell ist ein physikalisch motiviertes Modell zur Beschreibung der Hysterese von magnetischen Materialien.

2.1 Konstituierende Gleichungen

$$H_\mathrm{e} = H + \alpha M \qquad \text{Effektives Feld} \tag{1}$$

$$M_\mathrm{an} = M_\mathrm{sat}\,\mathrm{L}\left(\frac{H_\mathrm{e}}{a}\right) \qquad \text{Anhysteretische Magnetisierung} \tag{2}$$

$$\frac{\mathrm{d}M_\mathrm{irr}}{\mathrm{d}H_\mathrm{e}} = \frac{M_\mathrm{an} - M_\mathrm{irr}}{k\,\mathrm{sign}\left(\frac{\mathrm{d}H}{\mathrm{d}t}\right)} \qquad \text{Pinning} \tag{3}$$

$$M = M_\mathrm{rev} + M_\mathrm{irr} \qquad \text{Gesamte Magnetisierung} \tag{4}$$

$$M_\mathrm{rev} = c(M_\mathrm{an} - M_\mathrm{irr}) \qquad \text{Irreversible Magnetisierung} \tag{5}$$

In diesen fünf Modellgleichungen kommen fünf Modellparameter vor:

- α Interdomänenkopplung
- a Domänenwanddichte
- M_sat Sättigungsmagnetisierung
- k Pinning-Energie
- c Magnetisierungsreversibilität

Die Funktion $\mathrm{L}(x)$ ist die sogenannte Langevin-Funktion, die die anhysteretische Magnetisierungskurve beschreibt. Sie ist definiert durch

$$\mathrm{L}(x) = \coth(x) - \frac{1}{x} \tag{6}$$

2.2 Herleitung der Differentiale

Ziel ist es nun, einen differentiellen Zusammenhang zwischen B und H zu finden. Dazu:

$$M \underset{(4)}{=} M_{\mathrm{rev}} + M_{\mathrm{irr}} \underset{(5)}{=} c(M_{\mathrm{an}} - M_{\mathrm{irr}}) + M_{\mathrm{irr}} = cM_{\mathrm{an}} + (1-c)M_{\mathrm{irr}}. \tag{7}$$

Es gilt also

$$\mathrm{d}M = (1-c)\,\mathrm{d}M_{\mathrm{irr}} + c\,\mathrm{d}M_{\mathrm{an}} \tag{8}$$

und

$$M_{\mathrm{irr}} = \frac{M - cM_{\mathrm{an}}}{1-c}. \tag{9}$$

Weiterhin stimmt jeweils

$$\mathrm{d}M_{\mathrm{irr}} = \frac{\mathrm{d}M_{\mathrm{irr}}}{\mathrm{d}H_{\mathrm{e}}}\,\mathrm{d}H_{\mathrm{e}} \tag{10}$$

$$\mathrm{d}M_{\mathrm{an}} = \frac{\mathrm{d}M_{\mathrm{an}}}{\mathrm{d}H_{\mathrm{e}}}\,\mathrm{d}H_{\mathrm{e}} \tag{11}$$

$$\mathrm{d}H_{\mathrm{e}} \underset{(1)}{=} \mathrm{d}H + \alpha\,\mathrm{d}M. \tag{12}$$

Man erhält nun für $\mathrm{d}M$ zusammengefasst:

$$\begin{aligned} \mathrm{d}M &\underset{(8)}{=} (1-c)\,\mathrm{d}M_{\mathrm{irr}} + c\,\mathrm{d}M_{\mathrm{an}} \\ &= (1-c)\frac{\mathrm{d}M_{\mathrm{irr}}}{\mathrm{d}H_{\mathrm{e}}}\,\mathrm{d}H_{\mathrm{e}} + c\frac{\mathrm{d}M_{\mathrm{an}}}{\mathrm{d}H_{\mathrm{e}}}\,\mathrm{d}H_{\mathrm{e}}, \end{aligned} \tag{13}$$

in dieser Darstellung sind nun die aus den konstituierenden Gleichungen 2 und 3 leicht zugänglichen Größen $\mathrm{d}M_{\mathrm{irr}}/\mathrm{d}H_{\mathrm{e}}$ und $\mathrm{d}M_{\mathrm{an}}/\mathrm{d}H_{\mathrm{e}}$ enthalten.

$$\Rightarrow \frac{\mathrm{d}M}{\mathrm{d}H_{\mathrm{e}}} = (1-c)\frac{\mathrm{d}M_{\mathrm{irr}}}{\mathrm{d}H_{\mathrm{e}}} + c\frac{\mathrm{d}M_{\mathrm{an}}}{\mathrm{d}H_{\mathrm{e}}} \tag{14}$$

Dabei gilt mit Gleichung 2:

$$\frac{\mathrm{d}M_{\mathrm{an}}}{\mathrm{d}H_{\mathrm{e}}} = \frac{M_{\mathrm{sat}}}{a}\frac{\mathrm{d}\,\mathrm{L}(H_{\mathrm{e}}/a)}{\mathrm{d}(H_{\mathrm{e}}/a)} \tag{15}$$

Mit Gleichung 12 erhält man:

$$\begin{aligned} \mathrm{d}M &= \frac{\mathrm{d}M}{\mathrm{d}H_{\mathrm{e}}}\,\mathrm{d}H_{\mathrm{e}} \\ &= \frac{\mathrm{d}M}{\mathrm{d}H_{\mathrm{e}}}\,(\mathrm{d}H + \alpha\,\mathrm{d}M) \end{aligned} \tag{16}$$

$$\Rightarrow \frac{\mathrm{d}M}{\mathrm{d}H} = \frac{\mathrm{d}M/\mathrm{d}H_{\mathrm{e}}}{1 - \alpha\,\mathrm{d}M/\mathrm{d}H_{\mathrm{e}}}$$

Aus $B = \mu_0(H + M)$ folgt nun

$$\mathrm{d}B = \mu_0(\mathrm{d}H + \mathrm{d}M) \tag{17}$$

und somit

$$\begin{aligned} \frac{\mathrm{d}B}{\mathrm{d}H} &= \mu_0 \left(1 + \frac{\mathrm{d}M}{\mathrm{d}H}\right) \\ &= \mu_0 \frac{1 + (1-\alpha)\,\mathrm{d}M/\mathrm{d}H_\mathrm{e}}{1 - \alpha\,\mathrm{d}M/\mathrm{d}H_\mathrm{e}}. \end{aligned} \tag{18}$$

Entsprechend gilt für die vom B-Feld abhängige Formulierung

$$\frac{\mathrm{d}H}{\mathrm{d}B} = \frac{1}{\mu_0} \frac{1 - \alpha\,\mathrm{d}M/\mathrm{d}H_\mathrm{e}}{1 + (1-\alpha)\,\mathrm{d}M/\mathrm{d}H_\mathrm{e}} \tag{19}$$

Abhängigkeiten Die Differentiale sind, wenn man alle Terme ausschreibt, abhängig von B, H, $\mathrm{d}H/\mathrm{d}t$ und dem Parametervektor $\vec{p} = (\alpha, a, M_\mathrm{sat}, k, c)^T$:

$$\frac{\mathrm{d}B}{\mathrm{d}H} = f\left(B, H, \frac{\mathrm{d}H}{\mathrm{d}t}, \vec{p}\right) \tag{20}$$

3 Numerische Integration

Das Jiles-Atherton-Modell beschreibt – wie oben erkennbar – nur einen differentiellen Zusammenhang. Um Aussagen über den zeitlichen Verlauf der Magnetisierung und somit über die Kernverluste zu gewinnen, muss die jeweilige gewöhnliche Differentialgleichung integriert werden, zum Beispiel

$$B(H) = \int \frac{\mathrm{d}B}{\mathrm{d}H}\,\mathrm{d}H + \text{const.} \tag{21}$$

Diese Integration ist analytisch nicht zugänglich und muss numerisch ausgeführt werden. Man verwendet hierfür typischerweise integrierte Lösungen wie den Solver *ode45* in Matlab oder das Python-Äquivalent *scipy.integrate.ode*. Diese Methoden implementieren die üblichen Verfahren, explizit das Runge-Kutta-Fehlberg-Verfahren. Dieses ist eine Variante des klassischen Runge-Kutta-Verfahrens mit adaptiver Schrittweite.
Da im Nachfolgenden jedoch das Ergebnis einer solchen Integration nach dem Parametervektor $\vec{p}$ differenziert werden soll, benötigt man einen Integrationsalgorithmus, der den Werkzeugen der Automatischen Differenziation zugänglich ist. Aus diesem Grund muss die Integration manuell implementiert werden.

