Lehrstuhl für Umformtechnik

Masterarbeit

Inverse Finite-Elemente Analyse zur Auswertung des Heißtorsionsversuches

Alexander Gerald Wenda, BSc

Mai 2022

MONTANUNIVERSITÄT LEOBEN
www.unileoben.ac.at

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich die Richtlinien des Senats der Montanuniversität Leoben zu "Gute wissenschaftliche Praxis" gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 19.05.2022

Unterschrift Verfasser/in
Alexander Gerald Wenda

Danksagung

Diese Arbeit ist am Lehrstuhl für Umformtechnik an der Montanuniversität Leoben in Zusammenarbeit mit der voestalpine Forschungsservicegesellschaft Donawitz GmbH in Donawitz entstanden.

Mein besonderer Dank gilt Herrn Prof. Dipl.-Ing. Dr.techn. Martin Stockinger, dem Leiter des Lehrstuhls, der meine Masterarbeit betreut und begutachtet hat. Bei meinem Betreuer DDipl.-Ing. Dr.mont. Benjamin Ralph möchte ich mich ebenfalls für seine tatkräftige Unterstützung bedanken.

Außerdem möchte ich noch Dr.mont. Phillip Haslberger für die Durchführung und Betreuung der Torsionsversuche danken. Herrn Dr.mont. Robert Kaiser gilt mein Dank für seine Hilfestellungen zu numerischen Modellierung.

Weiters möchte ich mich noch bei meinen Freunden bedanken. Speziell möchte ich Maximilian Staudacher erwähnen, der mir immer freundschaftlich aber auch fachlich zu Seite gestanden ist.

Mein herzlicher Dank gilt auch meiner Familie, ganz besonders meinen Eltern, sie haben mich im Studium immer unterstützt.

Zu guter Letzt möchte ich mich noch bei meiner langjährigen Freundin Olivia bedanken, die mich während des Studiums immer motiviert hat und immer für mich da war.

Kurzfassung

Der Heißtorsionsversuch ermöglicht die Analyse und Simulation der Warmumformung bis zu großen Umformgraden. Die in der Probe auftretenden Gradienten von Temperatur, Umformgrad und Dehnrate resultieren in einem komplexen Werkstoffverhalten. Das Ziel dieser Arbeit ist, die Versuchsbedingungen besser zu verstehen, und eine Auswertung für den Versuch umzusetzen. Dafür wurden Torsionsversuche an einem austenitischen Stahl (A220) in einer Gleeble 3800 durchgeführt. Im Rahmen der Auswertung wurde ein Finite-Elemente Modell in ABAQUS® implementiert. Die Parametrisierung des Hensel-Spittel Materialmodells erfolgte durch eine inverse Analyse der experimentellen Torsionsdaten. Dabei wird von Startwerten ausgehend jener Parametersatz gesucht, der die geringste Abweichung zwischen numerischen und experimentellen Daten aufweist. Dieses Optimierungsproblem wurde über den Nelder-Mead Algorithmus gelöst. In den experimentellen Ergebnissen hat sich gezeigt, dass bestimmte Versuchsbedingungen eine lokale Instabilität forcieren. Das kann zu einem großen lokalen Temperaturanstieg führen, welcher in stark lokalisierter Verformung resultiert. Eine Auswertung nach der klassischen analytischen Methode kann dann nicht mehr zuverlässig durchgeführt werden. Die durchgeführte Auswertung über eine inverse Analyse, hat eine gute Übereinstimmung zu den experimentellen Ergebnissen gezeigt. Im Vergleich der bestimmten Materialparameter mit Literaturdaten aus Zylinderstauchversuchen, war ebenfalls eine gute Übereinstimmung zu sehen.

Keywords: Heißtorsionsversuch, Finite-Elemente Modell, inverse Analyse, Hensel-Spittel Werkstoffmodell

Abstract

The hot torsion test enables the analysis and simulation of hot forming up to large strains. The gradients of temperature, strain and strain rate occurring in the specimen result in complex material behavior. The aim of this work is to gain a better understanding of the test conditions, and to implement an evaluation for the test. For this purpose, torsion tests were carried out on an austenitic steel (A220) in a Gleeble 3800. As part of the evaluation, a finite element model was implemented in ABAQUS®. The parameterization of the Hensel-Spittel constitutive model was done by an inverse analysis of the experimental torsion data. Starting from an initial guess, the parameter set with the smallest deviation between numerical and experimental data is searched for. This optimization problem was solved using the Nelder-Mead algorithm. In the experimental results it has been shown that certain experimental conditions promote a local instability. This could lead to a significant temperature rise, which results in flow localization. An evaluation using the classical analytical method can then no longer be carried out reliably. The evaluation carried out via an inverse analysis, has shown a good agreement with the experimental results. The comparison of the determined constitutive parameters were in good agreement with literature data from compression tests.

Keywords: hot torsion test, finite element analysis, inverse analysis, Hensel-Spittel constitutive model

Inhaltsverzeichnis

Verwendete Symbole und Abkürzungen

Symbol	
γ	Scherung oder Scherwinkel
γ^*	Scherung am effektiven Radius des Torsionsversuches
$\varepsilon_{Strahlung}$	Emissionskoeffizient
ϑ	Temperatur
ν	Querkontraktionszahl
σ	Spannung
σ_1, σ_2, σ_3	Hauptnormalspannungen
σ_f	Fließspannung
σ_x, σ_y, σ_z	Normalspannungskomponenten des Spannungstensors
$\sigma_{Kontakt}$	Elektrische Kontaktleitfähigkeit
τ	Schubspannung
τ^*	Schubspannung am effektiven Radius des Torsionsversuches
τ_{xy}, τ_{yz}, τ_{zx}	Schubspannungskomponenten des Spannungstensors
ϕ	Gesamter Verdrehwinkel der Torsion
φ	Umformgrad
$\dot{\varphi}$	Umformgeschwindigkeit
a	Abstand von der Torsionsachse bis zur Zylinderoberfläche
A, m_1, m_2, m_4, m_5, m_7, m_8	Parameter des Hensel-Spittel Modells
E	E-Modul
h_T	Thermische Kontaktleitfähigkeit
h_W	Wärmeübergangkoeffizient
j	Stromdichte
K_d	Differentielle Verstärkung des PID-Reglers
K_i	Integrale Verstärkung des PID-Reglers
K_p	Proportionale Verstärkung des PID-Reglers
l	Länge des Torsionsquerschnitts

$\hat{M}_T$	Experimentelles Torsionsmoment
M_T	Torsionsmoment
Q	Wärmeenergie
r	Radialer Abstand
r^*	Effektiver Radius des Torsionsversuches
T_1	Temperatur der Torsionsprobe am Messpunkt 1
T_2	Temperatur der Torsionsprobe am Messpunkt 2
T_3	Temperatur der Torsionsprobe am Messpunkt 3
T_4	Temperatur der Torsionsprobe am Messpunkt 4
t_d	Differentielle Zeitkonstante des PID-Reglers
t_i	Integrale Zeitkonstante des PID-Reglers
U	Elektrisches Potential
u_1, u_2, u_3	Verschiebungen
u_{R1}, u_{R2}, u_{R3}	Rotationen
Abkürzung	
D.S.I	Dynamic Systems inc.
FE	Finite-Elemente
KNN	Künstliches neuronales Netz

1 Einleitung und Problemstellung

Für die Herstellung, Verarbeitung und Formgebung metallischer Werkstoffe spielt die Warmumformung eine entscheidende Rolle. Die präzise Vorhersage des Fließverhaltens, vor allem bei hohen Dehnungen und Dehnraten, stellt einen wichtigen Teil davon dar. Für diesen Zweck wurden in der Literatur etliche Materialmodelle entwickelt. Diese lassen sich anhand ihrer Herkunft in drei generelle Bereiche einteilen [1]: phänomenologisch basiert, physikalisch basiert und basierend auf künstlichen neuronalen Netzwerken. All diesen Modellen ist gemein, dass sie den sehr komplexen Zusammenhang zwischen Dehnung, Dehnrate, Temperatur und Fließspannung über eine unterschiedliche Anzahl an Parametern in eine geschlossene Form bringen. Die große Herausforderung steckt in der Parametrisierung dieser Modelle für einen konkreten Werkstoff. Das kann nur unter Berücksichtigung von experimentellen Versuchsdaten erfolgen, die über einen weiten Bereich von Dehnung, Dehnrate und Temperatur erfasst werden. Dazu wurden in der Vergangenheit einige Prüfverfahren entwickelt [2]. Zu den am häufigsten verwendeten Methoden zählt der Heißzugversuch und der Zylinderstauchversuch. Ein weiteres Verfahren ist der Heißtorsionsversuch, welcher einige besondere Vorteile bietet. Dazu zählen die Unabhängigkeit von Reibungsbedingungen, die Erreichung großer Verformungen (ohne eine geometrische Instabilität zu durchlaufen) und die Vielfalt der möglichen Versuchsführungen [3].

Ausgehend von den Ergebnissen der Umformversuche kann die Parametrisierung eines Materialmodells in einem direkten Ansatz oder über eine inverse Analyse erfolgen. Ein direkter Ansatz setzt eine analytische Beziehung voraus, welche die experimentellen Daten mit den für das Materialmodell benötigten Größen verknüpft. Diese Beziehungen sind für Heißzug- und Zylinderstauchversuch bekannt und existieren unter gewissen Bedingungen auch für den Heißtorsionsversuch. Im Gegensatz dazu wird in einer inversen Analyse der direkte Zusammenhang zwischen den Parametern des Materialmodells und den experimentellen Versuchsdaten über die Formulierung eines numerischen Ansatzes (beispielsweise mit der Finite Elemente Methode) hergestellt. Die Charakterisierung der Parameter erfolgt durch einen Optimierungsprozess, der in Hinblick auf die experimentellen Ergebnisse durchgeführt wird.

Ein direkter Ansatz für den Heißtorsionsversuch wurde erstmalig von Fields und

Backofen hergeleitet [4]. Analog zu dieser Auswertung wurden weitere Ansätze formuliert [5,6], die aber ebenfalls auf vereinfachenden Annahmen basieren. Die direkte Auswertung in diesen Methoden setzt immer eine homogene Verdrehung voraus, es darf zu keiner lokalisierten Torsion kommen. Als Auslöser einer solchen lokalen Instabilität kann ein Temperaturgradient oder auch entfestigendes Materialverhalten fungieren. Ein potentieller Temperaturgradient muss dabei nicht schon vor der Torsion vorhanden sein, sondern kann auch erst während dieser entstehen. Die Herausforderung besteht dann in der Auswertung der Versuchsdaten, sodass diese für die Charakterisierung der Materialmodelle verwendet werden können. Eine vielversprechende Lösung dieses Problems besteht in der inversen Analyse. Dieser Weg wurde in der Literatur schon öfter gewählt und hat zu erfolgversprechenden Ergebnissen geführt [7–11].

In dieser Arbeit wird der Heißtorsionsversuch in einer Gleeble 3800 untersucht und eine Auswertung der Versuchsdaten erarbeitet. Ziel ist es ein FE-Modell zu entwickeln, das alle für Mikrostruktur und Festigkeit relevanten Parameter ortsaufgelöst abbilden kann und so in der Lage ist, das lokale Materialverhalten in einer Torsionsprobe abzuschätzen. Im Rahmen der Auswertung soll die Parametrisierung des im FE-Modells verwendeten Materialmodells erfolgen. Diese soll auf den Rohdaten des Versuches basieren und damit die Möglichkeit zur Erstellung von temperatur- und dehnratenabhängigen Fließkurven bieten. Die erfolgreiche Auswertung soll anhand einer Versuchsserie mit der Gleeble und den daraus gewonnenen Daten validiert werden.

2 Theoretische Grundlagen

2.1 Warmumformung

In diesem Abschnitt werden die prinzipiellen Vorgänge bei der Warmumformung und insbesondere deren mathematische Beschreibung behandelt. Dabei wird auf die Fließkurve und auf die Fließbedingung näher eingegangen. Abschließend werden materialwissenschaftliche Phänomene, die wesentlichen Einfluss auf den Verlauf der Fließkurve haben, genauer erläutert.

2.1.1 Fließkurve

Die Fließkurve beschreibt den Zusammenhang zwischen der Fließspannung σ_f (wahre Spannung) und dem Umformgrad φ. Im Vergleich zur Spannungs-Dehnungskurve beginnt die Fließkurve erst bei Einsetzen von plastischer Verformung, der linear-elastische Bereich wird nicht berücksichtigt. Sie charakterisiert damit das Werkstoffverhalten unter fortwährender plastischer Verformung. Daher dient die Fließkurve als Grundlage zur Bestimmung vieler umformtechnischer Kennwerte, wie etwa Umformkraft, Umformarbeit, Formänderungen sowie Spannungen.

Als Fließspannung σ_f ist jene Spannung definiert, bei der im Werkstoff plastisches Fließen beginnt bzw. fortwährt. Der vorherrschende Spannungszustand wird dabei als einachsig und homogen angenommen.

Für die Bestimmung der Fließspannung werden etliche Verfahren angewandt. Dazu zählen unter anderem der Zugversuch, der Stauchversuch, sowie der Torsionsversuch. Eine gute Übersicht ist in [2, S. 66] zu finden. Die Wahl des verwendeten Prüfverfahrens hängt maßgeblich vom Umformverfahren, für das die Fließspannung bestimmt wird, ab. Je ähnlicher der im Werkstoff vorherrschende Spannungszustand ist, desto besser geeignet ist es. Zusätzlich muss das Prüfverfahren in der Lage sein, allfällige werkstoffkundliche Besonderheiten zu erfassen. Dazu zählt Anisotropie, die insbesondere bei Blechwerkstoffen häufig auftritt. Die klassischen Fließkriterien (zum Beispiel: Tresca, v. Mises) können diese in Betrachtung eines dreidimensionalen Spannungszustandes nicht berücksichtigen. Es bedarf besonderer Fließbedingungen, die die Beschreibung der Anisotropie ermöglichen [12–14].

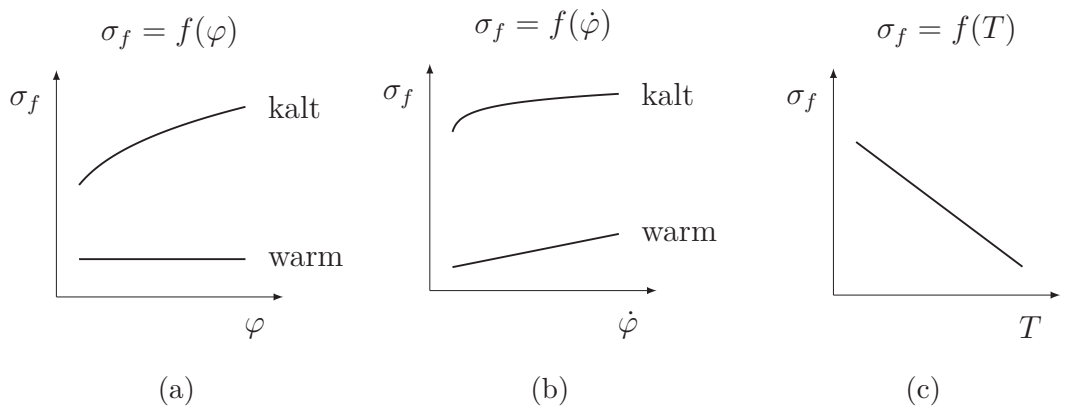


Abbildung 2.1: Prinzipielles Verhalten der Fließspannung in Abhängigkeit von (a) Umformgrad, (b) Umformgeschwindigkeit und (c) Temperatur. Nach [2].

Die Fließspannung σ_f ist von vielen Einflussfaktoren bestimmt. Diese können werkstoffabhängig sowie werkstoffunabhängig sein. Die mit Abstand wichtigsten, werkstoffunabhängigen Größen sind:

- Umformgrad φ,
- Umformgeschwindigkeit $\dot{\varphi}$ und
- Umformtemperatur T.

Deren schematischer Einfluss auf die Fließspannung σ_f ist in Abbildung 2.1 dargestellt. Dabei sind die beiden grundlegenden Temperaturbereiche der Kalt- und Warmumformung zu unterscheiden. Während die Kaltumformung bei Umgebungstemperatur stattfindet, liegt die Temperatur bei der Warmumformung über der Rekristallisationstemperatur des Werkstoffes. Dazwischen existiert noch die Halbwarmumformung, welche aber nur eine untergeordnete Rolle spielt. [2]

Generell ist zu sagen, dass bei der Kaltumformung die Fließspannung hauptsächlich vom Umformgrad abhängt, während die Umformgeschwindigkeit nur eine untergeordnete Rolle spielt. Die Begründung hierfür liegt in einem starken Anstieg der Versetzungsdichte während der Umformung, welche zu einer Zunahme der Fließspannung führt. Anders ist das Verhalten bei der Warmumformung. Da bei dieser die Temperatur über der Rekristallisationstemperatur liegt, kann es während der Umformung zu Erholungs- und Rekristallisationsprozessen kommen (siehe Abschnitt 2.1.3). Diese resultieren in mikrostrukturellen Umwandlungen, sodass die Fließspannung nur gering vom Umformgrad beeinflusst wird. Es entsteht ein Gleichgewicht zwischen Versetzungsverfestigung und Entfestigung durch Erholung oder Rekristallisation. Beeinflusst wird dieses Gleichgewicht hauptsächlich durch die

Umformgeschwindigkeit. Liegt die Umformgeschwindigkeit über der Rekristallisationsgeschwindigkeit, dann führt eine Zunahme der Umformgeschwindigkeit zu einem Anstieg der Fließspannung. [2, 15]

Die Modellierung der Fließkurve stellt eine große Herausforderung dar. Ein ideales Fließspannungsmodell wäre in der Lage, den Einfluss von Umformgrad, Umformgeschwindigkeit, Temperatur und Verfestigungsverhalten exakt zu beschreiben. Dazu wird in Abschnitt 2.1.4 ein tieferer Einblick über gängige Modelle gegeben.

2.1.2 Fließbedingung

In diesem Kapitel wird der Übergang von elastischem zu plastischem Materialverhalten beschrieben. Die sogenannte Fließbedingung bildet die mathematische Grundlage dafür. Um das plastische Verhalten an einem beliebigen Spannungszustand beschreiben zu können, müssen drei Beziehungen bekannt sein [16].

- Die Fließbedingung: Sie beschreibt den Zusammenhang zwischen den Komponenten des Spannungstensors im Augenblick, in dem das Material plastisch zu fließen beginnt.
- Die Fließregel: Sie beschreibt den Zusammenhang zwischen Umformgeschwindigkeit und Spannung.
- Das Verfestigungsgesetz: Es beschreibt die Änderung der anfänglichen Fließspannung mit fortschreitender Verformung.

Der Übergang von elastischem zu plastischem Materialverhalten findet statt, wenn die im Material auftretende Spannung einen kritischen Wert, die Fließspannung, übersteigt. Im Falle von einachsigem Zug kann die Fließspannung aus dem Spannungs-Dehnungsdiagramm als Streckgrenze ermittelt werden. Ist diese dabei nicht eindeutig zu erkennen, muss eine alternative Bestimmungsmethode gewählt werden. Dazu zählen beispielsweise die Verwendung einer Dehngrenze oder eine Streckgrenzbestimmung durch physikalische Größen.

Tritt ein mehrachsiger Spannungszustand auf, reicht der direkte Vergleich mit der Fließspannung nicht aus, um das Fließverhalten zu beurteilen. Es ist notwendig eine Fließbedingung zu formulieren, die die Spannungskomponenten in Beziehung zur Fließspannung (bestimmt im einachsigen Versuch) setzt. Üblicherweise wird dazu eine implizite Funktion F folgender Form definiert [16] :

$$F(\sigma_1, \sigma_2, \sigma_3, \sigma_f) = 0 \tag{2.1}$$

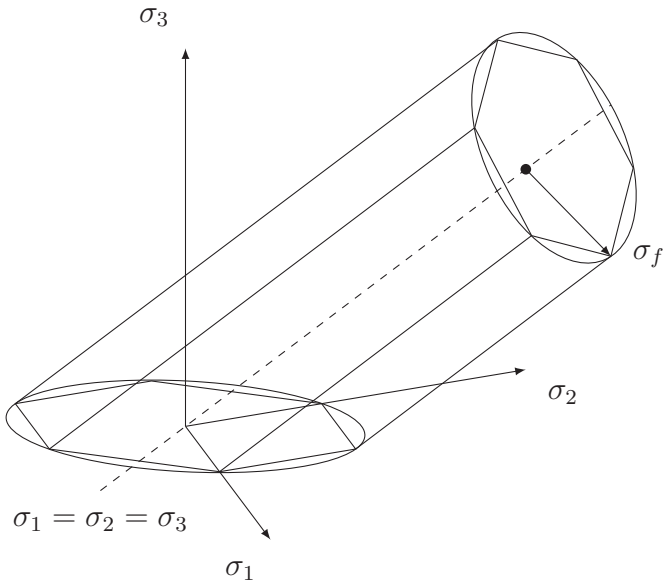


Abbildung 2.2: Fließortkurve nach Tresca (Sechseckiges Prisma) und v. Mises (Zylinder), angelehnt an [16].

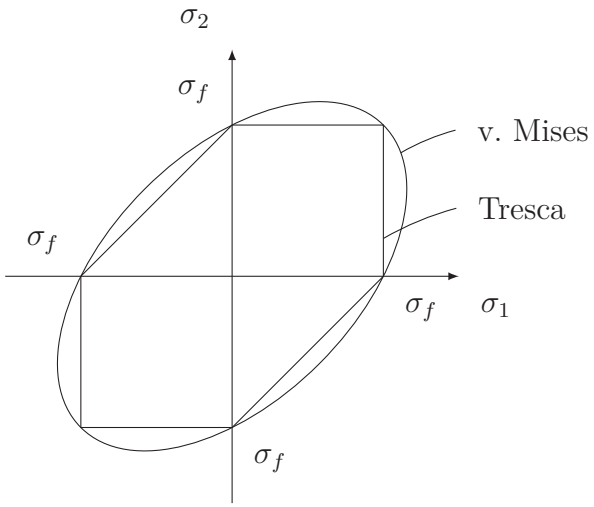


Abbildung 2.3: Fließbedingungen nach Tresca und von Mises, für den Sonderfall des ebenen Spannungszustandes $\sigma_3 = 0$. Angelehnt an [16].

σ_1, σ_2 und σ_3 sind dabei die Hauptnormalspannungen und σ_f ist die im einachsigen Versuch bestimmte Fließspannung. Die zwei am meisten angewandten Fließbedingungen stammen von Tresca (Schubspannungskriterium) und von Huber-Mises-Hencky (Gestaltänderungskriterium) [17]. Geometrisch betrachtet handelt es sich bei der Fließbedingung um eine Fläche, die im von den drei Hauptnormalspannungen aufgespannten Raum liegt. Diese wird auch Fließortkurve genannt. In Abbildung 2.2 ist die Fließortkurve für die Fließbedingungen von Tresca und von Mises dargestellt. Alle Punkte die im Inneren der Fläche liegen ($F < 0$), zeigen elastisches Materialverhalten. Punkte die direkt auf der Fläche liegen ($F = 0$), zeigen plastisches Materialverhalten. Die Punkte, die außerhalb der Fließfläche liegen ($F > 0$), haben keine physikalische Bedeutung. Der Sonderfall des ebenen Spannungszustandes ist in Abbildung 2.3 dargestellt. Dort vereinfacht sich die Fläche zu einem Sechseck für die Fließbedingung von Tresca und zu einer Ellipse für die Fließbedingung von Mises.

Tresca-Fließbedingung

Die Fließbedingung von Tresca geht von einem Gleiten der Kristallebenen durch auftretende Schubspannungen aus. Der Übergang vom elastischen in den plastischen Bereich ist durch das Erreichen einer kritischen Schubspannung τ_{krit} gegeben. Treten die drei Hauptnormalspannungen σ_1, σ_2 und σ_3 auf und gilt $\sigma_1 > \sigma_2 > \sigma_3$, dann ist die größte auftretende Schubspannung als

$$\tau = \pm \frac{\sigma_1 - \sigma_3}{2} \tag{2.2}$$

gegeben. Wird diese allein durch die Vergleichsspannung σ_V hervorgerufen, dann gilt $\tau = \pm\ \sigma_V/2$. Die Vergleichsspannung nach Tresca des dreiachsigen Spannungszustandes ergibt sich damit als

$$\sigma_V = \pm\ (\sigma_1 - \sigma_3). \tag{2.3}$$

Bei reinem Schub ergibt sich somit ein Spannungsverhältnis von

$$\sigma_V = 2\tau \tag{2.4}$$

und ein Dehnungsverhältnis von

$$\varepsilon_V = \frac{2}{3}\gamma \tag{2.5}$$

zwischen der Vergleichsdehung ε_V und dem Scherwinkel γ.

Huber-Mises-Hencky Fließbedingung

Die Huber-Mises-Hencky Fließbedingung, kurz genannt von Mises Fließbedingung, geht von der Annahme aus, dass nur eine Gestaltänderung zu plastischem Fließen führen kann. Damit haben alle hydrostatischen Anteile des Spannungstensors keinen Einfluss auf den Übergang zwischen elastischem und plastischem Bereich. Es kommt zu plastischem Fließen, wenn eine kritische Gestaltänderungsenergie $w_{G\ krit}$ überschritten wird.

Für den dreichachsigen Spannungszustand ergibt sich die Gestaltänderungsenergie als

$$w_G = \frac{1-\nu}{3E}\ \left[\sigma_x^2+\sigma_y^2+\sigma_z^2-(\sigma_x\sigma_y+\sigma_y\sigma_z+\sigma_z\sigma_x)\right]+\frac{1}{2G}(\tau_{xy}^2+\tau_{yz}^2+\tau_{zx}^2), \quad (2.6)$$

wobei ν die Querkontraktionszahl, σ_i die Normalspannungskomponente und τ_{ij} die Schubspannungskomponente in i,j Richtung bezeichnet. Wird diese allein durch die Vergleichsspannung σ_V verrichtet, dann folgt:

$$w_G = \frac{1-\nu}{3E}\ \sigma_V^2. \quad (2.7)$$

Die Vergleichsspannung nach Mises lässt sich unter Berücksichtigung des Schubmoduls $G=\frac{E}{2(1+\nu)}$ folglich als

$$\sigma_V = \pm\sqrt{\sigma_x^2+\sigma_y^2+\sigma_z^2-(\sigma_x\sigma_y+\sigma_y\sigma_z+\sigma_z\sigma_x)+3(\tau_{xy}^2+\tau_{yz}^2+\tau_{zx}^2)} \quad (2.8)$$

ausdrücken. Bei reinem Schub ergibt sich somit ein Spannungsverhältnis von

$$\sigma_V = \sqrt{3}\tau. \quad (2.9)$$

und ein Dehnungsverhältnis von

$$\varepsilon_V = \frac{\gamma}{\sqrt{3}} \quad (2.10)$$

zwischen der Vergleichsdehung ε_V und dem Scherwinkel γ.

2.1.3 Erholung & Rekristallisation

Durch die Verformung werden neue Versetzungen erzeugt und damit die freie Energie innerhalb der Kristallstruktur erhöht. Dies führt zu einer thermodynamischen Instabilität. Das Material ist bestrebt diese Änderung so gering wie möglich zu halten oder gänzlich abzubauen. Die Mechanismen, die wieder zu einer Reduktion der Versetzungsdichte führen, sind stark temperaturabhängig und laufen bei niedrigen homologen Temperaturen nur sehr langsam ab. Dadurch bleibt bei der Kaltumformung das instabile Verformungsgefüge nach beendeter Umformung weiter bestehen (Abbildung 2.4a).

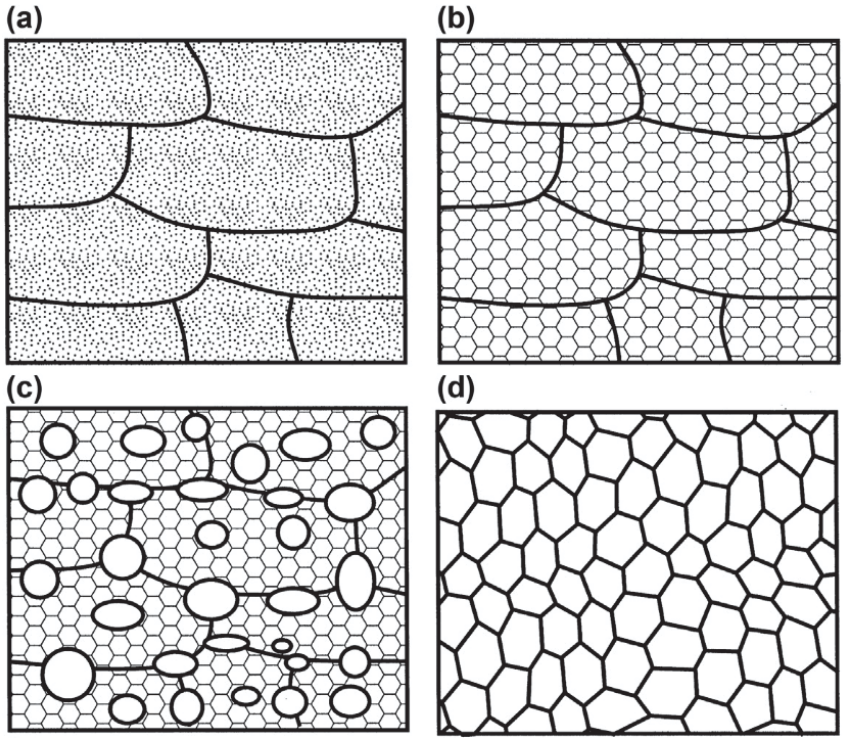


Abbildung 2.4: Schematische Darstellung der Prozesse Erholung und Rekristallisation. (a) verformter Zustand; (b) erholtes Gefüge; (c) teilweise rekristallisiertes Gefüge; und (d) vollständig rekristallisiertes Gefüge, [18].

Erst bei höheren homologen Temperaturen werden thermisch aktivierte Prozesse wie Festkörperdiffusion ermöglicht. Diese sind in der Lage, die Versetzungsdichte zu reduzieren oder die Versetzungsstruktur energetisch günstiger zu gestalten. Wird das Material auf eine höhere Temperatur gebracht, um diese Prozesse zu ermöglichen, spricht man vom Anlassen.

Ein wichtiger Prozess während des Anlassens ist die Erholung. Das Gefüge wird dabei durch Auslöschung und Restrukturierung der Versetzungen zurück in Richtung des unverformten Ausgangsgefüge geführt (Abbildung 2.4b). Die Korngrenzen zwischen den verformten Körnen werden davon üblicherweise nicht beeinflusst. Wenn die Erholung schon während der Umformung stattfindet, spricht man von dynamischer Erholung.

Die Erholung führt nicht zu einer vollständigen Wiederherstellung der ursprünglichen Eigenschaften, da die Versetzungsstruktur nicht komplett ausgelöscht wird. Der ausgebildete Zustand ist energetisch günstiger, aber trotzdem nur metastabil.

Einen Schritt weiter geht die sogenannte Rekristallisation. Hier entstehen aus dem verformten oder erholten Gefüge neue versetzungsarme Körner (Abbildung 2.4c). Diese wachsen und verdrängen das bestehende stark verformte Gefüge. Das

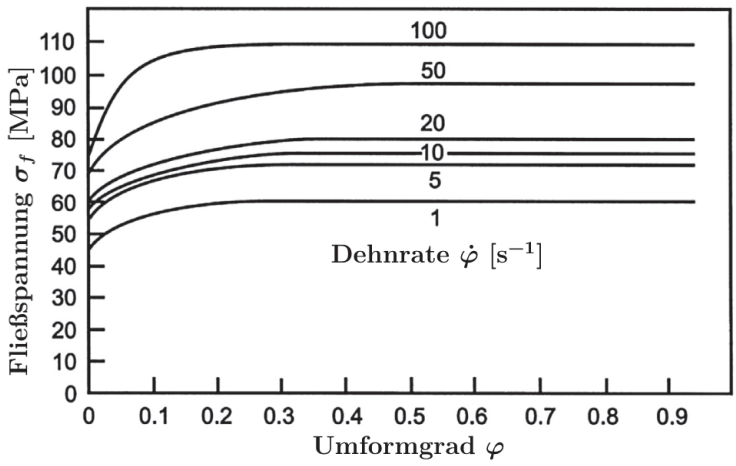


Abbildung 2.5: Fließkurven von Al-1%Mg bei 400 °C [19]. Gut zu erkennen ist der steady-state Bereich, der etwa ab einem Umformgrad von 0,3 auftritt.

Ergebnis ist ein neues Gefüge mit deutlich reduzierter Versetzungsdichte (Abbildung 2.4d). Wenn es noch während der Verformung zur Rekristallisation kommt, so spricht man von dynamischer Rekristallisation. [18]

Die dynamische Erholung und Rekristallisation spielen in der Warmumformung eine entscheidende Rolle und beeinflussen maßgeblich die Form der Fließkurve. Welcher der beiden Prozesse dabei vorwiegend stattfindet, hängt vom Werkstoff ab. So tritt bei Metallen mit hoher Stapelfehlerenergie (Aluminium und Aluminiumlegierungen, ferritische Stähle) hauptsächlich dynamische Erholung auf. Die Versetzungen können relativ einfach klettern und quergleiten. Die Fließkurve zeigt zu Beginn Verfestigung, die dann in einen Bereich konstanter Fließspannung übergeht (Abbildung 2.5). Zu Beginn der Verformung steigt die Versetzungsdichte an und führt somit zu einer höheren Fließspannung. Allerdings nimmt auch die Triebkraft für die Erholung zu und es werden Kleinwinkelkorngrenzen und Subkörner gebildet. Dies geschieht solange, bis die Verfestigung und Erholung im dynamischen Gleichgewicht stehen. Danach bleibt die Fließspannung konstant, im sogenannten steady-state Bereich (Abbildung 2.5). [18]

In Metallen mit geringer Stapelfehlerenergie (austenitische Stähle, Kupfer, ...) laufen die Erholungsprozesse nur sehr langsam ab. Bei der Warmumformung kommt es zu dynamischer Rekristallisation. Diese tritt ab einem kritischen Umformgrad auf. Bei der dynamischen Rekristallisation entstehen neue versetzungsfreie Körner, die zu wachsen beginnen. Durch die Umformung steigt auch in diesen Körnern die Versetzungsdichte an. Durch die steigende Zahl an Versetzungen wird das Wachstum gebremst und kommt schließlich zum Halt. Zusätzlich führt die Nukleation von neuen Körnern an den wachsenden Korngrenzen zu einem verlangsamenden Effekt.

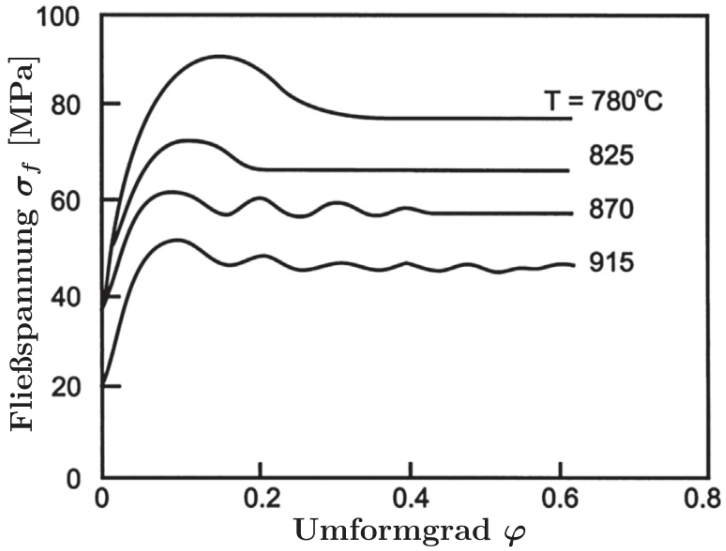


Abbildung 2.6: Fließkurve von 0.68%C-Stahl bei unterschiedlichen Temperaturen [20].

