Parameter-Optimierung eines Brake-by-Wire-Pedals

Jennifer Werner[1], Martin Düsing[1*], Bernhard Bachmann[2], Ali Kemal Kücükyavuz[1]

[1]HELLA GmbH & Co. KGaA Beckumer Straße 130, 59555 Lippstadt, Deutschland; **martin.duesing@hella.com*
[2]Fachhochschule Bielefeld, Fachbereich Ingenieurwissenschaften und Mathematik, Interaktion 1, 33609 Bielefeld

Kurzfassung. Das hydraulische Bremspedal wird aufgrund verschiedener Vorteile bei modernen Fahrzeugen durch Brake-by-Wire-Pedale ersetzt, die keine mechanische Verbindung zum Bremssystem haben, sondern nur noch elektronische Signale vom Pedalsensor zum Bremssteuergerät übertragen. Da sich das hydraulische und das Brake-by-Wire-Pedal für den Fahrer weitestgehend gleich bedienen und anfühlen sollen, ist die Kennlinie Pedalkraft über Weg eine entscheidende Anforderung. Diese Anforderung ist je nach Fahrzeughersteller und Fahrzeug unterschiedlich. In dieser Arbeit wird ein Simulationsmodell vorgestellt, dass während der Entwicklung eines Brake-by-Wire-Pedals zur Anwendung kam. Der Fokus dieser Arbeit liegt auf einer mathematischen Optimierung des ursprünglichen Modells zur Erreichung einer neuen Kundenkennlinie.

Einleitung

Brake-by-Wire-Pedale stellen eine grundlegende Änderung in Bremssystemen von Automobilen dar. In klassisch hydraulischen Bremssystemen betätigt ein Bremspedal mechanisch einen Hydraulikkolben. In Brake-by-Wire-Systemen gibt es nur noch elektronische Signale, die vom Bremspedalsensor an das Steuergerät zur Bremsung übergeben werden.

Diese Technik bietet verschiedene Vorteile. So lässt sich die Ansprechzeit verkürzen und dadurch der Bremsweg verringern. Da Komponenten wie Bremskraftverstärker, Hauptbremszylinder und ABS wegfallen oder Funktionen durch Software realisiert werden, können Kosten gespart werden. In Elektro- oder Hybridfahrzeugen kann das Betätigen des Brake-by-Wire-Pedals auch als Auslöser für die Rekuperation verwendet werden, sodass bei leichtem Auslösen Energie zurückverwandelt wird, anstatt sie als Wärme abzugeben.

Während der Entwicklung eines Brake-by-Wire-Pedals bei der Firma Hella wurden zu unterschiedlichen Konzepten jeweils Modelle zur Simulation verwendet. Die Modelle werden mit der Sprache Modelica in der 3DExperience Plattform und in Dymola entwickelt.

Ziel der Simulation ist die möglichst präzise Vorhersage der Kennlinie Pedalkraft über Pedalweg oder synonym über Pedalwinkel für festgelegte Geschwindigkeiten des Pedals. Diese Kennlinien werden von den Fahrzeugherstellern vorgeben und sind eine entscheidende Anforderung an das Produkt.

Nach der erfolgreichen Entwicklung und Produktion des Brake-by-Wire-Pedals richtet sich der Fokus stärker auf die Frage der Weiterentwicklung, die sich besonders auf die Erreichung anderer Kennlinien bezieht. Dazu wird hier eine mathematische Optimierung eingesetzt.

Ziel dieser Arbeit ist die Parameter-Optimierung eines Modelica-Modells, um vorgegebene Kennlinien zu treffen. Die Optimierung findet in MATLAB statt.

Die Arbeit gliedert sich in fünf Teile. Im ersten Teil wird das Modelica-Modell des Pedals beschrieben, während im zweiten Teil Grundlagen der Parameteroptimierung zusammengefasst werden. Der dritte Teil befasst sich mit einer neuen MATLAB-Klassenumgebung, die zur praktischen Umsetzung genutzt wird. Im vierten Teil wird die Optimierung durchgeführt und die Ergebnisse präsentiert. Abschnitt 5 fasst die Ergebnisse der Arbeit zusammen.