3.1 Explizites Euler-Verfahren

Das einfachste Integrationsverfahren ist das explizite Euler-Verfahren (Forward-Euler, Polygonzugverfahren), es entspricht dem einfachsten expliziten Finite-Differenzen-Ansatz:

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= f(y,t) \\ \Rightarrow \quad \frac{y_n - y_{n-1}}{t_n - t_{n-1}} &\approx f(y_{n-1}, t_{n-1}) \end{aligned} \tag{22}$$

So erhält man für den jeweils nächsten Zeitschritt

$$y_n \approx y_{n-1} + f(y_{n-1}, t_{n-1})(t_n - t_{n-1}) \tag{23}$$

3.2 Implizites Euler-Verfahren

Im vorherigen Abschnitt wurde ein explizites Verfahren gewählt, d.h. die Funktion $f(y,t)$ wurde durch bereit bekannte Werte von y und t angenähert, $f(y,t) \approx f(y_{n-1}, t_{n-1})$. Alternativ kann man auch einen impliziten Ansatz wählen, $f(y,t) \approx f(y_n, t_n)$, dann erhält man eine implizite Gleichung für den nächsten Zeitschritt

$$y_n = y_{n-1} + f(y_n, t_n)(t_n - t_{n-1}). \tag{24}$$

Diese Gleichung ist jedoch – da sie implizit und im Allgemeinen nichtlinear ist – ungleich schwerer zu lösen als die explizite, man benötigt hier üblicherweise numerische Methoden wie die Fixpunktiteration oder etwa, falls der Gradient vorhanden ist, das Newton-Verfahren.
Der erhöhte Rechenaufwand der impliziten Verfahren wird durch eine erhöhte Stabilität ausgeglichen, insbesondere bei der Problemklasse der steifen Probleme. Die Integration des Jiles-Atherton-Modells stellt jedoch keine hohen Anforderungen an die Stabilität, wodurch explizite Verfahren verwendet werden können.

3.3 Kurzer Vergleich

Der Nachteil von diesen einfachen Verfahren ist ihre geringe Genauigkeit, bzw. die sehr kleine Schrittweite und somit der erhöhte Rechenaufwand, welche man

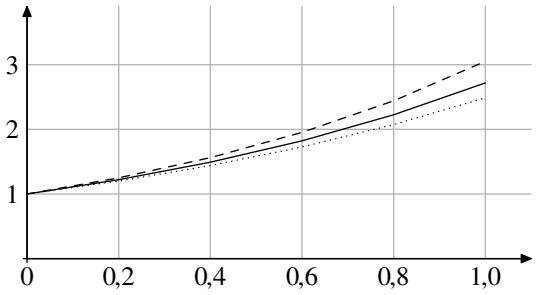

Abbildung 2: Vergleich von einfachen numerischen Integrationsverfahren (vgl. ODE 25). Die durchgezogene Linie ist die exakte Lösung, gepunktet sieht man das Resultat des expliziten Euler-Verfahrens, das implizite Verfahren ist gestrichelt dargestellt. Die vertikalen Gitterlinien beschreiben die diskretisierten Zeiten.

wählen muss, um eine ausreichende Genauigkeit zu erhalten. Als Beispiel soll nun die wohl einfachste Differentialgleichung

$$\frac{\mathrm{d}y}{\mathrm{d}t} = y(t), \quad y(0) = 1 \tag{25}$$

mit der Lösung $y(t) = \mathrm{e}^t$ betrachtet werden.
Wie man in Abbildung 2 erkennt, akkumulieren sich in beiden Verfahren (bei großer Schrittweite) schnell große Fehler. Das explizite Verfahren nutzt die Steigung im aktuellen Punkt für den nächsten Schritt, unterschätzt bei der e-Funktion also stets die Steigung, während das implizite Verfahren jeweils die Steigung im nächsten Punkt als Näherung verwendet. Die führt in diesem Fall jeweils zu einer Überschätzung der Steigung.

Um die Schrittweise nicht zu klein wählen zu müssen und somit den Rechenaufwand zu reduzieren, wurden viele verschiedene Integrationsverfahren entworfen. Die meisten sind Einschritt-Verfahren, wie etwa die Runge-Kutta-Methoden – für den nächsten Zeitschritt werden dabei jedoch Zwischenschritte berechnet, die die Näherung verbessern. Man erhöht so zwar den Aufwand für einen einzelnen Schritt, kann aber dafür die Schrittweite deutlich größer wählen.
Eine weitere Möglichkeit, den Rechenaufwand zu reduzieren ist eine adaptive Schrittweitensteuerung: Man schätzt mit einem weiteren Zwischenschritt den Fehler und kann so die Schrittweite jeweils so groß wie möglich wählen (vgl. Runge-Kutta-Fehlberg-Methode).
Die Wahl des richtigen Algorithmus ist stets vom genauen Problem abhängig, für die vorliegende Integration des Jiles-Atherton-Modells eignet sich laut [4] jedoch der viel verwendete RK4-Algorithmus mit fester Schrittweite.

3.4 Runge-Kutta-Methoden

Die eben vorgestellten Euler-Verfahren sind Sonderfälle der Runge-Kutta-Methoden, welche eine wichtige Klasse von Einschrittverfahren darstellen. Der

m-stufige Algorithmus benötigt Gewichte $\alpha_i, \gamma_i, 1 \leq i \leq m$ und $\beta_{i,l}, 1 \leq i, l \leq m$. Weiterhin müssen die Schrittweiten h_i vorgegeben werden, zur Vereinfachung soll jedoch eine feste Schrittweite h angenommen werden.
Für den nächsten Datenpunkt berechnet man:

$$\begin{aligned} &\text{Nächster Zeitpunkt} && t_n = t_{n-1} + h \\ &\text{i-ter Hilfswert} && k_i = f\left(y_{n-1} + h \sum_{l=1}^{m} \beta_{i,l} k_l, t_{n-1} + \alpha_i h\right) \\ &\text{Nächster Wert} && y_n = y_{n-1} + h \sum_{l=1}^{m} y_l k_l \end{aligned} \tag{26}$$

Die Gewichte werden meist in einem sogenannten Butcher-Tableau dargestellt

$$\begin{array}{c|ccc} \alpha_1 & \beta_{1,1} & \cdots & \beta_{1,m} \\ \alpha_2 & \beta_{2,1} & \cdots & \beta_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_m & \beta_{m,1} & \cdots & \beta_{m,m} \\ \hline & \gamma_1 & \cdots & \gamma_m \end{array} .$$

Die entstehenden Gleichungen für die Hilfswerte k_i sind im Allgemeinen implizit, sie müssten also iterativ gelöst werden. Sind jedoch die Werte k_i nur von schon vorher berechneten Werten $k_j, j < i$ abhängig, entsteht ein explizites Verfahren. Die Koeffizienten $\beta_{i,l}$ lassen sich im Butcher-Tableau nun als untere Dreiecksmatrix anordnen – das klassische Runge-Kutta-Verfahren (RK4) hat beispielsweise das Tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{27}$$

Schreibt man für dieses die einzelnen Schritte aus, so ist die Berechnung des nächsten Wertes im RK4-Verfahren:

$$\begin{aligned} t_n &= t_{n-1} + h \\ \\ k_1 &= f\left(y_{n-1}, t_{n-1}\right) \\ k_2 &= f\left(y_{n-1} + hk_1/2, t_{n-1} + h/2\right) \\ k_3 &= f\left(y_{n-1} + hk_2/2, t_{n-1} + h/2\right) \\ k_4 &= f\left(y_{n-1} + hk_3, t_{n-1} + h\right) \\ \\ y_n &= y_{n-1} + \tfrac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \end{aligned} \tag{28}$$

3.5 Integration des Jiles-Atherton-Modells

Genau dieses klassische Runge-Kutta-Verfahren soll nun auf das Jiles-Atherton-Modell angewendet werden. Das Differential dB/dH aus Gleichung 18 wird nun als die zu integrierende Funktion f betrachtet. Es gilt also

$$\mathrm{d}B = f\left(B, H, \tfrac{dH}{dt}, \vec{p}\right) \mathrm{d}H. \tag{29}$$

Als Eigabe benötigt der Algorithmus je ein Array für das magnetische Feld H und für die Zeitpunkte t, weiterhin eines für die Parameter. Da das RK4-Verfahren für jeden Schritt auch Funktionsauswertungen bei $t_n + h/2$ benötigt, müssen auch für H Werte bei $t_n + h/2$ vorliegen. Dies wird erreicht, indem die Schritte der Integration jeweils zwei Zeitschritte der Eingabedaten umfassen:

> H_{2n} und B_n beschreiben stets den gleichen Zeitpunkt t_{2n}. Für B existieren somit nur zu jedem zweiten Zeitpunkt Werte. Das Ergebnis B hat entsprechend nur die halbe Samplingrate der Eingangsdaten.

Dabei ist wichtig, dass der Zeitpunkt t_{2n+1} stets genau in der Mitte zwischen t_{2n} und t_{2n+2} für alle $n \in \mathbb{N}$ liegt. Da Messdaten typischerweise ohnehin äquidistant vorliegen ist das keine starke Einschränkung.

