Modellierung und Regelung eines Verbrennungsmotors in Dymola/Modelica

Masterarbeit

an der

Technischen Universität Graz

vorgelegt von

Arnold Loidl

Institut für Regelungs- und Automatisierungstechnik,
Technische Universität Graz
A-8010 Graz

31. Juli 2012

Modeling and control of a combustion engine in Dymola/Modelica

Master's Thesis

at

Graz University of Technology

submitted by

Arnold Loidl

Institute of Automation and Control,
Graz University of Technology
A-8010 Graz, Austria

31th July 2012

Kurzfassung

Der erste Teil dieser Arbeit beschäftigt sich mit dem Modellieren des Luftpfades eines Dieselmotors. Dabei wird die Modellierungssoftware Dymola mit der dazugehörigen Beschreibungssprache Modelica verwendet. Der verwendete Verbrennungsmotor ist mit seinem großen Hubraum von 10l für die Anwendung in Lastkraftwagen vorgesehen. Außerdem besitzt er eine Abgasturboaufladung zur Leistungssteigerung und eine Abgasrückführung zur besseren Kontrolle der Emissionswerte. Der Turbo hat eine Variable Turbinengeometrie (VGT) als Stellmöglichkeit. Die Modellierung des Verbrennungsvorgangs erfolgt durch ein Mittelwertmodell. Dies bedeutet, dass der Verbrennungsvorgang aufgrund der Eingangsparameter gemittelt, und nicht anhand des Kurbelwellenwinkels detailliert modelliert wird. Das Modell wird mit den bereits vorhandenen Messdaten verifiziert.
Zur Regelung werden in verschiedenen Arbeitspunkten linearisierte Modelle erstellt. Diese sind die Grundlage für die modellbasierte prädiktive Regelung (MPR), welche in Simulink realisiert ist. Um das Modell weiterhin in Dymola simulieren zu können, wird die Co-Simulationsumgebung Independent Co-Simulation (ICOS), welche am Virtuellen Fahrzeug entwickelt wurde, verwendet. Zur realitätsnahen Simulation ist es notwendig, Teile des Prüfstandes in Simulink nachzubilden. Abschließend wird das Verhalten der Regelung anhand verschiedener Arbeitspunktwechsel verifiziert.

Abstract

The first part of this work deals with the modeling of the air path of a diesel engine. For this purpose, the software Dymola with the accompanying description language Modelica is used. The investigated combustion engine with its large displacement of 10l is provided for an application in trucks . It also has an exhaust gas turbocharging equipment and exhaust gas recirculation for improved control of emissions. The turbo has a variable turbine geometry(VGT) as a control option.
The modeling of the combustion process is carried out by a mean value model. This means that the combustion is averaged based on the input parameters, and not modeled in detail by the crankshaft angle. The model is verified by comparison with the existing measured data.
To control the entire air path in different operating points linearized models are created. These models are the basis for the model predictive controller, which is implemented in Simulink. In order to further simulate the model in Dymola, the co-simulation environment Independent Co-Simulation (ICOS) is used. For realistic simulations, it is necessary to simulate the engine test bed in Simulink. Finally, the behavior of the system is verified by simulations of different operating point changes.

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.

Place	Date	Signature

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.

Ort	Datum	Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagung

Der Autor bedankt sich bei Herrn Ao.Univ.-Prof. Dr.techn. Anton Hofer vom Institut für Regelungs- und Automatisierungstechnik (IRT) für fachliche Unterstützung bei der Durchführung dieser Arbeit. Ebenfalls danke ich der unterstützenden Firma „Firma AVL“ sowie der Technischen Universität Graz. Insbesondere möchte ich den Mitarbeitern des Virtuellen Fahrzeugs, Herrn Dipl. Ing. Martin Benedikt, Dipl. Ing. Michael Stolz und Dr. Daniel Watzenig danken, da sie eine große Hilfe bei der Lösung von Problemen waren.
Der Autor dankt dem „COMET K2 Forschungsförderungs-Programm“ des Österreichischen Bundesministeriums für Verkehr, Innovation und Technologie (BMVIT), des Österreichischen Bundesministeriums für Wirtschaft, Familie und Jugend (BMWFJ), der Österreichischen Forschungsförderungsgesellschaft mbH (FFG), des Landes Steiermark sowie der Steirischen Wirtschaftsförderung (SFG) für die finanzielle Unterstützung.

Arnold Loidl
Graz, Österreich, Juli 2012

Kapitel 1

Einleitung

Auch in der Automobilindustrie werden die Produktzyklen immer kürzer, wodurch es von großer Bedeutung ist, jede Modifikation am Motor direkt simulieren zu können. Dies setzt jedoch voraus, dass mathematische Modelle der Komponenten in adäquaten Genauigkeiten vorhanden sind. Dabei stellt die Wahl der Abstraktion ein interessantes Problem dar. Einerseits ist es einsichtig, dass eine zu wenig detaillierte Beschreibung des Systems Nachteile mit sich bringt, andererseits kann die Beschreibung von unnötigen Effekten die Komplexität und damit den benötigten Rechenaufwand stark erhöhen.

1.1. Aufgabenstellung

Als Ergebnis dieser Arbeit soll ein Modell des Luftpfades eines Dieselmotors zur Verfügung stehen. Dabei soll ein vom Virtuelles Fahrzeug (ViF) erstelltes Matlab/Simulink Modell als Basis herangezogen werden. Da es in der bisherigen Entwicklungsumgebung zu Problem bei der Initialisierung und zu algebraischen Schleifen gekommen ist, soll in dieser Arbeit die Modellierung in Dymola/Modelica erfolgen. Dadurch erwartet man sich eine deutliche Verbesserung hinsichtlich der benötigten Rechenzeit. Das Modell soll als Bibliothek für die Simulation in anderen Projekten verwendet werden können.
Anschließend ist eine Regelung zu erstellen, welche einem Motorprüfstand nachempfunden sein wird. Dabei ist der Zustand des gesamten Motors zu berücksichtigen, um eine Regelung für den Luftpfad zu entwerfen. Das Hauptaugenmerk ist auf eine modellbasierte Regelung zu legen, da nur bei dieser das gesamte Modellwissen einfließen kann. Diese Regelungsstrategie soll durch unterschiedliche Solltrajektorien für die Regelgrößen verifiziert werden.

1.2. Prinzip eines Dieselmotors

Ein Dieselmotor ist ein selbstzündender Verbrennungsmotor, bei dem das Luft-Kraftstoffgemisch so weit komprimiert wird, bis es sich entzündet. Die darauf folgende exotherme Reaktion wird über den Kolben in mechanische Energie umgewandelt. Obwohl der erste Dieselmotor bereits Ende des 18. Jahrhunderts existierte, kann seine Effizienz durch neue Innovationen immer noch gesteigert werden. Vor allem die Möglichkeit zur Leistungssteigerung durch eine Turboaufladung und die elektronische Regelung des Motors haben in den letzten Jahrzehnten zu

seiner Beliebtheit beigetragen. Die momentanen Entwicklungen konzentrieren sich vor allem auf die Reduktion der Schadstoffemissionen.

1.3. Abgasverhalten

Bei der Verbrennung in einem Dieselmotor treten unterschiedliche Schadstoffe in Erscheinung. Die beiden wesentlichsten, welche auch einer gesetzlichen Limitierung unterliegen, sind Partikel und Stickoxide. Um eine Reduktion dieser Emissionen zu erreichen, werden große Anstrengungen unternommen. Durch eine verbesserte Regelung des Luftpfades kann die Entstehung der Emissionen reduziert werden. Eine zusätzliche Abgasnachbehandlung reduziert den Ausstoß nochmals deutlich. In Abbildung 1.1 sind die verschiedenen gesetzlichen Emissionsgrenzwerte dargestellt, welche von der Europäischen Union festgelegt wurden. Dabei gilt seit Oktober 2008 die Euro V Norm, wie in Abbildung 1.1 zu sehen ist, für Nutzfahrzeuge in der Europäischen Union. Diese Werte müssen unter möglichst geringen zusätzlichem Aufwand eingehalten werden. Zu Beginn des Jahres 2013 tritt nochmals eine drastische Verschärfung der Emissionsgrenzwerte durch das Inkrafttreten der *Euro VI* Abgasnorm ein. Ab diesem Zeitpunkt ist der Ausstoß von Rußpartikeln auf $0.01 g/kWh$ und von Stickoxiden auf $0.4 g/kWh$ gesetzlich begrenzt.