1 Modell des Brake-by-Wire-Pedals

Zur Modellierung des Brake-by-Wire-Pedals wird die objektorientierte Modellierungssprache Modelica verwendet. Als Entwicklungsumgebung wird sowohl Dymola als auch die 3DExperience Plattform von Dassault Systèmes verwendet. Die 3DExperience Plattform bietet den Vorteil, dass die Geometrie und damit die Hebellängen, Volumen, Dichten und Massenträgheitsmomente automatisch vom CAD-Modell in das Modelica-Modell übernommen werden können. Nur die Kinematik, Reibung und Kontakte müssen getrennt, speziell modelliert

ARGESIM Report 59 (ISBN 978-3-901608-93-3), p 267-272, DOI: 10.11128/arep.59.a59038

werden. Liegt allerdings noch kein fertiges CAD-Modell vor, so kann dieser Weg nicht beschritten werden. Ist das CAD-Modell noch nicht fertig, die kinematisch wichtigen Abmessungen liegen aber bereits vor und sollen auch durch eine Simulation auf ihre Funktionalität hin überprüft werden, so lässt sich mit Hilfe der Modelica Standard Library auch effizient ein Modell aufbauen. Mit Modelica.Mechanics.MultiBody.Parts.BodyBox, Modelica.Mechanics.MultiBody.Joints.Revolute und Modelica.Mechanics.MultiBody.Forces.Spring lassen sich große Teile des Pedals beschreiben.

In diesem Fall wurden die für die Kinematik wichtigen Massenträgheitsmomente nur grob angenähert. Hoch dynamische Lastfälle lassen sich damit nicht simulieren. Quasistatische Versuche können aber mit sehr kurzen Simulationszeiten durchgeführt werden.

Die Verifizierung des Modells mit einer im Entwicklungsprozess deutlich später durchgeführten Messung zeigt eine hervorragende Übereinstimmung. Abbildung 1 zeigt die Messdaten in blau und die Simulationsergebnisse in rot. Die Kraft in N wird über den Pedalwinkel in ° dargestellt. Es sind zwei Kurven mit Hin- und Rückweg zu sehen. Die oberen Kurven sind die Hin- und die unteren die Rückkurven. Die Hysterese ist typisch für Bremspedale.

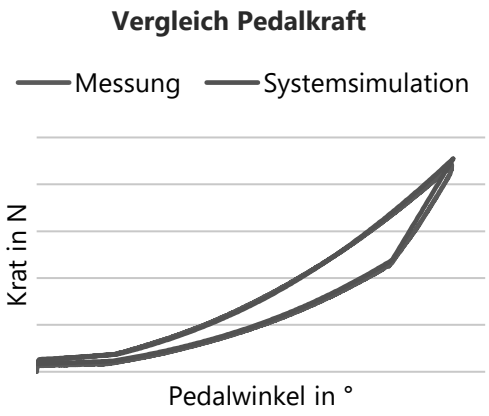

Abbildung 1: Vergleich von Messung und Simulation

2 Grundlagen der Parameteroptimierung

Während einer Optimierung werden verschiedene Alternativen bezüglich eines Zielkriteriums verglichen und unter allen betrachteten Alternativen wird die beste gesucht [1]. Die Alternativen unterscheiden sich durch verschiedene Parametereinstellungen des Modells, die variiert werden.

Die nichtlineare Optimierung kommt bei unterschiedlichen Modellen aus Natur-, Ingenieur- und Wirtschaftswissenschaften zum Einsatz, beispielsweise bei geometrischen Problemen, mechanischen Problemen, Parameter-Fitting-Problemen, Schätzproblemen, Approximationsproblemen und bei der Sensitivitätsanalyse [1]. Unterschieden wird zwischen unrestringierten und restringierten nichtlinearen Optimierungsproblemen.

Optimierungsprobleme ohne Nebenbedingungen sind unrestringierte Optimierungsprobleme [2]. Ein nichtlineares Optimierungsproblem ohne Restriktionen besitzt die Form

$$\min_{x\in\mathbb{R}^n} f(x) \tag{1}$$

mit $f(x)\colon \mathbb{R}^n \to \mathbb{R}$ [1].

