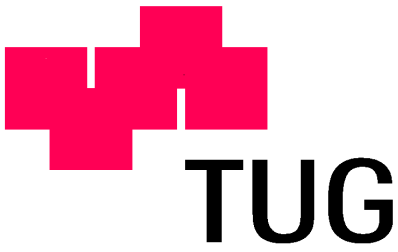
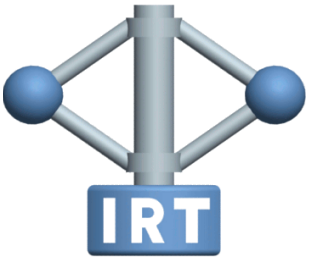




DIPLOMARBEIT

Modellbildung und Simulation von Wellenantriebssträngen mit nichtlinearen Elastomerkupplungen

MARTIN KIMMERSTORFER

ausgeführt am Institut für

Regelungstechnik

der Technischen Universität Graz

September 2012

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

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

Graz, am ……. ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

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

……. ……..
date (signature)

Kurzfassung

Der Einsatz von nichtlinearen Komponenten in Antriebssträngen stellt eine besondere Herausforderung in der Simulation dar. Neben dem Steifigkeitsverhalten ist besonders das praxistaugliche Dämpfungsverhalten eines der Kernthemen der Modellbildung. Diese Arbeit zeigt am Beispiel einer elastischen Kupplung Möglichkeiten, die zur Simulation erforderlichen Parameter aus gegebenen Momenten - Winkel Diagrammen zu bestimmen. Zusätzlich wurde ein auf Modelica basierendes Berechnungsmodell erstellt, wobei durch die zusätzlich entwickelten Bibliotheken eine Basis für weitere Untersuchungen gelegt wurde. Ebenfalls wurden Workflows in MATLAB erarbeitet, um die Open Source Simulationsumgebung OpenModelica zu steuern und deren Pre- und Postprocessing durchzuführen.

Abstract

The application of nonlinear components in drive shaft systems leads to particular challenges in simulation. Practically the stiffness and especially the damping behaviour are the main issues in modelling of such mechanical systems. This thesis deals with the evaluation of required damping parameters with the aid of torque - displacement curves obtained by measurements. The approach is done by a nonlinear coupling. Also a Modelica model library, including whole testebeds, was developed where the necessary components for further simulation are included. Furthermore workflows in MATLAB where developed to ensure the usage of open source simulationsoftware OpenModelica including pre- and postprocessing.

Inhaltsverzeichnis

Symbol- und Abkürzungsverzeichnis

$\varphi, \dot{\varphi}, \ddot{\varphi}$	Winkel, Winkelgeschwindigkeit, Winkebeschleunigung
$\varphi(s)$	Winkel im Bildbereich
M	Drehmoment
Θ	Massenträgheitsmoment
d	Dämpfungskonstante
d_k	kritische Dämpfungkonstante
D	Lehr'sches Dämpfungsmaß
c	Torsionsteifigkeit
t	Zeit
ω	Eigenfrequenz
ω_d	Eigenfrequenz der gedämpften Schwingung
Ω	Erregerfrequenz
A, B, C	Integrationskonstanten
$\bar{\varphi}$	statische Verdrehung bei $\hat{M}$ und c
V, V^*	Vergrößerungsfunktion
η	Frequenzverhältnis
δ	Abklingkonstante
Λ	Logarithmisches Dekrement
λ	Lösung der charakteristischen Gleichung
ψ	Phasenwinkel
$G(s)$	Übertragungsfunktion
$G(j\Omega)$	Frequenzgang
x	Kolbenweg
r	Kurbelradius
l	Pleuellänge
p_i	indizierter Mitteldruck
p_e	effektiver Mitteldruck
p_r	Reib-Mitteldruck
V_H	Hubvolumen
M_e	effektives Motormoment
n	Drehzahl

α	Drosselklappenstellung
ν	Poissonzahl
$p_1, p_2, ..., p_i$	Polynomkoeffizienten
F_R	Coulomb'sche Reibungskraft
d_{Reid}	Dämpfungskonstante im Reid'schen Dämpfungsansatz
$d_{Kelvin-Voigt}$	Dämpfungskonstante für Kelvin-Voigt'schen Dämpfungsansatz
$d_{Coulomb}$	Dämpfungskonstante fürCoulomb'schen Dämpfungsansatz
M_D	Dämpfungsmoment
$M_{Kelvin-Voigt}$	Dämpfungsmoment nach Kelvin-Voigt
M_{Reid}	Dämpfungsmoment nach Reid
$M_{Coulomb}$	Dämpfungsmoment nach Coulomb
EN, WN, UN, SN	Elastomerqualitäten der Fa Dipl. Ing. Herwarth Reich GMBH
ODE	gewöhnliche Differentialgleichung
FEM	Finite Elemente Methode
TVA	Torsions Vibrations Analyse
$I4T$	4 Zylinder Reihenmotor mit Turboaufladung
$t1000$	Produktbezeichnung Elastomerkupplung der tectos gmbh

Kapitel 1

Einleitung

1.1 Motivation

Die Entwicklung von Antriebssträngen wird durch die zügige Einführung von umweltfreundlichen Motoren (Downsizing, Downspeeding) vor immer anspruchsvollere Aufgaben gestellt. Dieser Fortschritt sorgt besonders im Bereich von Prüfständen zu äußerst komplexen Problemstellungen.

Zum Einen ist der Antriebsstrang Teil eines Messaufbaus und trägt erheblich zur Qualität der Tests bei, zum Anderen lassen sich klassische Ansätze der Maschinendynamik, die bisher meist ausgereicht haben, nicht mehr einsetzen.

Um eine Auslegung von Wellenverbindungen für Prüfstände zu erstellen, wird üblicherweise eine Drehschwingungssimulation im Frequenzbereich durchgeführt. Hierbei handelt es sich um eine Berechnung im eingeschwungenen Zustand, wobei keine instationären Effekte berücksichtigt werden können. Dieser Art der Berechnung erlaubt es nur eingeschränkt, nichtlineare Eigenschaften abzubilden. Eine zeitbasierte Simulation kann das dynamische Verhalten des Gesamtsystems deutlich detaillierter abbilden. Eine Gesamtsimulation von Motorprüfständen ist damit möglich. Es können auch Effekte wie Zylinderabschaltung und Power Cut hinreichend genau simuliert werden.

Zur numerischen Lösung stehen einige Softwaretools zur Verfügung. Um die physikalischen Systeme modellieren zu können, bietet sich vor allem Modelica an. Mit dieser Simulationsumgebung können interdisziplinäre Anwendungen simuliert werden, in denen verschiedene physikalischer Systeme gekoppelt sind. Der objektorientierte Zugang erleichtert zudem die Handhabung.

1.2 Zielsetzung

Ziel dieser wissenschaftlichen Arbeit ist die physikalische, zeitbasierte Modellbildung und Simulation von Antriebssträngen für Motoren und Getriebeprüfstände. Die Modellierung bezieht sich im Speziellen auf Elastomerkupplungen mit nichtlinearer Steifigkeit. Durch die detaillierten Untersuchungen

soll ein tieferes Systemverständnis erreicht und die Qualität der Auslegung wichtiger Komponenten gesteigert werden. Zusätzlich soll die Bestimmung der Dämpfung aus vorhandenen Messwerten in einer automatisierten Form entwickelt werden, um zukünftige Simulationen rasch und mit hoher Präzision bedaten zu können.

1.3 Zusammenfassung der Ergebnisse und Ausblick

Zu Beginn der Arbeit musste ein lauffähiges Simulationsmodell in Modelica aufgebaut werden. Dazu waren neben dem Einbau des Motors und des Reglers noch zusätzliche Funktionen zu entwickeln, die eine 3D Interpolation des Motorkennfelds ermöglichen. Besonderes Augenmerk galt der hinreichend genauen Modellierung der Anregung durch den Motor.

Für die einfachere Handhabung der Modelle wurde eine Routine entwickelt, die es erlaubt Modelica aus MATLAB heraus zu steuern und die Ergebnisse mit Hilfe der vorhandenen MATLAB Schnittstelle von Modelica wieder zurückzuführen.

Nach Abstimmung des linearen Modells wurden die nichtlinearen Funktionen implementiert und die entsprechenden Parameter bestimmt. Hier konnte nachgewiesen werden, dass die Hysterese in einen Steifigkeits- und einen Dämpfungsanteil zerlegt werden kann. Die Dämpfung ergibt sich bei näherer Betrachtung aus einer Überlagerung unterschiedlichster Dämpfungsmodelle. Zu deren Parametrierung wurde eine Methodik entwickelt, die es ermöglicht, die nötigen Kenngrößen aus Messdaten zu extrahieren.

Die Praxistauglichkeit des Modells wurde durch einen Messungs-Rechnungsvergleich gezeigt, wobei auch mögliche Verbesserungen sowohl des Motormodells als auch Details der nichtlinearen Komponenten erkannt wurden.

Bei den Analysen der Dämpfung konnte festgestellt werden, dass eine Abhängigkeit mancher Dämpfungsparameter vom anliegenden Moment sowie der Belastungsgeschwindigkeit vorliegt. Aufgrund fehlender Messdaten konnten diese Phänomene nicht näher untersucht werden. Sie stellen somit eine Herausforderung für ein weiterführendes Projekt dar.

Als weitere Verbesserung kann eine Detaillierung des Motormodells gesehen werden. Hier gilt ein besonderes Augenmerk der Motordämpfung und der Reibleistung.

Um einen praxistauglichen Einsatz von Modelica zu erreichen, sollte eine automatische Modellerstellung aus bestehenden Simulationsdatensätzen realisiert werden.

Kapitel 2

Grundlagen

Einleitend sollen die Grundlagen linearer und nichtlinearer dynamischer Systeme behandelt werden, um einen Einstieg in die Thematik zu finden und Begriffe und Zusammenhänge zu erklären. Die folgenden Beispiele behandeln Drehschwingungen mit dem Winkel φ, der Winkelgeschwindigkeit $\dot{\varphi}$ und der Winkelbeschleunigung $\ddot{\varphi}$. Die Massenträgheit ist mit Θ, die Dämpfung mit d, die Steifigkeit mit c und das äußere Drehmoment mit M, das auf das System einwirkt, definiert. Die Bewegung eines schwingfähigen Systems wird allgemein durch seine Bewegungsgleichung, resultierend aus dem Drallsatz, beschrieben.

$$\Theta(..., \varphi, t, ...)\ddot{\varphi} + d(..., \dot{\varphi}, \varphi, t, ...)\dot{\varphi} + c(..., \varphi, t, ...)\varphi = M(..., \ddot{\varphi}, \dot{\varphi}, \varphi, t, ...) \tag{2.1}$$

Vereinfachte Fälle der allgemeinen Bewegungsgleichung werden in den folgenden Abschnitten beschrieben und deren Lösungen behandelt.

2.1 Lineare Schwingungen

2.1.1 Einmassenschwinger

Der Einmassenschwinger ist die einfachste Art, ein schwingfähiges System zu beschreiben. Hierbei werden nur ein Massenträgheitsmoment und eine Steifigkeit berücksichtigt.

Zur Lösung der Differentialgleichung bietet sich der Übergang in den Bildbereich an, da diese, von eindimensionalen Torsionsschwingungssystemen, als gewöhnliche lineare Differentialgleichung mit konstanten Koeffizienten vorliegt. Mittels Laplace-Transformation und deren Rechenregeln kann die Lösung effizient ermittelt werden. Das allgemeine Vorgehen wird in Abbildung 2.1 dargestellt.

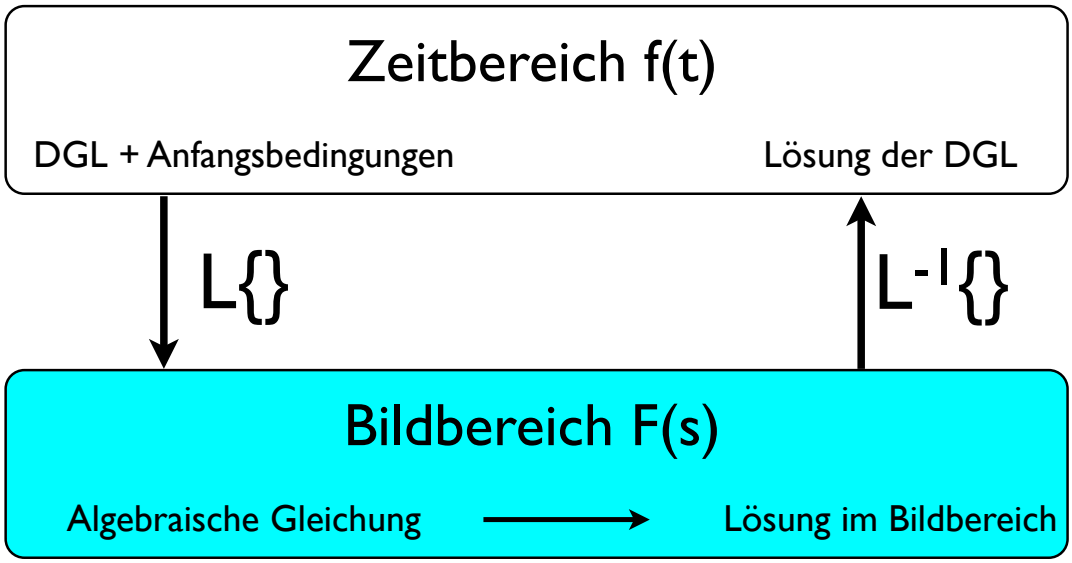


Abbildung 2.1: Lösungsweg mittels Laplace-Transformation

Zuerst wird die Differentialgleichung mit ihren Anfangsbedingungen bestimmt und mittels Laplace-Transformation in den Bildbereich übergeführt. Dadurch ergibt sich im Bildbereich eine algebraische Gleichung, die durch Umformung gelöst werden kann. Durch inverse Laplace-Transformation wird die Lösung aus dem Bildbereich in den Zeitbereich rücktransformiert. In [1] befindet sich eine detaillierte Beschreibung zur Laplace-Transformation, deren Rechenregeln und die nötigen Korrespondenz-Tabellen.

Für einen allgemeinen Fall eines linearen Torsionsschwingers gilt für die Bewegungsgleichung:

$$\begin{aligned} \theta\ddot{\varphi} + d\dot{\varphi} + c\varphi &= M(t) \\ bzw. \quad \ddot{\varphi} + 2\delta\dot{\varphi} + \omega^2\varphi &= M(t)\frac{\omega^2}{c}, \quad \varphi(0) = \varphi_0, \quad \dot{\varphi(0)} = \dot{\varphi}_0 \end{aligned} \tag{2.2}$$

$$mit \quad \omega^2 = \frac{c}{\Theta} \tag{2.3}$$

$$und \quad \frac{d}{\Theta} = 2\delta \tag{2.4}$$

Durch Laplace-Transformation der Bewegungsgleichung 2.2 in den Bildbereich erhält man:

$$\begin{aligned} \ddot{\varphi} + 2\delta\dot{\varphi} + \omega^2\varphi = M(t)\frac{\omega^2}{c} \quad &\circ\!\!-\!\!\bullet \\ s^2\varphi(s) - s\varphi_0 - \dot{\varphi}_0 + 2\delta(s\varphi(s) - \varphi_0) + \omega^2\varphi(s) &= M(s)\frac{\omega^2}{c} \end{aligned} \tag{2.5}$$

Die Lösung im Bildbereich kann durch Umformen der algebraischen Gleichung (2.5) ermittelt werden.

$$\varphi(s) = \frac{\dot{\varphi}_0 + \varphi_0 s + 2\delta\varphi_0}{s^2 + 2\delta s + \omega^2} + M(s)\frac{\frac{\omega^2}{c}}{s^2 + 2\delta s + \omega^2} \tag{2.6}$$

Hierbei beschreibt der erste Term ($\frac{\dot{\varphi}_0+\varphi_0 s+2\delta\varphi_0}{s^2+2\delta s+\omega^2}$) den Einfluss der Anfangsbedingungen und aus dem zweiten Term kann die Übertragungsfunktion abgelesen werden.

Übertragungsfunktion

Die Übertragungsfunktion beschreibt die Abhängigkeit des Ausgangssignals eines linearen, zeitinvarianten Systems vom Eingangssignal im Bildbereich. Sie wird durch den Quotient der Laplacetransformierten Ausgangsgröße zur Laplace-transformierten Eingangsgröße bei verschwindenden Anfangswerten definiert.

$$G(s) = \frac{\varphi(s)}{M(s)} \quad f\ddot{u}r\ \varphi_0 = \dot{\varphi}_0 = 0 \tag{2.7}$$

Die Übertragungsfunktion kann in vielen Fällen als gebrochen rationale Funktion angeschrieben werden

$$G(s) = \frac{P(s)}{Q(s)}, \tag{2.8}$$

wobei $P(s)$ das Zählerpolynom und $Q(s)$ das Nennerpolynom beschreibt. Das Nennerpolynom $Q(s)$ liefert das charakteristische Polynom, mit dem Aussagen über die Stabilität des dynamischen Systems getroffen werden können. Für den Einmassenschwinger ergibt sich für die Übertragungsfunktion:

$$G(s) = \frac{\frac{\omega^2}{c}}{s^2 + 2\delta s + \omega^2} \tag{2.9}$$

mit der charakteristischen Gleichung

$$\lambda^2 + 2\delta\lambda + \omega^2 = 0. \tag{2.10}$$

Stabilität

Mit Hilfe der Übertragungsfunktion können sehr leicht Aussagen über die Stabilität des Systems getroffen werden. Als Stabilitätsbegriff wird hier die so genannte BIBO-Stabilität verwendet. Demnach wird ein System BIBO-stabil genannt, wenn es ausgehend von seinem Ruhezustand auf jede beschränkte Eingangsfunktion mit einer beschränkten Ausgangsfunktion antwortet. Die notwendige und hinreichende Bedingung für die BIBO-Stabilität lautet:

> Alle Pole der Übertragungsfunktion $G(s)$ müssen einen negativen Realteil besitzen. Außerdem darf der Grad des Zählerpolynoms $P(s)$ den Grad des Nennerpolynoms $Q(s)$ nicht überschreiten.

Die Polstellen p_i der Übertragungsfunktion sind als Nullstellen des Nennerpolynoms $Q(s)$ bzw. durch $\lim_{s\to p_i} |G(s)| = \infty$ definiert.

Frequenzgang

Im Allgemeinen zeigt der Frequenzgang $G(j\Omega)$ die Amplitudenvergrößerung und die Phasenverschiebung des sinusförmigen Ausgangssignales, im eingeschwungenen Zustand, gegenüber dem sinusförmigen Eingangssignal eines linearen zeitinvarianten Systems. Ein bedeutender Vorteil ist die direkte Messbarkeit des Frequenzganges. Die Anregung wird vorteilhaft als komplexe Zeitfunktion angesetzt:

$$M(t) = \hat{M}e^{j\Omega t} = \hat{M}(cos(\Omega t) + jsin(\Omega t)) \tag{2.11}$$

Im Bildbereich gilt

$$\varphi(s) = G(s)M(s) = G(s)\frac{\hat{M}}{s - j\Omega} \tag{2.12}$$

wobei G(s) die Übertragungsfunktion darstellt. Nach dem Rücktransformieren und nach dem Abklingen der Einschwingvorgänge erhält man für $\varphi(t)$:

$$\varphi(t) = G(j\Omega)e^{j\Omega t}. \tag{2.13}$$

Die restliche Terme der Rücktransformation verschwinden, durch die Annahmen einer stabilen Übertragungsfunktion $G(s)$, vgl. [1]. Die Darstellung des Frequenzgangs als komplexe Größe erfolgt in Polarkoordinaten.

$$G(j\Omega) = |G(j\Omega)|e^{jarc(G(j\Omega))} \tag{2.14}$$

Durch die Linearität stellt der Realteil von $\varphi(t)$ in (2.13) die Antwort auf die Erregung $M(t) = \hat{M}cos(\Omega t)$ und der Imaginärteil die Antwort auf die Erregung durch $M(t) = \hat{M}sin(\Omega t)$ dar, d.h.

$$\varphi(t) = |G(j\Omega)|cos(\Omega t + arc(G(j\Omega))) + j|G(j\Omega)|sin(\Omega t + arc(G(j\Omega))). \tag{2.15}$$

2.1.2 Ungedämpfte freie Schwingungen

Im einfachsten Fall der ungedämpften Schwingung wird die Bewegung der Masse durch die Gleichung (2.16) beschrieben. Es gilt $\delta = 0$ und $M(t) = 0$.

$$\Theta\ddot{\varphi} + c\varphi = 0 \tag{2.16}$$

$$\ddot{\varphi} + \omega^2\varphi = 0, \quad \varphi(0) = \varphi_0, \quad \dot{\varphi(0)} = \dot{\varphi}_0 \tag{2.17}$$

$$\tag{2.18}$$

Die Laplace-Transformation ergibt:

$$\ddot{\varphi(t)} + \omega^2\varphi(t) = 0 \quad \circ\!\!-\!\!\bullet \quad s^2\varphi(s) - s\varphi_0 - \dot{\varphi_0} + \omega^2\varphi(s) = 0 \tag{2.19}$$

Durch Umformen ergibt sich die Lösung im Bildbereich.

$$\varphi(s) = \frac{\dot{\varphi_0} + \varphi_0 s}{s^2 + \omega^2} \tag{2.20}$$

Die Rücktransformation erfolgt per Laplace-Transformationstabelle, siehe [2], und ergibt die homogene Lösung der Differentialgleichung 2.17.

$$\varphi(s) = \frac{\dot{\varphi_0} + \varphi_0 s}{s^2 + \omega^2} \quad \bullet\!\!-\!\!\circ \quad \varphi(t)_{hom} = \varphi_0 sin(\omega t) + \frac{\dot{\varphi_0}}{\omega} sin(\omega t) \tag{2.21}$$

Eigenfrequenz

Die Eigenfrequenz ω (Gleichung 2.3) eines schwingfähigen Systems ist jene Frequenz, die sich durch einmalige Anregung einstellt, und als Eigenform schwingt. Auch kann die Eigenfrequenz durch das sensible Verhalten bei Erregung beschrieben werden, da bei kleinen Erregungsamplituden bereits sehr hohe Ausgangsamplituden entstehen, wenn die Anregungsfrequenz mit der Eigenfrequenz übereinstimmt, was einer Resonanz entspricht. Grundsätzlich entspricht die Zahl der Eigenfrequenzen eines Mehrkörpersystems, der Zahl der Freiheitsgrade. Zur Berechnung der Eigenfrequenzen eines dynamischen Systems mit mehreren Freiheitsgraden wird das Eigenwertproblem der Systemmatrix gelöst.

2.1.3 Ungedämpfte erzwungene Schwingungen

Für den Fall einer erzwungenen Schwingung wird das System von außen angeregt. Dies kann auf viele Arten z.B. winkelerregt, momenterregt, unwuchterregt oder selbsterregt sein.

$$\theta\ddot{\varphi} + c\varphi = M(..., \ddot{\varphi}, \dot{\varphi}, \varphi, t, ...) \tag{2.22}$$

Zur Einleitung wird eine harmonische Momenterregung gewählt. Es gilt $\delta = 0$ und $M(t) = \hat{M}cos\Omega t$. Die Bewegungsgleichung lautet nun

$$\ddot{\varphi} + \omega^2\dot{\varphi} = \hat{M}\frac{\omega^2}{c}cos\Omega t, \quad \varphi(0) = \varphi_0, \quad \dot{\varphi(0)} = \dot{\varphi_0} \tag{2.23}$$

wobei $\hat{M}$ die Amplitude und Ω die Frequenz der Erregung sind. Zum Lösen der Differentialgleichung wird der Übergang in den Bildbereich mittels Laplace-Transformation ausgeführt.

$$\ddot{\varphi(t)} + \omega^2\varphi(t) = \hat{M}\frac{\omega^2}{c}cos\Omega t \quad \circ\!\!-\!\!\bullet \quad s^2\varphi(s) - s\varphi_0 - \dot{\varphi_0} + \omega^2\varphi(s) = \hat{M}\frac{\omega^2}{c}\frac{s}{s^2 + \Omega^2} \tag{2.24}$$

Das Umformen der linearen algebraischen Gleichung im Bildbereich liefert die Lösung $\varphi(s)$, wobei die Lösung auf Grund der Anfangswerte derjenigen aus dem Abschnitt 2.1.2 gleicht.

$$\varphi(s) = \frac{\dot{\varphi_0} + \varphi_0 s}{s^2 + \omega^2} + \hat{M}\frac{s\frac{\omega^2}{c}}{(s^2 + \omega^2)(s^2 + \Omega^2)} \tag{2.25}$$

Die Transformation in den Zeitbereich mittels Transformationstabellen, siehe [2], ergibt:

$$\begin{aligned} \varphi(s) &= \frac{\dot{\varphi_0} + \varphi_0 s}{s^2 + \omega^2} + \hat{M}\frac{s\frac{\omega^2}{c}}{(s^2 + \omega^2)(s^2 + \Omega^2)} \quad \bullet\!\!-\!\!\circ \\ \varphi(t) &= \varphi_0 cos(\omega t) + \frac{\dot{\varphi_0}}{\omega} sin(\omega t) + \hat{M}\frac{\omega^2}{c}\frac{cos(\Omega t) - cos(\omega t)}{\omega^2 - \Omega^2}. \end{aligned} \tag{2.26}$$

Durch die Einführung des Frequenzverhältnisses von Erreger- zu Eigenfrequenz $\eta = \frac{\Omega}{\omega}$ kann die partikuläre Lösung auch als

$$\varphi_{part}(t) = \frac{\hat{M}}{c}\frac{1}{1 - \eta^2} cos(\Omega t) \tag{2.27}$$

$$\hat{\varphi}(\eta) = \frac{\hat{M}}{c}\frac{1}{1 - \eta^2} \tag{2.28}$$

angeschrieben werden, wobei $\bar{\varphi} = \frac{\hat{M}}{c}$ der statischen Verdrehung des Massepunktes mit dem Massenträgheitsmoment Θ durch das konstante Drehmoment $\hat{M}$ entspricht. Die Vergrößerungsfunktion V in Gleichung 2.29 beschreibt den Quotienten zwischen der dynamischen Überhöhung $\hat{\varphi}(\eta)$ zur statischen Auslenkung $\bar{\varphi}$. Die Vergrößerungsfunktion der ungedämpften Schwingung wird in Abbildung 2.2 über das Frequenzverhältnis η aufgetragen.

$$V(\eta) = \frac{\hat{\varphi}(\eta)}{\bar{\varphi}} \tag{2.29}$$

$$|V| = \left|\frac{1}{1 - \eta^2}\right| \tag{2.30}$$

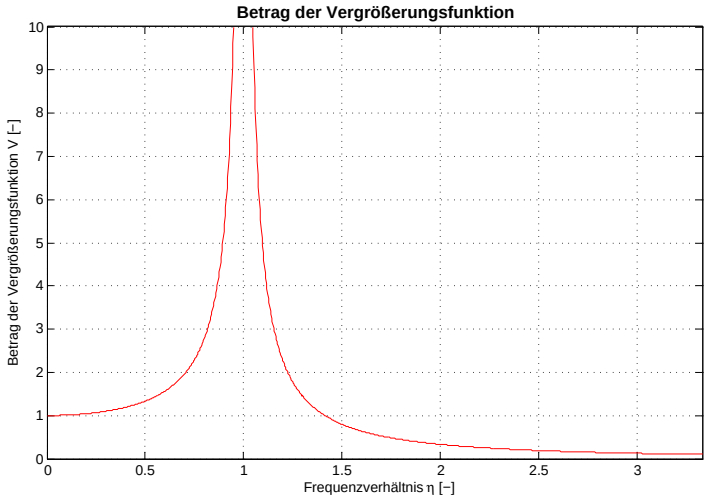


Abbildung 2.2: Betrag der Vergrößerungsfunktion ungedämpft

Die Resonanzkatastrophe bei $\Omega = \omega$ bzw. $\eta = 1$ ist durch ein asymptotisches Verhalten ersichtlich.

Resonanz

Eine Resonanz entsteht wenn Erreger- und Eigenfrequenz zusammenfallen ($\Omega = \omega$ bzw. $\eta = 1$). Wie in Abbildung 2.2 und 2.3 ersichtlich steigt bei konstanter Erregeramplitude die Vergrößerungsfunktion V ins Unendliche. In der Realität reduziert die stets vorhandene Dämpfung die Amplituden auf ein endliches Maß. Jedoch führt ein dauerhafter Betrieb der Anlage in Resonanz meist zu Schäden bzw. kann es zur Zerstörung führen.

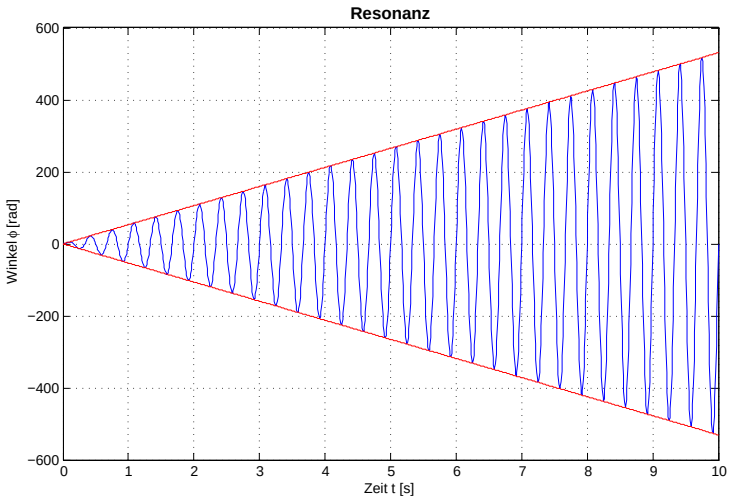


Abbildung 2.3: Resonanz

2.1.4 Gedämpfte freie Schwingungen

In der Realität treten ungedämpfte Schwingungen nicht auf, da jeder Zyklus verlustbehaftet ist. Bei ungedämpften Schwingungen wird die Energie jeweils zwischen potentieller und kinetischer Energie verschoben. Bei gedämpften Schwingungen wird ein Teil der Gesamtenergie in Wärmeenergie umgewandelt und somit für das mechanische System dissipiert. Auf die auftretenden Dämpfungsarten wird in Kapitel 4.4 genauer eingegangen. Hier soll der analytische Weg der Beschreibung der Dämpfung erfolgen. Es gilt $\delta > 0$ und $M(t) = 0$. Die Bewegungsgleichung muss mit einem geschwindigkeitsabhängigen Dissipationsterm erweitert werden, wodurch sich die Bewegungsgleichung

$$\Theta\ddot{\varphi} + d\dot{\varphi} + c\varphi = 0, \quad \varphi(0) = \varphi_0, \quad \dot{\varphi(0)} = \dot{\varphi}_0 \tag{2.31}$$

ergibt.

Durch Dividieren der Bewegungsgleichung (2.31) durch das Massenträgheitsmoment und Erweitern durch (2.3) und (2.4) ergibt sich die bekannte Form der allgemeinen Bewegungsgleichung eines gedämpften Schwingers (2.32) ohne äußere Anregung.

$$\ddot{\varphi} + 2\delta\dot{\varphi} + \omega^2\varphi = 0, \quad \varphi(0) = \varphi_0, \quad \dot{\varphi(0)} = \dot{\varphi}_0 \tag{2.32}$$

Die Laplace-Transformation ergibt:

$$\ddot{\varphi} + 2\delta\dot{\varphi} + \omega^2\varphi = 0 \quad \circ\!\!-\!\!\bullet \quad s^2\varphi(s) - s\varphi_0 - \dot{\varphi}_0 + 2\delta\big(s\varphi(s) - \varphi_0\big) + \omega^2\varphi(s) = 0 \tag{2.33}$$

Durch Umformen erhält man die Lösung im Bildbereich $\varphi(s)$

$$\varphi(s) = \frac{\dot{\varphi}_0 + \varphi_0 s + 2\delta\varphi_0}{s^2 + 2\delta s + \omega^2} \tag{2.34}$$

Die Lösung der charakteristischen Gleichung $\lambda^2 + 2\delta\lambda + \omega^2 = 0$ ergibt für die Eigenwerte $\lambda_{1,2}$

$$\lambda_{1,2} = -\delta \pm \sqrt{\delta^2 - \omega^2}. \tag{2.35}$$

In Abhängigkeit der Parameter ω und δ sind die Lösungen für die Eigenwerte $\lambda_{1,2}$ in 3 Bereiche einzuteilen.

$$\delta > \omega \tag{2.36}$$

$$\delta = \omega \tag{2.37}$$

$$\delta < \omega. \tag{2.38}$$

1. Aperiodische Dämpfung ($\delta > \omega$). Die Auslenkung aus der Ruhelage führt zu keiner Schwingung. Die Masse kehrt aperiodisch zur Ausgangslage zurück. Je nach Anfangsbedingungen wird höchstens ein Maximum bzw. ein Nulldurchgang durchlaufen, siehe Abbildung 2.4. Die

Lösung der Bewegungsgleichung (2.32), für den Fall $\delta > \omega$, ergibt sich im Bildbereich zu

$$\varphi(s) = \frac{\dot{\varphi}_0 + \varphi_0 s + 2\delta\varphi_0}{(s+\delta)^2 - \omega_a^2} \tag{2.39}$$

$$mit \quad \omega_a^2 = \delta^2 - \omega^2 \tag{2.40}$$

In diesem Fall gilt $\sqrt{\delta^2 - \omega^2} > 0$ und somit sind die Eigenwerte reelle und es gilt $\lambda_1 < 0, \lambda_2 < 0$. Die Lösung der Bewegungsgleichung $\varphi(t)$ ergibt eine exponentielle, aperiodische Bewegung. Durch Rücktransformieren mittels Transformationstabelle, siehe [2], erhält man die Lösung im Zeitbereich.

$$\begin{aligned} \varphi(s) &= \varphi_0 \frac{s+\delta}{(s+\delta)^2 - \omega_a^2} + \varphi_0 \frac{\delta}{(s+\delta)^2 - \omega_a^2} + \dot{\varphi}_0 \frac{1}{(s+\delta)^2 - \omega_a^2} \quad \bullet\!\!-\!\!\circ \\ \varphi(t) &= (\varphi_0 cosh(\omega_a t) + \frac{\dot{\varphi}_0 + \varphi_0\delta}{\omega_a} sinh(\omega_a t)) e^{-\delta t} \end{aligned} \tag{2.41}$$

2. Aperiodischer Grenzfall ($\delta = \omega$). Im Grenzfall gilt $\sqrt{\delta^2 - \omega^2} = 0$. Die Eigenwerte sind reele und gleich groß. Die kritische Dämpfungskonstante d_k kann durch Nullsetzen der Wurzel $\sqrt{(\frac{d_k}{2\Theta})^2 - \frac{c}{\Theta}}$ bestimmt werden und wird zu $d_k = 2\sqrt{\Theta c}$.

$$\lambda_1 = \lambda_2 = -\delta = -\frac{d_k}{2\Theta} = -\omega \tag{2.42}$$

Durch Umformen der allgemeinen Lösung $\varphi(s)$ im Bildbereich erhält man

$$\varphi(s) = \frac{\dot{\varphi}_0 + \varphi_0 s + 2\delta\varphi_0}{s^2 + 2\omega s + \omega^2}. \tag{2.43}$$

Durch inverse Laplace-Transformation ergibt sich die allgemeine Lösung der Bewegungsgleichung (2.32) zu

$$\begin{aligned} \varphi(s) &= \frac{s}{(s+\omega)^2}\varphi_0 + \frac{1}{(s+\omega)^2}(\dot{\varphi}_0 + 2\delta\varphi_0) \quad \bullet\!\!-\!\!\circ \\ \varphi(t) &= (\varphi_0 + (\dot{\varphi}_0 + \omega\varphi_0)t) e^{-\omega t} \end{aligned} \tag{2.44}$$

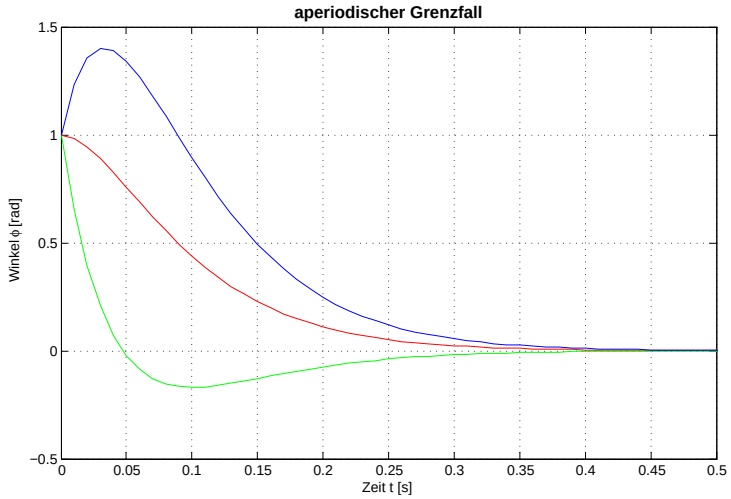


Abbildung 2.4: aperiodischer Grenzfall für $\dot{\varphi}_0 = 0$, $\dot{\varphi}_0 > 0$ und $\dot{\varphi}_0 < 0$

3. Periodische Dämpfung ($\delta < \omega$). Bei diesem Fall tritt eine gedämpfte Schwingung ein, die durch eine zeitlich abklingende Amplitude gekennzeichnet ist. Das Argument der Wurzel zur Bestimmung der Eigenwerte $\lambda_{1,2}$ ist negativ. Die daraus resultierenden Eigenwerte sind somit konjugiert komplex.

$$\lambda_{1,2} = -\delta \pm j\sqrt{\omega^2 - \delta^2}. \tag{2.45}$$

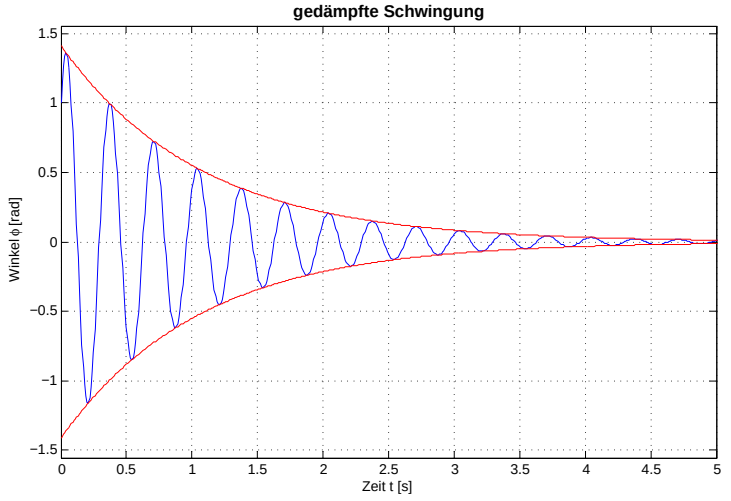


Abbildung 2.5: periodische Dämpfung

Die Frequenz der gedämpften Schwingung (ω_d) ergibt sich zu

$$\omega_d = \sqrt{\omega^2 - \delta^2} \tag{2.46}$$

und λ zu

$$\lambda_{1,2} = -\delta \pm j\omega_d \tag{2.47}$$

Es wird der Dämpfungsgrad D (2.48) eingeführt. Dieser findet in der Definition von ω_d Verwendung (2.49).

$$D = \frac{\delta}{\omega} \tag{2.48}$$

$$\omega_d = \omega\sqrt{1 - D^2} \tag{2.49}$$

Die Lösung der Bewegungsgleichung (2.32), für den Fall $\delta < \omega$, ergibt sich im Bildbereich zu

$$\varphi(s) = \frac{\dot{\varphi}_0 + \varphi_0 s + 2\delta\varphi_0}{(s+\delta)^2 + \omega_d^2} \tag{2.50}$$

Mit der inversen Laplace-Transformation mittels Transformationstabelle, siehe [2], ergibt sich die Lösung im Zeitbereich.

$$\begin{aligned} \varphi(s) &= \varphi_0 \frac{s+\delta}{(s+\delta)^2 + \omega_d^2} + \varphi_0 \frac{\delta}{(s+\delta)^2 + \omega_d^2} + \dot{\varphi}_0 \frac{1}{(s+\delta)^2 + \omega_d^2} \quad \bullet\!\!-\!\!\circ \\ \varphi(t) &= (\varphi_0 cos(\omega_d t) + \frac{\dot{\varphi}_0 + \delta\varphi_0}{\omega_d} sin(\omega_d t)) e^{-\delta t} \end{aligned} \tag{2.51}$$

$$\varphi(t) = Ce^{-\delta t} sin(\omega_d t + \alpha) \tag{2.52}$$

$$mit \quad C = \sqrt{\varphi_0^2 + (\frac{\dot{\varphi}_0 + \delta\varphi_0}{\omega_d})^2}, \alpha = arctan(\frac{(\frac{\dot{\varphi}_0 + \delta\varphi_0}{\omega_d})}{\varphi_0}) \tag{2.53}$$

Die Amplitudenfunktion $Ce^{-\delta t}$ beschreibt den Einfluss der Dämpfung als Hüllkurve der Amplituden (Abbildung 2.5).