Entsprechend der Definition in 28 gilt nun

$$
\begin{aligned}
h &= H_{2n} - H_{2n-2} \\
k_1 &= f\left(B_{n-1}, \quad H_{2n-2}, \quad (\tfrac{dH}{dt})_{2n-2}, \quad \vec{p}\right) \\
k_2 &= f\left(B_{n-1} + h\tfrac{k_1}{2}, \quad H_{2n-1}, \quad (\tfrac{dH}{dt})_{2n-1}, \quad \vec{p}\right) \\
k_3 &= f\left(B_{n-1} + h\tfrac{k_2}{2}, \quad H_{2n-1}, \quad (\tfrac{dH}{dt})_{2n-1}, \quad \vec{p}\right) \\
k_4 &= f\left(B_{n-1} + hk_3, \quad H_{2n}, \quad (\tfrac{dH}{dt})_{2n}, \quad \vec{p}\right) \\
B_n &= B_{n-1} + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
\tag{30}
$$

Dies beschreibt jeweils einen Schritt des Integrationsverfahrens. Führt man diesen für alle Zeitpunkte durch, erhält man die Magnetisierung M für die gewählte Feldvorgabe H und die Modellparameter $\vec{p}$.
Der beschriebene Algorithmus wurde, um eine hohe numerische Effizienz zu erzielen, in der Programmiersprache *C++* implementiert.

3.6 Numerische Details

Um die Stabilität des Algorithmus zu optimieren, gibt es einige Details, die beachtet werden müssen.

Die Langevin-Funktion Die Definition der Langevin-Funktion lautet

$$\mathrm{L}(x) = \coth(x) - \frac{1}{x}, \tag{31}$$

die beiden Terme haben jeweils in der Umgebung der 0 eine Polstelle, sind dabei jedoch etwa gleich groß. Durch die sehr großen Werte, die sich jedoch fast vollständig kompensieren tritt bei numerischen Berechnungen unweigerlich ein Signifikanzverlust auf. Entgegenwirken kann man diesem durch eine Approximation in der Nähe der 0. Es gilt laut [5]

$$\coth(x) \approx \frac{1}{x} + \frac{x}{3} - \frac{x^3}{45} + \cdots, \tag{32}$$

für die Langevinfunktion folgt also in der Nähe der 0

$$\mathrm{L}(x) \approx \frac{x}{3}. \tag{33}$$

Entsprechend folgt für die Ableitung:

$$\frac{\mathrm{d}\,\mathrm{L}(x)}{\mathrm{d}x} = \frac{1}{x^2} - \frac{1}{\sinh(x)^2} \approx \frac{1}{3} - \frac{1}{15}x^2 + \cdots \tag{34}$$

4 Anwendung des Jiles-Atherton-Modells

Im folgenden Abschnitt sollen nun einige simulierte Kurven präsentiert und anhand dieser die Eigenschaften des Modells erläutert werden.

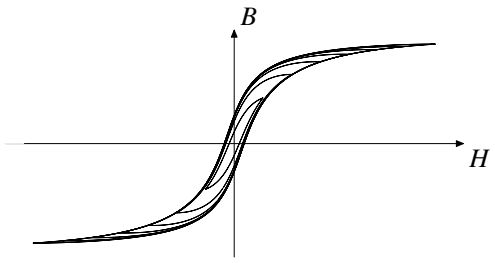

Abbildung 3: Schar von Hysteresekurven für verschiedene Aussteuerungen. Der nichthysterische Teil wird von der Langevinfunktion festgelegt.

Hysteresekurven Als erstes Beispiel dient eine Kurvenschar, die Hysteresekurven für verschiedene Aussteuerungen zeigt (Abbildung 3). Man erkennt, dass das Modell die Charakteristik typischer ferromagnetischer Hysteresekurven nachbildet:
Bei hohen Feldstärken wächst die Induktion stark an, dies ist die typische Sättigungserscheinung bei Kernmaterialien. Nachdem der Umkehrpunkt erreicht wurde, läuft die Kurve auf einem anderen Weg zurück (Hyterese), sodass die Flussdichte bei verschwindender Feldstärke nicht mehr den Wert 0 annimmt, sondern eine Restmagnetisierung (Remanenz) übrig bleibt. Im weiteren Verlauf kreuzt die Flussdichte nun die H-Achse, den entsprechenden Achsenabschnitt nennt man Koerzitivfeldstärke.

In Abbildung 4 ist eine einzelne Hysteresekurve gezeigt. In diesem Fall ist auch die Neukurve dargestellt, die vom Modell ebenfalls repliziert wird. Im Gegensatz zu den meisten Messkurven beginnt die Neukurve im Jiles-Atherton-Modell jedoch näherungsweise linear.

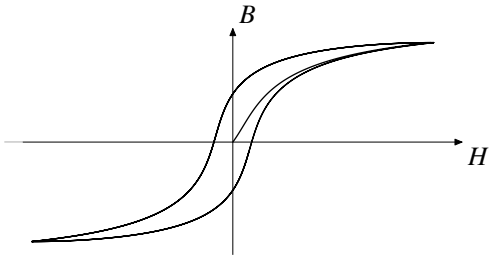

Abbildung 4: Einzelne Hysteresekurve mit Neukurve.

4.1 Sinusförmiger Strom

Wie oben gezeigt, kann man entweder $\mathrm{d}B/\mathrm{d}H$ oder $\mathrm{d}H/\mathrm{d}B$ integrieren. Gibt man einen Strom vor, so ist $\mathrm{d}H$ festgelegt, man verwendet also $\mathrm{d}B/\mathrm{d}H$. Wird der Kern mit einer Spannung angeregt, nutzt man $\mathrm{d}H/\mathrm{d}B$. Im Allgemeinen wird man den Kern ohnehin eingebettet in einem Netzwerk betrachten und auch die Integration entsprechend anpassen.

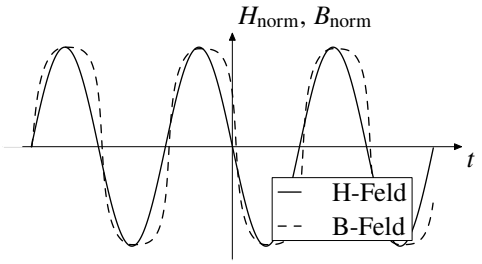

Abbildung 5: Verlauf von H- und B-Feld bei vorgegebenem sinusförmigen Strom. Durch die nichtlinearen Kerneigenschaften wird die Form des Sinus stark verbreitert, entsprechend treten hohe Steigungen bei den Nulldurchgängen der magnetischen Flussdichte auf.

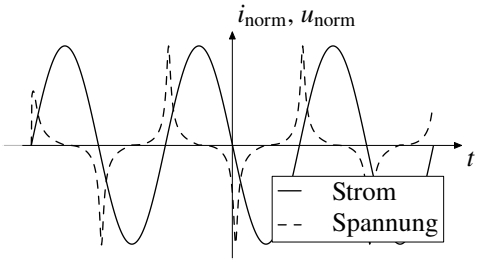

Abbildung 6: Verlauf von Strom und Spannung bei einem sinusförmigen Strom. Die steilen Flanken der Hysteresekurve führen zu großen Spannungsspitzen durch $\nabla \times E = -\partial B/\partial t$.

In Abbildung ist der Verlauf von B und H-Feld bei Anregung mit einem sinusförmigen Strom gezeigt – die Sättigungscharakteristik rundet die Spitzen des Sinus deutlich ab, je mehr die Aussteuerung in die Sättigung geht, desto mehr nähert sich die zeitliche Form des B-Feldes einer Rechteckschwingung an.
Da das Induktionsgesetz die induzierte Spannung mit der zeitlichen Ableitung der magnetischen Flussdichte verknüpft, wird deutlich, dass hier Probleme mit Überpannungen auftreten können – die steilen Flanken im Verlauf des B-Feldes

führen zu Spannungsspitzen, die die Isolation des induktiven Bauelements beschädigen können.

4.2 Sinusförmige Spannung

Als zweites Beispiel wird nun eine sinusförmige Spannung vorgegeben.

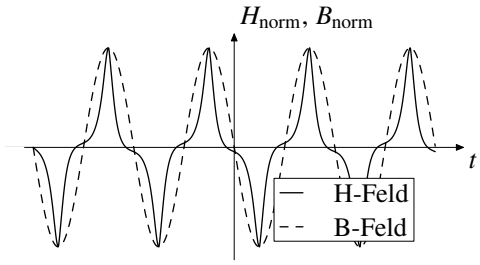

Abbildung 7: Verlauf von H- und B-Feld bei sinusförmiger Spannung. Der typische Verlauf des Magnetisierungsstroms wird deutlich – kommt die Induktion in die Nähe der Sättigung, steigt das H-Feld und somit der Strom stark an.

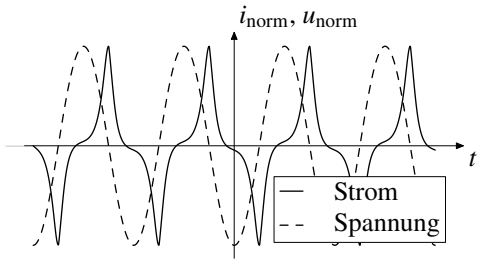

Abbildung 8: Verlauf von Strom und Spannung bei sinusförmiger Spannung.