Ein nichtlineares Optimierungsproblem mit Nebenbedingungen wird als restringiertes nichtlineares Optimierungsproblem bezeichnet. Es ist gegeben durch:

$$\min_{x\in\mathbb{R}^n} f(x) \tag{2}$$

so dass

$$g(x) \leq 0 \tag{3}$$

$$h(x) = 0 \tag{4}$$

und es wird angenommen, dass die Funktionen $f(x)\colon \mathbb{R}^n \to \mathbb{R},\ g(x)\colon \mathbb{R}^n \to \mathbb{R}^r,\ h(x)\colon \mathbb{R}^n \to \mathbb{R}^m$ zweimal stetig differenzierbar sind [3]. Die Funktion $f(x)$ wird dabei Zielfunktion genannt. Die Funktion $g(x)$ beschreibt die Ungleichheitsbedingungen und die Funktion $h(x)$ die Gleichheitsbedingungen des restringierten nichtlinearen Optimierungsproblems.

Ein nichtlineares Least-Square-Problem (NLLSP) ist ein Minimierungsproblem der Form

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\sum_{i=1}^{m} f_i(x)^2, \quad m \geq n \tag{5}$$

wobei jedes $f_i(x),\ i = 1, \dots, m$ eine nichtlineare Funktion im $\mathbb{R}^n$ ist [4].

Ein wichtiger Einsatzbereich für NLLSPs ist der Bereich des Data Fitting. Hier ist es das Ziel einen gegebenen Datensatz $(y_i, t_i), i = 1, \dots, m$ an eine Modell-Funktion $g(x, t)$ anzupassen. Sei

$$f_i(x) = y_i - g(x, t_i), i = 1, \dots, m, \tag{6}$$

so führt dies auf ein Least-Square-Problem der Art (5) (siehe [4]).

3 MATLAB-Klassenumgebung

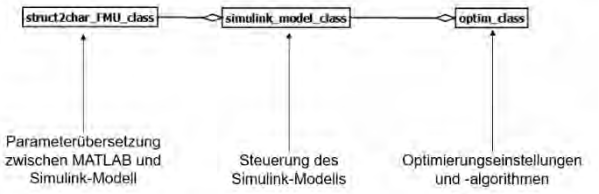

Abbildung 2: UML-Diagramm MATLAB-Klassenumgebung

Das in Abschnitt 2 beschriebene Modelica-Modell wird als FMU aus Dymola exportiert und in MATLAB Simulink importiert, siehe Abbildung 3. Der Block ist der FMU-Import aus Dymola, welcher das Modelica-Modell des Brake-by-Wire-Pedals beinhaltet. Dieser hat mehrere Ausgäng, allerdings sind für die Parameteroptimierung nur die Ausgänge *FootForce* (entspricht der Pedalkraft) und der Ausgang *PedalTravel* (entspricht dem Pedalweg) notwendig. Zur Optimierung soll das Simulink-Modell mit der FMU direkt aus MATLAB gestartet werden. Die MATLAB-Klassenumgebung dient zum einfachen Starten der Modellsimulation, zur Veränderung der Parameter des Modells und zum Starten der Optimierung. Abbildung 2 zeigt den Aufbau der Klassenumgebung mit den drei Klassen *struct2char_FMU_class*, *simulink_model_class* und *optim_class*.

struct2char_FMU_class

Diese Klasse dient als Übersetzer zwischen den MATLAB-Parametern und den Simulink-Parametern. Dies ist notwendig, da die Parameter aus dem Simulink-Modell in Form eines Strings vorliegen. Für die MATLAB-Klasse *simulink_model_class* muss dieser String in eine Struktur umgewandelt. Damit die Parameter wieder in das Simulink-Modell eingegeben werden können, wird in der Klasse *struct2char_FMU_class* die Struktur wieder in einen String umgewandelt. Die Struktur kann dabei beliebig oft verschachtelt sein oder auch eine mehrdimensionale Struktur sein und die Felder der Struktur können verschiedene Datentypen besitzen.