Logarithmisches Dekrement

Das logarithmische Dekrement Λ beschreibt das Dämpfungsverhalten eines freien Schwingungssystems. Es entspricht der logarithmischen Darstellung des Verhältnisses von zwei aufeinander folgenden Maxima, wobei φ_n den Winkel beim Zeitpunkt n und φ_{n+1} beim Zeitpunkt $n+1$ beschreibt.

$$\Lambda = ln\frac{\varphi_n}{\varphi_{n+1}} = \frac{2\pi D}{\sqrt{1 - D^2}} \tag{2.54}$$

Ein großer Vorteil der Beschreibung des Dämpfungsverhaltens mit dem logarithmischen Dekrement ist die direkte Messbarkeit dieser Größe. Dafür müssen zwei aufeinanderfolgende, durch eine Periodendauer T getrennte, Auslenkungsmaxima aufgenommen werden.

$$T = \frac{2\pi}{\omega_d} \tag{2.55}$$

2.1.5 Gedämpfte erzwungene Schwingung

Bei einer gedämpften erzwungenen Schwingung klingt die Eigenschwingung ab bis ein eingeschwungener Zustand erreicht wird, wo das System nur mehr mit der Erregerfrequenz schwingt. Es gilt $\delta > 0$ und $M(t) = \hat{M}cos(\omega t)$.

$$\theta\ddot{\varphi} + d\dot{\varphi} + c\varphi = M(t), \quad \varphi(0) = \varphi_0, \quad \dot{\varphi(0)} = \dot{\varphi}_0 \tag{2.56}$$

Die Bewegungsgleichung (2.56) wird durch Laplace-Transformation in den Bildbereich gebracht.

$$\begin{aligned} \ddot{\varphi(t)} + 2\delta\dot{\varphi} + \omega^2\varphi(t) = \hat{M}\frac{\omega^2}{c}cos\Omega t \quad \circ\!\!-\!\!\bullet \\ s^2\varphi(s) - s\varphi_0 - \dot{\varphi}_0 + 2\delta(s\varphi(s) - \varphi_0) + \omega^2\varphi(s) = \hat{M}\frac{\omega^2}{c}\frac{s}{s^2+\Omega^2} \end{aligned} \tag{2.57}$$

Durch Umformen wird die Lösung $\varphi(s)$ ermittelt.

$$\varphi(s) = \frac{\dot{\varphi}_0 + \varphi_0 s + 2\delta\varphi_0}{s^2 + 2\delta s + \omega^2} + \hat{M}\frac{\frac{\omega^2}{c}s}{(s^2 + 2\delta s + \omega^2)(s^2 + \Omega^2)} \tag{2.58}$$

Für den Fall der harmonischen Anregung des linearen zeitinvarianten Systems bietet sich die Lösung mit Hilfe des Frequenzganges an. Bei einer periodisch gedämpften, harmonisch erregten Schwingung ($\delta < \omega$) wird die Übertragungsfunktion G(s) aus Gleichung 2.58 in den Frequenzgang $G(j\Omega)$ übergeführt.

$$G(s) = \frac{\frac{\omega^2}{c}}{s^2 + 2\delta s + \omega^2} \tag{2.59}$$

$$G(j\Omega) = \frac{\frac{\omega^2}{c}}{\omega^2 - \Omega^2 + j2\delta\Omega} \tag{2.60}$$

$$|G(j\Omega)| = \frac{\frac{\omega^2}{c}}{\sqrt{(\omega^2 - \Omega^2)^2 + 4\delta^2\Omega^2}} \tag{2.61}$$

$$arcG(j\Omega) = -arctan(\frac{2\delta\Omega}{\omega^2 - \Omega^2}) \tag{2.62}$$

Dies liefert die Lösung im eingeschwungenen Zustand durch Einsetzen von 2.61 und 2.62 in Gleichung 2.14.

$$\varphi(t) = \hat{M}\frac{\frac{\omega^2}{c}}{\sqrt{(\omega^2 - \Omega^2)^2 + 4\delta^2\Omega^2}}cos(\Omega t - arctan(\frac{2\delta\Omega}{\omega^2 - \Omega^2})) \tag{2.63}$$

Vergrößerungsfunktion

Die Vergrößerungsfunktion V in Gleichung 2.64 beschreibt den Quotienten zwischen der dynamischen Überhöhung $\hat{\varphi}(\eta)$ zur statischen Auslenkung $\bar{\varphi}$.

$$V = \frac{\hat{\varphi}(\eta)}{\bar{\varphi}} = \frac{1}{(1-\eta^2) + 2jD\eta} \tag{2.64}$$

In Abbildung 2.6 ist den Verlauf des Betrages der Vergrößerungsfunktion und der Einfluss der Dämpfung in Abhängigkeit von η dargestellt.

$$|V| = \frac{1}{\sqrt{(1-\eta^2)^2 + 4D^2\eta^2}} \tag{2.65}$$

Der Phasenwinkel kann durch (2.66) beschrieben werden und ist in Abbildung 2.7 ersichtlich.

$$\psi = -arctan\frac{2D\eta}{1-\eta^2} \tag{2.66}$$

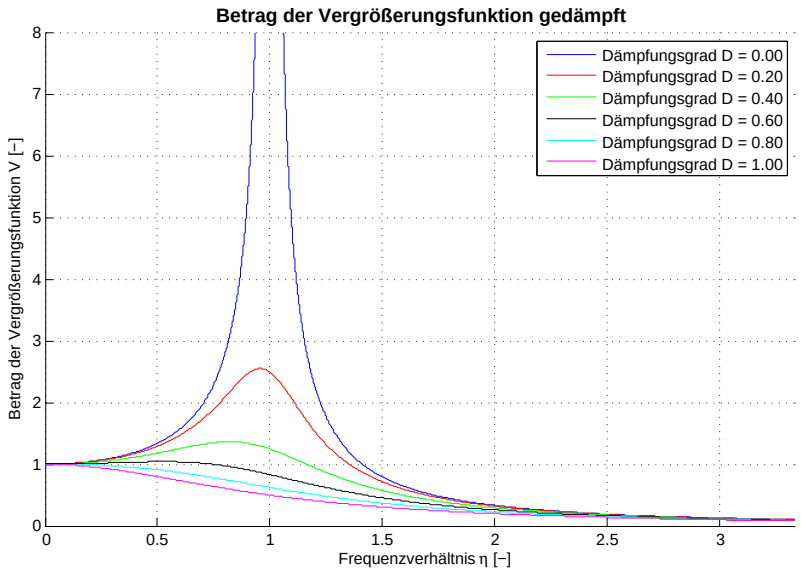


Abbildung 2.6: Betrag der Vergrößerungsfunktion der gedämpften Schwingung

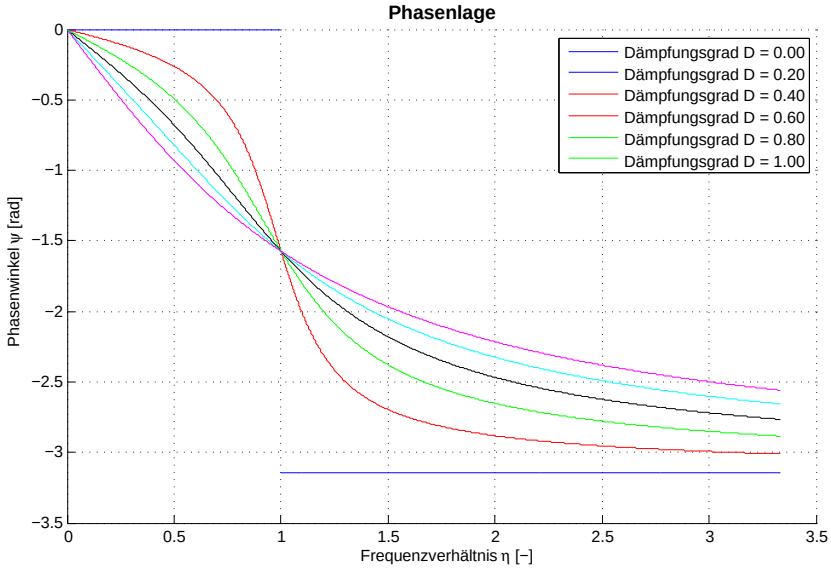


Abbildung 2.7: Phasenlage der gedämpften Schwingung

2.2 Nichtlineare Schwingungen

Im Wesentlichen unterscheiden sich nichtlineare von linearen dynamischen Systemen dadurch, dass in den Bewegungsgleichungen Terme auftreten, die nichtlineare Effekte in das System integrieren.

$$\Theta(...,\varphi,t,...)\ddot{\varphi} + d(...,\dot{\varphi},\varphi,t,...)\dot{\varphi} + c(...,\varphi,t,...)\varphi = M(...,\ddot{\varphi},\dot{\varphi},\varphi,t,...) \tag{2.67}$$

Als einfaches Beispiel dient der Duffing Schwinger (2.68).

$$\theta\ddot{\varphi} + d\dot{\varphi} + c_1\varphi + c_3\varphi^3 = M(t) \tag{2.68}$$

Die Gleichung (2.68) ähnelt der eines Einmassenschwingers wobei die Faktoren d und c_1 der Dämpfung und der Steifigkeit entsprechen. Neu hinzugekommen ist der Term $c_3\varphi^3$, der je nach dem Vorzeichen von c_3 eine Degression oder Progression der Steifigkeit bewirkt (siehe Abbildung 2.8).

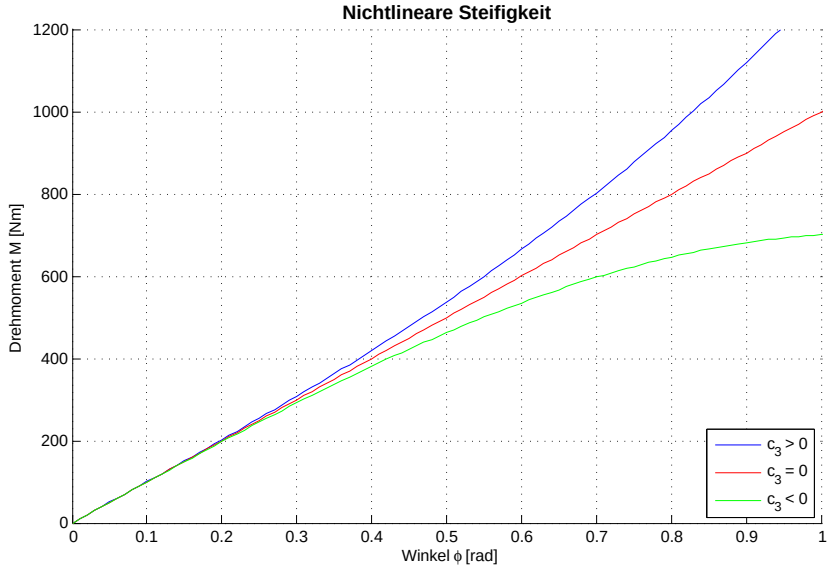


Abbildung 2.8: Nichtlineare Steifigkeit - progressiv und degressiv

Die Wirkung der nichtlinearen, progressiven Steifigkeit auf das dynamische System wird in Abbildung 2.9 dargestellt. Je größer die Amplitude wird, desto steifer verhält sich der Schwinger. Eine höhere Steifigkeit bedeutet jedoch gleichermaßen eine Anhebung der Eigenfrequenz. Im linearen Bereich befindet sich die Eigenfrequenz bei $\eta = 1$. Im progressiven Fall neigt sich die Vergrößerungsfunktion in Richtung steigendem η. Im degressiven Fall würde sich die Spitze sinngemäß in Richtung fallendem η neigen. Es gibt Bereiche, denen zwei Werte zugeordnet werden können. Welche Bewegung ausgeführt wird hängt hier im Wesentlichen von den Anfangsbedingungen und der Eingangfunktion $M(t)$ ab. Da es keine eindeutige Zuordnung zwischen der Antwort und η gibt, ist das Übertragungsverhalten genau genommen nicht mehr als Funktion darstellbar, und deshalb wird die Vergrößerungsfunktion als V^* bezeichnet.

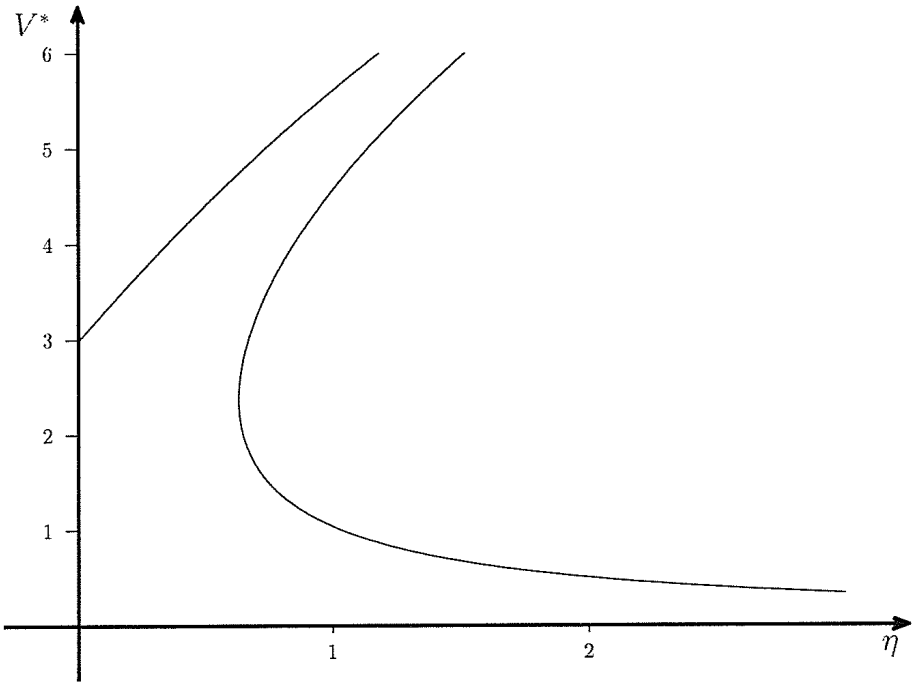


Abbildung 2.9: Vergrößerungsfunktion progressiver Duffing Schwinger [3]

Bei der numerischen Lösung des Duffing Schwingers mit stetig steigender bzw. sinkenden Anregungsfrequenz (aufsteigender bzw. abfallender Frequenz-Chirp) tritt bei Durchschreiten einer kritischen Frequenz ein hoher Amplitudensprung auf. Jedoch unterscheiden sich die kritischen Frequenzen von auf- und abwärts verlaufendem Frequenzdurchlauf deutlich (siehe Abbildung 2.10 und 2.11).

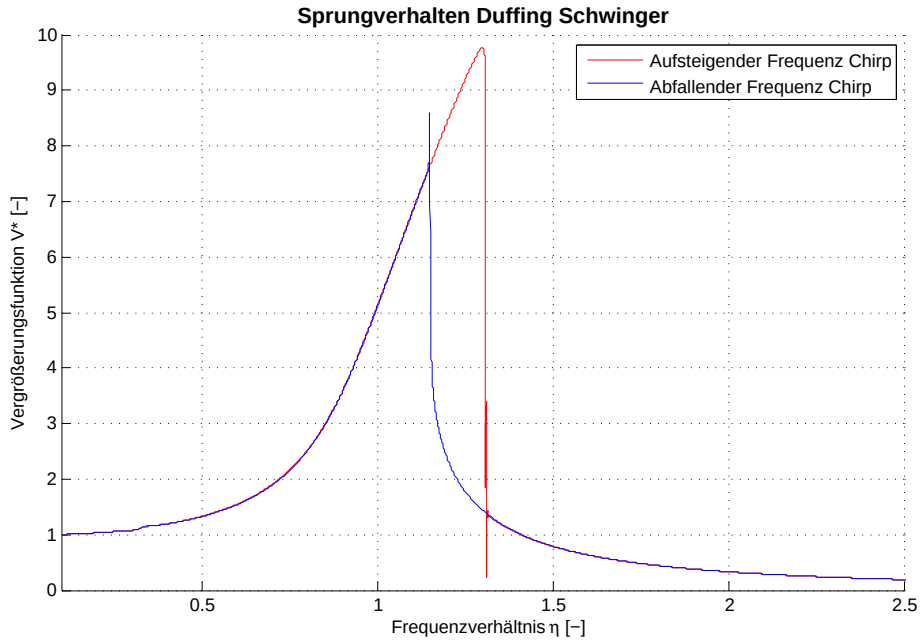


Abbildung 2.10: Numerisch berechnete Vergrößerungsfunktion progressiver Duffing Schwinger

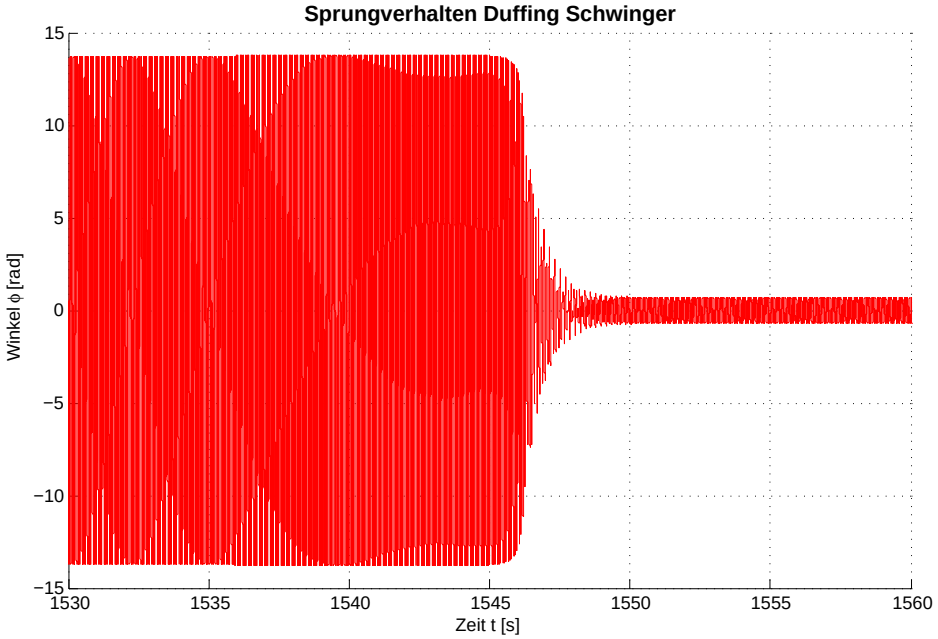


Abbildung 2.11: Numerisch berechnete Winkel progressiver Duffing Schwinger

Deshalb ist es im Allgemeinen nicht möglich, mit Frequenzantwortfunktionen zu arbeiten, wenn nichtlineare Schwingungen untersucht werden. Jedoch können Frequenzgänge bei schwach nichtlinearen Systemen zu hilfreiche Aussagen führen. Auch wenn kein direkter Zusammenhang zwischen der Frequenz und der Systemantwort besteht, kann eine Übertragungsfunktion analytisch angenä-

hert werden. Es ist zu beachten, dass bei verschieden Anfangsbedingungen unterschiedliche Systemstabilität möglich ist. Es kann bei nichtlinearen Systemen nicht davon ausgegangen werden, dass eine harmonische Anregung auch eine harmonische Systemantwort ergibt.

Nichtlineare Systeme können viele Phänomene aufweisen, die bei linearen Schwingern nicht auftreten können (vgl. [3]):

- Bei gleichen Systemparametern können verschiedene Bewegungen auftreten
- Die Systemantwort beinhaltet andere Frequenzen bei sub-harmonischen und super-harmonischen Schwingungen. Wird ein nichtlineares schwingfähiges System mit einer konstanten Frequenz Ω angeregt, entstehen sub-harmonische Frequenzen ($\frac{\Omega}{2}, \frac{\Omega}{3}, \frac{\Omega}{4}$) bzw. super-harmonische Frequenzen ($2\Omega, 3\Omega, 4\Omega$) in der Systemantwort.
- Es können chaotische Bewegungen entstehen. Dieses Verhalten ist stark von den Anfangsbedingungen abhängig.
- Einer periodischen Systemantwort muss nicht zwingend eine periodische Anregung zugrunde liegen.

Beim Übergang von linearen zu nichtlinearen Schwingungen ist zu beachten:

- Die analytische Lösung der Differentialgleichungen ist in den wenigsten Fällen bestimmbar. Die Berechnung erfolgt meistens numerisch.
- Es ist keine allgemeine Theorie mehr für alle nichtlinearen dynamischen Systeme vorhanden. Es gibt Methoden und Konzepte, die zielgerichtet auf bestimmte Klassen anwendbar sind.
- Im Allgemeinen hängt die Stabilität nicht mehr ausschließlich von der Systemgleichung, sondern auch von den Anfangsbedingungen ab.
- Die Berechnung im Frequenzbereich kann bei nichtlinearen Schwingungen nur eine Approximation des dynamischen Verhaltens liefern.

2.3 Dämpfung

Dämpfung wird als ein Mechanismus beschrieben, der schwingenden Systemen Energie entzieht. Diese Dissipation entspricht einer Energieumwandlung in Wärme, lokale plastische Verformungen oder vielen anderen Formen von Energieverlust. In Antriebssystemen entstehen Dämpfungskräfte hauptsächlich an den Kontaktflächen sich bewegender Körper, dies wird auch als äußere Dämpfung bezeichnet. Die inneren Dämpfungskräfte sind stark werkstoffabhängig. Häufig werden die

Dämpfungsdaten nur abgeschätzt bzw. aus Literaturquellen näherungsweise bestimmt. Um an verlässliche Dämpfungswerte zu kommen, müssen Messungen am realen System durchgeführt werden. Anhand dieser Messdaten kann ein linearisierter Ansatz gewählt werden, der eine hinreichend genaue Modellierung der Dämpfung gewährleistet, vgl. [4]. Im Allgemeinen hängen die Dämpfungskräfte in verschiedenster Weise von Kräften, Deformationen und deren Ableitungen ab. Wenn sich die Abhängigkeit der Dämpfung auf die Koordinate q und deren Ableitung $\dot{q}$ beschränkt, kann die Dämpfung wie in Gleichung (2.69) beschrieben werden. Grundsätzlich wirkt Dämpfung immer gegen die Bewegungsrichtung, um sicherzustellen, dass dem System mechanische Energie entzogen wird. Dieses Verhalten wird mathematisch durch die Signum Funktion beschrieben.

$$F_D = -|f(\dot{q}, q)|sign(\dot{q}) \tag{2.69}$$

Eine besondere Herausforderung stellt die Bestimmung der Dämpfungparameter dar, da diese oftmals nicht direkt gemessen werden können. Es muss dann auf indirekte Messverfahren zurückgegriffen werden, damit die Parameter rechentechnisch identifiziert werden können. Zur Bestimmung der Dämpfung werden verschiedene Methoden verwendet. Am bedeutendsten sind der Ausschwingversuch, ein Versuch mit harmonischer Anregung bzw. die Bestimmung des Dämpfungsgrades aus der Halbwertsbreite, siehe Abbildung 2.12. Beim Ausschwingversuch wird ein schwingfähiges System ausgelenkt und ohne Beeinflussung der zeitliche Verlauf der Schwingung aufgezeichnet. Bei harmonischer Anregung kann die Hysterese direkt über Last und Auslenkung bestimmt werden. Bei der Methode des Halbwertabfalles wird ein Körper einer Modalanalyse unterzogen. Dabei wird mittels Impulshammer eine Erregung aller Frequenzen durchgeführt. Mit Hilfe der gemessenen Inertanz[1], werden die Krümmungen der Resonanzspitzen bestimmt, indem die Werte des Frequenzgangs der linken und rechten Resonanzschulter bei Wert ($\frac{\hat{q}_{max}}{\sqrt{2}}$) ermittelt und ausgewertet werden. Eine Aufstellung der Verfahren und deren Berechnungsformalismen sind in Abbildung 2.12 zu sehen.

[1] Die Inertanz ist als $T(f) = \frac{a(f)}{F(f)}[\frac{m}{Ns^2}]$ definiert, und wird meist in der Akustik zur Bestimmung der Dämpfung herangezogen. Hierbei ist $a(f)$ die gemessene Beschleunigung und $F(f)$ die gemessene Kraft, die mittels Fast Fourier Transformation (FFT) in den Frequenzbereich transformiert wurden.

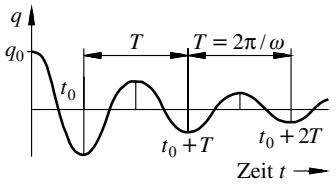
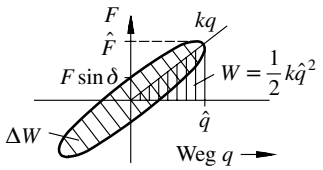
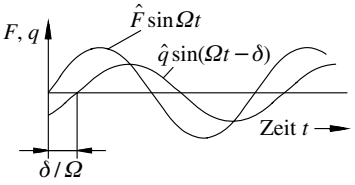
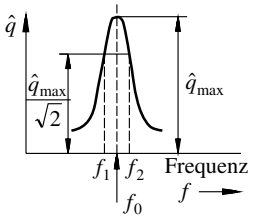
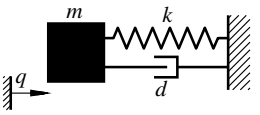
	Kenngröße	Herkunft, geometrische Größe
Ausschwingversuch	Logarithmisches Dämpfungsdekrement $\Lambda = \frac{1}{n} \ln \left\lvert \frac{q(t_0)}{q(t_0+nT)} \right\rvert$ (1)	Abklingkurve 
Erzwungene harmonische Schwingung	Relative Dämpfung $\psi = \frac{\Delta W}{W}$ (2)	Hysteresekurve 
	Verlustwinkel $\sin\delta = \frac{d\Omega\hat{q}}{\hat{F}} \approx \delta$ (3)	Zeitverlauf 
	Dämpfungsgrad aus Halbwertsbreite $D = \frac{f_2 - f_1}{2f_0}$ (4)	Resonanzkurve 
Parameterwerte	Dämpfungsgrad $D = \frac{d}{2\sqrt{km}}$ (5)	Berechnungsmodell 

Abbildung 2.12: Methoden zur Ermittlung von Dämpfungsparameter nach [5]

Mit der vereinfachenden Annahme einer rein viskosen Dämpfung $F_D = d\dot{q}$ ergibt sich folgende Beziehungen zwischen den Dämpfungskennwerten, wenn es sich um einen linearen Schwinger mit einem Freiheitsgrad handelt, siehe Abbildung (2.13).

	d	β	D	Λ	ψ	δ
Dämpfungskonstante d	d	$2m\beta$	$2m\omega D$	$\frac{m\omega\Lambda}{\pi}$	$\frac{k\psi}{2\pi\Omega}$	$\frac{k\delta}{\Omega}$
Abklingkonstante β	$\frac{d}{2m}$	β	ωD	$\frac{\omega\Lambda}{2\pi}$	$\frac{k\psi}{4\pi m\Omega}$	$\frac{k\delta}{2m\Omega}$
Dämpfungsgrad D (Lehrsches Dämpfungsmaß)	$\frac{d}{2\sqrt{km}}$	$\frac{\beta}{\omega}$	D	$\frac{\Lambda}{2\pi}$	$\frac{\psi}{4\pi\eta}$	$\frac{\delta}{2\eta}$
Logarithmisches Dämpfungsdekrement Λ	$\frac{\pi d}{m\omega}$	$\frac{2\pi\beta}{\omega}$	$2\pi D$	Λ	$\frac{\psi}{2\eta}$	$\frac{\pi\delta}{\eta}$
Relative Dämpfung ψ	$\frac{2\pi\Omega d}{k}$	$\frac{4\pi m\Omega\beta}{k}$	$4\pi\eta D$	$2\eta\Lambda$	ψ	$2\pi\delta$
Verlustwinkel δ	$\frac{\Omega d}{k}$	$\frac{2m\Omega\beta}{k}$	$2\eta D$	$\frac{\eta\Lambda}{\pi}$	$\frac{\psi}{2\pi}$	δ

Abbildung 2.13: Beziehungen zwischen den Dämpfungskennwerten nach [5]

Mathematisch ergeben sich deutliche Vorteile durch die linearen Differentialgleichungen. Diese Vereinfachungen werden oft als Näherung verwendet, obwohl durchaus bekannt ist, dass das reale Verhalten des Werkstoffes nichtlinear ist. Hierbei liegt der Fokus auf dem Detaillierungsgrad, der für die jeweilige Anwendung definiert sein muss, um hinreichend genaue Ergebnisse zu erzielen.

In Abbildung 2.14 sind Ansätze der Dämpfung vorgestellt, die in vielen Fällen zu den gewünschten Hystereseformen führen. Die dargestellte Hysterese wird durch eine harmonische Anregung $q = \hat{q}sin(\Omega t)$ erzeugt. Die Ansätze gelten auch für nicht harmonische Anregungen, da sie durch fixe Parameter an die Auslenkung gebunden sind. Somit kann durch ein weiteres Feld von Messdaten ein Kennfeld der Dämpfung erzeugt werden, welches in der Simulation berücksichtigt werden kann. Zur Bestimmung der äußeren Dämpfung werden in der Regel in der Bewegungsgleichung die Feder- (cq) und Dämpfungskräfte (F_D) geteilt.

$$m\ddot{q} + cq + F_D = \hat{F}sin(\Omega t) \tag{2.70}$$

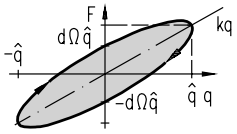
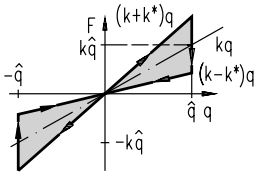
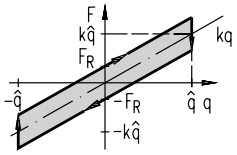
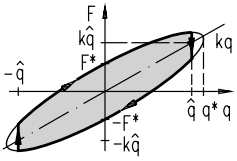
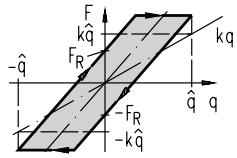
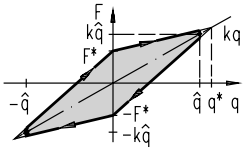
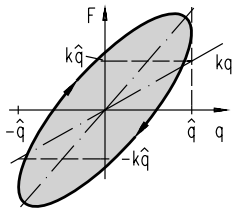
Ansatz für F_D	harmonische Bewegung $q = \hat{q} \sin \Omega t$, stationärer Zustand	
	Form der Hysteresekurve $(F = kq + F_D)$	relative Dämpfung ψ
Kelvin–Voigt $d \cdot \dot{q}$ (1)		$\frac{2\pi\, d\,\Omega}{k}$ (2)
Reid $k^* \cdot q \cdot \mathrm{sign}(\dot{q})$ (3)		$4\frac{k^*}{k}$ (4)
Coulomb $F_R \cdot \mathrm{sign}(\dot{q})$ (5)		$8\frac{F_R}{k\hat{q}}$ (6)
Sorokin $F^* \cdot \sqrt{1-\left(\frac{q}{q^*}\right)^2} \cdot \mathrm{sign}(\dot{q})$ $\lvert q \rvert \leqq q^*$ (7)		$\frac{4F^*q^*}{k\hat{q}^2} \cdot \left[\frac{\hat{q}}{q^*} \cdot \sqrt{1-\left(\frac{\hat{q}}{q^*}\right)^2} + \right.$ $\left. + \arcsin\left(\frac{\hat{q}}{q^*}\right)\right]$ (8)
Prandtl $F_R \cdot \mathrm{sign}\left(\dot{q} - \frac{\dot{F}_D}{k^*}\right)$ $\dot{q} \neq \frac{\dot{F}_D}{k^*}$ (9)		$\frac{8F_R}{k\hat{q}} \cdot \left(1-\frac{F_R}{k^*\hat{q}}\right)$ (10)
Kortschinski $\dot{q} \neq 0$: $F^* \cdot \left(1-\frac{\lvert q \rvert}{q^*}\right) \mathrm{sign}(\dot{q})$ $\lvert q \rvert \leqq q^*$ (11)		$\frac{8F^*}{k\hat{q}} \cdot \left(1-\frac{1}{2}\,\frac{\hat{q}}{q^*}\right)$ (12)
Maxwell $\dot{F}_D + \frac{k^*}{d} \cdot F_D = k^* \dot{q}$ (13)		$\frac{2\pi \cdot d\,\Omega}{k \cdot \sqrt{1+\left(\frac{d\,\Omega}{k^*}\right)^2}}$ (14)

Abbildung 2.14: Dämpfungsansätze nach [5]

Oftmals wird auch ein komplexer Ansatz gewählt, da dieser nicht nur bei harmonischer sondern auch bei allgemeiner periodischer Anregung anwendbar ist. Jedoch ist zu bedenken, dass ein solcher kein kausales Verhalten des Werkstoffes aufweist. Dadurch sind die Lösungen physikalisch falsch, da die Wirkung vor der Ursache auftritt. Jedoch bietet die komplexe Dämpfung einen gut geeigneten Ansatz für die Berechnung im Frequenzbereich, da der Ansatz eine geschwindigkeitsunabhängige Hysteresekurve liefert, vgl. [5].

Kapitel 3

Modellbildung

3.1 Allgemeines

Zur Simulation von dynamischen Effekten in mechanischen Systemen stehen viele unterschiedliche Softwarepakete zur Verfügung. Für diesen Zweck eignet sich Modelica, von der Modelica Organisation (www.modelica.org), besonders gut. Diese ist eine objektorientierte Simulationsumgebung für physikalische Modelle hybrider Systeme. Als Open Source Variante wird OpenModelica von der Universität Linköping Schweden angeboten. Mechanische, elektrische, hydraulische, thermische sowie signalbasierte Komponenten werden standardmäßig unterstützt. Die Schnittgrößen zwischen den einzelnen Komponenten stellen die physikalischen Schnittgrößen dar, die auch im realen System vorhanden sind. Beispielsweise ist ein Flansch einer Welle durch seinen Winkel und das wirkende Moment beschrieben. Durch die akausale Modellierung entstehen weitere Vorteile in der Modellierung, da keine Eventerkennung bei Richtungsumkehr vom Benutzer von Nöten ist. Weiters wird eine umfangreiche Standardbibliothek (www.modelica.org) angeboten, die für eigene Anwendungen durch eigene Bibliotheken erweiterbar ist. Durch die Möglichkeit, rein textuell in der Sprache Modelica zu modellieren, bietet sich auch eine Codegenerierung für sich wiederholende Problemstellungen an, um sehr schnell ein angepasstes Modell zu erhalten. Besonders eignet sich die Schnittstelle mit MATLAB, da auch die Ergebnisse im MATLAB nativen .mat Format ausgegeben werden. Im Gegensatz zu graphisch basierten Umgebungen, wie Simulink, lassen sich auch Änderungen besser über branchenübliche Sourcecodeverwaltungssoftware dokumentieren und sichern. Die Modellierung selbst erfolgt in den beschreibenden Grundgleichungen und muss nicht wie in Simulink üblich ausmodelliert werden. Das Aufstellen des Differentialgleichungssystems des Modells erfolgt durch den Modelica Compiler. Aus den gewöhlichen Differentialgleichungen (ODEs) wird C-Code generiert, der dann kompiliert und exekutiert wird, vgl. [6].