Wie man in den Abbildungen oder auch den Hysteresekurven sieht, wächst der Strom für hohe Induktionen extrem stark an, die Sättigung tritt ein und verursacht enorm hohe Verluste in den Leitern. Daher muss dieser Zustand bei der Auslegung von induktiven Bauelementen im Allgemeinen unbedingt vermieden werden.

4.3 Allgemeiner Fall

Im allgemeinen Fall kann man das induktive Bauelement natürlich nicht isoliert betrachten. Vielmehr muss man das Modell des Kerns in ein allgemeines Schaltungsmodell einbinden und dieses transient lösen. So kann man dann auch diverse Themen wie das Einschaltverhalten, das Schaltungsverhalten an der Grenze zur Sättigung oder die Kernverluste im Betrieb untersuchen.

4.4 Berechnung der Kernverluste

Ein transientes Hysteresemodell ist dann sinnvoll einsetzbar, wenn es in der Praxis von Ingenieuren Vorteile bringt. Entscheidend ist daher einerseits, dass man damit Sättigungsprobleme erkennen kann, andererseits aber, dass man Kernverluste vorhersagen kann: Kennt man die Geometrie der Anordnung sowie Spannungs- und Stromform, möchte man ermitteln können, wo im Kern welche Verlustleistung auftritt.
Die Verlustleistung lässt sich – wie immer im Zusammenhang mit Maxwell-Gleichungen – aus dem Poynting-Vektor bestimmen. Es gilt

$$-\nabla \cdot \vec{S} = \vec{E} \cdot \vec{J} + \vec{E} \cdot \frac{\partial \vec{D}}{\partial t} + \vec{H} \cdot \frac{\partial \vec{B}}{\partial t}. \tag{35}$$

Der erste Term auf der rechten Seite beschreibt die ohmschen Verluste, die in den Leitern auftreten. Der zweite Teil ist die elektrische Energiedichte, die auch dielektrische Verluste enthält. Der letzte Term ist der relevante für die magnetischen Größen, er beschreibt die magnetische Energiedichte.
Nimmt man im ersten Schritt an, $\vec{H}$ und $\vec{B}$ sind (reell) linear verknüpft, so wird sofort klar, dass bei einer periodischen Aussteuerung keine Verlustleistung auftritt: H und $\partial B/\partial B$ wären um 90° phasenverschoben und sind somit orthogonal zueinander – ein Zeitmittel ergibt den Wert 0.
Nimmt man nun jedoch hysteretisches Verhalten an, so kann man schnell sehen, dass nun Leistungsdissipation vorhanden ist. Es gilt

$$P_{\mathrm{v,mag}} = \frac{1}{T} \int_0^T H \cdot \frac{\partial B}{\partial t} \,\mathrm{d}t = \frac{1}{T} \int_{\mathrm{Schleife}} H \cdot \mathrm{d}B, \tag{36}$$

man kann also das Zeitintegral in ein Integral über die Hysteresekurve umwandeln, die dissipierte Energie kann man also als die Fläche in der Hystereseschleife auffassen – multipliziert mit der Frequenz f.

4.5 Grenzen des Modells

Wie bei den verschiedenen Versuchen bei der Implementation gesehen werden konnte, lässt das Jiles-Atherton-Modell viele unphysikalische Parameterkombinationen zu.
So lassen sich beispielsweise Kurven erzeugen, bei denen die Magnetisierung weiter ansteigt, obwohl das erregende Feld bereits abfällt. Dies lässt sich in der Realität nicht beobachten – $\partial B/\partial t$ und $\partial H/\partial t$ haben stets das gleiche Vorzeichen.

Dies wäre kein Problem, wenn leicht bestimmbar wäre, ob eine Parameterkomination zulässig ist oder nicht – eine solche Regel lässt sich in der Fachliteratur jedoch nicht finden.

Der andere entscheidende Fehler des Modells ist, dass es sogenannte *minor loops*, die z.B. durch Oberwellen in der Stromform entstehen, nicht richtig reproduziert. In Messungen beobachtet man, dass Subschleifen auf den Ästen der Hystereseschleife stets geschlossen sind. Im Jiles-Atherton-Modell sind diese Subschleifen jedoch nicht geschlossen, der Endpunkt liegt meist ein Stück ober- oder unterhalb vom Startpunkt der Subschleife.
Bei Modellen, die Stromformen mit mehreren Richtungswechseln in einer Periode enthalten, ist das Jiles-Atherton-Modell somit intrinsisch fehlerhaft.

5 Parameteridentifikation

Wie am Anfang beschrieben wurde, ist das Ziel dieses Projektes einerseits, das Jiles-Atherton zu untersuchen, andererseits aber vor allem, eine neue Möglichkeit zu finden, um die Modellparameter aus Messdaten zu extrahieren.

5.1 Methode der kleinsten Fehlerquadrate

Der typische Ansatz, um Modelle an Messdaten anzupassen ist die *Methode der kleinsten Fehlerquadrate*. Bei ihr nimmt man an, dass die beste Übereinstimmung von Modell und Messung gegeben ist, wenn die Summe der quadratischen Abweichungen minimal ist. Ist $f(\vec{x}, \vec{p})$ die Modellfunktion, die von Eingangswerten $\vec{x}$ und Parametern $\vec{p}$ abhängt, so muss man das Minimum der Kostenfunktion

$$\text{cost}(\vec{p}) = \sum_i \left(f(x_i, \vec{p}) - y_i\right)^2 \tag{37}$$

finden, um die beste Übereinstimmung des Modells mit den gemessenen Daten $\vec{y}$ zu erhalten.
Die zu lösende mathematische Aufgabe ist also die Minimierung einer im Allgemeinen nichtlinearen und nicht konvexen Funktion. Für die Lösung dieser Problemstellung wurden diverse Algorithmen entworfen, die sich grob in zwei Klassen unterteilen lassen können: Ableitungsfreie Methoden und Methoden, die Ableitungen – zumindest die Erste – der zu minimierenden Funktion benötigen. Im Allgemeinen sind Methoden, die Ableitungen nutzen, erfolgreicher, da sie für den nächsten Optimierungsschritt jeweils über Informationen verfügen, in

welcher Richtung der Umgebung des aktuellen Schätzwertes die Kostenfunktion am steilsten abfällt. So kann insbesondere die Zahl der Funktionsauswertungen reduziert werden, was insbesondere dann relevant ist, wenn die Funktion rechnerisch teuer ist. Daher kommen gradientenfreie Verfahren üblicherweise dann zum Einsatz, wenn der Rechenaufwand für die Berechnung der auszuwertenden Funktion sehr klein ist oder aber der Gradient nur sehr schwer ermittelbar ist.

5.2 Ansätze

In der Fachliteratur gibt es diverse Ansätze für die Minimierung dieser Kostenfunktion:

- Neuronale Netze [6],
- Simulated annealing [7],
- Genetische Algorithmen [8], [9],
- Differential Evolution [10],
- Partikelschwarmoptimierung [11],
- Softcomputing [12] und
- Random Search [13],

einen Überblick bietet [14].

All diese Ansätze sind gradientenfrei, da der Gradient schwer zu beschaffen ist. Weiterhin ist all diesen Ansätzen gemein, dass sie meistens erst dann zum Einsatz kommen, wenn die typischen, *einfachen* Ansätze nicht funktionieren. Entsprechend sind sie alle eher langsam und rechenaufwändig.
Hier kommen nun die Methoden des algorithmischen Differenzierens zum tragen: Schafft man es mit diesen, numerisch stabile Informationen über die Ableitungen der Kostenfunktion zu generieren, lässt sich der einfachste Ansatz zur Parameteridentifikation nutzen: die Minimierung der Kostenfunktion mit ganz normalen Gradientenverfahren.
Hat man Ableitungsdaten, kann man auf eine ganze Palette von Minimierungsalgorithmen zurückgreifen, die schnell (im Sinne von Zeit und von Funktionsaufrufen) konvergieren und somit auch die Anpassung von sehr vielen Messdaten in tragbarer Zeit erlauben.

5.3 Geometrie und Parameteridentifikation

Neben den bereits genannten Problemen der bisherigen Ansätze zur Parameteridentifikation kommt ein weiteres, entscheidendes hinzu: Die Geometrie der Probe wird schlicht vernachlässigt – während man dies bei toroidförmigen Kernen herausrechnen kann, führt es bei Kernen mit keiner derartig großen Symmetrie zu Fehlern, da das inhomogene Feld im Kern als homogen angenommen wird.
In Wahrheit wird die Hysteresekurve jedoch an verschiedenen Punkten im Material mit verschiedenen Phasenlagen und Amplituden durchlaufen, die messbaren Größen – Spannung und Strom – lassen somit nur einen Schluss auf die integrale und somit geometrieabhängige Hysteresekurve zu. Dies gilt insbesondere auch für Messungen mit dem Epsteinrahmen, dem normierten Testaufbau für Elektrobleche.