simulink_model_class

Diese Klasse ist zur Steuerung des Simulink-Modells programmiert worden. Es ist möglich, die Parameter des Simulink-Modells anzusprechen. Außerdem ist es möglich, Default-Werte für die Parameter festzulegen und das Simulink-Modell mit einer Funktion auf die entsprechenden Default-Werte zurückzusetzen. Die Simulation des Simulink-Modells kann ebenfalls mit der Klasse gestartet

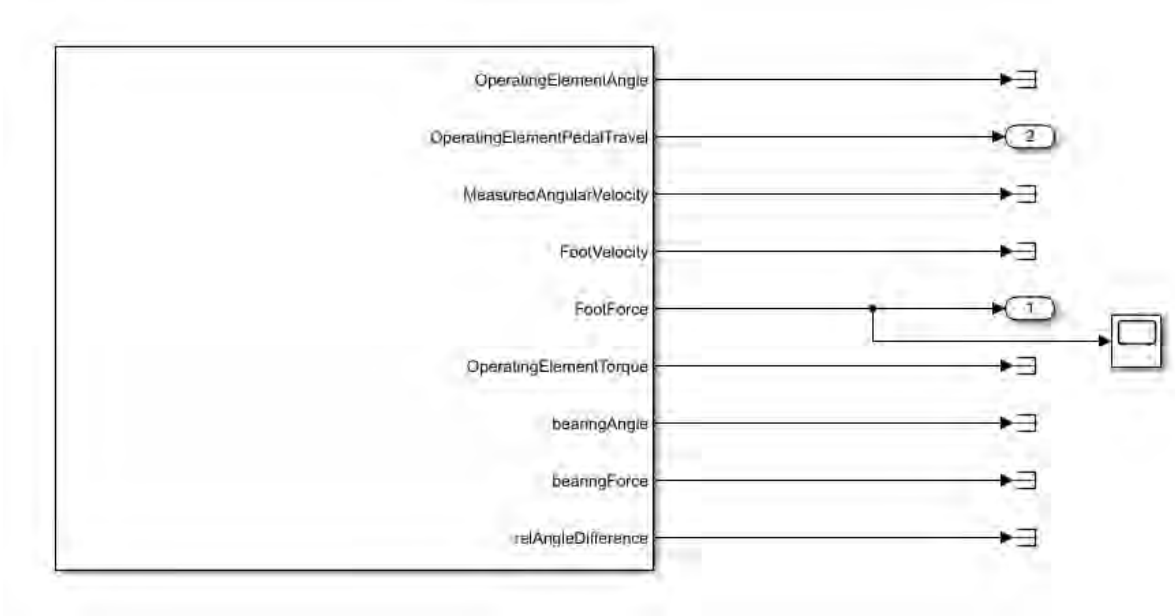

Abbildung 3: FMU-Import in Simulink des Modelica-Brake-by-Wire-Pedals

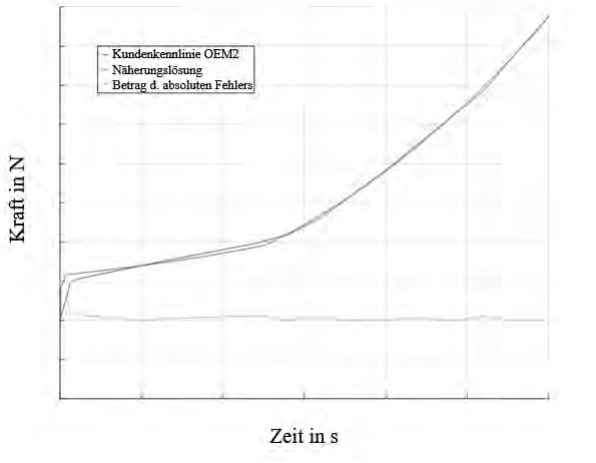

Abbildung 4: Ergebnis der Optimierung für OEM2

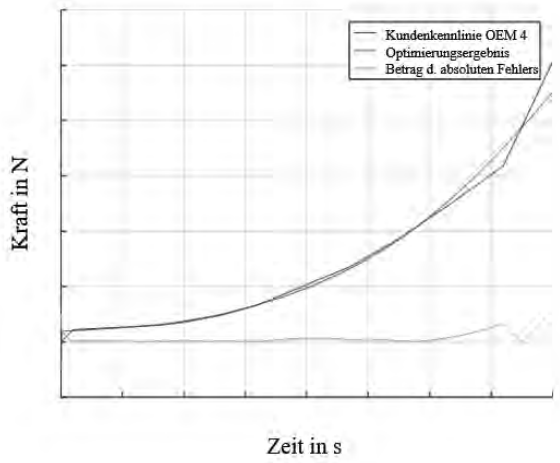

Abbildung 5: Ergebnis der Optimierung für OEM4

werden und das Ergebnis der Simulation wird automatisch in einem Attribut der Klasse abgespeichert.

Nach verschiedenen Tests in Simulink zeigt sich, dass der Solver ode23s (stiff/Mod. Rosenbrock) am schnellsten und robustesten im Vergleich zu den anderen verfügbaren Solvern ist. Dabei ist eine relative Fehlertoleranz von 10^{-4} ausgewählt. Die absolute Fehlertoleranz wird auf den Wert *auto* gestellt. Bei dem Bremspedal-Modell handelt es sich um ein sehr steifes System und an den Kontaktstellen kommt es zu numerischen Schwierigkeiten. In den Tests hat sich gezeigt, dass diese Stellen mit einer sehr kleinen minimalen Schrittweite überwunden werden können.

optim_class

Mit dieser Klasse ist die Steuerung des Optimierungsprozesses des Simulink-Modells möglich. In der Klasse wird zur Optimierung die MATLAB-Funktion *lsqcurvefit* aufgerufen. Dabei handelt es sich um ein nichtlineares Least-Square-Data-Fitting-Problem. Diese Funktion ist Teil der Optimization Toolbox und wird beschrieben durch

$$\min_x \sum_{k=1}^{n} (F(x, xdata_k) - ydata_k)^2 \qquad (7)$$