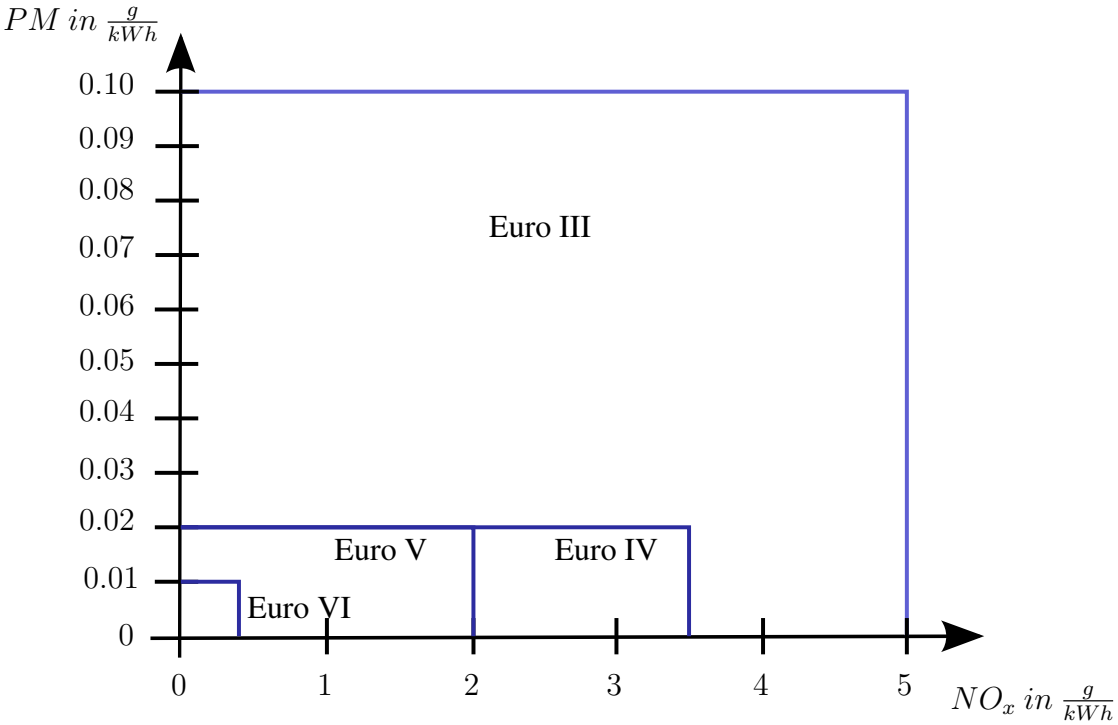

Abbildung 1.1.: Vorgeschriebene Grenzwerte in der EU für Nutzfahrzeuge, [DieselNet, 2012] und [Becher und Eggert, 2004]

Kapitel 2

Modellierungsumgebung Dymola/Modelica

Modelica ist eine offene Programmiersprache, die seit 1996 von einer europäischen Forschergruppe (Modelica Association) entwickelt wird, und von unterschiedlichen Softwarepaketen übersetzt werden kann,Fritzson [2003]. Im Rahmen dieser Arbeit wird die proprietäre Software *Dymola* für die Implementierung und Simulation verwendet.

2.1. Modellierungssprache Modelica

Das wichtigste Merkmal der Programmiersprache *Modelica* ist deren Objektorientiertheit. Dadurch hat man die Möglichkeit, komplexe physikalische Modelle sinnvoll in Klassen aufzuteilen und einzelne Komponenten wiederverwenden zu können. Außerdem ist es einfach möglich, physikalische Systeme aus unterschiedlichen Domänen, wie Elektrik, Mechanik, Thermodynamik, usw., in einem Gesamtmodell zu simulieren.

2.1.1. Aufbau eines Modells

Jedes Modell wird in *Modelica* durch eine eigene Klasse repräsentiert. Erst durch die Verwendung des Modells, entweder als Teil eines anderen oder in einer Testumgebung, wird ein konkretes Objekt der Klasse instanziiert. Der Aufbau eines einfachen Modells ist exemplarisch in Listing 2.1 dargestellt. In diesem Beispiel ist die Bewegung einer rotierenden Masse angeführt. Dazu muss die Differentialgleichung 2.1 für die Drehzahl $N(t)$ implementiert werden:

$$\frac{dN(t)}{dt} = \frac{M(t)}{2 \cdot \pi \cdot J} \; mit \; N(0) = 100 \tag{2.1}$$

In diesem Beispiel wird in Zeile 3 das für die Modellierung erforderliche Drehmoment $M(t)$ definiert und in Zeile 7 das Massenträgheitsmoment J, welches ein konstanter Parameter ist. Die Implementierung der Differentialgleichung erfolgt ausschließlich in Zeile 15. Der andere Teil des Modells stellt die Definition der benötigten Größen und die Initialisierung der Drehzahl dar. Zu beachten ist, dass alle Zuweisungen, die im Abschnitt *initial algorithm* sind, ausschließlich vor Simulationsbeginn zur konsistenten Initialisierung des gesamten Systems herangezogen werden.

```
1  model Rotationsmasse
2      // Definition der Ein- und Ausgänge
3      Modelica.Blocks.Interfaces.RealSignal M;
4      Modelica.Blocks.Interfaces.RealSignal N;
5
6      //Definition bzw. importieren der benötigten Konstanten
7      parameter Modelica.SIunits.MomentOfInertia J = 1;
8      import Modelica.Constants.pi;
9
10   algorithm
11       // Implementation von Zuweisungssequenzen
12
13   equation
14       // Abbildung der physikalischen Beziehungen zwischen den
             Größen
15       der(N) = M/(2*pi*J)
16
17   initial algorithm
18       // Initialisierung der speichernden Elemente
19       N = 100;
20 end Rotationsmasse;
```

Listing 2.1: Beispiel eines einfachen Modells in Modelica

2.1.2. Datenstruktur Record

In Modelica beschreiben sogenannte *records* Klassen, welche nur Daten enthalten. Durch dieses Konzept besitzt der Anwender die Möglichkeit, die Objektorientiertheit durch das gesamte Modell beizubehalten. Es handelt sich dabei um eine Klasse, in der weder Gleichungen noch Algorithmen vorkommen. Diese kann, wie sämtliche anderen Klassen auch, initialisiert und mit den zum Modell gehörigen Daten ausgestattet werden. Besonders eignet sich diese Komponente zur Speicherung von modellspezifischen Daten.

2.1.3. Verbindung von Modellen

Die Kommunikation zwischen zwei Objekten erfolgt über Anschlüsse (engl. *connectoren*), welche einfach durch Linien symbolisiert werden. In Abbildung 2.1 sind schematisch drei Modelle dargestellt, die miteinander verbunden werden. Um zwei Anschlüsse durch eine Verbindung (engl. *connection*) zu verbinden müssen beide vom selben Typ sein. Dies ist vor allem deshalb notwendig, da in *Modelica* mehrere physikalische Größen durch eine Verbindung verbunden werden können. Ein elektrischer Anschluss repräsentiert zum Beispiel die Spannung $u(t)$ und den Strom $i(t)$.

Um das gesamte Modell in einer anschaulichen Weise gestalten zu können, ist es oft erforderlich, einen passenden *Connector* zu erstellen. Dies geschieht in Listing 2.2 am Beispiel eines elektrischen Anschlusses.