Die Fließkurve zeigt dabei ein Spannungsmaximum, das dann in einen Bereich konstanter, aber niedrigerer Fließspannung übergeht (Abbildung 2.6). Die resultierende Korngröße steigt dabei monoton mit sinkender Fließspannung. [18]

2.1.4 Materialmodell

Mit dem Materialmodell wird die mathematische Verbindung zwischen Umformgrad, Umformgeschwindigkeit, Temperatur und Fließverhalten hergestellt. Grundlage für die Beschreibung dieses Verhaltens bilden thermo-mechanische Experimente (beispielsweise Stauch-, Zug- und Torsionsversuche), in welchen die Umformparameter über weite Bereich variiert werden und davon abhängig die Fließspannung bestimmt wird. Die Beschreibung der so gemessenen Zusammenhänge obliegt dann dem Materialmodell. In den letzten Jahrzehnten wurden etliche Modelle entwickelt, um das Warmumformverhalten möglichst exakt zu beschreiben. Generell lassen sich diese in drei Kategorien einteilen [1]:

1. **Phänomenologisch**: Die Beschreibung des Fließverhaltens basiert rein auf empirischen Observationen. Dazu werden mathematische Funktionen verwendet, die den experimentellen Daten möglichst nahe kommen. Durch den fehlenden physikalischen Hintergrund ist die Gültigkeit dieser Modelle auf kleine Bereiche, hinsichtlich Umformgeschwindigkeit, Spannungszustand, Temperatur, Werkstoff und Ausgangszustand des Werkstoffes limitiert. Vorteile bestehen in der oft reduzierten Anzahl von Materialkonstanten und der relativ

leichten Kalibrierung dieser. Ferner lassen sich diese Modelle oft mit relativ geringem Aufwand in Computersimulationen implementieren. Einige Beispiele stellen das Johnson-Cook Modell [21], das Kocks-Mecking Modell [22] oder das Hensel-Spittel Modell [23] dar.

2. **Physikalisch-basiert**: Diese Modelle berücksichtigen physikalische Hintergründe in der Beschreibung des Fließverhaltens. Dazu zählen Thermodynamik, thermisch aktivierte Versetzungsbewegung und die Kinetik der Versetzungsbewegung. Im Vergleich zu den phänomenologischen Modellen haben die physikalisch basierten Modelle einen deutlich erweiterten Gültigkeitsbereich bezüglich der Verformungsbedingungen. Damit einher geht aber oft eine größere Anzahl an Materialkonstanten. Zu diesen Modellen zählen beispielsweise das Zerilli–Armstrong Modell [24] oder spezielle Zelluläre Automaten [25].

3. **Künstliche neuronale Netzwerke**: Die Verwendung von künstlichen neuronalen Netzen (KNN) ermöglicht die Erfassung von hoch nichtlinearem Verhalten, so wie es beim temperatur- und dehnratenabhängigen Fließverhalten auftritt. Der große Vorteil bei der Verwendung eines KNN ist, dass kein mathematisches Modell erforderlich ist –, damit entfällt die Bestimmung der Materialkonstanten. Die Zuverlässigkeit dieser Modelle ist aber stark an die Qualität und Quantität der experimentellen Daten geknüpft, die zur Erstellung erforderlich sind. Beispiele finden sich in [26–28].

Hensel-Spittel Modell

Nach dem phänomenologischen Fließkurvenmodell von Hensel und Spittel [23] lässt sich die Warmfestigkeit als Produkt der Einflüsse aus Dehnung, Dehnrate, Temperatur und Werkstoff darstellen. Die Beziehung nimmt dabei folgende Form an:

$$\sigma_f = \sigma_{f_0} K_\vartheta K_\varphi K_{\dot{\varphi}}. \tag{2.11}$$

Die Konstante σ_{f_0} ist abhängig vom Werkstoff und dem Gefügezustand, K_ϑ beschreibt die Temperaturabhängigkeit, K_φ die Dehnungsabhängigkeit und $K_{\dot{\varphi}}$ die Dehnratenabhängigkeit. Für die einzelnen Faktoren werden jeweils Potenz- oder Exponentialgesetze gewählt, sodass die endgültige Form folgendermaßen aussieht:

$$\sigma_F = A \cdot e^{m_1\vartheta} \cdot \varphi^{m_2} \cdot e^{m_4/\varphi} \cdot (1+\varphi)^{m_5\vartheta} \cdot e^{m_7\varphi} \cdot \dot{\varphi}^{m_8\vartheta}. \tag{2.12}$$

Die Fließspannung berechnet sich damit aus der Temperatur ϑ, dem Umformgrad φ, der Umformgeschwindigkeit $\dot{\varphi}$ und den Koeffzienten $A, m_1 - m_8$. Durch die eher einfache Form eignet sich dieses Modell sehr gut für die Implementierung in numerischen Simulationen.

2.2 Torsion

In diesem Kapitel werden die fundamentalen Grundlagen der Torsion erörtert. Darauf aufbauend folgt die Erklärung des Torsionsversuches mit Fokus auf dessen Auswertung. Zuletzt wird noch auf die Besonderheiten der Heißtorsion eingegangen.

2.2.1 Grundlagen

Von reiner Torsion ist die Rede, wenn an einem prismatischen Stab an beiden Enden zwei entgegengesetzte Momente angreifen. Diese Momente liegen auf parallelen Flächen zueinander und führen zu einer Verdrehung entlang der Stabachse. Die Voraussetzung prismatischer Stäbe bedeutet, dass die Stabachse gerade ist und die Querschnittsfläche entlang dieser konstant ist. Die Form der Querschnittsfläche kann dabei beliebig sein, beispielsweise axialsymmetrisch, rechteckig oder aber auch ellipsenförmig. Der einfachste Fall tritt ein, wenn es sich um einen axialsymmetrischen Querschnitt handelt (Kreis- oder Kreisringförmig). Dann bleiben die Querschnittsflächen bei der Verdrehung eben und es kommt zu keiner Verwölbung. Die einzelnen Querschnitte rotieren wie starre Scheiben, bei der die Verdrehung zu benachbarten Scheiben über die ganze Stabachse konstant ist.

In Abbildung 2.7 ist ein Hohlzylinder der Länge l zu sehen, an dem die Verformung mithilfe eines Volumenelementes darstellt ist. Das Element wird durch zwei Mantelflächen bei r und $r + dr$ in radialer Richtung begrenzt. In axialer und tangentialer Richtung wird es durch jeweils zwei parallele Flächen begrenzt, wobei eine axiale Fläche die Gerade A–B enthält. Durch das anliegende Torsionsmoment M_T kommt es zu einer Verdrehung der Enden zueinander um den unbekannten Winkel ϕ. Dadurch wird die ursprünglich zur Stabsachse parallele Gerade A–B, zu einer Schraubenlinie mit konstanter Steigung $\gamma(r)$. Diese hängt nur vom Radius r ab.

Das Volumenelement aus Abbildung 2.7 erfährt während der Verformung eine Scherung um den Steigungswinkel $\gamma(r)$. Wenn der Winkel $\gamma(r)$ klein bleibt, dann kann die Länge des Kreisbogens B-C sowohl durch r und ϕ, als auch durch $\gamma(r)$ und l ausgedrückt werden. Die Beziehungen

$$\gamma(r)l = \phi r \tag{2.13}$$

und

$$\gamma(r) = \phi \cdot \frac{r}{l} \tag{2.14}$$

folgen somit für die Scherung des Volumenelements. Folglich kann die Scherdehnrate

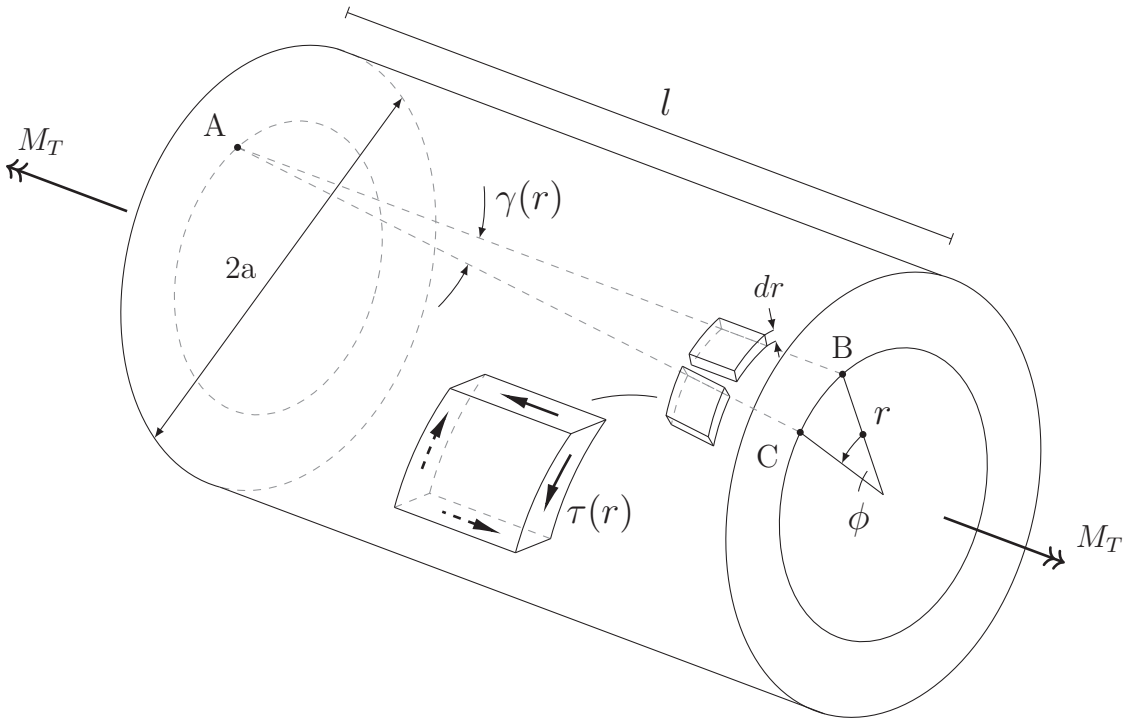


Abbildung 2.7: Darstellung reiner Torsion an einem Hohlzylinder. Dabei zu sehen ist die Verformung der Linie A-B zu A-C und des daran befindlichen Volumenelements. In der Vergrößerung sind alle auftretenden Spannungen eingezeichnet (Angelehnt an [29]).

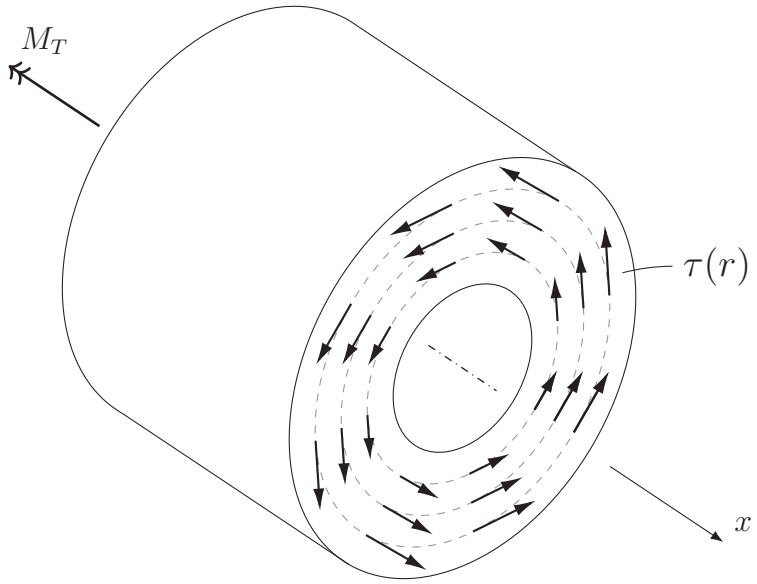


Abbildung 2.8: Schubspannungen am Schnittufer, angelehnt an [29].

als

$$\dot{\gamma}(r) = \dot{\phi} \cdot \frac{r}{l} \tag{2.15}$$

ausgedrückt werden.

Sonst kommt es zu keiner weiteren Verformung des Volumenelementes. Diese Schubspannungen sind damit die einzigen, die am Volumenelement auftreten. Die Richtungen und Flächen, auf denen diese Spannungen wirken, sind in der Vergößerung von Abbildung 2.7 eingezeichnet.

Um einen Zusammenhang zwischen den Schubspannung $\tau(r)$ und dem Torsionsmoment M_T herzustellen, muss eine Gleichgewichtsbedingung formuliert werden. Dazu wird der Stab an beliebiger Stelle x normal zur Stabachse geschnitten, siehe Abbildung 2.8. Die an der Schnittfläche tangential angreifenden Schubspannungen $\tau(r)$ sind dann mit dem Torsionsmoment M_T im Gleichgewicht, wenn das resultierende Moment am Schnittufer um die x-Achse verschwindet. Daraus folgt die Gleichung

$$\int_A \tau(r)\ r \,\mathrm{d}A - M_T = 0. \tag{2.16}$$

Dabei muss über die gesamte Querschnittsfläche A des Stabes integriert werden. Handelt es sich bei dem Querschnitt A um einen Kreis mit Radius a, kann folgende vereinfachte Beziehung gefunden werden:

$$2\pi \int_0^a \tau(r)\ r^2 \,\mathrm{d}r - M_T = 0. \tag{2.17}$$

2.2.2 Torsionsversuch

Beim Torsionsversuch wird eine zylinderförmige Probe (Vergleiche Abbildung 2.7) durch das Torsionsmoment M_T um eine Rotationsachse verdreht. Im Gegensatz zum Zug- oder Zylinderstauchversuch bleibt dabei die Geometrie während des Versuches annähernd konstant. Dadurch ist bei konstanter Verdrehrate auch die Umformgeschwindigkeit während des Versuches konstant. Eine Eigenschaft, die sehr vorteilhaft bei der Simulation umformtechnischer Prozesse ist.

Das gemessene Torsionsmoment M_T ist abhängig von der auftretenden Schubspannungsverteilung $\tau(r)$, während die dazugehörende Scherung $\gamma(r)$ durch die Verdrehung ϕ vorgegeben ist. Die analytische Beschreibung dieser Zusammenhänge ist nur unter gewissen Voraussetzungen möglich:

- Die Verformung entlang der gesamten Länge l muss homogen erfolgen, es darf keine lokale Instabilität auftreten. [4]
- Der Torsionsquerschnitt muss rotationssymmetrisch sein, sodass keine Wölbung der Querschnittsflächen auftritt. [30]
- Der Werkstoff muss isotrop, homogen und inkompressibel sein. Die selben Bedingungen gelten auch für die Anwendung der Fließbedingungen nach Tresca und von Mises.

Sind all diese Voraussetzungen erfüllt, kann unter Verwendung von Gleichung 2.17, die Schubspannung an der Oberfläche ($r = a$) bestimmt werden. Dort gilt der folgende Zusammenhang für die Schubspannung τ nach [31]:

$$\tau(\gamma_a, \dot{\gamma}_a) = \frac{3M_T(\gamma_a, \dot{\gamma}_a)}{2\pi a^3} \left\{ 1 + \frac{1}{3M_T} \left[\gamma_a \frac{\partial M_T}{\partial \gamma_a} + \dot{\gamma}_a \frac{\partial M_T}{\partial \dot{\gamma}_a} \right] \right\}. \tag{2.18}$$

Die Scherdehnung γ_a und Scherdehnrate $\dot{\gamma}_a$ sind dabei nach Gleichung 2.14 und 2.15 an der Oberfläche ($r = a$) zu berechnen.

Da die Auswertung an der Probenoberfläche erfolgt, ist diese Methode anfällig für Abweichungen, die aus Oberflächendefekten resultieren. Dazu zählen Bearbeitungsspuren, Oberflächenoxidation und auch eventuelle Kerbeffekte. Deshalb wurden Methoden entwickelt, bei denen die Spannungen knapp unter der Oberfläche ausgewertet werden. Dazu wird ein effektiver Radius r^* definiert, an dem die Form der Fließkurve nahezu keinen Einfluss auf die Höhe der Schubspannungen hat [6]. Voraussetzung dafür ist, dass die Fließkurve der generellen Beziehung

$$\sigma_f(\varphi, \dot{\varphi}) = \sigma_{f_0} \varphi^n \dot{\varphi}^m \tag{2.19}$$

entspricht. n ist dabei der Verfestigungsexponent und m die Dehnratenabhängigkeit. Lach et al. [6] verglichen verschiedene Definitionen für den effektiven Radius und legten diesen als

$$r^* = \frac{3}{4}a \tag{2.20}$$

fest. Unter Berücksichtigung von Gleichung 2.14 ist die Scherung durch

$$\gamma^* = \frac{3}{4}\gamma_a \tag{2.21}$$

gegeben. Schlussendlich kann noch die Schubspannung am effektiven Radius als

$$\tau^* = \frac{3M_T}{2\pi a^3} \tag{2.22}$$

bestimmt werden.

2.2.3 Heißtorsionsversuch

Der Heißtorsionsversuch basiert auf dem im vorherigen Absatz beschriebenen Torsionsversuch. Das Ziel ist die Simulation der Warmumformung in Hinblick auf die Fließkurve. Da der Versuch bei erhöhten Temperaturen stattfindet, spielt die Temperaturverteilung innerhalb der Probe eine wichtige Rolle. Für eine Auswertung wie in Absatz 2.2.2 beschrieben, muss das Werkstoffverhalten isotrop sein. Das setzt ein komplett homogenes Temperaturfeld voraus. Dieses ist praktisch schwer umsetzbar, sodass in der Realität Temperaturgradienten innerhalb der Torsionsprobe auftreten [32–36]. Diese können sich sowohl in axialer, als auch in radialer Richtung ausbilden. Dabei gibt es mehrere Faktoren für die Entstehung:

- **Resistive Erwärmung**: Die Erwärmung durch Joulesche Verlustheizung ist stark an die Querschnittsfläche der Torsionsprobe gebunden. Wenn der Querschnitt der Probe nicht konstant ist, variiert die Stromdichte und damit die resistive Erwärmung. Daraus kann ein axialer Temperaturgradient entstehen.
- **Wärmestrahlung**: An der Oberfläche kommt es zu einem Wärmefluss durch Wärmestrahlung, dieser ist im Inneren der Torsionsprobe nicht vorhanden. Dadurch kann sich ein radialer Temperaturgradient bilden.
- **Probeneinspannung**: An der Probeneinspannung tritt ein Wärmefluss in diese auf. Das kann einen axialen Temperaturgradienten forcieren.
- **Umformenergie**: Die bei der Verfomung in die Torsionsprobe eingebrachte Energie wird durch innere Reibung als Wärme dissipiert. Mit zunehmender Entfernung von der Torsionsachse steigt die Verformung und führt somit zu

einem ausgeprägterem Temperaturanstieg. Das kann zu einem radialen Temperaturgradienten führen. Dieser Effekt spielt besonders bei hohen Umformgeschwindigkeiten eine wichtige Rolle, da dort fast kein Temperaturausgleich durch Wärmeleitung erfolgt.

Tritt ein solcher Temperaturgradient während des Heißtorsionsversuches auf, kann das in weiterer Folge schnell zu einer lokalisierten Verformung führen. Damit kann die Auswertung des Versuches nicht mehr nach den Beziehungen aus 2.2.2 durchgeführt werden. Die Auswertung über eine inverse Analyse (Für eine allgemeine Erklärung siehe [37]) wird notwendig. Dazu wird ein numerisches Modell (Finite Elemente Methode) formuliert, das den Heißtorsionsversuch in seiner Gesamtheit beschreibt. Die Auswertung besteht dann in der Parametrisierung des in der Simulation verwendeten Materialmodells. Wenn dieses erfolgreich kalibriert wurde und dadurch die experimentellen Versuchsergebnisse abbilden kann, beschreibt das Materialmodell die gesuchte Fließkurve. Diese Auswertung wurde in der Literatur schon öfters beschrieben und hat zu guten Ergebnissen geführt [7–11].

Neben dem Temperaturgradienten müssen auch große Verformungen berücksichtigt werden. So kommt es bei der Verformung zu geometrischen Nichtlinearitäten, die aus der großen Verdrehung resultieren. Diese werden in der klassischen Auswertung nach Absatz 2.2.2 nicht berücksichtigt. Der Weg über den inversen Ansatz kann diese aber im numerischen Modell sehr wohl berücksichtigen.

2.3 Finite-Elemente Methode

Die Finite-Elemente Methode stellt ein numerisches Lösungsverfahren dar, das ursprünglich zur Behandlung von Problemen der Strukturmechanik entwickelt wurde. Durch diverse Erweiterungen eignet sich die Formulierung aber auch sehr gut für Berechnungen von Festkörpern, thermischen Wärmeleitungsproblemen und elektrischen Feldproblemen. Dazu muss ein mathematisches Modell gelöst werden, das dem auftretenden physikalischen Problem entspricht. Die Finite-Elemente Methode erlaubt dabei die Lösung von sehr komplexen mathematischen Problemen.

Eine typische Anwendung ist die Berechnung des Verhaltens einer Struktur oder Strukturkomponente unter beliebiger Belastung. Das physikalische Problem wird dazu unter gewissen Annahmen in einem mathematischen Modell idealisiert. Meistens führt das auf ein System von Differentialgleichungen. Die Finite-Elemente-Methode überführt dieses Differentialgleichungssystem, durch den Einsatz von lokalen Ansatzfunktionen, in ein numerisch lösbares Gleichungssystem. Die Lösung obliegt

dann der Finite-Elemente Berechnung. Da es sich bei dieser um ein numerisches Lösungsverfahren handelt, ist die Rechnung von den gewählten Lösungsparametern abhängig. Wird die geforderte Lösungsgenauigkeit nicht erreicht, dann müssen die Lösungsparameter (beispielsweise die Vernetzungsdichte) solange verändert werden, bis ein akzeptables Ergebnis vorliegt.

Dabei ist besonders wichtig, dass die Finite-Elemente Lösung nur die Bestimmung des mathematischen Modells darstellt. Daher können keine zusätzlichen Informationen über das physikalische Phänomen gewonnen werden, welche nicht schon in der Formulierung des mathematischen Modells berücksichtigt wurden. Allerdings kann durch die Finite-Elemente Methode ein tieferer Einblick in das physikalische Problem gewonnen werden.

Bei der konkreten Anwendung muss ein mathematisches Modell gewählt werden, das alle gesuchten Größen des physikalischen Problems zuverlässig und wirksam beschreiben kann. Das gewählte Modell soll die geforderte Antwort mit vorgegebener Genauigkeit (Zuverlässigkeit) und mit dem geringsten Aufwand (Wirksamkeit) erfüllen. [38]

Durch ihre Vielfältigkeit hat sich die Finite-Elemente Methode mittlerweile als Standardverfahren für ingenieurtechnische Fragestellungen etabliert. Auch nur die ansatzweise Beschreibung der einzelnen konkreten Finite-Elemente Methoden würde den Rahmen dieser Arbeit weit übersteigen. Weiterführende Beschreibungen sind zum Beispiel in [38–40] zu finden.

2.4 Optimierungalgorithmus Nelder-Mead

Die inverse Analyse des Heißtorsionsversuches stellt in seiner mathematischen Behandlung ein Optimierungsproblem dar. Generell ist bei Optimierungsproblemen das Ziel die Lösung von

$$\min f(\mathbf{x}), \tag{2.23}$$

wobei $f : \mathbb{R}^n \to \mathbb{R}$ die zu minimierende Funktion, n die Dimension und $\mathbf{x}$ das n-dimensionale Funktionsargument ist. Dazu wurden einige Optimierungsalgorithmen entwickelt, ein sehr häufig verwendetes Verfahren bei der unbeschränkten Suche ist das Nelder-Mead-Verfahren [41].

Der Nelder-Mead-Algorithmus ist geeignet zur unbeschränkten Suche von Lösungen für ein auftretendes Minimierungsproblem. Die Ableitungen $\partial f/\partial \mathbf{x}_i$ müssen dabei nicht bekannt sein. Es handelt sich damit um ein direktes Suchverfahren. Der Algorithmus basiert auf der Formulierung eines n-Dimensionalen Simplex, genannt

Δ. Begrenzt wird dieses durch $n+1$ Eckpunkte, $\mathbf{p}_1, \mathbf{p}_2, ... , \mathbf{p}_{n+1}$. Im Laufe der Minimierungsprozedur werden iterativ neue Simplexe generiert, sodass das Optimum von Problem 2.23 gefunden wird. Während jedes Optimierungsschritts werden die Eckpunkte $\{\mathbf{p}_i\}_{i=1}^{n+1}$ nach deren Funktionswerte $f(\mathbf{p}_i)$ geordnet. Dabei muss gelten

$$f(\mathbf{p}_1) \leq f(\mathbf{p}_2) \leq ... \leq f(\mathbf{p}_{n+1}). \tag{2.24}$$

Bestimmt durch die Funktionswerte der Eckpunkte kann in einem Iterationsschritt eine von vier möglichen Operationen durchgeführt werden. Diese sind Spiegelung, Erweiterung, Verkleinerung und Schrumpfen. Quantifiziert werden diese jeweils durch einen skalaren Wert: k_1 (Spiegelung) ; k_2 (Erweiterung) ; k_3 (Verkleinerung) ; k_4 (Schrumpfen). Für die Standard Implementierung der Nelder-Mead Methode werden die Werte als

$$\{k_1;\ k_2;\ k_3;\ k_4\} = \{1;\ 2;\ 0,5;\ 0,5\} \tag{2.25}$$

festgelegt [41, 42]. Zuletzt wird noch der Mittelpunkt $\bar{\mathbf{p}}$ der n besten Eckpunkte als

$$\bar{\mathbf{p}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{p}_i \tag{2.26}$$

definiert. [43]

Schematische Beschreibung eines Iterationsschrittes nach Lagarias et al. [42].

1. **Sortieren.** Auswerten der Funktion f an allen $n+1$ Eckpunkten von Δ und Reihung entsprechend Bedingung 2.24.

2. **Spiegelung.** Berechnen des Spiegelpunktes $\mathbf{p}_r$ als

   $$\mathbf{p}_r = \bar{\mathbf{p}} + k_1(\bar{\mathbf{p}} - \mathbf{p}_{n+1}). \tag{2.27}$$

   Berechnen von $f_r = f(\mathbf{p}_r)$. Wenn $f_1 \leq f_r < f_n$, dann $\mathbf{p}_{n+1}$ durch $\mathbf{p}_r$ ersetzen.

3. **Erweiterung.** Wenn $f_r < f_1$, dann Erweiterungpunkt $\mathbf{p}_e$ als

   $$\mathbf{p}_e = \bar{\mathbf{p}} + k_2(\mathbf{p}_r - \bar{\mathbf{p}}) \tag{2.28}$$

   berechnen und Funktionswert $f_e = f(\mathbf{p}_e)$ bestimmen. Wenn $f_e < f_r$ dann $\mathbf{p}_{n+1}$ durch $\mathbf{p}_e$ ersetzen, sonst $\mathbf{p}_{n+1}$ durch $\mathbf{p}_r$ ersetzen.

4. **Äußere Verkleinerung.** Wenn $f_n \leq f_r < f_{n+1}$, dann äußeren Verkleinerungspunkt $\mathbf{p}_{oc}$ als

$$\mathbf{p}_{oc} = \bar{\mathbf{p}} + k_3(\mathbf{p}_r - \bar{\mathbf{p}}) \tag{2.29}$$

berechnen und Funktionswert $f_{oc} = f(\mathbf{p}_{oc})$ bestimmen. Wenn $f_{oc} \leq f_r$, dann $\mathbf{p}_{n+1}$ durch $\mathbf{p}_{oc}$ ersetzen, sonst zu Schritt 6 springen.

5. **Innere Verkleinerung.** Wenn $f_r \geq f_{n+1}$, dann inneren Verkleinerungspunkt $\mathbf{p}_{ic}$ als

$$\mathbf{p}_{ic} = \bar{\mathbf{p}} - k_3(\mathbf{p}_r - \bar{\mathbf{p}}) \tag{2.30}$$

berechnen und Funktionswert $f_{ic} = f(\mathbf{p}_{ic})$ bestimmen. Wenn $f_{ic} \leq f_{n+1}$, dann $\mathbf{p}_{n+1}$ durch $\mathbf{p}_{ic}$ ersetzen, sonst zu Schritt 6 springen.

6. **Schrumpfung.** Für $2 \leq i \leq n+1$, definiere

$$\mathbf{p}_i = \mathbf{p}_1 + k_4(\mathbf{p}_i - \mathbf{p}_1) \tag{2.31}$$

Diese Prozedur wird wiederholt, bis das gesuchte Minimum der Funktion f gefunden ist. Der Punkt $\mathbf{p}_1$ stellt dann die Lösung des Problems dar.

Das Nelder-Mead Verfahren konvergiert nicht zwangsweise im Optimum der Zielfunktion. Auch wenn diese Funktion strikt konvex ist, kann es vorkommen, dass die Methode nicht zu einem optimalen Punkt konvergiert. Hier gibt es mittlerweile einige Ansätze, um die Konvergenz zu verbessern, siehe beispielsweise [44–46].

Trotz der limitierten Konvergenz bleibt die Methode ein solides Verfahren, um Optimierungsprobleme in wenigen Dimensionen mittels direktem Ansatz zu lösen. Bei Problemstellungen in höheren Dimensionen ($n > 20$) kann der Algorithmus aber an seine Grenzen stoßen und sehr ineffizient werden. Hierfür wurden schon veränderte Ansätze entwickelt, um diese Limitationen zu umgehen [43].

2.5 PID-Regelung

Kontrollsysteme werden häufig benötigt, um Prozesse zu regeln oder zu steuern. Sind die genauen Eigenschaften zur Steuerung des Prozesses nicht bekannt, bildet der PID-Algorithmus eine Möglichkeit den Prozess trotzdem zu regeln. Durch die einfache Struktur, die simple Implementierung und die Möglichkeit automatischer Parametrisierung wird er sehr oft eingesetzt.

Abbildung 2.9 zeigt einen schematischen Regelkreis, wie er generell verwendet wird. Die Regelabweichung $e(t)$ (Zielwert $r(t)$ - Istwert $y(t)$) führt zu einer Steuer-

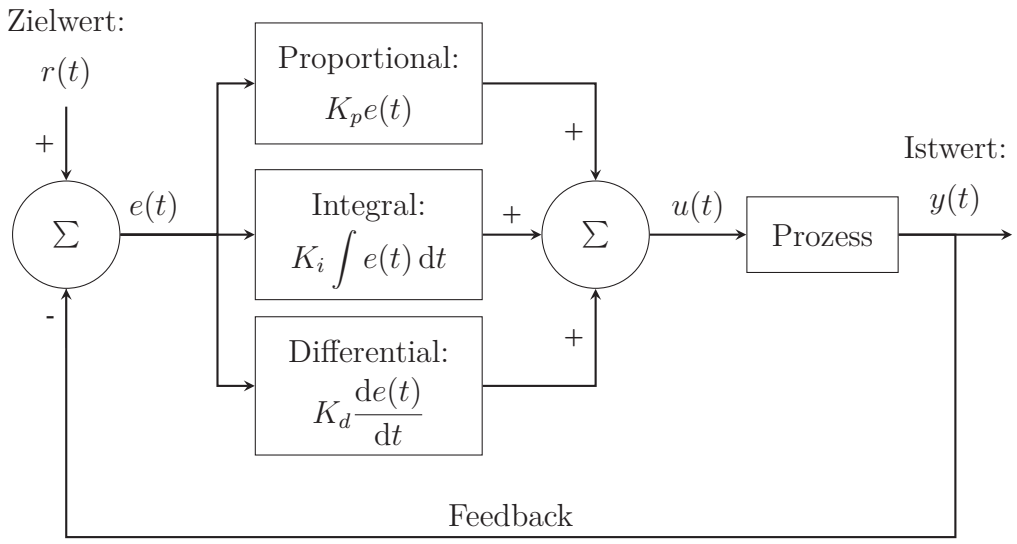


Abbildung 2.9: Schematischer PID-Regelkreis, angelehnt an [47].

größe $u(t)$, welche dann im Prozess/System im Istwert(Ausgangsgröße) $y(t)$ resultiert. Der PID-Algorithmus berechnet dabei die Steuergröße $u(t)$ durch Summation von drei Termen (Proportional, Integral und Differential). Der proportionale Term reagiert proportional zur aktuellen Regelabweichung. Der integrale Term reagiert rückblickend und betrachtet den zeitlichen Verlauf der Regelabweichung. Der differentiale Term reagiert vorausschauend und regelt abhängig von der Änderung der Regelabweichung. In der parallelen Form werden alle drei Terme summiert und ergeben folgendes Kontrollgesetz:

$$u(t) = K_p \left(e(t) + \frac{1}{t_i} \int e(t)\,\mathrm{d}t + t_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right). \tag{2.32}$$

K_p entspricht der proportionalen Verstärkung. Die integrale Zeitkonstante t_i bildet mit $K_i = \frac{K_p}{t_i}$ die integrale Verstärkung K_i und die differentielle Verstärkung K_d ergibt sich mit der differentiellen Zeitkonstante t_d nach $K_d = K_p t_d$. [47]

Um den PID-Regelkreis an einen konkreten Prozess anzupassen, müssen die drei Parameter (proportionale Verstärkung, integrale Zeitkonstante und differentiale Zeitkonstante) bestimmt werden. Dazu wurden in der Literatur etliche Methoden entwickelt. Zu den beliebtesten zählen Trial und Error, die Ziegler-Nichols Step Response Methode und die Ziegler-Nichols Frequency Response Methode [48].

3 Vorbereitung & Versuchsdurchführung

Alle Versuche[1] wurden auf einer Gleeble 3800-GTC mit Hot Torsion Einheit durchgeführt (siehe Abbildung 3.1). Die Torsionseinheit erlaubt den Einbau von zylindrischen Proben, welche an beiden Enden mittels wassergekühlter Klemmen fixiert werden. Eine Seite ist dabei an eine hydraulische Torsionseinheit gekoppelt und erlaubt dadurch Verdrehungen mit einem maximalen Torsionsmoment von 100 N m.

Durch die Einspannung ist es möglich die Torsionsprobe resistiv zu beheizen. Die Temperatursteuerung erfolgt dabei über ein Thermoelement, das an der Probenoberfläche angeschweißt ist. Somit besteht in der Temperatursteuerung der Probe jegliche Flexibilität und es können verschiedenste Wärmebehandlungen simuliert werden. Auch ein Abschrecken mittels Gas oder Druckluft ist möglich.

Um Oxidation an der Probenoberfläche zu vermeiden, befindet sich der gesamte Versuchsaufbau in einer Vakuumkammer. Somit ist es möglich, die Versuche unter Schutzgas oder im Vakuum durchzuführen.

Die Kombination aus Torsion und Temperatur ermöglicht die Simulation verschiedenster Umformbedingungen.

3.1 Versuchsmaterial

3.1.1 Material

Die Torsionsproben für die Versuche wurden aus A220 gefertigt. Der Stahl A220 ist ein nichtrostender, stabilaustenitischer Cr-Ni-Mo-Stahl mit niedrigem Kohlenstoffgehalt für hohe Anforderungen an die Gefügehomogenität. Die nominelle Zusammensetzung ist in Tabelle 3.1 angegeben.

Dieser Werkstoff zeichnet sich durch sein einphasiges Gefüge aus. Durch den sehr niedrigen Kohlenstoffgehalt ist der Werkstoff frei von Karbiden, welche Einfluss auf das Umformverhalten hätten. Die Bildung von verformungsinduzierten Karbidaus-

[1] Die Durchführung der Versuche erfolgte extern bei der voestalpine Forschungsservicegesellschaft Donawitz GmbH.

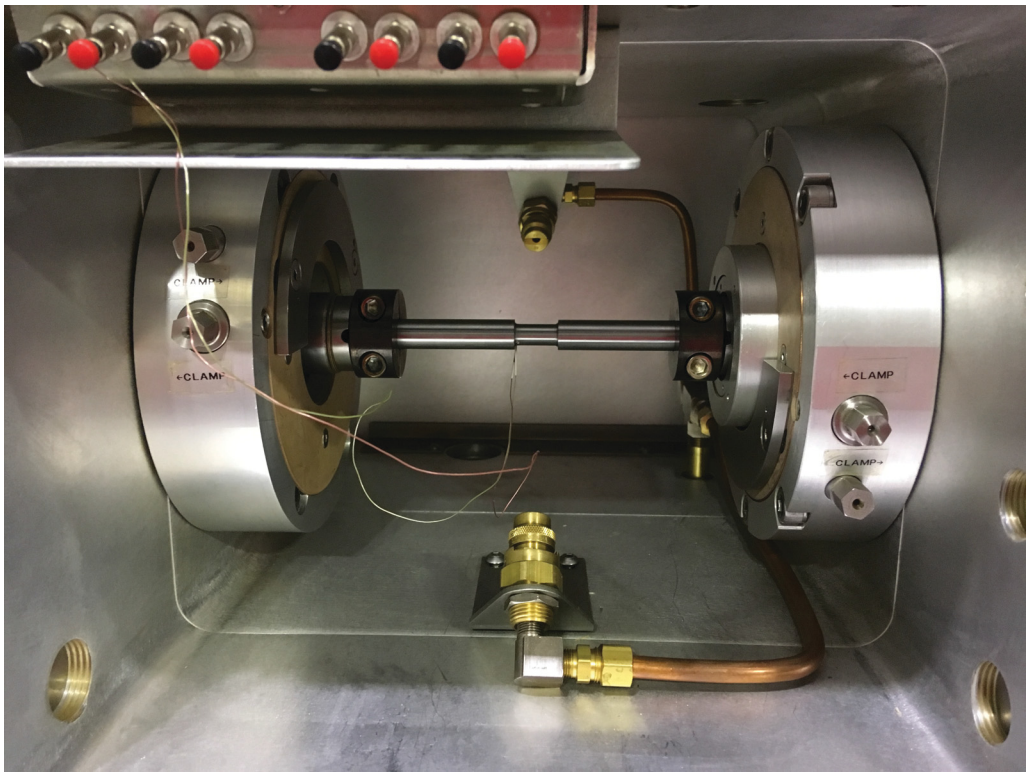


Abbildung 3.1: Versuchaufbau Heißtorsion Gleeble 3800-GTC. Eingebaute Torsionsprobe SMT001 mit angeschweißtem Thermoelement.