Dabei ist die Funktion $F(x, xdata)$ vom Anwender definiert. In diesem Fall ist es das Simulationsergebnis des Bremspedal-Modells, konkret die diskreten Werte der Pedalkraft. Die Größe der Vektoren $xdata$ und $ydata$ entspricht n und ist gleich der Anzahl der Zeitschritte. Der Vektor $xdata$ enthält die diskreten Zeitpunkte, der Vektor $ydata$ die diskreten Zielwerte der Optimierung. Die Tuner-Parameter befinden sich im Vektor x der Länge m, die der Anzahl alle Tuner-Parameter entspricht.

Der Funktion *lsqcurvefit* müssen die Funktion $F(x, xdata)$, die Startwerte x_0 sowie die unteren und oberen Grenzen *lowerBound* und *upperBound* für die Tuner-Parameter übergeben werden. Außerdem können noch Einstellungen hinsichtlich des von der Funktion verwendeten Optimierungsalgorithmus übergeben werden. Diese Übergabeparameter sind in der Klasse *optim_class* als Attribute definiert. Die MATLAB-Funktion *lsqcurvefit* stellt den Levenberg-Marquardt-Algorithmus (Details siehe [4] und [5]) sowie ein Trust-Region-Verfahren [5] zur Lösung des NLLSP zur Verfügung.

4 Ergebnisse der Optimierung

Abbildungen 4 und 5 zeigen Ergebnisse der Optimierung für zwei verschiedene Kundenkennlinien mit dem gleichen Pedalmodell. Zehn verschiedene Parameter des Modells werden während der Optimierung mit dem Ziel angepasst, die jeweils blau dargestellte Kundenkennlinie im Kraft-Zeit Diagramm zu erreichen. Anstelle des Kraft-Weg-Diagramms wird in der Optimierung auf das Kraft-Zeit-Diagramm zurückgegriffen, da dieses Vorgehen für die Optimierung einfacher ist. Die Pedale werden mit einer niedrigen konstanten Geschwindigkeit bewegt. Zeit und Position des Pedals sind also linear abhängig, sodass beide Varianten die gleiche Aussage haben.