Das zu untersuchende Verhalten von nichtlinearen Elastomerkupplungen kann als eindimensionales Schwingungssystem in Drehrichtung beschrieben werden. Zur weiteren Bestimmung der Betriebsparameter ist oft auch eine FEM Analyse nötig, um die kritische Drehzahl, also die erste Biegeeigenfrequenz, zu bestimmen. Die Torsionseigenschaften können für die Anwendung auf Prüfstän-

den als Punktmassesystem angenommen werden und deren Verbindung mit Federsteifigkeiten und Dämpfungen modelliert werden.

Ein wesentlicher Punkt der Modellierung ist die Anregung des Prüflings, welche meist durch eine Verbrennungskraftmaschine erfolgt. Bei der Modellierung ist darauf zu achten, dass die Art der Anregung auf alle Typen von Motoren anwendbar sein soll. Das Spektrum der Anregung zeigt eine geometriebedingte Motorhauptordnung, d.h. durch die Kurbelwellenkröpfung, dem Ablauf der Verbrennung und der Motorart ist die Ordnung der Anregung bestimmt.

3.2 Motorprüfstand

Ein Motorprüfstand besteht im einfachsten Fall aus einem Motor, einer Belastungseinheit und einer Wellenverbindung mit einem elastischen Element. Es existieren auch komplexere Aufbauten für spezielle Test wie z.B.:

- Prüfstände mit Fahrzeugantriebssträngen
- Akustikprüfstände
- Schwenkprüfstände
- thermische Prüfstände
- Prüfstände mit Prüfstandsgetriebe für Motoren mit hohen Drehzahlen

Hier soll jedoch nur der klassische Aufbau im Umfang dieser Arbeit betrachtet werden.

In der Auslegung eines Wellenstranges spielt die elastische Kupplung eine Hauptrolle, da diese mit ihrer niedrigen Torsionssteifigkeit die Haupteinflussgröße für die erste Torsionseigenfrequenz ist. Die Kupplung muss auf den speziellen Anwendungsfall abgestimmt werden. Als Ziel der Abstimmung wird definiert, dass das mechanische Verhalten des Prüfstandes, dem des Fahrzeuges so ähnlich wie möglich sein soll. Ein uneingeschränkter Betriebsbereich des Prüfstandes wird nur erreicht, wenn die Eigenfrequenzen des mechanischen Systems mit ausreichendem Abstand außerhalb des gewünschten Drehzahlbandes liegen. In DIN 740 ist festgelegt, dass die Eigenfrequenz eine Entfernung zum Betriebsbereich von $\sqrt{2}$ aufweisen soll, da bei einer überkritischen Auslegung die Vergrößerungsfunktion der ungedämpften Schwingung ab diesem Abstand kleiner 1 wird, siehe Abbildung 2.6.

Der für diese Arbeit betrachtete Aufbau umfasst einen Otto 4 Zylinder Reihenmotor mit Turboaufladung, eine elastische Kupplung tectos t1000-800-2-WN, eine GKN Gleichlaufgelenkwelle CV15 und einen Dynamometer, siehe Abbildung 3.1. Die tectos t1000-800-2-WN ist eine spezielle Konstruktion einer Klauenkupplung in zweireihiger Ausführung mit einem WN Elastomergürtel. Eine detaillierte Beschreibung dieser Kupplung befindet sich in Kapitel 4.

Die Berechnung im Frequenzbereich, mittels des tectos Simulationswerkzeuges tProgress, liefert schon einige Aussagen über das Schwingungssystem. Hierbei wird zuerst eine Diskretisierung des vorliegenden Prüfstandes in ein Punktmassesystem vorgenommen. Die Diskretisierung ist in Abbildung 3.1 abgebildet.

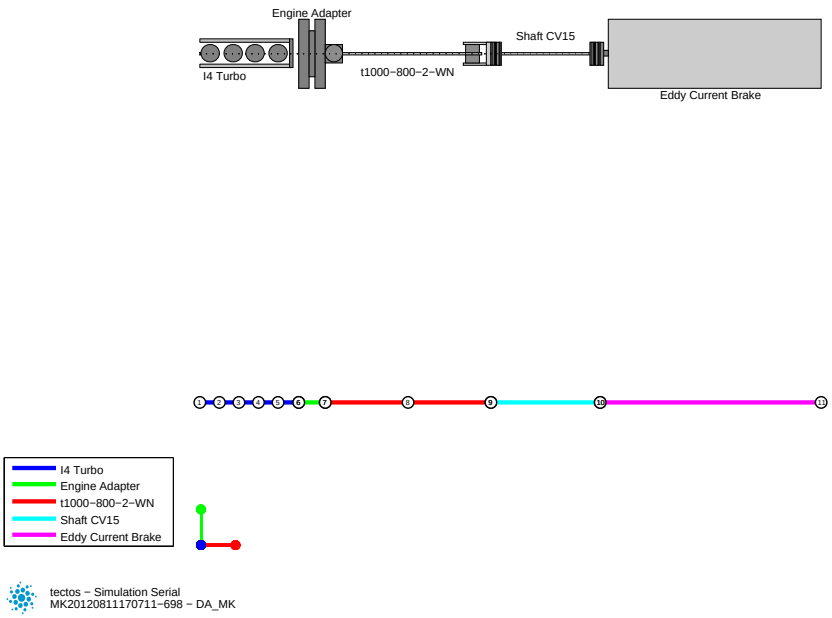


Abbildung 3.1: Aufbau Torsions Vibrations Analyse (TVA)

Die Verkettung erfolgt durch Zusammenlegung des letzten Massenträgheitsmoment des vorherigen Bauteils mit dem ersten Massenträgheitsmoments des folgenden Bauteils. Hierzu sind alle Massenträgheitsmomente, Torionssteifigkeiten, Dämpfungskonstanten und die Anregung zu bestimmen. Die Anregung wird über eine nulldimensionale Motorprozessrechnung mit den Motordaten ermittelt. Hierzu wird mit den Leistungs-, Drehmoment- und Kurbeltriebsdaten des Motors der Zylinderdruck rekonstruiert, vgl. [7]. Die nulldimensionale Motorprozessrechnung betrachtet den Brennraum als instationäres offenes System und liefert den Zylinderdruck mittels der Erhaltungssätze und Zustandsgleichungen der Arbeitsgase.

Die Analyse in tProgress teilt sich in zwei Hauptbereiche: die Modalanalyse und die Frequenzbereichsberechnung.

Die Modalanalyse liefert die Torsionseigenfrequenzen durch Lösen des Eigenwertproblems der Systemmatrix. Mit den entprechenden Eigenwerten können auch die dazugehörigen Eigenformen dargestellt werden. In Abbildung 3.2 und 3.3 sind die ersten beiden Eigenfrequenzen und deren normierte Eigenform dargestellt. Hierbei wird die berechnete Amplitude der Schwingungsform auf eine Konstante normiert, da nur die Schwingungsform und nicht deren Wert von Interesse ist. Die Richtung der Schwingung kann durch das Vorzeichen der Darstellung gedeutet werden. In Abbildung 3.2

ist deutlich der Schwingungsknoten in der Komponente t1000-800-2-WN und die Schwingung vom Motor gegen den Dynamometer erkennbar.

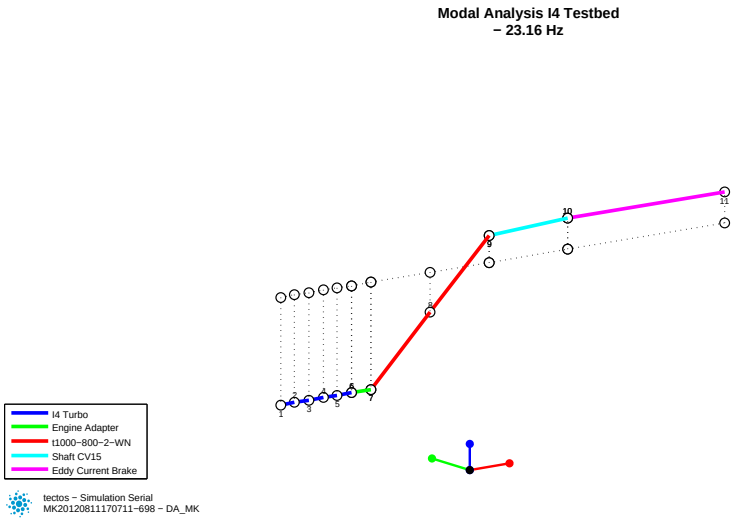


Abbildung 3.2: erste Torsionseigenform

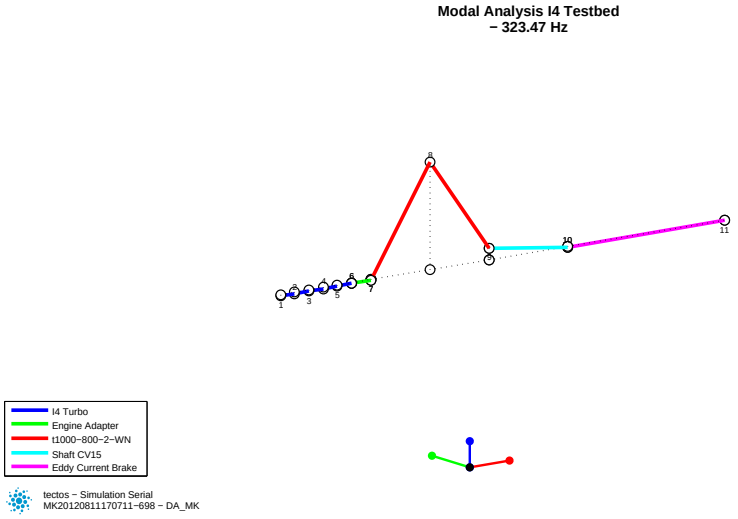


Abbildung 3.3: zweite Torsionseigenform

In der folgenden Torsions Vibrations Analyse (TVA) wird mit Hilfe des komplexen Frequenzganges und den Anregungen die Systemantwort der jeweiligen Frequenzschritte berechnet. Durch Berechnung aller Frequenzschritte können, wie in Abbildung 3.4 abgebildet, die auftretenden statischen und dynamischen Momente der motornahen Elastomerebene der t1000-800-2-SN dargestellt werden.

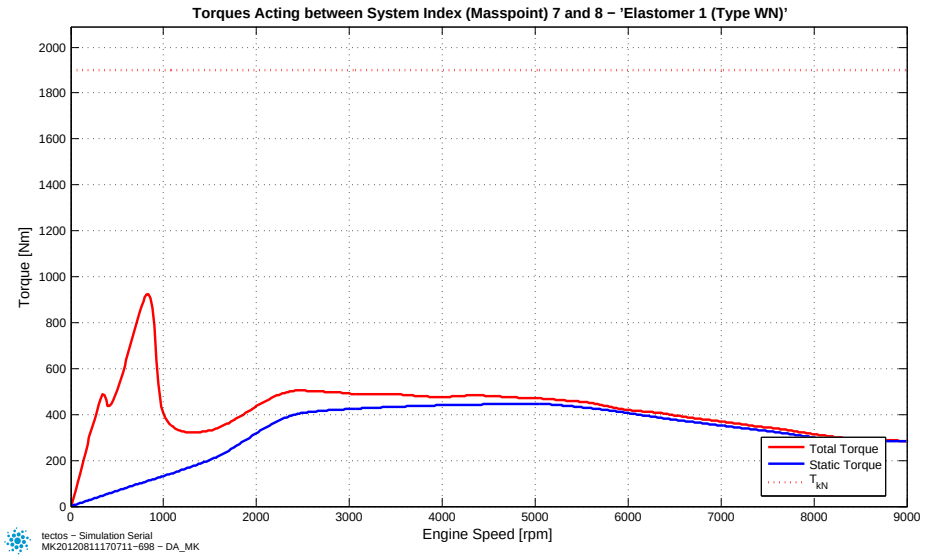


Abbildung 3.4: statische und dynamische Momente im ersten Elastomerring

Um eine Zuordnung der Drehmomentspitzen zu den Eigenfrequenzen bilden zu können, bedient man sich einer Spektralanalyse. Das errechnete Spektrum wird üblicherweise in einem Campbell Diagramm, auch Waserfalldiagramm genannt, dargestellt (Abbildung 3.5). In einem Cambell Diagramm werden die Eigenfrequenzen über der Drehzahl aufgetragen. In Farbe wird als die dritte Dimension, die Momentenamplitude in dB abgebildet und zeigt die zur Ordnung der Erregung gehörigen Amplituden, siehe [8].

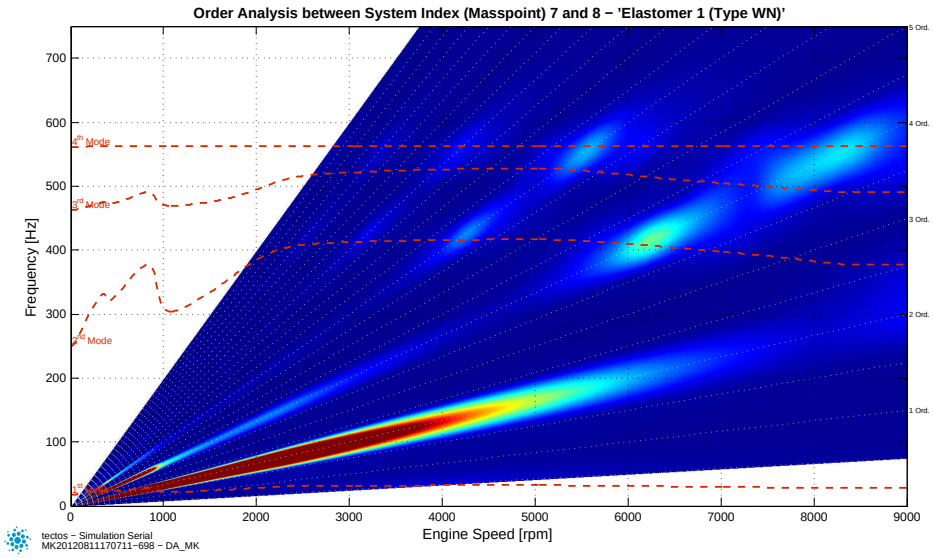


Abbildung 3.5: Campbell Diagramm des ersten Elastomerringes

Somit kann die Ursache der im Momentenverlauf ersichtlichen dynamischen Drehmomentspitzen den Eigenfrequenzen klar zugeordnet werden.

Um dynamische Effekte und eine detailliertere Dämpfung des System berücksichtigen zu können, erfolgt nun der Schritt in die zeitbasierte Simulation.

3.2.1 Modellierung in Modelica

Bei der Modellierung in Modelica wurde der Prüfstand entsprechend der TVA in einzelne Komponenten aufgeteilt: In einen Motor, eine Elastomerkupplung t1000-800-2, eine Gleichlaufgelenkwelle CV15 und einem AVL Indy Dynamometer. Das Modell der Frequenzbereichsrechnung muss im Zeitbereich mit einer Regelung erweitert werden, die den gewünschten Betriebspunkt einstellen kann.

Die Diskretisierung des Prüfstandes erfolgt gleich jener der Frequenzbereichsrechnung. Es wurden die dort verwendeten Parameter für Massenträgheitsmoment, Steifigkeit und Dämpfung übernommen, sofern es sich um linear modellierte Bauteile handelt.

3.2.2 Motor

Bei der Modellierung des Motors wurde der Punktmassenansatz durch die Kinematik des Kurbeltriebs mit dessen Massenkräften erweitert. Die Anregung erfolgt über eine Aufprägung der Gaskräfte und der Kompressionskräfte auf den Kolben. Auf Berücksichtigung von Ventiltrieb, Ladungswechsel und Motorsteuerung wurde verzichtet, da diese Betrachtungen für den Zweck der Simulation, der Auslegung von Wellensträngen mit Elastomerkupplungen, von untergeordneter Bedeutung sind. Bei einzelnen leistungsschwächeren Zylindern können niedrigere Frequenzen als die Hauptordnung der Motoranregung vorkommen. Auch spezielle Brennverfahren können durch stark abweichendes Zünden zum Standard Prozess andere Ordnungen der Anregung einbringen.

Die wichtigsten Thematiken der Motormodellierung sind das mechanische System und die Anregungsmodellierung, vgl. [5] und [9].

Um das Motorverhalten richtig abzubilden, wurden das Zylinderdruckkennfeld über Drosselklappenstellung, Drehzahl und Kurbelwinkel ermittelt und mit Hilfe der Geometrie und Kinematik des Motors umgesetzt.

Mechanisches System

Das mechanische System ist durch die Massenträgheit, Torsionssteifigkeit, Dämpfungen und Geometriedaten schon hinreichend genau modeliert. Ein Überblick über die Modellierung wird in Abbildung 3.6 gewährt.

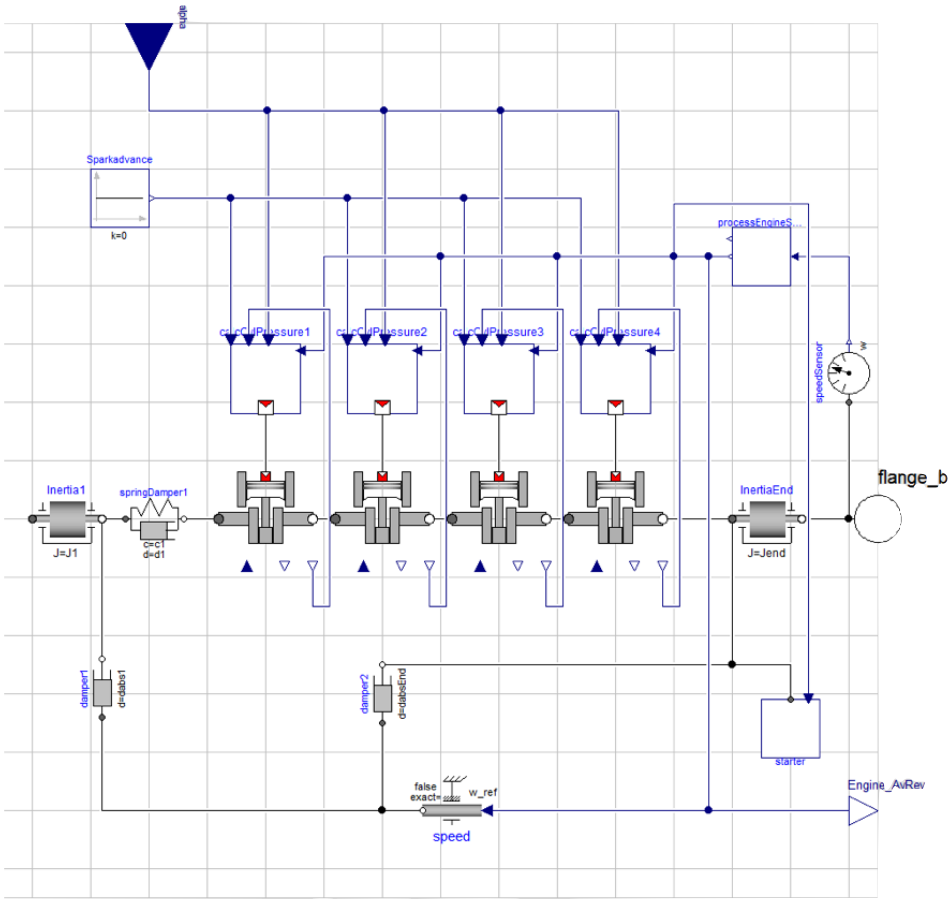


Abbildung 3.6: Aufbau I4T Motor in Modelica

Der Aufbau als Punktmassenmodell wird durch die Zylinder und deren Kurbeltrieb mit hinterlegtem Anregungsmodell erweitert.

Die Motordämpfung stellt eine besondere Herausforderung dar. Für diese liegen nur in den seltensten Fällen Messdaten vor. Deshalb ist man hier gezwungen, auf Näherungen und Faustformeln zurückzugreifen, vlg. [4], [10], [11] und [12].

Der Aufbau des Zylinders wurde so gestaltet, dass dieser in Reihe geschalten werden kann und nur durch die Parametrierung den jeweils benötigten Motortyp darstellt. Hierbei ist der Kurbelwinkelversatz bzw. der Zündwinkel der ausschlaggebende Parameter, da so die Zündreihenfolge des Motors bestimmt wird und so die Hauptordnungen der Anregung definiert werden. Die Geometriedaten wie Bohrung, Hub, Pleuellänge, Kompressionsverhältnis, Komponententrägheitsmomente und Masse werden für alle Zylinder als gleich angenommen. Als Variabilität ist ein Faktor im Zylinder vorgesehen, der eine prozentuale Über- oder Unterbewertung des Drehmomentes des jeweiligen Zylinders erlaubt, um so die auftretenden Anregungsordnungen der Messdaten ebenfalls in der Simulation berücksichtigen zu können.

Motordaten		
Parameter	Wert	Einheit
Hub r	80	$[mm]$
Bohrung b	$75,6$	$[mm]$
Pleuellänge l	$146,3$	$[mm]$
Kompressionsverhältnis	$12,5$	$[-]$
Kolbenmasse m_K	0.41	$[kg]$
oszilierende Masse m_{osz}	0.16	$[kg]$
rotierende Masse m_{rot}	0.31	$[kg]$
Zündwinkel	$360 - 540 - 180 - 0$	$[degKW]$

Tabelle 3.1: I4 Turbo Motordaten

Die Drehmomentberechnung erfolgt wie im realen Motor. Durch die Möglichkeit der hybriden Modellierung wurde eine selbst erstellte Schnittstelle, welche Druck als Transportgröße verwendet, erzeugt und eingesetzt. Somit kann durch ein Drucksignal direkt am Kolben die Gaskraft auf den Kolben berechnet werden. Die benötigten Motordaten werden in einer eigenen Komponente geladen und mittels Signal den einzelnen Komponenten bereitgestellt. Der Kurbeltrieb beschreibt den kinematischen Zusammenhang der Kurbelwelle mit dem Kolben und kann als

$$x = r + l - rcos(\varphi) + \sqrt{l^2 - r^2 sin(\varphi)^2} \tag{3.1}$$

angeschrieben werden, wobei die Variablen in Tabelle 3.1 definiert sind. Die benötigte erste und zweite Ableitung erfolgt nur noch über einen Ableitungsbefehl in Modelica. Die rotierende Massenträgheit wird an der Kurbelwellenseite berücksichtigt, die translatorische am Kolben. Hierfür wurden die jeweiligen Trägheitsterme $\theta\ddot{\varphi}$ bzw. $(m_{osz} + m_{Kolben})\ddot{x}$ angesetzt, vgl [13] und [14]. Mit diesen Daten kann nun die Kraftübertragung vom Kolben bis zur Kurbelwelle beschrieben und der entsprechende Drehmomentverlauf berechnet werden.

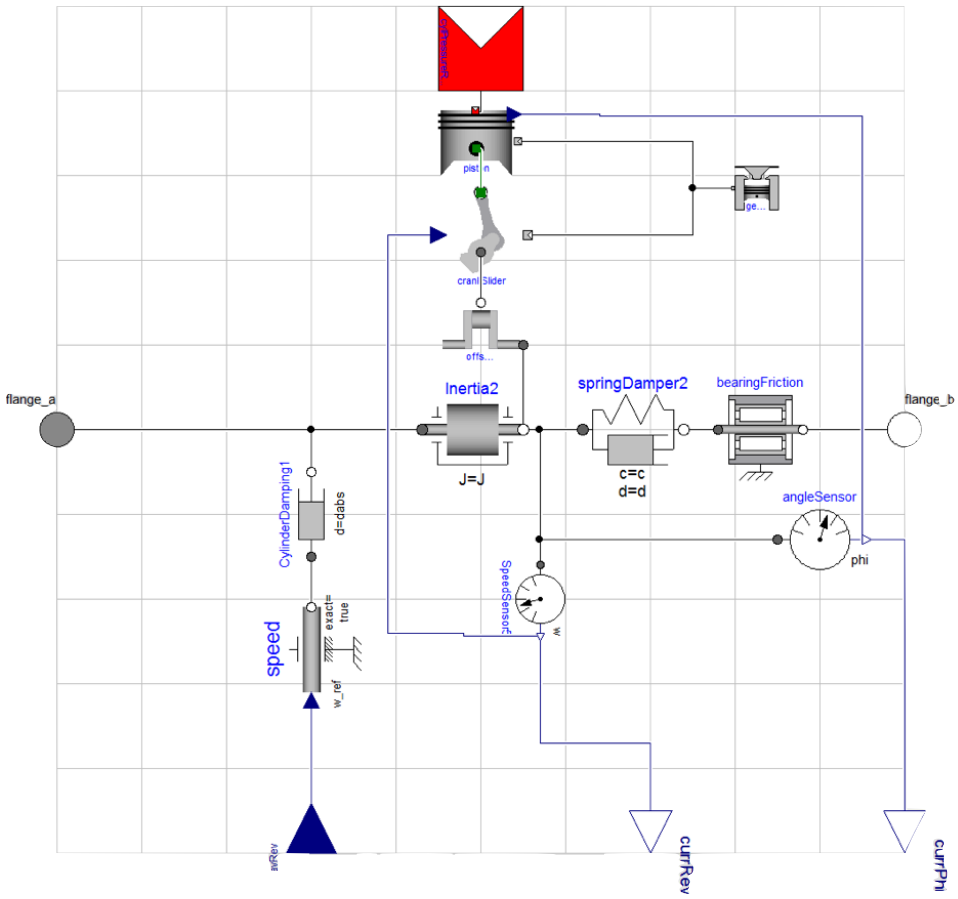


Abbildung 3.7: Kurbeltrieb und Zylinder in Modelica

Die Zuordnung des Zündwinkels erfolgt an der `OffsetShaft` Komponente in Abbildung 3.7, wo der Winkelversatz je nach Zündwinkel aufgeprägt wird, d.h., dass die Anregung für einen Winkel berechnet wird und das Aufbringen des Momentes an der Kurbelwelle per Phasenversatz erfolgt. Durch diese Art der Modellierung können sämtliche Motorformen (Reihe, V, W, VR) beschrieben werden. Das restliche mechanische System besteht aus Punktmassen und den jeweiligen Steifigkeiten und Dämpfungen. Zusätzlich wurde auch eine Motorlagerreibung (`bearingFriction`) berücksichtigt, siehe Abbildung 3.7. Diese errechnet sich aus dem über die Drehzahl nahezu konstanten Reibmitteldruck p_r, welcher bei ca. 1 bar liegt. Dieser Reibmitteldruck beinhaltet die mechanischen Verluste des Motors, vgl [13]. Weiters ist p_i und p_e als indizierter und effektiver Mitteldruck, V_H als Hubvolu-

men, W_i und W_e als Effektive und innere Arbeit, M_e und p als Zylinderdruck definiert.

$$p_r = p_i - p_e \tag{3.2}$$

$$p_i = \frac{W_i}{V_H} = \frac{\int pdV}{V_H} \tag{3.3}$$

$$p_e = \frac{W_e}{V_H} \tag{3.4}$$

$$p_e = 4\pi \frac{M_e}{V_H} \text{(für 4-Takt Motoren)} \tag{3.5}$$

$$p_e = 2\pi \frac{M_e}{V_H} \text{(für 2-Takt Motoren)} \tag{3.6}$$

Nachfolgend ist eine Aufstellung der Parameter des Motormodells abgebildet, die das Punktmassesystem bedatet.

Massendaten				Steifigkeitsdaten					
Masse Nummer	System Index	Massenträgheitsmoment $[kgm^2]$	Abs. Dämpfung $[Nms/rad]$	Feder Nummer	Connection	Steifigkeit $[Nm/rad]$	Rel. Dämpfung $[Nms/rad]$	Mat. Dämpfung $[\psi]$	Übersetzung [-]
1	1	$1.100e-03$	0	1	1 – 2	$5.872e+05$	0.5	0.04	1
2	2	$3.850e-03$	1.5	2	2 – 3	$3.320e+05$	0.5	0.04	1
3	3	$3.850e-03$	1.5	3	3 – 4	$3.515e+05$	0.5	0.04	1
4	4	$3.850e-03$	1.5	4	4 – 5	$3.320e+05$	0.5	0.04	1
5	5	$3.850e-03$	1.5	5	5 – 6	$5.700e+05$	0.5	0.04	1
6	6	$2.270e-02$	1						

Tabelle 3.2: I4 Turbo Motordaten Punktmassensystem

Anregungsmodelle

Die Anregung muss für jeden Betriebspunkt vorhanden sein. Dafür wurde eine 0 dimesionale Motorprozessrechnung eingesetzt, um über Drehzahl und Drosselklappenstellung ein Motorkennfeld (Abbildung 3.8) zu generieren. Diese betrachtet den Brennraum als instationäres offenes System, und liefert den Zylinderdruck mittels der Erhaltungssätze und Zustandsgleichungen der Arbeitsgase.Diese Abbildung beschreibt den Zylinderdruck im Zünd OT (Oberer Totpunkt). Zur Berechnung der Zylinderdrücke wurde das Verfahren von Vibe gewählt [7]. Diese Kennfeldermittlung wurde bereits in die tectos eigene Simulationssoftware tProgress integriert. Es wurden jeweils ein Kennfeld für die Zylinderkompression und die reine Verbrennung erstellt. Somit kann nun jeder Belastungszustand aus Kombination der Verbrennung und Kompression hergestellt werden. Auch kann aktiv in den Druckverlauf eingegriffen werden, um Effekte wie Zylinderabschaltung, Zündwinkeländerung bis hin zum Power Cut für Schaltvorgänge zu simulieren. Dieser Aufbau liefert den höchsten Grad an Flexibilität.

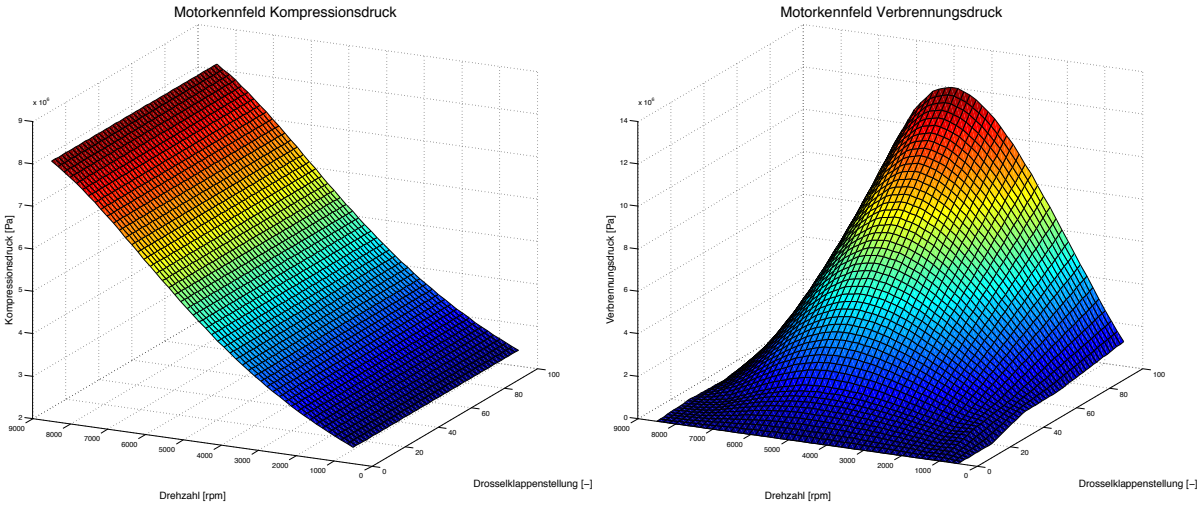


Abbildung 3.8: Motorkennfeld - Kompression (li.) und Verbrennung (re.) - über Drehzahl und Drossenklappenstellung beim oberen Totpunkt(OT)

Es ist auch möglich, gemessene Zylinderdruckindizierdaten zu verwenden, um alle Feinheiten des zu testenden Motors zu berücksichtigen, wenn dies erwünscht bzw. gefordert ist.

Die mehrdimensionale Interpolation stellte eine besondere Herausforderung in der Umsetzung in Modelica dar. In der Standard-Bibliothek ist keine solche Funktion vorhanden. Die Umsetzung dieser Funktionalität wird in Abschnitt 5.1.1 beschrieben. Es wurde versucht, die Implementation der Funktion in Modelica intern durchzuführen. Da aber die Rechenzeit der Interpolation in Modelica nicht den gestellten Ansprüchen genügte, wurde eine C-Funktion erstellt, welche durch die dafür vorgesehene Schnittstelle mit OpenModelica eingehängt wurde. Wie zu erwarten war, verarbeitet die externe C-Funktion den Algorithmus sehr performant.

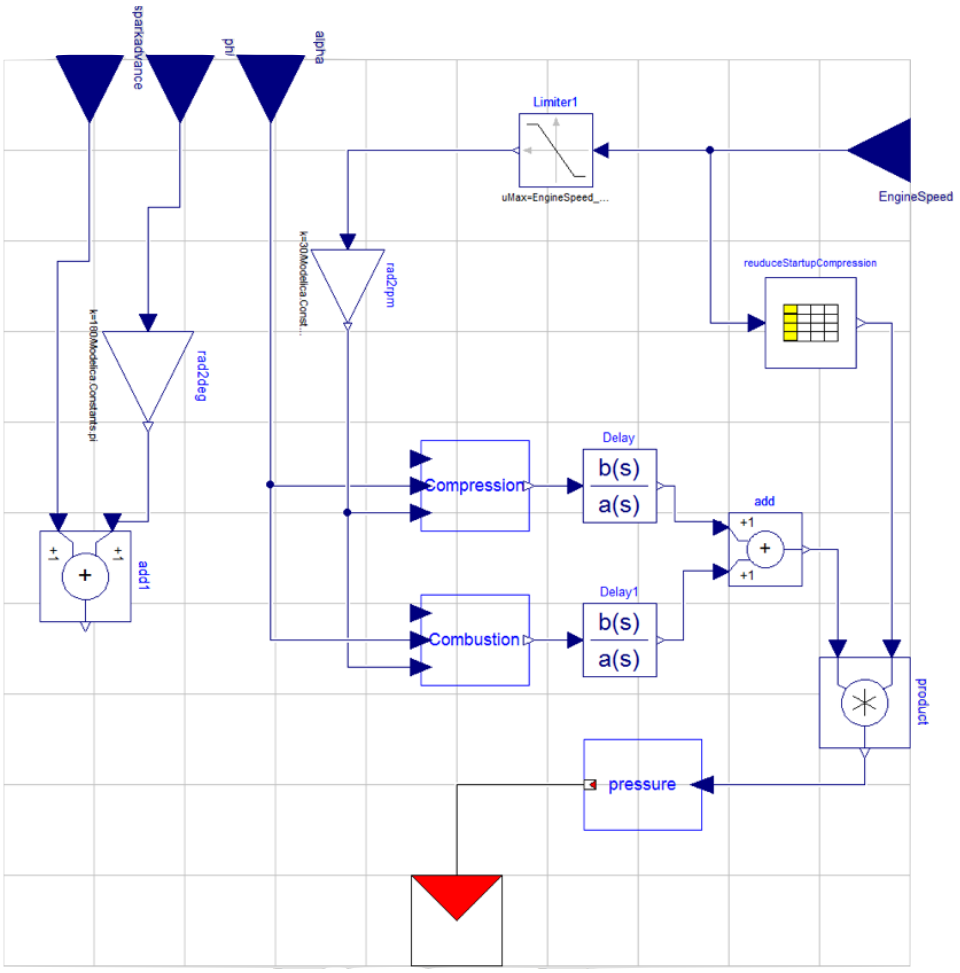


Abbildung 3.9: Zylinderdruckberechnung in Modelica

Mit dem definierten Motorkennfeld kann nun mit Hilfe der selbst entwickelten dreidimensonalen Interpolation jeder gewünschte Betriebszustand eingestellt werden. Als Eingangsgrößen dienen die Drosselklappenstellung in $\%$, die mittlere Motordrehzahl in rpm und der Kurbelwinkel in rad. Die Motordrehzahl wird am Schwungrad abgegriffen und durch einen Filter zur mittleren Motordrehzahl verarbeitet. Dies impliziert zwar eine Phasenverschiebung, die aber aufgrund der realen Motorträgheit als akzeptabel angesehen werden kann. Bezüglich des Kurbelwinkels wurde mit Hilfe der Modulo Funktion eine Sägezahnfunktion für den Kurbelwinkel von 0 bis 720° erstellt, damit die Interpolation den geforderten Verlauf von 0 - 720° KW erhält. Zusätzlich wurden Funktionen zur Detaillierung der Anregung berücksichtigt, wie z.B. einer Einstellung des Zündwinkels, der eine Verschiebung des Druckverlaufes zur Folge hat. Auch eine Möglichkeit, zylinderselektiv die Anregung eines Zylinders betragsmäßig zu erhöhen oder zu senken, wurde implementiert. Dieses Verhalten tritt sehr oft auf, da die Strömung in die Zylinder eines Mehrzylindermotors durch unterschiedliche Krümmungsradien und verschiedene Längen in den fluidführenden Leitungen zu unterschiedlicher Ladung der Zylinder führt. Die somit schwächeren Zylinder sorgen für weitere Anregungsordnungen. Aufgrund numerischer Probleme beim Starten des Motors wurde ein Bypass der Kompression angewendet, um ein freies Starten des Motors zu erreichen. Es wurde mit Hilfe einer Tabelle das Signal des Zy-

linderdruckes für die ersten 0.2s mit 0 multipliziert, um nicht schon zu Beginn eine Anregung ins System zu bekommen. Dies ist zulässig, da der Motorstart nicht Teil des zu untersuchenden Tests ist.