Tabelle 3.1: Chemische Zusammensetzung A220 (Werte in wt%).

C	Si	Mn	Cr	Mo	Ni	N
<0,03	0,30	1,70	17,50	2,70	14,50	0,07

scheidungen muss somit auch nicht berücksichtigt werden.

3.1.2 Geometrie

Als Probengeometrie wurde die von Dynamic Systems inc. (D.S.I.) empfohlene Torsionsprobe SMT001 gewählt (Fertigungszeichnung siehe Anhang A). Deren Prüflänge besteht aus einem zylinderförmigen Querschnitt mit 10 mm Durchmesser und einer Länge von 20 mm. Daran anschließend geht der Querschnitt über in ein Rohr mit 14 mm Außendurchmesser und einem Innendurchmesser von 8,33 mm. Durch die spezielle Geometrie beschränkt sich die Verformung auf die Prüflänge und der Temperaturgradient zwischen Probenmitte und dem Randbereich ist minimiert.

3.2 Versuchsmethoden

In diesem Abschnitt werden die verwendeten Versuchsmethoden näher erläutert. Dazu zählen die Messung des axialen Temperaturgradienten, der Heißtorsionsversuch selbst und die Messung der lokalen Verdrehung nach dem Torsionsversuch.

3.2.1 Ermittlung des axialen Temperaturgradienten

Das Ziel dieses Versuchs war die Bestimmung des axialen Temperaturgradienten innerhalb der Probe. Dieser hat einen großen Einfluss auf die Homogenität der Verformung. Deshalb ist es wichtig, die Temperaturverteilung möglichst genau zu kennen. Auch der radiale Temperaturgradient spielt ein Rolle, jedoch ist dieser messtechnisch schwer zugänglich. Zhang et al. [49] versuchte diesen durch dünne Bohrungen mit Thermoelementen zu bestimmen. Jene Vorgangsweise setzt aber voraus, dass der Temperaturgradient nicht durch die Bohrlöcher beeinflusst wird. Gerade bei resistiver Probenbeheizung ist diese Annahme aber sehr ungewiss.

Zur Bestimmung des Temperaturgradienten wurden vier Thermoelemente an der Probenoberfläche platziert. So konnte die Temperatur an verschiedenen axialen Positionen während des kompletten Versuches detektiert werden. Die genaue Positionierung der Thermoelemente findet sich in Abbildung 3.2. Da hauptsächlich der Temperaturverlauf entlang der Prüflänge von Interesse war, wurden dort drei Thermoelemente platziert. Das vierte wurde in größerer Entfernung platziert, um auch Information über den Wärmefluss zur Einspannung zu erhalten. Die Messungen wurden bei vier verschiedenen Temperaturen T_1 durchgeführt: 900 °C, 1000 °C, 1100 °C und 1200 °C. Das genaue Temperaturprogramm ist in Abbildung 3.3 zu finden. Die

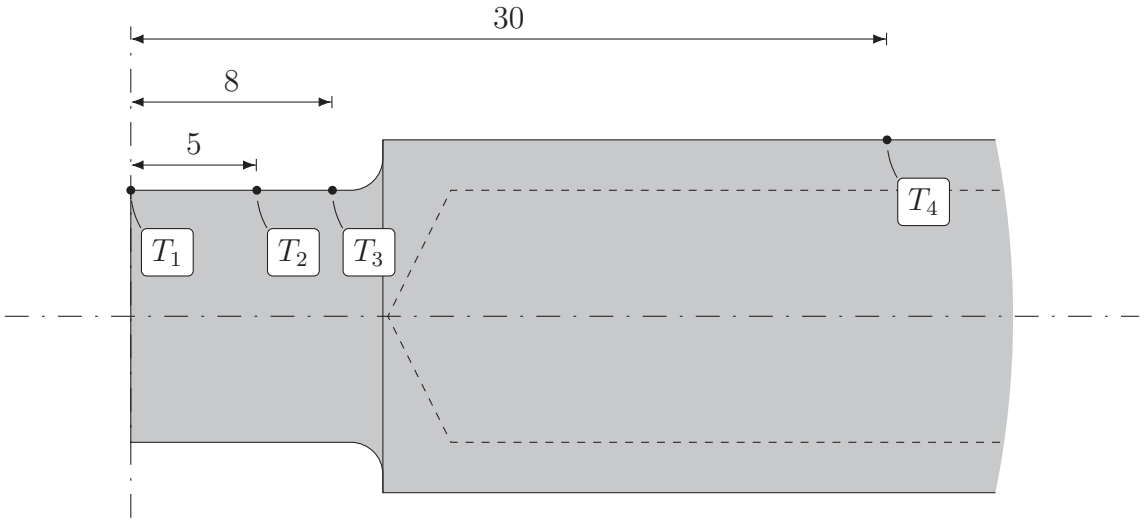


Abbildung 3.2: Positionen der Temperaturmessung mit den dazugehörigen Temperaturen T_1 bis T_4. Angaben in mm.

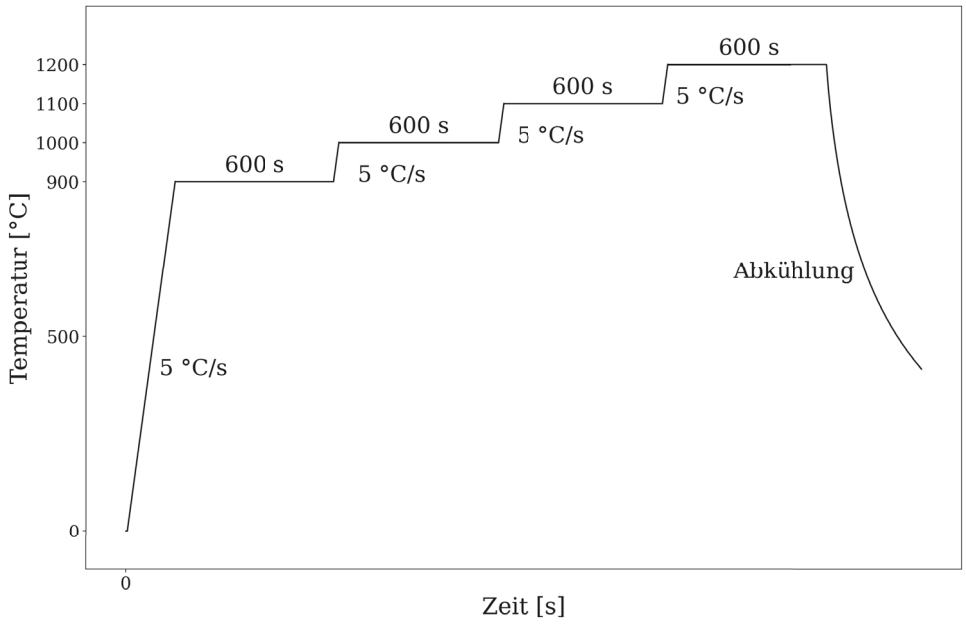


Abbildung 3.3: Vorgegebenes Temperaturprofil an T_1, zur Messung des axialen Temperaturgradienten.

Tabelle 3.2: Versuchsplan der Torsionsversuche

Parametersatz	Umform-temperatur ϑ [°C]	Haltetemperatur ϑ [°C]	Vergleichs-dehnrate $\dot{\varepsilon}_v[s^{-1}]$
1	800	1000	1
2	900	1000	1
3	1000	1000	1
4	1100	1100	1
5	1200	1200	1
6	800	1000	10
7	900	1000	10
8	1000	1000	10
9	1100	1100	10
10	1200	1200	10

Torsionsprobe startet bei Raumtemperatur und wird mit $5\,°C\,s^{-1}$ auf 900 °C aufgeheizt. Dort wird sie dann für 600 s gehalten. Die Haltezeit dient zur Stabilisierung des auftretenden Temperaturgradienten. Danach wird die Temperatur um 100 °C mit ebenfalls $5\,°C\,s^{-1}$ auf 1000 °C erhöht und diese Temperatur erneut für 600 s gehalten. Das erfolgt anschließend auch noch für 1100 °C und 1200 °C. Nachdem die Torsionsprobe 600 s bei 1200 °C gehalten wurde folgt die freie Abkühlung. Damit ist der Versuch abgeschlossen.

Um den Einfluss der Probenoberfläche auf den axialen Temperaturgradienten zu erfassen, wurde diese in zwei verschiedenen Zuständen gemessen. Erstens mit aus der Probenbearbeitung resultierender Oberfläche, und zweitens mit oxidierter Oberfläche. Zur Oxidation wurde die Torsionsprobe für 600 s bei 1200 °C an Luft geglüht.

3.2.2 Heißtorsionsversuch

Der folgende Abschnitt schildert das Versuchsprogramm, anhand dessen die Torsionsversuche in der Gleeble unter Vakuum durchgeführt wurden. Um den Einfluss von Temperatur und Dehnrate zu erfassen, wurde eine Versuchsmatrix erstellt. Diese befindet sich in Tabelle 3.2. Daraus ergaben sich in Summe zehn verschiedene Parametersätze, welche gemessen wurden.

Der erste Schritt einer konkreten Messung war die Wärmebehandlung der Torsionsprobe. Diese war abhängig von der Temperatur, bei der die Verformung stattfinden sollte. Die Wärmebehandlung bestand aus Aufheizen auf die Haltetemperatur (siehe Tabelle 3.2) mit $5\,°C\,s^{-1}$ und anschließender Haltezeit von 300 s. Die Proben bei einer Umformtemperatur von 1000 °C, 1100 °C und 1200 °C wurden direkt

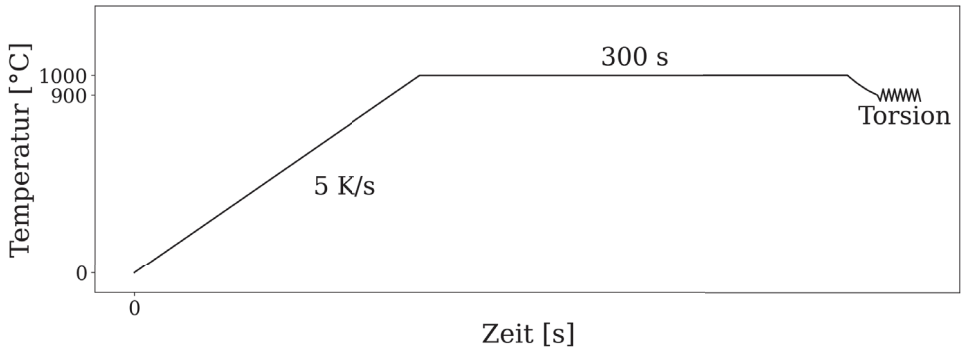


Abbildung 3.4: Wärmebehandlung der Torsionsprobe vor der Verformung. Umformtemperatur = 900 °C.

danach verformt. Die Proben bei der Umformtemperatur von 800 °C und 900 °C, mussten davor noch auf diese abgekühlt werden. Erst dann wurde die Verformung gestartet. Für die Umformtemperatur von 900 °C ist das komplette Temperaturprogramm in Abbildung 3.4 dargestellt. Die Verformung selbst betrug zwei Umdrehungen. Dabei wurde die Rotationsgeschwindigkeit so gewählt, dass sich die gewünschte Vergleichsdehnrate[2] aus der Versuchsmatrix ergab (siehe Tabelle 3.2). Für die Vergleichsdehnrate von $1\,s^{-1}$ ergab das eine Drehzahl von $44{,}7\,min^{-1}$, respektive $447\,min^{-1}$ für die Vergleichsdehnrate $10\,s^{-1}$.

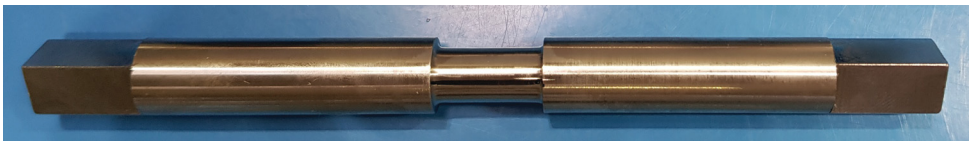
Während der Verformung wurde die Temperaturregelung deaktiviert, sodass die Proben nicht mehr beheizt wurden. Dadurch war das Messsignal weniger von Störungen der resistiven Erwärmung betroffen.

In dem Versuch wird das Drehmoment $\hat{M}_T$ mit dem dazugehörigen Drehwinkel ϕ bestimmt. Zusätzlich wird noch die Temperatur T_1 mitgemessen. Die Auswertung des Versuches ist in Abschnitt 2.2.3 beschrieben.

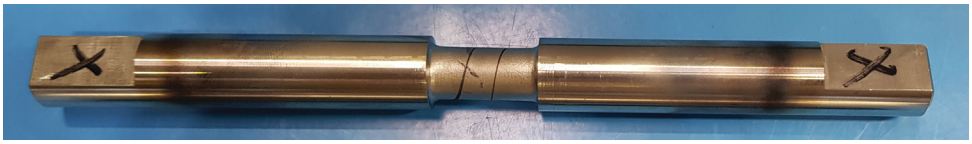
3.2.3 Lokale Verdrehung

Im vorherigen Abschnitt wurde der Torsionsversuch beschrieben. Dabei wird im Versuch direkt das Drehmoment $\hat{M}_T$ und der Verdrehwinkel ϕ gemessen. Unter der Voraussetzung von homogener Verformung kann dann aus dem Drehwinkel die lokale Verformung bestimmt werden (siehe Abschnitt 2.2.2). Die Bedingung dafür ist, dass keine lokale Instabilität – Verformungslokalisation – auftritt. Ist dies nicht erfüllt, dann bildet die Verdrehung ϕ nur die Summe der lokalen Verdrehungen. Die exakte Zuordnung geht verloren. Im Gegensatz zum Zugversuch, bei dem eine lokale

[2]Die Vergleichsdehnrate wurde über die Huber-Mises Vergleichsdehnung nach $\dot{\varepsilon}_v = \dot{\gamma}/\sqrt{3}$ am Vergleichsradius $r_V = 0{,}724\ r_{probe}$ definiert.



(a) Vor dem Torsionsversuch.



(b) Nach dem Torsionsversuch.

Abbildung 3.5: Oberflächenmarkierung zur Auswertung der lokalen Verdrehung.

Instabilität als Einschnürung auftritt, lässt sich im Torsionsversuch keine geometrische Änderung feststellen. Es wird also eine Methode benötigt, mit der die lokale Verdrehung am Ende des Versuches bestimmt werden kann. Im folgenden Absatz wird eine solche beschrieben.

Die lokale Verdrehung kann durch eine Markierung an der Probenoberfläche sichtbar gemacht werden. Dazu wird mittels Permanentmarker (die Pigmente im Stift sind auch nach Temperaturen von über 1200 °C noch erkennbar) eine Linie auf der Probenoberfläche gezeichnet (siehe Abbildung 3.5a). Diese muss parallel zur Torsionsachse sein. Bei der Verformung wird aus der ursprünglich geraden Linie eine Schraubenlinie (siehe Abbildung 3.5b). Da die Verformung nicht homogen ist, variiert die Steigung dieser entlang der Prüflänge. Die lokale Verformung ist dabei immer proportional zur Steigung der Linie. Die Auswertung erfolgt dann durch Vermessung dieser Linie.

Der dazu verwendete Prüfaufbau ist in Abbildung 3.6 dargestellt. Die Torsionsprobe wird innerhalb der Dorne eingespannt. Der Winkel Θ, sowie die axiale Position x können genau vorgegeben werden. Mittels des in Grün markierten Fühlers wird die axiale Startposition an der Probenschulter angefahren. Der Rotationswinkel Θ wird dann solange verändert bis die Probenmarkierung direkt unter dem Fühler liegt. Von diesem Startpunkt ausgehend wird der axiale Abstand verändert und die dazugehörige Rotation Θ gemessen. So kann die gesamte Markierung vermessen werden und der Verlauf der Verdrehung über die lokale Position bestimmt werden.

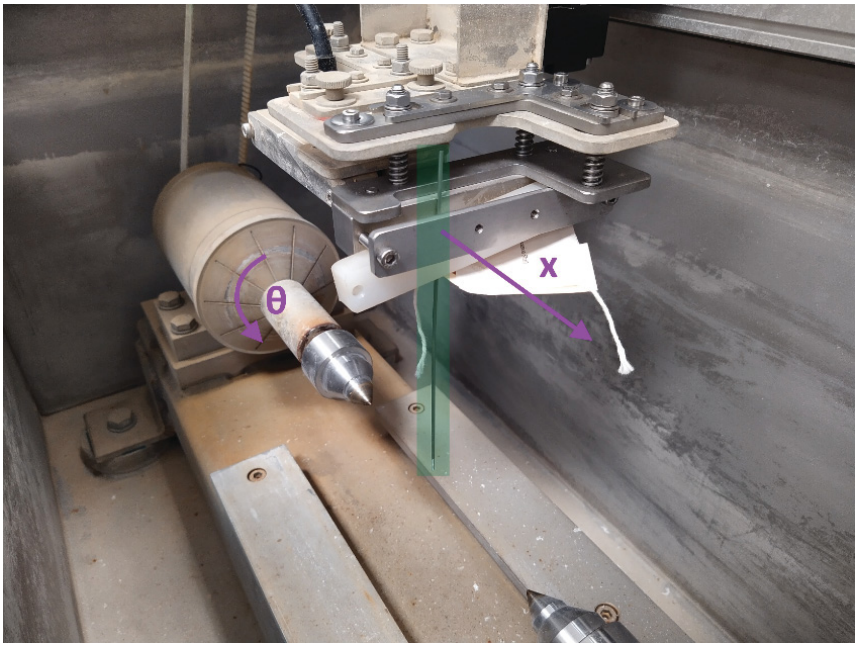


Abbildung 3.6: Versuchsaufbau zur Messung der lokalen Verdrehung.

4 Numerische Simulation

4.1 FE-Modelle

Im Rahmen dieser Arbeit wurden 2 FE-Modelle entwickelt. Diese wurden im kommerziellen Finite-Elemente-Programm Abaqus 2019 der Firma Dassault Systems® implementiert. Das Ziel war die Simulation des Heißtorsionsversuchs, wie er in Abschnitt 3.2.2 durchgeführt wurde.

Die Aufgabe des ersten FE-Modells war die Abbildung der kompletten Temperaturverteilung innerhalb der Probe vor der Umformung. Dazu waren zwei prinzipielle Ansätze möglich.

Der erste ist eine rein thermische Betrachtung. Die Temperaturverteilung wird durch experimentelle Messungen an der Probenoberfläche bestimmt und dient als Randbedingung für die Simulation. Ausgehend von diesen Temperaturen kann die Temperaturverteilung dann stationär berechnet werden. Dieser Ansatz wurde öfters in der Literatur verwendet [50,51], hat aber den Nachteil, dass immer experimentelle Informationen über die Temperaturverteilung benötigt werden.

Der zweite Ansatz berücksichtigt zusätzlich zum thermischen Problem auch das auftretende elektrische Problem. Im Gleeble Heißtorsionsversuch wird die Probe durch Joulsche Verlustenergie erwärmt, das kann durch einen thermisch-elektrisch gekoppelten Ansatz modelliert werden. Die Temperaturverteilung wird dann nicht durch thermische Randbedingungen vorgegeben, sondern resultiert aus der elektrischen Verlustleistung innerhalb der Probe. Durch diesen Ansatz ist das FE-Modell in der Lage das komplette Temperaturfeld zu lösen, ohne auf experimentelle Temperaturverläufe zurückgreifen zu müssen. Das hat den Vorteil, dass auch transiente Probleme simuliert werden können. Dieser Ansatz ist in der Literatur schon erfolgreich implementiert worden [11, 52–54] und wurde deshalb auch in dieser Arbeit gewählt.

Die Aufgabe des zweiten FE-Modells war die Simulation der Umformung. Dies ermöglicht die ortsaufgelöste Auswertung von Spannungen, Dehnungen und Temperaturen. Dazu wurde ein thermisch-mechanisch gekoppelter Ansatz gewählt, wie er auch schon in der Literatur verwendet wurde [10, 55, 56].

Da die programmtechnische Umsetzung der Modelle modular nach den jeweiligen physikalischen Interaktionen erfolgte, ist die folgende Beschreibung genauso gegliedert.

4.1.1 Grundmodell

Im Grundmodell wurde die Geometrie der Torsionsprobe modelliert. Die Einspannung, in der sich die Probe befindet, wurde ebenfalls nachgebaut. Die genauen Abmessungen der Einspannung konnten nicht bestimmt werden, da diese im Inneren des Prüfaufbaus lagen. Somit wurden für diese Geometrie, vereinfachende Annahmen getroffen.

Die Geometrie der Torsionsprobe ist grundsätzlich axialsymmetrisch, nur im Bereich der Einspannung befinden sich nicht rotationssymmetrische Merkmale. Ebenso ist die Einspannung selbst nicht rotationssymmetrisch. Da diese Bereiche aber in einiger Entfernung von der Prüflänge liegen, wurde das FE-Modell trotzdem als 2D-axialsymmetrisch mit Verdrehung aufgebaut. Zusätzlich stellt die Fläche direkt in der Mitte der Probe eine Spiegelsymmetrieebene dar. Auch diese wurde berücksichtigt. Dies führt zu einer enormen Reduktion der Rechenzeit. Der genaue Modellaufbau ist in Abbildung 4.1 dargestellt. Die Vernetzung im Bereich der Prüflänge wurde feiner gewählt (siehe Abbildung 4.2). In Summe ergaben sich so bei der gewählten Vernetzung 10134 Elemente auf der Torsionsprobe und 249 im Bereich der Einspannung.

Abbildung 4.1: Gemeshtes Grundmodell der Torsionsprobe mit Einspannung.

4.1.2 Thermisches Modell

Im thermischen Modell wurden alle für die Probe relevanten physikalischen Effekte berücksichtigt. Das umfasst die Wärmeleitung, sowohl in der Probe als auch in der Einspannung, sowie Wärmestrahlung und Wärmekonvektion. Die genaue Darstellung aller relevanten Flächen ist in Abbildung 4.3 zu sehen. Die Wärmeleitung in

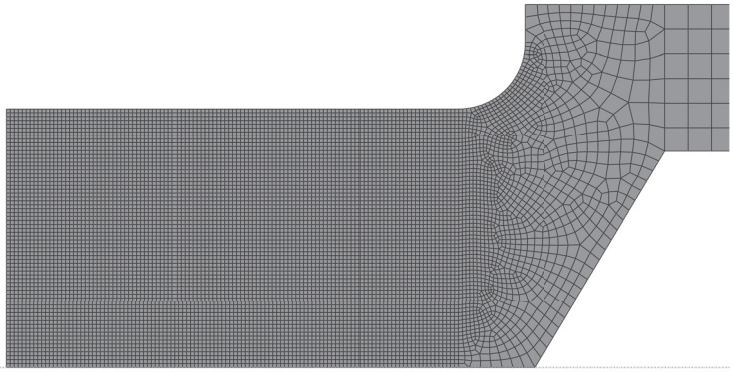


Abbildung 4.2: Vergrößerung der Torsionsprobe mit Darstellung des verwendeten Meshs.

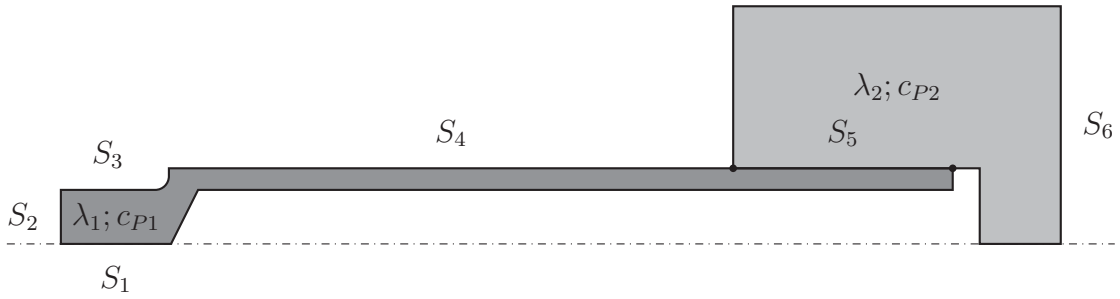


Abbildung 4.3: Randbedingungen des thermischen Problems

Tabelle 4.1: Verwendete Wärmeleitfähigkeiten und spezifische Wärmekapazitäten, entnommen aus [57].

Temperatur ϑ [°C]	Wärmeleitfähigkeit λ [W m^{-1} K]		spezifische Wärmekapazität c_P [J kg^{-1} K]	
	λ_1	λ_2	c_{P1}	c_{P2}
20	13,01	400	493,31	390
100	15,08		500,10	
200	16,79		508,73	
300	18,16		517,50	
400	19,51		526,43	
500	20,86		535,51	
600	22,25		544,74	
700	23,70		554,14	
800	25,20		563,69	
900	26,78		573,42	
1000	28,43		583,31	
1100	30,16		593,37	
1200	31,98		603,60	

der Probe ist bestimmt durch die Wärmeleitfähigkeit λ_1 und die spezifische Wärmekapazität c_{P1}, in der Einspannung respektive durch λ_2 & c_{P2}. Die verwendeten Werte sind in Tabelle 4.1 zu finden. Für das thermische Modell wurde der Elementtyp DCAX8 in Abaqus verwendet.

An den beiden Flächen S_1 & S_2 kommt es aufgrund der Symmetriebedingungen zu keinem Wärmefluss, damit gilt $\frac{\partial Q_1}{\partial t} = 0$ und $\frac{\partial Q_2}{\partial t} = 0$. An den Flächen S_3 & S_4 kommt es zu einem Wärmefluss aufgrund von Wärmestrahlung mit dem Emissionskoeffizienten $\varepsilon_{Strahlung}$. Die Sichtfaktoren zwischen Probe und Umgebung wurden nicht berücksichtigt. Die Konvektion zur umgebendem Atmosphäre wurde nicht berücksichtigt, da der Versuch unter Vakuum stattfindet.

Die Fläche S_5 wird von der Torsionsprobe und der Einspannung geteilt. An ihr kommt es zu einem Wärmefluss durch Kontakt. Die Modellierung erfolgte durch einen thermischen Kontaktkoeffizienten h_T. Zuletzt tritt an der Fläche S_6 noch ein Wärmestrom auf. Dieser beschreibt die Wasserkühlung der Einspannung. Modelliert wurde der Wärmestrom durch Konvektion mittels Wärmeübergangskoeffizient h_W. Die Fluidtemperatur wurde dabei als konstant 25 °C festgelegt.

Für den Emissionskoeffizienten $\varepsilon_{Strahlung}$, den thermischen Kontaktkoeffizienten h_T und den Wärmeübergangskoeffizient h_W waren keine Daten vorhanden. Diese können aber aus den experimentellen Versuchsdaten der Abkühlung aus Abschnitt 3.2.1 ermittelt werden. Aus Vorstudien des FE-Modells war bekannt, dass das Abkühlverhalten der Probemitte (T_1) fast ausschließlich durch Wärmestrahlung be-

Tabelle 4.2: Verwendete elastische Eigenschaften, entnommen aus [58].

Temperatur ϑ [°C]	E-Modul E·10^5 [MPa]		Poissonzahl ν	
	E_1	E_2	ν_1	ν_2
20	197,4	180	0,27	0,3
100	191,1		0,27	
200	183,4		0,27	
300	176,1		0,27	
400	169,0		0,27	
500	162,0		0,27	
600	154,9		0,27	
700	147,3		0,27	
800	138,8		0,27	
900	129,0		0,27	
1000	117,4		0,27	
1100	103,2		0,27	
1200	85,8		0,27	

stimmt wird. Der Anteil der durch Wärmeleitung in die Einspannung fließt kann bei den Temperaturen von über 600 °C vernachlässigt werden. Somit hängt die Abkühlung nur vom Emissionskoeffizienten an der Oberfläche ab. Dadurch kann dieser durch einen Fit an experimentelle Daten bestimmt werden. Die zwei weiteren Parameter des FE-Modells (Wärmeübergangswiderstand h_W und Kontaktkoeffizient h_T) können dann in einem zweiten Fit angepasst werden. Diese haben den größten Einfluss auf die Temperatur T_4. Sie werden so bestimmt, dass die Übereinstimmung zwischen Simulation und Experiment am größten ist.

4.1.3 Mechanisches Modell

Im mechanischen Modell wurden alle für den Versuch relevanten mechanischen und translatorischen Randbedingungen implementiert. Das elastische Verhalten der Torsionsprobe wurde durch einen temperaturabhängigen E-Modul E_1 und eine konstante Poissonzahl ν_1 beschrieben. Für die Einspannung folgte analog dazu E_2 & ν_2, jedoch ohne Temperaturabhängigkeit. Die verwendeten Werte sind in Tabelle 4.2 zu finden. Für das mechanische Modell wurde der Elementtyp CGAX4 in Abaqus verwendet.

An der Fläche S_1 wurde die Radialverschiebung fixiert ($u_1 = 0$). Dies folgt als Konsequenz aus der Axialsymmetrie. Die gleiche Randbedingung wurde auch für die Einspannung an der Fläche S_5 erstellt. An der Fläche S_2 wurde die Axialverschiebung und Rotation um die Symmetrieachse gesperrt ($u_2 = 0;\ u_{R2} = 0$), begründet durch die Spiegelsymmetrie. Die Vorgabe der Rotation erfolgte durch den Referenz-

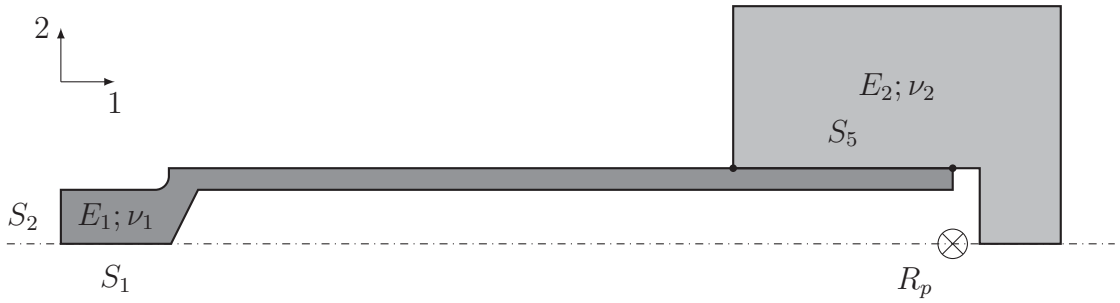


Abbildung 4.4: Randbedingungen des mechanischen Problems

punkt R_P. Dieser wurde auf der Symmetrieachse fixiert ($u_2 = 0;\ u_{R2} = \phi$). Um die Rotation auf die Probe und die Einspannung zu übertragen wurden folgende Bedingungen formuliert (Die Fläche S_5 wird von Probe und Einspannung geteilt):

- $u_2(R_P) = u_2(S_5)$ und
- $u_{R2}(R_P) = u_{R2}(S_5)$.

Da die modellierten Verdrehungen sehr groß waren, wurden bei diesem Modell auch die geometrischen Nichtlinearitäten berücksichtigt.

4.1.4 Thermisch-Elektrisches Modell

Für das thermisch-elektrische Modell wurde der Elementtyp DCAX4E in Abaqus verwendet. Im thermisch-elektrischen Modell wurden alle thermischen Randbedingungen aus Abschnitt 4.1.2 implementiert. Zusätzlich wurden noch elektrische Randbedingungen formuliert. Diese waren:

- An der Fläche S_2 kommt es zu einem Stromfluss mit der Stromdichte j.
- An der Fläche S_5 wirkt die elektrische Kontaktleitfähigkeit $\sigma_{Kontakt}$ zwischen Probe und Einspannung. Da keine Information über diese bekannt waren, wurde diese als $10^{10}\ \Omega^{-1}\,\mathrm{m}^{-2}$ angenommen. Dies ist ein großer Wert, sodass der Widerstand an der Kontaktfläche sehr gering wird.
- Die Fläche S_6 bildet die elektrische Masse und hat damit das Potential $U = 0$.

Der Zusammenhang zwischen Stromstärke und elektrischem Potential in der Torsionsprobe ist durch die spezifische Leitfähigkeit σ_{LFK1} gegeben. In der Einspannung respektive durch σ_{LFK2}. Die dafür verwendeten Werte sind in Tabelle 4.3 zu finden.

Tabelle 4.3: Verwendete elektrische Eigenschaften, berechnet von JMatPro [59].

Temperatur ϑ [°C]	elektrische Leitfähigkeit $\sigma_{LFK} \cdot 10^6$ [$\Omega^{-1}\,\mathrm{m}^{-1}$]	
	σ_{LFK1}	σ_{LFK2}
25	1,238	58
100	1,153	
200	1,070	
300	1,009	
400	0,962	
500	0,925	
600	0,895	
700	0,870	
800	0,846	
900	0,825	
1000	0,810	
1100	0,797	
1200	0,785	
1300	0,775	

Abbildung 4.5: Randbedingungen des elektrischen Problems

Tabelle 4.4: Bestimmte Parameter PID-Regelkreis.

proportionale Verstärkung K_p	integrale Zeitkonstante t_i	differentielle Zeitkonstante t_d
1200000	0,392	0

Die Vorgabe der Temperatur erfolgte durch die Stromdichte j an der Fläche S_2. Eine höhere Stromdichte resultiert folglich in einer höheren Temperatur. Im realen Versuch erfolgt diese Steuerung über eine PID-Regelung des Stroms mithilfe eines Regelthermoelementes. Um dieses Verhalten zu modellieren, wurde in Abaqus eine PID-Regelung der Stromdichte über die UserRoutine UAMP implementiert (siehe Anhang B.2). In Abbildung 4.6 ist das dazugehörige Flussdiagramm der implementierten Subroutine dargestellt. Die Stromdichte j an der Fläche S_2 bildet dabei die Steuergröße und die Temperatur T_1 am Punkt P die Ausgangsgröße, deren Zielwert T_{set} darstellt. Der aktuelle Fehler im Inkrement der Länge dt ist e_i, der Fehler des vorherigen Inkrements e_{i-1}, die Fehlersumme e_{sum} und die maximal mögliche Stromdichte j_{max}. Eine prinzipiell ähnliche Implementierung wurde von Kardoulaki et al. [53] verwendet, wobei in diesem Zusammenhang ein anderer Algorithmus als der PID-Algorithmus verwendet wurde.

Die Parameter für den PID-Regler wurden mit der Frequency Response Methode nach Ziegler-Nichols bestimmt (siehe Kapitel 2.5) und befinden sich in Tabelle 4.4. Zusätzlich zur standardmäßigen PID-Implementierung wurde der Algorithmus mit zwei Methoden erweitert. Die erste verhindert, dass die Fehlersumme im Integralteil sich übermäßig aufsummiert. Dazu wird bei einem Vorzeichenwechsel des aktuellen Fehlers die Fehlersumme auf Null zurück gesetzt. Die zweite Methode begrenzt den Ausgang der Stromdichte auf sinnvolle Werte.

4.1.5 Thermisch-Mechanisches Modell

Im thermisch-mechanischen Modell wurden alle thermischen Randbedingungen aus Abschnitt 4.1.2 und alle mechanischen Randbedingungen aus Abschnitt 4.1.3 implementiert. Zusätzlich wurde noch der Wärmeeintrag durch die plastische Umformenergie berücksichtigt. Diese Wärmeenergie Q lässt sich bestimmen als:

$$Q = \kappa \sigma \varepsilon_p \tag{4.1}$$

mit κ als inelastischer Wärmefaktor, der Spannung σ und der plastischen Verformung ε_p. Der Faktor κ wurde mit 0,9 angenommen [60]. Für das thermisch-mechanische Modell wurde der Elementtyp CGAX4T in Abaqus verwendet.