Die roten Kurven in Abbildungen 4 und 5 sind die Näherungslösungen, die mit der Klassenfunktion *optimize_simulink_model* erreicht werden. Der Betrag des

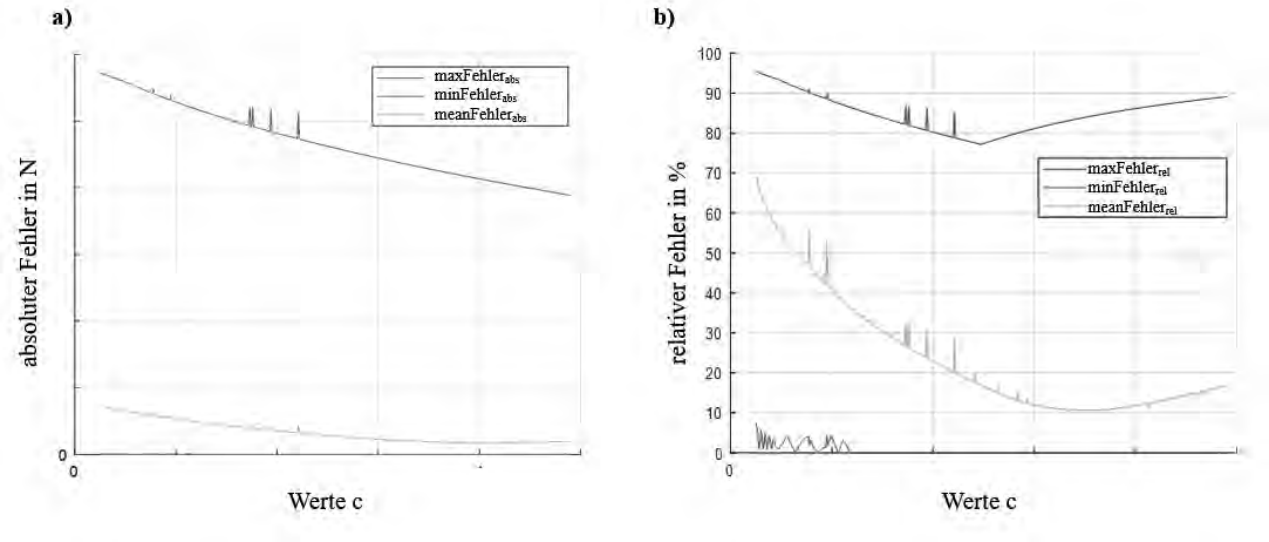

Abbildung 6: Absoluter (a) und relativer Fehler (b) einer Variation über den Parameter c

absoluten Fehlers ist in gelb eingetragen. Für die Optimierung in Abbildung 4 ist der Betrag des absoluten Fehlers im Zeitpunkt 0 s am höchsten während der Durchschnittsfehler bei nur 3,11 % liegt. Das Ergebnis für die Optimierung in Abbildung 5 hat einen Durchschnittsfehler von 8,80 %.

Die Ergebnisse zeigen, dass mit einem Pedal, nur durch die Änderung einzelner Komponenten eine deutlich andere Kennlinie erreichbar ist. Die Abweichung im hinteren Bereich in Abbildung 5 ist auf den Kontakt des Pedals mit dem Anschlag zurückzuführen. Dieser Kontakt wurde im untersuchten Modell vernachlässigt und kann daher hier nicht besser getroffen werden.

Abbildung 6.a zeigt den Betrag des absoluten Fehlers in N über der Federkonstante c einer Feder für verschiedene Werte von c. Dargestellt werden der maximale Fehler, der minimale Fehler und der Durchschnittsfehler über die Zeit oder äquivalent den Weg. Die Werte der Federkonstanten sind in N/m dargestellt. Der minimale Fehler ist immer Null. Das bedeutet, dass für jeden getesteten Parameter zumindest zu einem Zeitpunkt die Zielkurve perfekt erreicht wird. Der maximale Fehler zeigt wiederum, dass zu einem Zeitpunkt der Fehler sehr hoch ist. Der Durchschnittsfehler zeigt den Fehler als Durchschnitt über den gesamten Zeitraum an. Abbildung 6.b zeigt die relativen Fehler der gleichen Parametervariation in % an.