Die Gaskraft, die auf den Kolben wirkt, wird durch Addition des Kompressions- und Verbrennungsdrucks erstellt, welche über eine Übertragungsfunktion von $G_{(s)} = \frac{1}{1+0.0001s}$ geführt wird, um einen sehr ruckartigen Anstieg des Zylinderdruckes zu entschärfen.

3.2.3 Welle

Die verbaute Welle entspricht einer handelsüblichen Gleichlaufgelenkwelle und kann durch die Katalogwerte der Fa. GKN beschrieben werden. Bezüglich der Dämpfung konnte auf Daten der Firma tectos zurückgegriffen werden. Die nachfolgende Tabelle beinhaltet die verwendeten Parameter.

Massendaten				Steifigkeitsdaten					
Masse Nummer	System Index	Massenträgheitsmoment $[kgm^2]$	Abs. Dämpfung $[Nms/rad]$	Feder Nummer	Connection	Steifigkeit $[Nm/rad]$	Rel. Dämpfung $[Nms/rad]$	Mat. Dämpfung $[\psi]$	Übersetzung [-]
1	9	$5.050e-03$	0	1	$1-2$	$7.400e+04$	0	0.01	1
2	10	$5.050e-03$	0						

Tabelle 3.3: Gleichlaufgelenkwelle CV15

3.2.4 Kupplung

Der mechanische Aufbau der nichtlinearen Elastomerkupplung unterscheidet sich durch zwei Teilbereiche von einer üblichen Modellierung von Punktmassensystemen. Erstens durch das nichtlineare Feder-Dämpfersystem und zweitens durch den Drehzahlsignaleingang. Die Herleitung und Parameterbestimmung der Komponentenbestückung werden in Kapitel 4 detailliert diskutiert. Grundsätzlich wird die Steifigkeit über ein ermitteltes Polynom dritter Ordnung angenähert und die Dämpfung als Kombination von Kelvin-Voigt-, Coulomb- und Reid-Dämpfung angesetzt, siehe Abschnitt 2.3.

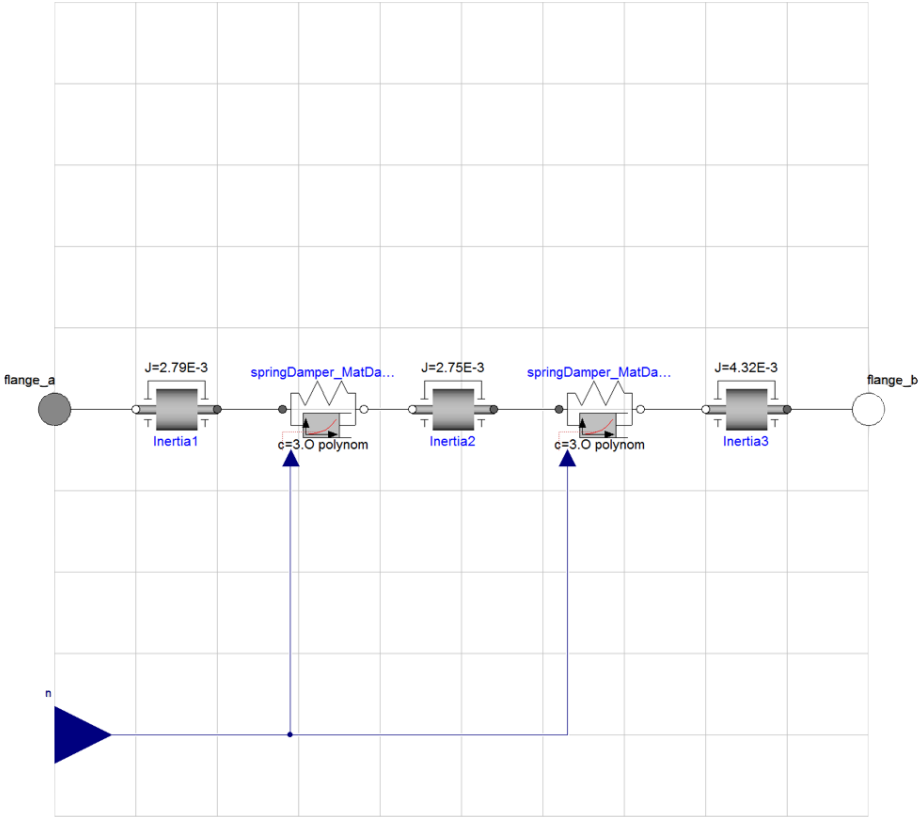


Abbildung 3.10: Aufbau t1000-800-2 in Modelica

Massendaten			Steifigkeitsdaten				
Masse Nummer	System Index	Massenträgheitsmoment $[kgm^2]$	Feder $[Nms/rad]$	Nummer	Steifigkeit Connection	Dämpfung $[Nm/rad]$	Übersetzung [-]
1	7	$7.851e-03$	1	$1-2$	$3.173e5\varphi^3 + 2.159e3\varphi$	$d(M)$ siehe Kapitel 4.4.2	1
2	8	$2.138e-03$	2	$2-3$	$3.173e5\varphi^3 + 2.159e3\varphi$	$d(M)$ siehe Kapitel 4.4.2	1
3	9	$4.266e-03$					

Tabelle 3.4: t1000-800-2-WN

Bei der Simulation treten numerische Schwierigkeiten auf, da die Dämpfungen beim Vorzeichenwechsel von $\dot{\varphi}$ eine sehr hohe Steigung aufweisen, siehe Abbildung 3.11. Deswegen wurde eine PT1 System zur Filterung von $\dot{\varphi}$ eingeführt, wobei die Zeitkonstante des PT1-Gliedes als Funktion der Drehzahl zu $\frac{7 \cdot 10^{-2}}{(2n\pi)}$ gesetzt wurde, um eine Anpassung an die dynamischen Verhältnisse zu ermöglichen. Ohne eine Entschärfung dieses Verhaltens wurden extrem kleine Schrittweiten, bis hin zu 10^{-9} s, in der Simulation festgestellt. Die Auswirkung dieses Behelfs ist in Abbildung 3.11 ersichtlich, wo der typische Verlauf einer Sättigungscharakteristik eines PT1 Systems nach dem Nulldurchgang zu erkennen ist.

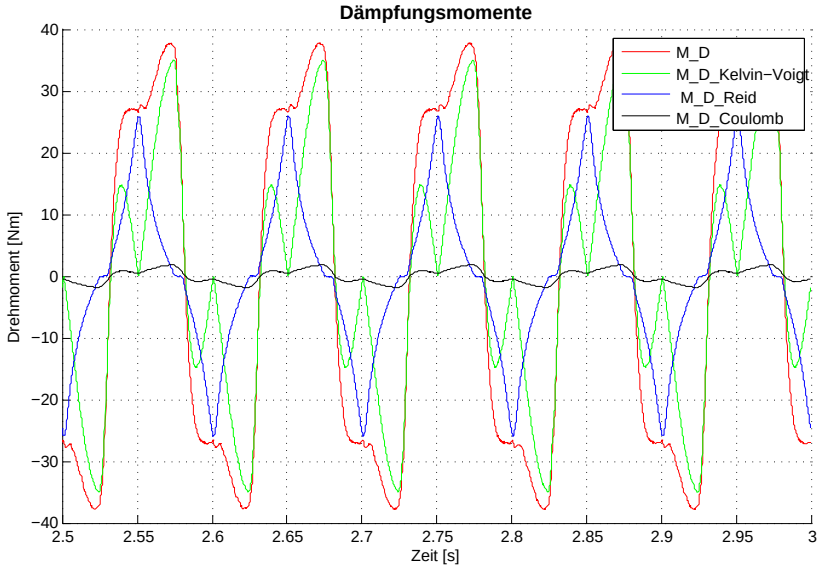


Abbildung 3.11: Dämpfungskräfte über Zeit

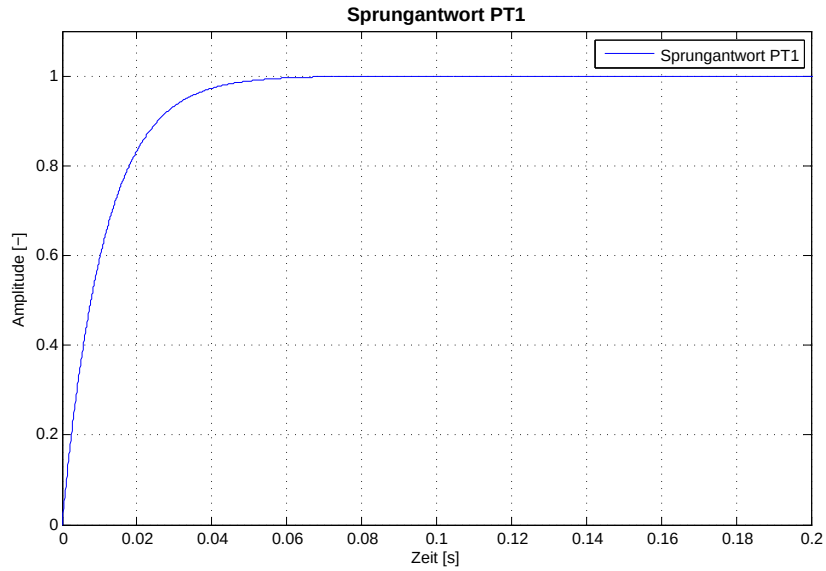


Abbildung 3.12: /Sprungantwort PT1

Die Dämpfungsbeiwerte des Ansatzes sind als lineare Regression über das anliegende Drehmoment vorhanden. Diese werden mittels einer Geradengleichung und dem anliegenden Drehmoment berechnet und verwendet. Um eine unnötige hochfrequente Dynamik der Dämpfung in der Kupplung zu unterbinden, wird das Moment mittels eines ZeroOrderHold Elements für $10^{-3}\ s$ konstant gehalten. Die Dynamik der Kupplung wird nicht negativ beeinflusst, da die höchsten auftretenden Frequenzen bei 750 Hz angenommen werden können. Im realen System ist die Dämpfung zu hoch, um höherfrequente mechanische Phänomene beobachten zu können.

Die Modellierung des Feder-Dämpfersystems erfolgt aufgrund besserer Übersichtlichkeit nur mehr

auf textueller Ebene und ist unter Code Listing A.4 einsehbar. Grundsätzlich werden die Dämpfungsmomente der einzelnen Ansätze gebildet und der folgenden Punktmasse aufgeprägt. Gleiches gilt für die Steifigkeitsmomente.

3.2.5 Dynamometer

Das mechanische System des Dynamometers wird nach Katalogwerten bedatet. Auf eine genauere Modellierung der E-Maschine wird bewusst verzichtet, da der Detaillierungsgrad für die vorliegende Betrachtung hinreichend genau ist. Es wird, wie in Abbildung 3.13 ersichtlich, mittels einer Drehmomentquelle aus der Modelica-Bibliothek das Stellmoment des Reglers auf den Dynamometer aufgebracht. Weiters wird das Drehzahlsignal an der letzen Punktmasse abgegriffen, welches nach außen zur Verfügung gestellt wird.

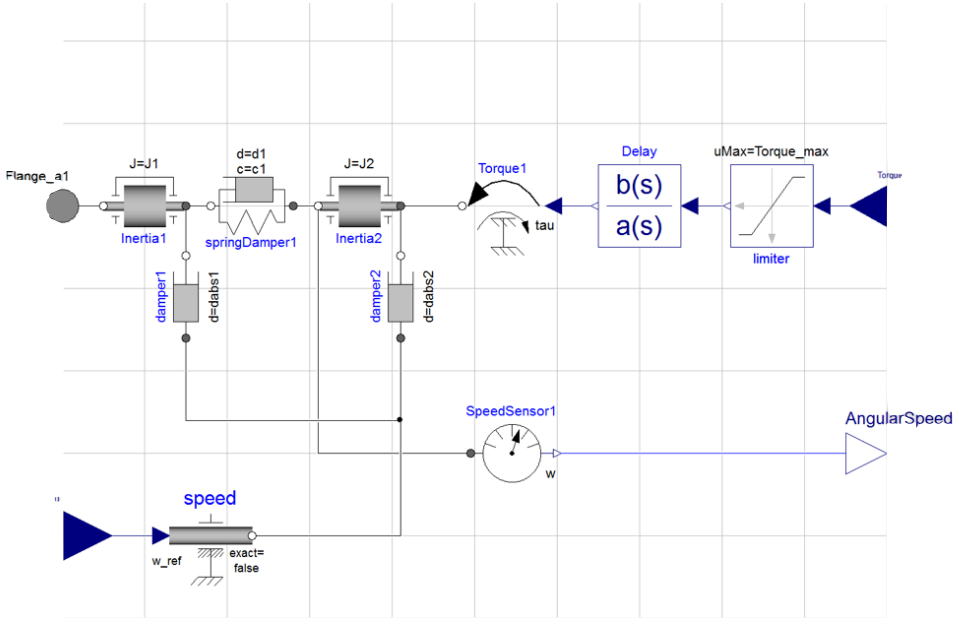


Abbildung 3.13: Aufbau Dynamometer in Modelica

Massendaten				Steifigkeitsdaten					
Masse Nummer	System Index	Massenträgheitsmoment $[kgm^2]$	Abs. Dämpfung $[Nms/rad]$	Feder Nummer	Connection	Steifigkeit $[Nm/rad]$	Rel. Dämpfung $[Nms/rad]$	Mat. Dämpfung $[\psi]$	Übersetzung [-]
1	10	$2.300e-01$	0.1	1	$1-2$	$5.000e+08$	0.01	0.01	1
2	11	$2.300e-01$	0.1						

Tabelle 3.5: AVL Indy Dynamometer

3.2.6 Regler

Es wurde eine Implementierung mit einem PI Regler gewählt, da diese für den hier benötigten Einsatz ausreichend ist. Aufgrund des Betriebs nur in Stationärpunkten bzw. das Durchfahren leichter Rampen, stellte sich die Parametrierung des Reglers als einfach heraus.

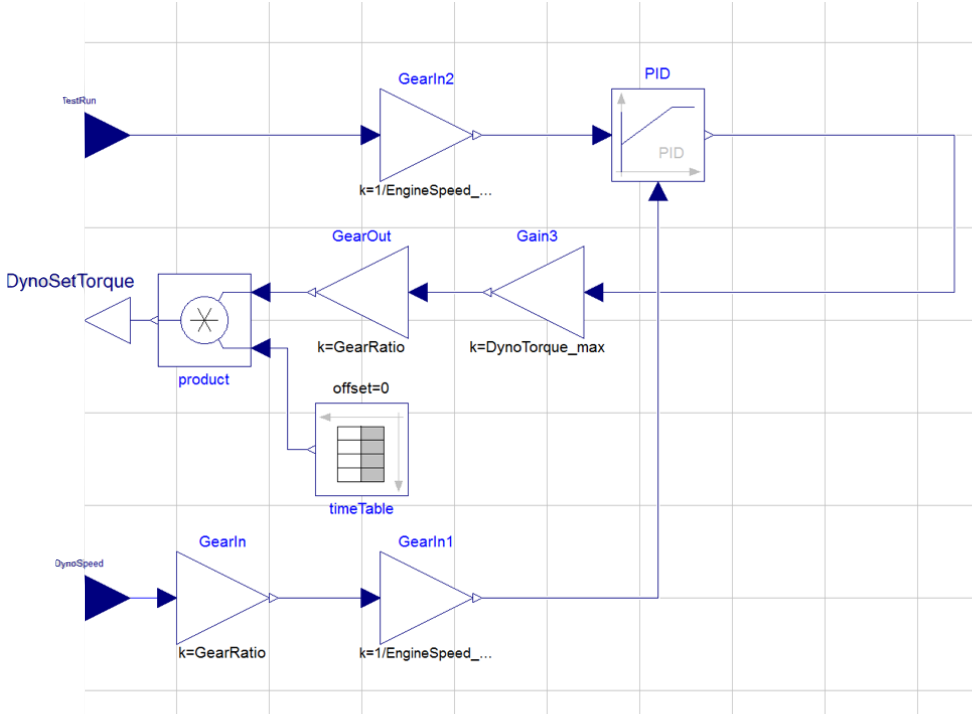


Abbildung 3.14: Aufbau Regler in Modelica

Um eine möglichst hohe Flexibilität zu erreichen, wurden die Stellgrößen und die Regelgrößen auf die Nominalwerte des Dynamometers normiert. Diese Modellierung erreicht ein gleichwertiges Verhalten unabhängig von der Größe der Signale und kann somit für verschieden große Maschinen angewendet werden. In Abbildung 3.14 ist für den Regler ein PID Regler eingesetzt, dessen differentieller Anteil jedoch auf Null gesetzt wurde. Die Regelstrategie umfasst das Moment als Drehzahlstellgröße und die Motordrehzahl als Sollgröße.

3.3 Modellaufbau

Der Gesamtaufbau des Motorprüfstandes ist in Abbildung 3.15 zu sehen und zeigt auch die benötigten Signale für das Pre- und Postprocessing.

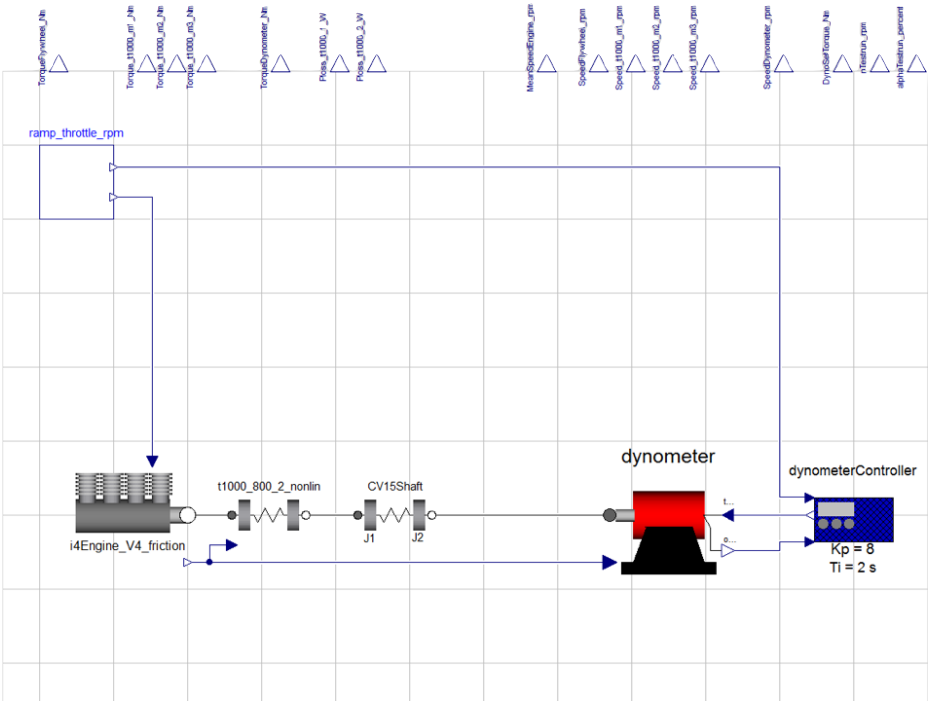


Abbildung 3.15: Aufbau gesamter Prüfstand in Modelica

Wie in Abschnitt 5.2 beschrieben, wird vor dem Kompilieren des Modells eine vordefinierte Auswahl an Parametern festgelegt, die in der Ergebnisdatei abgespeichert werden. Diese werden als Ausgang auf der Hauptebene des Modells textuell einem Ausgang zugewiesen. Dieser Eingriff in die Ergebnisverwaltung ist unter anderem deswegen nötig, da die Dateigröße des Ergebnisfiles beschränkt ist. Dieses Verhalten ist von der Problemgröße abhängig und ein bekanntes Problem des OpenModelica Compilers. Theoretisch sollte eine Ergebnisdateigröße von 4096MB möglich sein, jedoch berichten Anwender von 800MB oder weniger, bevor das Deiteiformat defekt wird.

Die Vorgabe des Betriebspuntkes erfolgt mittels einer Prüflaufkomponente. Diese erzeugt die gewünschten Drehzahl- und Drosselklappensollwerte über der Zeit mit Hilfe von Source-Blocks der Modelica-Bibliothek.

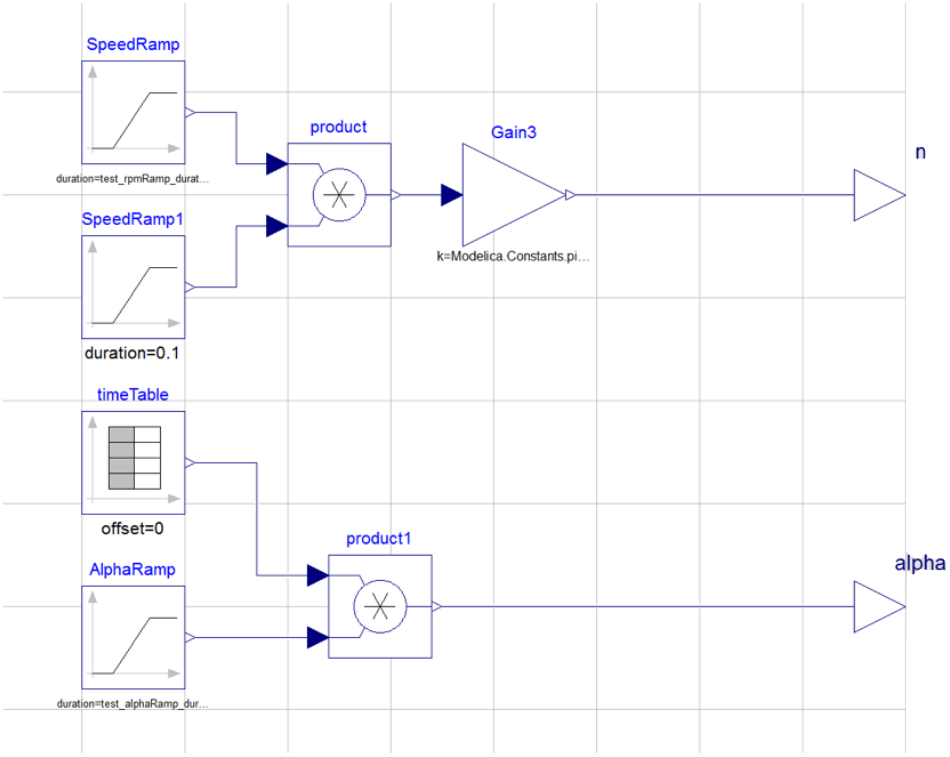


Abbildung 3.16: Aufbau Prüflaufvorgaben in Modelica

Für den Versuch ist es wichtig, bei vorgegebener Drehzahl den Motor in einem bestimmten Lastpunkt zu betreiben. Durch die Regelung auf Drehzahl des Dynamometers und die Stellung der Drosselklappe, wird der gewünschte Lastpunkt eingestellt. Diese Art der Regelung wird n/α Regelung genannt und ist eine übliche Betriebsart von Motorprüfständen.

Kapitel 4

Nichtlineare Elastomerkupplungen

4.1 Allgemeines

Nichtlineare Elastomerkupplungen in Antriebssträngen besitzen einen bedeutsamen Vorteil gegenüber linearen Kupplungen. Sie können aufgrund ihrer Nichtlinearität im Bereich um eine Eigenfrequenz eingesetzt werden, ohne einen sofortigen Schadensfall zu riskieren. Durch die Nichtlinearität der Steifigkeit kann kein stationärer Zustand im Bereich der Eigenfrequenz erreicht werden, da ein Ansteigen der Amplitude eine Änderung der Eigenfrequenz zur Folge hat. Dieses Verhalten ist besonders jetzt in Zeiten von Downsizing und Downspeeding von Bedeutung, da die klassische Art der überkritischen Auslegung der Wellenverbindung oftmals physikalisch unmöglich wird.

Anhand der t1000 Kupplung der tectos gmbh sollen das Steifigkeits- und Dämpfungsverhalten bestimmt und modelliert werden. Der Aufbau dieser Kupplung ist in der Explosionszeichnung (Abbildung 4.1) ersichtlich.

Abbildung 4.1: Explosionszeichnung einer t1000-800-2

Grundsätzlich ist die Kupplung als Klauenkupplung mit geschlossenem Volumen konzipiert. Der Gummi in jeder Zelle ist somit in seiner räumlichen Ausdehnung bei Torsionsbelastung eingeschränkt. Durch die sehr hohe Poissonzahl[1] des Werkstoffes von $\nu = 0.49$ verhält sich der Werkstoff nahezu inkompressibel. Als Folge tritt bei steigender Belastung eine Progressivität der Torsionssteifigkeit ein, da das freie Volumen in der Zelle begrenzt ist, bis das Elastomer die Zelle vollkommen ausfüllt.

Durch die gelagerte Konstruktion der drei relativ zueinander drehbaren Scheiben kann sogar bei Versagen der Elastomere die Verbindung zwischen beiden Seite aufrecht erhalten und ein Wellenbruch vermieden werden.

Der flexible Aufbau bietet die Möglichkeit, eine, zwei oder drei Gummireihen zu verwenden, um durch ein in Serie Schalten der Steifigkeiten eine Absenkung der Gesamtsteifigkeit zu erzielen. Eine Feineinstellung wird durch die Wahl der Shorehärten des Elastomers erreicht. Die Shorehärte wird durch den Eindringwiderstand eines genormten Prüfkörpers bestimmt und gibt Auskunft über die Elastizität und Dämpfung des Werkstoffes.

4.2 Messungen der t1000-800-2

Die Messung der t1000-800-2 wurde bei der Fa. Dipl.-Ing. Herwarth Reich GmbH in Bochum durchgeführt. Der Aufbau bestand aus einem starren und einem momentaufbringenden Flansch. Gemes-

[1] Die Poissonzahl ist eine Materialkonstante und beschreibt die Querdehnung eines Werkstoffes in Abhängigkeit zur Längsdehnung, und wird durch $\nu = \frac{E}{2G} - 1$ definiert. E entspricht dem Elastizitätsmodul und G dem Schubmodul.

sen wurden der Winkel und das Moment mit einer Abtastrate von 5kHz. Die Last wurde hydraulisch aufgebracht. In Abbildung 4.2 ist der Messaufbau abgebildet.

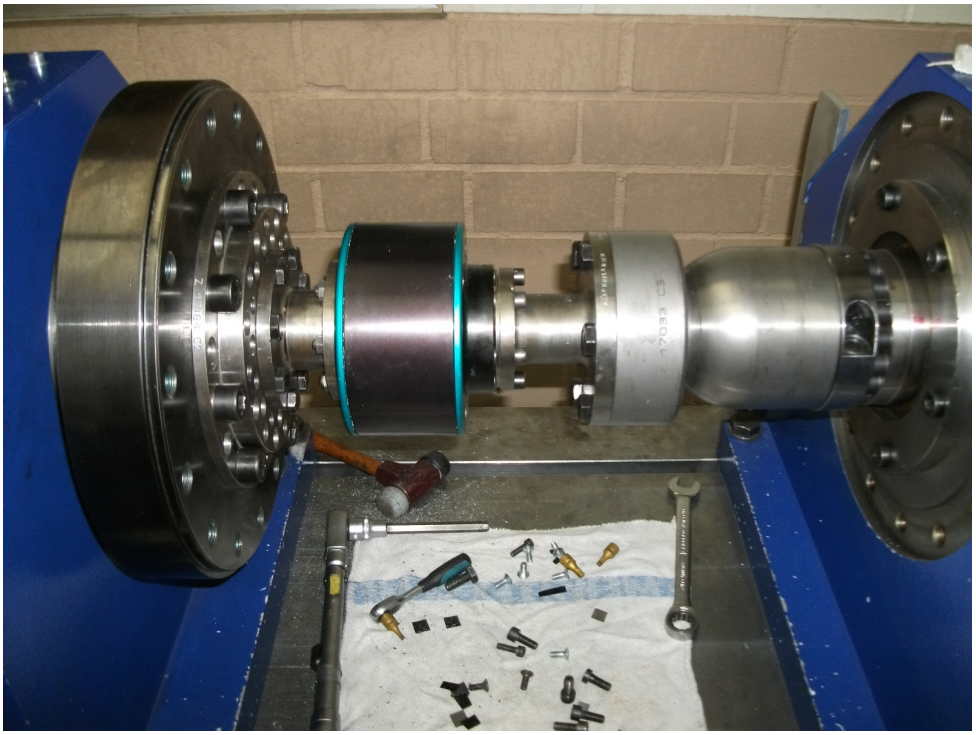


Abbildung 4.2: Messaufbau der t1000-800-2, Fa.Reich

Es wurden statische und dynamische Tests durchgeführt. Die dynamischen Test umfassten wechselnde Belastung bei 10Hz mit Amplituden von 100 - 600 Nm.

Für die Betrachtung der Simulation im Zeitbereich wurden die Messdaten geprüft und ausgewertet. Aufgrund sehr hohen Rauschens mussten die Messkurven mit niedrigen Amplituden bis 200Nm entfernt werden.

Grundsätzlich wurde bei der Parameterbestimmung großer Wert auf eine generische Umsetzung gelegt. Die Erarbeitung des Formalismus hilft auch in zukünftigen Messungen, um schnell und präzise die gewünschten Ersatzmodelle angeben zu können. Mithilfe von Mehrfachmessungen der Betriebspunkte kann ein überbestimmtes Gleichungssystem analytischer Ansätze erreicht werden. Diese Umstände bieten die Methode des Least Squares Optimierung zum Lösen von überbestimmten Systemen an, siehe Kapitel 4.4. Damit konnten für die angesetzten Dämpfungen die Parameter errechnet werden.

Um ein günstiges Verhalten in der Simulation zu gewährleisten, wurde nach einer Beschreibung der Dämpfung in Form einer stetigen Funktion gesucht. Dafür wurden Regressionsfunktionen gesucht, die über Funktionswerte des anliegenden Momentes berechnet werden können.

4.3 Torsionssteifigkeit

Bei der t1000-800-2 Kupplung musste eine Vereinfachung hinsichtlich der Steifigkeit getroffen werden, da die Kupplung aus zwei Reihen besteht und die mittlere Ebene nicht vermessen werden kann. Dabei wurde angenommen, dass beide Elastomerebenen sich gleich verhalten. Das Messergebnis ist somit die Summe der beiden Elastomerebenen.

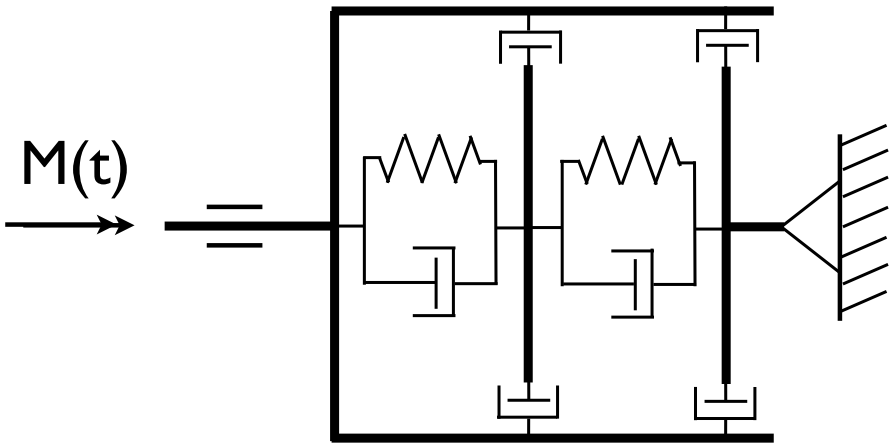


Abbildung 4.3: Ersatzschaubild t1000-800-2

Im Ersatzschaubild (Abbildung 4.3) ist das angenommene System aus Federn und Dämpfern ersichtlich. Durch die Reihenschaltung der nichtlinearen Steifigkeiten der Elastomere und den verschiedenen relativen Winkel bzw. Winkelgeschwindigkeiten am Deckel der Kupplung treten Effekte auf, die die Vereinfachung, welche vorher besprochen wurde, nicht abbilden kann. Im Betrieb von t1000 Kupplungen wurde beobachtet, dass die Reihe mit dem größeren Relativwinkel zum Deckel zuerst verschleißt. Dieses Verhalten wurde auch durch die FEM Analyse bestätigt, siehe [15]. Aufgrund fehlender Messungen muss aber auf diese Vereinfachung, der Annahme von gleichem Verhalten beider Elastomerebenen, zurückgegriffen werden.

Für die weiteren Untersuchungen wird die Bewegungsbeschreibung in der Form

$$\Theta\ddot{\varphi} + M_D(\varphi, \dot{\varphi}) + M_C(\varphi) = M(t) \tag{4.1}$$

verwendet. Darin beschreibt $M_D(\varphi, \dot{\varphi})$ die auftretenden Dämpfungsmomente und $M_C(\varphi)$ die auftretenden Steifigkeitsmomente. Bei der Analyse der Messdaten wurden Ähnlichkeiten zu einem Polynom dritter Ordnung, der Form 4.2, erkannt und mit Hilfe von MATLAB angenähert. Es wurde der Mittelwert der beiden Hysterese-Äste ermittelt. Hierzu wurde eine MATLAB-Routine erstellt, die für jeden Winkel der Hysterese die benachbarten, innerhalb eines bestimmten Winkels liegenden, Drehmomentwerte arithmetisch mittelt. Die so erhaltene Datenreihe wurde mit einer Polynomregression dritter Ordnung versehen und die Polynomkoeffizienten gespeichert, siehe Code Listing

A.7. Folgend sind $p_{1,2,3,4}$ als Polynomkoeffizienten und φ als Winkel definiert.

$$M_C(\varphi) = p_1\varphi^3 + p_2\varphi^2 + p_3\varphi + p_4 \tag{4.2}$$

Durch den Aufbau der Kupplung muss die Steifigkeit nullpunktssymmetrisch sein. Um Einflüsse durch Messfehler zu minimieren, werden nur die Polynomfaktoren dritter und erster Ordnung verwendet, da die Faktoren der zweiten Ordnung einen Verlust der Punktsymmetrie um den Ursprung und p_4 eine Verschiebung zur Folge hätten. Somit verkürzt sich der Ansatz des Torsionssteifigkeitspolynoms zu Gleichung 4.3.

$$M_C(\varphi) = p_1\varphi^3 + p_3\varphi \tag{4.3}$$

Durch die Auswertung der verschiedenen Materialhärten konnten folgende Steifigkeiten der einzelnen Elastomere bestimmt werden (Abbildung 4.4).

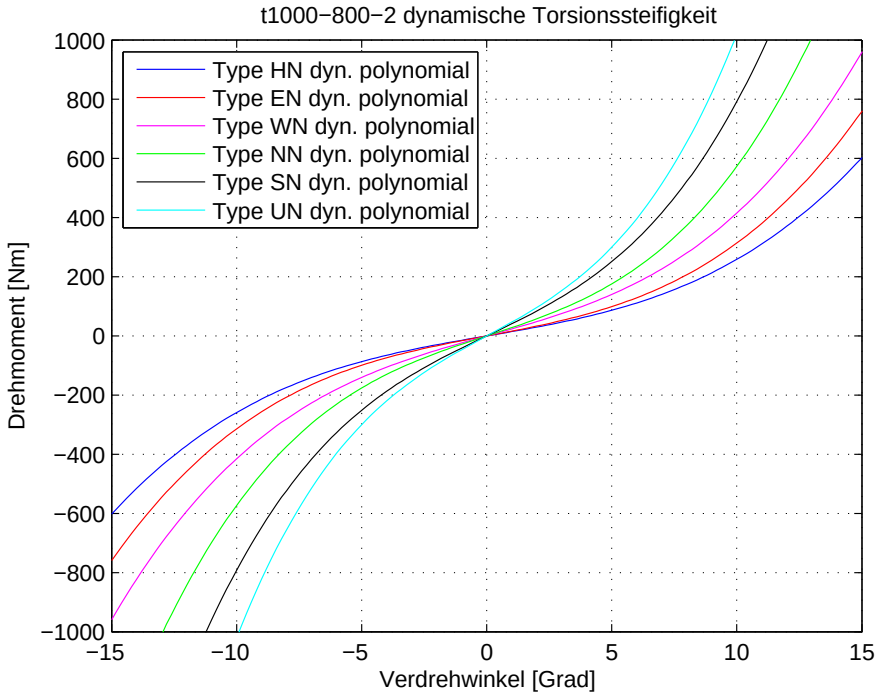


Abbildung 4.4: Steifigkeitsverlauf der t1000-800-2 mit unterschiedlichen Elastomeren

Diese Steifigkeiten deckten sich auch mit FEM Analysen mit hyperelastischer Materialmodellierung. Zu diesem Thema wurde bereits eine Bakkalaureatsarbeit von Herrn Wolfgang Mayr verfasst, die sich mit der Modellierung dieser Kupplung und des Materialgesetzes beschäftigt hat, siehe [15].

4.4 Dämpfungshypothesen

Grundsätzlich gibt es zwei Herangehensweisen, um die Dämpfung in einem Modell zu berücksichtigen.

- Phänomenologischer Ansatz, der die gemessen Daten ohne weiteres Hinterfragen ihrer Herkunft in die Simulation mittels Kennlinien oder Tabellen hinterlegt
- Analytischer Ansatz, der die physikalischen Vorgänge berücksichtigt und durch eine mathematische Formulierung beschreibt

Als Beispiel dient die Hysterese der Materialqualität SN in Abbildung 4.5. Diese soll mit den zuvor genannten Methoden beschrieben werden. Der progressive Verlauf der Steifigkeit ist deutlich zu erkennen. Auch eine leichte Zunahme der Dämpfung zu höheren Winkeln kann festgestellt werden.