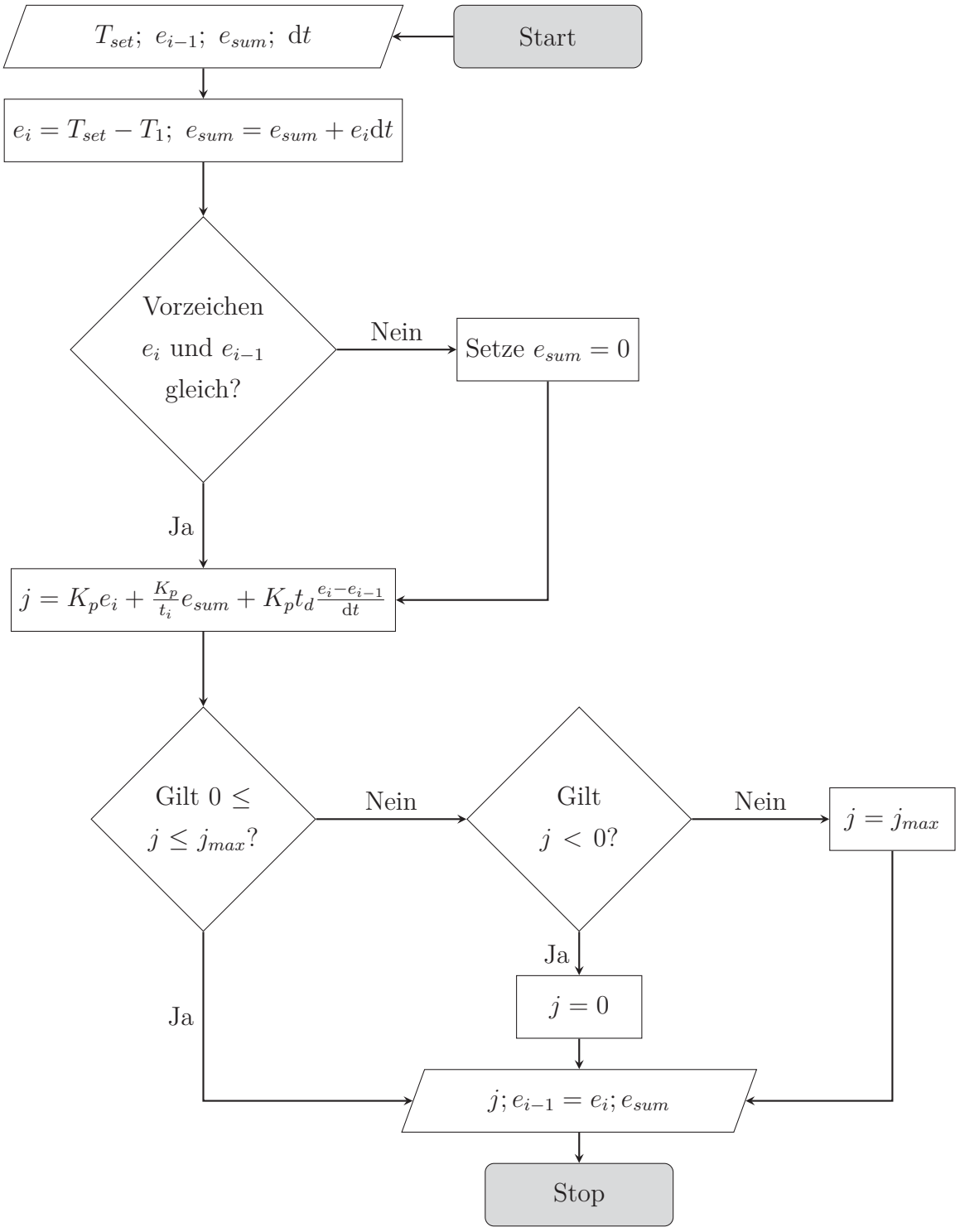


Abbildung 4.6: PID-Regelkreis zur Temperatursteuerung. Implementiert in der Abaqus Subroutine UAMP.

4.1.6 Materialmodell

Für die Simulation wurde das Hensel-Spittel Materialmodell mit 7 Parametern gewählt:

$$\sigma_f = A \cdot e^{m_1\vartheta} \cdot \varphi^{m_2} \cdot e^{m_4/\varphi} \cdot (1+\varphi)^{m_5\vartheta} \cdot e^{m_7\varphi} \cdot \dot{\varphi}^{m_8\vartheta}. \tag{4.2}$$

Die Umsetzung in Abaqus erfolgte über die UserRoutine UHARD (siehe Anhang B.1). Das Materialmodell kann keine Werte für sehr niedrige Umformgrade liefern ($\varphi < 0,01$). Der Bereich ist in der Simulation aber sehr wichtig, da dies genau der Beginn der plastischen Verformung ist. Um dieses Problem zu umgehen, wurde der Umformgrad für das Materialmodell nach $\varphi = \varphi + 0,01$ berechnet. Ein weiteres Problem tritt bei großen Umformgraden ($\varphi > 1,5$) auf. Hier beginnen die Exponentialterme unrealistisch starken Einfluss zu nehmen. Deshalb wurde die Fließspannung ab dann konstant gelassen, sodass gilt: $\sigma_f(\varphi > 1,5) = \sigma_f(\varphi = 1,5)$. Diese Annahme lässt sich als steady-state Bereich, wie er bei der dynamischen Erholung und Rekristallisation auftritt, deuten.

4.2 Inverse Analyse des Torsionsversuches

Der Ansatz einer inversen FE-Analyse dient der Auswertung der Messdaten aus dem Heißtorsionsversuch wie er in Absatz 2.2.3 beschrieben ist. Instrumentell werden in diesem Versuch das Torsionsmoment $\hat{M}_T$ und die Verdrehung ϕ erfasst. Diese Auswertung wird dann notwendig, wenn es während des Versuches zu einer lokalen Instabilität gekommen ist.

Für die inverse Analyse wird das thermo-mechanische FE-Modell aus dem vorherigen Kapitel herangezogen. Dieses ermöglicht die Berechnung eines Drehmoment-Drehwinkel-Verlaufs ausgehend von einem Parametersatz des Hensel-Spittel Modells. Durch die Optimierung der Parameter wird dann die simulierte $M_T - \phi$ -Kurve an die experimentelle $\hat{M}_T - \phi$ -Kurve angepasst. In den so bestimmten Parametern des Materialmodells steckt die Auswertung der Versuchsdaten. Durch das parametrisierte Hensel-Spittel Modell folgt der gesuchte Zusammenhang von Dehnung, Dehnrate, Temperatur und Fließspannung. Über das thermo-mechanische FE-Modell können dann zusätzlich Spannung, Dehnung und Temperatur während der Torsion berechnet werden.

Die Parameteroptimierung stellt die größte Schwierigkeit der inversen Analyse dar. Die konkrete Umsetzung mittels Nelder-Mead Algorithmus ist im folgenden Absatz beschrieben.

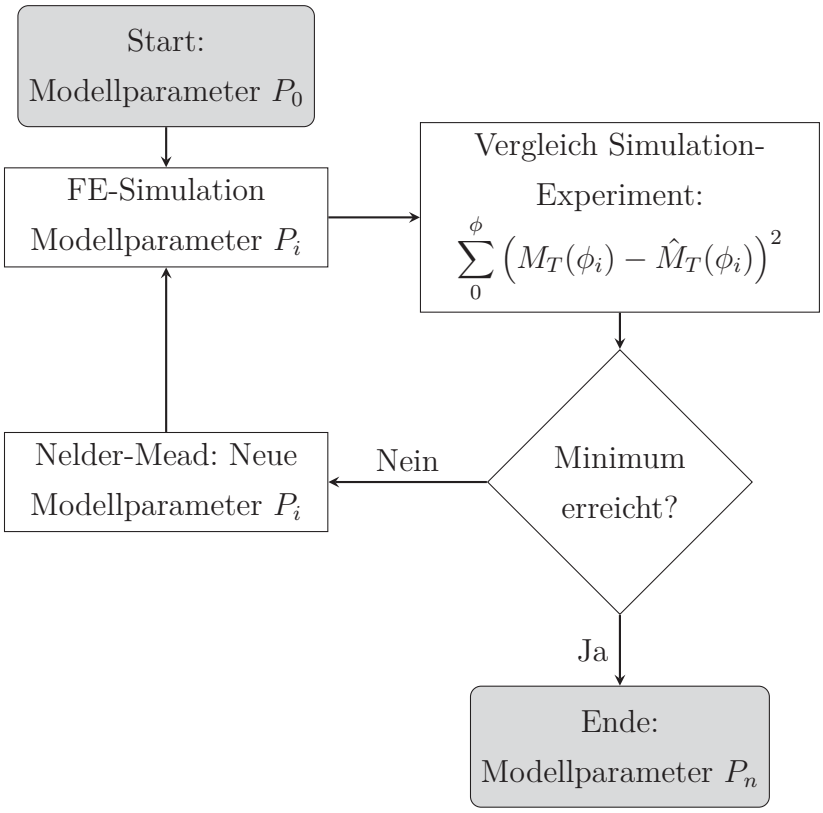


Abbildung 4.7: Optimierungszyklus zur Bestimmung der Materialparameter. Implementiert in Python mittels Scipy.

Zu Beginn werden Startwerte P_0 für die Parameter des Hensel-Spittel Modells gewählt. Ausgehend von diesen wird eine FE-Simulation gerechnet, deren Ergebnis ein $M_T - \phi$ Verlauf ist. Dann wird die quadratische Differenz aus dem simulierten und experimentellen[1] Drehmoment gebildet und über alle Drehwinkel aufsummiert. Das entspricht der Quadratfehlersumme. Daraus ergibt sich ein Wert, der als Maß für die Übereinstimmung dient. Basierend auf dieser Abweichung berechnet der Nelder-Mead Algorithmus dann einen neuen Parametersatz P_i. Dieser Zyklus wird wiederholt, bis die Abweichung ihr Minimum erreicht hat. Dann liegen die finalen Materialparameter P_n als Ergebnis der inversen Analyse vor. Dieser eben beschriebene Zyklus ist auch in Abbildung 4.7 dargestellt.

Aufgrund der vielen Parameter des Materialmodells stellt die Abweichung zu den Versuchsergebnissen keine konkave Fläche dar. Der Nelder-Mead Algorithmus

[1] Als experimentelle Daten können nicht die Rohdaten des Torsionsversuches verwendet werden, denn deren Erfassung erfolgt in einem regelmäßigen Zeitintervall. Im Vergleich dazu wird in der FE-Simulation ein regelmäßiges Drehungsintervall vorgegeben. Deshalb werden die Messdaten mittels Savitzky-Golay Filter [61] geglättet und anschließend linear interpoliert. Damit können die experimentellen Drehmomente an den selben Drehungen ϕ, wie in der Simulation, berechnet werden.

ist nur in der Lage ein lokales Minimum zu finden, welches durch die Startwerte der Optimierung bestimmt wird. Idealerweise sollte ein Optimierungsalgorithmus verwendet werden, der in der Lage ist, ein globales Minimum zu suchen [62]. Da in jedem Iterationsschritt allerdings eine FE-Simulation gerechnet werden muss, führt dies sehr schnell zu einem enormen Rechenaufwand.

5 Ergebnisse & Diskussion

5.1 Axialer Temperaturgradient

In diesem Abschnitt werden die Ergebnisse der Messungen des axialen Temperaturgradienten präsentiert. Zusätzlich wird noch die Kalibrierung und Validierung des thermisch-elektrischen FE-Modells behandelt.

5.1.1 Experimentelle Ergebnisse

Der gemessene axiale Temperaturgradient ist in Abbildung 5.1 dargestellt. Darauf ist der Temperaturverlauf der vier Thermoelemente zu erkennen. Das Thermoelement T_1 befindet sich auf der eingestellten Solltemperatur, während die anderen Thermoelemente ihre Gleichgewichtstemperatur annehmen.
Bei 900 °C beträgt die Temperatur T_2: 896 °C, die Temperatur T_3: 888 °C und die Temperatur T_4 schwankt zwischen 793 °C und 820 °C.
Bei 1000 °C beträgt die Temperatur T_2: 995 °C, die Temperatur T_3: 986 °C und die Temperatur T_4 schwankt zwischen 914 °C und 933 °C.
Bei 1100 °C beträgt die Temperatur T_2: 1093 °C, die Temperatur T_3: 1082 °C und die Temperatur T_4 schwankt zwischen 1000 °C und 1028 °C.
Bei 1200 °C beträgt die Temperatur T_2: 1194 °C, die Temperatur T_3: 1184 °C und die Temperatur T_4 schwankt zwischen 1090 °C und 1133 °C.
Der gemessene Temperaturgradient innerhalb der Prüflänge (T_1 - T_3) ist immer geringer als 20 °C und nimmt mit steigender Temperatur zu. Der axiale Temperaturgradient ist somit bei 1200 °C größer als bei 900 °C.

Die Temperatur innerhalb der Prüflänge (T_1 - T_3) schwankt wenig und befindet sich rasch im Gleichgewicht. Ein anderes Verhalten ist beim vierten Thermoelement zu beobachten. Dort schwankt die Temperatur noch deutlich, während sie in der Mitte schon konstant ist. Die Erklärung für dieses Verhalten liegt vermutlich in der Änderung des Emissionskoeffizienten an der Oberfläche. Die Versuche wurden in Vakuum durchgeführt, allerdings scheint es trotzdem während des Versuches zu einer Veränderung der Oberfläche zu kommen. Bestätigung dafür waren Versuche, bei denen die Probe zuerst unter Luft erhitzt wurde und somit gänzlich oxidierte. Diese

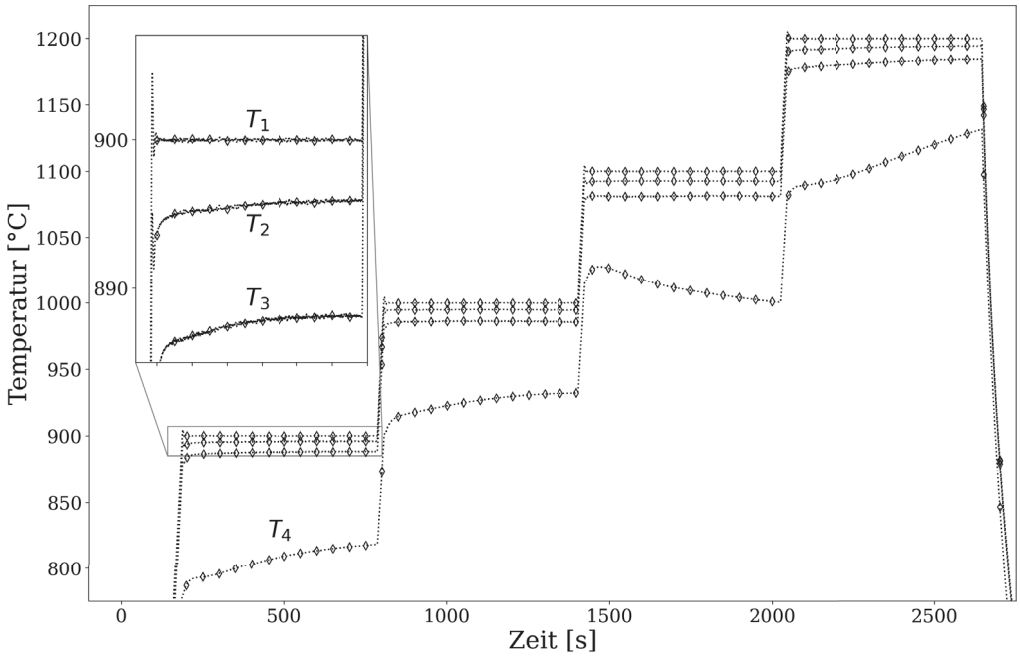


Abbildung 5.1: Im Vakuum gemessener axialer Temperaturgradient der Torsionsprobe an den vier Messpositionen T_1 bis T_4. In der Vergrößerung ist der Temperaturgradient bei 900 °C zu sehen.

zeigte bei der darauffolgenden Messung des Temperaturgradienten unter Vakuum nahezu keine Schwankung der einzelnen Temperaturen, siehe Abbildung 5.2.

Deutlicher zeigt sich die Auswirkung der Oberflächenoxidation bei Betrachtung der Abkühlgeschwindigkeit (siehe Abbildung 5.3). Jene Probe, bei der die Oberfläche zuerst oxidiert wurde, kühlt erheblich schneller ab. Die Begründung liegt in einem höheren Emissionskoeffizienten. So braucht die nicht-oxidierte Probe 185 s um von 1200 °C auf 600 °C abzukühlen, die oxidierte Probe jedoch nur 85 s.

Der Einfluss der Oberflächenoxidation auf den axialen Temperaturgradienten im stationären Zustand ist hingegen sehr gering. So lässt sich beim Vergleich zwischen oxidierter und nicht oxidierter Probe kaum ein Unterschied erkennen (siehe Abbildung 5.1 und 5.2). Der Temperaturgradient innerhalb der Prüflänge liegt bei beiden Proben unter 20 °C. Bei der oxidierten Probe ist er geringfügig größer.

5.1.2 Thermisch-elektrisches FE-Modell

Kalibrierung

Mit den Ergebnissen der Messungen des axialen Temperaturgradienten aus dem vorherigen Abschnitt konnte das thermisch-elektrische FE-Modell kalibriert werden. Dazu wurde, wie in Abschnitt 4.1.2 beschrieben, das experimentelle Abkühlverhal-

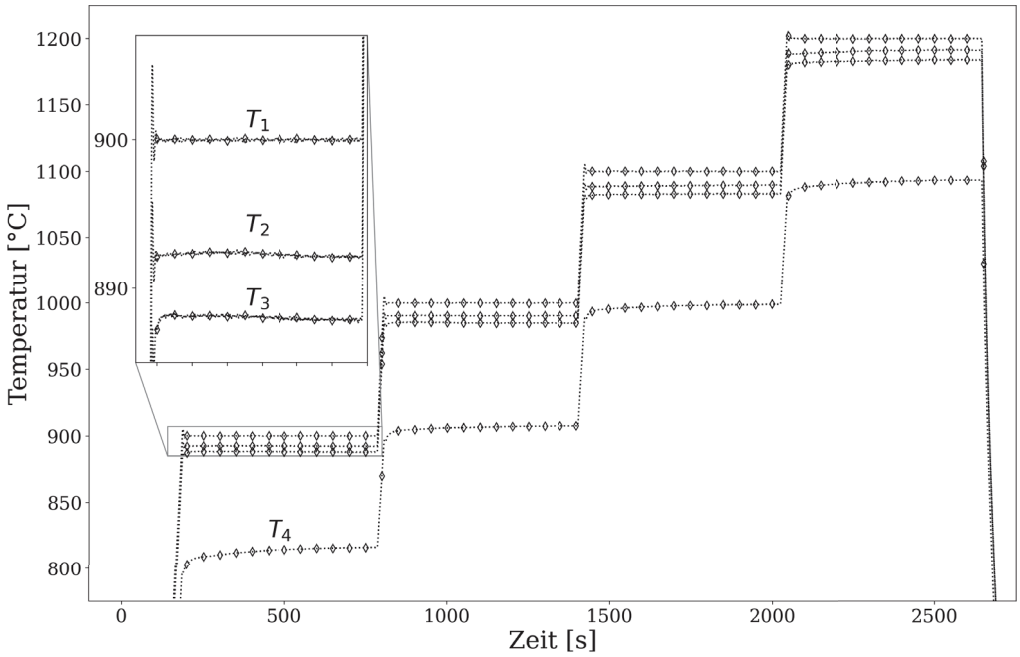


Abbildung 5.2: Im Vakuum gemessener axialer Temperaturgradient der Torsionsprobe an den vier Messpositionen T_1 bis T_4. Die Probe wurde vor den Messungen 600 s bei 1200 °C an Luft oxidiert. In der Vergrößerung ist der Temperaturgradient bei 900 °C zu sehen.

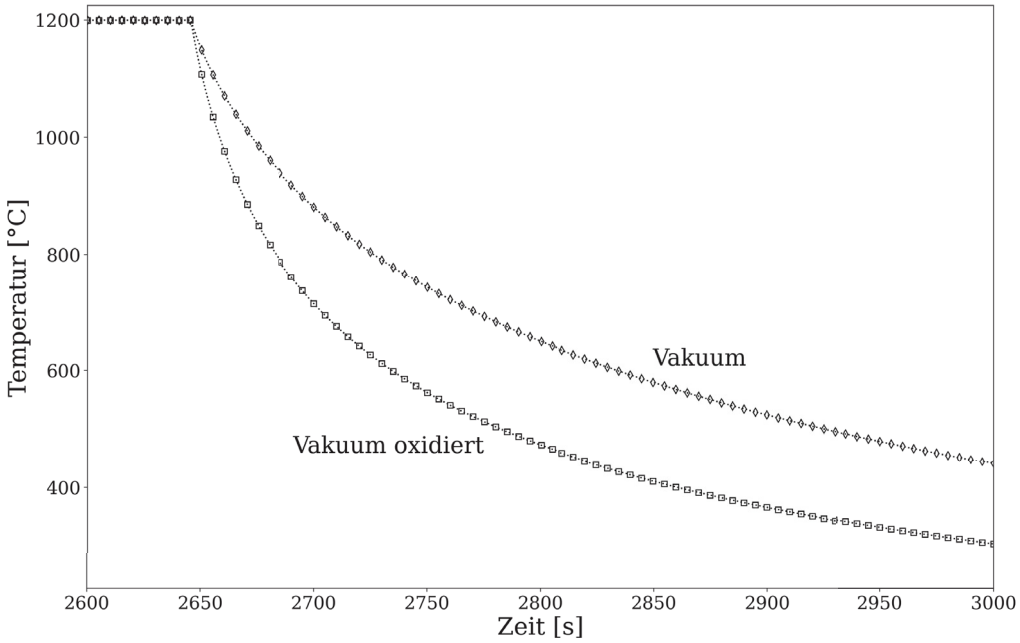


Abbildung 5.3: Unterschied des Abkühlverlaufes der Torsionsprobe für die Temperatur T_1 in Abhängigkeit der Oberflächenoxidation.

Tabelle 5.1: Parameter des thermisch-elektrischen FE-Modells.

Emissions-koeffizient $\varepsilon_{Strahlung}$	Wärmeübergangs-koeffizient h_W [$\mathrm{W\,m^{-2}\,K^{-1}}$]	thermische Kontakt-koeffizient h_T [$\mathrm{W\,m^{-2}\,K^{-1}}$]
0,26	430	160

ten zur Bestimmung des Emissionskoeffizienten $\varepsilon_{Strahlung}$ herangezogen. In einem zweiten Schritt wurden dann noch der Wärmeübergangskoeffizient h_W an der Einspannung und der thermische Kontaktkoeffizient h_T ermittelt. In Tabelle 5.1 finden sich die so bestimmten Parameter.

Der Emissionskoeffizient der nicht-oxidierten Probe wurde mit 0,26 bestimmt. Die Temperaturabhängigkeit wurde vernachlässigt, somit bildet dieser den Mittelwert über den gesamten Temperaturbereich. Ein Vergleich mit Literaturwerten ist schwierig, da der Emissionskoeffizient sehr stark von der Oberfläche (Rauigkeit, Oxidation,...) abhängt. Der Wärmeübergangskoeffizient h_W an der Einspannung wurde mit $430\,\mathrm{W\,m^{-2}\,K^{-1}}$ bestimmt und der thermische Kontaktkoeffizient h_T mit $160\,\mathrm{W\,m^{-2}\,K^{-1}}$. Da die Geometrie der Einspannung vereinfacht angenommen wurde, haben die exakten Größen der beiden Parameter keinen realen Bezug zu den in der Literatur [53] erwähnten Werten.

Bei der Kalibrierung des Wärmeübergangskoeffizienten h_W und des thermischen Kontaktkoeffizienten h_T fiel auf, dass deren Einfluss auf den axialen Temperaturgradienten innerhalb der Prüflänge sehr gering ist. Diese bewirken vor allem eine Änderung der Temperatur T_4.

Validierung

Nach der Kalibrierung folgte die Validierung des thermisch-elektrischen FE-Modells. Dazu wurde der experimentelle Temperaturverlauf aus Abbildung 3.3 simuliert. Das Modell liefert dann das komplette Temperaturfeld innerhalb der Probe. Zur Validierung sind die Temperaturen T_2 und T_3 herangezogen worden. Der Vergleich zum gemessen Temperaturgradienten befindet sich in Abbildung 5.4. Hier wird sofort die gute Übereinstimmung zwischen realem Versuch und Modell ersichtlich. Bei 900 °C liegt die Abweichung bei unter 1 °C. Bei den höheren Temperaturen steigt die Abweichung an, liegt jedoch immer unter 4 °C. Die Abweichung liegt damit unter der Messgenauigkeit der Thermoelemente. So wird die Präzision für Typ-K Thermoelemente üblicherweise mit 0,75% spezifiziert, bei 1200 °C bedeutet das ein maximaler Fehler von 9 °C. In den Randbereichen bei der Temperatur T_4 ist die Abweichung

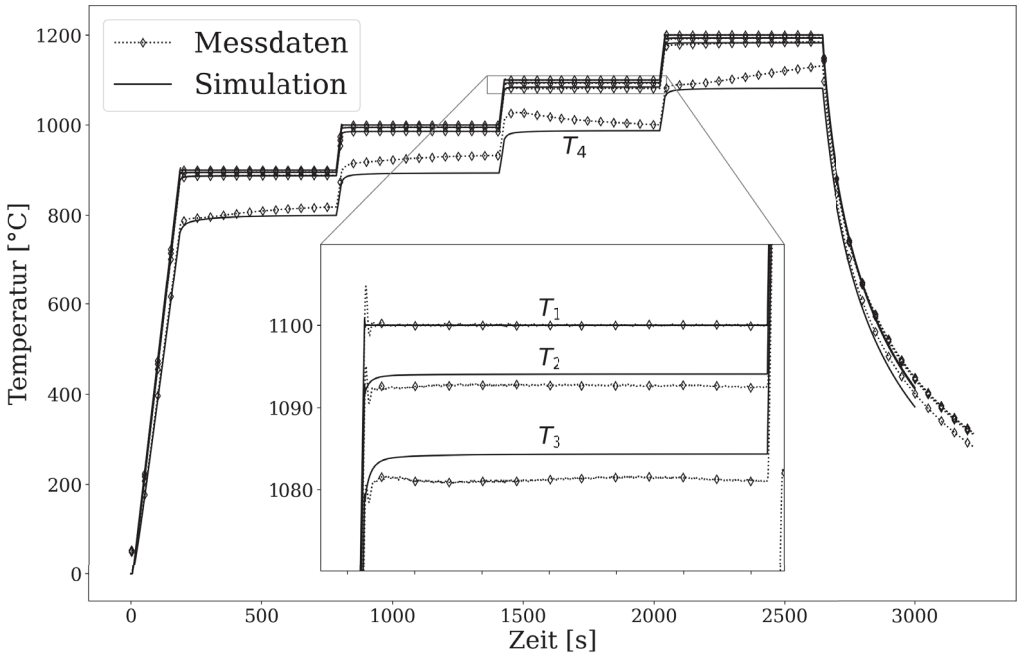


Abbildung 5.4: Validierung des thermisch-elektrischen FE-Modells. Vergleich zwischen experimentellem und simuliertem axialen Temperaturgradienten. In der Vergößerung ist der Temperaturgradient bei 1100 °C dargestellt.

deutlich größer. Für die Verformung spielen diese Bereiche aber keine wichtige Rolle, deshalb kann die vorliegende Abweichung akzeptiert werden.

Neben dem axialen Temperaturgradienten erlaubt das FE-Modell auch Einblick in den radialen Temperaturgradienten. Die Temperaturverteilung innerhalb der Probe bei 1200 °C ist in Abbildung 5.5 dargestellt. Hier wird der radiale Gradient in der Probe erkenntlich. In der Mitte beträgt die Differenz zwischen Kern und Oberfläche etwa 6,5 °C. Semiatin et al. [35] haben sich genauer mit dem radialen Temperaturgradienten beschäftigt. Deren Ergebnisse sind in guter Übereinstimmung zu den hier bestimmten Temperaturen.

Das thermisch-elektrische Modell ist somit in der Lage, die exakte Temperaturverteilung in der Prüflänge zu bestimmen. Und zwar sowohl im stationären Fall als auch für transiente Wärmeverläufe. Das ist besonders interessant, da dadurch beliebige Wärmebehandlungen vor der Verformung simuliert werden können.

Abbildung 5.5: Temperaturfeld innerhalb der Torsionsprobe nach 300 s bei 1200 °C.

5.2 Torsionsversuche

In diesem Abschnitt werden die Ergebnisse der Torsionsversuche präsentiert. Die Messungen wurden bei 5 verschiedenen Temperaturen und 2 Dehnraten durchgeführt.

5.2.1 Drehmomentverlauf

Vergleichsdehnrate $1\,s^{-1}$

Die Ergebnisse, die bei einer Vergleichsdehnrate von $1\,s^{-1}$ gemessen wurden, sind in Abbildung 5.6 dargestellt. Dabei fällt zuerst der große Einfluss der Temperatur auf das maximal auftretende Drehmoment auf. So liegt das maximale Drehmoment während der Torsion bei der Probe mit einer Starttemperatur von 800 °C bei 63 N m. Bei einer Starttemperatur von 1200 °C beträgt das maximale Drehmoment 18 N m. Das Drehmoment nimmt also um den Faktor 3,5 ab. Auch bei den Temperaturen 900 °C, 1000 °C und 1100 °C ist das Verhalten ähnlich. Das maximale Drehmoment bei 900 °C beträgt 49,5 N m, bei 1000 °C 37,5 N m und bei 1100 °C 25 N m.

Auch beim generellen Verlauf des Drehmoments über den Drehwinkel gibt es starke Unterschiede. Bei 1200 °C nimmt das Drehmoment zu Beginn der Verformung zu und erreicht dann ab einem Winkel von 2,7 rad sein Maximum. Das Drehmoment bleibt dann bis zum Ende der Verformung auf nahezu gleichem Niveau. Bei 1100 °C ändert sich das Verhalten leicht, das maximale Drehmoment tritt auch bei 2,7 rad auf, aber die Festigkeit nimmt dann über die weitere Verformung kontinuierlich ab. Am Umformende beträgt das Drehmoment noch 22 N m, also eine Abnahme um 12 %. Dieser Trend der Entfestigung während der Verformung tritt bei 1000 °C noch deutlicher auf. So liegt das Maximum des Drehmoments leicht verschoben bei 2,8 rad

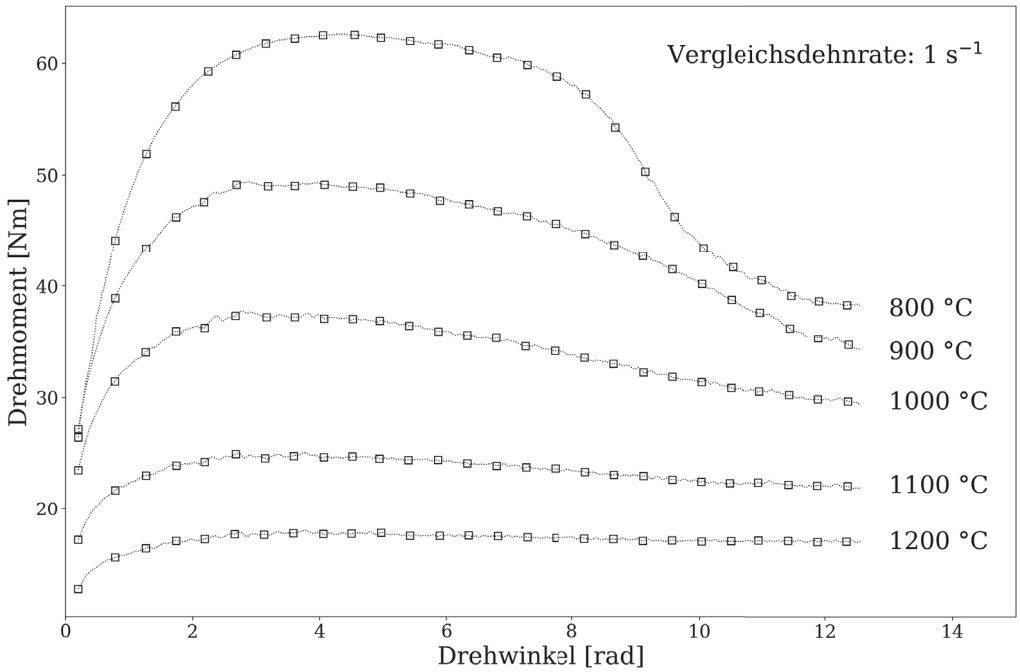


Abbildung 5.6: Experimenteller Drehmomentverlauf der Torsionsproben, bestimmt bei einer Vergleichsdehnrate von $1\,\mathrm{s}^{-1}$.

und fällt dann während der Verformung von 37,5 N m auf 29,5 N m ab. Das entspricht einer Abnahme von 21,3 %. Noch stärker tritt die Entfestigung bei 900 °C und 800 °C auf. So fällt bei 900 °C das Drehmoment vom Maximum bei 3 rad von 49,5 N m auf 34 N m. Die Probe entfestigt um 31,3 % über die gesamte Verformung. Dabei läuft die Entfestigung zu Beginn der Verformung noch langsam ab und wird dann mit fortwährender Verdrehung immer schneller. Enorm stark tritt die Entfestigung bei 800 °C auf. Das Drehmomentmaximum liegt bei 4 rad und 63 N m. Zu Beginn der Verformung ist die Entfestigung der Probe ähnlich schnell wie bei 900 °C und 1000 °C, jedoch nimmt sie ab einem Winkel von 8 rad stark zu. Gegen Ende der Verformung stabilisiert sich die Entfestigung wieder teilweise und nimmt dann mit 38 N m fast den gleichen Wert wie bei 900 °C an. Die Probe bei 800 °C entfestigt während der Verformung schlussendlich um 39,7 %.

Vergleichsdehnrate $10\,\mathrm{s}^{-1}$

Die Ergebnisse, die bei einer Vergleichsdehnrate von $10\,\mathrm{s}^{-1}$ gemessen wurden, sind in Abbildung 5.7 dargestellt. Dabei fällt ebenfalls der große Einfluss der Temperatur auf das maximal auftretende Drehmoment auf. So liegt das maximale Drehmoment während der Torsion bei der Probe mit einer Starttemperatur von 800 °C bei 65 N m. Bei einer Starttemperatur von 1200 °C beträgt das maximale Drehmoment 24 N m. Das Drehmoment nimmt also um den Faktor 2,7 ab. Auch bei den Temperaturen

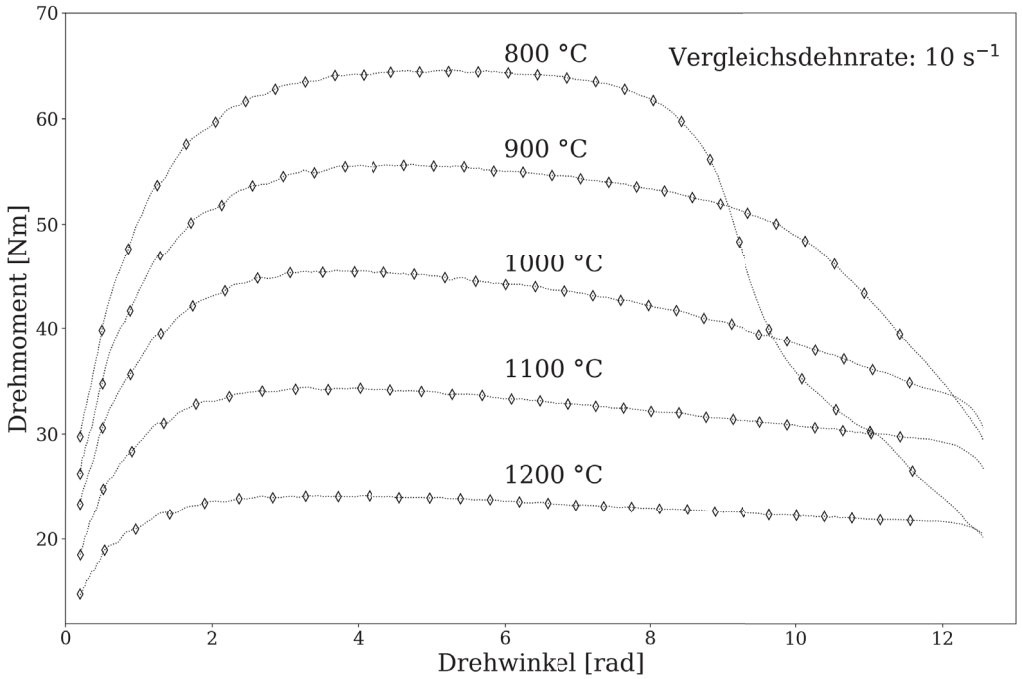


Abbildung 5.7: Experimenteller Drehmomentverlauf der Torsionsproben, bestimmt bei einer Vergleichsdehnrate von $10\,\mathrm{s}^{-1}$.

900 °C, 1000 °C und 1100 °C ist das Verhalten ähnlich. Das maximale Drehmoment bei 900 °C beträgt 56 N m, bei 1000 °C 45 N m und bei 1100 °C 34,5 N m.