Ziel dieser Arbeit ist es nun, die Geometrie in die Rechnung einzubeziehen. Dazu muss das Jiles-Atherton-Modell in eine orts- und zeitaufgelöste Simulation integriert werden (vgl. [15], [16]), anschließend müssen Strom und Spannung aus dem Modell gewonnen und mit den Messdaten in die Kostenfunktion überführt werden.
Offensichtlich ist nun die ursprünglich numerisch wenig aufwändige Funktionsauswertung des Vorwärtsmodells um Größenordnungen teurer geworden. Ein gradientenfreies Verfahren, welches eine kleinere Konvergenzrate hat, dafür aber keine Gradienten berechnen muss, kommt nun noch weniger infrage – die Zahl der Funktionsauswertungen ist um Größenordnungen größer als bei Gradientenverfahren.

Um das Ziel der Parameteridentifikation unter Beachtung der Geometrie zu ermöglichen, muss also eine Möglichkeit gefunden werden, die Ableitung(en) der Kostenfunktion zu bestimmen. Dafür gibt es drei Möglichkeiten:

- Symbolische Berechnung
- Numerische Berechnung (Differenzenquotienten)
- Automatisches Differenzieren

Um die Ableitungen symbolisch zu bestimmen, müssten die relativ komplizierten Gleichungen einzeln differenziert und die jeweiligen Funktionen ebenfalls in den Programmcode geschrieben werden. Dies ist gerade unter dem Aspekt, dass schon eine analytische Integration des Jiles-Atherton-Modells unmöglich ist, nicht möglich, da man auch die Integration symbolisch differenzieren müsste.
Die Effizienz ist bei der Berechnung mittels Differenzenquotienten kein Problem

– sie ist hier besonders hoch. Im Kontrast dazu ist jedoch die Genauigkeit sehr klein: Ist die Schrittweite sehr gering, liegen die zwei Funktionswerte eines Quotienten sehr nah beieinander – dies kann z.B. dazu führen, dass sich zwei Zahlen, die jeweils 32 Bit breit sind, nur in den letzten 5 Stellen unterscheiden. Ein enormer Genauigkeitsverlust ist die Folge.

Einen Mittelweg zwischen Geschwindigkeit und Genauigkeit bietet die dritte Methode, das automatische Differenzieren:

6 Automatisches Differenzieren

Die Grundidee des automatischen Differenzierens ist relativ simpel: Die Ableitung einer gegebenen Funktion wird mithilfe der Kettenregel so lange umgeformt, bis nur noch Basisfunktionen vorkommen, für die die Ableitung hinterlegt ist.
Man unterscheidet zwischen zwei grundlegenden Vorgehensweisen, *Forward-* und dem *Reverse-Mode*. Um den Unterschied zwischen diesen zu veranschaulichen betrachten wir die Funktion

$$f(x_1, x_2) = \frac{\sin(x_1/x_2)}{\exp(x_1/x_2)}. \tag{38}$$

Es ist klar einsehbar, dass man diese Funktion als Graph darstellen kann, der die Abfolge der einzelnen Rechenschritte veranschaulicht: Die Kanten stellen Eingangs- und Zwischenwerte dar, die Knoten die Rechenoperatoren. Der Graph ist in Abbildung 9 dargestellt. Für die Zwischenwerte gilt:

$$\begin{aligned} v_1 &= \frac{x_1}{x_2} & v_2 &= \sin(v_1) \\ v_3 &= \exp(v_1) & f &= v_4 = \frac{v_2}{v_3} \end{aligned}$$

Aufgabe ist es nun, aus diesem Graphen die Ableitung der Funktion (also dem Wert v_4) nach den Eingangsparametern x_1 und x_2 zu bestimmen. Für die Anwendung der beiden im Folgenden dargestellten Algorithmen müssen neben der Struktur der Funktion lediglich die Ableitungen der Operatoren (runde Knoten) bekannt sein. Es gilt:

$$\begin{aligned} \frac{\partial v_1}{\partial x_1} &= \frac{1}{x_2} & \frac{\partial v_1}{\partial x_2} &= -\frac{x_1}{x_2^2} & \frac{\partial v_2}{\partial v_1} &= \cos(v_1) \\ \frac{\partial v_3}{\partial v_2} &= \exp(v_2) & \frac{\partial v_4}{\partial v_2} &= \frac{1}{v_3} & \frac{\partial v_4}{\partial v_3} &= -\frac{v_2}{v_3^2}. \end{aligned}$$

Diese Ableitungen müssen hinterlegt werden. Da die Basisfunktionen jedoch in den meisten Fällen aus einer relativ kleinen Gruppe stammen, sind diese

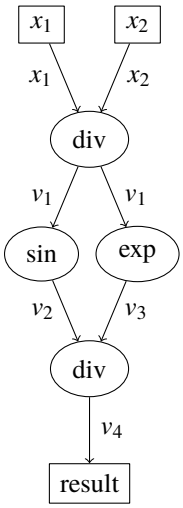

Abbildung 9: Darstellung der Beispielfunktion aus Gleichung 38 als Rechengraph (*computational graph*). Es gilt $v_1 = x_1/x_2$, $v_2 = \sin(v_1)$, $v_3 = \exp(v_1)$ und $v_4 = v_2/v_3$.

in den Paketen für automatische Differenziation meistens schon definiert. Für Funktionen, die darüber hinaus gehen, müssen der Operator und seine Ableitung bzw. Ableitungen registriert werden.

6.1 Forward Mode

Im *Forward-Mode* wertet man die Funktion nun genauso aus wie sonst – bezüglich der Klammern von innen nach außen. Dabei notiert man jedoch auch neben der Funktion stets die Ableitungen der einzelnen Ausdrücke nach dem gewünschten Parameter. Soll beispielsweise die Ableitung nach x_2 im Punkt $(1, 2)$ ermittelt

werden:

$$
\begin{aligned}
x_1 &= 1 \\
\dot{x}_1 &= 0 \\
x_2 &= 2 \\
\dot{x}_2 &= 1 \\
v_1 &= x_1/x_2 = 0{,}5 \\
\dot{v}_1 &= \frac{\partial v_1}{\partial x_1}\dot{x}_1 + \frac{\partial v_1}{\partial x_2}\dot{x}_2 = \frac{1}{x_2}\dot{x}_1 - \frac{x_1}{x_2^2}\dot{x}_2 = \frac{1}{2}0 - \frac{1}{2^2}1 = -0{,}25 \\
v_2 &= \sin(v_1) = \sin(1/2) = 0{,}479\,425\,539 \\
\dot{v}_2 &= \frac{\partial v_2}{\partial v_1}\dot{v}_1 = \cos(v_1)\dot{v}_1 = \cos(0{,}5)(-0{,}25) = -0{,}219\,395\,640 \\
v_3 &= \exp(v_1) = 1{,}648\,721\,271 \\
\dot{v}_3 &= \frac{\partial v_3}{\partial v_1}\dot{v}_1 = \exp(v_1)\dot{v}_1 = -0{,}412\,180\,317 \\
v_4 &= \frac{v_2}{v_3} = 0{,}290\,786\,288 \\
\dot{v}_4 &= \frac{\partial v_4}{\partial v_2}\dot{v}_2 + \frac{\partial v_4}{\partial v_3}\dot{v}_3 = \frac{1}{v_3}\dot{v}_2 - \frac{v_2}{v_3^2}\dot{v}_3 = -0{,}060\,373\,610
\end{aligned}
$$

Wie man sieht, tritt durch das Differenzieren kein Signifikanzverlust auf, wie bei der Verwendung von Differenzenquotienten. Weiterhin ist der Algorithmus relativ einfach und somit nicht viel langsamer als die Berechnung der Ableitungen mit Differenzenquotienten. Das algorithmische Differenzieren schafft also den oben genannten Mittelweg zwischen symbolischer Differenziation und numerischer Differenziation, den Mittelweg zwischen Genauigkeit und Geschwindigkeit.

Der dargestellte Algorithmus weist jedoch im Fall großer Parameterzahlen einen Nachteil auf: Bei jedem Zwischenschritt muss sowohl die Funktionsauswertung selbst, als auch die Ableitung nach jedem Parameter gespeichert werden. Bei großen Parameterzahlen wird der Algorithmus viel Speicher und Rechenleistung brauchen, die Rechenzeit skaliert mit der Parameterzahl.