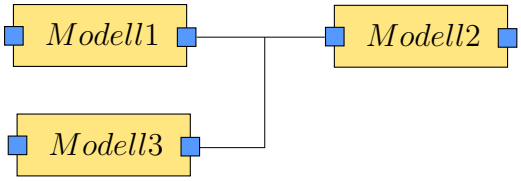

Abbildung 2.1.: Verbindung mehrerer Modelle in Modelica

```
1   connector Elektrisch
2
3     Modelica.SIunits.Voltage v;
4     Modelica.SIunits.Current i;
5
6   end Elektrisch;
```

Listing 2.2: Beispiel einer elektrischen Verbindung

2.1.4. Unterschied zwischen Algorithmen und Gleichungen

Ein wesentlicher Unterschied zu anderen Programmiersprachen ergibt sich durch die Möglichkeit, eine gleichungsbasierte Eingabe vornehmen zu können. Dies hat zur Folge, dass durch die Beschreibung der Beziehung der physikalischen Größen im *equation* Abschnitt keine Datenflussrichtung vorgegeben ist. Auch die Reihenfolge der Beziehungen ist irrelevant für das Verhalten des Modells. Für den Programmierer setzt dies im Gegensatz zu herkömmlichen Sprachen eine veränderte Denkweise voraus. [Fritzson, 2003, S.237]
Diese Vorgehensweise wird in Dymola durch eine symbolische Manipulation (engl. „*symbolic manipulation*“) durchgeführt. Dabei werden im ersten Schritt die implementierten Gleichungen bezüglich ihrer Abhängigkeiten analysiert. Danach werden die unbekannten Größen identifiziert und gegebenenfalls Eliminierungen durchgeführt, ohne dabei die modellierte Funktionalität zu verändern. Das Ergebnis der symbolischen Manipulation sind Gleichungen, die einerseits in einer geeigneten Reihenfolge zur Berechnung angeordnet sind, andererseits bereits auf die zu berechnende Größe umgeformt wurden. Zusätzlich kann durch eine geringere Anzahl von Gleichungen der Berechnungsaufwand reduziert werden. [Claytex]
Da es in vielen Fällen trotzdem notwendig ist, eine Reihe von Befehlen in zeitlicher Reihenfolge abzuarbeiten, wird auch diese Möglichkeit durch den Abschnitt *algorithm* zur Verfügung gestellt. Dieser Abschnitt kann in einem Modell auch in eine *function* ausgegliedert werden, um die Übersichtlichkeit zu erhöhen. Dadurch ist es z.B. elegant möglich, Werte aus modellierten Funktionen abzufragen. Die Definition einer solchen Funktion ist in Listing 2.3 dargestellt.

```
1   function kennflaeche
2         // Definition der Ein- und Ausgänge
3         input Real in1;
4         input Real in2;
5         output Real out;
6
7     algorithm
8         // Implementation von Zuweisungssequenzen
9         out = in1 + in1^2 + in2;
```

```
10    end kennflaeche;
```

Listing 2.3: Definition einer Funktion

2.2. Simulationswerkzeug Dymola

Die Software Dymola wird in dieser Arbeit verwendet, um das in Modelica erstellte Modell des Luftpfades zu übersetzen und zu simulieren. Dabei handelt es sich um eine ausgereifte Software von der Firma *Dynasim* ©. Als Löser wird ein in Dymola implementierter differential-algebraischer Gleichungslöser (engl. „*differential algebraic system solver*“, DASSL) verwendet, da dieser positive Eigenschaften bezüglich der Rechengeschwindigkeit bei komplexen hybriden Systemen aufweist [Felgner]. DASSL verwendet eine Rückwärtsdifferentiation, sogenannte „*Backward Differentiation Formulas*“(BDFs) Verfahren bis zur 5. Ordnung.
Diese Mehrschrittverfahren sind besonders bei steifen System beliebt, da sie bessere Stabilitätseigenschaften aufweisen.

2.2.1. Handhabung von Dymola

Neben der zuvor beschriebenen Texteingabe bietet Dymola die Möglichkeit, im sogenannten Diagramm Abschnitt die Funktionalität durch Verbindung von Teilkomponenten herzustellen. Dies ist in Abb. 2.2 dargestellt. Die Schnittstellen zur Kommunikation mit anderen Komponenten symbolisieren die beiden Dreiecke. In diesem Beispiel wird über die Schnittstelle *Bremsen* ein Bremsmoment vorgegeben, mit dem die Masse *inertia* abgebremst werden kann. Über den *speedSensor* kann die Geschwindigkeit ermittelt und über den Port *v* ausgegeben werden. Nach

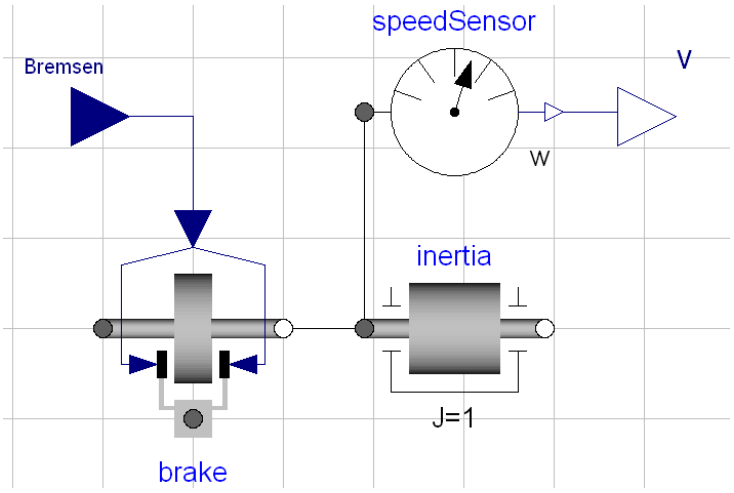

Abbildung 2.2.: Dymola in der Diagramm Ansicht

der Implementierung der Funktionalität ist es bequem möglich, über den Simulationsknopf zur Simulationsansicht zu wechseln. Bevor nun die Simulation gestartet werden kann, sollte man die Werte in den Simulationseinstellungen überlegt einstellen. Dies umfasst, wie in Abb. 2.3 dargestellt die Startzeit (engl. *starttime*), Endzeit (engl. *stoptime*) und Anzahl der Berechnungsintervalle (engl. *number of intervals*), um die Berechnungspunkte festzulegen. Weiters kann man den gewünschten Algorithmus und die verwendete Toleranz auswählen.

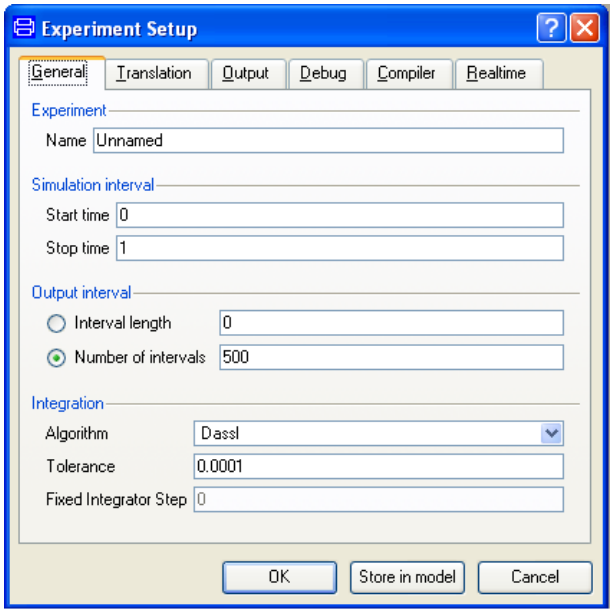

Abbildung 2.3.: Einstellungen für die Simulation in Dymola

Kapitel 3

Modellierung des Luftpfades eines Dieselmotors

Um die wichtige Entscheidung der Modellierungstiefe treffen zu können, muss zunächst die Verwendung des Modells klar abgegrenzt werden. Anhand des in Dymola erstellten Modells soll es möglich sein, verschiedene Regelungsstrategien testen zu können. Das Modell des Luftpfades dient für die Simulation der Regelstrecke beim Reglerentwurf. Grundsätzlich existieren zwei unterschiedliche Ansätze zur Modellierung des Luftpfades eines Verbrennungsmotors, welche sich hinsichtlich des Detaillierungsgrades unterscheiden.

- *Discrete-Event Model* (DEM)
  Bei dieser Art der Modellierung wird die exotherme Reaktion in jedem Zylinder als diskretes Ereignis betrachtet. Dies bedeutet, dass es notwendig ist, die Position der Zylinder genau zu wissen. Deshalb wird der Zustand des Modells über den Kurbelwellenwinkel dargestellt. Dadurch besteht die Möglichkeit, detaillierte Effekte, die bei der Verbrennung auftreten, zu berücksichtigen. Diesem Vorteil steht der deutlich höhere Rechenaufwand gegenüber. [Guzzella und Onder, 2004, Seite 129]
- *Mean-Value Model* (MVM)
  Dem gegenüber steht die Modellierung durch eine Mittelwertbildung des Verbrennungsvorganges. Dabei wird auf die detaillierten Phänomene, die bei der Zündung des Kraftstoff-Luftgemisches auftreten verzichtet. Für die beiden relevanten Größen, dem erzeugten Motordrehmoment und der Brenntemperatur werden Kennlinien erstellt, welche für die Modellierung herangezogen werden.