Bei 1200 °C ist das Maximum des Drehmoments bei 3 rad mit 24 N m und es kommt zur Entfestigung während der Verformung. Das Drehmoment am Umformende liegt bei 21 N m, die Probe entfestigt somit um 12,5 %. Einen ähnlichen Verlauf hat die Probe bei 1100 °C. Das Drehmoment nimmt im Verlauf der Umformung vom Maximum bei 34,5 N m und einem Drehwinkel von 3,5 rad auf 28,5 N m ab. Das entspricht einer Festigkeitsabnahme von 17,4 %. Bei der Starttemperatur von 1000 °C tritt das maximale Drehmoment bei 3,7 rad mit 45,5 N m auf. Im Zuge der Umformung nimmt dieses kontinuierlich ab und beträgt am Ende 33 N m, also eine Abnahme um 27,5 %. Bei 900 °C und 800 °C fällt sofort auf, dass die Proben am Umformende ein geringeres Festigkeitsniveau haben als die Proben, deren Starttemperatur höher lag. So entfestigt die Probe bei 900 °C zu Beginn ähnlich schnell wie bei 1000 °C. Ab einem Drehwinkel von 10 rad steigt die Entfestigung jedoch rasant an und das Drehmoment am Umformende liegt niedriger als das bei 1000 °C. Am Maximum bei 4 rad beträgt das Torsionsmoment 56 N m und fällt dann im Zuge der Umformung um 43,8 % auf 31,5 N m ab. Zuletzt folgt noch der Verlauf bei 800 °C. Dieser hat das Maximum bei 65 N m und einem Drehwinkel von 4,5 rad. Bis zu einer Umdrehung von 8 rad nimmt das Torsionsmoment nur gering ab, dann jedoch kommt es zu einem rapiden Abfall. Dieser geht soweit, dass das Drehmoment

am Umformende fast niedriger ist als das bei einer Starttemperatur von 1200 °C. Es liegt dann bei 22 N m – die Probe entfestigt somit um 66,2 % bezogen auf das Drehmomentmaximum.

Der generelle Verlauf des Drehmoments sieht ähnlich zu den Versuchen bei der Vergleichsdehnrate von $1\,s^{-1}$ aus. Der größte Unterschied – neben den höheren Drehmomenten – liegt in der stärkeren Entfestigung.

Der Abfall des Torsionsmoments am Umformende (Drehwinkel > 12,5 rad) resultiert aus der Versuchssteuerung. Damit die vorgegebene Umdrehungszahl von zwei nicht überschritten wurde, verlangsamte die Drehung schon geringfügig davor. Es wirkt damit so, als ob die Proben entfestigen würden.

5.2.2 Temperaturverlauf

Im vorherigen Abschnitt sind die Drehmomentverläufe der Torsionsproben beschrieben worden. Vor allem die sehr starke Entfestigung der Proben über die Verdrehung fällt auf. Dabei spielt die Temperatur eine wichtige Rolle, da diese während der Versuche nicht konstant ist. Aufgrund der eingebrachten Verformungsenergie steigt die Temperatur im Verlauf der Verformung an. Zur Quantifizierung dieses Einflusses wurde die Temperatur T_1 der Proben gemessen. Die Ergebnisse sind in Abbildung 5.8 dargestellt. Die Thermoelemente liegen dort genau im Bereich der Umformung. Daher kann es zu elektrischen Kontaktproblemen kommen. Diese haben großen Einfluss auf die Präzision der Messungen und können schnell zu ungültigen Ergebnissen führen.

Bei einer Starttemperatur von 1200 °C und einer Vergleichsdehnrate von $1\,s^{-1}$ sinkt die Temperatur während der Verformung. Ab einem Winkel von 7 rad zeigt sie sprunghafte Änderungen. Diese Messung war nicht erfolgreich und deren Ergebnisse sind nicht zu verwerten. Im Gegensatz dazu nimmt bei der Vergleichsdehnrate $10\,s^{-1}$ die Temperatur monoton über den Versuch zu. Die Temperatur steigt von den anfänglichen 1200 °C auf 1246 °C am Ende der Umformung. Somit liegt der gesamte Temperaturanstieg bei 46 °C.
Bei einer Starttemperatur von 1100 °C und einer Vergleichsdehnrate von $1\,s^{-1}$ war die Messung ebenfalls erfolgreich. Die Temperatur am Ende des Versuches beträgt 1140 °C und stellt somit einen Anstieg von 40 °C dar. Bei der Messung mit Vergleichsdehnrate $10\,s^{-1}$ traten Kontaktprobleme auf. So steigt die Temperatur bis zu einer Umdrehung von 6 rad an, fängt dann jedoch an zu schwanken.
Bei einer Starttemperatur von 1000 °C und einer Vergleichsdehnrate von $1\,s^{-1}$ war die Messung bis zu einem Winkel von 9,5 rad erfolgreich. Der Temperaturanstieg

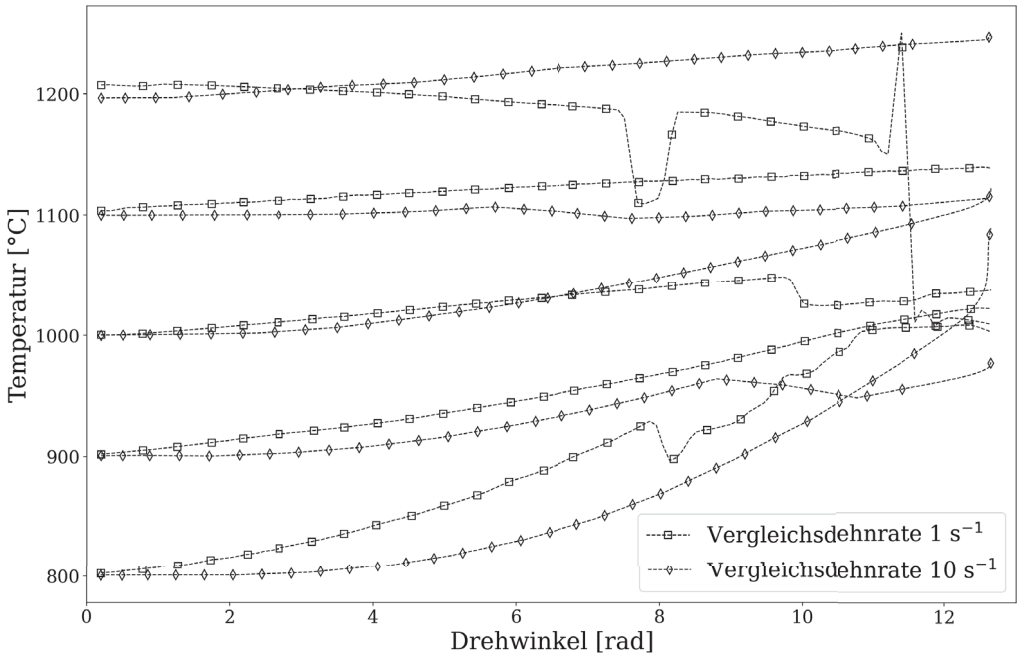


Abbildung 5.8: Verlauf der Temperatur T_1 während dem Heißtorsionsversuch.

an diesem Punkt liegt bei 50 °C. Danach versagte jedoch das Thermoelement. Die Messung bei der Vergleichsdehnrate 10 s^{-1} konnte den kompletten Verlauf erfassen. Die Temperatur steigt zu Umformende auf 1122 °C. Der gesamte Anstieg über den Versuch beträgt somit 122 °C.
Bei einer Starttemperatur von 900 °C und einer Vergleichsdehnrate von 1 s^{-1} steigt die Temperatur monoton an. Am Ende des Versuches liegt sie bei 1024 °C und ist somit um 124 °C angestiegen. Die Messung bei der Vergleichsdehnrate 10 s^{-1} war bis zu einem Winkel von 8,7 rad erfolgreich, danach versagte das Thermoelement. Der Temperaturanstieg bis zu diesem Punkt beträgt 65 °C.
Bei einer Starttemperatur von 800 °C und einer Vergleichsdehnrate von 1 s^{-1} stieg die Temperatur monoton bis zu einem Winkel von 7,8 rad an. Danach traten Kontaktprobleme beim Thermoelement auf. Der Temperaturanstieg bis zu diesem Punkt liegt bei 128 °C. Bei der Messung der höheren Vergleichsdehnrate 10 s^{-1} konnte der gesamte Verlauf erfasst werden. Dieser steigt von den anfänglichen 800 °C auf bis zu 1090 °C an. Die maximale Erwärmung durch die Umformung liegt somit bei 290 °C.

Eine Besonderheit in den Messungen fällt auf, wenn jeweils der Temperaturanstieg bei den unterschiedlichen Dehnraten verglichen wird. Bei gleicher Verdrehung ist die Temperatur bei Vergleichsdehnrate 1 s^{-1} immer höher. Dies ist bei allen Messungen zu beobachten. Der Temperaturanstieg resultiert aus der eingebrachten Verformungsenergie. Diese ist bei Vergleichsdehnrate 10 s^{-1} größer als bei Vergleichsdehnrate 1 s^{-1}. Damit müsste der Temperaturanstieg, bei gleichem Drehwinkel, grö-

ßer sein. Das zeigen die Messungen aber nicht. Eine Erklärung für dieses Verhalten liegt in der Temperaturmessung mittels Thermoelement. Damit ein Thermoelement einen Temperaturanstieg erfassen kann, muss Wärme über Wärmeleitung in dieses fließen. Das passiert aber nicht sofort, sondern benötigt eine gewisse Zeit. Das Thermoelement hat eine bestimmte Ansprechzeit, die abhängig von der Verbindung zur Messstelle und dem Drahtdurchmesser ist [63]. Genau dieser Effekt scheint bei den Versuchen aufzutreten. Die Messungen bei Vergleichsdehnrate $10\,s^{-1}$ haben eine gesamte Versuchsdauer von weniger als 150 ms. Der Temperaturanstieg wird damit zeitverzögert erfasst und ist scheinbar geringer als bei der niedrigeren Dehnrate. Das erklärt auch den starken Temperaturanstieg am Umformende der Probe bei 800 °C und Vergleichsdehnrate $10\,s^{-1}$. Die Temperatur steigt dort noch um 50 °C an, obwohl nahezu keine Umformung mehr stattfindet.

Unter Berücksichtigung des Temperaturverlaufes während der Verformung lässt sich ein entscheidender Grund für die teilweise starke Entfestigung der Proben finden. So ist der Temperaturanstieg für Vergleichsdehnrate $10\,s^{-1}$ bei 1200 °C mit 46 °C eher gering. Der Drehmomentverlauf zeigt dementsprechend nur eine geringe Entfestigung um 12,5 %, bezogen auf das maximale Drehmoment. Gänzlich anders sieht das Verhalten bei einer Starttemperatur von 800 °C und einer Vergleichsdehnrate von $10\,s^{-1}$ aus. Hier beträgt der maximale Temperaturanstieg 290 °C. Im Verlauf des Drehmoments über die Umformung zeigt sich das durch die sehr starke Entfestigung um 66,2 %, bezogen auf das Drehmomentmaximum.

Wünschenswert bei diesen Ergebnissen wäre eine zuverlässigere Temperaturmessung. Da die Thermoelemente genau in der Verformungszone sitzen, hängt deren Erfolg oft vom Zufall ab. Eine mögliche Alternative stellt die kontaktlose Temperaturmessung über Wärmestrahlung dar. Bei dieser kann es zu keinen Kontaktproblemen kommen. Ein weiterer Vorteil wäre die reduzierte Ansprechzeit, welche auch exakte Messungen bei höheren Verformungsgeschwindigkeiten ermöglicht. Hier würde sich besonders ein Quotientenpyrometer anbieten. Bei diesem kann auch die Veränderung des Emissionskoeffizienten ohne Probleme korrigiert werden.

5.2.3 Lokale Verdrehung

Der Zusammenhang zwischen Temperaturanstieg und Entfestigung ist generell vorhanden, kann aber nur unter Kenntnis der lokalen Verformung mit dem Werkstoffverhalten korreliert werden. Die lokale Verdrehung ist aus dem Torsionsmomentenverlauf nicht ableitbar. Gemessen wird die gesamte Verdrehung, die sich als Summe der lokalen Verdrehungen ergibt. Diese wiederum korreliert stark mit der lokalen

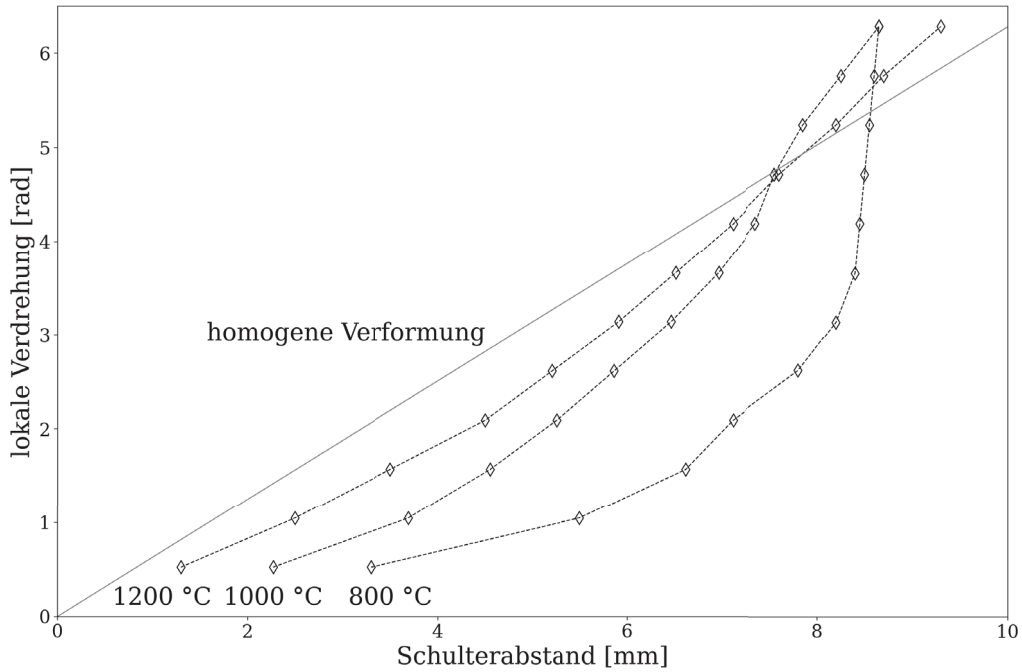


Abbildung 5.9: Lokale Verdrehung der Torsionsproben, bestimmt bei einer Vergleichsdehnrate von $10\,s^{-1}$. Der lokale Umformgrad ist in dieser Darstellung aus der Steigung der lokalen Verdrehung abzuleiten.

Temperatur, über die experimentell nur der Wert in der Mitte der Prüflänge bekannt ist. Zur Erfassung der lokalen Verdrehung muss die Oberfläche markiert werden und nach dem Versuch ausgewertet werden. Dies wurde nach der in Abschnitt 3.2.3 beschriebenen Methode durchgeführt. Die Ergebnisse dieser Auswertung sind in Abbildung 5.9 dargestellt. Die Auswertung wurde jeweils für die Versuche mit Vergleichsdehnrate $10\,s^{-1}$ bei 800 °C, 1000 °C und 1200 °C durchgeführt. In der Auswertung dargestellt ist auch die lokale Verdrehung unter Annahme homogener Verformung. Diese ist bezogen auf die betrachtete Länge von der Schulter bis zur Probenmitte, also 10 mm. In dieser Darstellung ist der lokale Umformgrad proportional zur Steigung der Kurven. Bei der Probe mit einer Starttemperatur von 1200 °C ist ersichtlich, dass der lokale Umformgrad im Randbereich (Schulterabstand < 4 mm) leicht geringer ist als bei der homogenen Verformung. Zwischen 4 mm bis 6 mm Schulterabstand entspricht der lokale Umformgrad etwa der gleichförmigen Verdrehung und im Bereich darüber ist der lokale Umformgrad etwas größer als dies bei homogener Verformung der Fall wäre. Damit kommt es während der Verformung zu einer teilweisen Lokalisation im Bereich der Probenmitte.
Bei der Probe mit einer Starttemperatur von 1000 °C sieht das Verhalten sehr ähnlich aus. Im weit von der Mitte entfernten Bereich (Schulterabstand < 4 mm) liegt der lokale Umformgrad unter dem der homogenen Verformung. Zwischen 4 mm bis

6 mm Schulterabstand ist die Änderung der lokalen Verdrehung etwa gleich groß. Im Bereich über 6 mm Schulterabstand nimmt der lokale Umformgrad zu und liegt dort über der gleichförmigen Verformung. Die Verformung hat somit zu einer lokalisierten Verdrehung im Bereich der Probenmitte geführt, die im Vergleich zur Starttemperatur von 1200 °C stärker ausgeprägt ist.
Die Probe mit einer Starttemperatur von 800 °C zeigt ein deutlich ausgeprägteres Lokalisierungsverhalten. Bis zu einem Schulterabstand von 6,5 mm liegt der lokale Umformgrad unter dem der homogenen Verformung. Im Randbereich (Schulterabstand < 4 mm) liegt er sogar deutlich darunter. Im Bereich zwischen 6,5 mm bis 8 mm Abstand zur Schulter liegt der lokale Umformgrad etwa auf gleichem Niveau wie bei der homogenen Verformung. In der Probenmitte (Schulterabstand > 8 mm) steigt der lokale Umformgrad sehr stark an und liegt weit über dem der homogenen Verformung. Für die Verformung dieser Probe bedeutet das eine sehr starke Lokalisation im Bereich der Probenmitte.

Zu diesen Ergebnissen muss noch angemerkt werden, dass die lokale Verdrehung der Proben erst bei einem Schulterabstand von 10 mm eine Umdrehung (6,28 rad) erreichen müssten. Die Probe bei 1200 °C erreicht diesen Wert aber schon bei 9,3 mm und die Proben bei 800 °C und 1000 °C sogar schon bei 8,65 mm. Dazu kommt es einerseits, weil die Proben während der Torsion geringfügig kürzer werden (Verkürzung um bis zu 1 mm gemessen über die gesamte Probe) und andererseits vermutlich aus der Ungenauigkeit der verwendeten Bestimmungsmethode. Die Erkennung der aufgebrachten Markierung wird mit steigender Lokalisierung deutlich schwieriger, da die Sichtbarkeit der Linie abnimmt.

Aus diesen Messungen wird klar, warum manche Versuche eine solch starke Entfestigung und großen Temperaturanstieg aufweisen. Gerade bei den niedrigeren Starttemperaturen und der höheren Dehnrate wird sehr viel Umformenergie in die Torsionsprobe eingebracht. Die Verformung findet dabei jedoch nicht homogen statt, sondern bildet eine lokale Instabilität. Damit konzentriert sich die Verformung auf einen sehr kleinen Bereich. Das Zusammenspiel aus weiterer Verdrehung und daraus resultierendem Temperaturanstieg führen zu einer starken Rückkoppelung, die die lokale Instabilität stark forciert. Bei der Starttemperatur von 800 °C und Vergleichsdehnrate $10\,\mathrm{s}^{-1}$ ist das sehr gut zu sehen. Generell kann gesagt werden, dass die Verformungslokalisierung steigt, wenn die Starttemperatur sinkt oder die Verformungsgeschwindigkeit steigt.

Damit ist klar, dass die Auswertung des Heißtorsionsversuches nach der klassischen Torsionsanalyse (Abschnitt 2.2.2) an ihre Grenzen stößt. Diese setzt eine homogene Verformung voraus. Um dieses Problem zu umgehen, wurde eine inverse

Tabelle 5.2: Startwerte zur Parameterbestimmung mittels inverser FE-Analyse. Aus [64].

A	m_1	m_2	m_4	m_5	m_7	m_8
7958,3	$-3{,}19 \cdot 10^{-3}$	$2{,}5804 \cdot 10^{-1}$	$4{,}8 \cdot 10^{-4}$	$4{,}9 \cdot 10^{-4}$	$-7{,}7808 \cdot 10^{-1}$	$9{,}9 \cdot 10^{-5}$

Tabelle 5.3: Bestimmte Parameter aus der inversen FE-Analyse.

A	m_1	m_2	m_4	m_5	m_7	m_8
7908,6	$-3{,}2613 \cdot 10^{-3}$	$1{,}0266 \cdot 10^{-1}$	$-1{,}5289 \cdot 10^{-2}$	$-7{,}528 \cdot 10^{-6}$	$-1{,}8834 \cdot 10^{-1}$	$1{,}0069 \cdot 10^{-4}$

FE-Analyse durchgeführt. Deren Ergebnisse werden im folgenden Abschnitt dargestellt.

5.3 Inverse Analyse

Die Versuchsergebnisse aus Abschnitt 5.2.1 wurden herangezogen, um eine inverse FE-Analyse, wie sie in Kapitel 4.2 beschrieben ist, durchzuführen. Für diesen Optimierungsprozess war die Wahl von Startwerten für das Hensel-Spittel Materialmodell erforderlich. Dazu wurden aus der Literatur bekannte Werte für einen chemisch ähnlichen Stahl verwendet. Diese sind in Tabelle 5.2 angeführt. Ausgehend von diesen Werten wurden die Materialparameter optimiert, bis die Abweichung zu den experimentellen Torsionsergebnissen minimal wurde. Die Abweichung wurde dabei nur im Intervall von 0.5 rad bis 10 rad berechnet. Die untere Grenze wurde gesetzt, um Einflüsse der Maschinensteifigkeit zu verringern, während die obere Grenze den Einfluss der lokalen Instabilitäten minimieren sollte.

Zusätzlich war aus Vorversuchen im Zylinderstauchversuch [65] bekannt, dass das Hensel-Spittel Modell nicht in der Lage ist, den kompletten Temperaturbereich mit einem Parametersatz abzudecken. Deshalb wurden die Versuchsdaten von 800 °C in der Optimierung nicht berücksichtigt.

Die Ergebnisse der inversen Analyse sind in Tabelle 5.3 angegeben. Insgesamt waren dazu knapp 1400 Iterationen notwendig.

5.3.1 Drehmomentverlauf

Die Ergebnisse für die optimierten Parameter werden in den folgenden Absätzen dargestellt und diskutiert. Dazu muss erwähnt werden, dass die Simulation auch für eine Starttemperatur von 800 °C gerechnet wurde. Für die Optimierung wurde diese

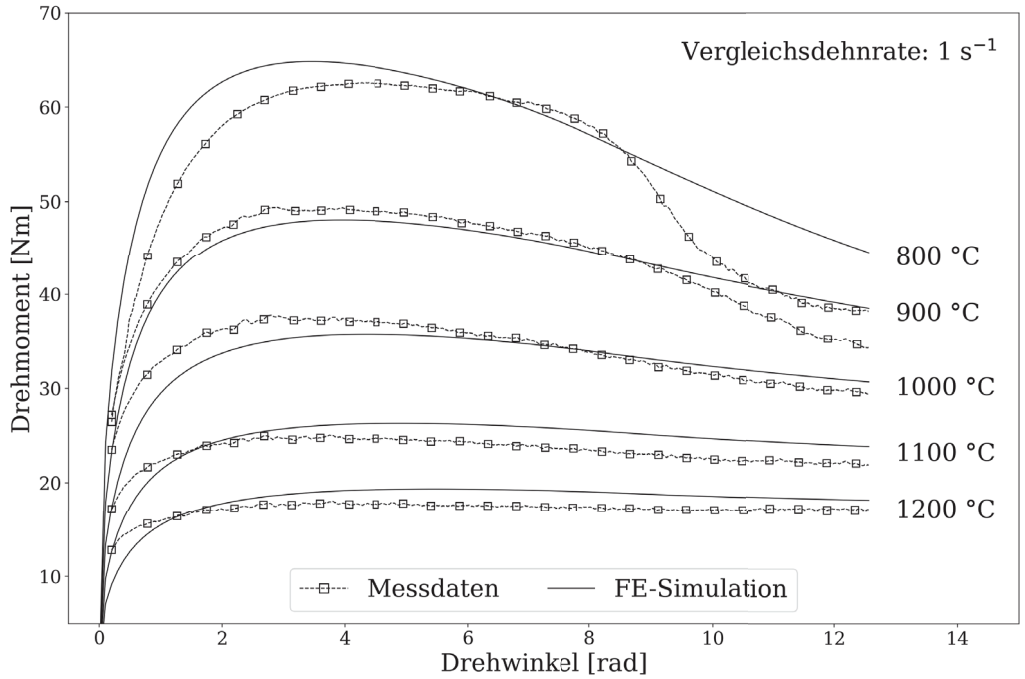


Abbildung 5.10: Simulierter und experimenteller Drehmomentverlauf der Torsionsproben, bestimmt bei einer Vergleichsdehnrate von $1\,s^{-1}$.

Temperatur aber nicht berücksichtigt.

Vergleichsdehnrate $1\,s^{-1}$

Die Ergebnisse der Torsionssimulation mit einer Vergleichsdehnrate von $1\,s^{-1}$ sind in Abbildung 5.10 dargestellt.

Bei einer Starttemperatur von 1200 °C steigt das simulierte Drehmoment am Beginn weniger stark an als im Versuch gemessen wurde. Das gilt bis zu einem Drehwinkel von 1,5 rad, danach ist das simulierte Drehmoment etwas höher. Der Verlauf der Entfestigung ist nahezu gleich, wobei die Simulation ein um 1 N m bis 1,5 N m höheres Drehmoment ergibt.

Bei einer Starttemperatur von 1100 °C sieht das Verhalten ähnlich dem bei 1200 °C aus. Der Anstieg des Drehmoments ist geringer als in den Messungen. Ab einem Drehwinkel von 2 rad liegt das Drehmoment konstant über den experimentellen Werten. Die Entfestigung zeigt einen ähnlichen Verlauf zwischen Simulation und Experiment. Die Simulation liefert eine leicht höhere Festigkeit, welche in einem um 2 N m größeren Drehmoment resultiert.

Bei einer Starttemperatur von 1000 °C liegt das Drehmoment in der Simulation bis zu einem Drehwinkel von 7,5 rad unter dem der Messung. Die Entfestigung in der Simulation wird leicht unterschätzt, sodass ab 7,5 rad das experimentelle Drehmo-

ment geringer ist als das simulierte. Die Abweichung von Simulation zu Messung geht von $-4\,\mathrm{N\,m}$ bei $2\,\mathrm{rad}$ bis zu $1{,}5\,\mathrm{N\,m}$ bei $12{,}56\,\mathrm{rad}$.
Bei einer Starttemperatur von $900\,^{\circ}\mathrm{C}$ liefern die Simulation und das Experiment eine sehr gute Übereinstimmung. Der Verlauf der Entfestigung ist bis zu einem Drehwinkel von $9\,\mathrm{rad}$ nahezu gleich. Über diesen Bereich liegt das Drehmoment in der Simulation niedriger, ist jedoch maximal um $2\,\mathrm{N\,m}$ zu gering. Ab einem Drehwinkel von $9\,\mathrm{rad}$ beginnen die Ergebnisse voneinander abzuweichen. Die im Experiment beobachtete Entfestigung findet in der Simulation weit weniger stark statt.
Die Simulation mit einer Starttemperatur von $800\,^{\circ}\mathrm{C}$ zeigt deutliche Abweichungen zur Messung. So ist das Drehmoment in der Simulation zu Beginn höher als gemessen und fällt dann kontinuierlich ab. Im Experiment kommt es aber zu keiner kontinuierlichen Entfestigung, sondern eher zu einem Plateau gefolgt von einer stark ausgeprägten Entfestigung.

Zusammenfassend kann gesagt werden, dass die Simulationen bei $1100\,^{\circ}\mathrm{C}$ und $1200\,^{\circ}\mathrm{C}$ sehr gut mit den Messungen übereinstimmen. Bei $900\,^{\circ}\mathrm{C}$ und $1000\,^{\circ}\mathrm{C}$ wird die Entfestigung, die während des Torsionsversuches auftritt, in der Simulation leicht unterschätzt. Nur die Messung bei $800\,^{\circ}\mathrm{C}$ weicht deutlich von der Simulation ab. Das war aus Vorversuchen mittels Stauchversuch [65] zu erwarten und hat sich hier bestätigt.

Vergleichsdehnrate $10\,\mathrm{s^{-1}}$

Die Ergebnisse der Torsionssimulation mit einer Vergleichsdehnrate von $10\,\mathrm{s^{-1}}$ sind in Abbildung 5.11 dargestellt.
Bei einer Starttemperatur von $1200\,^{\circ}\mathrm{C}$ ist der simulierte Verlauf des Drehmoments nahezu ident mit der Messung. Zu Beginn ist das Drehmoment um maximal $1\,\mathrm{N\,m}$ niedriger als im Experiment. Das gilt bis zu einem Drehwinkel von $2{,}75\,\mathrm{rad}$, danach liegt die Simulation höher als die Messung, jedoch um maximal $1\,\mathrm{N\,m}$.
Auch bei einer Starttemperatur von $1100\,^{\circ}\mathrm{C}$ ist eine sehr gute Übereinstimmung zwischen Simulation und Experiment festzustellen. Hier liegt das simulierte Drehmoment unter dem gemessenen. Die maximale Abweichung tritt am Beginn der Verformung mit $2\,\mathrm{N\,m}$ auf und wird während der Verdrehung geringer und beträgt dann weniger als $0{,}75\,\mathrm{N\,m}$.
Bei einer Starttemperatur von $1000\,^{\circ}\mathrm{C}$ deckt sich der simulierte und experimentelle Drehmomentverlauf sehr gut. Bis zu einem Drehwinkel von $10\,\mathrm{rad}$ liegt das simulierte Drehmoment unter dem gemessenen. Die maximale Abweichung beträgt $1{,}75\,\mathrm{N\,m}$. Nur gegen Ende der Verformung (Drehwinkel $> 10\,\mathrm{rad}$) zeigt die Messung eine stär-

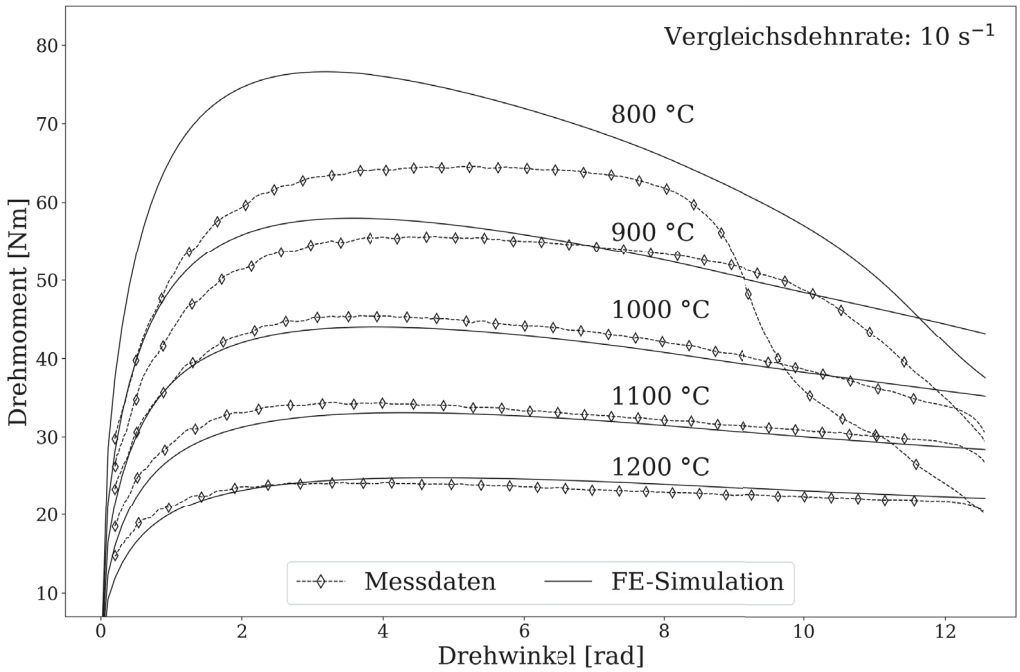


Abbildung 5.11: Simulierter und experimenteller Drehmomentverlauf der Torsionsproben, bestimmt bei einer Vergleichsdehnrate von $10\,s^{-1}$.

kere Entfestigung als die Simulation.
Bei einer Starttemperatur von 900 °C ist die Simulation nicht in der Lage, den experimentellen Verlauf optimal zu beschreiben. Zu Beginn ist das Drehmoment in der Simulation höher als in der Messung und nimmt dann über den Verlauf der Umformung kontinuierlich ab. Das führt dazu, dass ab einem Drehwinkel von 7 rad die Simulation ein niedrigeres Drehmoment vorhersagt als im Experiment gemessen wurde. Dies gilt jedoch nur bis zu einer Verdrehung von 10 rad, da gegen Ende der Verformung eine deutliche Entfestigung im Experiment folgt. In der Simulation tritt keine solche auf.
Bei einer Starttemperatur von 800 °C weichen die Simulation und das Experiment sehr stark voneinander ab. Hier liegt das simulierte Drehmoment deutlich höher als das der Messung. Auch ist das Entfestigungsverhalten zwischen Simulation und Experiment anders. In der Messung tritt ein steady-state Bereich auf, welcher von enormer Entfestigung gefolgt wird. Im Gegensatz dazu kommt es in der Simulation zu einer starken kontinuierlichen Entfestigung.

Zusammenfassend kann gesagt werden, dass die Simulationen bei 1000 °C, 1100 °C und 1200 °C sehr gut mit dem Messungen übereinstimmen. Bei 900 °C wird die Entfestigung, die während des Torsionsversuches auftritt, in der Simulation leicht überschätzt. Bei 800 °C tritt eine deutliche Abweichung zwischen Simulation und Messung auf, welche besonders im unterschiedlichen Entfestigungsverhalten zu se-

hen ist. Dieser Unterschied war aus Vorversuchen mittels Stauchversuch [65] zu erwarten und hat sich bestätigt.

Zusammenfassung Drehmomentverlauf

Aus dem Vergleich zwischen den experimentellen und simulierten Verläufen wird ersichtlich, dass die inverse FE-Analyse zu einem guten Ergebnis geführt hat. Mit den ermittelten Materialparametern können die gemessenen Versuchsdaten sehr gut beschrieben werden. Nur bei 800 °C treten deutliche Unterschiede auf. Hier kann das Hensel-Spittel Modell nicht den kompletten Temperaturbereich abdecken. Ein anderes, möglicherweise präziseres Materialmodell könnte hier zu besseren Ergebnissen führen.

5.3.2 Temperaturverlauf

Bei der inversen FE-Analyse wurden die experimentellen Temperaturergebnisse nicht in die Optimierung einbezogen. Dadurch ergibt sich die Möglichkeit, die gefundenen Parameter über einen Vergleich der Temperaturen zu validieren.