6.2 Reverse Mode

Einen Ausweg aus diesem Problem bietet der zweite Basisalgorithmus des automatischen Differenzierens, der *Reverse Mode*. Bei diesem steigt der Aufwand für Ableitungen nach zusätzlichen Parametern nicht an, wie das folgende Beispiel

deutlich macht. Als Funktion wird wieder die aus Gleichung 38 betrachtet. Wie der Name *Reverse Mode* sagt, wird der Informationsfluss für die Gradientenbildung rückwärts durchschritten. Damit dies möglich ist, muss also die Funktion vorher einmal in Vorwärtsrichtung ausgeführt und alle Zwischenwerte aufgenommen worden sein. Es gilt also:

$$\begin{aligned} x_1 &= 1 \\ x_2 &= 2 \\ v_1 &= 0{,}5 \\ v_2 &= 0{,}479\,425\,539 \\ v_3 &= 1{,}648\,721\,271 \\ f = v_4 &= 0{,}290\,786\,288 \end{aligned}$$

$$\begin{aligned} \mathrm{d}f &= \tfrac{\partial f}{\partial v_2}\,\mathrm{d}v_2 + \tfrac{\partial f}{\partial v_3}\,\mathrm{d}v_3 \\ \mathrm{d}v_2 &= \tfrac{\partial v_2}{\partial v_1}\,\mathrm{d}v_1 = \cos(v_1)\,\mathrm{d}v_1 \\ \mathrm{d}v_3 &= \tfrac{\partial v_3}{\partial v_1}\,\mathrm{d}v_1 = \exp(v_1)\,\mathrm{d}v_1 \\ \mathrm{d}v_1 &= \tfrac{\partial v_1}{\partial x_1}\,\mathrm{d}x_1 + \tfrac{\partial v_1}{\partial x_2}\,\mathrm{d}x_2 = \tfrac{1}{x_2}\,\mathrm{d}x_1 - \tfrac{x_1}{x_2^2}\,\mathrm{d}x_2 \end{aligned}$$

Zusammengefasst folgt

$$\mathrm{d}f = \frac{1}{v_3}\cos(v_1)\left(\frac{1}{x_2}\,\mathrm{d}x_1 - \frac{x_1}{x_2^2}\,\mathrm{d}x_2\right) - \frac{v_2}{v_3^2}\exp(v_1)\left(\frac{1}{x_2}\,\mathrm{d}x_1 - \frac{x_1}{x_2^2}\,\mathrm{d}x_2\right). \quad (39)$$

Wie man sieht, erhält man sofort die Ableitungen nach den beiden Parametern:

$$\frac{\mathrm{d}f}{\mathrm{d}x_1} = \frac{1}{v_3}\cos(v_1)\frac{1}{x_2} - \frac{v_2}{v_3^2}\exp(v_1)\frac{1}{x^2} \quad (40)$$

und

$$\frac{\mathrm{d}f}{\mathrm{d}x_2} = \frac{1}{v_3}\cos(v_1)\left(-\frac{x_1}{x_2^2}\right) - \frac{x_2}{v_3^2}\exp(v_1)\left(\frac{x_1}{x_2^2}\right) \quad (41)$$

Für große Parameteranzahlen ist somit immer der *Reverse Mode* zu bevorzugen, der *Forward Mode* hat seine Vorteile dort, wo eine Vielzahl von Funktionen nach einem Parametersatz abgeleitet werden soll. Dies kommt in der Praxis üblicherweise seltener vor.
Ein zentrales Beispiel für die Ableitung einer Funktion nach sehr vielen Parametern ist die Backpropagation in Neuronalen Netzen aus dem Forschungsfeld

der künstlichen Intelligenz. Hier werden Netze aus nichtlinearen Funktionen programmiert, die näherungsweise jede nichtlineare Funktion nachbilden können. Die genaue Übertragungsfunktion eines solchen neuronalen Netzes wird dabei von den sogenannten *weights* und *biases* bestimmt, dies sind mehr oder weniger skalare Eingangsparameter der Funktion.
Nach genau diesen Gewichten wird abgeleitet, wenn ein neuronales Netz trainiert wird: Mittels Optimierungsalgorithmen werden die Gewichte so angepasst, dass die Trainingsdaten möglichst genau repliziert werden. Das Anlernen eines neuronalen Netzes ist also genau die gleiche Problemstellung wie die Anpassung von Modellparametern an Messdaten – lediglich die innere Struktur des Modells und des neuronalen Netzes unterscheiden sich.

6.3 Weiterführendes

Die beiden dargestellten Grundalgorithmen lassen sich auf beliebige numerische Programme anwenden – einzige Bedingung ist, dass für alle Grundoperationen die einfache Ableitung hinterlegt ist.
Auch Unterprogramme, Verzweigungen oder Schleifen sind kein Problem – der Rechengraph kann stets durch „Mitschreiben" der ausgeführten Blöcke generiert werden. Eine typische Technik hierfür ist das Überladen von Datentypen und Operatoren – immer wenn ein Operator auf eine Variable angewandt wird, wird im Rechenbaum eine Kante für die Variable und ein Knoten für den Operator hinterlegt. Beliebig komplexe numerische Modelle können so in einen Rechengraphen überführt und anschließend mit den Methoden des automatischen Differenzierens abgeleitet werden.
Der andere Ansatz, um das automatische Differenzieren zu implementieren, ist die Quelltexttransformation. Statt den Rechengraphen zur Laufzeit aufzustellen, wird er aus dem vorliegenden Vorwärts-Algorithmus im Vorhinein generiert. Dies ist technisch schwieriger umzusetzen, bietet jedoch Vorteile in der Geschwindigkeit, da bei der anschließenden Kompilierung Optimierungen vorgenommen werden.
Abgesehen von den beiden Basisalgorithmen wurden weitere Methoden entwickelt, um die Komplexität und damit die Rechenzeit der Algorithmen weiter zu reduzieren. Diese basieren oft auf der Idee der Taylor-Entwicklung oder der Operator-Algebra nach Heaviside.

Weiterhin kann das Prinzip des automatischen Differenzierens abstrahiert werden. Die Variablen müssen keine einzelnen Zahlen und die Operatoren keine einfa-

chen Funktionen sein. Es lässt sich genauso auf Vektoren, Matritzen, Tensoren allgemeiner Art und beliebige höherwertige Operatoren anwenden.

Dies macht sich das Paket *dolfin-adjoint* zunutze. Es wendet die dargestellten Grundprinzipien auf Berechnungen mit der Finite-Elemente-Methode an. Dabei sind Funktionen auf Finite-Elemente-Räumen (Vektoren mit Werten für die verschiedenen Werte in den Finiten Elementen) die Variablen, die Operatoren sind z.B. die *Assemblies*, also das Aufstellen von Steifigkeitsmatritzen aus den Differentialgleichungen, das Anwenden von Randbedingungen, das Lösen von Gleichungssystemen oder Operatoren und Funktionen im klassischen Sinn, wie z.B. die Addition von zwei FEM-Funktionen.
Für den oben erwähnten zweiten Projektteil soll dieses Paket verwendet werden – die Parameteridentifikation nach dem vorgeschlagenen Schema wird damit auch unter Einbezug der Geometrie möglich: Man kann so Gradienten eines Funktionals auf Finite-Elemente-Räumen nach beliebigen Eingangsparametern bestimmen.

Weitere Informationen zu *dolfin-adjoint* finden sich auf der Projektseite[1] und in der zugehörigen Veröffentlichung [17]. Das Finite-Elemente-Paket *Fenics/Dolfin*, auf das *dolfin-adjoint* aufsetzt, ist unter anderem in [18] und [19] beschrieben. Es gibt aber auch eine ausführliche Dokumentation im Internet[2], sowie Bücher [20], [21].

Informationen zum automatischen Differenzieren im Allgemeinen findet man z.B. bei [22] oder [23].

7 Implementation für das skalare Modell

Im folgenden Abschnitt soll nun erläutert werden, wie das Modell, seine Ableitung und die Optimierung genau implementiert wurden.
Die ersten Versuche, das Jiles-Atherton-Modell zu formulieren und abzuleiten, fanden aufgrund der hohen Entwicklungsgeschwindigkeit in der Programmiersprache *Python* statt. In dieser stehen mit *Algopy* [24], *autograd*[3] und *ad* [25] diverse Pakete zur Verfügung, die alle eine leicht zugängliche Programmierschnittstelle haben. Jedoch haben Tests gezeigt, dass die Rechengeschwindigkeit der Implementationen deutlich zu klein für eine sinnvolle Anwendung ist. Dies

[1] http://www.dolfin-adjoint.org
[2] https://fenicsproject.org
[3] https://github.com/HIPS/autograd

liegt in der Natur der Sprache Python, die aufgrund ihrer Einfachheit sehr schnelles Entwickeln erlaubt, jedoch nicht besonders effizient ist, da sie auf statische Typisierung verzichtet und zur Laufzeit interpretiert wird.
Das Thema der Geschwindigkeit wird (teils für andere Pakete) in [26] diskutiert. Im Vergleich verschiedener Implementationen von Automatischer Differenziation zeigt der Artikel, dass von den verglichenen Paketen ein auf *CppAD* [27] basierendes am schnellsten ist.
Aus diesem Grund wurden das Modell und die Differenziation in die Programmiersprache *C++* übertragen, welche numerisch um Größenordnungen schneller als Python ist und in der *CppAD* geschrieben ist. Um dennoch einfache und schnelle Auswertungen zu ermöglichen, wurde ein Wrapper mithilfe von *Cython*[4] implementiert – dadurch kann man den schnellen Algorithmus aus *Python* aufrufen.
CppAD wurde von Bradley M. Bell entwickelt und ist Teil des Mathematik-Software-Projekt *COIN-OR*.[5]

Um weiterhin die Portabilität des Codes zu erhöhen, wurde ein Docker-Container[6] für das Projekt aufgesetzt. Container bieten eine Möglichkeit, definierte Systemumgebungen (Betriebssystem, installierte Software, Build-Umgebungen etc.) auf einfache Weise zu definieren und insbesondere auf jedem Computer laufen zu lassen, auf dem Docker installiert ist. Um die Software der Arbeit auf einem PC zu nutzen, muss also nicht jede Abhängigkeit der Software[7] für das spezifische System kompiliert werden – stattdessen kann man einfach den Container starten, der alles enthält, was benötigt wird.
Der Aufbau der entwickelten Software ist in Abbildung 10 gegeben.