Um die Fehler besser analysieren zu können bieten sich dreidimensionale Darstellungen an, die die Zeitkomponente zusätzlich aufnehmen.

Die Diagramme in Abbildung 7 zeigen eine entsprechende Darstellung der gleichen Parametervariation. Es wird deutlich, dass wie auch in Abbildung 5 zu erkennen ist, die großen Fehler erst zum Ende der Simulation auftreten. An dieser Stelle ist das Pedal maximal ausgelenkt und trifft auf einen Anschlag. Dieser Anschlag wurde hier nicht speziell modelliert und führt daher zu den hier dargestellten Abweichungen. Die maximale Fehlerkurve aus Abbildung 6.a lässt sich in Abbildung 7.a gut als die oberste Kante identifizieren.

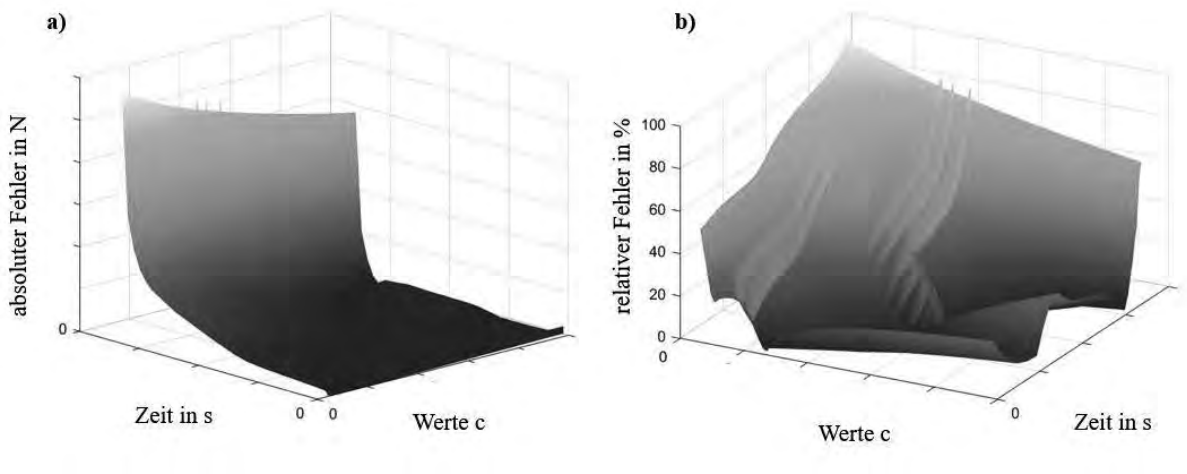

Abbildung 7: Absoluter (a) und relativer Fehler (b) einer Variation über den Parameter c4 und die Zeit

5 Zusammenfassung

Die Ergebnisse der Arbeit zeigen, dass ein vorhandenes und verifiziertes Simulationsmodell genutzt werden kann, um die Weiterentwicklung des Produktes massiv zu unterstützen. Ein fertig entwickeltes Pedal mit einer charakteristischen Kennlinie zwischen Kraft und Auslenkung lässt sich mit Hilfe der Parameteroptimierung durch den Austausch einzelner Komponenten, wie zum Beispiel der Federn, auf eine andere Kennlinie hin anpassen.

Um die neuen Werte für die Parameter zu finden wird ein nichtlineares Optimierungsproblem formuliert und gelöst. Ein vorhandenes und verifiziertes Modelica-Modell wird als FMU aus Dymola exportiert und mit einem Least-Square-Ansatz in MATLAB optimiert.

Literaturverzeichnis

[1] Stein, Oliver. *Grundzüge der Nichtlinearen Optimierung*. Springer-Verlag, 2017.

[2] Reinhardt, Rüdiger; Hoffmann, Armin; Gerlach, Tobias. *Nichtlineare Optimierung: Theorie, Numerik und Experimente*. Springer-Verlag, 2012.

[3] Biegler, Lorenz T. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. Society for Industrial and Applied Mathematics, 2010.

[4] Björck, Åke. *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics, 1996.

[5] Nocedal, Jorge; Wright, Stephen. *Numerical optimization*. Springer Science & Business Media, 2006.