Vergleichsdehnrate $1\,s^{-1}$

In Abbildung 5.12 ist der Temperaturanstieg während des Torsionsversuches abgebildet. Dieser bezieht sich auf die Temperatur T_1 der Torsionsproben.
Bei einer Starttemperatur von 1200 °C steigt in der Simulation die Temperatur um 27 °C auf 1227 °C. Ein Vergleich mit der Messung ist nicht möglich, da das Thermoelement versagt hat und keine sinnvollen Ergebnisse liefern konnte.
Bei einer Starttemperatur von 1100 °C sind gültige Messdaten vorhanden. Der Vergleich zu den in der Simulation bestimmten Werten zeigt eine sehr gute Übereinstimmung. Die Temperatur steigt in der Simulation zu Verformungsbeginn geringfügig langsamer an, zeigt am Ende aber insgesamt einen höheren Temperaturanstieg. 1148 °C in der Simulation zu 1139 °C in der Messung.
Auch bei der Starttemperatur 1000 °C passen die Ergebnisse der Simulation und der Messung sehr gut zusammen. Bei einem Drehwinkel von 9,5 rad liegt die Temperatur in der Simulation bei 1056 °C, im Vergleich zu 1048 °C bei der Messung. Danach hat das Thermoelement versagt und damit keine gültigen Temperaturdaten mehr geliefert.
Bei einer Starttemperatur von 900 °C stimmen die Daten ebenfalls sehr gut überein. Zu Verformungsbeginn ist die Erwärmung nahezu ident. Erst ab einer Verdrehung

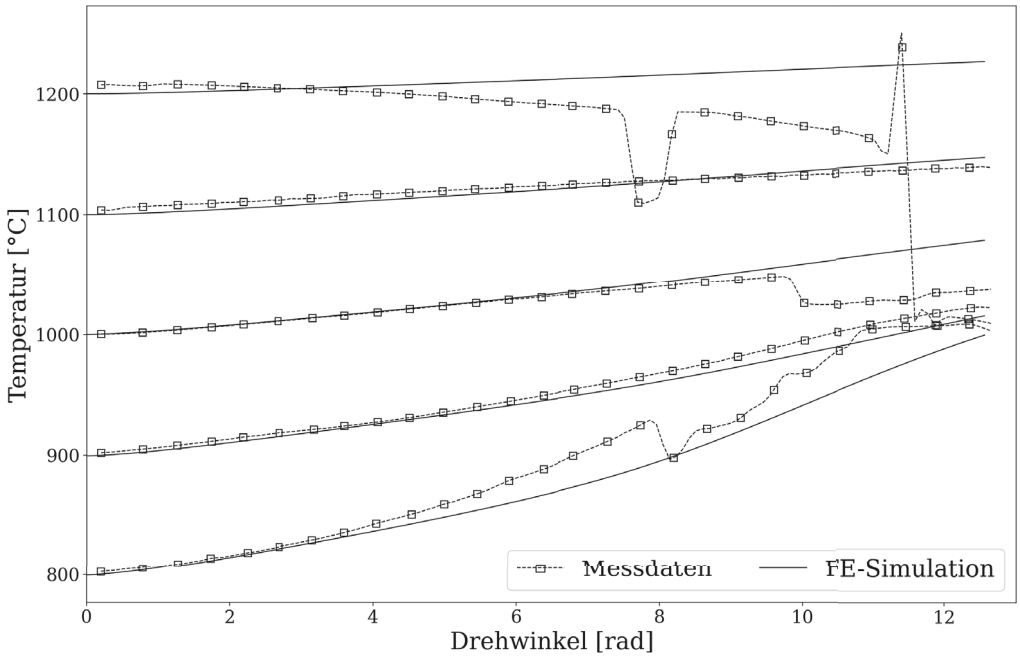


Abbildung 5.12: Vergleich zwischen simuliertem und gemessenem Temperaturanstieg T_1, bestimmt bei einer Vergleichsdehnrate von $1\,\mathrm{s}^{-1}$.

von 5,5 rad steigt die Temperatur in der Messung stärker an. Das führt dann zu einer Endtemperatur von 1023 °C in der Messung und 1013 °C in der Simulation.
Bei einer Starttemperatur von 800 °C beginnt die Erwärmung sehr ähnlich, ab einer Drehung von 3 rad erwärmt sich die Messung jedoch stärker. Die Abweichung wird dabei immer größer, bis schlussendlich das Thermoelement ab einer Drehung von 8 rad versagt. Danach ist kein sinnvoller Vergleich mehr möglich.

Vergleichsdehnrate $10\,\mathrm{s}^{-1}$

In Abbildung 5.13 ist der Temperaturanstieg während des Torsionsversuches abgebildet. Dieser bezieht sich auf die Temperatur T_1 der Torsionsproben.
Bei einer Starttemperatur von 1200 °C führt die Simulation zu einem größeren Temperaturanstieg während der Verformung. Die Differenz zwischen Simulation und Messung nimmt während der Verdrehung kontinuierlich zu und beträgt dann am Ende der Verformung 22 °C.
Bei einer Starttemperatur von 1100 °C konnten keine sinnvollen Messergebnisse ermittelt werden, das Thermoelement hat vorzeitig versagt. Nur über den Temperaturanstieg in der Simulation kann festgestellt werden, dass dieser bei 95 °C liegt.
Bei einer Starttemperatur von 1000 °C trat sehr ähnliches Verhalten wie bei 1200 °C auf. Die Simulation sagt einen größeren Temperaturanstieg als die Messungen voraus. Diese Differenz wächst kontinuierlich über die Verformung an und beträgt dann

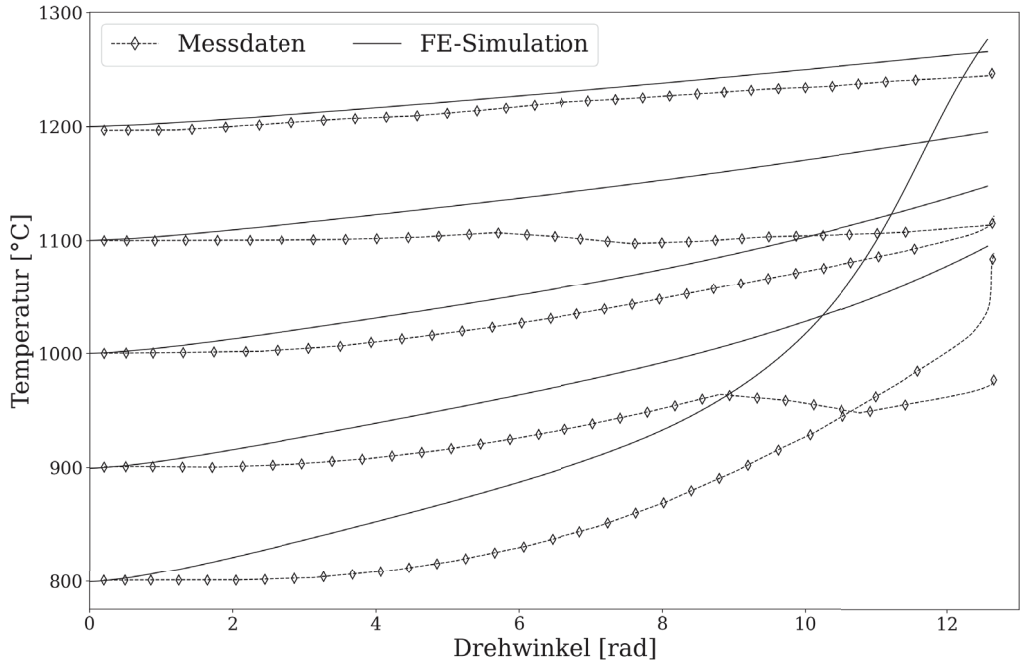


Abbildung 5.13: Vergleich zwischen simuliertem und gemessenem Temperaturanstieg T_1, bestimmt bei einer Vergleichsdehnrate von $10\,\mathrm{s}^{-1}$.

am Ende 30 °C.
Bei einer Starttemperatur von 900 °C war die experimentelle Temperaturmessung bis zu einer Drehung von 8,8 rad erfolgreich, danach versagte das Thermoelement. In der Simulation ist der Temperaturanstieg an jedem Punkt größer und beträgt 40 °C bei 8,8 rad.
Bei einer Starttemperatur von 800 °C zeigte sich dasselbe Verhalten wie bei den höheren Temperaturen. Die Simulation ergibt einen höheren Temperaturanstieg während der Verformung. Die Differenz zwischen Simulation und Messung ist während der Verformung kontinuierlich größer geworden und nimmt ab einer Drehung von 10 rad verstärkt zu. Die Temperaturdifferenz am Ende liegt bei 195 °C.

Zusammenfassung Temperaturanstieg

Bei der niedrigeren Dehnrate führte die inverse Analyse zu Ergebnissen, die sehr gut mit den experimentell bestimmten Werten übereinstimmen. Anders sieht das bei der höheren Dehnrate aus. Dort führten die Simulationen immer zu einem größeren Temperaturanstieg. Der experimentell bestimmte Anstieg war deutlich geringer, vor allem bei den niedrigeren Starttemperaturen. In Abschnitt 5.2.2 wurde schon die Zuverlässigkeit der Thermoelementmessungen diskutiert. Hier spielt vermutlich die Anspruchszeit der Thermoelemente eine wichtige Rolle. Der Temperaturanstieg in den experimentellen Messungen wird erst mit zeitlicher Verzögerung erfasst, wäh-

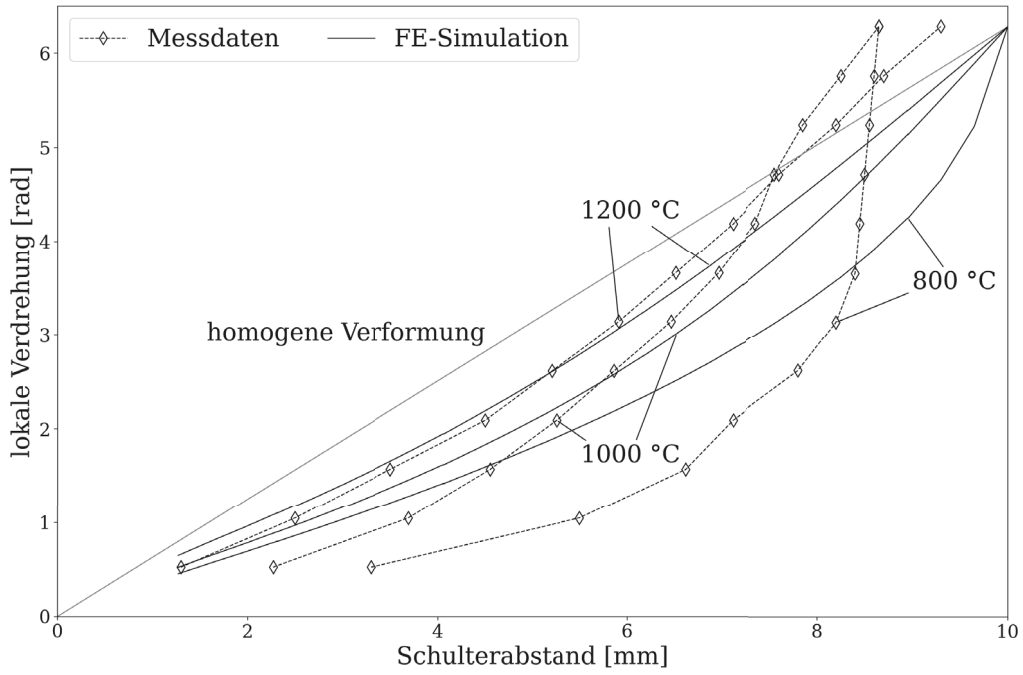


Abbildung 5.14: Vergleich zwischen simulierter und gemessener lokaler Verdrehung, bestimmt bei einer Vergleichsdehnrate von $10\,s^{-1}$.

rend die Simulationen diesen sofort widerspiegeln. Somit entsteht ein methodischer Fehler, dessen genaue Größe nicht bekannt ist.

Eine aussichtsreiche Möglichkeit bietet sich durch eine Präzisierung der Temperaturmessung an, denn dann könnten die Ergebnisse aus dem experimentellen Temperaturanstieg auch in der inversen Analyse berücksichtigt werden. Damit steigt die Eindeutigkeit der gefundenen Lösung. Das hat allerdings den Nachteil, dass die Messungen dann nicht mehr zur Validierung der gefundenen Parameter herangezogen werden können.

5.3.3 Lokale Verdrehung

Die zweite Möglichkeit zur Validierung der Ergebnisse aus der FE-Analyse bietet der Vergleich der lokalen Verdrehung. Mithilfe dieser können die Ergebnisse aus der Simulation beurteilt werden. In Abbildung 5.14 ist die lokale Verdrehung im Experiment und in der Simulation dargestellt. Die Werte stammen von den Starttemperaturen bei 800 °C, 1000 °C und 1200 °C und einer Vergleichsdehnrate von $10\,s^{-1}$. In dieser Darstellung lässt sich der lokale Umformgrad anhand der Steigung der Verläufe beurteilen. Zusätzlich können aus der Simulation auch die exakten Umformgrade bestimmt werden.

Bei einer Starttemperatur von 1200 °C zeigt die Simulation eine Lokalisierung

der Verformung. So liegt die Steigung der Verdrehung bis zu einem Schulterabstand von 4 mm unter jener der homogenen Verformung. Zwischen 4 mm bis 7 mm Schulterabstand liegt der lokale Umformgrad auf dem gleichen Niveau wie bei homogener Verdrehung. In der Probenmitte (Schulterabstand > 7 mm) ist die lokale Verdrehung größer als bei homogener Verformung. In der Simulation ergibt sich aus diesem Verlauf ein Umformgrad von 1,22 in einem Abstand von 2 mm und 2,45 in der Mitte der Torsionsprobe. Im Vergleich zum experimentellen Torsionsversuch ist die Verformungslokalisation in der Simulation geringfügig kleiner.
Bei einer Starttemperatur von 1000 °C ergibt die Simulation eine deutliche Verformungslokalisation. Am Rand der Prüflänge (Schulterabstand < 5 mm) liegt der lokale Umformgrad unter dem der homogenen Verformung. Zwischen 5 mm bis 7 mm ist die lokale Verdrehung sehr ähnlich zur der bei homogener Verdrehung, darüber kommt es zu stärkerer Verdrehung. Daraus ergibt sich ein Umformgrad von 1,03 in einem Abstand von 2 mm und 3,2 in der Probenmitte jeweils für die Probenoberfläche. Im Vergleich zum experimentellen Torsionsversuch ist die Verformungslokalisation in der Simulation kleiner.
Bei einer Starttemperatur von 800 °C ergibt die Simulation eine starke Verformungslokalisation. Bis zu einem Schulterabstand von 7 mm ist die lokale Umformung geringer als bei der homogenen Verdrehung. Im kleinen Bereich zwischen 7 mm bis 8 mm ist der lokale Umformgrad ungefähr auf gleicher Höhe wie bei der homogenen Verformung. Näher zur Probenmitte hin nimmt die lokale Verformung dann stark zu. Aus der Simulation ergibt sich an der Probenoberfläche ein Umformgrad von 0,94 am Abstand von 2 mm und 10,8 in der Probenmitte. Die experimentelle Messung liefert im Vergleich dazu eine noch größere Lokalisation der Verformung.

Bei allen drei Temperaturen haben die experimentellen Ergebnisse ein stärkere Lokalisierung der Verdrehung ergeben. Der Unterschied wurde dabei mit sinkender Temperatur größer, sodass die Abweichung bei 800 °C am größten war. Das deckt sich mit dem Entfestigungsverhalten im Drehmomentverlauf. Bei 1200 °C stimmt die Entfestigung zwischen dem simulierten und experimentellen Drehmomentverlauf gut überein, sodass die auftretende lokale Verdrehung sehr ähnlich ist. Bei 800 °C ist die Entfestigung im Experiment deutlich größer. Das zeigt sich auch in der größeren Verdrehungslokalisation im Vergleich zur Simulation.

5.3.4 Fließkurven

In den Ergebnissen des Drehmomentverlaufes, des Temperaturanstieges und der lokalen Verdrehung war die sehr gute Übereinstimmung zwischen den experimen-

tellen Messungen und den Simulationen zu sehen. Nur die sehr starke Entfestigung bei niedrigen Temperaturen und höherer Dehnrate konnte in der Simulation nicht gut erfasst werden. Eine teilweise Begründung hierfür liegt im Optimierungsprozess, denn dieser wurde nur mit Verdrehungen bis 10 rad gerechnet. Meist begannen die Torsionsproben aber erst ab diesem Winkel stark zu entfestigen. Um das besser zu berücksichtigen, müsste die komplette Verformung für den Optimierungsprozess herangezogen werden.
Der Hauptfokus in der Auswertung liegt aber in der Bestimmung der Fließkurven. Hierfür steht vor allem die Fließspannungsabhängigkeit von Dehnung und Dehnrate im Vordergrund. Die Temperaturabhängigkeit wurde schon über die fünf verschiedenen Starttemperaturen gut erfasst. Dadurch ist der Bereich, in denen die Torsionsproben stark lokalisieren, für die Fließkurvenbestimmung nur bedingt von Interesse. Dort wird das Werkstoffverhalten zusätzlich signifikant von der temperaturbedingten Entfestigung beeinflusst.

Im klassischen Torsionsversuch wird die Fließspannung durch Umrechnung des gemessenen Torsionsmoments bestimmt. Für die inverse FE-Analyse ist dies nicht notwendig, da als Ergebnis die sieben Parameter des Hensel-Spittel Fließkurvenmodells erhalten werden. Die Fließkurven für die bestimmten Parameter (siehe Tabelle 5.3) sind in Abbildung 5.15 dargestellt. Die Fließkurven zeigen eine generelle Entfestigung bei höheren Dehnungen. Diese ist bei den niedrigen Temperaturen stärker ausgeprägt. Die Fließkurven bei 1100 °C und 1200 °C bleiben nach Erreichen ihres Spannungsmaximums auf nahezu konstanter Höhe.

Vergleich Zylinderstauchversuch

Um die Fließkurven aus der inversen Analyse beurteilen zu können, wurden diese mit Ergebnissen aus Zylinderstauchversuchen verglichen. Dazu waren aus vorherigen Messungen [65] Daten für das Versuchsmaterial vorhanden. Zusätzlich wurden Werte aus der Literatur [66] für einen AISI 316L hinzugezogen. Der Vergleich ist in Abbildung 5.16 dargestellt.

Die Fließkurven bei einer Temperatur von 900 °C bis 1200 °C und Dehnrate $1\,s^{-1}$, sowie bei einer Temperatur von 1000 °C bis 1200 °C und Dehnrate $10\,s^{-1}$ zeigen eine gute Übereinstimmung zu den experimentellen Daten aus den Stauchversuchen. Bei 1100 °C und 1200 °C ergibt der Torsionsversuch tendenziell eine etwas höhere Fließspannung. Vor allem bei der niedrigeren Vergleichsdehnrate von $1\,s^{-1}$ ist das ersichtlich. Bei 1000 °C sind die Ergebnisse nahezu ident. Hier ist die Übereinstimmung außerordentlich gut. Das trifft auch auf die Fließkurve bei 900 °C und Vergleichsdehn-

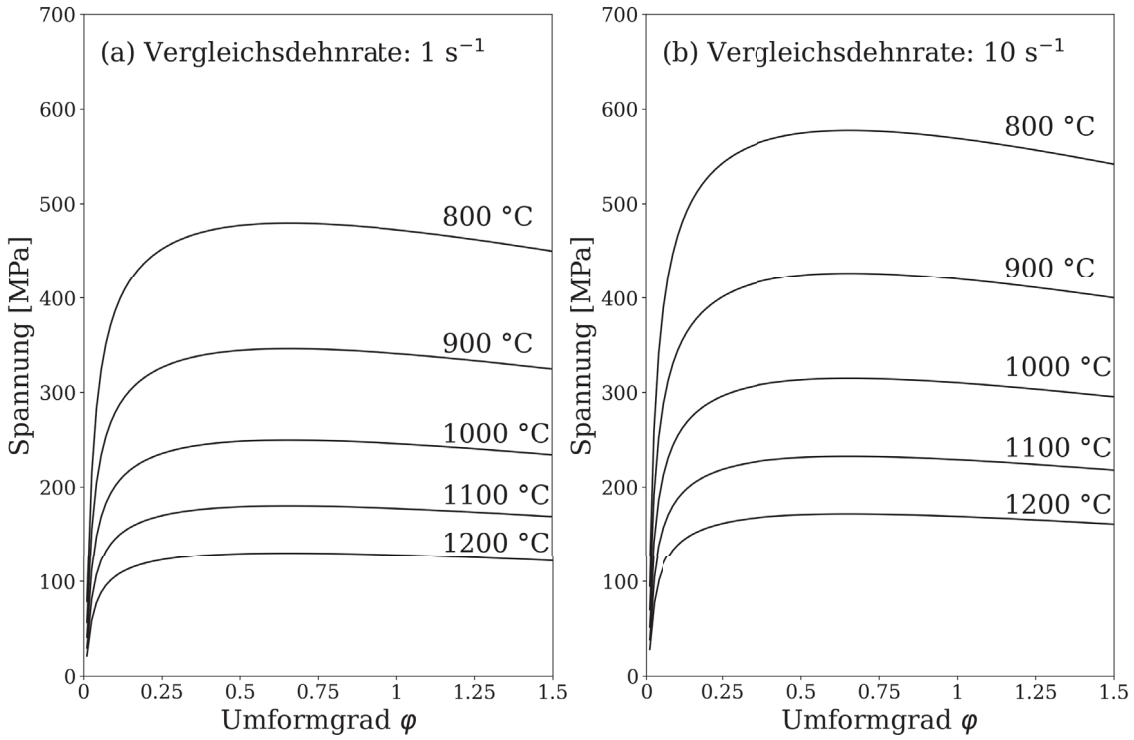


Abbildung 5.15: Fließkurven, bestimmt aus dem Torsionsversuch mittels inverser FE-Analyse.

rate $1\,s^{-1}$ zu. Bei 900 °C und Vergleichsdehnrate $10\,s^{-1}$ wird im Torsionsversuch eine höhere Festigkeit als im Stauchversuch gemessen. Zu großen Abweichungen kommt es bei einer Temperatur von 800 °C. Hier stimmen die Ergebnisse nicht überein. Das war bereits bei den experimentellen Drehmomentkurven zu sehen (Abbildungen 5.10 & 5.11). In diesem Bereich sagt das Hensel-Spittel-Modell eine größere Fließspannung voraus, als das experimentell bestätigt wurde. Das Materialmodell stößt an seine Grenzen. Es ist nicht in der Lage den gesamten Temperaturbereich von 800 °C bis 1200 °C präzise abzubilden. Ein Modell mit mehr Einflussfaktoren könnte hier bessere Ergebnisse liefern. Allerdings muss dabei beachtet werden, dass jeder weitere Parameter im Materialmodell zu einem stark erhöhten Rechenaufwand in der inversen FE-Analyse führt.

Zu den Ergebnissen muss erwähnt werden, dass diese auf der Annahme der Gestaltänderungshypothese nach Huber- v. Mises basieren.

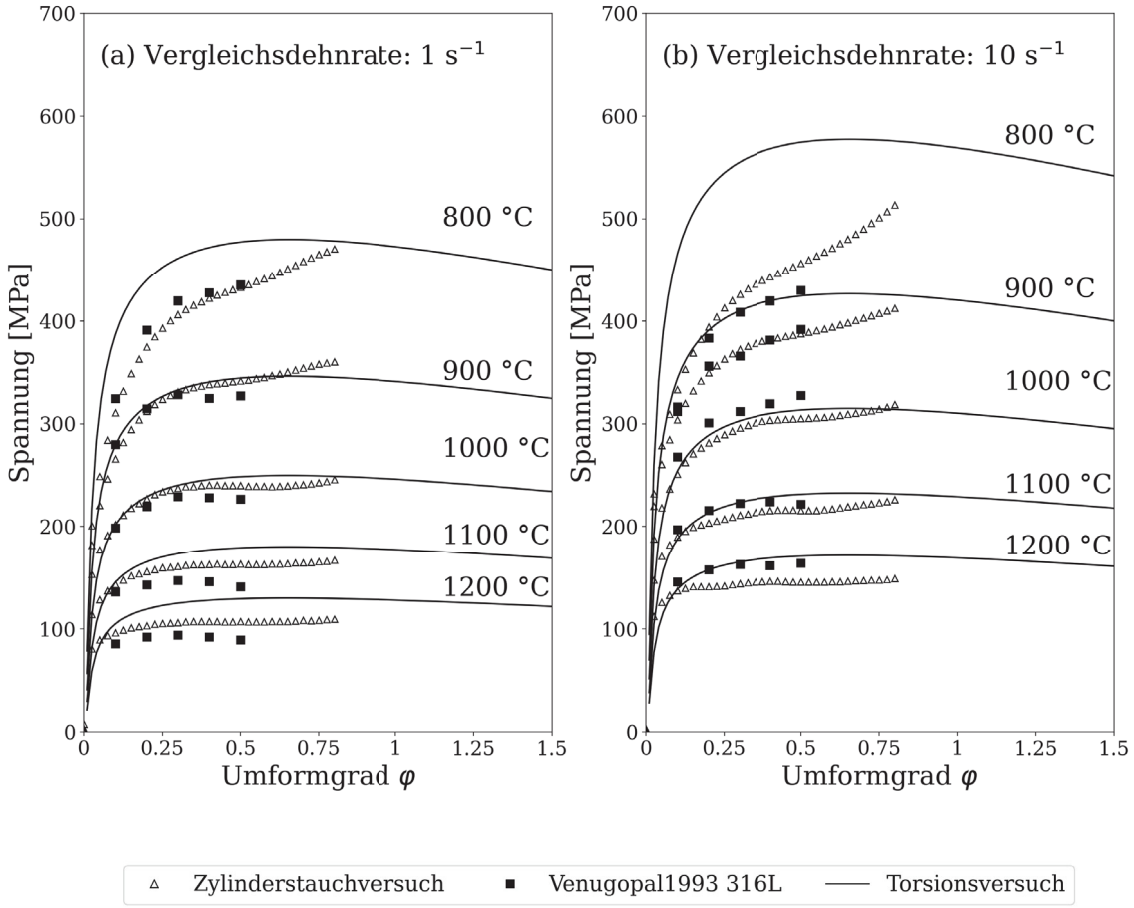


Abbildung 5.16: Fließkurven bestimmt aus dem Torsionsversuch mittels inverser FE-Analyse im Vergleich zu Ergebnissen aus dem Zylinderstauchversuch [65, 66].

6 Zusammenfassung & Ausblick

Aus den Versuchsergebnissen wurde klar, dass es unter bestimmten Bedingungen zu einer starken Lokalisation der Verformung kommt. Die klassische analytische Auswertung war dadurch nicht mehr zulässig. Die lokale Instabilität wurde durch eine niedrige Starttemperatur und eine hohe Verformungsgeschwindigkeit besonders forciert. Zur Quantifizierung der lokalen Verdrehung wurde die Verformung einer Linie auf der Probenoberfläche verfolgt und vermessen. In der Art und Aufbringung dieser Markierung liegt noch Potential zur Verbesserung.

Eine weitere Erkenntnis aus den Versuchen war die Störungsanfälligkeit der Temperaturmessung mittels Thermoelement. Durch die zwangsweise Platzierung inmitten der Verformungszone kann es schnell zu Kontaktproblemen kommen. Zusätzlich hat sich gezeigt, dass die Ansprechzeit der Thermoelemente zu gering war, um die Versuche bei der Vergleichsdehnrate $10\,\mathrm{s}^{-1}$ zuverlässig zu detektieren. Eine alternative Temperaturmessung könnte hier eine deutliche Verbesserung bringen.

Im Rahmen der Auswertung wurde ein thermisch-elektrisches FE-Modell erstellt, mit dem die Temperaturverteilung in der Torsionsprobe exakt vorhergesagt werden kann. Dieses war in seiner endgültigen Form komplett unabhängig von experimentellen Daten und beruht auf rein physikalischen Interaktionen. Bei der Validierung zeigte sich, dass dieses Modell in der Lage ist, die Temperatur im für die Verformung relevanten Bereich mit einem maximalen Fehler von $4\,^{\circ}\mathrm{C}$ zu bestimmen. Sowohl für stationäre als auch für transiente Wärmebehandlungen ist das thermisch-elektrische FE-Modell gleichermaßen gut geeignet.

Für die Modellierung der Verformung wurde ein thermisch-mechanisches FE-Modell entwickelt. Über eine inverse FE-Analyse konnten die Parameter des Materialmodells erfolgreich bestimmt werden. Das hat die sehr gute Übereinstimmung zu den experimentellen Versuchen gezeigt. Für die Verläufe von Drehmoment über Drehwinkel war das zu erwarten, da diese als Grundlage für den Optimierungsprozess dienten. Jedoch auch die experimentellen Ergebnisse des Temperaturanstiegs und der lokalen Verdrehung zeigten hohe Ähnlichkeit zu den Ergebnissen der Simulation. Da diese nicht in die Optimierung integriert wurden, konnten die gefundenen Parameter dadurch erfolgreich validiert werden.

Das auftretende Optimierungsproblem in der inversen Formulierung wurde er-

folgreich mittels Nelder-Mead-Algorithmus gelöst. Dieser führt zwangsweise nur zu einem lokalen Optimum. Wünschenswert für die Auswertung wäre ein Algorithmus, der in Lage ist, nach einem globalen Optimum zu suchen. Zum derzeitigen Stand war dies aber nicht mit den vorhanden Rechenkapazitäten vereinbar.

Im abschließenden Vergleich mit Fließkurvendaten aus der Literatur hat sich gezeigt, dass die Auswertung über den Heißtorsionsversuch zu vergleichbaren Werten führt. Dies beruht auf der Annahme, dass die Spannungszustände über die Gestaltänderungshypothese nach von Mises vergleichbar sind.

Besonders durch das in dieser Arbeit entwickelte thermisch-elektrische FE-Modell konnte einen besserer Einblick in den Heißtorsionsversuch gewonnen werden. Ausgehend von diesem wäre eine Analyse und Variation der Probengeometrie, mit dem Ziel der Minimierung des Temperaturgradienten, sehr interessant.

Weiterer Forschungsbedarf besteht noch bei der Analyse des Spannungszustandes im Heißtorsionsversuch. Die Beschreibung durch die von Mises Vergleichsspannung ist möglicherweise nicht immer zulässig. Das gewinnt besonders an Bedeutung, wenn die Ergebnisse aus dem Heißtorsionsversuch mit anderen Verfahren verglichen werden sollen.

Der in dieser Arbeit verwendete Ansatz der inversen Analyse ist durch seine Flexibilität beliebig auf andere Werkstoffe erweiterbar. Dazu müssen einzig die thermophysikalischen, sowie elektrischen Eigenschaften des konkreten Werkstoffes bekannt sein. Aus der inversen Analyse können dann die Materialparameter bestimmt werden. Auch für andere Prüfverfahren eignet sich dieser Ansatz, dann muss jedoch das zugrunde liegende FE-Modell dementsprechend modifiziert werden.

Literaturverzeichnis

[1] Y. Lin and X.-M. Chen, "A critical review of experimental results and constitutive descriptions for metals and alloys in hot working," *Materials & Design*, vol. 32, pp. 1733–1759, apr 2011.

[2] E. Doege and B.-A. Behrens, *Handbuch Umformtechnik*. VDI-Buch, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[3] G. E. Dieter, H. a. Kuhn, and S. L. Semiatin, eds., *Handbook of Workability and Process Design*. Materials Park, OH: ASM international, 2003.

[4] D. S. Fields and W. A. Backofen, "Determination of strain hardening characteristics by torsion testing," *Proceedings of the ASTM*, vol. 57, pp. 1259–1272, 1957.

[5] T. Sheppard and D. S. Wright, "Determination of flow stress: Part 1 constitutive equation for aluminium alloys at elevated temperatures," *Metals Technology*, vol. 6, pp. 215–223, jan 1979.

[6] E. Lach and K. Pöhlandt, "Testing the plastic behaviour of metals by torsion of solid and tubular specimens," *Journal of Mechanical Working Technology*, vol. 9, pp. 67–80, feb 1984.

[7] A. Gavrus, E. Massoni, and J. Chenot, "An inverse analysis using a finite element model for identification of rheological parameters," *Journal of Materials Processing Technology*, vol. 60, pp. 447–454, jun 1996.

[8] R. Forestier, E. Massoni, and Y. Chastel, "Estimation of constitutive parameters using an inverse method coupled to a 3D finite element software," *Journal of Materials Processing Technology*, vol. 125-126, pp. 594–601, sep 2002.

[9] H. Aguir, H. BelHadjSalah, and R. Hambli, "Parameter identification of an elasto-plastic behaviour using artificial neural networks–genetic algorithm method," *Materials & Design*, vol. 32, pp. 48–53, jan 2011.

[10] S. Khoddam, P. Hodgson, and M. Jafari Bahramabadi, "An inverse thermal–mechanical analysis of the hot torsion test for calibrating the constitutive parameters," *Materials & Design*, vol. 32, pp. 1903–1909, apr 2011.

[11] C. Zhang, M. Bellet, M. Bobadilla, H. Shen, and B. Liu, "Inverse finite element modelling and identification of constitutive parameters of UHS steel based on Gleeble tensile tests at high temperature," *Inverse Problems in Science and Engineering*, vol. 19, pp. 485–508, jun 2011.

[12] R. Hill, "A theory of the yielding and plastic flow of anisotropic metals," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 193, pp. 281–297, may 1948.

[13] R. Hill, "Constitutive modelling of orthotropic plasticity in sheet metals," *Journal of the Mechanics and Physics of Solids*, vol. 38, pp. 405–417, jan 1990.

[14] F. Barlat, R. C. Becker, Y. Hayashida, Y. Maeda, M. Yanagawa, K. Chung, J. C. Brem, D. J. Lege, K. Matsui, S. J. Murtha, and S. Hattori, "Yielding description for solution strengthened aluminum alloys," *International Journal of Plasticity*, vol. 13, no. 4, pp. 385–401, 1997.

[15] H. S. Valberg, *Applied Metal Forming*. Cambridge: Cambridge University Press, 2010.

[16] D. Banabic, H. Bunge, K. Pöhlandt, and A. E. Tekkaya, *Formability of Metallic Materials*. Engineering Materials, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[17] M. J. Michno and W. N. Findley, "An historical perspective of yield surface investigations for metals," *International Journal of Non-Linear Mechanics*, vol. 11, no. 1, pp. 59–82, 1976.

[18] F. J. Humphreys, G. S. Rohrer, and A. Rollett, *Recrystallization and Related Annealing Phenomena (Third Edition)*. Amsterdam, Oxford, Cambridge: Elsevier, third ed., 2017.

[19] J. H. Beynon and C. M. Sellars, "Modelling Microstructure and Its Effects during Multipass Hot Rolling.," *ISIJ International*, vol. 32, no. 3, pp. 359–367, 1992.

[20] R. Petković, M. Luton, and J. Jonas, "Recovery and recrystallization of carbon steel between intervals of hot working," *Canadian Metallurgical Quarterly*, vol. 14, pp. 137–145, apr 1975.

[21] G. R. Johnson and W. H. Cook, "A Computational Constitutive Model and Data for Metals Subjected to Large Strain, High Strain Rates and High Pres-

sures," in *Proceedings 7th International Symposium on Ballistics*, pp. 541–547, 1983.

[22] H. Mecking and U. Kocks, "Kinetics of flow and strain-hardening," *Acta Metallurgica*, vol. 29, pp. 1865–1875, nov 1981.

[23] A. Hensel and T. Spittel, *Kraft- und Arbeitsbedarf bildsamer Formgebungsverfahren*. Leipzig: VEB Deutscher Verlag für Grundstoffindustrie Leipzig, 1. auflage ed., 1978.

[24] F. J. Zerilli and R. W. Armstrong, "Dislocation-mechanics-based constitutive relations for material dynamics calculations," *Journal of Applied Physics*, vol. 61, pp. 1816–1825, mar 1987.

[25] R. Goetz and V. Seetharaman, "Modeling Dynamic Recrystallization Using Cellular Automata," *Scripta Materialia*, vol. 38, pp. 405–413, jan 1998.

[26] Y. Lin, J. Zhang, and J. Zhong, "Application of neural networks to predict the elevated temperature flow behavior of a low alloy steel," *Computational Materials Science*, vol. 43, pp. 752–758, oct 2008.

[27] S. Mandal, P. Sivaprasad, S. Venugopal, and K. Murthy, "Artificial neural network modeling to evaluate and predict the deformation behavior of stainless steel type AISI 304L during hot torsion," *Applied Soft Computing*, vol. 9, pp. 237–244, jan 2009.

[28] Y. Sun, W. Zeng, Y. Zhao, Y. Qi, X. Ma, and Y. Han, "Development of constitutive relationship model of Ti600 alloy using artificial neural network," *Computational Materials Science*, vol. 48, pp. 686–691, may 2010.

[29] E. Pestel and J. Wittenburg, *Technische Mechanik, Band 2: Festigkeitslehre*. Mannheim, Leipzig, Wien, Zürich: BI Wissenschaftsverlag, 2. ed., 1992.

[30] C. Weber and W. Günther, *Torsionstheorie*, vol. 4. Wiesbaden: Viweg+Teubner Verlag, 1958.

[31] H.-P. Stüwe and H. Turck, "Zur Messung von Fließkurven im Torsionsversuch," *International Journal of Materials Research*, vol. 55, pp. 699–703, nov 1964.

[32] S. Khoddam, Y. Lam, and P. Thomson, "Thermal analysis of the hot torsion test taking into account radiation," *Mechanics of Materials*, vol. 22, pp. 1–9, jan 1996.

[33] M. Zhou and M. P. Clode, "Thermal analysis of the torsion test under hot-working conditions," *Computational Materials Science*, vol. 9, pp. 411–419, jan 1998.

[34] B. Mirzakhani, S. Khoddam, H. Arabi, M. T. Salehi, S. H. Seyedein, and M. R. Aboutalebi, "Influence of specimen geometry of hot torsion test on temperature distribution during reheating treatment of API-X70," *Journal of Iron and Steel Research International*, vol. 17, pp. 34–39, mar 2010.

[35] S. L. Semiatin, D. W. Mahaffey, N. C. Levkulich, and O. N. Senkov, "The Radial Temperature Gradient in the Gleeble® Hot-Torsion Test and Its Effect on the Interpretation of Plastic-Flow Behavior," *Metallurgical and Materials Transactions A*, vol. 48, pp. 5357–5367, nov 2017.