7.1 Resultate

Um die Funktion der Algorithmen zu testen, wurden Messdaten simuliert, es wurden also mithilfe des Vorwärtsmodells Kurven erzeugt, welche mit Rauschen versehen wurden. Anschließend wurden davon deutlich abweichende Startwerte für die Optimierung festgelegt und die Optimierung (unter Verwendung verschiedener Algorithmen aus dem Paket *SciPy* [28]) durchgeführt. Es konnte so gezeigt werden, dass die (simulierten) Messdaten in den meisten Fällen vom Programm repliziert werden konnten.

[4] http://cython.org

[5] Computational Infrastructure for Operations Research, https://www.coin-or.org

[6] https://www.docker.com

[7] Die verwendeten Pakete hängen v.a. von der *C++*-Bibliothek *Boost* (www.boost.org) ab, welche stets aus Quelldateien kompiliert werden muss.

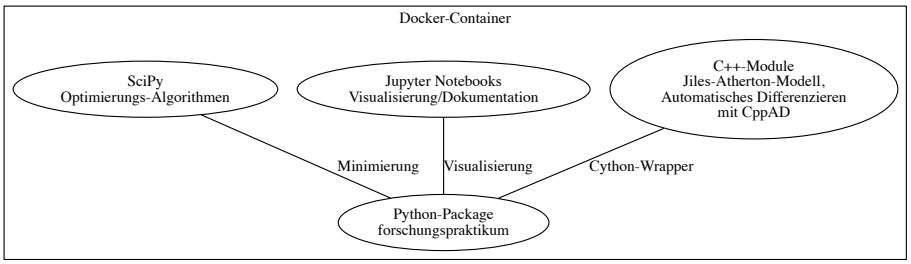

Abbildung 10: Struktur der Software.

Nach einigen Tests mit verschiedenen Algorithmen, hat sich TNC[8] als robust erwiesen, die Parameteridentifikation konvergierte in den meisten Fälle sehr gut und schnell. Auch bei großen Rauschleveln konnte fast immer die vorgegebene Toleranz erreicht werden.

Bei den Versuchen mit verschiedenen Rauschniveaus konnte jedoch ein weiteres Problem des Jiles-Atherton-Modells erkannt werden: Durch die Rauschanteile konvergierte die Funktion teilweise nicht zum Minimum, in dem die generierten Testdaten lagen, sondern zu einem Parametersatz, der eine sehr ähnliche Hysteresekurve hat, dessen Parameter jedoch deutlich von den eigentlichen abweichen. Abhilfe für dieses Problem schafft die Verwendung von vielen verschiedenen Aussteuerungen und Frequenzen, um die Parameter zu identifizieren. Je mehr Daten für die Anpassung verwendet werden und je verschiedener diese sind, desto besser werden die Werte des Algorithmus die Realität beschreiben – unter der Annahme, dass das Jiles-Atherton-Modell das reale Material ausreichend genau beschreibt.

Tatsächlich konnte auch festgestellt werden, dass die Verwendung des Gradienten die Parameteridentifikation massiv beschleunigt: Für die angepasste Kurve aus Abbildung 11 benötigt beispielsweise das gradientenfreie *Downhill-Simplex-Verfahren* [30] 989 Funktionsaufrufe, während das TNC-Verfahren mit dem Gradienten für die gleiche Zieltoleranz lediglich 25 Aufrufe benötigt. Ohne Gradientendaten konvergierte der TNC-Algorithmus überhaupt nicht – dieses Verhalten ließ sich auch für andere Verfahren feststellen.

[8]Truncated Newton Method, Implementation von SciPy [28]. Geht zurück auf eine Doktorarbeit von Stephen Nash, Übersichtsartikel vom gleichen Autor: [29].

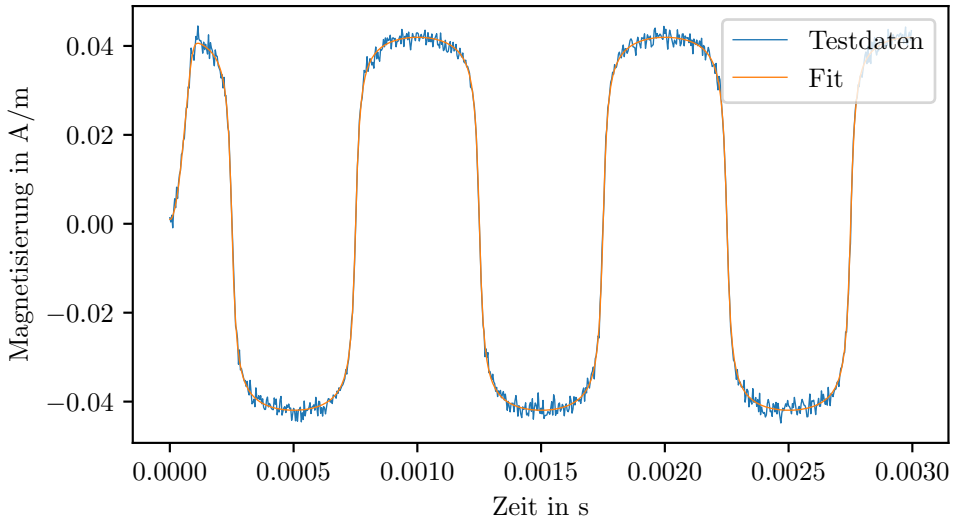

Abbildung 11: Beispiel eines angepassten Magnetisierungsverlaufes.

8 Ausblick

Wie zu Beginn beschrieben, ist das Ziel, das Jiles-Atherton-Modell ortsaufgelöst anzuwenden und insbesondere ein ortsaufgelöstes inverses Modell zu entwicklen, mit dem man die Geometrieabhängigkeit und auch den Einfluss von Wirbelströmen auf die Hysteresemessungen eliminieren kann. Die Grundlage hierfür wurden mit den ausgeführten Betrachtungen und dem erfolgreichen Test ohne Ortsauflösung gelegt. Der nächste Schritt ist der Entwurf des ortsaufgelösten Vorwärts-Modells mithilfe des Finite-Elemente-Pakets *Fenics*. Danach gilt es, einen Jiles-Atherton-Funktionsblock für *dolfin-adjoint* zu schreiben, der genaue Aufbau des zweiten Projektteils ist in Abbildung 12 dargestellt.
Neben diesen theoretischen Arbeiten müssen dann auch Messdaten gewonnen werden. Dazu ist im Rahmen eines anderen Vorhabens geplant, einen automatisierten Messaufbau zu entwerfen, der Kernverluste und Hysteresekurven Frequenz-, Temperatur- und Amplituden-abhängig vermisst.

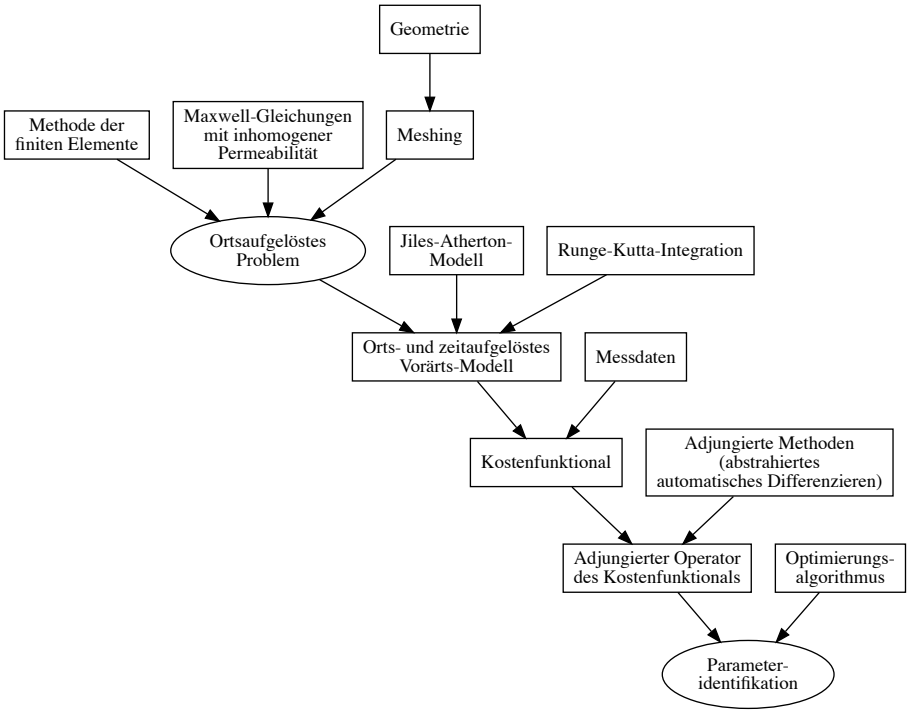

Abbildung 12: Vorgehen zur Parameteridentifikation im ortsaufgelösten Fall.