Das in dieser Arbeit entworfene Modell des Luftpfades dient zur Verifikation von unterschiedlichen Regelungsstrategien. Die einzigen Eingriffsmöglichkeiten für die Regler bestehen über die Klappenstellungen im Luftpfad, worauf das zu erstellende Modell jedoch relativ träge reagiert. Deshalb ist es im Rahmen dieser Arbeit ausreichend sich auf das *Mean-Value Model* zu beschränken.

3.1. Überblick über den gesamten Luftpfad

Bevor in die detaillierte Modellierung näher eingegangen wird, ist in Abbildung 3.1 ein Luftpfad eines Dieselmotors abgebildet. Dieser enthält die wesentlichen Komponenten, wie eine Abgasrückführung mit der dazugehörigen Klappenposition $x_{egr}(t)$ und eine Abgasturboaufladung

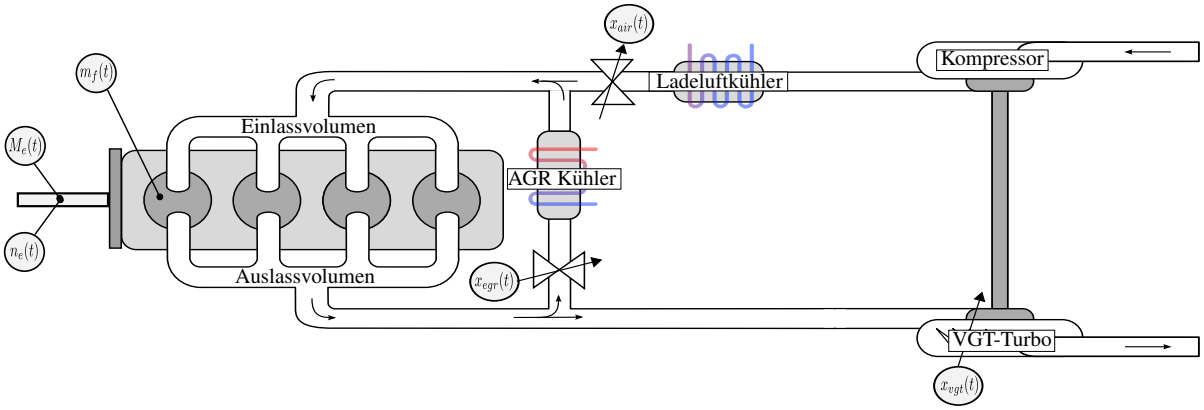

Abbildung 3.1.: Luftpfad eines modernen Dieselmotors

mit der Turbinenstellung $x_{vgt}(t)$ als Eingangsgröße in das System. Weiters ist eine Drosselklappe bei der Luftzufuhr verbaut, welche mit dem Signal $x_{air}(t)$ angesteuert wird. Die Drehzahl des Motors wird mit $n_e(t)$ und die eingespritzte Kraftstoffmenge mit $m_f(t)$ dargestellt. Das erzeugte Motordrehmoment $M_e(t)$ liegt schließlich an der Welle an.
Über die bereits erwähnten Größen Motordrehzahl $n_e(t)$ und Kraftstoffmenge $m_f(t)$ wird dem Luftpfad der aktuelle Zustand des gesamten Verbrennungsmotors aufgeprägt. Der Abgasturbolader ist ein beliebtes Mittel zur Leistungssteigerung eines Verbrennungsmotors. Um die Steigerung der Leistung beeinflussen zu können wurde die Stellung der Turbinenschaufeln variabel ausgeführt. Aufgrund der bereits beschriebenen gesetzlichen Emissionsbeschränkungen wird in modernen Dieselmotoren eine Abgasrückführung verbaut. Eine hohe Abgasrückführrate führt zu geringeren Stickoxidemissionen bei der Verbrennung. Über das Volumen am Eingang gelangt das Kraftstoff-Luftgemisches zu den Zylindern, in denen die Verbrennung abläuft. Anschließend wird das heiße Abgas über das Auslassvolumen zur Turboaufladnung und über die Abgasrückführung wieder zum Einlassvolumen geleitet.

3.2. Darstellung von Volumina

Um die Dynamik des Systems ausreichend erfassen zu können, ist es notwendig, die markanten Volumina des Luftpfades als solche zu modellieren. Wiederum stellt sich die Frage der Modellierungstiefe. Um den Aufwand in Grenzen zu halten, ohne dabei eine zu große Ungenauigkeit in Kauf zu nehmen, werden folgende Vereinfachungen getroffen [Guzzella und Onder, 2004]:

- Homogene Zusammensetzung des Stoffes im Volumen
- Homogene Verteilung der Größen Temperatur und Druck
- Konstante Volumengröße
- Keine sonstigen Energieverluste

Die maßgebliche Beziehung stellt die *thermische Zustandsgleichung idealer Gase* dar [Baehr und Kabelac, 2006, S.208]:

$$p(t) \cdot V = m(t) \cdot R \cdot T(t) \tag{3.1}$$

Sie verbindet, die für das Volumen V bestimmenden Größen, Druck $p(t)$, Masse $m(t)$, Temperatur $T(t)$ und die spezifische Gaskonstante R. An dieser Stelle wird eine neue Zustandsgröße

der Thermodynamik, die Enthalpie $H(t)$ in J, eingeführt:

$$H(t) = U + p(t) \cdot V \tag{3.2}$$

Diese Enthalpie setzt sich aus der Summe von innerer Energie U und dem Produkt von Druck $p(t)$ und Volumen V zusammen und kennzeichnet dadurch den Energiegehalt eines Stoffstromes. Die innere Energie setzt sich wiederum aus der termischen-,chemischen- und potentiellen Energie eines Stoffes zusammen. Um eine Energiebilanz über ein Volumen mit einer Eintritts- und Austrittsöffnung erstellen zu können ist es des Weiteren notwendig, die Situation an dessen Öffnungen zu beschreiben. Dies geschieht mit dem Enthalpiestrom:

$$\begin{aligned} \dot{H}_{zu}(t) &= c_p \cdot T_{zu}(t) \cdot W_{zu}(t) \\ \dot{H}_{ab}(t) &= c_p \cdot T(t) \cdot W_{ab}(t) \end{aligned} \tag{3.3}$$

$W_{zu}(t)$ und $W_{ab}(t)$ repräsentieren die Massenzu- und Abflüsse und c_p den spezifischen Wärmekoeffizienten bei konstantem Druck. Dabei wird von einer guten Durchmischung des Mediums in dem Volumen ausgegangen, wodurch für den austretenden Enthalpiestrom $\dot{H}_{ab}(t)$ die Temperatur $T(t)$ im Volumen verwendet werden kann. Weiters lässt sich, die im Volumen enthaltene Masse $m(t)$ und die Änderung der im Gas enthaltenen Energie $E(t)$ über die Zu- und Abflüsse darstellen:

$$\begin{aligned} \dot{m}(t) &= W_{zu}(t) - W_{ab}(t) \\ \dot{E}(t) &= \dot{H}_{zu}(t) - \dot{H}_{ab}(t) \end{aligned} \tag{3.4}$$

Da von einer adiabatischen Zustandsänderung ausgegangen wird, das heißt es treten keine Wärmetransmissionen durch die Wand des Volumens auf, ist es für die Energiebilanz ausreichend, die beiden Enthalpieflüsse $\dot{H}_{zu}(t)$ und $\dot{H}_{ab}(t)$ durch die Rohre zu verwenden. Ersetzt man die Größen in Glg. 3.4 durch die Beziehungen 3.3 aus der Wärmelehre, resultiert daraus die für diese Arbeit hinreichend genaue Beschreibung eines im Volumen enthaltenen Gases:

$$\begin{aligned} \dot{p}(t) = \frac{dp(t)}{dt} &= \frac{\gamma \cdot R}{V} \left(W_{zu}(t) \cdot T_{zu}(t) - W_{ab}(t) \cdot T(t) \right) \\ \dot{m}(t) = \frac{dm(t)}{dt} &= W_{zu}(t) - W_{ab}(t) \\ T(t) &= \frac{p(t) \cdot V}{R \cdot m(t)} \end{aligned} \tag{3.5}$$

Das Modell enthält als Zustandsgrößen den Druck $p(t)$ und die Masse $m(t)$. Die Temperatur lässt sich aus der algebraischen Glg. 3.1 ermitteln. Der Parameter γ symbolisiert das spezifische Wärmeverhältnis.