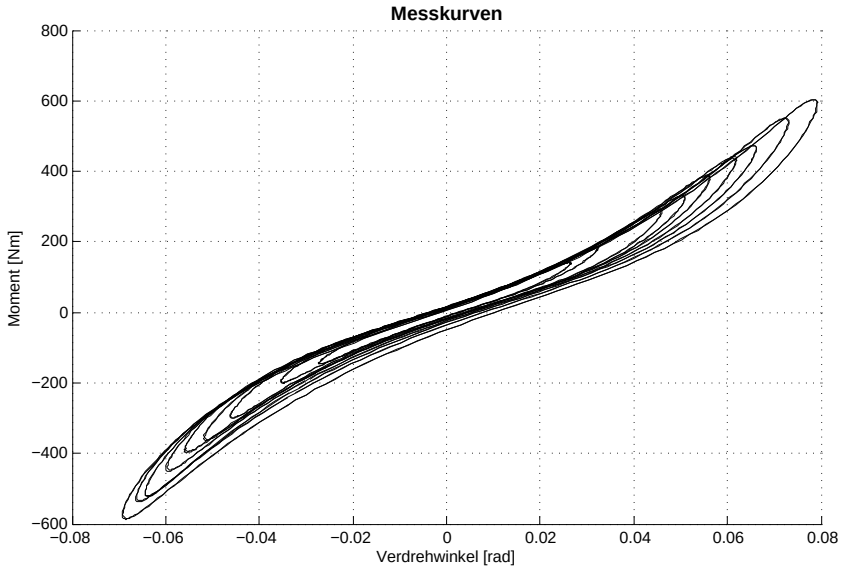


Abbildung 4.5: Hysterese der t1000-800-2 SN über mehrere Wechselmomente

4.4.1 Phänomenologischer Ansatz

Der phänomenologische Ansatz umfasst eine Modellierung der Feder- und Dämpfercharakteristik mittels einer Tabelle, die für den jeweiligen Winkel interpoliert wird. Modelica bietet in seiner Standard-Bibliothek die Möglichkeit der Interpolation in Datenreihen. Der Aufbau des Simulationsmodells ist in Abbildung 4.6 abgebildet, wobei in der t1000 Komponente eine Interpolation der Moment-Winkel Beziehung hinterlegt ist.

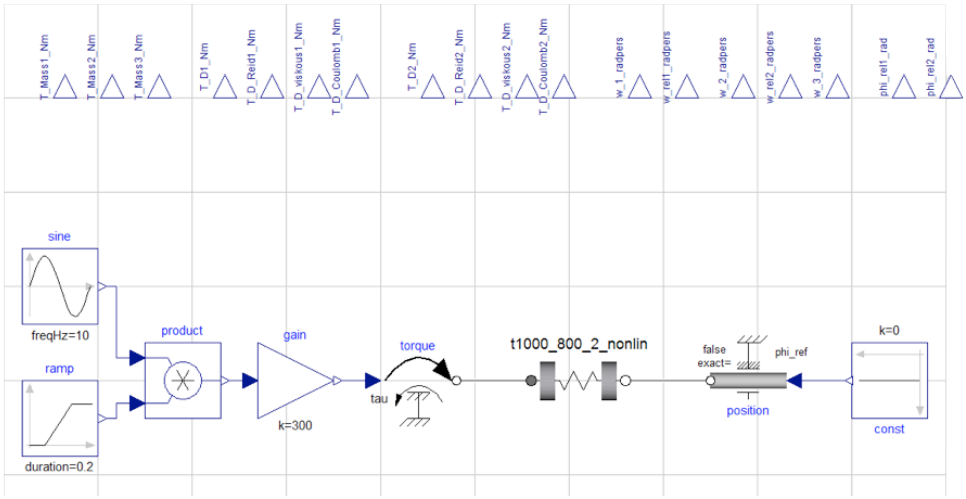


Abbildung 4.6: Aufbau t1000 Prüfstand Interpolation

Das Erregermoment wird als Sinus-Funktion auf der einen Seite des Prüflings aufgetragen. Auf der anderen Seite wird die Kupplung durch die Vorgabe von $\varphi = 0$ fixiert. Die Modellierung der Erregung mittels einer Rampe zu Beginn hilft zum schnelleren Erreichen des eingeschwungenen Zustands. Der Nachteil der phänomenlogischen Modellierungsmethode ist das instabile Verhalten in der Simulation bei Richtungsumkehr der Winkelgeschwindigkeit. Bei dieser Modellierung muss je nach Vorzeichen der Winkelgeschwindigkeit zwischen dem aufsteigendem und abfallenden Ast der Hysterese umgeschaltet werden. Diese Umschaltung kann hohe Drehmomentänderungen in kurzer Zeit bewirken und somit zu starken Schwingungen anregen. Bereits bei stationären Simulationsbedingungen kann diese Anregung zu erheblichen Schwierigkeiten führen, siehe Abbildung 4.7.

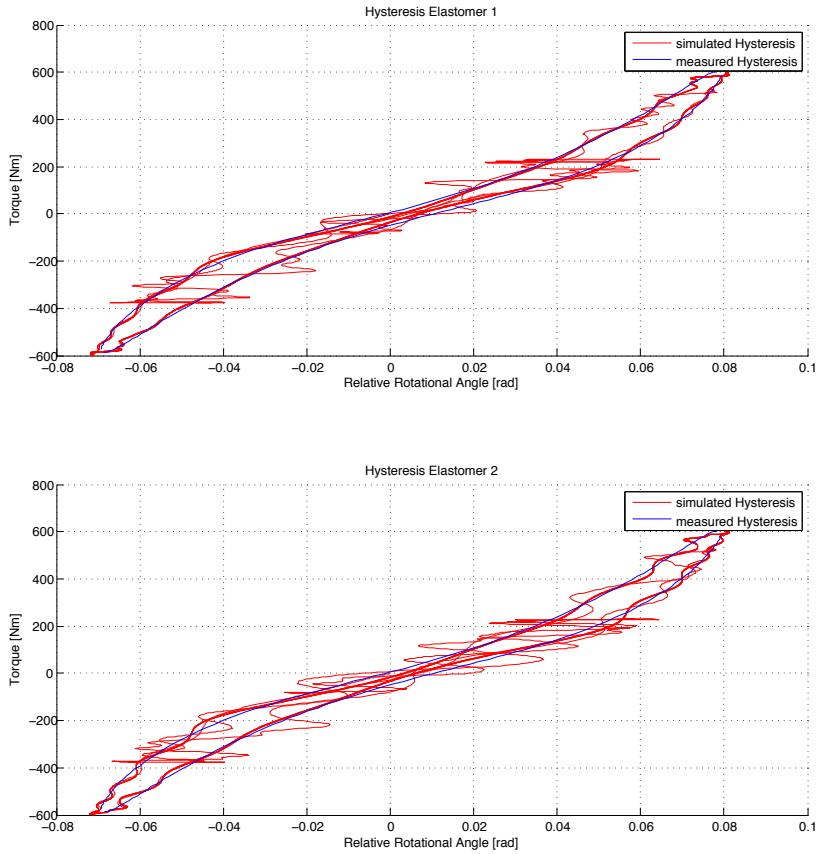


Abbildung 4.7: Hysterese der t1000-800-2 SN, Modellierung mit Lookup

Diese Probleme führten zu den Überlegungen, einen stetigen Ansatz für die Feder- und Dämpfercharakteristik zu erarbeiten.

4.4.2 Analytische Ansatz

Beim analytischen Ansatz wird basierend auf physikalischen Überlegungen eine passende Modellierung erarbeitet. Hierzu werden nicht Simulationen sondern die Messergebnisse, des in Abschnitt 4.2 beschriebenen harmonisch erregten Prüflaufes, verwendet, um die Dämpfungsparameter zu bestimmen. Um die Dämpfung eigenständig betrachten zu können, wurden nach Ansetzen der Systemgleichung die Dämpfungsmomente durch Umformen ausgedrückt.

$$\theta\ddot{\varphi} + M_D(\dot{\varphi}, \varphi) + M_C(\varphi) = \hat{M}sin(\Omega t) \tag{4.4}$$

$$M_D(\dot{\varphi}, \varphi) = \hat{M}sin(\Omega t) - M_C(\varphi) - \theta\ddot{\varphi} \tag{4.5}$$

Die reinen Dämpfungsdrehmomente können nun durch Subtrahieren der Steifigkeitsdrehmomente und der Trägheitsdrehmomente von den gemessenen Drehmomenten berechnet und der Parame-

terbestimmung zugeführt werden.

Da bei den Messdaten nur Informationen des Winkel, aber nicht die Winkelgeschwindigkeit, bzw. die Winkelbeschleunigung ermittelt wurden, mussten die Ableitungen des Winkels numerisch gebildet werden. Die Durchführung der numerischen Ableitung konnte nicht direkt erfolgen, da kein Zeitschrieb vorhanden war. Durch Betrachten einer Periode der Schwingung (10Hz Erregerfrequenz) und der Tatsache, dass mit 5kHz Abtastrate aufgezeichnet wurde, konnte im Nachhinein der Zeitvektor rekonstruiert werden.

Der Verlauf der Dämpfungsmomente über den Winkel wird in der Abbildung 4.8 dargestellt.

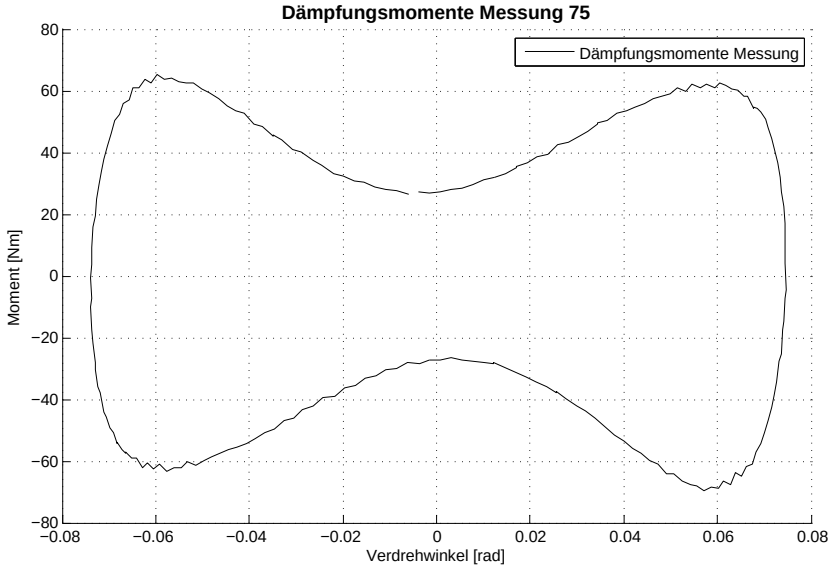


Abbildung 4.8: Dämpfungsmomente der t1000-800-2 SN bei 600Nm, 10Hz

Die Form der Dämpfung über den Winkel bestätigt die vorangegangenen Überlegungen einer Winkelabhängigkeit der Dämpfung. Das vorliegende Verhalten kann aus den Dämpfungsansätzen nach Coulomb, Reid und Kelvin-Voigt angenähert werden. Im Detail wird ein prozentualer Anteil der Reibung zur Dämpfung um $\varphi = 0$ angenommen. Die Parameter für Kelvin-Voigt und Reid werden per Least Squares Methode bestimmt. Diese Vorgehensweise bietet sich an, da mehrere Zyklen von Messdaten vorhanden sind und der mathematische Ansatz als überbestimmtes Gleichungssystem mit dem niedrigsten quadratischen Fehler gelöst werden kann. Eine detaillierte Beschreibung dieses Verfahrens wird in [16] beschrieben.

Der gewählte Ansatz lässt sich mathematisch als Addition der drei Ansätze darstellen.

$$M_D = M_{Kelvin-Voigt} + M_{Reid} + M_{Coulomb} \tag{4.6}$$

$$M_D = d_{Kelvin-Voigt}\dot{\varphi} + d_{Reid}|\varphi|sign(\dot{\varphi}) + d_{Coulomb}sign(\dot{\varphi}) \tag{4.7}$$

Zur Parameterbestimmung wurde eine Auswerteroutine erstellt, die zuerst die benötigten Parameter

aus den Messdaten ermittelt und für die Least Squares Methode vorbereitet. Zuerst muss das Dämpfungsmoment nach Gleichung 4.5 gebildet werden. Die Daten, wie in Abbildung 4.8 dargestellt, stellen die Grundlage für die Verarbeitung dar. Zuerst wird der coulomb'sche, winkelunabhängige Teil der Dämpfung abgezogen. Das Ergebnis der Subtraktion dient als Vorgabe für die Bestimmung der Reid Parameter. Nach weiterer Subtraktion der berechneten Reid Dämpfungsmomente wird der Rest mittels Kelvin-Voigt Dämpfung beschrieben. Die Staffelung der Parameterbestimmung wurde bewusst in mehrere Schritte unterteilt, da bei Kombination aller Ansätze das Risiko besteht, dass die Lösung mit Hilfe der Methode der kleinsten Fehler-Quadrate unter Umständen eine negative Dämpfung ergeben kann, was jedoch physikalisch nicht möglich wäre.

Die Coulomb'sche Reibung äußert sich als konstante Reibung unabhängig von der Auslenkung. Um die dissipativen Effekte zu beschreiben, wird sie mit dem Vorzeichen von $\dot{\varphi}$ multipliziert.

$$M_{Coulomb} = d_{Coulomb} sign(\dot{\varphi}) \tag{4.8}$$

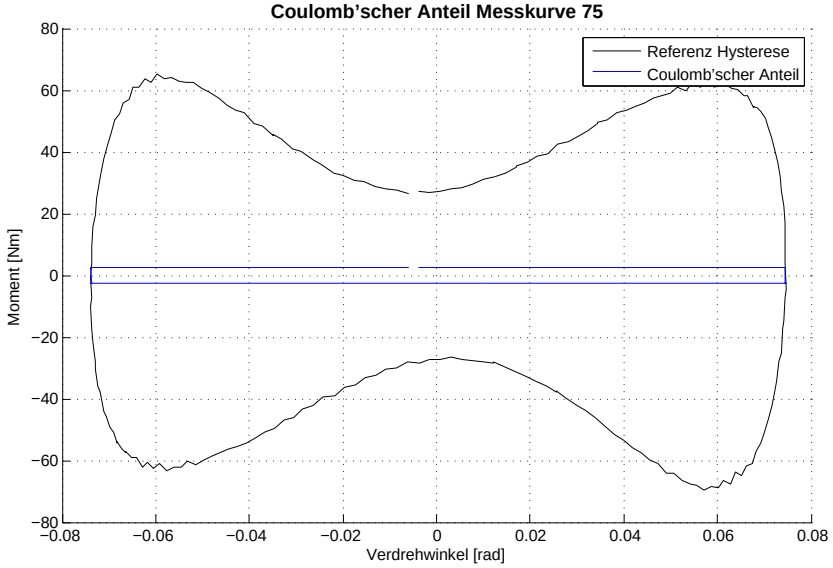


Abbildung 4.9: Dämpfungsmomente nach Coulomb der t1000-800-2 SN bei 600Nm, 10Hz

Wie zuvor beschrieben ergibt der Abzug der errechneten Coulomb'schen Dämpfung die Vorgabe für den nächsten Analyseschritt.

Die Reid Dämpfung zeichnet sich durch ihre Abhängigkeit vom Winkel aus. Mit zunehmender Auslenkung steigt auch die betragsmäßige Dämpfung.

$$M_{Reid} = d_{Reid} |\varphi| sign(\dot{\varphi}) \tag{4.9}$$

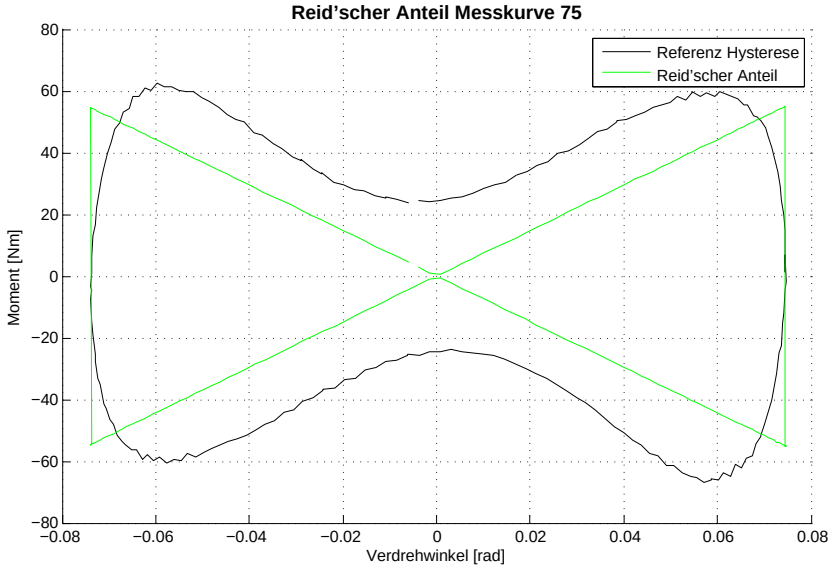


Abbildung 4.10: Dämpfungsmomente nach Reid der t1000-800-2 SN bei 600Nm, 10Hz

Die Vorgabe mit abgezogener Coulomb'scher Dämpfung und die errechnete Reid-Dämpfung sind in Abbildung 4.10 dargestellt.

Die Kelvin-Voigt Dämpfung beschreibt, wie in Kapitel 2.3 behandelt, die klassische Form der viskosen Dämpfung, die sich als Ellipse ausprägt und durch

$$M_{Kelvin-Voigt} = d_{Kelvin-Voigt}\dot{\varphi} \tag{4.10}$$

ausgedrückt wird. Die Vorgabe erfolgt durch Abzug der zuvor berechneten Dämpfungsmomente. Der Rest der vorhandenen Dämpfung und das Ergebnis der berechneten Kelvin-Voigt'schen Dämpfung ist in Abbildung 4.11 dargestellt.

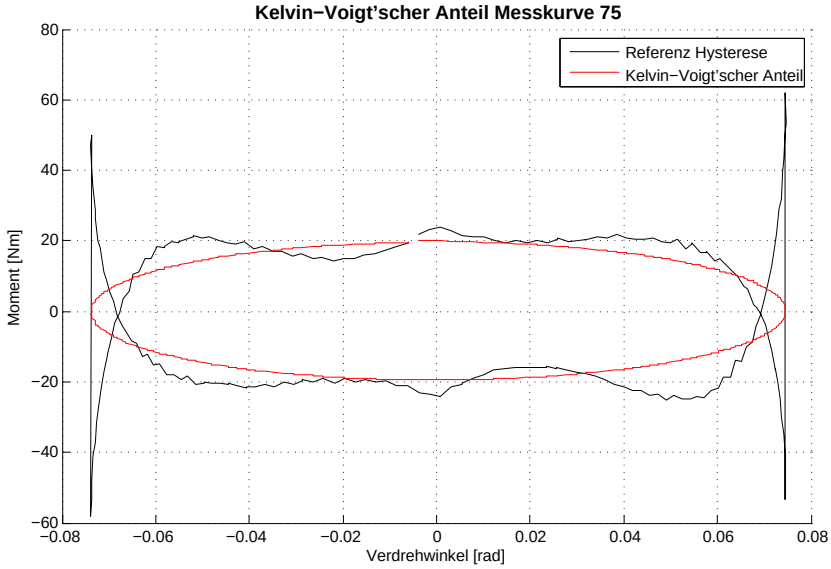


Abbildung 4.11: Dämpfungsmomente nach Kelvin-Voigt der t1000-800-2 SN bei 600Nm, 10Hz

Durch Aufsummieren der errechneten Dämpfungen kann die analytisch beschriebene Dämpfung mit der gemessenen verglichen werden, siehe Abbildung 4.12.

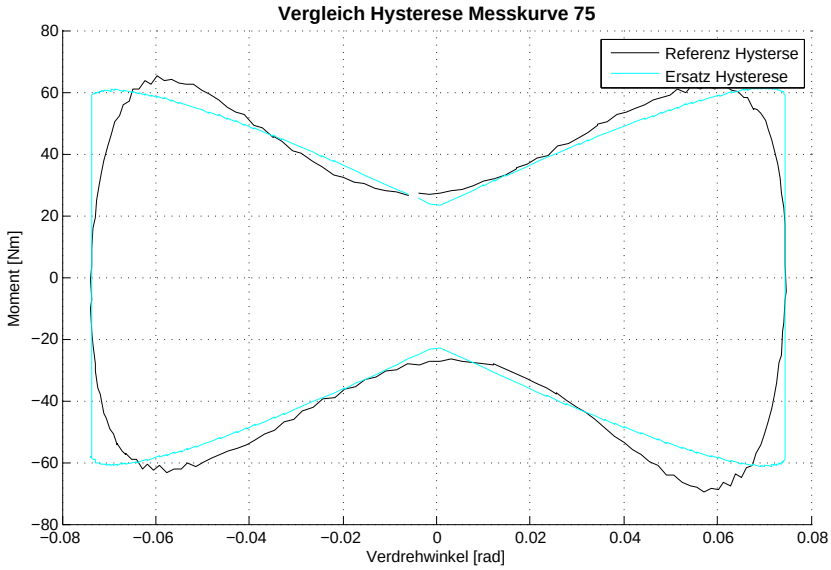


Abbildung 4.12: Vergleich der gemessenen zur simulierten Hysterese t1000-800-2 SN bei 600Nm, 10Hz

Bei dieser Modellierung ist mit einem Fehler zu rechnen, da das System bei geringen Winkeln über- und bei größeren Winkeln untergedämpft ist. Jedoch liefert der analytische Ansatz eine sehr gute Übereinstimmung zur gemessenen Kurve.

Dieses Verfahren wurde über alle Messpunkte angewendet und führt zu folgenden Ergebnissen aufgetragen über das Wechselmoment.

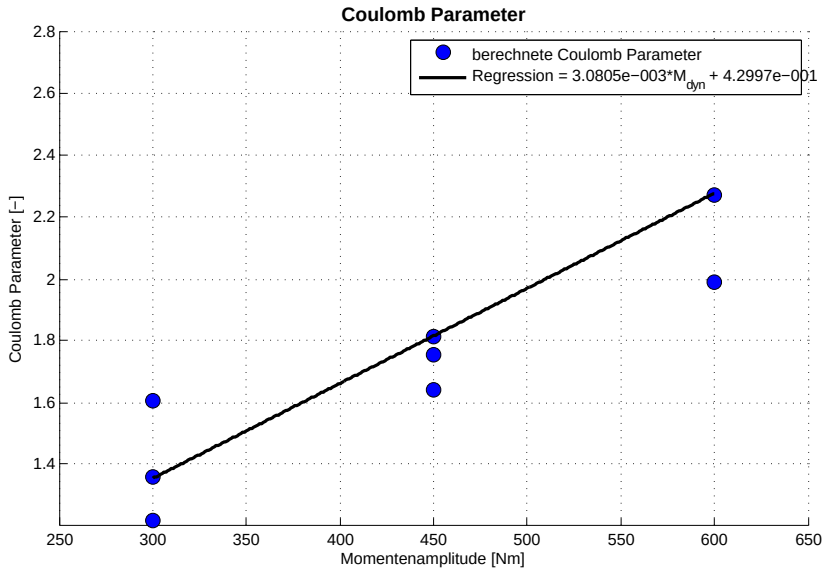


Abbildung 4.13: Coulomb Parameter t1000-800-2 SN bei 600Nm, 10Hz

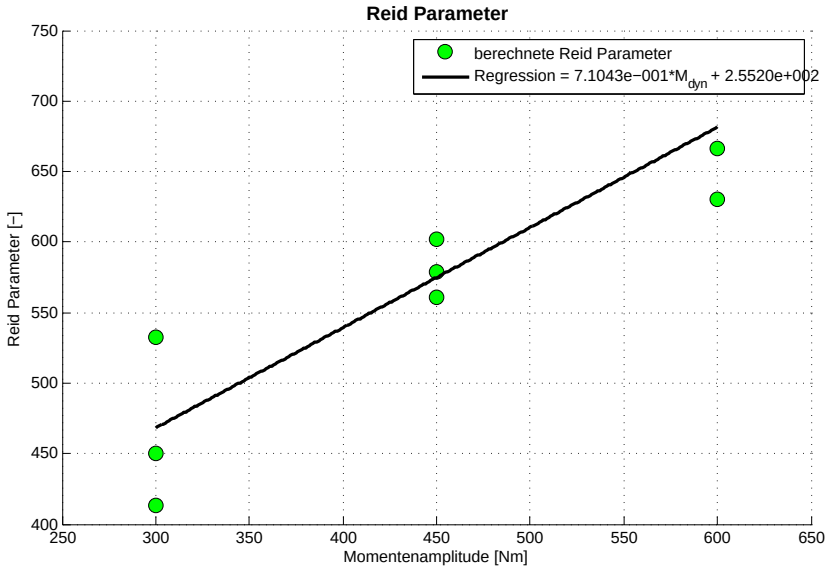


Abbildung 4.14: Reid Parameter t1000-800-2 SN bei 600Nm, 10Hz

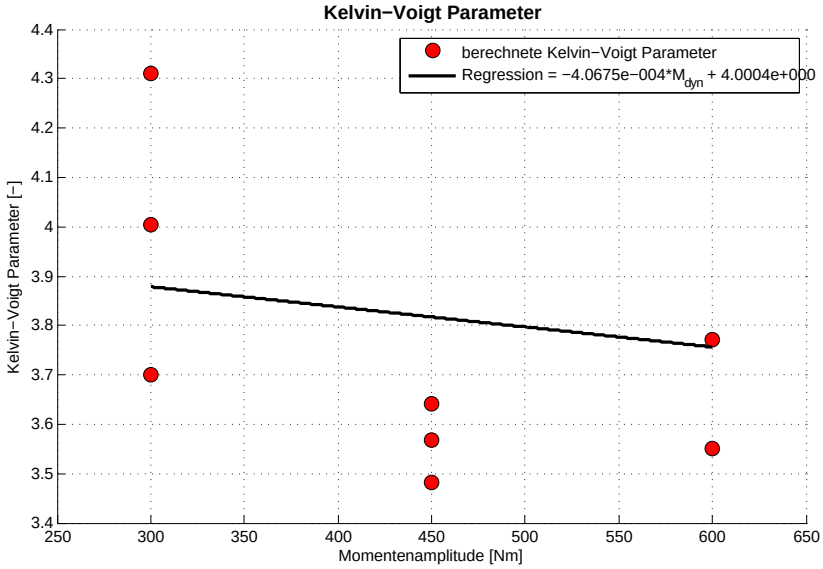


Abbildung 4.15: Kelvin-Voigt Parameter t1000-800-2 SN bei 600Nm, 10Hz

Um eine Korrelation zum wirkenden Wechselmoment zu erhalten, wurde eine lineare Regression erstellt. Im Falle der äußeren Dämpfung (Ansatz nach Coulomb und Reid) konnte ein Zunahme der Faktoren über die wirkende Last beobachtet werden. Bei der inneren Dämpfung (Ansatz nach Kelvin-Voigt), die eher lastunabhängig eingestuft wird, zeigt sich auch ein eher konstantes Verhältnis zum Wechselmoment. Die Wahl der Dämpfungsansätze und die Korrelationsfunktionen bieten noch Potential für weitere Verbesserungen der Beschreibung der Dämpfung.

Ein zusätzlicher Ansatz zur Erweiterung der Dämpfungsmodellierung wäre die Berücksichtigung der Erregerfrequenz. Besonders in der dynamischen und instationären Betrachtung ist dieses Verhalten wichtig. Da keine Daten für diese Belastungsfälle gemessen werden konnten, musste auf die Untersuchung des Frequenzeinflusses verzichtet werden.

Die Ermittlung der Parameter wurde für alle Elastomerhärten (EN, WN, SN, UN) ausgeführt und ausgewertet. Die Ergebnisse der anderen Materialien gleichen sich in der Modellierung zur Gänze und werden hier nicht weiter dokumentiert.

Die Validierung der Ergebnisse mittels eines Modelica Modells des Vermessungsprüfstandes ist in Kapitel 6.1 beschrieben.

Die Untersuchungen haben gezeigt, dass diese Art der Kombination aus Coulomb'scher, Reid'scher und Kelvin-Voigt'scher Dämpfung eine gute Übereinstimmung mit den Messdaten erzielen. Jedoch wird die Dämpfung bei niedrigen Winkeln über- und bei großen Winkeln unterbewertet.

Als Ausblick für weitere Untersuchungen würde sich ein Ansatz höherer Ordnung für den Reid Parameter anbieten, der den Verlauf zum Maximalwinkel besser beschreibt und eine weitere Verbesserung des Verhaltens liefern sollte. Auch die Betrachtung bei weiteren Erregerfrequenzen wäre bei der Beschreibung im dynamischen Betriebsbereich von großem Vorteil.

Kapitel 5

Workflow OpenModelica mit MATLAB

Ein grundlegender Teil dieser Arbeit war das Erstellen einer MATLAB Umgebung, die die Simulation sowie das Preprocessing und Postprocessing beherrscht und die passenden Schnittstellen zu den bereits bestehenden Systemen der tectos gmbh aufweist. Es gibt schon bestehende Ansätze und Formalismen, die das Handling des Compilers mit Hilfe von MATLAB Funktionen steuern, jedoch waren diese für die folgenden Ansprüche zu erweitern.

Im Laufe der Modellierung wurde klar, dass das Motorkennfeld der Zylinderdrücke, welches von Drosselklappenstellung, Drehzahl und Kurbelwinkel abhängt, nur durch eine dreidimensionale Interpolation realisierbar ist. Diese ist in der Standard-Bibliothek von Modelica nicht verfügbar und musste daher im Laufe dieser Arbeit erstellt werden.

5.1 Preprocessing

Das Preprocessing besteht im Wesentlichen in der Datenbereitstellung aus bereits bestehenden Datensätzen. Die wesentlichsten Inhalte sind das Auslesen der vorbereiteten Anregungsdaten aus einer Motorprozessrechnung oder aus Zylinderindizierkurven sowie die Vorbereitung der externen C-Funktion, die in der Motormodellierung von Nöten ist und somit neu erstellt werden musste.

5.1.1 Preprocessing der Motordaten

Die Zylinderdrücke liegen in Form von Kompressions- und Verbrennungsdrücken vor, welche sich über den gesamten Betriebsbereich eines Motors erstrecken. Die Aufgabe der Matlab Funktion (`createCLut.m`) besteht darin, eine kompilerbare Datei zu erzeugen, die in den Modelica Code eingebettet und aufgerufen werden kann und alle benötigten Informationen aus den Quellen einliest.

Der Workflow dieser Funktion ist in Abbildung 5.1 dargestellt.

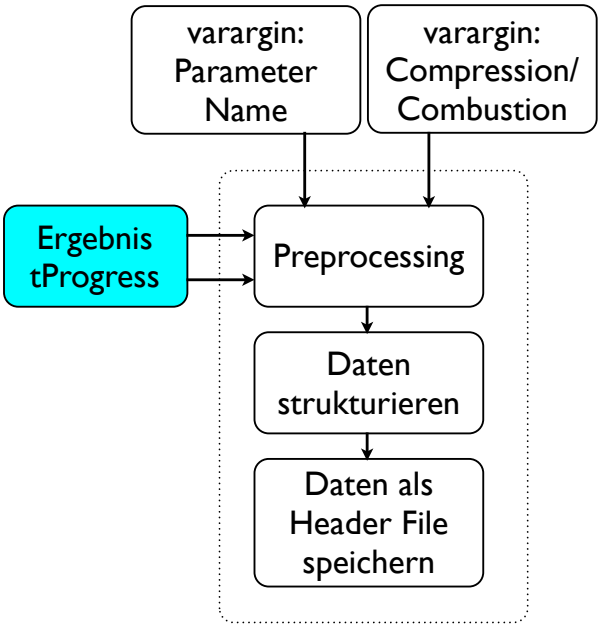


Abbildung 5.1: Workflow createCLut.m

Abhängig von den Eingangsdaten der Funktion werden Kompressionsdaten oder Verbrennungsdaten geladen. Die Daten werden auf eine in C verwertbare Matrix übersetzt und als C Header File abgespeichert.

Der Interpolation liegt das System der binären Suche zu Grunde. Dieser Algorithmus findet in sortierten Datenreihen sehr schnell die Position des gewünschten Wertes. Damit ist der Index des linken und rechten Nachbarwertes bestimmt. Der Algorithmus geht wie folgt vor. Zuerst wird der mittlere Wert evaluiert. Der Wert wird mit dem gewünschten verglichen und bewertet, ob dieser größer oder kleiner ist. Somit kann der Suchbereich um die Hälfte eingeschränkt werden. Diese Vorgehensweise wird wiederholt, bis die Suche auf ein einzelnes Element bzw. die nähesten zwei Nachbarelemente stößt.

Die Interpolation der Zylinderdruckdaten wird in der Funktion `interpolation3D.c` ausgeführt und wird schematisch in Abbildung 5.2 dargestellt.

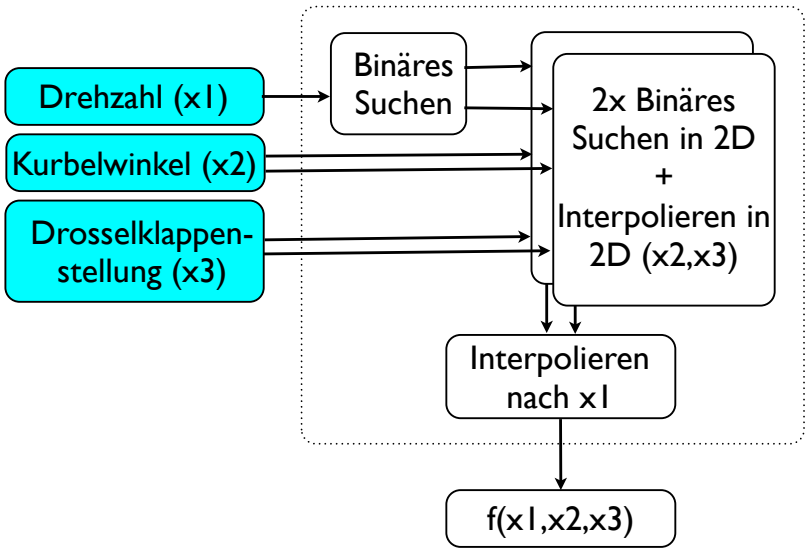


Abbildung 5.2: Workflow interpolation3D.c

Als Eingangsgrößen gelten $x1, x2, x3$. In Abhängigkeit dieser Größen soll der Funktionswert aus der Datenmatrix interpoliert werden. Zuerst wird binäres Suchen für $x1$ angewendet. Mit dessen Index in der Matrix können zwei zweidimensionale Datenreihen bestimmt werden. Durch weiteres binäres Suchen in beiden zweidimensionalen Datensätzen nach $x2$ und $x3$ kann der Funktionswert in beiden zweidimensionalen Wertereihen bestimmt werden. Abschließend wird eine eindimensionale Interpolation nach $x1$ zwischen den beiden zuvor errechneten Werten durchgeführt, um somit den Funktionswert $f(x1, x2, x3)$ bestimmen zu können.

5.2 Berechnung

Die Steuerung des OpenModelica Compilers in MATLAB wurde bereits an der TU Darmstadt durch Christian Schaad erarbeitet und unter dem Namen OpenModelica MATLAB Interface veröffentlicht [17]. Da aber bei den ersten Berechnungen Limiterungen seitens OpenModelica deutlich wurden, musste dieser Formalismus überarbeitet werden. Es musste die Größe der Ergebnisdatei so klein wie möglich gehalten werden, da bei Überschreiten eines Grenzwertes die Ergebnisdatei korrupt wird. Das wurde mittels einer im Compiler integrierten Funktion des `variableFilter` umgesetzt. Somit werden nur die zuvor definierten Variablen in der gewünschten Abtastrate in der Ergebnisdatei abgespeichert.

Diese Liste der Variablen ist durch die Ausgänge im Modell definiert und muss extern in MATLAB gesetzt werden. Diese Funktionalität wird in der MATLAB Datei `assignOutput.m` (Code Listing A.5) zur Verfügung gestellt und bei der Erstellung des Modelica Skripts als Erweiterung des Compilers eingefügt.

Eine weitere Funktionserweiterung des OpenModelica Compiler ist das Abbruchkriterium des Residuums für die Berechnung eines Zeitschrittes, welches standardmäßig auf 10^{-10} gesetzt ist. Für mechanische Schwingungen liefert eine Toleranz von 10^{-4} für das Residuum immer noch hinreichend genaue Ergebnisse und wird somit beim Vorbereiten des Compilers mit 10^{-4} gesetzt.

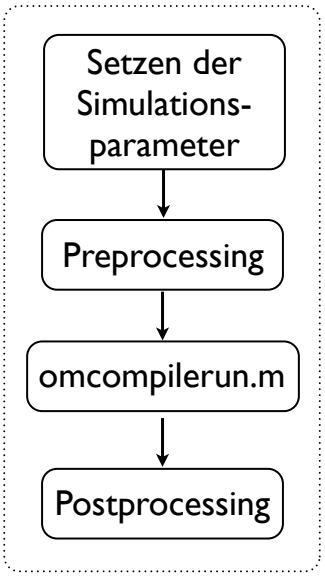


Abbildung 5.3: Workflow r.m - Ablauf der Simulation

Der Ablauf einer Berechnung wird in der Matlabroutine `r.m` festgelegt und startet mit dem Setzen der Simulationsparameter. Zu diesen gehören die Simulations Start- und Stopzeit, der Name des zu simulierenden Modells, die Bibiotheken, die Abtastrate sowie das Solver Residuenkriterium. Danach werden mit Hilfe der Funktion `assignOutput.m` die Variablen, welche in der Ergebnisdatei gespeichert werden sollen, als Parameter hinterlegt.