[36] S. L. Semiatin, D. W. Mahaffey, D. J. Tung, W. Zhang, and O. N. Senkov, "A Comparison of the Plastic-Flow Response of a Powder-Metallurgy Nickel-Base Superalloy Under Nominally-Isothermal and Transient-Heating Hot-Working Conditions," *Metallurgical and Materials Transactions A*, vol. 48, pp. 1864–1879, apr 2017.

[37] R. C. Aster, B. Borchers, and C. H. Thurber, *Parameter estimation and inverse problems*. Amsterdam, Oxford, Cambridge: Elsevier, 3. ed., 2019.

[38] K.-J. Bathe, *Finite-Elemente-Methoden*. Berlin, Heidelberg, New York: Springer-Verlag Berlin Heidelberg, 2. ed., 2002.

[39] O. Zienkiewicz, R. Taylor, and J. Z. Zhu, *The Finite Element Method: its Basis and Fundamentals*. Oxford: Butterworth-Heinemann, 2013.

[40] O. Zienkiewicz, R. Taylor, and D. Fox, "The Finite Element Method for Solid and Structural Mechanics," in *The Finite Element Method for Solid and Structural Mechanics*, Oxford: Butterworth-Heinemann, 7. ed., 2014.

[41] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, pp. 308–313, jan 1965.

[42] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions," *SIAM Journal on Optimization*, vol. 9, pp. 112–147, jan 1998.

[43] F. Gao and L. Han, "Implementing the Nelder-Mead simplex algorithm with adaptive parameters," *Computational Optimization and Applications*, vol. 51, pp. 259–277, jan 2012.

[44] J. E. Dennis, Jr. and V. Torczon, "Direct Search Methods on Parallel Machines," *SIAM Journal on Optimization*, vol. 1, pp. 448–474, nov 1991.

[45] C. T. Kelley, "Detection and Remediation of Stagnation in the Nelder–Mead Algorithm Using a Sufficient Decrease Condition," *SIAM Journal on Optimization*, vol. 10, pp. 43–55, jan 1999.

[46] C. Price, I. Coope, and D. Byatt, "A Convergent Variant of the Nelder–Mead Algorithm," *Journal of Optimization Theory and Applications*, vol. 113, pp. 5–19, apr 2002.

[47] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar, "A review of PID control, tuning methods and applications," *International Journal of Dynamics and Control*, vol. 9, no. 2, pp. 818–827, 2020.

[48] J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers," *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, pp. 220–222, jun 1993.

[49] C. Zhang, M. Bellet, M. Bobadilla, H. Shen, and B. Liu, "A Coupled Electrical–Thermal–Mechanical Modeling of Gleeble Tensile Tests for Ultra-High-Strength (UHS) Steel at a High Temperature," *Metallurgical and Materials Transactions A*, vol. 41, pp. 2304–2317, jun 2010.

[50] D. R. Forrest and M. F. Sinfield, "Numerical Simulation of Gleeble Torsion Testing of HSLA-65 Steel," 2008.

[51] E. B. da Fonseca, T. F. A. Santos, S. T. Button, and A. J. Ramirez, "Physical Simulation of a Duplex Stainless Steel Friction Stir Welding by the Numerical and Experimental Analysis of Hot Torsion Tests," *Metallurgical and Materials Transactions A*, vol. 47, pp. 4543–4552, sep 2016.

[52] S. D. Norris and I. Wilson, "Application of 3D numerical modelling for thermal profile optimization on the Gleeble thermomechanical simulator," *Modelling and Simulation in Materials Science and Engineering*, vol. 7, pp. 297–309, may 1999.

[53] E. Kardoulaki, J. Lin, D. Balint, and D. Farrugia, "Investigation of the effects of thermal gradients present in Gleeble high-temperature tensile tests on the strain state for free cutting steel," *The Journal of Strain Analysis for Engineering Design*, vol. 49, pp. 521–532, oct 2014.

[54] J. M. P. Martins, J. L. Alves, D. M. Neto, M. C. Oliveira, and L. F. Menezes, "Thermal Gradients Prediction in the Gleeble System Using a 3D Finite Element Model," 2015.

[55] M. Zhou and M. Clode, "A finite element analysis for the least temperature rise in a hot torsion test specimen," *Finite Elements in Analysis and Design*, vol. 31, pp. 1–14, nov 1998.

[56] B. Mirzakhani, S. Khoddam, H. Arabi, M. T. Salehi, and J. Sietsma, "A coupled thermal-mechanical FE model of flow localization during the hot torsion test," *Steel research international*, vol. 80, no. 11, pp. 846–854, 2009.

[57] M. Spittel and T. Spittel, "Steel symbol/number: X2CrNiMo19–12/1.4430," in *Landolt-Börnstein - Group VIII Advanced Materials and Technologies 2C1* (H. Warlimont, ed.), pp. 678–683, Springer-Verlag Berlin Heidelberg, 2009.

[58] M. Spittel and T. Spittel, "Steel symbol/number: X2CrNiMo17–12–2/1.4404," in *Landolt-Börnstein - Group VIII Advanced Materials and Technologies 2C1* (H. Warlimont, ed.), pp. 654–659, Springer-Verlag Berlin Heidelberg, 2009.

[59] P. Haslberger, "Private Kommunikation 01.06.2021, voestalpine Forschungsservicegesellschaft Donawitz GmbH."

[60] S. Kobayashi, S.-I. Oh, and T. Altan, *Metal Forming and the Finite-Element Method*. Oxford University Press, may 1989.

[61] A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures.," *Analytical Chemistry*, vol. 36, pp. 1627–1639, jul 1964.

[62] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization An overview," *Swarm Intelligence*, pp. 33–57, 2007.

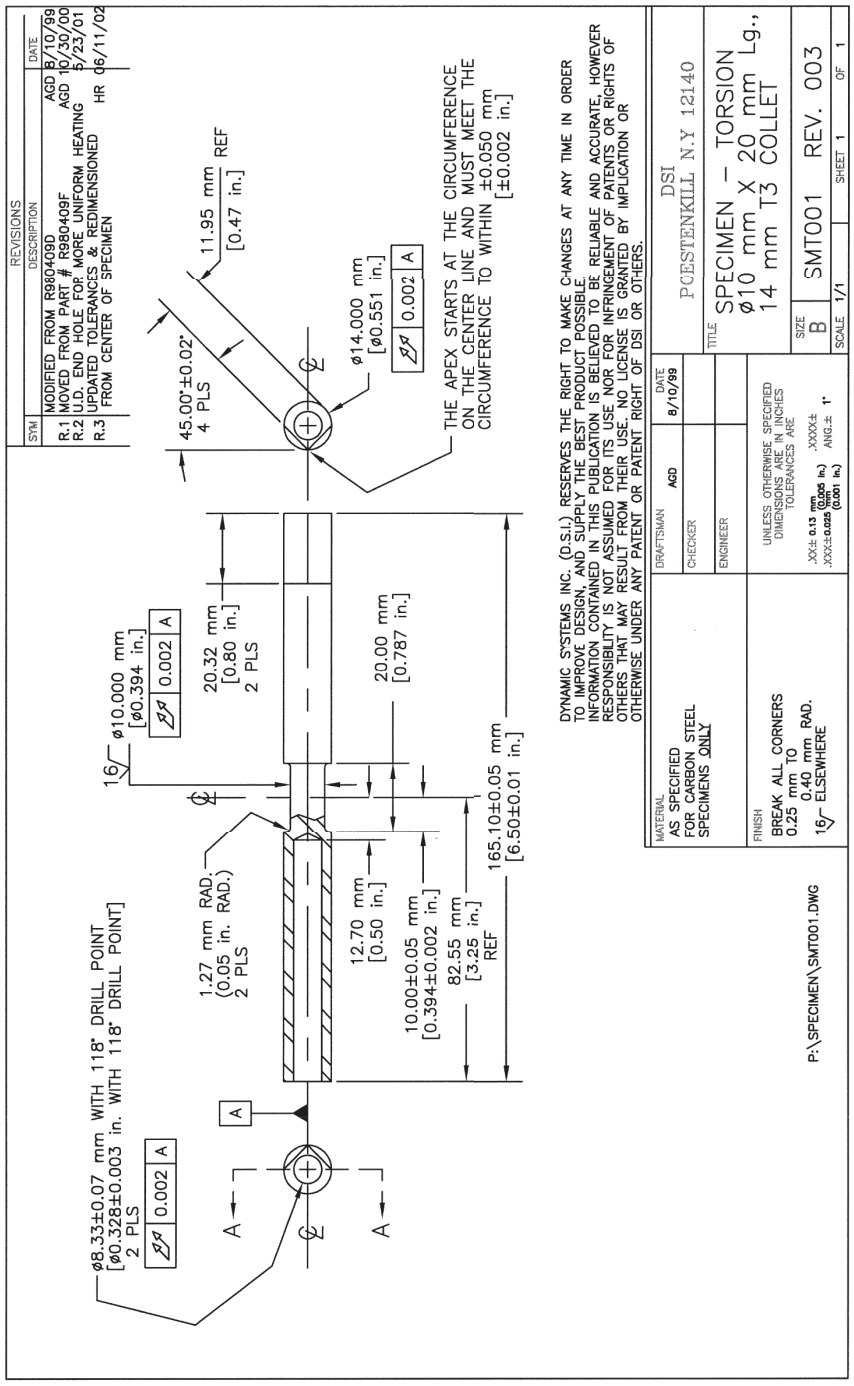
[63] A. Oliveira, A. Avrit, and M. Gradeck, "Thermocouple response time estimation and temperature signal correction for an accurate heat flux calculation in inverse heat conduction problems," *International Journal of Heat and Mass Transfer*, vol. 185, p. 122398, apr 2022.

[64] M. Spittel and T. Spittel, "Steel symbol/number: X2CrNiMoN17–13–5/1.4439," in *Landolt-Börnstein - Group VIII Advanced Materials and Technologies 2C1* (H. Warlimont, ed.), pp. 684–689, 2009.

[65] P. Haslberger, "Private Kommunikation 08.10.2021, voestalpine Forschungsservicegesellschaft Donawitz GmbH."

[66] S. Venugopal, S. L. Mannan, and Y. V. R. K. Prasad, "Processing map for hot working of stainless steel type AISI 316L," *Materials Science and Technology*, vol. 9, pp. 899–906, oct 1993.

Anhang A Probengeometrie



Anhang B Subroutinen Abaqus

B.1 UHARD

```
      SUBROUTINE UHARD(SYIELD,HARD,EQPLAS,EQPLASRT,TIME,DTIME,TEMP,
     1 DTEMP,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,CMNAME,NSTATV,
     2 STATEV,NUMFIELDV,PREDEF,DPRED,NUMPROPS,PROPS)
C
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*80 CMNAME
      DIMENSION HARD(3),STATEV(NSTATV),TIME(*),
     $ PREDEF(NUMFIELDV),DPRED(*),PROPS(*)
      real(8) :: sigma, phi, phidot, T, A, m1, m2, m4, m5, m7, m8
      A=PROPS(1)
      m1=PROPS(2)
      m2=PROPS(3)
      m4=PROPS(4)
      m5=PROPS(5)
      m7=PROPS(6)
      m8=PROPS(7)


C     user coding to define SYIELD,HARD(1),HARD(2),HARD(3)
C     write(7, *) EQPLAS, EQPLASRT

      T=TEMP+DTEMP

      phi=EQPLAS+1d-2

      IF (EQPLASRT.le.1e-4) then
          phidot=1e-4
      ELSE
          phidot=EQPLASRT
      END IF

      IF(phi.ge.1.5) then
          phi=1.5d0
      END IF
```

```
36
37
38          SYIELD=A*EXP(m1*T)*phi**m2*EXP(m4/phi)*
39         $     (1+phi)**(m5*T)*EXP(m7*phi)*phidot**(m8*T)*1d6
40
41          HARD(1)=
42         $     A*phi**(m2-2)*phidot**(m8*T)*(phi+1)**(m5*T-1)*
43         $     EXP(m1*T+m7*phi+m4/phi)*(m2*phi**2+m2*phi-m4*phi-
44         $     m4+m5*phi**2*T+m7*phi**3+m7*phi**2)*1d6
45
46          IF(phi.ge.1.5) then
47              HARD(1)=0
48          END IF
49
50          HARD(2)=
51         $     A*(m8*T)*EXP(m1*T)*phi**m2*EXP(m4/phi)*
52         $     (1+phi)**(m5*T)*EXP(m7*phi)*phidot**(m8*T-1)
53         $     *1d6
54
55          HARD(3)=
56         $     A*phi**m2*phidot**(m8*T)*(phi+1)**(m5*T)*
57         $     EXP(m1*T+m7*phi+m4/phi)*
58         $     (m1+m8*LOG(phidot)+m5*LOG(phi+1))*1d6
59
60 c        write(7, *) HARD(1), HARD(2), HARD(3)
61
62
63          RETURN
64          END
```

B.2 UAMP

```
1  C     UAMP_Abaqus - Temperaturecontrol for simulations
2  C
3  C     Alexander Wenda 2021
4  C     <alexander-gerald.wenda@stud.unileoben.ac.at>
5  C
6  C     UAMP Subroutine to control the current in a thermoelectric
7  C     simulation. The control algorithm is PID based and modifies
8  C     the current in the sample according to a temperature setpoint.
9  C
10          Subroutine UAMP(
```

```
C           passed in for information and state variables
      *     ampName, time, ampValueOld, dt, nProps, props, nSvars,
      *     svars, lFlagsInfo,
      *     nSensor, sensorValues, sensorNames,
      *     jSensorLookUpTable,
C           to be defined
      *     ampValueNew,
      *     lFlagsDefine,
      *     AmpDerivative, AmpSecDerivative, AmpIncIntegral,
      *     AmpDoubleIntegral)

      include 'aba_param.inc'

C     svars - additional state variables, similar to (V)UEL
      dimension sensorValues(nSensor), svars(nSvars),
     * props(nProps)
      character*80 sensorNames(nSensor)
      character*80 ampName

C     time indices
      parameter( iStepTime          = 1,
     *           iTotalTime         = 2,
     *           nTime              = 2)
C     flags passed in for information
      parameter( iInitialization    = 1,
     *           iRegularInc        = 2,
     *           iCuts              = 3,
     *           ikStep             = 4,
     *           nFlagsInfo         = 4)
C     optional flags to be defined
      parameter( iComputeDeriv          = 1,
     *           iComputeSecDeriv       = 2,
     *           iComputeInteg          = 3,
     *           iComputeDoubleInteg    = 4,
     *           iStopAnalysis          = 5,
     *           iConcludeStep      = 6,
     *           nFlagsDefine           = 6)

      parameter(initCurrent=1d6, K=1.2d6, Ti=3.92d-1, Td=0d0)

      dimension time(nTime), lFlagsInfo(nFlagsInfo),
     *          lFlagsDefine(nFlagsDefine)
      dimension jSensorLookUpTable(*)

      real e, e1, kp, ki, kd, delta_I, T_set, T_def, esum
```

```
56        real Pout, Iout, Dout
57
58        lFlagsDefine(iComputeDeriv)        = 0
59        lFlagsDefine(iComputeSecDeriv)     = 0
60        lFlagsDefine(iComputeInteg)        = 0
61        lFlagsDefine(iComputeDoubleInteg) = 0
62
63 C      get sensor value
64        center_temp = GetSensorValue('TEMP_CONTROL',
65       *                             jSensorLookUpTable,
66       *                             sensorValues)
67        tme = time(iTotalTime)
68        T_def = 1200d0
69
70        if (ampName(1:11) .eq. 'AMP_CURRENT' ) then
71
72           if (lFlagsInfo(iInitialization).eq.1) then
73              ampValueNew = initCurrent
74              svars(1)=0
75              svars(2)=0
76 C            write(7,*) 'The initial current is:', ampValueNew
77
78           else
79              if (tme.le.240) then
80                 T_set = 5d0*tme
81              else if ((tme.ge.240).and.(tme.le.540)) then
82                 T_set = 1200
83              else
84                 T_set = T_def
85                 if (center_temp.le.T_set) then
86                    lFlagsDefine(iConcludeStep)=1
87                 end if
88              end if
89
90 C            Implementation of a PID controller
91              e = T_set-center_temp
92              e1 = svars(1)
93              esum = svars(2)
94
95              Pout = K*e
96
97              esum = esum+e*dt
98              Iout = K/Ti*esum
99
100             Dout=K*Td*(e-e1)/dt
```

```
101
102               ampValueNew=Pout+Iout+Dout
103
104 C             Zero esum when e changes sign
105 C             Only while current is out of bounds
106               if (ampValueNew.le.0) then
107                  ampValueNew=0d0
108                  if (e*e1.le.0) then
109                     esum=0
110                  end if
111               else if (ampValueNew.ge.1d8) then
112                  ampValueNew=1d8
113                  if (e*e1.le.0) then
114                     esum=0
115                  end if
116               end if
117               svars(1)=e
118               svars(2)=esum
119 C              write(7,*) T_set, center_temp
120 C              write(7,*) e, e1, esum
121 C              write(7,*) 'The changed current is:', ampValueNew,
122 C     *         'Old:', ampValueOld
123            end if
124         end if
125
126         return
127         end
```