3.3. Klappen als Rohrrestriktionen

Unter Rohrrestriktion versteht man einen Widerstand eines durchströmten Rohres, der zu einem Druckabfall führt. Dieser Widerstand kann sich durch die Bauform des Rohres ergeben, oder wie im Fall dieser Arbeit, gezielt als Ventil eingebaut werden, siehe [Jankovic und Kolmanivski, 2000, S.288] und [Heywood, 1988, S.226]. Durch die Ventilstellung $x(t)$ ist die effektive Fläche der Rohrrestriktion $A_{eff}(t)$ eine zeitvariable Größe. Da die Rohre von Luft durchströmt werden,

ist es sinnvoll, bei der Betrachtung des Massenflusses $\dot{m}(t)$ von einem kompressiblen Medium auszugehen, um die Genauigkeit zu erhöhen. Dieser Massenfluss hängt von den thermodynamischen Eigenschaften des einströmenden Gases $p_{zu}(t)$ und $T_{zu}(t)$, sowie dem Gegendruck $p_{ab}(t)$ ab:

$$\dot{m}(t) = A_{eff}(t) \cdot \frac{p_{zu}}{\sqrt{R \cdot T_{zu}(t)}} \cdot \Psi\left(\frac{p_{zu}(t)}{p_{ab}(t)}\right) \tag{3.6}$$

Die Definition der Funktion $\Psi(\frac{p_{zu}(t)}{p_{ab}(t)})$ erfolgt in Gleichung 3.8. In einem stationären Betriebspunkt tritt es auf keinen Fall ein, dass sich die Richtung des Massenflusses ändert. In jedem Rohrabschnitt ist die Richtung des Fluids im stationären Fall vorgegeben. Bei raschen Betriebspunktwechsel, also im transienten Fall, ist es jedoch sehr wohl möglich, dass sich aufgrund geänderter Druckverhältnisse auf beiden Seiten der Rohrrestrition die Richtung des Massenflusses kurzfristig ändert. Deshalb ist es notwendig, die in Gleichung 3.7 dargestellte Fallunterscheidung vorzunehmen. Eingangs- und Ausgangsgrößen werden vertauscht, und zusätzlich das Vorzeichen des Massenflusses geändert:

$$\dot{m}(t) = \begin{cases} A_{eff}(t) \cdot \dfrac{p_{ab}}{\sqrt{R \cdot T_{ab}(t)}} \cdot \Psi\left(\dfrac{p_{ab}(t)}{p_{zu}(t)}\right) & falls\ p_{zu}(t) > p_{ab}(t) \\ 0 & falls\ p_{zu}(t) = p_{ab}(t) \\ -A_{eff}(t) \cdot \dfrac{p_{zu}}{\sqrt{R \cdot T_{zu}(t)}} \cdot \Psi\left(\dfrac{p_{zu}(t)}{p_{ab}(t)}\right) & falls\ p_{zu}(t) < p_{ab}(t) \end{cases} \tag{3.7}$$

In der Korrekturfunktion für den Massenfluss $\Psi(\frac{p_{zu}(t)}{p_{ab}(t)})$ tritt ebenfalls eine Fallunterscheidung auf:

$$\Psi\left(\frac{p_{zu}(t)}{p_{ab}(t)}\right) = \begin{cases} \sqrt{\gamma \cdot \left(\dfrac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} & falls\ \dfrac{p_{zu}(t)}{p_{ab}(t)} \leq \left(\dfrac{2}{\gamma+1}^{\frac{\gamma}{\gamma-1}}\right) \\ \sqrt{\dfrac{2 \cdot \gamma}{\gamma-1}\left(\left(\dfrac{p_{zu}}{p_{ab}}\right)^{\frac{2}{\gamma}} - \left(\dfrac{p_{zu}}{p_{ab}}\right)^{\frac{\gamma+1}{\gamma}}\right)} & sonst \end{cases} \tag{3.8}$$

Diese ist notwendig, da Gase bei Erreichen der Schallgeschwindigkeit ein anderes Verhalten aufweisen. An der engsten Stelle der Drosselklappe wird die Schallgeschwindigkeit erreicht, sobald das anliegende Druckverhältnis einen kritischen Wert überschreitet, [Guzzella und Onder, 2004].

3.4. Modellierung des Verbrennungsvorgangs

Wie bereits erwähnt erfolgt die Modellierung im Rahmen dieser Arbeit durch das sogenannte Mittelwertmodell. Deshalb wird der Verbrennungsvorgang nicht explizit modelliert. Für die beiden Größen Verbrennungstemperatur $T_e(t)$ und erzeugtes Drehmoment $M_e(t)$ werden Kennlinien hinterlegt. Entscheidend für die Modellierung des Luftpfades ist der Massenfluss, welcher durch den Motorblock gepumpt wird. Dieser lässt sich einfach wie in Gleichung 3.9 darstellen, indem die Masse in den Zylindern $m_{zyl}(t)$ mit der Anzahl der Pumpzyklen multipliziert wird. Diese ergeben sich aus der Motordrehzahl $N_e(t)$ und den Umdrehungen pro Zyklus N_{re}. Die Verluste des Pumpprozesses werden durch die volumetrische Effizienz η_{vol} repräsentiert.

$$W_{zu,ab} = \eta_{vol} \cdot m_{zyl}(t) \cdot \frac{N_e(t)}{60 \cdot N_{re}} \tag{3.9}$$

Wird die Masse durch die *thermische Zustandsgleichung idealer Gase* aus Gleichung 3.1 ersetzt, erhält man Glg. 3.10. In dieser stellt die Anzahl der Zylinder N_{zyl} in Kombination mit dem Zylindervolumen V_d das gesamte zur Verfügung stehende Volumen dar:

$$W_{zu,ab} = \eta_{vol} \cdot \frac{p_{zu}(t) \cdot N_{zyl} \cdot V_d}{R \cdot T_{zu}(t)} \cdot \frac{N_e(t)}{60 \cdot N_{re}} \tag{3.10}$$

Eine Veränderung der eingespritzten Brennstoffmasse kann sich erst auf das erzeugte Drehmoment auswirken, wenn eine Verbrennung in einem der Zylinder stattgefunden hat. Da durch die Verwendung des Mittelwertmodells dies nicht in physikalischen Gleichungen berücksichtigt werden kann, muss eine Totzeit [1] $\tau_{IPS}(t)$ diese Verzögerung modellieren. Das selbe Verhalten weist die ausströmende Luftmasse aus dem Motorblock auf, auch hier wird eine weitere Totzeit [2] $\tau_{IEG(t)}$ berücksichtigt [Guzzella und Onder, 2004].

3.5. Kühlereinheiten

Die Kühler in einem Verbrennungsmotor dienen dazu, das einströmende heiße Gasgemisch abzukühlen. Dazu muss dem Medium Energie entzogen werden. Die niedrigeren Temperaturen an der Ausgangsseite senken die Temperatur in den Zylindern ab, wodurch sich verbesserte Eigenschaften bezüglich der Schadstoffbildung ergeben. Die entzogene Energie wird durch ein flüssiges Kühlmittel abgeführt. Man kann sich einen an dieser Stelle modellierten Kühler als einen Wärmetauscher vorstellen. Die Modellierung der den Wärmeübertragungen zugrunde liegenden physikalischen Beziehungen würde die Modellierungstiefe des in dieser Arbeit erforderlichen Modells übersteigen. Deshalb wird ein verhältnismäßig einfaches Modell, aus [Jung, 2003] für die Darstellung der Kühler verwendet, wobei die Temperatur der Kühlmittelflüssigkeit T_{KM} als konstant angenommen und ein Effizienzfaktor η des gesamten Kühlers eingeführt wird. Damit erhält man die Austrittstemperatur $T_{ab}(t)$ des ausströmenden Gases in Abhängigkeit der Eintrittstemperatur $T_{zu}(t)$:

$$T_{ab}(t) = \eta \cdot T_{KM} + (1 - \eta) \cdot T_{zu}(t) \tag{3.11}$$

Durch die Einführung der Effizienz η besteht die Möglichkeit festzulegen, in welchem Maße die Übertragung der Wärme stattfindet. Dies ist wiederum vom Aufbau des Kühlers abhängig, wird an dieser Stelle jedoch nicht näher behandelt.