Als Nächstes erfolgt das Preprocessing der Motordaten und deren Funktion der 3D Interpolation durch die Funktion `createCLut.m` (Code Listing A.1). In dieser Funktion werden die Daten für die 3D Interpolation aus der Quelle ausgelesen und als C-Header aufbereitet und die eigentliche C-Funktion erzeugt und kompiliert, siehe Kapitel 5.1.

Danach erfolgt in der Funktion `omcompilerun.m` (siehe Abbildung 5.4) die eigentliche Berechnung. Zuerst wird mit den gesetzten Simulationsparametern ein .mos Skript erzeugt. Dieses Skript umfasst das Laden der gewünschten Bibliotheken, das Umsetzen des Codes in C und auch den Befehl zum Simulieren des gewählten Modells. Das Skript kann dem OpenModelica Compiler `(omc.exe)` zugeführt werden, der das Skript zeilenweise ausführt.

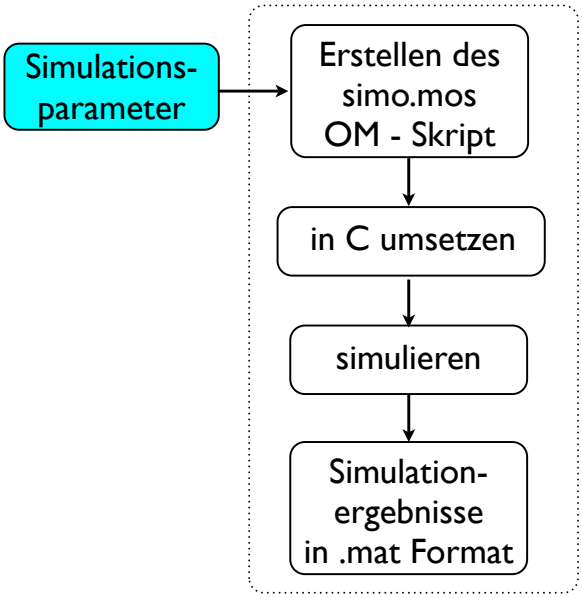


Abbildung 5.4: Workflow omcompilerun.m

Als Ergebnis erhält man eine .mat Datei mit den Ergebnissen. Die Benennung erfolgt als Kombination aus dem Modellnamen und dem Postfix `_res.mat`.

5.3 Postprocessing

Nach der erfolgreichen Simulation des Modells werden die Simulationsergebnisse mittels Postprocessing aufbereitet. Eine gute Aufbereitung hilft dabei, die wesentlichen Resultate der Berechnung zu erkennen bzw. mit Messdaten zu vergleichen.

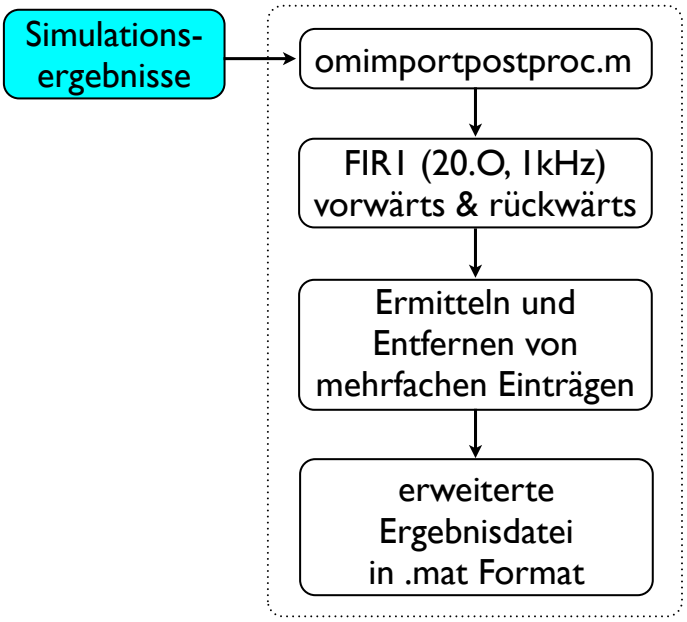


Abbildung 5.5: Workflow Postprocessing

Um die Ergebnisse direkt mit dem tectos eigenen Postprocessing tool `tPostProc` auswerten zu können, wurde eine Routine entwickelt, die die Ergebnisse im geforderten Format ausgibt. Die Funktion `omimportpostproc.m` (Code Listing A.6) konvertiert die aus Modelica erhaltenen Ergebnisse in die geforderte Struktur. Hierbei wird jedem Datensatz der entsprechende Name und die entsprechende Einheit zugeordnet. Da im realen System ein Grenzwert der auftretenden Frequenzen existiert, was auf eine Impedanz[1] zurückzuführen ist, wird das Simulationsergebnis, das deutlich höhere Frequenzen beinhalten kann, mittels `filtfilt` in MATLAB gefiltert. Dieses Vorgehen ermöglicht eine Filterung ohne eine Phasenverschiebung der Datensätze, da zuerst eine Filterung in aufsteigende Richtung erfolgt und danach das gefilterte Signal noch einmal in abfallende Richtung gefiltert wird, um den unvermeidbaren Phasenversatz zu kompensieren, vgl. [18]. Der gewählte Filter entspricht einem FIR Filter 20. Ordnung mit einer Cut-Off Frequenz von 1kHz. Weiters wird in den Signalen nach mehrfachem Vorkommen von Zeitwerten gesucht und diese mit dem entsprechenden Funktionswert entfernt, da die weitere Verarbeitung sehr restriktiv auf nicht stetige Datensätze reagiert. Die Existenz von doppelten Werten kann auf einen Programmfehler in OpenModelica zurückgeführt werden, da dieses Verhalten auch in den originalen Ergebnisdateien zu sehen ist.

[1] Die Impedanz beschreibt die Widerstände, die einer Ausbreitung von Schwingungen entgegenwirken, vgl [8]

Kapitel 6

Simulationsergebnisse und Validierung

6.1 Elastomerkupplung t1000-800-2

Zum Überprüfen der Ansätze für die Steifigkeits- und Dämpfungsmomente wurde ein Modelica Modell des Prüfstandes erstellt. Der Aufbau ist in Abbildung 6.1 dargestellt.

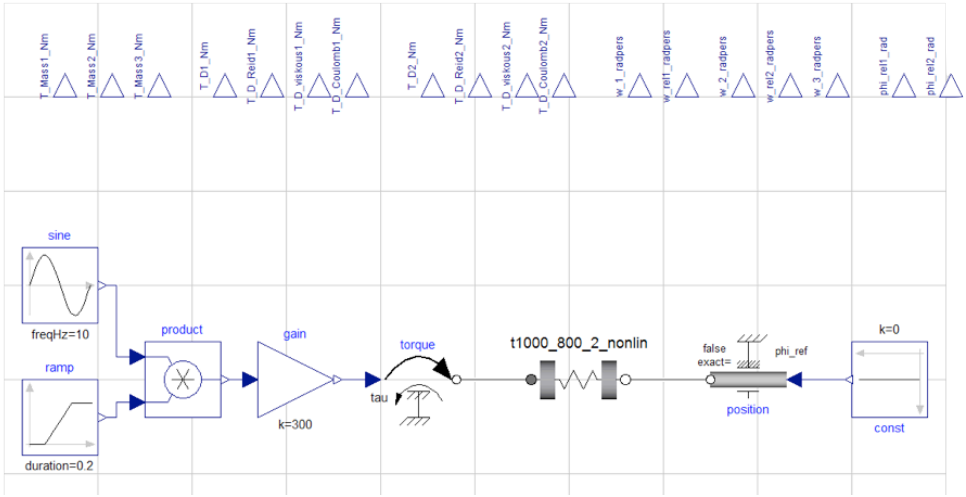


Abbildung 6.1: Hysterese bei 300Nm Wechselmoment, 10 Hz

Das Moment wird als Sinus-Funktion mit dem der Messung entsprechenden Wechselmoment aufgetragen. Zum Steigern der Stabilität der Simulation wurde das Erregersignal mit einer 0.2 s dauernden Rampe multiplizert, um den Anstieg des Sinus zu Beginn der Simulation und damit die Anregung des mechanischen Systems zu verkleinern. Somit ergibt sich ein eingeschwungener Zustand schneller, wodurch die Simulationszeit reduziert werden kann. Im eingeschwungenen Zustand werden nun die Simulationsergebnisse betrachtet und bewertet.

Zur Validierung werden die Messungen gewählt, die bereits zur Parameterbestimmung herangezogen wurden. Als Material wird die Qualität SN herangezogen. Wie zuvor in Abschnitt 4.4 beschrie-

ben, gleichen sich die mathematischen Modelle und die Daten der anderen Materialien. Sie werden daher nicht weiter dokumentiert.

Die Auswertung der Daten erfolgt über eine Simulationsdauer von 3s bei 10Hz. Es wurde geprüft, dass der eingeschwungene Zustand jeweils erreicht wurde.

Zur Veranschaulichung des Messungs-Rechungsvergleichs wurden 3 dynamische Messungen gewählt:

- 300Nm Wechselmoment mit 10Hz Erregerfrequenz

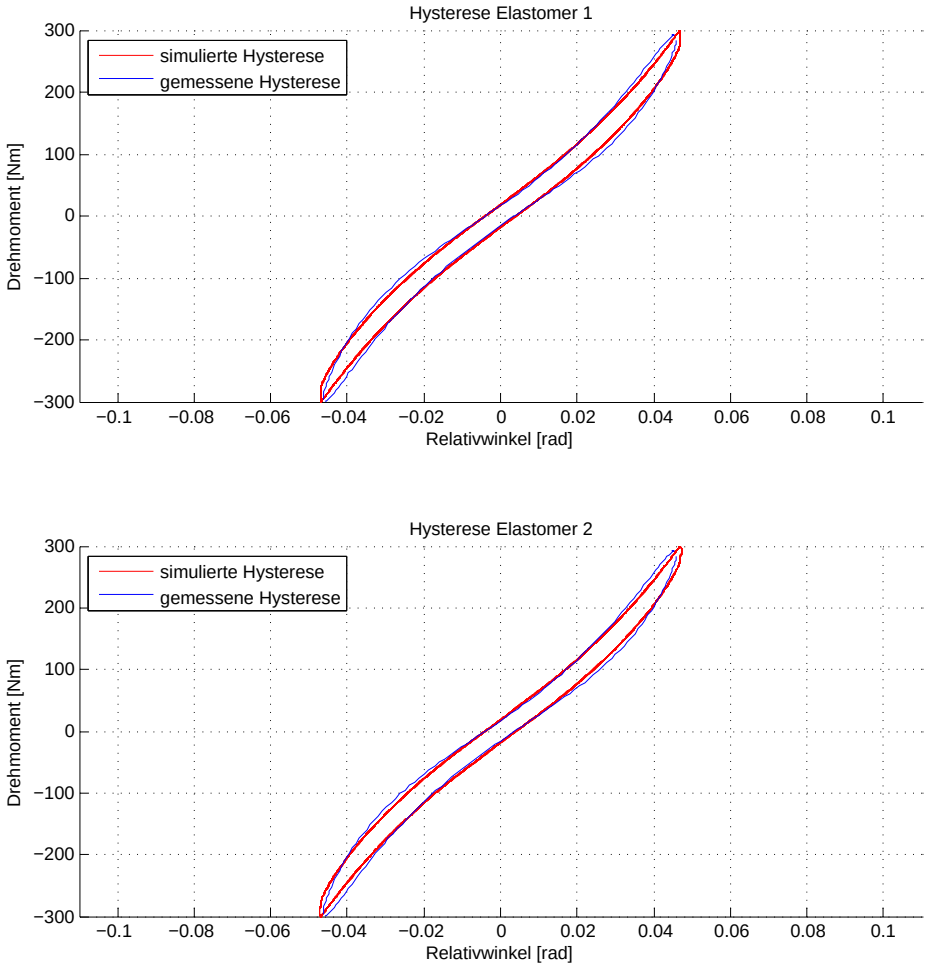


Abbildung 6.2: Hysterese bei 300 Nm Wechselmoment, 10 Hz

Die Hystereseschleife zeigt eine sehr gute Übereinstimmung der Simulationergebnisse mit den gemessenen Werten. Sowohl der Verlauf der Steifigkeit als auch der Dämpfungsmomente korreliert hervorragend mit den gemessenen Daten.

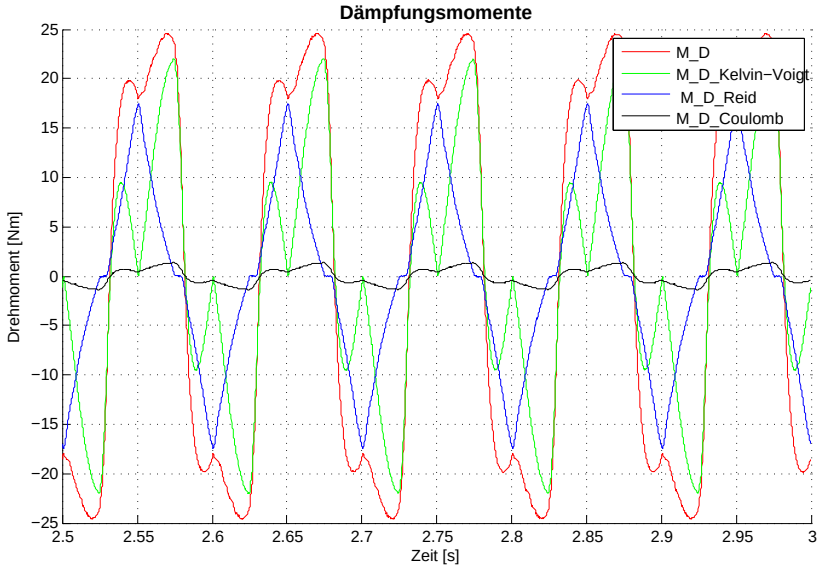


Abbildung 6.3: Dämpfungsmomente bei 300 Nm Wechselmoment, 10Hz

In Abbildung 6.3 zeigt sich der zeitliche Verlauf der gesamten sowie der einzelnen Dämpfungskräfte für ein Wechselmoment von 300Nm bei 10Hz. Der Einfluss der Modellierung der Dämpfungsmomente mittels PT1-Übertragungsfunktion ist beim Nulldurchgang der Dämpfungsmomente deutlich zu erkennen. Ohne diese Modellierung würde ein massives numerisches Schwingen um die Nulllage stattfinden, da die Signumfunktion der Winkelgeschwindigkeit $\dot{\varphi}$ einen großen Drehmomentensprung einleiten würde, siehe Abschnitt 3.2.4.

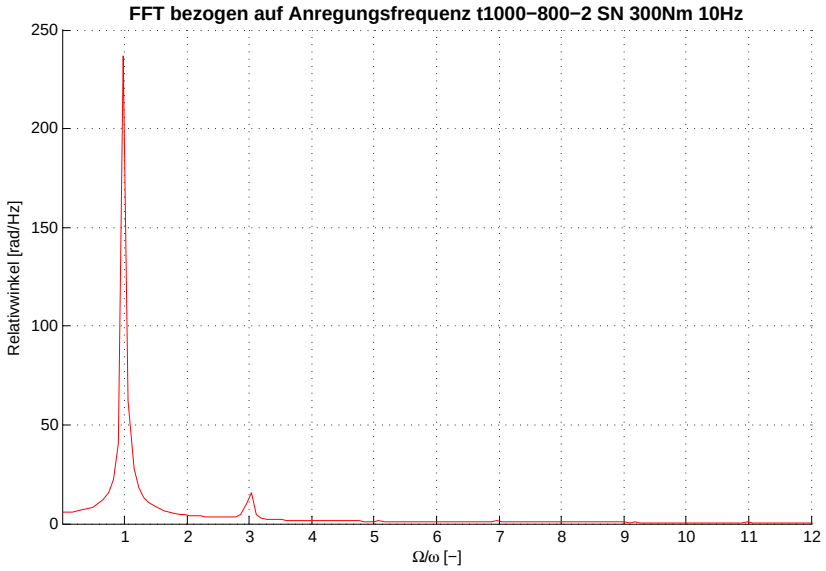


Abbildung 6.4: FFT des Relativwinkels bei 300 Nm Wechselmoment, 10 Hz

In der Fast Fourier Transformation des Relativwinkels ist ein weiterer Effekt der Nichtlinearität zu erkennen. Es treten super-harmonische Frequenzen auf, die durch ein Vielfaches der

Erregerfrequenz beschrieben werden. Besonders die dritte super-harmonische Frequenz tritt verhältnismäßig stark auf.

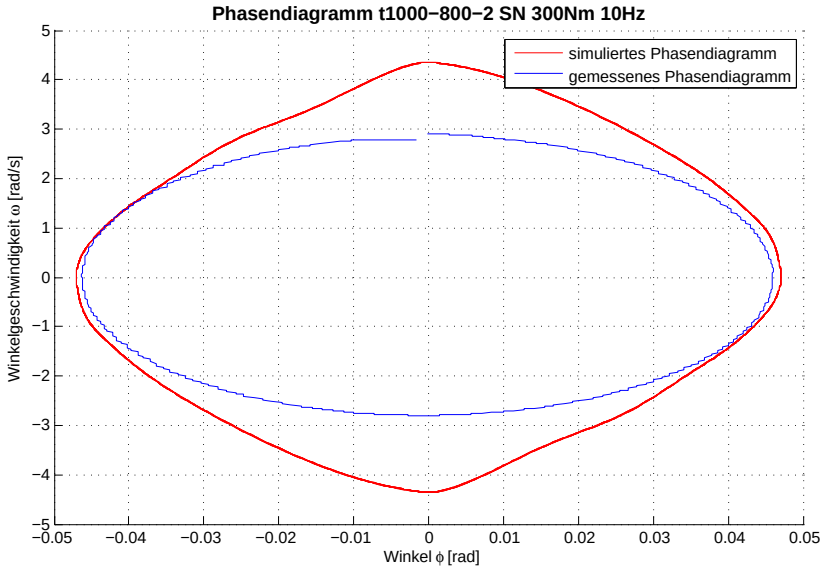


Abbildung 6.5: Phasendiagramm bei 300 Nm Wechselmoment, 10 Hz

Im Phasendiagramm ist die Amplitude des Winkel auf der Ordinate und die der Winkelgeschwindigkeit auf der Abszisse aufgetragen. Es ist deutlich zu sehen, dass ein regelmäßiges Phasendiagramm vorliegt und kein chaotischer Zustand herrscht. In der Simulation treten bei kleinen Winkeln höhere Geschwindigkeiten auf. Dies kann auf die Modellierung der Dämpfung zurückgeführt werden, da bei geringen Winkel niedrigere Dämpfungsmomente, verglichen zur Messung, verwendet werden, siehe Abschnitt 4.4.

- 450Nm Wechselmoment mit 10Hz Erregerfrequenz

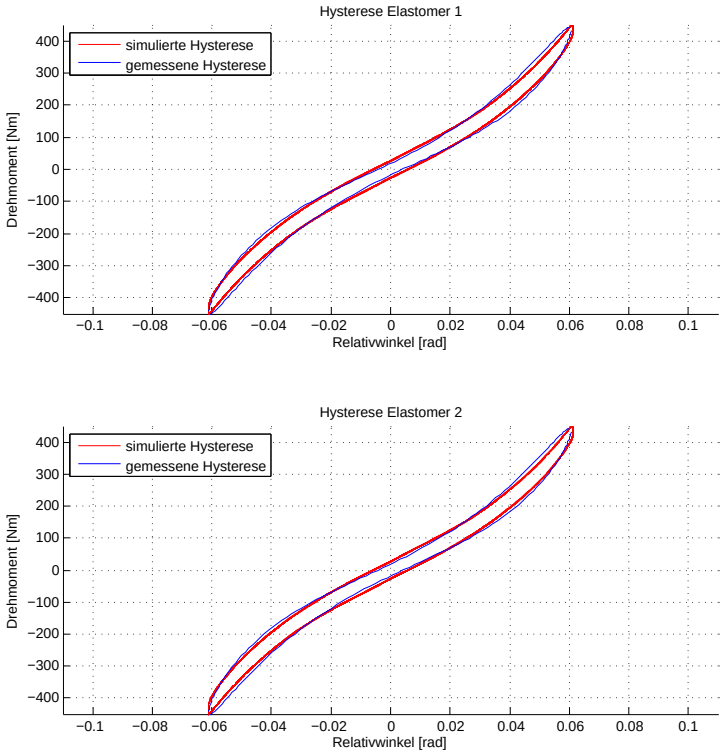


Abbildung 6.6: Hysterese bei 450 Nm Wechselmoment, 10 Hz

Die gute Übereinstimmung der Hysterese konnte auch bei der Messung von 450 Nm bei 10 Hz erzielt werden.

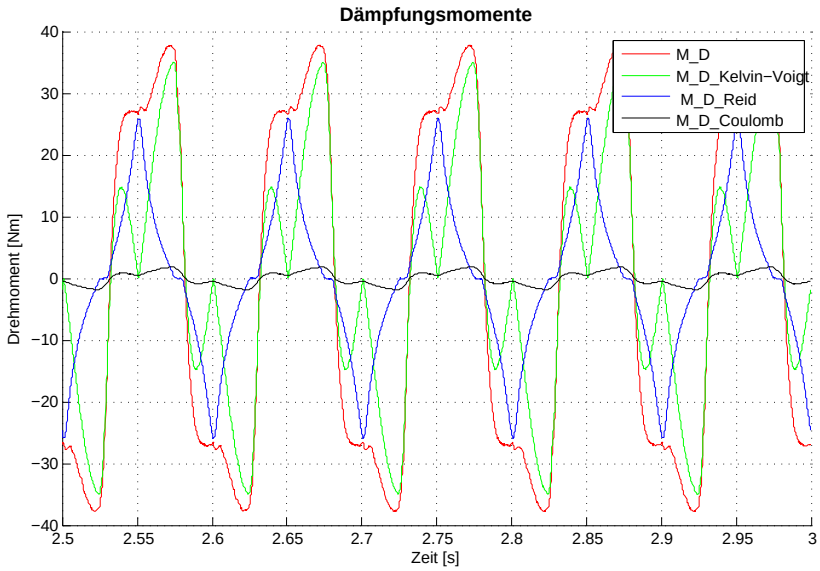


Abbildung 6.7: Dämpfungsmomente bei 450 Nm Wechselmoment, 10 Hz

In Abbildung 6.7 zeigt sich der zeitliche Verlauf der gesamten sowie der einzelnen Dämpfungsmomente für ein Wechselmoment von 450 Nm bei 10 Hz.

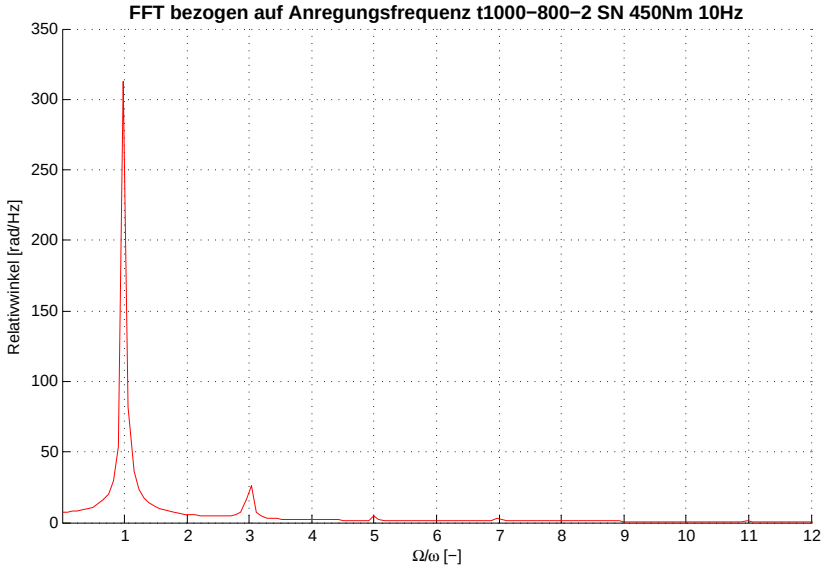


Abbildung 6.8: FFT des Relativwinkels bei 450 Nm Wechselmoment, 10 Hz

Auch bei 450 Nm Wechselmoment zeigt sich die super-harmonische dritte Ordnung.

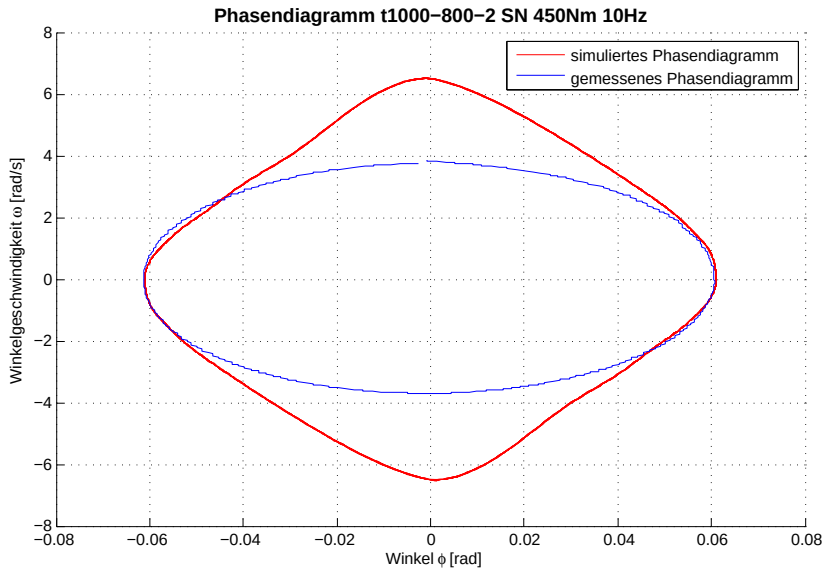


Abbildung 6.9: Phasendiagramm bei 450 Nm Wechselmoment, 10 Hz

Bei 450Nm beschreibt das Phasendiagramm das gleiche Verhalten wie zuvor.

- 600 Nm Wechselmoment mit 10 Hz Erregerfrequenz

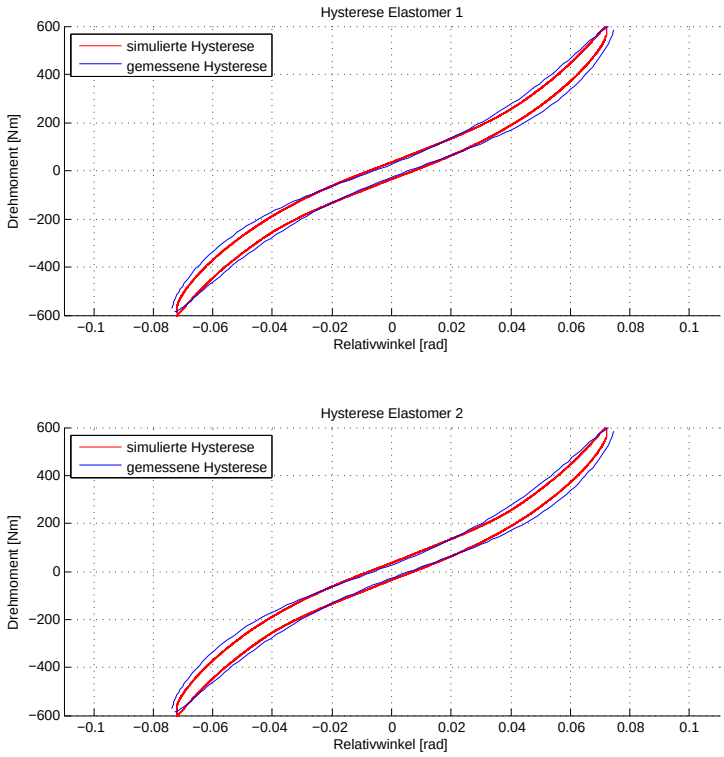


Abbildung 6.10: Hysterese bei 600 Nm Wechselmoment, 10 Hz

Auch bei dieser Messung liefert die Hysterese nahezu deckungsgleiche Ergebnisse.

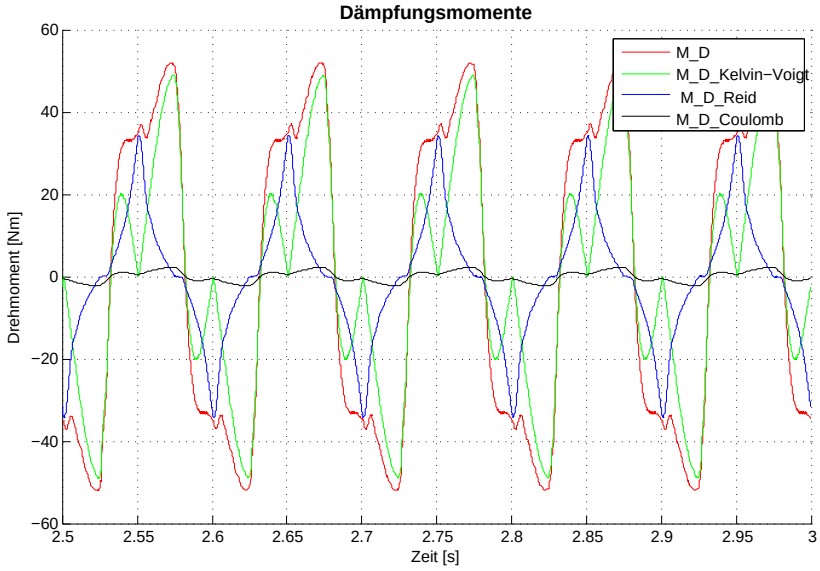


Abbildung 6.11: Dämpfungsmomente bei 600 Nm Wechselmoment, 10 Hz

In Abbildung 6.11 zeigt sich der zeitliche Verlauf der gesamten sowie der einzelnen Dämpfungskräfte für 600 Nm bei 10 Hz.

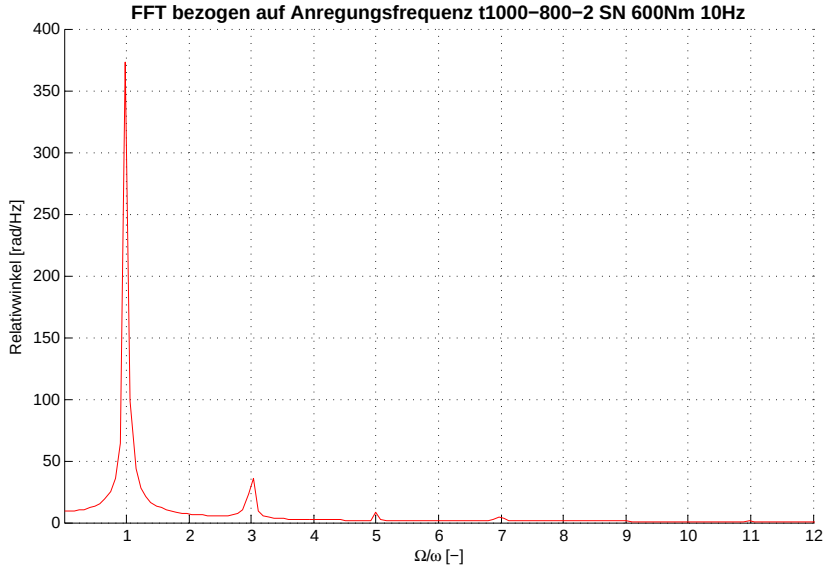


Abbildung 6.12: FFT des Relativwinkels bei 600 Nm Wechselmoment, 10 Hz

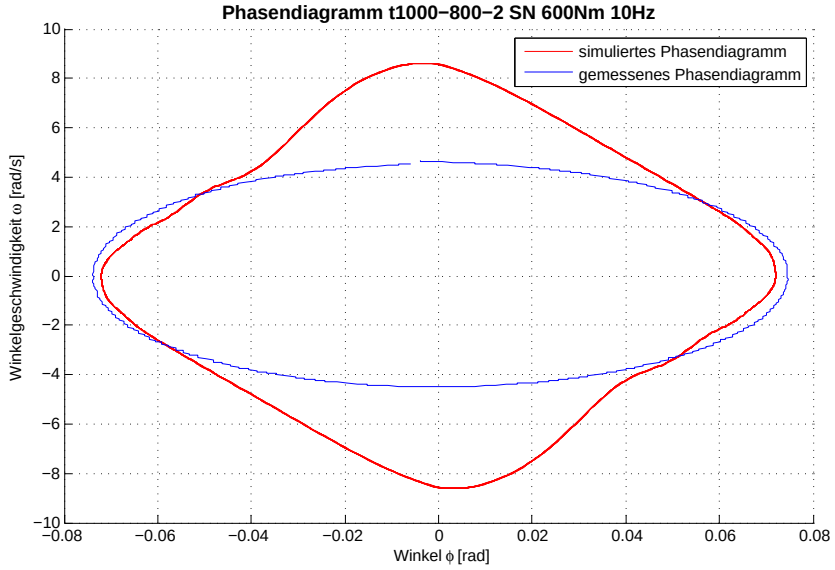


Abbildung 6.13: Phasendiagramm bei 600 Nm Wechselmoment, 10 Hz

Die FFT und das Phasendiagramm zeigen qualitativ gleiches Verhalten wie zuvor beschrieben.

6.2 Gesamtprüfstand

6.2.1 Messung mit linearer Reich TOK Kupplung

Zum Abgleichen des Modelica Modells wurde zu Beginn eine lineare Elastomerkupplung am Prüfstand vermessen. Hierbei wurden die Drehzahlen an beiden Kupplungsseiten induktiv gemessen. Zusätzlich wurden mittels Beschleunigungssensoren am Motor, am Dynamomenter und am Fundament weitere Daten aufgenommen. Der Prüflauf war ein Volllast Hochlauf von Leerlauf- bis zur Maximaldrehzahl. Zur Bestimmung der Aufbau Eigenfrequenzen wurde eine Modalanalyse mit Hammerschlag durchgeführt.

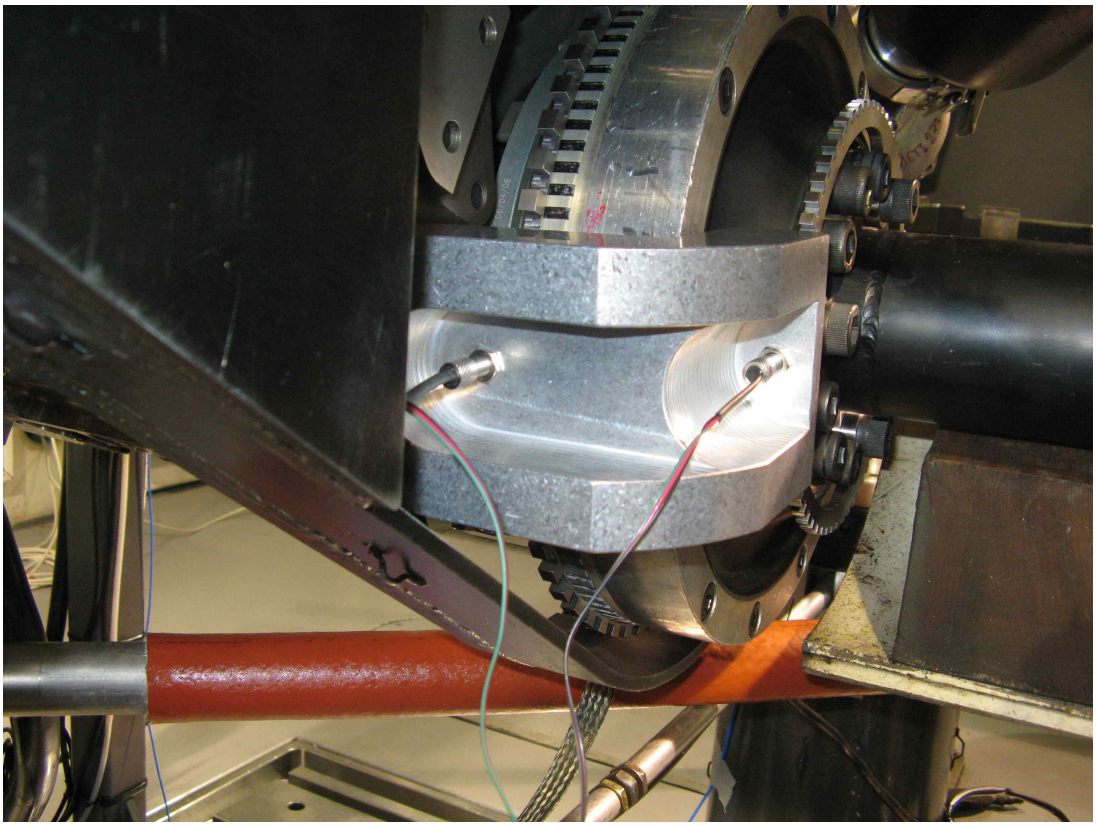


Abbildung 6.14: Messaufbau Prüfstand

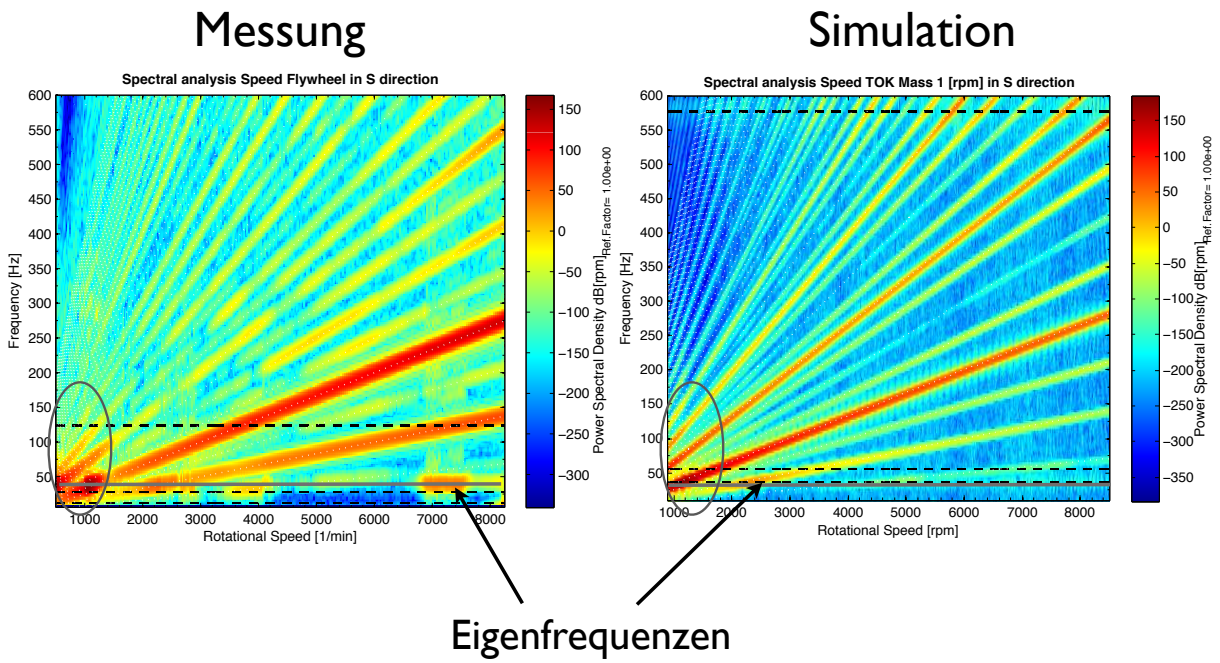


Abbildung 6.15: Messungs - Simulationsvergleich Prüfstand TOK Kupplung

Die Ergebnisse der Berechnung zeigen eine gute Übereinstimmung bei der Lage der Eigenfrequenzen. In den Messungen sind im Gegensatz zu der Simulation noch andere Schwingungen erkennbar. Insbesonders ist die Resonanzstelle bei 7000rpm hervorzuheben, die in einer Modalanalyse als eine Biegeeigenfrequenz der Kurbelwelle festgestellt werden konnte. Aufgrund dieser Ergebnisse wurde dieser Prüfstand für die weiteren Vergleiche ausgewählt.