Anhang C FE Modell Abaqus

```python
# -*- coding: utf-8

"""
Written by: Alexander Wenda 2021-2022
E-mail: alexander@wenda.at
"""
import sys

from abaqus import *
from abaqusConstants import *
from caeModules import *
import part

import math
import os
import json

mm = 1e-3


class SMT001Base(object):
    """
    A class to generate the basics for a hot torsion test. This model only contains information like
    parts and material properties for the simulation. This class acts as the parent class for further
    models including electrical, thermal and mechanical behaviour as well as their combinations.
    It also includes all relevant material properties.

    In the basic model the SMT001 specimen and the Gleeble torsion clamps are modelled. Any necessary sets/surface
    are already defined for the parts. Depending on the simulation different meshes and element are required.
```

```
30      The class supports these changes via optional keywords.
31
32      Material properties are read from a json-file in the same
                                    directory as the script with the
                                    property names
33      equal to their implementation in Abaqus. If there is no file
                                    present an error will be raised!
34
35      :param:
36      material_data: path to a json with material data for the
                                    simulation -> str (Path to
                                    material file)
37      user_subroutine: possible subroutine for the simulation -> str
                                    (Path to subroutine)
38
39      """
40
41      def __init__(self, material_data="",  user_subroutine="", **
                                    kwargs):
42
43          if not material_data:
44              material_data = os.path.join(os.path.dirname(__file__),
                                     "..", "src", "materials.json")
45
46          if not os.path.isfile(material_data):
47              raise WindowsError("Material file not found! Specify a
                                    correct path!")
48          else:
49              with open(material_data, 'r') as f:
50                  self.material_data = json.load(f)
51
52          self.user_subroutine = user_subroutine
53          self.ambientT = 25
54
55          self.model = None
56          self.model_name = 'Torsion-Simulation'
57
58          self.assembly = None
59
60          self.sketch_specimen = None
61          self.sketch_clamp = None
62
63          self.part_specimen = None
64          self.part_specimen_instance = None
65          self.part_specimen_instance_name = 'SMT001'
```

```
66          self.part_specimen_partitions = []
67
68          self.part_clamp = None
69          self.part_clamp_instance = None
70          self.part_clamp_instance_name = 'Clamp'
71
72          self.material_specimen = None
73          self.material_specimen_name = 'Steel'
74          self.material_clamp = None
75          self.material_clamp_name = 'Copper'
76
77          self.interaction_contact = None
78          self.interaction_contact_name = 'Clamp_Contact'
79          self.interaction_contact_property = None
80          self.interaction_contact_property_name = 'Clamp_Properties'
81
82          # Sets for the loads/boundary/symmetry conditions
83          # Specimen sets
84          self.set_specimen_symR = None
85          self.set_specimen_symR_name = 'Set_Symmetry_R'
86          self.set_specimen_symZ = None
87          self.set_specimen_symZ_name = 'Set_Symmetry_Z'
88          self.set_specimen_clamping = None
89          self.set_specimen_clamping_name = 'Set_Clamping'
90          self.set_specimen_temp_control = None
91          self.set_specimen_temp_control_name = "TEMP_NODE"
92          self.set_specimen_face = None
93          self.set_specimen_face_name = 'Set_Total'
94
95          # Clamp sets
96          self.set_clamp_ground = None
97          self.set_clamp_ground_name = 'GROUND'
98          self.set_clamp_clamping = None
99          self.set_clamp_clamping_name = 'Set_Clamping'
100         self.set_clamp_face = None
101         self.set_clamp_face_name = 'Set_Total'
102
103         # Radiation, clamp and current surfaces
104         # Specimen surfaces
105         self.surface_specimen_radiation = None
106         self.surface_specimen_radiation_name = 'Surface_Radiation'
107         self.surface_specimen_clamping = None
108         self.surface_specimen_clamping_name = 'Surface_Clamping'
109         self.surface_specimen_current = None
110         self.surface_specimen_current_name = 'Surface_Current'
```

```
111
112         # Clamp surfaces
113         self.surface_clamp_clamping = None
114         self.surface_clamp_clamping_name = 'Surface_Clamping'
115         self.surface_clamp_water = None
116         self.surface_clamp_water_name = 'Surface_Water'
117
118         # Reference point for mechanical deformation
119         self.referencePoint = None
120         self.set_referencePoint = None
121         self.set_referencePoint_name = 'Rotation_Point'
122
123         self.steps = []
124         self.fieldOutput = None
125         self.fieldOutput_name = None
126
127         # Setup basic Abaqus model database to allow access to all
                                    Abaqus classes
128         self._create_model_database()
129
130     def _create_model_database(self):
131         Mdb()
132         self.model_name = 'Torsion-Simulation'
133         mdb.models.changeKey(fromName='Model-1', toName=self.
                                    model_name)
134
135         self.model = mdb.models[self.model_name]
136         self.model.setValues(absoluteZero=-273.15,
137                              stefanBoltzmann=5.670374419e-8)
138
139     def _create_specimen_sketch(self):
140         """
141         Generate the sketch and the part according to DSI sample
142         specifications SMT001 REV. 003.
143         :return: None
144         """
145         global mm
146
147         sketch_geom = self.model.ConstrainedSketch(name='SMT001',
                                    sheetSize=200 * mm)
148         g, v, d, c = sketch_geom.geometry, sketch_geom.vertices,
                                    sketch_geom.dimensions,
                                    sketch_geom.constraints
149
150         sketch_geom.sketchOptions.setValues(viewStyle=AXISYM)
```

```
151         sketch_geom.setPrimaryObject(option=STANDALONE)
152         center_line = sketch_geom.ConstructionLine(point1=(0.0, -
                                100.0 * mm), point2=(0.0, 100.0 *
                                 mm))
153         sketch_geom.FixedConstraint(entity=center_line)
154         bottom_line = sketch_geom.Line(point1=(8.33/2 * mm, 0 * mm)
                                , point2=(7 * mm, 0 * mm))
155         sketch_geom.FixedConstraint(entity=bottom_line)
156
157         l1 = sketch_geom.Line(point1=(7 * mm, 0 * mm), point2=(7 *
                                mm, 72.55 * mm))
158         l2 = sketch_geom.Line(point1=(7 * mm, 72.55 * mm), point2=(
                                6.5 * mm, 72.55 * mm))
159         arc = sketch_geom.ArcByCenterEnds(center=(6.5 * mm, 73.55 *
                                 mm), point1=(6.5 * mm, 72.55 *
                                mm),
160                                         point2=(5.5 * mm, 73.55 *
                                 mm), direction=CLOCKWISE)
161         l3 = sketch_geom.Line(point1=(5.5 * mm, 73.55 * mm), point2
                                =(5.5 * mm, 80.55 * mm))
162         l4 = sketch_geom.Line(point1=(5.5 * mm, 80.55 * mm), point2
                                =(0 * mm, 80.55 * mm))
163         l5 = sketch_geom.Line(point1=(0 * mm, 80.55 * mm), point2=(
                                0 * mm, 70.55 * mm))
164         l6 = sketch_geom.Line(point1=(0 * mm, 70.55 * mm), point2=(
                                4 * mm, 60.55 * mm))
165         l7 = sketch_geom.Line(point1=(4 * mm, 60.55 * mm), point2=(
                                8.33/2 * mm, 0 * mm))
166
167         sketch_geom.VerticalConstraint(entity=l1)
168         sketch_geom.VerticalConstraint(entity=l3)
169         sketch_geom.VerticalConstraint(entity=l5)
170         sketch_geom.VerticalConstraint(entity=l7)
171         sketch_geom.HorizontalConstraint(entity=l2)
172         sketch_geom.HorizontalConstraint(entity=l4)
173
174         sketch_geom.TangentConstraint(entity1=l2, entity2=arc)
175         sketch_geom.TangentConstraint(entity1=arc, entity2=l3)
176         sketch_geom.RadialDimension(curve=arc, textPoint=(0, 0),
                                radius=1.27 * mm)
177
178         sketch_geom.DistanceDimension(entity1=center_line, entity2=
                                l7, textPoint=(0, 0), value=8.33/
                                2 * mm)
```

```
179         sketch_geom.DistanceDimension(entity1=center_line, entity2=
                                    l1, textPoint=(0, 0), value=7 *
                                    mm)
180         sketch_geom.DistanceDimension(entity1=center_line, entity2=
                                    l3, textPoint=(0, 0), value=5 *
                                    mm)
181         sketch_geom.DistanceDimension(entity1=center_line, entity2=
                                    l5, textPoint=(0, 0), value=0)
182         sketch_geom.DistanceDimension(entity1=bottom_line, entity2=
                                    l4, textPoint=(0, 0), value=82.55
                                     * mm)
183         sketch_geom.DistanceDimension(entity1=l4, entity2=l2,
                                    textPoint=(0, 0), value=10 * mm)
184         sketch_geom.DistanceDimension(entity1=l4, entity2=v[9],
                                    textPoint=(0, 0), value=12.7 * mm
                                    )
185         sketch_geom.AngularDimension(line1=l6, line2=l7, textPoint=
                                    (0, 0), value=121)
186
187         self.sketch_specimen = sketch_geom
188
189     def _create_clamp_sketch(self):
190         """
191
192         :return:
193         """
194         global mm
195
196         s = self.model.ConstrainedSketch(name='Clamp', sheetSize=
                                    200 * mm)
197         g, v, d, c = s.geometry, s.vertices, s.dimensions, s.
                                    constraints
198
199         s.sketchOptions.setValues(viewStyle=AXISYM)
200         s.setPrimaryObject(option=STANDALONE)
201         center_line = s.ConstructionLine(point1=(0.0, -100.0 * mm),
                                     point2=(0.0, 100.0 * mm))
202         s.FixedConstraint(entity=center_line)
203         x_axis = s.ConstructionLine(point1=(-100 * mm, 0), point2=(
                                    100.0 * mm, 0))
204         s.FixedConstraint(entity=x_axis)
205         x_axis_offset = s.ConstructionLine(point1=(-100 * mm, -10.
                                    * mm), point2=(100.0 * mm, -10. *
                                     mm))
206         s.FixedConstraint(entity=x_axis_offset)
```

```
207
208         l1 = s.Line(point1=(5 * mm, 10 * mm), point2=(15 * mm, 10 *
                                mm))
209         l2 = s.Line(point1=(15 * mm, 10 * mm), point2=(15 * mm, -10
                                * mm))
210         l3 = s.Line(point1=(15 * mm, -10 * mm), point2=(0 * mm, -10
                                * mm))
211         l4 = s.Line(point1=(0 * mm, -10 * mm), point2=(0 * mm, -5 *
                                mm))
212         l5 = s.Line(point1=(0 * mm, -5 * mm), point2=(5 * mm, -5 *
                                mm))
213         l6 = s.Line(point1=(5 * mm, -5 * mm), point2=(5 * mm, 10 *
                                mm))
214
215         s.HorizontalConstraint(entity=l1)
216         s.VerticalConstraint(entity=l2)
217         s.HorizontalConstraint(entity=l3)
218         s.VerticalConstraint(entity=l4)
219         s.HorizontalConstraint(entity=l5)
220         s.VerticalConstraint(entity=l6)
221
222         s.DistanceDimension(entity1=x_axis, entity2=l1, textPoint=(
                                0, 0), value=20.32 * mm)
223         s.DistanceDimension(entity1=center_line, entity2=l2,
                                textPoint=(0, 0), value=(7 + 15)
                                * mm)
224         s.DistanceDimension(entity1=l3, entity2=x_axis_offset,
                                textPoint=(0, 0), value=0 * mm)
225         s.DistanceDimension(entity1=center_line, entity2=l4,
                                textPoint=(0, 0), value=0)
226         s.DistanceDimension(entity1=l5, entity2=x_axis_offset,
                                textPoint=(0, 0), value=7.5 * mm)
227         s.DistanceDimension(entity1=center_line, entity2=l6,
                                textPoint=(0, 0), value=7 * mm)
228         self.sketch_clamp = s
229
230     def _create_specimen_part(self, twist=ON):
231         """
232
233         :return: None
234         """
235         self._create_specimen_sketch()
236         self.part_specimen = self.model.Part(name='SMT001',
                                dimensionality=AXISYMMETRIC,
```

```
237                                                 type=DEFORMABLE_BODY,
                                    twist=twist)
238         self.part_specimen.BaseShell(sketch=self.sketch_specimen)
239         session.viewports['Viewport: 1'].setValues(displayedObject=
                                    self.part_specimen)
240
241     def _create_clamp_part(self, twist=ON):
242         """
243
244                 :return: None
245                 """
246         self._create_clamp_sketch()
247         self.part_clamp = self.model.Part(name='Clamp',
                                    dimensionality=AXISYMMETRIC,
248                                             type=DEFORMABLE_BODY,
                                    twist=twist)
249         self.part_clamp.BaseShell(sketch=self.sketch_clamp)
250
251     def _create_specimen_basic_partitions(self):
252         """
253         Create to basic partitons on the specimen sketch.
254         :return: None
255         """
256         # Partitions required for a correct Mesh
257         self.add_partition_specimen([82.55 - 12.7, 82.55 - 10 + 1.
                                    27])
258
259         # Partition for the clamping area of the sample
260         self.add_partition_specimen((20.32,))
261
262     def add_partition_specimen(self, partition_pos):
263         """
264         Add a single or multiple additional partitions on the
                                    specimen face if they don't
                                    already exist.
265         :return: None if no partition was created, else a list of
                                    the created partitions
266         """
267         global mm
268
269         if type(partition_pos) in [int, float]:
270             partition_pos = tuple(partition_pos)
271
272         if type(partition_pos) not in [tuple, list]:
273             return None
```

```
274
275         s = self.model.ConstrainedSketch(name='__profile__',
                                             sheetSize=200 * mm)
276         x_axis = s.ConstructionLine(point1=(-30 * mm, 0), point2=(
                                             30 * mm, 0))
277         s.FixedConstraint(entity=x_axis)
278         p1, p2 = (0, 10 * mm), (10 * mm, 10 * mm)
279         for pos in partition_pos:
280             if pos not in self.part_specimen_partitions:
281                 part_line = s.Line(point1=p1, point2=p2)
282                 s.DistanceDimension(entity1=x_axis, entity2=
                                             part_line, textPoint=(0, 0),
                                             value=pos * mm)
283                 self.part_specimen_partitions.append(pos)
284
285         f = self.part_specimen.faces
286         self.part_specimen.PartitionFaceBySketch(faces=f, sketch=s)
287         del self.model.sketches['__profile__']
288
289         return partition_pos
290
291     def _create_specimen_sets(self):
292         """
293         Create all the sets on the specimen part. These include
                                             symmetry/boundary/section and
                                             control sets.
294         :return:
295         """
296         # Edge for torsion boundary condition
297         clamping_edge = self.part_specimen.edges.getClosest(
                                             coordinates=((7 * mm, 10 * mm, 0.
                                             0),))
298         self.set_specimen_clamping = self.part_specimen.Set(edges=
                                             part.EdgeArray((clamping_edge[0][
                                             0],)),
299                                                                 name=
                                             self.set_specimen_clamping_name)
300
301         # Edges for symmetry boundary condition
302         edge_symmetry_r = self.part_specimen.edges.getClosest(
                                             coordinates=((0.0, 80 * mm, 0.0),
303
                                                     (0.0, (82.55-10) * mm, 0.
                                             0)))
```

```
304         edges = part.EdgeArray((edge_symmetry_r[0][0],
                                    edge_symmetry_r[1][0]))
305         self.set_specimen_symR = self.part_specimen.Set(edges=edges
                                    ,
306                                                         name=self.
                                    set_specimen_symR_name)
307
308         edge_symmetry_z = self.part_specimen.edges.getClosest(
                                    coordinates=((3 * mm, 82.55 * mm,
                                     0.0),))
309         self.set_specimen_symZ = self.part_specimen.Set(edges=part.
                                    EdgeArray((edge_symmetry_z[0][0]
                                    ,)),
310                                                         name=self.
                                    set_specimen_symZ_name)
311
312         # Definition of the face-set of the specimen
313         self.set_specimen_face = self.part_specimen.Set(faces=self.
                                    part_specimen.faces,
314                                                         name=self.
                                    set_specimen_face_name)
315
316         # Definition of the control node for the temperature
                                    control
317         control_node = self.part_specimen.vertices.getClosest(
                                    coordinates=((5 * mm, 82.55 * mm,
                                     0), ))
318         self.set_specimen_temp_control = self.part_specimen.Set(
                                    vertices=part.VertexArray((
                                    control_node[0][0],)),
319
                                                         name=self.
                                    set_specimen_temp_control_name)
320
321     def _create_clamp_sets(self):
322         """
323
324         :return:
325         """
326
327         # Edge for torsion boundary condition
328         clamping_edge = self.part_clamp.edges.getClosest(
                                    coordinates=((7 * mm, 10 * mm, 0.
                                    0),))
```

```
329         self.set_clamp_clamping = self.part_clamp.Set(edges=part.
                                      EdgeArray((clamping_edge[0][0],))
                                      ,
330                                                           name=self.
                                      set_clamp_clamping_name)
331
332         # Edge for ground boundary condition
333         ground = self.part_clamp.edges.getClosest(coordinates=((5 *
                                       mm, -10.0 * mm, 0.0),))
334         self.set_clamp_ground = self.part_clamp.Set(edges=part.
                                      EdgeArray((ground[0][0],)),
335                                                             name=self.
                                      set_clamp_ground_name)
336
337         # Definition of the face-set of the clamp
338         self.set_clamp_face = self.part_clamp.Set(faces=self.
                                      part_clamp.faces,
339                                                         name=self.
                                      set_clamp_face_name)
340
341     def _create_specimen_surfaces(self):
342         """
343
344         :return:
345         """
346         global mm
347
348         # Definition of the clamping surface.
349         edge_clamp = self.part_specimen.edges.getClosest(
                                      coordinates=((7 * mm, 10 * mm, 0)
                                      ,))
350         self.surface_specimen_clamping = self.part_specimen.Surface
                                      (side1Edges=part.EdgeArray((
                                      edge_clamp[0][0],)),
351
                                       name=self.
                                      surface_specimen_clamping_name)
352
353         # Definition of the current surface.
354         self.surface_specimen_current = self.part_specimen.Surface(
                                      side1Edges=self.set_specimen_symZ
                                      .edges,
355
                                      name=self.
                                      surface_specimen_current_name)
```

```
356
357         # Definition of the radiation surface
358         edge_radiation = self.part_specimen.edges.getClosest(
                                    coordinates=((5 * mm, (82.55 - 5)
                                     * mm, 0),
359
                                            (5 * mm, (82.55 - 10) * mm
                                    , 0),
360
                                            (6.5 * mm, (82.55 - 10) *
                                    mm, 0),
361
                                            (7 * mm, (82.55 - 12) * mm
                                    , 0),
362
                                            (7 * mm, (82.55 - 25) * mm
                                    , 0)))
363         edge_radiation_list = []
364         for key, edge in edge_radiation.items():
365             edge_radiation_list.append(edge[0])
366
367         self.surface_specimen_radiation = self.part_specimen.
                                    Surface(side1Edges=part.EdgeArray
                                    (edge_radiation_list),
368
                                      name=self.
                                    surface_specimen_radiation_name)
369
370     def _create_clamp_surfaces(self):
371         """
372
373         :return:
374         """
375         global mm
376
377         # Definition of the clamping surface.
378         edge_clamp = self.part_clamp.edges.getClosest(coordinates=
                                    ((7 * mm, 10 * mm, 0),))
379         self.surface_clamp_clamping = self.part_clamp.Surface(
                                    side1Edges=part.EdgeArray((
                                    edge_clamp[0][0],)),
380                                                             name=
                                    self.surface_clamp_clamping_name)
381
382         # Definition of the watercooled surface.
```

```
383         self.surface_clamp_water = self.part_clamp.Surface(
                                    side1Edges=self.set_clamp_ground.
                                    edges,
384                                                                 name=
                                    self.surface_clamp_water_name)
385
386     def _create_sets(self):
387         self._create_specimen_sets()
388         self._create_clamp_sets()
389
390     def _create_surfaces(self):
391         self._create_specimen_surfaces()
392         self._create_clamp_surfaces()
393
394     def _create_materials(self):
395         """
396
397         :return:
398         """
399         self.material_specimen = self.model.Material(name=self.
                                    material_specimen_name)
400         self.material_clamp = self.model.Material(name=self.
                                    material_clamp_name)
401
402     def _assign_sections(self):
403         # Section for the specimen
404         specimen_section_name = 'Steel_Section'
405         self.model.HomogeneousSolidSection(name=
                                    specimen_section_name, material=
                                    self.material_specimen_name,
406                                             thickness=None)
407
408         region = regionToolset.Region(faces=self.part_specimen.
                                    faces)
409         self.part_specimen.SectionAssignment(region=region,
                                    sectionName=specimen_section_name
                                    , offset=0.0,
410                                                 offsetType=
                                    MIDDLE_SURFACE, offsetField='',
411                                                 thicknessAssignment=
                                    FROM_SECTION)
412         # Section for the clamp
413         clamp_section_name = 'Copper_Section'
414         self.model.HomogeneousSolidSection(name=clamp_section_name,
                                     material=self.
```

```
                                material_clamp_name,
415                                     thickness=None)
416         region = regionToolset.Region(faces=self.part_clamp.faces)
417         self.part_clamp.SectionAssignment(region=region,
                                sectionName=clamp_section_name,
                                offset=0.0,
418                                       offsetType=
                                MIDDLE_SURFACE, offsetField='',
419                                       thicknessAssignment=
                                FROM_SECTION)
420
421     def _create_assembly(self, ref_point=False):
422         """
423
424         :return:
425         """
426         global mm
427
428         self.assembly = self.model.rootAssembly
429         self.assembly.DatumCsysByThreePoints(coordSysType=
                                CYLINDRICAL, origin=(0.0, 0.0, 0.
                                0),
430                                       point1=(1.0 * mm, 0.0,
                                 0.0), point2=(0.0, 0.0, -1.0 *
                                mm))
431
432         self.part_specimen_instance = self.assembly.Instance(name=
                                self.part_specimen_instance_name,
433                                                    part=
                                self.part_specimen, dependent=ON)
434         self.part_clamp_instance = self.assembly.Instance(name=self
                                .part_clamp_instance_name,
435                                                   part=self
                                .part_clamp, dependent=ON)
436
437         if ref_point:
438             self.referencePoint = self.assembly.ReferencePoint(
                                point=(0.0, 0.0, 0.0))
439             self.set_referencePoint = self.assembly.Set(
                                referencePoints=
440                                                  (self.
                                assembly.referencePoints[self.
                                referencePoint.id],),
441                                                  name=self.
                                set_referencePoint_name)
```

```
442
443     def _create_interactions(self):
444         """
445
446         :return:
447         """
448         self.interaction_contact_property = self.model.
                                    ContactProperty(self.
                                    interaction_contact_property_name
                                    )
449
450         slave_surface = self.assembly.instances[self.
                                    part_specimen_instance_name]. \
451             surfaces[self.surface_specimen_clamping_name]
452         master_surface = self.assembly.instances[self.
                                    part_clamp_instance_name]. \
453             surfaces[self.surface_clamp_clamping_name]
454
455         self.interaction_contact = self.model.
                                    SurfaceToSurfaceContactStd(
456             name=self.interaction_contact_name, createStepName='
                                    Initial',
457             master=master_surface, slave=slave_surface, sliding=
                                    FINITE, thickness=ON,
458             interactionProperty=self.
                                    interaction_contact_property_name
                                    , adjustMethod=OVERCLOSED,
459             initialClearance=OMIT, datumAxis=None, clearanceRegion=
                                    None)
460
461     def create_step(self, steps=(('Torsion', STATIC_GENERAL, {}),))
                                    :
462         """
463         Create the steps specified in the tuple steps. Each step
                                    consist of its name, the
                                    symbolicConstant for
464         the procedureType and a dictionary for the optional
                                    arguments applicable to
                                    procedureType.
465         Steps are added chronological.
466         :return: None
467         """
468         previous_step = 'Initial' if not self.steps else self.steps
                                    [-1].name
469         for stepName, procedureType, stepOptions in steps:
```

```
470
471             if procedureType is STATIC_GENERAL:
472                 step = self.model.StaticStep(name=stepName,
                                    previous=previous_step, **
                                    stepOptions)
473
474             elif procedureType is HEAT_TRANSFER:
475                 step = self.model.HeatTransferStep(name=stepName,
                                    previous=previous_step, **
                                    stepOptions)
476
477             elif procedureType is COUPLED_THERMAL_ELECTRIC:
478                 step = self.model.CoupledThermalElectricStep(name=
                                    stepName, previous=previous_step,
                                     **stepOptions)
479
480             elif procedureType is COUPLED_TEMP_DISPLACEMENT:
481                 step = self.model.CoupledTempDisplacementStep(name=
                                    stepName, previous=previous_step,
                                     **stepOptions)
482
483             else:
484                 continue
485
486             self.steps.append(step)
487             previous_step = step.name
488
489         # Delete outputs that are created automatically
490         try:
491             del self.model.fieldOutputRequests['F-Output-1']
492         except KeyError:
493             pass
494
495         try:
496             del self.model.historyOutputRequests['H-Output-1']
497         except KeyError:
498             pass
499
500     def create_field_output(self, name, create_step_name):
501         """
502
503         :return:
504         """
505         self.fieldOutput_name = name
```

```
506         self.fieldOutput = self.model.FieldOutputRequest(name,
                                create_step_name, frequency=5)
507
508     def suppress_steps(self, steps):
509         """
510         Function to suppress steps, mainly for testing to increase
                                the simulation speed.
511         Step Initial is 0 and every created step has index i > 0.
512         :param steps: tuple of the steps to be suppressed
513         :return: None
514         """
515         for i_step in steps:
516             self.steps[i_step - 1].suppress()
517
518     def create_specimen_mesh(self, seedsize=0.225, elemCode=CGAX4):
519         """
520
521         :return:
522         """
523
524         region = regionToolset.Region(faces=self.part_specimen.
                                faces)
525         e = self.part_specimen.edges
526         self.part_specimen.seedPart(size=seedsize * mm,
                                deviationFactor=0.1,
                                minSizeFactor=0.1)
527
528         # Seed the deformation
529         pickedEdges = e.getByBoundingBox(xMin=0, yMin=73 * mm, xMax
                                =5 * mm, yMax=83 * mm)
530
531         self.part_specimen.seedEdgeBySize(edges=pickedEdges, size=
                                seedsize / 3 * mm,
                                deviationFactor=0.1,
532                                         minSizeFactor=0.1,
                                constraint=FINER)
533
534         # Seed the filet
535         pickedEdges = e.getClosest(coordinates=((5 * mm, (82.55 -
                                10) * mm, 0),))
536         self.part_specimen.seedEdgeBySize(edges=part.EdgeArray((
                                pickedEdges[0][0],)), size=
                                seedsize / 3 * mm,
                                deviationFactor=0.1,
```

```
537                                         minSizeFactor=0.1,
                                    constraint=FINER)
538
539         # Seed the transitions between the clamp and the
                                    deformation
540         pickedEdges = e.getByBoundingBox(xMin=0, yMin=0, xMax=10 *
                                    mm, yMax=70 * mm)
541
542         self.part_specimen.seedEdgeBySize(edges=pickedEdges, size=2
                                     * seedsize * mm, deviationFactor
                                    =0.1,
543                                         minSizeFactor=0.1,
                                    constraint=FINER)
544
545         elemType = mesh.ElemType(elemCode=elemCode, elemLibrary=
                                    STANDARD)
546         self.part_specimen.setElementType(regions=region, elemTypes
                                    =(elemType,))
547         self.part_specimen.setMeshControls(regions=region.faces,
                                    elemShape=QUAD, algorithm=
                                    ADVANCING_FRONT)
548
549         self.part_specimen.generateMesh()
550
551     def _create_clamp_mesh(self, elemCode=CGAX4):
552
553         seed_size = 2 * mm
554         region = regionToolset.Region(faces=self.part_clamp.faces)
555         self.part_clamp.seedPart(size=seed_size, deviationFactor=0.
                                    1, minSizeFactor=0.1)
556
557         elemType = mesh.ElemType(elemCode=elemCode, elemLibrary=
                                    STANDARD)
558         self.part_clamp.setElementType(regions=region, elemTypes=(
                                    elemType,))
559         self.part_clamp.setMeshControls(regions=region.faces,
                                    elemShape=QUAD, algorithm=
                                    ADVANCING_FRONT)
560
561         self.part_clamp.seedEdgeBySize(edges=self.
                                    surface_clamp_clamping.edges,
                                    size=seed_size/2,
562                                         deviationFactor=0.1,
                                    minSizeFactor=0.1, constraint=
                                    FINER)
```

```
563         self.part_clamp.generateMesh()
564
565     @staticmethod
566     def coord_y_outer_edge(x_pos):
567         """
568         Function to return the outer edge position of the specimen
                                    with a given x-position.
569         All values in mm!
570         :param x_pos: Distance from the centersection of the
                                    specimen
571         :return: y-coordinate of the specimen-outside to the given
                                    x-pos
572         """
573         if x_pos <= 10 - 1.27:
574             return 5.0
575
576         elif x_pos < 10:
577             return 5 + math.sqrt(1.27 ** 2 - (10 - x_pos) ** 2)
578
579         elif x_pos <= 82.55:
580             return 7.0
581
582         else:
583             AttributeError("x-value out of range!!")
584
585     @staticmethod
586     def coord_y_inner_edge(x_pos):
587         """
588         Function to return the inner edge position of the specimen
                                    with a given x-position.
589         All values in mm!
590         :param x_pos: Distance from the centersection of the
                                    specimen
591         :return: y-coordinate of the specimen-inside to the given x
                                    -pos
592         """
593         if x_pos < 10.197:
594             return 0.0
595
596         elif x_pos < 12.7:
597             return (8.33 / 2) / 2.503 * (x_pos - 10.197)
598
599         elif x_pos <= 82.55:
600             return 8.33 / 2
601
```

```
602         else:
603             AttributeError("x-value out of range!!")
604 
605     def setup_basic_model(self, twist=ON, element_code=CGAX4,
                                  reference_point=False, mesh=True)
                                  :
606         """
607         Basic steps to create the model.
608         :return: None
609         """
610         self._create_specimen_part(twist=twist)
611         self._create_clamp_part(twist=twist)
612         self._create_specimen_basic_partitions()
613         self._create_sets()
614         self._create_surfaces()
615         self._create_materials()
616         self._assign_sections()
617         self._create_assembly(ref_point=reference_point)
618         self._create_interactions()
619         self._create_clamp_mesh(elemCode=element_code)
620         if mesh:
621             self.create_specimen_mesh(elemCode=element_code)
622 
623     def setup_model(self, *args):
624         """
625         All arguments are passed to setup_basic_model
626         """
627 
628         self.setup_basic_model(*args)
629 
630 
631 class SMT001Mechanical(SMT001Base):
632     """
633         Mechanical implementation of the hot torsion test. This
                                  model contains all relevant
                                  mechanical boundaries
634         and loads. The materialmodel is implemented via the UHARD
                                  Subroutine, as a Hensel-Spittel
                                  model.
635 
636         :param hs_param: tuple -- Parameters for the HS-model
637 
638         :param theta: float -- Angle for the rotation (Care for
                                  mirrorsymmetry only half the
                                  angle is necessary)
```

```
639
640         :param strainrate: float -- The Strainrate is calculated at
                                        the effective radius of r_eff =
                                       0.74 * r
641
642         :kwargs: Any arguments for inherited base class
643     """
644
645     def __init__(self, hs_param=((1, ), ), theta=0, strainrate=1.0,
                                        **kwargs):
646         super(SMT001Mechanical, self).__init__(**kwargs)
647         self.hs_parameters = hs_param
648         self.theta = theta
649         self.strainrate = strainrate
650
651     def setup_mechanical_model(self):
652         self.update_material()
653         self.gen_interactions()
654         self.gen_boundary_conditions()
655
656     def update_material(self):
657         youngs_mod = self.material_data[self.material_specimen_name
                                       .lower()]["elastic"]
658         self.material_specimen.Elastic(temperatureDependency=ON,
                                       table=youngs_mod)
659
660         youngs_mod = self.material_data[self.material_clamp_name.
                                       lower()]["elastic"]
661         self.material_clamp.Elastic(temperatureDependency=OFF,
                                       table=youngs_mod)
662
663         self.material_specimen.Plastic(hardening=USER, table=[(x, )
                                        for x in self.hs_parameters])
664
665     def gen_interactions(self):
666         """
667         Define the interactions between the reference point and the
                                        end of the torsion sample.
668         The torsion set is created on the part, therefore the set
                                       name in the assembly must be
                                       found first.
669         :return:
670         """
671         torsion_set_specimen = ".".join([self.
                                       part_specimen_instance.name, self
```

```
                                .set_specimen_clamping_name])
672         torsion_set_clamp = ".".join([self.part_clamp_instance.name
                                , self.set_clamp_clamping_name])
673
674         self.model.Equation(name='U2_specimen', terms=((1.0,
                                torsion_set_specimen, 2),
675                                                        (-1.0, self.
                                set_referencePoint_name, 2)))
676         self.model.Equation(name='U2_clamp', terms=((1.0,
                                torsion_set_clamp, 2),
677                                                     (-1.0, self.
                                set_referencePoint_name, 2)))
678         self.model.Equation(name='UR2_specimen', terms=((1.0,
                                torsion_set_specimen, 5),
679                                                         (-1.0, self
                                .set_referencePoint_name, 5)))
680         self.model.Equation(name='UR2_clamp', terms=((1.0,
                                torsion_set_clamp, 5),
681                                                      (-1.0, self.
                                set_referencePoint_name, 5)))
682
683     def gen_boundary_conditions(self):
684         """
685         Define the boundary conditions of the torsion sample.
686         The symR, symZ set is created on the part, therefore the
                                set name in the assembly must be
                                found first.
687         :return:
688         """
689
690         region = self.assembly.instances[self.
                                part_specimen_instance_name].sets
                                [self.set_specimen_symR_name]
691         self.model.DisplacementBC(name='SymR', createStepName='
                                Initial',
692                                region=region, u1=SET, u2=UNSET,
                                ur2=UNSET, ur3=UNSET, amplitude=
                                UNSET,
693                                distributionType=UNIFORM,
                                fieldName='', localCsys=None)
694
695         region = self.assembly.instances[self.
                                part_specimen_instance_name].sets
                                [self.set_specimen_symZ_name]
```

```
696         self.model.DisplacementBC(name='SymZ', createStepName='
                                       Initial',
697                                    region=region, u1=UNSET, u2=SET,
                                       ur2=SET, ur3=UNSET, amplitude=
                                       UNSET,
698                                    distributionType=UNIFORM,
                                       fieldName='', localCsys=None)
699
700         region = self.assembly.instances[self.
                                       part_clamp_instance_name].sets[
                                       self.set_clamp_clamping_name]
701         self.model.DisplacementBC(name='SymR_Clamp', createStepName
                                       ='Initial',
702                                    region=region, u1=SET, u2=UNSET,
                                       ur2=UNSET, ur3=UNSET, amplitude=
                                       UNSET,
703                                    distributionType=UNIFORM,
                                       fieldName='', localCsys=None)
704
705         self.model.DisplacementBC(name='Torsion', createStepName='
                                       Initial', region=self.
                                       set_referencePoint,
706                                    u1=SET, u2=UNSET, ur2=SET, ur3=
                                       UNSET, amplitude=UNSET,
707                                    distributionType=UNIFORM,
                                       fieldName='', localCsys=None)
708
709         self.model.boundaryConditions['Torsion'].setValuesInStep(
                                       stepName=self.steps[-1].name, ur2
                                       =self.theta)
710
711     def set_output(self):
712         self.model.fieldOutputRequests['F-Output-1'].setValues(
713             variables=PRESELECT, timeInterval=0.25)
714         self.model.historyOutputRequests['H-Output-1'].setValues(
                                       variables=(
715             'UR', 'RM'), region=self.set_referencePoint,
                                       sectionPoints=DEFAULT, rebar=
                                       EXCLUDE,
716             numIntervals=50)
717
718     def setup_model(self):
719         self.setup_basic_model(twist=ON, element_code=CGAX4,
                                       reference_point=True)
720
```

```
721         steps = (('Torsion', STATIC_GENERAL, {"maxNumInc": 10000, "
                                  initialInc": 0.01,
722                                               "minInc": 1e-08, "
                                  maxInc": 0.1, "nlgeom": ON}),)
723         self.create_step(steps)
724
725         self.setup_mechanical_model()
726
727
728 class SMT001ThermalEmpirical(SMT001Base):
729     """
730     !!!Class is depreciated, no use anymore!!!
731
732     Was used to apply a temperature boundary on the torsion sample
                                  via thermocouple measurements.
733     """
734     def __init__(self, tc_data=((0, 1000), (5, 986), (8, 973), (10,
                                   945), (20, 912), (40, 818), (50,
                                   680)),
735                  **kwargs):
736
737         super(SMT001ThermalEmpirical, self).__init__(**kwargs)
738         self.tc_data = tc_data
739         self.tc_partitions = list(zip(*tc_data))[0]
740
741         self.fieldOutput = None
742         self.fieldOutput_name = 'Thermal_results'
743
744     def setup_thermal_model(self):
745         """
746
747         :return:
748         """
749         self.update_material_thermal_properties()
750         self._partition_specimen_sketch(self.tc_partitions)
751         self.set_temperature_bounds()
752
753     def update_material_thermal_properties(self):
754         """
755         Add thermophysical data to the simulation material
756         :return: None
757         """
758         material_table = ((20, 13.01, 493.31, 16.50e-6, 7847.1),  #
                                   temperature, conductivity, Cp,
                                  alpha, density
```

```
759                             (100, 15.08, 500.10, 16.73e-6, 7807.7),
760                             (200, 16.79, 508.73, 17.31e-6, 7765.6),
761                             (300, 18.16, 517.50, 17.02e-6, 7726.9),
762                             (400, 19.51, 526.43, 17.02e-6, 7686.9),
763                             (500, 20.86, 535.51, 17.92e-6, 7636.2),
764                             (600, 22.25, 544.74, 18.23e-6, 7589.6),
765                             (700, 23.70, 554.14, 18.54e-6, 7541.5),
766                             (800, 25.20, 563.69, 18.87e-6, 7491.8),
767                             (900, 26.78, 573.42, 19.19e-6, 7440.5),
768                             (1000, 28.43, 583.31, 19.53e-6, 7387.4),
769                             (1100, 30.16, 593.37, 19.86e-6, 7332.7),
770                             (1200, 31.98, 603.60, 20.21e-6, 7276.2))
771
772         self.material_specimen.Conductivity(temperatureDependency=
                                    ON, table=[[a[1], a[0]] for a in
                                    material_table])
773         self.material_specimen.SpecificHeat(temperatureDependency=
                                    ON, law=CONSTANTPRESSURE,
774                                             table=[[a[2], a[0]] for
                                     a in material_table])
775         self.material_specimen.Density(temperatureDependency=ON,
                                    table=[[a[4], a[0]] for a in
                                    material_table])
776         self.material_specimen.InelasticHeatFraction(fraction=0.9)
777
778     def set_temperature_bounds(self):
779
780         for i, tc in enumerate(self.tc_data):
781             tc_pos, tc_temp = tc
782             edge = self.part_specimen.edges.getClosest(coordinates=
                                    (((self.coord_y_outer_edge(tc_pos
                                    ) - 0.25) * mm,
783
                                      (62.23 - tc_pos) * mm, 0),))
784             edge_arr = part.EdgeArray((edge[0][0],))
785             tc_set_name = 'TC' + str(i + 1)
786             self.part_specimen.Set(edges=edge_arr, name=tc_set_name
                                    )
787             region = self.assembly.instances[self.
                                    part_specimen_instance_name].sets
                                    [tc_set_name]
788             self.model.TemperatureBC(name=tc_set_name,
789                                      createStepName=self.steps[0].
                                    name, region=region, fixed=OFF,
```

```
790                                     distributionType=UNIFORM,
                                    fieldName='', magnitude=tc_temp,
791                                     amplitude=UNSET)
792             for step in self.steps[1:]:
793                 self.model.boundaryConditions[tc_set_name].
                                    deactivate(step.name)
794
795     def setup_model(self):
796         """
797         Setup the complete model with all interactions.
798         :return:
799         """
800         steps = (('Heating', HEAT_TRANSFER, {"response":
                                    STEADY_STATE, "amplitude": RAMP})
                                    ,)
801
802         self.create_step(steps)
803
804         elemCode = DCAX4
805         self.setup_basic_model(twist=OFF, element_code=elemCode,
                                    mesh=False)
806         self.setup_thermal_model()
807         self.create_specimen_mesh(elemCode=elemCode)
808         self.create_field_output('Thermal_output', self.steps[0].
                                    name)
809         # Field output isnt setup correctly, right now it is only
                                    created but no options are set
810         self.suppress_steps(self.suppressed_steps)
811
812
813 class SMT001ThermalPhysical(SMT001Base):
814     """
815         Model for the thermal interactions of the torsion test.
                                    Heat transfer via radiation and
                                    conduction
816         along the clamps is considered.
817
818         :param epsilon_radiation: float -- Emissivity of the
                                    torsion sample.
819         :param clamp_thermCond: float -- thermal heat conductivity
                                    of the Clampinterface.
820         :param water_filmCoeff: float -- filmcoefficient of the
                                    watercooled surface.
821         :param thermocouples: tuple -- positions of thermocouples
                                    starting from the center.
```

```
822         :param kwargs: Arguments for the base classes
823         """
824     def __init__(self, epsilon_radiation=0.8, clamp_thermCond=1000,
                                 water_filmCoeff=1000,
825                  thermocouples=(0, 5, 8, 10, 20, 40, 50), **kwargs)
                                :
826         super(SMT001ThermalPhysical, self).__init__(**kwargs)
827         self.radiation_epsilon = epsilon_radiation
828         self.filmcoeff_water = water_filmCoeff
829         self.contact_thermCond = clamp_thermCond
830 
831         self.interaction_filmcoeff_water = None
832         self.interaction_filmcoeff_water_name = 'FilmCoeff_Water'
833 
834         self.thermocouples_pos = thermocouples
835         self.thermocouples = []
836 
837     def setup_thermal_model(self):
838         """
839 
840         :return:
841         """
842         self.update_material_thermal_properties()
843         self._thermal_interactions()
844 
845     def update_material_thermal_properties(self):
846         """
847         Add thermophysical data to the simulation material
848         :return: None
849         """
850         # Set material properties for A220 steel
851 
852         self.material_specimen.Conductivity(temperatureDependency=
                                    ON,
853                                             table=(self.
                                    material_data[self.
                                    material_specimen_name.lower()]
854                                                    ["conductivity"]
                                    ))
855         self.material_specimen.SpecificHeat(temperatureDependency=
                                    ON, law=CONSTANTPRESSURE,
856                                             table=(self.
                                    material_data[self.
                                    material_specimen_name.lower()]
```

```
857                                                    ["specificheat"]
                                    ))
858         self.material_specimen.Density(temperatureDependency=ON,
859                                        table=(self.material_data[
                                    self.material_specimen_name.lower
                                    ()]
860                                                    ["density"]))
861
862         self.material_specimen.InelasticHeatFraction(fraction=(self
                                    .material_data[self.
                                    material_specimen_name.lower()]
863                                                                    ["
                                    inelasticheatfraction"]))
864
865         # Set material properties for copper
866         self.material_clamp.Conductivity(temperatureDependency=OFF,
867                                          table=(self.material_data[
                                    self.material_clamp_name.lower()]
868                                                   ["conductivity"]))
869         self.material_clamp.SpecificHeat(temperatureDependency=OFF,
870                                          table=(self.material_data[
                                    self.material_clamp_name.lower()]
871                                                   ["specificheat"]))
872         self.material_clamp.Density(temperatureDependency=OFF,
873                                       table=(self.material_data[self.
                                    material_clamp_name.lower()]
874                                                ["density"]))
875
876     def _thermal_interactions(self):
877         """
878         Create the thermal interactions of the model.
879         :return:
880         """
881         global mm
882
883         # Radiation interaction for the Surface
884         surface = self.assembly.instances[self.
                                    part_specimen_instance_name]. \
885             surfaces[self.surface_specimen_radiation_name]
886         self.model.RadiationToAmbient(name=self.
                                    surface_specimen_radiation_name,
887                                       createStepName=self.steps[0].
                                    name, surface=surface,
888                                       radiationType=AMBIENT,
```

```
889                                     distributionType=UNIFORM,
                                    field='', emissivity=self.
                                    radiation_epsilon,
890                                     ambientTemperature=25.0,
                                    ambientTemperatureAmp='')
891
892         # Watercooling surface film interaction of the Clamp
893         surface = self.assembly.instances[self.
                                    part_clamp_instance_name].
                                    surfaces[self.
                                    surface_clamp_water_name]
894         self.interaction_filmcoeff_water = \
895             self.model.FilmCondition(name=self.
                                    interaction_filmcoeff_water_name,
896                                        createStepName=self.steps[0].
                                    name, surface=surface,
897                                        definition=EMBEDDED_COEFF,
                                    filmCoeff=self.filmcoeff_water,
898                                        filmCoeffAmplitude='',
                                    sinkTemperature=self.ambientT,
899                                        sinkAmplitude='',
                                    sinkDistributionType=UNIFORM,
                                    sinkFieldName='')
900
901         # Set the Thermal conductance of the contact
902         self.interaction_contact_property.ThermalConductance(
903             definition=TABULAR, clearanceDependency=ON,
                                    pressureDependency=OFF,
904             temperatureDependencyC=OFF, massFlowRateDependencyC=OFF
                                    , dependenciesC=0,
905             clearanceDepTable=((self.contact_thermCond, 0.0),
906                                (self.contact_thermCond, 1e-05),
907                                (0, 2e-05)))
908
909     def setup_model(self):
910         """
911         Setup the complete model with all interactions.
912         """
913         steps = (('Heating', HEAT_TRANSFER, {"response":
                                    STEADY_STATE, "amplitude": RAMP})
                                    ,
914                  ('Cooldown', HEAT_TRANSFER, dict(deltmx=0.5,
                                    timePeriod=360,
915                                                  maxNumInc=10000,
                                    initialInc=0.01, minInc=1e-08,
```

```
                                    maxInc=10)))
916         self.create_step(steps)
917
918         self.setup_basic_model(twist=OFF, element_code=DCAX8)
919         self.setup_thermal_model()
920         self.create_field_output('Thermal_output', self.steps[0].
                                    name)
921         # Field output isnt setup correctly, right it is only
                                    created but no options are set
922         self.suppress_steps(self.suppressed_steps)
923
924     def setpoint_temperature(self):
925         """
926         name is work in progress!
927         Method to extract the temperature of TC-Control over all
                                    steps and frames
928         :return: None, Outputs are written to a json -> list of the
                                     path values in all steps/frames
929         """
930         odb = session.openOdb(self.job_name + '.odb')
931         session.viewports['Viewport: 1'].setValues(displayedObject=
                                    odb)
932
933         temperature_variable = (('NT11', NODAL,),)
934         temperature_data = []
935         for n_step, step in enumerate(odb.steps.values()):
936             for n_frame, _ in enumerate(step.frames):
937                 path_obj = self.thermocouples[0]
938                 path_name = path_obj.name + '_result'
939                 res = session.XYDataFromPath(path=path_obj, name=
                                    path_name, shape=UNDEFORMED,
940                                                 pathStyle=
                                    UNIFORM_SPACING, labelType=
                                    X_COORDINATE,
941                                                 includeIntersections=
                                    False,
942                                                 step=n_step, frame=
                                    n_frame, variable=
                                    temperature_variable)
943                 temperature_data.append(res.data[-1][-1])
944         wr_json(self.job_name, temperature_data)
945
946
947 class SMT001ThermalPhysicalElectric(SMT001ThermalPhysical):
948     """
```

```
949         Class for the thermal-electric simulation of the hot
                                    torsion test.
950
951         :param clamp_elecCond: float -- Conductivity of the
                                    clamping interface
952         :param steady_state_potential: float -- Steady state
                                    potential for the sample, supress
                                     if temperature control
953                                                  is needed
954         :param kwargs: arguments for the base classes
955     """
956     def __init__(self, clamp_elecCond=1e10, steady_state_potential=
                                    0.0, **kwargs):
957         super(SMT001ThermalPhysicalElectric, self).__init__(**
                                    kwargs)
958         self.clamp_elecCond = clamp_elecCond
959         self.steady_state_potential = steady_state_potential
960
961     def setup_electric_model(self):
962         """
963         Setup the model
964         :return:
965         """
966         self.update_material_electric_properties()
967         self._create_electric_ground()
968         if not self.steady_state_potential:
969             self._create_electric_loads()
970             self._create_current_control()
971         else:
972             self._create_electric_potential()
973
974     def update_material_electric_properties(self):
975         self.material_specimen.ElectricalConductivity(
                                    temperatureDependency=ON,
976                                                  table=(self.
                                    material_data[self.
                                    material_specimen_name.lower()]
977                                                         ["
                                    electricalconductivity"]))
978         self.material_clamp.ElectricalConductivity(
                                    temperatureDependency=OFF,
979                                                  table=(self.
                                    material_data[self.
                                    material_clamp_name.lower()]
```

```
980                                                          ["
                                    electricalconductivity"]))
981
982         self.interaction_contact_property. \
983             ElectricalConductance(definition=TABULAR,
                                    clearanceDependency=ON,
                                    pressureDependency=OFF,
984                                 temperatureDependencyC=OFF,
                                    dependenciesC=0,
985                                 clearanceDepTable=((self.
                                    clamp_elecCond, 0.0),
986                                                    (self.
                                    clamp_elecCond, 1.e-5),
987                                                    (0.0, 2.e-5)))
988
989     def _create_electric_ground(self):
990         """
991
992         :return:
993         """
994
995         region = self.assembly.instances[self.
                                    part_clamp_instance_name].\
996             sets[self.set_clamp_ground_name]
997         self.model.ElectricPotentialBC(name=self.
                                    set_clamp_ground_name,
                                    createStepName=self.steps[0].name
                                    ,
998                                         region=region, magnitude=0)
999
1000    def _create_electric_potential(self):
1001        """
1002
1003        :return:
1004        """
1005        region = self.assembly.instances[self.
                                    part_specimen_instance_name].\
1006            sets[self.set_specimen_symZ_name]
1007        self.model.ElectricPotentialBC(name='Steadystate_Potential'
                                    , createStepName=self.steps[0].
                                    name,
1008                                        region=region, magnitude=
                                    self.steady_state_potential)
1009
1010    def _create_electric_loads(self):
```

```
1011
1012            region = self.assembly.instances[self.
                                    part_specimen_instance_name]. \
1013                surfaces[self.surface_specimen_current_name]
1014
1015            # !!!DONT change this name without changing it
                                    simultaneously in the UAMP
                                    Subroutine!!!
1016            current_amplitude = 'AMP_CURRENT'
1017            self.model.UserAmplitude(name=current_amplitude,
                                    numVariables=2, timeSpan=TOTAL)
1018
1019            self.model.SurfaceCurrent(name='Current', createStepName=
                                    self.steps[0].name, region=region
                                    , magnitude=1.0,
1020                                    amplitude=current_amplitude)
1021
1022        def _create_current_control(self):
1023            region = self.assembly.instances[self.
                                    part_specimen_instance_name].sets
                                    [self.
                                    set_specimen_temp_control_name]
1024
1025            # !!!DONT change this name without changing it
                                    simultaneously in the UAMP
                                    Subroutine!!!
1026            control_node_name = 'TEMP_CONTROL'
1027            self.model.HistoryOutputRequest(name=control_node_name,
1028                                            createStepName=self.steps[0
                                    ].name, variables=('NT',),
1029                                            region=region,
1030                                            sectionPoints=DEFAULT,
                                    rebar=EXCLUDE, sensor=ON)
1031
1032        def setup_model(self):
1033            """
1034            Setup the complete model with all interactions
1035            :return:
1036            """
1037            if self.steady_state_potential:
1038                step_options = dict(response=STEADY_STATE, timePeriod=1
                                    , amplitude=RAMP,
1039                                    maxNumInc=100, initialInc=0.1,
                                    minInc=1e-05, maxInc=1)
1040            else:
```

```
1041            step_options = dict(deltmx=5, timePeriod=9999,
1042                                maxNumInc=100000, initialInc=0.01,
                                      minInc=1e-08, maxInc=0.1)
1043
1044        steps = (('Thermal', COUPLED_THERMAL_ELECTRIC, step_options
                                    ),)
1045
1046        self.create_step(steps)
1047
1048        self.setup_basic_model(twist=OFF, element_code=DCAX4E)
1049        self.setup_thermal_model()
1050        self.setup_electric_model()
1051
1052        self.create_field_output('ThermElec_output', self.steps[0].
                                    name)
1053        self.fieldOutput.setValuesInStep(self.steps[0].name,
                                    variables=('NT', ))
1054
1055
1056 class SMT001ThermalMechanical(SMT001Mechanical,
                                    SMT001ThermalPhysical):
1057     """
1058     Model for the thermo-mechanical simulation of the hot torsion
                                    test. Inital temperature is
                                    supplied via a
1059     .odb from a previous FE-Analysis.
1060
1061     :param temperature_field: str -- Path to the .odb containing
                                    temperature information for the
                                    deformation.
1062     """
1063
1064     def __init__(self, temperature_field="", **kwargs):
1065         SMT001Mechanical.__init__(self, **kwargs)
1066         SMT001ThermalPhysical.__init__(self, **kwargs)
1067         self.temp_field = temperature_field
1068
1069     def _load_temperature_field(self):
1070         # Method to determine number of increments in a odb
1071
1072         odb_temp = session.openOdb(self.temp_field)
1073         last_inc = odb_temp.steps["Thermal"].frames[-1].
                                    incrementNumber
1074         odb_temp.close()
1075
```

```
1076            self.model.Temperature(name='Start_Temperature',
1077                                   createStepName='Initial',
                                          distributionType=FROM_FILE,
1078                                   fileName=self.temp_field,
1079                                   beginStep=1, beginIncrement=last_inc
                                          , endStep=None, endIncrement=None
                                          ,
1080                                   interpolate=OFF,
                                          absoluteExteriorTolerance=0.0,
1081                                   exteriorTolerance=0.05)
1082
1083        def setup_model(self):
1084            # The Strainrate is calculated at the effective radius of
                                          r_eff = 0.74 * r
1085            deformation_time = 0.74 * 5 * self.theta / (self.strainrate
                                           * math.sqrt(3) * 10)
1086
1087            steps = (('Deformation', COUPLED_TEMP_DISPLACEMENT, dict(
                                          deltmx=2, timePeriod=
                                          deformation_time,
1088
                                          nlgeom=ON, maxNumInc=10000,
                                          initialInc=0.0005,
1089
                                          minInc=1e-010, maxInc=
                                          deformation_time,
1090
                                          extrapolation=PARABOLIC)),)
1091            self.create_step(steps)
1092
1093            self.setup_basic_model(twist=ON, element_code=CGAX4T,
                                          reference_point=True)
1094            self.setup_mechanical_model()
1095            self.setup_thermal_model()
1096            self._load_temperature_field()
1097            self.model.FieldOutputRequest(name='FieldOutput',
1098                                              createStepName=self.steps[-1]
                                          .name,
1099                                              variables=('S', 'PE', 'PEEQ',
                                           'LE',
1100                                                         'U', 'UR', 'RM', '
                                          NT', 'HFL', 'ER'),
1101                                              numIntervals=4)
1102            time_step_size = deformation_time / self.theta * 0.05
1103            self.model.HistoryOutputRequest(name='UR-RM',
```

```
1104                                        createStepName=self.steps[-
                                    1].name,
1105                                        variables=('UR', 'RM'),
1106                                        region=self.assembly.sets[
                                    self.set_referencePoint_name],
1107                                        sectionPoints=DEFAULT,
                                    rebar=EXCLUDE, timeInterval=
                                    time_step_size)
1108            self.model.HistoryOutputRequest(name='Temperature',
1109                                        createStepName=self.steps[-
                                    1].name,
1110                                        variables=('NT', ),
1111                                        region=self.assembly.
                                    instances[self.
                                    part_specimen_instance_name].sets
                                    [self.
                                    set_specimen_temp_control_name],
1112                                        sectionPoints=DEFAULT,
                                    rebar=EXCLUDE, timeInterval=
                                    time_step_size)
1113            self.model.steps[self.steps[-1].name].DiagnosticPrint(
1114                frequency=1, contact=ON, plasticity=ON, residual=ON,
                                    solve=ON,
1115                modelChange=OFF)
```