3.6. Darstellung des Anteils verbrannter Luft

Die von dem Kompressor aus der Umgebung angesaugte Luft besteht zu ungefähr 78% aus Stickstoff (N_2), zu 21% aus Sauerstoff (O_2) und zu 1% aus einer Vielzahl von anderen Gasen. Für die Verbrennung in einem Dieselmotor ist jedoch nur der Sauerstoff relevant, da nur

[1] induction to power stroke - Verzögerung vom Eintritt des Kraftstoffes in den Verbrennungsraum bis zum Anliegen des Motordrehmoments an der Welle

[2] induction to exhaust gas - Verzögerung vom Eintritt in den Verbrennungsraum bis zum Austritt in den Abgasstrang

dieser in der Verbrennung eine Reaktion eingeht. Er reagiert mit dem im Kraftstoff enthaltenen Kohlenstoff (C) zu Kohlendioxid (CO_2). Da der Sauerstoff ohne Einwirkung einer anderen Komponente aus der Luft eine Reaktion eingeht, kann die Betrachtung auf verbrannte Luft und unverbrannte Luft eingeschränkt werden. Um die Zusammensetzung des Gases im Motor zu beschreiben, ist es notwendig, den Anteil der verbrannten Luft $F(t)$ im Einlass sowie im Auslass zu modellieren. Die logische Definition aus [Chauvin und Petit, 2007] lautet:

$$F(t) = \frac{m_{verbrannt}(t)}{m(t)} = \frac{m(t) - m_{Frischluft}(t)}{m(t)} \tag{3.12}$$

und deren Ableitung:

$$\frac{dF(t)}{dt} = \frac{\dot{m}_{verbrannt}(t) \cdot m(t) - m_{verbrannt}(t) \cdot \dot{m}(t)}{m^2(t)} \tag{3.13}$$

Die Summe aus der Masse der verbrannten Luft $m_{verbrannt}(t)$ und der unverbrannten Luft $m_{Frischluft}(t)$ ergibt die gesamte im Volumen befindliche Masse $m(t)$. Da nun der Anteil der unverbrannten Luft bekannt ist, ist es einfach möglich dies in Verhältnis zu jener Frischluft zu setzen, die notwendig ist, um eine vollständige Verbrennung[3] zu erreichen:

$$\lambda = \frac{\dot{m}(t) \cdot (1 - F)}{f_{st} \cdot \dot{m}_f(t)} \tag{3.14}$$

Der Zähler repräsentiert die vorhandene Frischluft, im Nenner befindet sich die vorhandene Kraftstoffmenge m_f multipliziert mit dem Stöchiometrischen Faktor für Diesel f_{st}, der beschreibt wie viel Sauerstoff notwendig ist, um den Kraftstoff zu verbrennen. Es wird ein λ von mindestens 1 benötigt, um eine vollständige Verbrennung zu erreichen. In der Praxis muss das Verhältnis sogar höher sein, da nicht von einer idealen Vermischung der Gase im Brennraum ausgegangen werden kann.
Wird $m_{verbrannt}(t)$ durch den 1. Teil der Gleichung 3.12 ersetzt, führt dies zu:

$$\frac{dF(t)}{dt} = \frac{\dot{m}_{verbrannt}(t) - F(t) \cdot \dot{m}(t)}{m(t)} \tag{3.15}$$

Diese Ableitung wird später in Kapitel 4 benötigt, um die Differentialgleichungen für die Anteile der verbrannten Luft in den jeweiligen Rohrabschnitten zu erhalten.

3.7. Modellierung der Turboaufladung

In dem zu modellierenden Motor wird ein Abgasturbolader verwendet. Dieser nutzt die im Abgas enthaltene kinetische Energie, um über ein Schaufelrad eine Welle anzutreiben. Am anderen Ende der Welle ist ebenfalls ein Schaufelrad angebracht, welches die einströmende Luft verdichtet. Da die Aufladung über eine VGT verfügt, ist es möglich, über die Stellung $x_{vgt}(t)$ der Turbinenschaufel die Stärke der Kompression der Frischluft einzustellen. Die Drehzahl der starren Welle kann mit Hilfe des Drallsatzes über eine Leistungsbilanz gebildet werden, Glg.

[3] Bei einer vollständigen Verbrennung geht der gesamte im Kraftstoff enthaltene Kohlenstoff zu Kohlendioxid über

3.16. Dieser beschreibt die Änderung der Drehzahl $N_t(t)$ aufgrund des einwirkenden Drehmoments $M(t)$, unter der gegebenen Geometrie mit dem Massenträgheitsmoment J:

$$\frac{dN_t(t)}{dt} = \frac{60}{2\pi}\frac{M(t)}{J} \tag{3.16}$$

Das Drehmoment wird durch eine Bilanzierung der Leistungen und der momentan vorliegenden Turbinendrehzahl $N_t(t)$ errechnet:

$$M(t) = \frac{60}{2\pi}\frac{P_{zu}(t) - P_{ab}(t)}{N(t)} \tag{3.17}$$

Darin fließen die von der Turbine auf die Welle gebrachte Leistung $P_{zu}(t)$ und die von dem Kompressor entzogene Leistung $P_{ab}(t)$ ein. Setzt man Gleichung 3.17 in Gleichung 3.16 ein erhält die Differentialgleichung zur Beschreibung der Turbinenrehzahl:

$$\frac{dN_t(t)}{dt} = \left(\frac{60}{2\pi}\right)^2 \frac{P_{zu}(t) - P_{ab}(t)}{J \cdot N_t(t)} \tag{3.18}$$

Zur Beschreibung des Kompressorverhaltens wird ein üblicher Ansatz aus [Moraal und Kolmanovsky, 1999] und [Jung, 2003] verwendet. Dabei wird zuerst ein isentropischer Prozess angenommen, das heisst, die Entropie $S(t)$ ist konstant. Damit kann das Verhältnis der Temperaturen und Drücke, die vor und nach dem isentropischen Prozess vorherrschen, wie folgt beschrieben werden:

$$\frac{T_{ab,is}(t)}{T_{zu}(t)} = \left(\frac{p_{ab}(t)}{p_{zu}(t)}\right)^{\frac{\gamma-1}{\gamma}} \tag{3.19}$$

Aufgrund der Tatsache, dass ein Prozess in der Realität nicht ohne Verluste ablaufen kann, muss eine Maßnahme getroffen werden, diese in der Modellierung zu berücksichtigen. An dieser Stelle wird eine Effizienz $\eta_c(t)$ eingeführt, welche die Verluste, die bei der Kompression auftreten, ausreichend mitberücksichtigt:

$$\eta_c(t) = \frac{T_{ab,is}(t) - T_{zu}(t)}{T_{ab}(t) - T_{zu}(t)} \tag{3.20}$$

Gleichung 3.20 wird auf $T_{ab}(t)$ umgeformt und die darin enthaltene Größe $T_{ab,is}(t)$ durch die Beziehung 3.19 ersetzt. Dadurch erhält man als Ergebnis mit Gleichung 3.21 eine Beziehung für die Ausgangstemperatur $T_{ab}(t)$ des Kompressors in Abhängigkeit der Eigenschaften des einströmenden Gases, Temperatur $T_{zu}(t)$, Druck $p_{zu}(t)$, des Druckes am Ausgang $p_{ab}(t)$, zuzüglich einer Effizienz $\eta_c(t)$:

$$T_{ab}(t) = T_{zu}(t) + \frac{T_{zu}(t)}{\eta_c(t)}\left(\left(\frac{p_{ab}(t)}{p_{zu}(t)}\right)^{\frac{\gamma-1}{\gamma}} - 1\right) \tag{3.21}$$