6.2.2 Messung mit tectos t1000-800-2-WN

Die Messungen der nichtlinearen Elastomerkupplung t1000-800-2-WN auf diesem Prüfstand konnten aufgrund eines Schwingungsproblems der Kurbelwelle nur einmal durchgeführt werden. In Abbildung 6.16 und 6.17 ist der Messaufbau dargestellt. Es wurden die Drehzahlsignale mittels induktiver Messmethoden als auch Beschleunigungen am Motor, am Fundament und am Dynamometer unter Volllast gemessen.

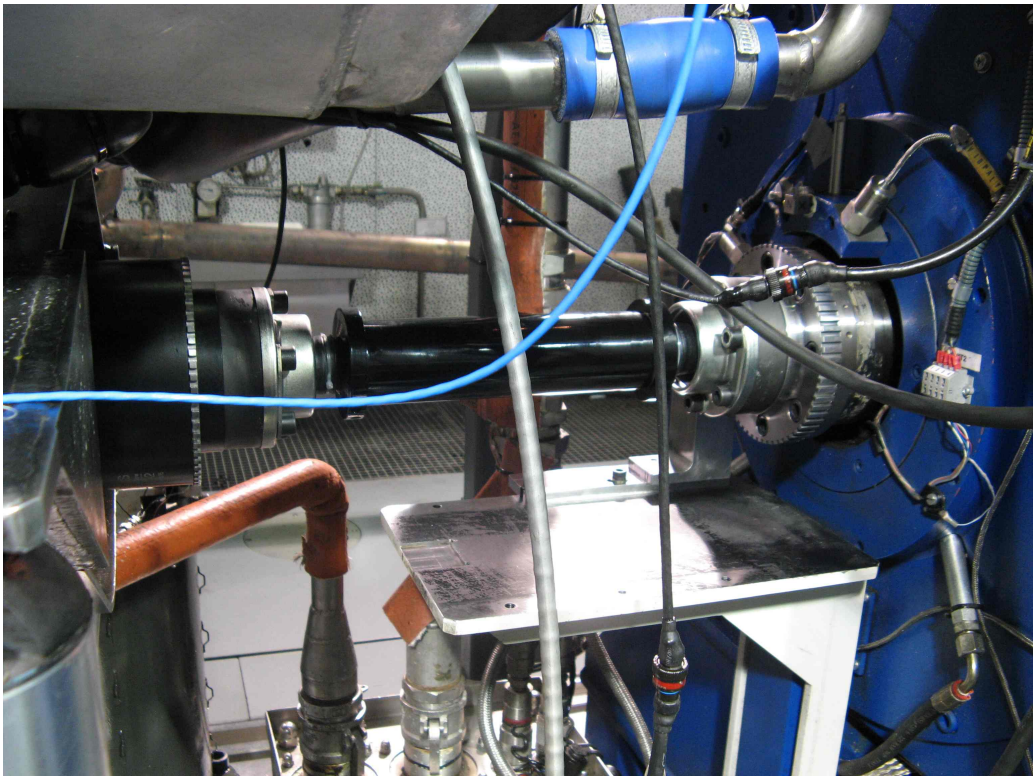


Abbildung 6.16: Messaufbau Prüfstand

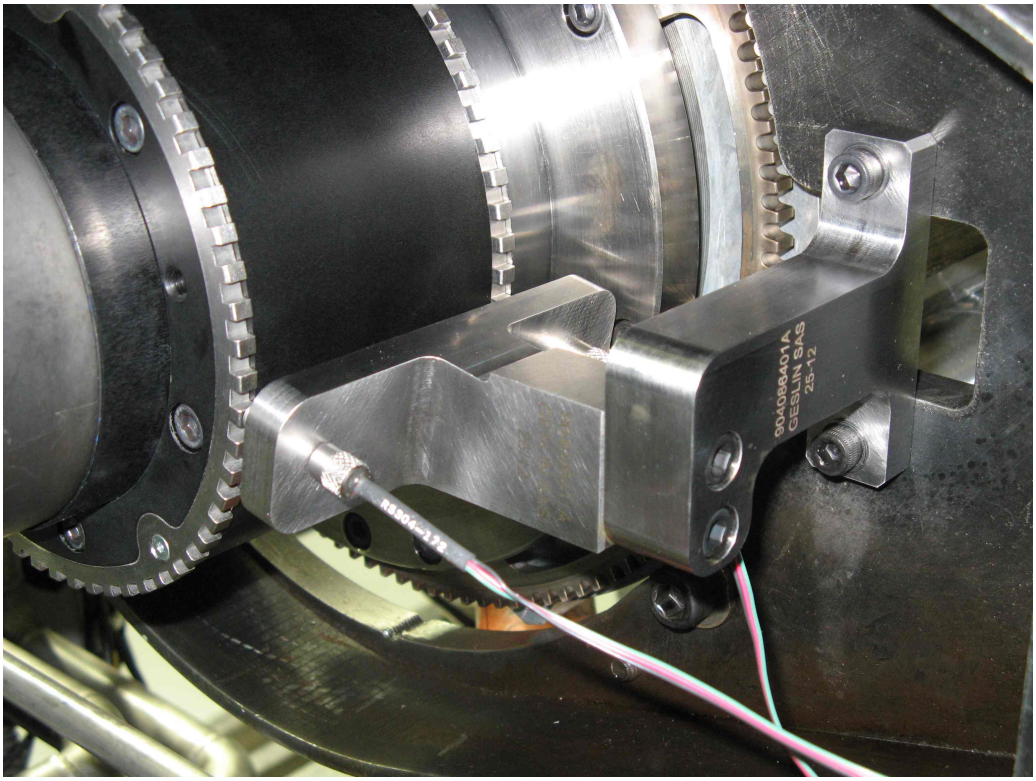


Abbildung 6.17: Messaufbau Prüfstand t1000-800-2-WN

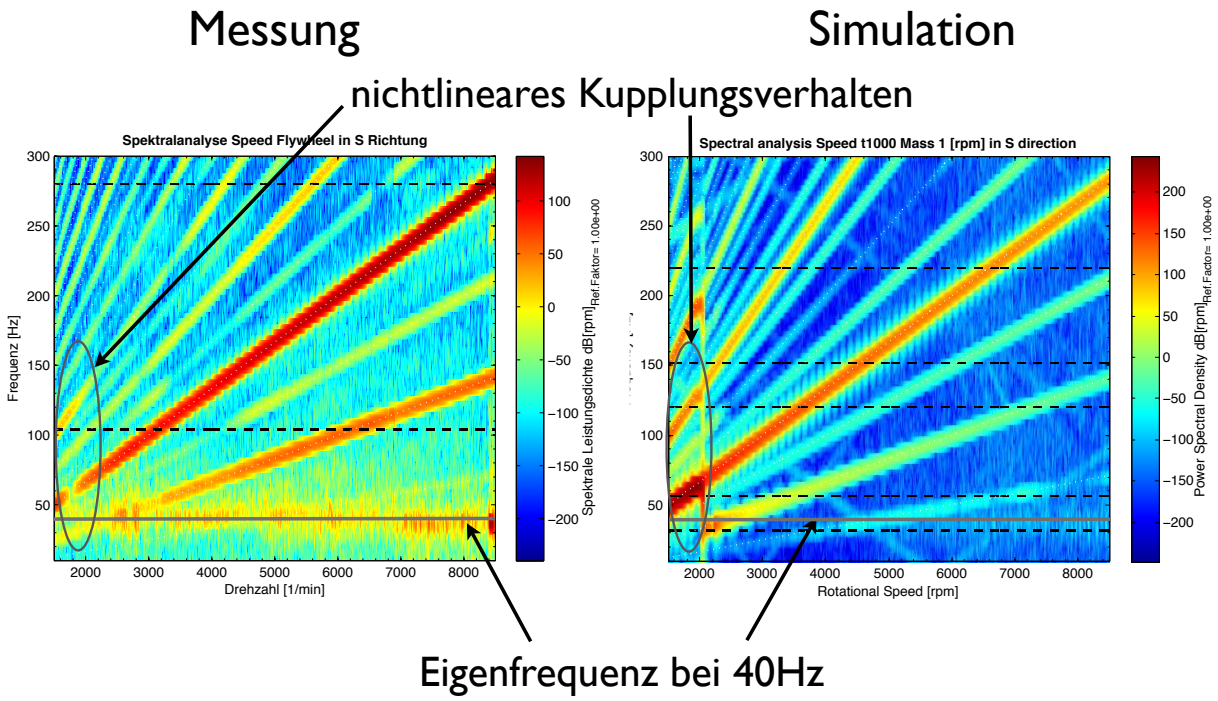


Abbildung 6.18: Messungs - Simulationsvergleich Prüfstand t1000-800-2-WN

Die Abbildung 6.18 zeigt den Vergleich der Campbell Diagramme der Messung und der Simulation. Die erste Torsionseigenfrequenz bei 40Hz kann sowohl in der Messung als auch in der Simulation festgestellt werden. Auch das nichtlineare progressive Kupplungsverhalten ist im Bereich von 2000rpm erkennbar. In der Messung tritt jedoch dieses Verhalten bei einer etwas niedrigeren Drehzahl und mit deutlich niedrigeren Pegeln auf. In der Simulation tritt ein Phänomen bei 2000rpm auf, das alle Frequenzen anregt. Die Ursache für dieses Verhalten konnte nicht eindeutig bestimmt werden. In der Messung wird die erste Ordnung stärker angeregt als in der Simulation. Dies kann unter Umständen seine Ursache im unterschiedlichen Saugrohrdruck der einzelnen Zylinder haben.

Anhang A

Code

Listing A.1: createCLut.m

```
function createCLut(varargin)
%CREATECLUT Create lookup tables for Open Modelica from MDL.
%
%
%  mdl = CREATECLUT
%    Read a MAT file, a file open dialog is used to select
%    the file
%
%  mdl = CREATECLUT(filename)
%    Read a MAT file specified by filename
%

%--------------------------------------------------------
% (c) 2012 by tectos gmbh, 2011-02-02, MK
%--------------------------------------------------------

if nargin > 0
  fname = varargin{1};
else
  [fname, fpath, FilterIndex] = uigetfile( ...
    {  '*.mat','tProgress real-time files (*.mat)'}, ...
    'Pick a file');
  if FilterIndex==0
    fprintf('Reading MAT file canceled by user.\n');
    return;
  end
  fname=fullfile(fpath,fname);
end
load(fname);

if nargin > 1
```

```
  lutName = varargin{2};
else
  lutName='CombustionPressure';
  % lutName='CompressionPressure';
end

cFname=sprintf('%s_data.h',lutName);

if strcmp(lutName, 'CompressionPressure')

  if exist(cFname,'file')
    delete(cFname);
  end

  fprintf('Writing output file <%s>...\n',cFname);
  fid = fopen(cFname, 'w');

  v=MDL.Engine.(lutName).v;
  x=MDL.Engine.(lutName).x;
  y=MDL.Engine.(lutName).y;
  z=MDL.Engine.(lutName).z;

  n1=length(x);
  n2=length(y);
  n3=length(z);

  VV=zeros(n2+1,(n1+1)*n3);

  for aa = 1:n3
    % Fill Data from n3 Vektor to Matrix
    VV(1,(aa*(n1+1)-n1))=z(aa);

    % Fill Data from n1 Vektor to Matrix
    for bb = 1:n2
      VV((1+bb),(aa*(n1+1)-n1))=y(bb);
    end

    % Fill up Data from v -> VV
    for cc = 1:n2
      for dd = 1:n1
        VV((1+cc),((n1+1)*aa-n1+dd)) = v(cc,((n1)*aa-n1+dd));
      end
    end

    % Fill Data from n2 Vektor to Matrix
    for ee = 1:n1
      VV(1,((1+n1)*aa-n1+ee))=x(ee);
    end
  end
```

```
  WW=[];
  WW=[WW sprintf('{')];
   for ii = 1:(n2+1)
     WW=[WW sprintf('%e,',VV(ii,:))];
     if ii < n2+1
       WW=[WW(1:end-1) sprintf(',\\\n')];
     else
       WW=[WW(1:end-1) sprintf('}')];
     end
   end
else % CombustionPressure->shift matrix -360deg- 360deg to 0deg - 720deg
   if exist(cFname,'file')
     delete(cFname);
   end

   fprintf('Writing output file <%s>...\n',cFname);
   fid = fopen(cFname, 'w');

   v=MDL.Engine.(lutName).v;
   x=MDL.Engine.(lutName).x;
   y=MDL.Engine.(lutName).y;
   z=MDL.Engine.(lutName).z;

   n1=length(x);
   n2=length(y);
   n3=length(z);

   VV=zeros(n2+1,(n1+1)*n3);

   for aa = 1:n3
     % Fill Data from n3 Vektor to Matrix
     VV(1,(aa*(n1+1)-n1))=z(aa);

     % Fill Data from n1 Vektor to Matrix
     for bb = 1:n2
       VV((1+bb),(aa*(n1+1)-n1))=y(bb);
     end

     % Fill up Data from v -> VV
     for cc = 1:n2
       for dd = 1
         VV((1+cc),((n1+1)*aa-n1+dd)) = v(cc,((n1)*aa-n1+dd));
       end
       for dd = 2:361
         VV((1+cc),((n1+1)*aa-n1+dd)) = v(cc,((n1)*aa-n1+dd)+360);
       end
       for dd = 362:721
         VV((1+cc),((n1+1)*aa-n1+dd)) = v(cc,((n1)*aa-n1+dd)-360);
```

```
      end
    end

    % Fill Data from n2 Vektor to Matrix
    for ee = 1:n1
      VV(1,((1+n1)*aa-n1+ee))=x(ee);
    end
  end

  WW=[];
  WW=[WW sprintf('{')];
  for ii = 1:(n2+1)
    WW=[WW sprintf('%e,',VV(ii,:))];
    if ii < n2+1
      WW=[WW(1:end-1) sprintf(',\\\n')];
    else
      WW=[WW(1:end-1) sprintf('}')];
    end
  end
end

XX=[];
XX=[XX sprintf('#ifndef %s_DATA_H\n',upper(lutName))];
XX=[XX sprintf('#define %s_DATA_H\n\n',upper(lutName))];
XX=[XX sprintf('static double %s_v[%d] = %s;\n',lutName,...
  (n1+1)*n3*(n2+1),WW)];
XX=[XX sprintf('static int %s_x1size = %d;\n',lutName,n1)];
XX=[XX sprintf('static int %s_x2size = %d;\n',lutName,n2)];
XX=[XX sprintf('static int %s_x3size = %d;\n\n',lutName,n3)];
XX=[XX sprintf('#endif //INTERPOLATION3D_DATA_H\n')];
fprintf(fid,'%s',XX);

fclose(fid);
```

Listing A.2: interpolation3D.c

```
double interpolation3D(double* v, int x1size, int x2size,
                                          int x3size, double x1,
                                          double x2, double x3)
{
        double  value = 0.0;
  int     p1x1;
  int           p2x1;
  int           p1x2;
  int           p2x2;
  int           p1x3;
  int           p2x3;
  int           p12[2]= {1,1};
  double        vx2[2]= {0.0,0.0};
  double        vx1IP1  = 0.0;
  double        vx1IP2  = 0.0;
  int           i;
  double  v1IP1;
  double  v2IP1;
  int           x1IP1;
  int           x2IP1;
  double  v1IP2;
  double  v2IP2;
  int           x1IP2;
  int           x2IP2;
  int           x1IP3;
  int           x2IP3;
  int           x1IP4;
  int           x2IP4;

  p1x1  = 0;
        p2x1    = x1size-1;
        p1x2    = 0;
        p2x2    = x2size-1;
        p1x3    = 0;
        p2x3    = x3size-1;
  while ((p2x3 - p1x3) > 1)
  {
        int p = ((p1x3 + p2x3) / 2);
    if (x3 < v[(x1size+1)*p])
      p2x3 = p;
    else
        p1x3 = p;
  }
  p12[0] = p1x3;
  p12[1] = p2x3;

  for (i=0; i<2; i++)
```

```
  {
        while ((p2x1 - p1x1) > 1)
    {
        int p = ((p1x1 + p2x1) / 2);
      if (x1 < v[(x1size+1)*p12[i]+1+p])
        p2x1 = p;
      else
        p1x1 = p;
    }

    while ((p2x2 - p1x2) > 1)
    {
      int p = ((p1x2 + p2x2) / 2);
      if (x2 < v[(x1size+1)*x3size*(p+1)+(x1size+1)*p12[i]])
        p2x2 = p;
      else
        p1x2 = p;
    }

    v1IP1  = v[(x1size+1)*x3size*(p1x2+1)+(x1size+1)*p12[i]+(p1x1+1)];
    v2IP1  = v[(x1size+1)*x3size*(p1x2+1)+(x1size+1)*p12[i]+(p2x1+1)];
    x1IP1  = v[(x1size+1)*p12[i]+1+p1x1];
    x2IP1  = v[(x1size+1)*p12[i]+1+p2x1];

    v1IP2  = v[(x1size+1)*x3size*(p2x2+1)+(x1size+1)*p12[i]+(p1x1+1)];
    v2IP2  = v[(x1size+1)*x3size*(p2x2+1)+(x1size+1)*p12[i]+(p2x1+1)];
    x1IP2  = v[(x1size+1)*p12[i]+1+p1x1];
    x2IP2  = v[(x1size+1)*p12[i]+1+p2x1];

    x1IP3  = v[(x1size+1)*x3size*(p1x2+1)+(x1size+1)];
    x2IP3  = v[(x1size+1)*x3size*(p2x2+1)+(x1size+1)];

    x1IP4  = v[(x1size+1)*p12[0]];
    x2IP4  = v[(x1size+1)*p12[1]];

    vx1IP1 = v1IP1 + (v2IP1-v1IP1) / (x2IP1-x1IP1) * (x1-x1IP1);
    vx1IP2 = v1IP2 + (v2IP2-v1IP2) / (x2IP2-x1IP2) * (x1-x1IP2);

    vx2[i] = vx1IP1 + (vx1IP2-vx1IP1) / (x2IP3-x1IP3) * (x2-x1IP3);
  }

  value = vx2[0] + (vx2[1] - vx2[0]) / (x2IP4-x1IP4) * (x3-x1IP4);

        return value;
};
```

Listing A.3: ipo3DTotal.mo

```
package Modelica "Modelica Standard Library (Version 3.2)"
extends Modelica.Icons.Package;

  package Blocks
  "Library of basic input/output control blocks
  (continuous, discrete, logical, table blocks)"
  import SI = Modelica.SIunits;
  extends Modelica.Icons.Package;

    package Interfaces
    "Library of connectors and partial models for input/output blocks"
      import Modelica.SIunits;
        extends Modelica.Icons.InterfacesPackage;

    connector RealInput = input Real "'input Real' as connector";

    connector RealOutput = output Real "'output Real' as connector";
    end Interfaces;
  end Blocks;

  package Icons "Library of icons"
    extends Icons.Package;

    partial package Package "Icon for standard packages"

    end Package;

    partial package InterfacesPackage
    "Icon for packages containing interfaces"
    //extends Modelica.Icons.Package;
    end InterfacesPackage;
  end Icons;

  package SIunits
  "Library of type and unit definitions based on SI units
   according to ISO 31-1992"
    extends Modelica.Icons.Package;
  end SIunits;
end Modelica;

package tModelicaLibMK

  package Functions

    function interpolation3D
      input Real x1;
      input Real x2;
```

```
      input Real x3;
      input Integer iTableID;
      output Real value;
      external "C" value = calcCylPressure(x1,x2,x3,iTableID);
    end interpolation3D;

    model ipo3D

      Modelica.Blocks.Interfaces.RealInput x1;
      Modelica.Blocks.Interfaces.RealInput x2;
      Modelica.Blocks.Interfaces.RealInput x3;
      Modelica.Blocks.Interfaces.RealOutput y;
      parameter Integer iTableID = 0 "0 = compression, 1 = combustion ";
    equation
      y = tModelicaLibMK.Functions.interpolation3D(
            x1,
            x2,
            x3,
            iTableID);
    end ipo3D;
  end Functions;
end tModelicaLibMK;
model tModelicaLibMK_Functions_ipo3D
 extends tModelicaLibMK.Functions.ipo3D;
  annotation(experiment(
    StopTime=1,
    NumberOfIntervals=500,
    Tolerance=0.0001,
    Algorithm="dassl"));
end tModelicaLibMK_Functions_ipo3D;
```

Listing A.4: t1000_800_2_nonlin_PT1_Reid_Kelvin_Coulomb_Total.mo

```
package Modelica "Modelica Standard Library (Version 3.2)"
extends Modelica.Icons.Package;

  package Blocks
  "Library of basic input/output control blocks
  (continuous, discrete, logical, table blocks)"
  import SI = Modelica.SIunits;
  extends Modelica.Icons.Package;

    package Discrete
    "Library of discrete input/output blocks with fixed sample period"
      extends Modelica.Icons.Package;

      block ZeroOrderHold "Zero order hold of a sampled-data system"
        extends Interfaces.DiscreteSISO;
        output Real ySample(start=0, fixed=true);

      equation
        when {sampleTrigger, initial()} then
          ySample = u;
        end when;
        /* Define y=ySample with an infinitesimal delay to break potential
       algebraic loops if both the continuous and the discrete part have
       direct feedthrough
    */
        y = pre(ySample);
      end ZeroOrderHold;
    end Discrete;

    package Interfaces
    "Library of connectors and partial models for input/output blocks"
      import Modelica.SIunits;
        extends Modelica.Icons.InterfacesPackage;

    connector RealInput = input Real "'input Real' as connector";

    connector RealOutput = output Real "'output Real' as connector";

        partial block DiscreteBlockIcon
      "Graphical layout of discrete block component icon"

        end DiscreteBlockIcon;

        partial block DiscreteBlock "Base class of discrete control blocks"
          extends DiscreteBlockIcon;

          parameter SI.Time samplePeriod(min=100*Modelica.Constants.eps, start = 0.1)
```

```
      "Sample period of component";
        parameter SI.Time startTime=0 "First sample time instant";
  protected
        output Boolean sampleTrigger "True, if sample time instant";
        output Boolean firstTrigger
      "Rising edge signals first sample instant";
      equation
        sampleTrigger = sample(startTime, samplePeriod);
        when sampleTrigger then
          firstTrigger = time <= startTime + samplePeriod/2;
        end when;
      end DiscreteBlock;

      partial block DiscreteSISO
    "Single Input Single Output discrete control block"

        extends DiscreteBlock;

        Modelica.Blocks.Interfaces.RealInput u "Continuous input signal";
        Modelica.Blocks.Interfaces.RealOutput y "Continuous output signal";
      end DiscreteSISO;
  end Interfaces;
end Blocks;

package Mechanics
"Library of 1-dim. and 3-dim. mechanical components
(multi-body, rotational, translational)"
extends Modelica.Icons.Package;

  package Rotational
  "Library to model 1-dimensional, rotational mechanical systems"
    extends Modelica.Icons.Package;
    import SI = Modelica.SIunits;

    package Components "Components for 1D rotational mechanical drive trains"
      extends Modelica.Icons.Package;

      model Inertia "1D-rotational component with inertia"
        import SI = Modelica.SIunits;
        Rotational.Interfaces.Flange_a flange_a "Left flange of shaft";
        Rotational.Interfaces.Flange_b flange_b "Right flange of shaft";
        parameter SI.Inertia J(min=0, start=1) "Moment of inertia";
        parameter StateSelect stateSelect=StateSelect.default
        "Priority to use phi and w as states";
        SI.Angle phi(stateSelect=stateSelect)
        "Absolute rotation angle of component";
        SI.AngularVelocity w(stateSelect=stateSelect)
        "Absolute angular velocity of component (= der(phi))";
        SI.AngularAcceleration a
```

```
    "Absolute angular acceleration of component (= der(w))";

  equation
    phi = flange_a.phi;
    phi = flange_b.phi;
    w = der(phi);
    a = der(w);
    J*a = flange_a.tau + flange_b.tau;
  end Inertia;
end Components;

package Interfaces
"Connectors and partial models for 1D rotational mechanical components"
  extends Modelica.Icons.InterfacesPackage;

  connector Flange_a
  "1-dim. rotational flange of a shaft (filled square icon)"
    SI.Angle phi "Absolute rotation angle of flange";
    flow SI.Torque tau "Cut torque in the flange";
  end Flange_a;

  connector Flange_b
  "1-dim. rotational flange of a shaft (non-filled square icon)"
    SI.Angle phi "Absolute rotation angle of flange";
    flow SI.Torque tau "Cut torque in the flange";
  end Flange_b;

  partial model PartialTwoFlanges
  "Partial model for a component with two rotational 1-dim. shaft flanges"

    Flange_a flange_a "Flange of left shaft";
    Flange_b flange_b "Flange of right shaft";
  end PartialTwoFlanges;

  partial model PartialCompliantWithRelativeStates
  "Partial model for the compliant connection of two rotational 1-dim. shaft f
   langes where the relative angle and speed are used as preferred states"

    Modelica.SIunits.Angle phi_rel(start=0, stateSelect=stateSelect, nominal=phi_nominal)
    "Relative rotation angle (= flange_b.phi - flange_a.phi)";
    Modelica.SIunits.AngularVelocity w_rel(start=0, stateSelect=stateSelect)
    "Relative angular velocity (= der(phi_rel))";
    Modelica.SIunits.AngularAcceleration a_rel(start=0)
    "Relative angular acceleration (= der(w_rel))";
    Modelica.SIunits.Torque tau "Torque between flanges (= flange_b.tau)";
    Flange_a flange_a
    "Left flange of compliant 1-dim. rotational component";
    Flange_b flange_b
    "Right flange of compliant 1-dim. rotational component";
```

```
        parameter SI.Angle phi_nominal(displayUnit="rad")=1e-4
        "Nominal value of phi_rel (used for scaling)";
        parameter StateSelect stateSelect=StateSelect.prefer
        "Priority to use phi_rel and w_rel as states";

      equation
        phi_rel = flange_b.phi - flange_a.phi;
        w_rel = der(phi_rel);
        a_rel = der(w_rel);
        flange_b.tau = tau;
        flange_a.tau = -tau;
      end PartialCompliantWithRelativeStates;
    end Interfaces;
  end Rotational;
end Mechanics;

package Math
"Library of mathematical functions (e.g., sin, cos) and of functions
operating on vectors and matrices"
import SI = Modelica.SIunits;
extends Modelica.Icons.Package;

function asin "Inverse sine (-1 <= u <= 1)"
  extends baseIcon2;
  input Real u;
  output SI.Angle y;

external "builtin" y=  asin(u);
end asin;

partial function baseIcon2
  "Basic icon for mathematical function with y-axis in middle"

end baseIcon2;
end Math;

package Constants
"Library of mathematical constants and constants of nature (e.g., pi, eps, R, sigma)"
  import SI = Modelica.SIunits;
  import NonSI = Modelica.SIunits.Conversions.NonSIunits;
  extends Modelica.Icons.Package;

  final constant Real pi=2*Modelica.Math.asin(1.0);

  final constant Real eps=1.e-15 "Biggest number such that 1.0 + eps = 1.0";
end Constants;

package Icons "Library of icons"
```

```
    extends Icons.Package;

    partial package Package "Icon for standard packages"

    end Package;

    partial package InterfacesPackage "Icon for packages containing interfaces"
    //extends Modelica.Icons.Package;
    end InterfacesPackage;

    partial package Library2
    "This icon will be removed in future Modelica versions, use Package instead"

    end Library2;
  end Icons;

  package SIunits
  "Library of type and unit definitions based on SI units according to ISO 31-1992"
    extends Modelica.Icons.Package;

    package Conversions
    "Conversion functions to/from non SI units and type definitions of non SI units"
      extends Modelica.Icons.Package;

      package NonSIunits "Type definitions of non SI units"
        extends Modelica.Icons.Package;
      end NonSIunits;
    end Conversions;

    type Angle = Real (
        final quantity="Angle",
        final unit="rad",
        displayUnit="deg");

    type Time = Real (final quantity="Time", final unit="s");

    type AngularVelocity = Real (
        final quantity="AngularVelocity",
        final unit="rad/s");

    type AngularAcceleration = Real (final quantity="AngularAcceleration", final unit=
            "rad/s2");

    type MomentOfInertia = Real (final quantity="MomentOfInertia", final unit=
          "kg.m2");

    type Inertia = MomentOfInertia;

    type Torque = Real (final quantity="Torque", final unit="N.m");
```

```
    type RotationalDampingConstant=Real
    (final quantity="RotationalDampingConstant", final unit="N.m.s/rad");
  end SIunits;
end Modelica;

package tModelicaLibMK

  package Components
    extends Modelica.Icons.Library2;

    package Driveline

      package ElasticCoupling

        model t1000_800_2_nonlin_PT1_Reid_Kelvin_Coulomb_test2
          extends Modelica.Mechanics.Rotational.Interfaces.PartialTwoFlanges;
          parameter Real c_0_O;
          parameter Real c_1_O;
          parameter Real c_2_O;
          parameter Real c_3_O;
          parameter Real d_visk_1_O;
          parameter Real d_visk_0_O;
          parameter Real M_Coulomb_1_O;
          parameter Real M_Coulomb_0_O;
          parameter Real k_Reid_1_O;
          parameter Real k_Reid_0_O;

          Modelica.Mechanics.Rotational.Components.Inertia Inertia2(J=2.75E-3);
          Modelica.Mechanics.Rotational.Components.Inertia Inertia1(J=2.79E-3);
          Modelica.Mechanics.Rotational.Components.Inertia Inertia3(J=4.32E-3);
          SpringDamper_MatDamp_PT1_Reid_Kelvin_Coulomb_test2 springDamper_MatDamp1(
            c_const0_O=c_0_O,
            c_const1_O=c_1_O,
            c_const2_O=c_2_O,
            c_const3_O=c_3_O,
            d_visk_1_O=d_visk_1_O,
            d_visk_0_O=d_visk_0_O,
            M_Coulomb_1_O=M_Coulomb_1_O,
            M_Coulomb_0_O=M_Coulomb_0_O,
            k_Reid_1_O=k_Reid_1_O,
            k_Reid_0_O=k_Reid_0_O);
          SpringDamper_MatDamp_PT1_Reid_Kelvin_Coulomb_test2 springDamper_MatDamp2(
            c_const0_O=c_0_O,
            c_const1_O=c_1_O,
            c_const2_O=c_2_O,
            c_const3_O=c_3_O,
            d_visk_1_O=d_visk_1_O,
            d_visk_0_O=d_visk_0_O,
```

```
      M_Coulomb_1_O=M_Coulomb_1_O,
      M_Coulomb_0_O=M_Coulomb_0_O,
      k_Reid_1_O=k_Reid_1_O,
      k_Reid_0_O=k_Reid_0_O);
    Modelica.Blocks.Interfaces.RealInput n;
  equation
    connect(Inertia3.flange_a, flange_b);
    connect(Inertia1.flange_b, flange_a);
    connect(Inertia1.flange_a, springDamper_MatDamp1.flange_a);

    connect(springDamper_MatDamp1.flange_b, Inertia2.flange_b);
    connect(Inertia2.flange_a, springDamper_MatDamp2.flange_a);
    connect(springDamper_MatDamp2.flange_b, Inertia3.flange_b);
    connect(n, springDamper_MatDamp2.n);
    connect(n, springDamper_MatDamp1.n);
  end t1000_800_2_nonlin_PT1_Reid_Kelvin_Coulomb_test2;

  model SpringDamper_MatDamp_PT1_Reid_Kelvin_Coulomb_test2
  "Linear 1D rotational spring and damper in parallel"
    extends
    Modelica.Mechanics.Rotational.Interfaces.PartialCompliantWithRelativeStates;
    import SI = Modelica.SIunits;
    parameter Real c_const0_O(final min=0, start=0)
    "Zero order constant of t1000 elastomere";
    parameter Real c_const1_O(final min=0, start=0)
    "First order constant of t1000 elastomere";
    parameter Real c_const2_O(final min=0, start=0)
    "Second order constant of t1000 elastomere";
    parameter Real c_const3_O(final min=0, start=0)
    "Third order constant of t1000 elastomere";
    Real d_visk(final min=0, start=0) "viskose Damoing factor";
    parameter Real d_visk_1_O;
    parameter Real d_visk_0_O;
    Real M_Coulomb "Coulomb Torque ";
    parameter Real M_Coulomb_1_O;
    parameter Real M_Coulomb_0_O;
    Real k_Reid "pitch constant of Reid Damping";
    parameter Real k_Reid_1_O;
    parameter Real k_Reid_0_O;
    parameter SI.RotationalDampingConstant d(final min=0, start=0)
    "Damping constant";
    SI.AngularVelocity w "Rotations Speed";
    parameter SI.Angle phi_rel0=0 "Unstretched spring angle";
    Real PT1u;
    Real PT1y;
    Real PT1t;
    Real lossPower;
    Modelica.SIunits.Torque tau_c "Spring torque";
    Modelica.SIunits.Torque tau_d "Damping torque";
```

```
        Modelica.SIunits.Torque tau_d_viskous "Viskous damping torque";
        Modelica.SIunits.Torque tau_d_Reid "Reid damping torque";
        Modelica.SIunits.Torque tau_d_Coulomb "Coulomb damping torque";
        Modelica.Blocks.Discrete.ZeroOrderHold tauHold(samplePeriod=1e-3);
        Modelica.Blocks.Interfaces.RealInput n;
      equation
        w = der(flange_a.phi);
        PT1t = 7e-2/(n/(2*Modelica.Constants.pi));
        PT1u = sign(w_rel);
        der(PT1y) = (PT1u - PT1y)/PT1t;
        tauHold.u = tau;
        tau_c = c_const3_0*(phi_rel - phi_rel0)^3+c_const2_0*(phi_rel - phi_rel0)^2
          +c_const1_0*(phi_rel - phi_rel0) + c_const0_0;
        d_visk = d_visk_1_0*abs(tauHold.y) + d_visk_0_0;
        tau_d_viskous = d_visk*w_rel;
        k_Reid = k_Reid_1_0*abs(tauHold.y) + k_Reid_0_0;
        tau_d_Reid = k_Reid*PT1y*abs(phi_rel - phi_rel0);
        M_Coulomb = M_Coulomb_1_0*abs(tauHold.y) + M_Coulomb_0_0;
        tau_d_Coulomb = M_Coulomb*PT1y;
        tau_d = tau_d_viskous + tau_d_Reid + tau_d_Coulomb;
        tau = tau_c + tau_d;
        lossPower = tau_d*w_rel;
      end SpringDamper_MatDamp_PT1_Reid_Kelvin_Coulomb_test2;
    end ElasticCoupling;
  end Driveline;
 end Components;
end tModelicaLibMK;
model tModelicaLibMK_Components_Driveline_ElasticCoupling_t1000_800_2_nonlin_
      PT1_Reid_Kelvin_Coulomb_test2
 extends tModelicaLibMK.Components.Driveline.ElasticCoupling.t1000_800_2_nonlin_
      PT1_Reid_Kelvin_Coulomb_test2;
 annotation(experiment(
   StopTime=1,
   NumberOfIntervals=500,
   Tolerance=0.0001,
   Algorithm="dassl"));
end tModelicaLibMK_Components_Driveline_ElasticCoupling_t1000_800_2_nonlin_
      PT1_Reid_Kelvin_Coulomb_test2;
```