Literatur

[1] D. C. Jiles und D. L. Atherton, „Theory of ferromagnetic hysteresis (invited)“, *Journal of Applied Physics*, Jg. 55, Nr. 6, S. 2115–2120, 1984. DOI: 10.1063/1.333582. eprint: https://doi.org/10.1063/1.333582. Adresse: https://doi.org/10.1063/1.333582.

[2] F. Preisach, „Über die magnetische Nachwirkung“, *Zeitschrift für Physik*, Jg. 94, Nr. 5, S. 277–302, Mai 1935, ISSN: 0044-3328. DOI: 10.1007/BF01349418. Adresse: https://doi.org/10.1007/BF01349418.

[3] V. François-Lavet, F. Henrotte, L. Stainier, L. Noels und C. Geuzaine, „Vectorial Incremental Nonconservative Consistent Hysteresis model“, Nov. 2011.

[4] R. Szewczyk, „Computational Problems Connected with Jiles-Atherton Model of Magnetic Hysteresis“, in *Recent Advances in Automation, Robotics and Measuring Techniques*, R. Szewczyk, C. Zieliński und M. Kaliczyńska, Hrsg., Cham: Springer International Publishing, 2014, S. 275–283, ISBN: 978-3-319-05353-0.

[5] M. Abramowitz und I. A. Stegun, *Handbook of Mathematical Functions*, Ser. Applied Mathematics Series. National Bureau of Standards, Juni 1964, Bd. 55, ISBN: 9781614276173.

[6] M. Trapanese, „Identification of parameters of the Jiles–Atherton model by neural networks“, *Journal of Applied Physics*, Jg. 109, Nr. 7, S. 07D355, 2011. DOI: 10.1063/1.3569735.

[7] D. Lederer, H. Igarashi, A. Kost und T. Honma, „On the parameter identification and application of the Jiles-Atherton hysteresis model for numerical modelling of measured characteristics“, *IEEE Transactions on Magnetics*, Jg. 35, Nr. 3, S. 1211–1214, Mai 1999, ISSN: 0018-9464. DOI: 10.1109/20.767167.

[8] J. V. Leite, S. L. Avila, N. J. Batistela, W. P. Carpes, N. Sadowski, P. Kuo-Peng und J. P. A. Bastos, „Real coded genetic algorithm for Jiles-Atherton model parameters identification“, *IEEE Transactions on Magnetics*, Jg. 40, Nr. 2, S. 888–891, März 2004, ISSN: 0018-9464. DOI: 10.1109/TMAG.2004.825319.

[9] S. Cao, B. Wang, R. Yan, W. Huang und Q. Yang, „Optimization of hysteresis parameters for the Jiles-Atherton model using a genetic algorithm“, *IEEE Transactions on Applied Superconductivity*, Jg. 14, Nr. 2, S. 1157–1160, Juni 2004, ISSN: 1051-8223.

[10] M. Toman, G. Stumberger und D. Dolinar, „Parameter Identification of the Jiles–Atherton Hysteresis Model Using Differential Evolution“, *IEEE*

Transactions on Magnetics, Jg. 44, Nr. 6, S. 1098–1101, Juni 2008, ISSN: 0018-9464. DOI: 10.1109/TMAG.2007.915947.

[11] R. Marion, R. Scorretti, N. Siauve, M. Raulet und L. Krahenbuhl, „Identification of Jiles–Atherton Model Parameters Using Particle Swarm Optimization“, *IEEE Transactions on Magnetics*, Jg. 44, Nr. 6, S. 894–897, Juni 2008, ISSN: 0018-9464. DOI: 10.1109/TMAG.2007.914867.

[12] F. R. Fulginei und A. Salvini, „Softcomputing for the identification of the Jiles-Atherton model parameters“, *IEEE Transactions on Magnetics*, Jg. 41, Nr. 3, S. 1100–1108, März 2005, ISSN: 0018-9464. DOI: 10.1109/TMAG.2004.843345.

[13] E. D. M. Hernandez, C. S. Muranaka und J. R. Cardoso, „Identification of the Jiles–Atherton model parameters using random and deterministic searches“, *Physica B: Condensed Matter*, Jg. 275, Nr. 1, S. 212–215, 2000, ISSN: 0921-4526. DOI: https://doi.org/10.1016/S0921-4526(99)00766-8. Adresse: http://www.sciencedirect.com/science/article/pii/S0921452699007668.

[14] K. Chwastek und J. Szczyglowski, „Estimation methods for the Jiles-Atherton model of hysteresis - a review“, Dez. 2008.

[15] N. Sadowski, N. J. Batistela, J. P. A. Bastos und M. Lajoie-Mazenc, „An inverse Jiles-Atherton model to take into account hysteresis in time-stepping finite-element calculations“, *IEEE Transactions on Magnetics*, Jg. 38, Nr. 2, S. 797–800, März 2002, ISSN: 0018-9464. DOI: 10.1109/20.996206.

[16] A. J. Bergqvist, „A simple vector generalization of the Jiles-Atherton model of hysteresis“, *IEEE Transactions on Magnetics*, Jg. 32, Nr. 5, S. 4213–4215, Sep. 1996, ISSN: 0018-9464. DOI: 10.1109/20.539337.

[17] P. Farrell, D. Ham, S. Funke und M. Rognes, „Automated Derivation of the Adjoint of High-Level Transient Finite Element Programs“, *SIAM Journal on Scientific Computing*, Jg. 35, Nr. 4, S. C369–C393, 2013. DOI: 10.1137/120873558.

[18] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. Rognes und G. Wells, „The FEniCS Project Version 1.5“, *Archive of Numerical Software*, Jg. 3, Nr. 100, 2015, ISSN: 2197-8263. DOI: 10.11588/ans.2015.100.20553. Adresse: https://journals.ub.uni-heidelberg.de/index.php/ans/article/view/20553.

[19] A. Logg und G. N. Wells, „DOLFIN: Automated Finite Element Computing“, *ACM Trans. Math. Softw.*, Jg. 37, Nr. 2, 20:1–20:28, Apr. 2010, ISSN: 0098-3500. DOI: 10.1145/1731022.1731030.

[20] H. P. Langtangen und A. Logg, *Solving PDEs in Python: The FEniCS Tutorial I*, 1st. Springer Publishing Company, Incorporated, 2017, ISBN: 3319524615, 9783319524610.

[21] A. Logg, K.-A. Mardal und G. Wells, *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. Springer Publishing Company, Incorporated, 2012, ISBN: 3642230989, 9783642230981.

[22] U. Naumann, *The Art of Differentiating Computer Programs*.: Society for Industrial und Applied Mathematics, 2011. DOI: 10.1137/1.9781611972078.

[23] A. Griewank und A. Walther, *Evaluating Derivatives*, Second.: Society for Industrial und Applied Mathematics, 2008. DOI: 10.1137/1.9780898717761.

[24] S. F. Walter und L. Lehmann, „Algorithmic differentiation in Python with AlgoPy“, *Journal of Computational Science*, Jg. 4, Nr. 5, S. 334–344, 2013, ISSN: 1877-7503. DOI: http://dx.doi.org/10.1016/j.jocs.2011.10.007. Adresse: http://www.sciencedirect.com/science/article/pii/S1877750311001013.

[25] A. D. Lee, *ad: a Python package for first- and second-order automatic differentation*. Adresse: http://pythonhosted.org/ad/.

[26] A. Turkin und A. Thu, „Benchmarking Python Tools for Automatic Differentiation“, *CoRR*, Jg. abs/1606.06311, 2016. Adresse: http://arxiv.org/abs/1606.06311.

[27] B. M. Bell, „CppAD: a package for C++ algorithmic differentiation“, *Computational Infrastructure for Operations Research*, 2012.

[28] E. Jones, T. Oliphant, P. Peterson u. a., *SciPy: Open source scientific tools for Python*, [Online; accessed <today>], 2001–. Adresse: http://www.scipy.org/.

[29] S. G. Nash, „A survey of truncated-Newton methods“, *Journal of Computational and Applied Mathematics*, Jg. 124, Nr. 1, S. 45–59, 2000, Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations, ISSN: 0377-0427. DOI: https://doi.org/10.1016/S0377-0427(00)00426-X. Adresse: http://www.sciencedirect.com/science/article/pii/S037704270000426X.

[30] J. A. Nelder und R. Mead, „A Simplex Method for Function Minimization“, *The Computer Journal*, Jg. 7, Nr. 4, S. 308–313, 1965. DOI: 10.1093/comjnl/7.4.308. Adresse: http://dx.doi.org/10.1093/comjnl/7.4.308.