Somit fehlt noch der Zusammenhang zwischen dem Massenfluss $W_c(t)$ und der in der Differentialgleichung 3.17 benötigten Leistung $P(t)$. Diese Leistung ist erforderlich, um die Luftmasse $W(t)$ befördern zu können. In diese Gleichung fließen die Temperatur vor dem Kompressor, also die Außentemperatur $T_a(t)$, jene nach der Kompression $T_c(t)$, sowie der spezifische Wärmekoeffizient bei konstantem Druck c_p ein:

$$P_c(t) = W_c(t) \cdot c_p \cdot (T_c(t) - T_a(t)) \tag{3.22}$$

Kapitel 4

Umsetzung in Modelica

Dieses Kapitel beschreibt die konkrete Implementierung des Luftpfadmodells einer Verbrennungskraftmaschine. Als markante Eigenschaften sind der Abgasturbolader und die Abgasrückführung zu erwähnen, welche beide über ein Steuerungsventil verfügen. Das gesamte System wird in Teilkomponenten zerlegt, um die Übersichtlichkeit und Modularität zu erhalten. Die Parametrierung der verwendeten Kennlinien erfolgte einerseits durch Herstellerdaten andererseits aus den vorhandenen Messdaten des Motors. Diese Kennlinien werden aus dem bereits vorhandenem Matlab/Simulink Modell verwendet und in diesem Kapitel nochmals abgebildet. Die meisten Kennlinien wurden über die Methode der kleinsten Fehlerquadrate an die vorliegenden Messdaten angepasst.

4.1. Notation

Sämtliche Größen aus dem in Modelica implementierten System werden in Tabelle 4.1 angeführt. Die Gruppierung erfolgt anhand der erstellten Teilkomponenten. Alle in dieser Tabelle angeführten Größen sind von der Zeit abhängig, wobei dies aufgrund der Übersichtlichkeit in der Notation nicht berücksichtigt wurde.

Benennung	**Einheit**	**Erklärung**
p_i	Pa	Druck im Einlassvolumen
m_i	kg	Masse im Einlassvolumen
F_i	1	Anteil verbrannter Luft im Einlassvolumen
T_i	K	Temperatur im Einlassvolumen
p_x	Pa	Druck im Auslassvolumen
m_x	kg	Masse im Auslassvolumen
F_x	1	Anteil verbrannter Luft im Auslassvolumen
T_x	K	Temperatur im Auslassvolumen
N_e	rpm	Motordrehzahl
T_e	K	Temperatur im Motorblock
M_e	Nm	Erzeugtes Drehmoment des Motors
m_f	$\frac{kg}{cyl}$	Kraftstoff pro Zyklus
W_f	$\frac{kg}{s}$	Massenfluss Kraftstoff

W_{ie}	$\frac{kg}{s}$	Massenfluss vom Einlass in den Motorblock
W_{ex}	$\frac{kg}{s}$	Massenfluss aus dem Motorblock in den Auslass
λ_{cyl}	1	Luft-Kraftstoffverhältnis im Zylinder
τ_{IPS}	s	Verzögerung von der Einspritzung zum Drehmoment
τ_{IEG}	s	Verzögerung von der Einspritzung zum Abgas
p_{ic}	Pa	Druck im Ladeluftkühler
m_{ic}	kg	Masse im Ladeluftkühler
W_{ci}	$\frac{kg}{s}$	Massenfluss vom Ladeluftkühler in den Einlass
$\tilde{T}_{ic}$	K	Temperatur im Volumen des Ladeluftkühler
T_{ic}	K	Temperatur nach dem Ladeluftkühler
x_{air}	$\%$	Stellung der Drosselklappe
W_{xi}	$\frac{kg}{s}$	Massenfluss über die Abgasrückführung
T_{xi}	K	Temperatur des rückgeführten Gases
x_{egr}	$\%$	Stellung der Klappe in der Agbasrückführung
N_t	rpm	Turbodrehzahl
W_c	$\frac{kg}{s}$	Massenfluss der Frischluft über den Kompressor
W_{xt}	$\frac{kg}{s}$	Massenfluss des Abgases über die Turbine
p_t	Pa	Druck am Turbinenausgang
p_a	Pa	Umgebungsdruck
T_a	K	Umgebungstemperatur
T_c	K	Temperatur am Kompressorausgang
T_t	K	Temperatur am Turbinenausgang
x_{vgt}	$\%$	Position der Turbinenschaufel
η_c	1	Effizienz des Kompressors
η_t	1	Effizienz der Turbine
P_c	W	Leistung des Kompressors
P_t	W	Leistung der Turbine

Tabelle 4.1.: Notation der verwendeten Größen

4.1.1. Konstanten

Die in Tabelle 4.2 angeführten Konstanten sind nicht vom Aufbau des Motors abhängig, sondern Eigenschaften des Mediums Luft und des verwendeten Kraftstoffs Diesel. In Dymola werden diese Konstanten in einem *record* repräsentiert. Dadurch ergibt sich der Vorteil, dass alle von der Umwelt bestimmten Konstanten, kompakt an einer Stelle des Modells zusammengefasst dargestellt sind. Aus den in Tabelle 4.2 definierten Konstanten werden noch die *spezifische Gaskonstante für Luft* R und das *spezifische Wärmeverhältnis* γ berechnet [Guzzella und Onder, 2004]:

$$\begin{aligned} R &= c_p - c_v = 287 \frac{J}{kg \cdot K} \\ \gamma &= \frac{c_p}{c_v} = 1.395 \end{aligned} \tag{4.1}$$

Benennung	Wert	Einheit	Erklärung
c_p	1014.4	$\frac{J}{kg \cdot K}$	Spez. Wärmekoeffizient bei konst. Druck
c_v	727.4	$\frac{J}{kg \cdot K}$	Spez. Wärmekoeffizient bei konst. Temperatur
f_{st}	14.6	1	Stöchiometrischer Faktor für Diesel
HW_u	$43 \cdot 10^6$	$\frac{J}{kg}$	unterer Heizwert für Diesel

Tabelle 4.2.: Allgemeine Konstanten

Sämtliche motorspezifischen Konstanten, welche durch die Bauart festgelegt sind, werden in Tabelle 4.3 definiert.

Benennung	Wert	Einheit	Erklärung
N_{zyl}	6	1	Zylinderanzahl
N_{re}	2	1	Umdrehungen pro Zyklus
V_d	0.03	m^3	Volumen eines Zylinders
V_i	0.0324	m^3	Volumen des Motoreinlasses
V_x	0.0054	m^3	Volumen des Motorauslasses
V_{ic}	0.0324	m^3	Volumen des Ladeluftkühlers
$T_{KM,egr}$	356	K	Temperatur des Kühlmittels in der Abgasrückführung
$T_{KM,ic}$	297	K	Temperatur des Kühlmittels im Ladeluftkühler
η_{egr}	0.96	1	Effizienz der Abgasrückführung
η_{ic}	0.96	1	Effizienz des Ladeluftkühlers

Tabelle 4.3.: Motorspezifische Konstanten

4.2. Thermodynamische Verbindung der Komponenten

Am realen Motor sind die unterschiedlichen funktionalen Einheiten durch Rohre verbunden. Um diese Verbindung auch in Dymola repräsentieren zu können, wird ein neuer *Connector* erstellt. Dadurch erhält man die Möglichkeit in einem Schritt die Größen Druck $p(t)$, Massenfluss $\dot{m}(t)$ und Temperatur $T(t)$ zu koppeln. Der Aufbau des Gesamtsystems kann dadurch wesentlich übersichtlicher gestaltet werden:

```
1 connector Pipe
2   Modelica.SIunits.Pressure p;
3   Modelica.SIunits.MassFlowRate W;
4   Modelica.SIunits.Temperature T;
5 end Pipe;
```

Listing 4.1: Thermodynamische Verbindung

Dieses Modell einer Rohrverbindung zweier Komponenten überträgt die zuvor aufgezählten thermodynamischen Zustände ohne Verzögerung. Außerdem unterliegen sie keiner Veränderung, zum Beispiel Wärmeverlust an den Rohrwänden. Diese Eigenschaften werden an dieser Stelle nicht modelliert, sondern in das Verhalten der Komponenten hinzugefügt. Dadurch erhält

man idealisierte Verbindungen, die die berechneten Größen direkt an die jeweils verbundenen Komponenten weitergeben.