Listing A.5: assignOuput.m

```
project.om.omOutputVar          = {};
project.om.omOutputVar(end+1,:) = {'TorqueFlywheel_Nm'     ...
        'Torque Flywheel [Nm]'};
project.om.omOutputVar(end+1,:) = {'Torque_TOK_m1_Nm'      ...
        'Torque TOK Mass 1 [Nm]'};
project.om.omOutputVar(end+1,:) = {'Torque_TOK_m2_Nm'      ...
        'Torque TOK Mass 2 [Nm]'};
project.om.omOutputVar(end+1,:) = {'TorqueDynometer_Nm'    ...
        'Torque Dynometer [Nm]'};
project.om.omOutputVar(end+1,:) = {'MeanSpeedEngine_rpm'   ...
        'Mean Speed Engine [Nm]'};
project.om.omOutputVar(end+1,:) = {'SpeedFlywheel_rpm'     ...
        'Speed Flywheel [Nm]'};
project.om.omOutputVar(end+1,:) = {'Speed_TOK_m1_rpm'      ...
        'Speed TOK Mass 1 [rpm]'};
project.om.omOutputVar(end+1,:) = {'Speed_TOK_m2_rpm'      ...
        'Speed TOK Mass 2 [rpm]'};
project.om.omOutputVar(end+1,:) = {'SpeedDynometer_rpm'    ...
        'Speed Dynometer [rpm]'};
project.om.omOutputVar(end+1,:) = {'Ploss_TOK_1_W'         ...
        'Power Loss t1000 [W]'};
project.om.omOutputVar(end+1,:) = {'nTestrun_rpm'          ...
        'Speed Testrun [rpm]'};
project.om.omOutputVar(end+1,:) = {'alphaTestrun_percent' ...
        'Alpha Testrun [%]'};
project.om.omOutputVar(end+1,:) = {'DynoSetTorque_Nm'      ...
        'Dynometer Set Torque [Nm]'};
project.om.omOutputVar(end+1,:) = {'Torque_Cyl1_Nm'        ...
        'Cylinder 1 Torque [Nm]'};
project.om.omOutputVar(end+1,:) = {'Torque_Cyl2_Nm'        ...
        'Cylinder 2 Torque [Nm]'};
project.om.omOutputVar(end+1,:) = {'Torque_Cyl3_Nm'        ...
        'Cylinder 3 Torque [Nm]'};
project.om.omOutputVar(end+1,:) = {'Torque_Cyl4_Nm'        ...
        'Cylinder 4 Torque [Nm]'};
```

Listing A.6: omimportpostproc.m

```
function omimportpostproc(varargin)
%
% imports results from Open Modelica Compiler and structures data for
% PostProc
%
global project

switch nargin
  case 0
    modelname = project.om.modelname;
  otherwise
    modelname = varargin{1};
end

load([modelname '_res.mat']);

nall = length(name');
nvar = length(data_2(:,1));
npar = length(data_1(:,1)); %nall-nvar;

project.om.result=[];
project.om.parameter=[];
project.om.model.result=[];
project.om.model.parameter=[];

time=data_2(1,:);

for ii = 1:(nvar-1)   % restructure time dependant simulation data
  pos = strcmp(deblank(name(:,ii+1)'), project.om.omOutputVar(:,1));
  posLine = find(pos,1);
  project.om.result(posLine).name  = project.om.omOutputVar{posLine,2};
  project.om.result(posLine).label = project.om.omOutputVar{posLine,1};
  project.om.result(posLine).time  = time;
  project.om.result(posLine).value = data_2(ii+1,:);
  project.om.result(posLine).description = deblank(description(:,ii+1)');

  resultname=deblank(name(:,ii+1)');
  resultname=strrep(resultname,'[','(');
  resultname=strrep(resultname,']',')');
  s=sprintf('project.om.model.result.%s.t=project.om.result(posLine).time;',...
    resultname);eval(s);
  s=sprintf('project.om.model.result.%s.v=project.om.result(posLine).value;',...
    resultname);eval(s);
end

for ii = 1:(npar-1)   % restructure constant simulation data
  project.om.parameter(ii).name  = deblank(name(:,(ii+nvar))');
```

```
    project.om.parameter(ii).value = data_1(ii+1,1);
    project.om.parameter(ii).description = deblank(description(:,ii+nvar)');

    paramname=project.om.parameter(ii).name;
    paramname=strrep(paramname,'[','(');
    paramname=strrep(paramname,']',')');
    s=sprintf('project.om.model.parameter.%s.v=project.om.parameter(ii).value;',...
      paramname);eval(s);
end
```

Listing A.7: t1000_extractHystersisWN.m

```
% t1000_extractParameterPsi according to DIN740-2

% Dynamic curves:
% T_off=0Nm:
%   - Some curve don't start propely at 0Nm => measurement offset
%
% Some do start at high torque

clc
global tResult

if isempty(tResult)
  t1000_loadData;
end

sampletime = 1/(5E3); % sampletime of measurment (t1000-800-2 - 5kHz)


param=[];

param(3).type='WN';
param(3).shore=(53+58)/2;
param(3).dynamic.Idx=130:138;

% param(4).type='NN';
% param(4).shore=(63+68)/2;
% param(4).dynamic.Idx=[];
% param(2).dynamic.Tmax=zeros(1,length(param(4).dynamic.Idx))*NaN;
% param(2).dynamic.Psi=zeros(1,length(param(4).dynamic.Idx))*NaN;
%
% param(3).type='SN';
% param(3).shore=(73+78)/2;
% param(3).dynamic.Idx=[65  66 67:75];
% param(3).dynamic.Idx=67:75;
%
% param(2).dynamic.Tmax=zeros(1,length(param(3).dynamic.Idx))*NaN;
% param(2).dynamic.Psi=zeros(1,length(param(3).dynamic.Idx))*NaN;
%
% param(6).type='UN';
% param(6).shore=(83+88)/2;
% param(6).dynamic.Idx=[];

J = 2.75e-3+2.79e-3;
Omega=10*2*pi; % Frequency of dynamic measurement

for ii = 1:length(param(3).dynamic.Idx)
  figure
```

```
  hold on
  grid on
  plot(tResult.Experiment(param(3).dynamic.Idx(ii)).Data(:,2)/2/180*pi,...
    tResult.Experiment(param(3).dynamic.Idx(ii)).Data(:,1));
  text(max(get(gca,'XLim')),max(get(gca,'YLim')),...
  sprintf('static torque: %g Nm, dynamic torque: %g Nm, Idx: %g',...
  tResult.Experiment(param(3).dynamic.Idx(ii)).Static,...
  tResult.Experiment(param(3).dynamic.Idx(ii)).Dynamic,...
  param(3).dynamic.Idx(ii)),...
  'VerticalAlignment','Top','HorizontalAlignment','Right');
  hold off
end
if exist('TorqueCorrection.mat','file')
  CheckTorque = false;
else
  CheckTorque = true;
end

hfig = figure;
hold on
grid on
TorqueCorrection = [];
KeepFile = 1;
for ii = 1:length(param(3).dynamic.Idx)
  figure(hfig)
  hLine = plot(tResult.Experiment(param(3).dynamic.Idx(ii)).Data(:,2)...
    /2/180*pi, tResult.Experiment(param(3).dynamic.Idx(ii)).Data(:,1),...
        '-r','Linewidth',2);
  formatFigures('Messkurven',...
    'Verdrehwinkel [rad]','Moment [Nm]');
  fprintf('static torque: %g Nm, dynamic torque: %g Nm, Idx: %g\n',...
    tResult.Experiment(param(3).dynamic.Idx(ii)).Static,...
    tResult.Experiment(param(3).dynamic.Idx(ii)).Dynamic,...
    param(3).dynamic.Idx(ii));
  if CheckTorque
    prompt = {'Static Torque:','Dynamic Torque:','Keep File:'};
    dlg_title = 'Input Torque';
    num_lines = 1;
    options.WindowStyle='normal';
    def = {sprintf('%g',...
      tResult.Experiment(param(3).dynamic.Idx(ii)).Static),...
      sprintf('%g',...
      tResult.Experiment(param(3).dynamic.Idx(ii)).Dynamic),...
      sprintf('%g',1)};
    answer = inputdlg(prompt,dlg_title,num_lines,def,options);
    if ~isempty(answer)
      Static = str2num(answer{1});
      Dynamic = str2num(answer{2});
      KeepFile = str2num(answer{3});
```

```
    else
      break
    end
    TorqueCorrection(ii).Idx = param(3).dynamic.Idx(ii);
    TorqueCorrection(ii).Static = Static;
    TorqueCorrection(ii).Dynamic = Dynamic;
    TorqueCorrection(ii).KeepFile = KeepFile;
  end
  if KeepFile
    %     subplot(1,2,1)
    set(hLine,'color','k','Linewidth',0.5);
    hold on;
  else
    delete(hLine);
  end
  xlim([-0.11 0.11]);
end
hold off
fig2pdf('images', 'Messkurven_WN')
close(hfig)

cc=['b' 'r' 'm' 'g' 'k' 'c']; % color scheme

% Block limit: cell completly filled with rubber
phi_max_deg=7.5;
phi_max_rad=phi_max_deg/180*pi;

cc=['b' 'r' 'm' 'g' 'k' 'c']; % color scheme

for ii=3:3%length(param)
  if isempty(param(ii).dynamic.Idx)
    continue;
  end

  for jj=1:length(param(ii).dynamic.Idx)
    param(ii).dynamic.torque.v = ...
      tResult.Experiment(param(ii).dynamic.Idx(jj)).Data(:,1);
    % f_sample=20.48kHz, 10Hz sine
    param(ii).dynamic.phi.v = ...
      tResult.Experiment(param(ii).dynamic.Idx(jj)).Data(:,2)/2/180*pi;
    % scale to one elastomere and deg to rad

    % get mean stiffness polynom
    [b1,m1] = unique(param(ii).dynamic.phi.v, 'first');
    for kk=1:length(b1)
      delta = 1e-3;
      v_kk=find(param(ii).dynamic.phi.v  < (b1(kk)+delta)...
        & param(ii).dynamic.phi.v  > (b1(kk)-delta));
      param(ii).dynamic.torque_unique.v(kk) = ...
```

```
      sum(param(ii).dynamic.torque.v(v_kk))/length(v_kk);
    param(ii).dynamic.phi_unique.v(kk)     = ...
      b1(kk);
  end
  p=polyfit(param(ii).dynamic.phi_unique.v, ...
    param(ii).dynamic.torque_unique.v,3);
  param(ii).dynamic.torque.v2 = smooth(param(ii).dynamic.torque.v,100);

  phi_pzc_idx = find(((param(ii).dynamic.phi.v(2:end)>0)-...
    (param(ii).dynamic.phi.v(1:end-1)>0)) > 0);
  % points with positive zero crossing

  a=param(ii).dynamic.torque.v2;

  torque_max_idx = find((a(2:end-1)>a(1:end-2)) .* ...
    (a(2:end-1)>a(3:end)));
  torque_min_idx = find((a(2:end-1)<a(1:end-2)) .* ...
    (a(2:end-1)<a(3:end)));
  phi = param(ii).dynamic.phi.v(phi_pzc_idx(1):phi_pzc_idx(2));
  tq  = param(ii).dynamic.torque.v(phi_pzc_idx(1):phi_pzc_idx(2));

  % Build Polynom 3. Order for derivation of phi
  t          = (1:1:length(phi))*sampletime';
  phi_x      = (1:1:length(phi))';
  p_phi      = polyfit(phi_x,phi,10);

  phi_poly   = p_phi(1).*phi_x.^10 + p_phi(2).*phi_x.^9 + ...
    p_phi(3).*phi_x.^8 + p_phi(4).*phi_x.^7 + p_phi(5).*phi_x.^6 ...
    + p_phi(6).*phi_x.^5 + p_phi(7).*phi_x.^4 + p_phi(8).*phi_x.^3 ...
    + p_phi(9).*phi_x.^2 + p_phi(10).*phi_x + p_phi(11);
  omega = zeros(length(phi),1);
  for mm=2:(length(phi_poly))
    omega(mm) = (phi_poly(mm)-phi_poly(mm-1))/sampletime;
  end

  omega_dot = zeros(length(phi),1);
  for mm=2:(length(omega))
    omega_dot(mm) = (omega(mm)-omega(mm-1))/sampletime;
  end

  phi_temp = min(param(ii).dynamic.phi.v):1e-4:...
    max(param(ii).dynamic.phi.v);
  param(ii).dynamic.torque_C.v2 = p(1)*phi_temp.^3 + ...
    p(2)*phi_temp.^2 + p(3)*phi_temp + p(4);

  phi_offset = interp1(param(ii).dynamic.torque_C.v2,phi_temp,0);

  % Stiffness with new polynom
  param(ii).dynamic.torque_C.v = (p(1).*param(ii).dynamic.phi.v.^3 + ...
```

```
    p(2).*param(ii).dynamic.phi.v.^2 + p(3).*param(ii).dynamic.phi.v ...
    + p(4))-phi_offset;

% Excitation of Testrun
param(ii).dynamic.torque_Exc.v = ...
    tResult.Experiment(param(ii).dynamic.Idx(jj)).Static + ...
    tResult.Experiment(param(ii).dynamic.Idx(jj)).Dynamic * ...
    sin(Omega * t);

% Inertia of Masses
param(ii).dynamic.torque_Inertia.v = J * omega_dot;

phi = phi - phi_offset;
param(ii).dynamic.phi.v = param(ii).dynamic.phi.v - phi_offset;


phi_up    = param(ii).dynamic.phi.v((torque_min_idx(1)+10):...
    torque_max_idx(1));
tq_up     = param(ii).dynamic.torque.v((torque_min_idx(1)+10):...
    torque_max_idx(1));

phi_down = ...
    param(ii).dynamic.phi.v((torque_max_idx(1)+15):torque_min_idx(2));
tq_down   = ...
    param(ii).dynamic.torque.v((torque_max_idx(1)+15):torque_min_idx(2));

% Calculate Damping Component of Hystersis
% J*omega_dot + M_d + M_c = M*sin(Omega t)
% M_d = M*sin(Omega t)(Measurement) - J*omega_dot - M_c
tq_D =  param(ii).dynamic.torque.v(phi_pzc_idx(1):phi_pzc_idx(2)) ...
    - param(ii).dynamic.torque_C.v(phi_pzc_idx(1):phi_pzc_idx(2)) - ...
    param(ii).dynamic.torque_Inertia.v; ...

T_max    = (max(tq)-min(tq))/2;
phi_max = (max(phi)-min(phi))/2;

Ael = T_max*phi_max;
Ad  = trapz(phi,tq);
Psi = Ad/Ael;

%% get damping parameters with least square method
% Coulomb
% omega = omega;  % omega of testrun
% phi = phi;   % phi of testrun
Coulomb_ratio = 0.1;              % percent Coulomb Damping at phi=0

R = tq_D;
d_Coulomb = Coulomb_ratio*abs(tq_D(1));
R2 = d_Coulomb*sign(omega);
```

```
R_Coulomb = R2;

% print hystersis coefficients
hfig = figure;
hold on
grid on
plot(phi(3:end),R(3:end),'k')
plot(phi(3:end),R2(3:end),'b')
formatFigures(sprintf('Coulomb''scher Anteil Messkurve %g',...
  param(ii).dynamic.Idx(jj)),...
  'Verdrehwinkel [rad]','Moment [Nm]');
hold off
legend 'Referenz Hysterese' 'Coulomb''scher Anteil'
fig2pdf('images', sprintf('Coulomb_WN_Messkurve_%g',...
  param(ii).dynamic.Idx(jj)))
close(hfig)

% Reid
R = R - R2;      % Hysteresis
M = zeros(length(omega),1);      % preallocate size of M

for nn = 1:length(omega)
  M(nn) = abs(phi(nn))*sign(omega(nn));
end

P = M\R;  %least square solving

d_Reid=P(1);
R2=M*P;
R_Reid = R2;

% plot Reid
hfig = figure;
hold on
grid on
plot(phi(3:end),R(3:end),'k')
plot(phi(3:end),R2(3:end),'g')
formatFigures(sprintf('Reid''scher Anteil Messkurve %g',...
  param(ii).dynamic.Idx(jj)),...
  'Verdrehwinkel [rad]','Moment [Nm]');
hold off
legend 'Referenz Hysterese' 'Reid''scher Anteil'
fig2pdf('images', sprintf('Reid_WN_Messkurve_%g',...
  param(ii).dynamic.Idx(jj)))
close(hfig)

% visk
R = R-R2;                        % Hysteresis
M = zeros(length(omega),1);      % preallocate size of M
```

```
for nn = 1:length(omega)
  M(nn) = omega(nn);
end

P = M\R;  %least square solving

d_visk=P(1);
R2=M*P;
R_visk = R2;
% print hystersis coefficients
hfig = figure;
hold on
grid on
plot(phi(3:end),R(3:end),'k')
plot(phi(3:end),R2(3:end),'r')
formatFigures(sprintf('Kelvin-Voigt''scher Anteil Messkurve %g',...
  param(3).dynamic.Idx(jj)),...
  'Verdrehwinkel [rad]','Moment [Nm]');
hold off
legend 'Referenz Hysterese' 'Kelvin-Voigt''scher Anteil'
fig2pdf('images', sprintf('KV_WN_Messkurve_%g',...
  param(3).dynamic.Idx(jj)))
close(hfig)

D_calc = R_Coulomb + R_Reid + R_visk;

% print calculated hysteresis approach
hfig = figure;
hold on
grid on
plot(phi(3:end),tq_D(3:end),'k')
plot(phi(3:end),D_calc(3:end),'c')
formatFigures(sprintf('Vergleich Hysterese Messkurve %g',...
  param(3).dynamic.Idx(jj)),...
  'Verdrehwinkel [rad]','Moment [Nm]');
hold off
legend 'Referenz Hysterse' 'Ersatz Hysterese'
fig2pdf('images', sprintf('VergleichHysterese_WN_Messkurve_%g',...
  param(3).dynamic.Idx(jj)))
close(hfig)

%% restructure data
param(ii).dynamic.T_max(jj)     = T_max;
param(ii).dynamic.phi_max(jj)   = phi_max;
param(ii).dynamic.Ael(jj)       = Ael;
param(ii).dynamic.Ad(jj)        = Ad;
param(ii).dynamic.phi.v2        = phi;
param(ii).dynamic.tq.v2         = tq;
```

```
    param(ii).dynamic.tq_D.v2        = tq_D;
    param(ii).dynamic.phi.up(jj).v   = phi_up;
    param(ii).dynamic.tq.up(jj).v      = tq_up;
    param(ii).dynamic.phi.down(jj).v   = phi_down;
    param(ii).dynamic.tq.down(jj).v    = tq_down;
    param(ii).dynamic.tc.v           = param(ii).dynamic.torque_C.v;
    param(ii).dynamic.t_unique.v     = param(ii).dynamic.torque_unique.v;
    param(ii).dynamic.phi_unique.v   = param(ii).dynamic.phi_unique.v;
    param(ii).dynamic.c_polynom(jj,:) = [p(1) p(2) p(3) p(4)];
    param(ii).dynamic.omega          = omega;
    param(ii).dynamic.omega_dot      = omega_dot;
    param(ii).dynamic.phi_offset     = phi_offset;
    param(ii).dynamic.D_visk(jj)     = d_visk;
    param(ii).dynamic.D_Reid(jj)     = d_Reid;
    param(ii).dynamic.D_Coulomb(jj)  = d_Coulomb;
    param(ii).dynamic.Psi(jj)        = Psi;

    clear p

  end
end

up300       = [param(3).dynamic.phi.up(3).v ...
                 param(3).dynamic.tq.up(3).v];
down300     = [param(3).dynamic.phi.down(3).v ...
                 param(3).dynamic.tq.down(3).v];
up450       = [param(3).dynamic.phi.up(6).v ...
                              param(3).dynamic.tq.up(6).v];
down450     = [param(3).dynamic.phi.down(6).v ...
                 param(3).dynamic.tq.down(6).v];
up600       = [param(3).dynamic.phi.up(9).v ...
                 param(3).dynamic.tq.up(9).v];
down600     = [param(3).dynamic.phi.down(9).v ...
                 param(3).dynamic.tq.down(9).v];

save t1000WN10Hz300Nm.mat up300 down300 -V4
save t1000WN10Hz450Nm.mat up450 down450 -V4
save t1000WN10Hz600Nm.mat up600 down600 -V4

Static = [];
Dynamic = [];
Visk = [];
Reid = [];
Coulomb = [];
for pp = 1:length(param(3).dynamic.Idx)
  Visk.Val(pp) = param(3).dynamic.D_visk(pp);
  Visk.Static(pp) = tResult.Experiment(param(3).dynamic.Idx(pp)).Static;
  Visk.Dynamic(pp) = ...
    tResult.Experiment(param(3).dynamic.Idx(pp)).Dynamic;
```

```
    Reid.Val(pp) = param(3).dynamic.D_Reid(pp);
    Reid.Static(pp) = tResult.Experiment(param(3).dynamic.Idx(pp)).Static;
    Reid.Dynamic(pp) = ...
      tResult.Experiment(param(3).dynamic.Idx(pp)).Dynamic;
    Coulomb.Val(pp) = param(3).dynamic.D_Coulomb(pp);
    Coulomb.Static(pp) = tResult.Experiment(param(3).dynamic.Idx(pp)).Static;
    Coulomb.Dynamic(pp) = ...
     tResult.Experiment(param(3).dynamic.Idx(pp)).Dynamic;
end

% plot Kelvin Voigt Parameter
hfig = figure;
hold on
grid on
plot(Visk.Dynamic,Visk.Val,'or','LineWidth',0.5,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor','r',...
    'MarkerSize',10)
formatFigures(sprintf('Kelvin-Voigt Parameter'),...
    'Momentenamplitude [Nm]','Kelvin-Voigt Parameter [-]');
p_KV = polyfit(Visk.Dynamic,Visk.Val,1);
plot(min(Visk.Dynamic):max(Visk.Dynamic),(p_KV(1)*...
    (min(Visk.Dynamic):max(Visk.Dynamic)))+p_KV(2),'-k','Linewidth',2)
hold off
legend(sprintf('berechnete Kelvin-Voigt Parameter'), ...
    sprintf('Regression = %1.4e*M_d_y_n + %1.4e',p_KV(1),p_KV(2)))
xlim([min(Visk.Dynamic)-50 max(Visk.Dynamic)+50])
fig2pdf('images', sprintf('KV_WN_Parameter'))
close(hfig)

% plot Reid Parameter
hfig = figure;
hold on
grid on
plot(Reid.Dynamic,Reid.Val,'og','LineWidth',0.5,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor','g',...
    'MarkerSize',10)
formatFigures(sprintf('Reid Parameter'),...
    'Momentenamplitude [Nm]','Reid Parameter [-]');
p_Reid = polyfit(Reid.Dynamic,Reid.Val,1);
plot(min(Reid.Dynamic):max(Reid.Dynamic),(p_Reid(1)*...
    (min(Reid.Dynamic):max(Reid.Dynamic))+p_Reid(2)),'-k','Linewidth',2)
hold off
legend(sprintf('berechnete Reid Parameter'), ...
    sprintf('Regression = %1.4e*M_d_y_n + %1.4e',p_Reid(1),p_Reid(2)))
xlim([min(Reid.Dynamic)-50 max(Reid.Dynamic)+50])
fig2pdf('images', sprintf('Reid_WN_Parameter'))
close(hfig)
```

```
% plot Coulomb Parameter
hfig = figure;
hold on
grid on
plot(Coulomb.Dynamic,Coulomb.Val,'ob','LineWidth',0.5,...
  'MarkerEdgeColor','k',...
  'MarkerFaceColor','b',...
  'MarkerSize',10)
formatFigures(sprintf('Coulomb Parameter'),...
  'Momentenamplitude [Nm]','Coulomb Parameter [-]');
p_Coulomb = polyfit(Coulomb.Dynamic,Coulomb.Val,1);
plot(min(Coulomb.Dynamic):max(Coulomb.Dynamic), (p_Coulomb(1)*...
  (min(Coulomb.Dynamic):max(Coulomb.Dynamic))+p_Coulomb(2)),...
  '-k','Linewidth',2)
hold off
legend(sprintf('berechnete Coulomb Parameter'), ...
  sprintf('Regression = %1.4e*M_d_y_n + %1.4e',p_Coulomb(1),p_Coulomb(2)))
xlim([min(Coulomb.Dynamic)-50 max(Coulomb.Dynamic)+50])

fig2pdf('images', sprintf('Coulomb_WN_Parameter'))
close(hfig)

fprintf('Finished <%s> successfully.\n',mfilename);
```

Listing A.8: r.m

```
clc;
close all;
global project

%% ---------------------------------------------------------------------
% Project data
workdir = pwd;
mm=0;

project              = [];
project.number       = '99-0000-00032';
project.customer     = 'tectos gmbh';
project.language     = 'de';
project.resultpath   = 'Results';
project.workdir      = pwd;
project.error        = false;

mm=mm+1;
project.models(mm).name          = 'I4T Testbed';
project.models(mm).modelname     = ...
  'I4T_Testbed_t1000_nonlin_Kelvin_Reid_Coulomb_Ramp_WN';
project.models(mm).simno         = 0;
project.models(mm).parametername = ...
  'RealTimeModel_MK20120117111247-076_build-2012-01-17-11-19-11.mat';
project.models(mm).testruns      = {'RunUp100rpm/s WN'};  %

% mm=mm+1;
% project.models(mm).name          = 'I4T Testbed';
% project.models(mm).modelname     = ...
%   'I4T_Testbed_t1000_nonlin_Kelvin_Reid_Coulomb_Ramp_SN';
% project.models(mm).simno         = 0;
% project.models(mm).parametername = ...
%   'RealTimeModel_MK20120117111247-076_build-2012-01-17-11-19-11.mat';
% project.models(mm).testruns      = {'RunUp100rpm/s SN'};  %

%% ---------------------------------------------------------------------
% Define parameters for simulation

project.om           = [];
project.om.filename  = 'tModelicaLibMK.mo';
project.om.modelname = sprintf('tModelicaLibMK.TestBeds.%s',...
  project.models(mm).modelname);   % Set Modelname
project.om.libraries = {'Modelica'};
project.om.debug     = 1;
project.om.starttime = 0;
project.om.stoptime  = 75; %75
project.om.sampletime = 1/8192;
```

```
project.om.PostProcStartTime = 3;
% in s - time when interested event starts - cuts former data
project.om.tolerance  = 1e-4;


if project.tPostProc
  assignOuput;
  s=[];
  for ii = 1:length(project.om.omOutputVar(:,1))
    s = sprintf('%s|%s',s,project.om.omOutputVar{ii,1});
  end
  project.om.variableFilter=s(2:end);
else
  project.om.variableFilter='.*';
end

% Simulate and PostProc - manual or automatic
if ~project.error
  if ~project.tPostProc  % manual run
    %% ------------------------------------------------------------------
    % Create Data for C 3D Interpolation
    tic;
    createCLut(project.models(mm).parametername, 'CombustionPressure');
    createCLut(project.models(mm).parametername, 'CompressionPressure');
    toc;

    %% ------------------------------------------------------------------
    % Compile C function with Open Modelica gcc
    tic;
    %om_path=[getenv('OPENMODELICAHOME') 'bin' filesep];
    [err,msg]=system(sprintf('%s -c -o calcCylPressure.o ...
          calcCylPressure.c',...
      fullfile(getenv('OPENMODELICAHOME'),'MinGW','bin','gcc')));
    toc;
    if err
      msg;
      project.error=true;
    end
    %% ------------------------------------------------------------------
    %Compile and simulate model
    fprintf('Start compiling and simulating model <%s> started at %s\n',...
      project.om.modelname, datestr(now,13));
    tic;
    [err,msg]=omcompilerun(project.om.filename,project.om.modelname,...
      project.om.libraries,project.om.debug,project.om.starttime,...
      project.om.stoptime,project.om.sampletime,...
      project.om.variableFilter, project.om.tolerance);
    fprintf('Finished compiling and simulating model <%s> in %fs\n',...
      project.om.modelname,toc);
```

```
    if err
      msg;
      project.error=true;
    end

    % Postprocessing
    fprintf('Starting postprocessing model <%s>\n',project.om.modelname);
    tic;
    omimportpostproc(project.om.modelname);

    % Plot
    n=1;
    for ii=1:n
      figure
      set(gcf, 'color', 'white');
      plot(project.om.result(ii).time,project.om.result(ii).value,'b')
      grid on
      title(char(project.om.result(ii).name),'FontSize',14);
      xlim([0 project.om.stoptime]);
      xlabel('time [s]');
      ylabel([char(project.om.result(ii).name) ' ']);
    end

    fprintf('Finished postprocessing model <%s> in %fs\n',...
          project.om.modelname,toc);

  else % automatic run - tPostProc
    for mm=1:length(project.models)
      sensor.dt1  = project.om.sampletime;
      sensor.Fs   = 1/sensor.dt1;
      sensor.Fc   = 1000;
      sensor.Wn   = sensor.Fc/(sensor.Fs/2);
      sensor.b    = fir1(20,sensor.Wn);
      sensor.dt2  = 1/8192;
      sensor.tmax = project.om.stoptime;

      tResult=[];
      tResult.Project.ProjectNumber = strrep(project.number,'_','-');
      tResult.Project.ProjectName   = ...
        sprintf('Time domain simulation for engine %s',project.models(mm).name);
      tResult.Project.Customer      = project.customer;
      tResult.Project.Language      = project.language;
      tResult.Project.Experiment    = [];

      for ee=1:length(project.models(mm).testruns)
        tResult.Project.Experiment(ee).ExperimentName = ...
          project.models(mm).testruns{ee};
        tResult.Project.Experiment(ee).Description    = project.om.modelname;
        tResult.Project.Experiment(ee).Type           = 'time';
```

```
tResult.Project.Experiment(ee).Sensor          = [];

%% ------------------------------------------------------------------
% Create Data for C 3D Interpolation
tic;
createCLut(project.models(mm).parametername, ...
      'CombustionPressure');
createCLut(project.models(mm).parametername, ...
          'CompressionPressure');
toc;

%% ------------------------------------------------------------------
% Compile C function with Open Modelica gcc
tic;
[err,msg]=system(...
          sprintf('%s -c -o calcCylPressure.o calcCylPressure.c',...
  fullfile(getenv('OPENMODELICAHOME'),'MinGW','bin','gcc')));
toc;
if err
  msg;
  project.error=true;
end

%% ------------------------------------------------------------------
% Compile and simulate model
fprintf('Start compiling and simulating model <%s> started at %s\n',...
  project.om.modelname, datestr(now,13));
tic;
[err,msg]=omcompilerun(project.om.filename,project.om.modelname,...
  project.om.libraries,project.om.debug,project.om.starttime,...
  project.om.stoptime,project.om.sampletime,...
  project.om.variableFilter,project.om.tolerance);
fprintf('Finished compiling and simulating model <%s> in %fs\n',...
  project.om.modelname,toc);
tic
if err
  msg;
  project.error=true;
end

%% ------------------------------------------------------------------
% PostProc
omimportpostproc(project.om.modelname);
for ss=1:length(project.om.result)
  sensor.label=project.om.result(ss).label;
  sensor.p1=strfind(sensor.label,'_');
  sensor.unit = char(sensor.label(sensor.p1(end)+1:end));
  sensor.name = project.om.result(ss).name;
  switch sensor.unit
```

```
    case 'Nm'
      sensor.quantity = 'Torque';
    case {'rpm', 'radpers', 'radPers'}
      sensor.quantity = 'Speed';
      if  strcmp(sensor.unit,'radpers')
        sensor.unit = 'rad/s';
      end
      if strcmp(sensor.unit,'radPers')
        sensor.unit = 'rad/s';
      end
      if  strcmp(sensor.unit,'rpm')
        sensor.unit = 'rpm';
      end
    otherwise
      sensor.quantity=sensor.name;

  end
  tResult.Project.Experiment(ee).Sensor(ss).Name         = ...
    sensor.name;
  tResult.Project.Experiment(ee).Sensor(ss).SerialNumber = ...
    project.om.result(ss).label;
  tResult.Project.Experiment(ee).Sensor(ss).Direction    = ...
    {'T' 'S'};
  tResult.Project.Experiment(ee).Sensor(ss).Quantity     = ...
    {'Time' sensor.quantity};
  tResult.Project.Experiment(ee).Sensor(ss).Unit         = ...
    {'s' sensor.unit};

  % Resampling data to 8192Hz with filter cut frequency of 1kHz
  sensor.t1 = project.om.result(ss).time';
  sensor.v1 = project.om.result(ss).value';


  sensor.v12=filtfilt(sensor.b,1,sensor.v1);   % Filter to fc=1kHz

  sensor.t2=(0:sensor.dt2:sensor.tmax)';

  % detect doubled entries in sensor.t1 and delete the rows
  u = unique(sensor.t1);
  n = histc(sensor.t1,u);
  isMultiple = find(ismember(sensor.t1, u(n > 1)));

  sensor.t1 = ...
    sensor.t1(not(ismember(1:length(sensor.t1),isMultiple)));
  sensor.v12 = ...
    sensor.v12(not(ismember(1:length(sensor.v12),isMultiple)));

  sensor.v2 = ...
    interp1(sensor.t1,sensor.v12,sensor.t2);  % Resample to 8192Hz
```

```
          first = 1;
          last  = length(sensor.t2);

          sensor.t2 = sensor.t2(not(ismember(1:length(sensor.t2),last)));
          sensor.t2 = sensor.t2(not(ismember(1:length(sensor.t2),first)));

          sensor.v2 = sensor.v2(not(ismember(1:length(sensor.v2),last)));
          sensor.v2 = sensor.v2(not(ismember(1:length(sensor.v2),first)));

          clear u
          clear n
          clear isMultiple
          clear first
          clear last

          sensor.t2 = ...
            sensor.t2(1+project.om.PostProcStartTime*(1/sensor.dt2):end);
          if project.om.PostProcStartTime > 0
            sensor.t2 = sensor.t2 - sensor.t2(1);
          end
          sensor.v2 = ...
            sensor.v2(1+project.om.PostProcStartTime*(1/sensor.dt2):end);

          tResult.Project.Experiment(ee).Sensor(ss).Data = ...
            [sensor.t2 sensor.v2];
        end
      end
      cd(tmodellibroot);
      tResult.Tool.Name    = 'tModeLib';
      tResult.Tool.Version = tmodellibver;
      [~,tResult.Tool.Sha1]= system('git log -1 --pretty=format:%H');
      %[err,msg]=system('git stash pop');
      cd(project.workdir);

      [~,~,~]=mkdir(project.resultpath);
      sensor.fname = ...
        fullfile(project.resultpath,sprintf('%s-SimResult_%s.mat',...
        datestr(now,'yyyymmddHHMM'),project.models(mm).modelname));
      save(sensor.fname,'tResult');
      fprintf('Saved result to <%s.mat>\n',sensor.fname);
      clear sensor
    end

    fprintf('Finished postprocessing model <%s> in %fs\n',...
      project.om.modelname,toc);
  end
end
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] HOFER, A.: *Skriptum zur Vorlesung Regelungstechnik 1*. Technische Universität Graz, 2007.

[2] BARTSCH, HANS-JOCHEN: *Taschenbuch Mathematischer Formeln*. Carl Hanser Verlag, 2007.

[3] K.ELLERMANN: *Skriptum zur Vorlesung Nichtlineare Schwingungen*. TU Graz, 2012.

[4] STÜHLER, W.: *Dämpfung - Entstehung, Beschreibungsformen, Auswirkungen und Abhängigkeiten*. VDI Berichte 1082.

[5] DRESIG, HANS: *Schwingungen und mechanische Antriebssysteme*. Springer, 2006.

[6] TILLER, MICHAEL M.: *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers, 2001.

[7] A.WIMMER: *Thermodynamik des Verbrennungskraftmotors*. VKM THD TU Graz, 2004.

[8] PIERSOL, A. und T. PAEZ: *Harris' Shock and Vibration Handbook*. McGraw-Hill Handbooks. McGraw-Hill Companies,Incorporated, 2009.

[9] S. MÜLLER, L.CZERNY: *Simulation des Drehschwingungsverhaltens verbrennungsmotorischer Antriebsanlagen bei instationären Betriebszuständen*. VDI Berichte Nr.1220, 1998.

[10] H.DRESIG, J.I. VULFSON: *Zur Dämpfungstheorie bei nichtharmonischer Belastung*. VDI Berichte.

[11] HÖFLER, DIETER: *Torsionsschwingungen von Motorenprüfständen*. Diplomarbeit, Technische Universität Graz, July 2001.

[12] RIVIN, EUGENE I.: *Stiffness and Damiping in Mechanical Design*. Marcel Decker AG, 1999.

[13] EICHELSEDER, H.: *Kolbenmaschienen*. TU Graz, 2000.

[14] EICHELSEDER, H.: *Verbrennungskraftmaschinen Vertiefte Ausbildung*. TU Graz, 2005.

[15] MAYR, WOLFGANG: *Simulationstechnische Analyse von Elastomerklauenkupplungen basierend auf der tectos t1000-Serie*, 2011.

[16] MAYRHOFER, JOSEF: *Modeling and Testing of Analog and Mixed-Signal Devices Based on Least Squares Techniques*. Diplomarbeit, Technische Universität Graz, 1991.

[17] SCHAAD, CHRISTIAN: *OpenModelica-MATLAB Interface*, August 2012.

[18] OPPENHEIM, A.V. und R.W. SCHAFER: *Discrete-time signal processing*. Prentice-Hall signal processing series. Prentice Hall, 1989.

[19] PÁLFI, L., T. GODA und K. VÁRADI: *Theoretical prediction of hysteretic rubber friction in ball on plate configuration by finite element method*. eXPRESS Polymer Letters, 3(11):713–723, 2009.

[20] ÖZTÜRK, KEMAL: *Zur nichtlineare Simulation der Wärmeentwicklung in Elastomer-Kupplungen infolge schwingender Beanspruchung*. Doktorarbeit, Technische Universität Berlin, April 2006.

[21] FRITZSON, PETER: *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. WILEY-IEEE PRESS, 2003.

[22] *Modelica Specification Version 3.1*. www.modelica.org.

[23] TOBIAS HIRSCH, MARKUS ECK: *4-Dimension Table Interpolation with Modelica*. 2008.