4.3. Lufteinlass

Diese Komponente besteht im Wesentlichen aus einem Volumen, welches wie in Kapitel 3 beschrieben modelliert wird. Das Volumen V_i wird durch einen Luftmassenstrom $W_{ci}(t)$ aus dem Ladeluftkühler und einem Luftmassenstrom $W_{xi}(t)$ aus der Abgasrückführung gespeist, und gibt die Masse an den Motorblock in Form von $W_{ie}(t)$ zur Verbrennung ab. Um den Anteil der verbrannten Luft im Einlassvolumen $F_i(t)$ berechnen zu können, muss zuvor die Beziehung 4.2 für die Änderung der verbrannten Masse $\dot{m}_{i,verbrannt}(t)$ hergestellt werden. Dies ergibt sich aus dem Massenstrom, der aus dem Abgas zurückgeführt wird und jenem, der vom Einlass in den Motor geleitet wird:

$$\dot{m}_{i,verbrannt}(t) = W_{xi}(t) \cdot F_x(t) - W_{ie}(t) \cdot F_i(t) \tag{4.2}$$

Zusätzlich wird die Änderung der Masse $\dot{m}_i(t)$ benötigt. Diese ergibt sich aus den bereits beschriebenen Massenflüssen, wobei $W_{ie}(t)$ für eine spätere Vereinfachung in die beiden Anteile verbrannte Luft und Frischluft aufgeteilt wird:

$$\dot{m}_i(t) = W_{ci}(t) + W_{xi}(t) - W_{ie}(t) \cdot F_i(t) - W_{ie}(t) \cdot (1 - F_i(t)) \tag{4.3}$$

Die beiden zuvor aufgestellten Gleichungen 4.2 und 4.3 werden in die in Kapitel 3 ermittelte Gleichung 3.15 eingesetzt, um nach Vereinfachungen auf den Anteil der verbrannten Luft im Verhältnis zur gesamten Luftmasse $F_i(t)$ im Einlassvolumen zu kommen. Das gesamte mathematische Modell für den Lufteinlass, welches in Dymola mittels der Texteingabe realisiert wurde, ist in den Gleichungen 4.4 zusammengefasst:

$$\begin{aligned}
\frac{dp_i(t)}{dt} &= \frac{\gamma \cdot R}{V_i} \left[T_{ic}(t) \cdot W_{ci}(t) + T_{xi}(t) \cdot W_{xi}(t) - T_i(t) \cdot W_{ie}(t)\right] \\
\frac{dm_i(t)}{dt} &= W_{ci}(t) + W_{xi}(t) - W_{ie}(t) \\
T_i(t) &= \frac{p_i(t) \cdot V_i}{R \cdot m_i(t)} \\
\frac{dF_i(t)}{dt} &= \frac{[F_x(t) - F_i(t)] \cdot W_{xi}(t) - F_i(t) \cdot W_{ci}(t)}{m_i(t)}
\end{aligned} \tag{4.4}$$

Für die Berechnung des Drucks im Einlass $p_i(t)$ ist die Temperatur aus dem Ladeluftkühler $T_{ic}(t)$, aus der Abgasrückführung $T_{xi}(t)$ und jene aus dem Einlassvolumen $T_i(t)$ notwendig. In Abb. 4.1 ist die grafische Darstellung des Lufteinlasses in Dymola abgebildet. Darauf sind die orangen Rohrverbindungen sichtbar und die zusätzlich für die Interaktion mit anderen Blöcken benötigten Signale, Einlasstemperatur $T_i(t)$, Anteil verbrannter Luft im Auslass $F_x(t)$, Anteil verbrannter Luft im Einlass $F_i(t)$ und Ladedruck $p_i(t)$.

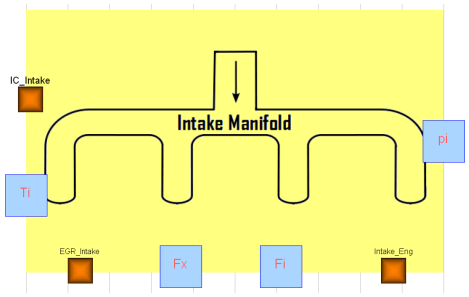

Abbildung 4.1.: Darstellung des Lufteinlassvolumen in Dymola

4.4. Luftauslass

Prinzipiell werden für die Modellierung dieser Komponente die gleichen Grundlagen benötigt, die bereits für den Lufteinlass herangezogen wurden. Das Volumen wird vom Massenfluss aus dem Motorblock $W_{ex}(t)$ gefüllt, und entleert sich über die Abgasrückführung $W_{xi}(t)$ und über den Turbolader $W_{xt}(t)$. Zu beachten ist für den Anteil der verbrannten Luft $F_x(t)$, dass man die Masse des Brennstoffes $W_{f,\tau}(t)$ hinzuzählt. Dafür werden wieder die beiden Beziehungen für die Änderung der verbrannten Masse im Auslass $\dot{m}_{x,verbrannt}(t)$ und für die Änderung der Masse im Auslass $\dot{m}_x(t)$ in den Gleichungen 4.5 und 4.6 aufgestellt.

$$\dot{m}_{x,verbrannt}(t) = W_{ie}(t) \cdot F_i(t) + W_{f,\tau}(t) \cdot (1 + f_{st}) - (W_{xi}(t) + W_{xt}(t)) \cdot F_x(t) \tag{4.5}$$

Für die Berechnung der Änderung der verbrannten Masse im Auslass $\dot{m}_{x,verbrannt}(t)$ in Gleichung 4.5 wird von einer vollständigen Verbrennung der um eine Totzeit $\tau_{IEG(t)}$ verzögerten eingespritzten Kraftstoffmenge $W_{f,\tau}(t)$ ausgegangen. Eine genauere Beschreibung der Totzeit erfolgt in Kapitel 4.5. Deshalb ist es notwendig, zur verbrannten Masse die Kraftstoffmenge $W_{f,\tau}(t)$ multipliziert mit dem stöchiometrischen Faktor für Diesel f_{st} hinzu zu addieren. Dieser gibt an, wie viel Teile Frischluft benötigt werden, um 1 Teil Diesel zu verbrennen. Abgezogen werden die beiden Komponenten, welche in die Abgasrückführung und in den Turbolader abfließen:

$$\dot{m}_x(t) = W_{ex}(t) - W_{xt}(t) - W_{xi}(t) \cdot F_x(t) - W_{xi}(t) \cdot (1 - F_x(t)) \tag{4.6}$$

An dieser Stelle werden die beiden zuvor ermittelten Zusammenhänge 4.5 und 4.6 in die Gleichung 3.15 aus Kapitel 3 eingesetzt, um die endgültige Beziehung für den Anteil an verbrannter Luft im Auslass $F_x(t)$ zu erhalten. Das gesamte Luftauslassmodell ist in den Gleichungen 4.7 zusammengefasst:

$$\begin{aligned}
\frac{dp_x(t)}{dt} &= \frac{\gamma \cdot R}{V_x} \left[T_e(t) \cdot W_{ex}(t) - T_x(t) \cdot W_{xi}(t) - T_x(t) \cdot W_{xt}(t)\right] \\
\frac{dm_x(t)}{dt} &= W_{ex}(t) - W_{xi}(t) - W_{xt}(t) \\
T_x(t) &= \frac{p_x(t) \cdot V_x}{R \cdot m_x(t)} \\
\frac{dF_x(t)}{dt} &= \frac{F_i(t) \cdot W_{ie,\tau}(t) + [1 + f_{st}] \cdot W_{f,\tau}(t) - F_x(t) \cdot W_{ex}(t)}{m_i(t)}
\end{aligned} \tag{4.7}$$

Für die Berechnung des Drucks im Auslass $p_x(t)$ wird das Volumen V_x desselben, sowie die Temperatur aus dem Verbrennungsvorgang $T_e(t)$ und die Temperatur im Auslassvolumen $T_x(t)$

benötigt. In Abbildung 4.2 ist wieder die grafische Darstellung in Dymola abgebildet. Neben Größen, die durch die Rohrverbindungen resultieren, wird für die Koppelung noch der Anteil an verbrannter Luft im Auslass $F_x(t)$, der Anteil an verbrannter Luft im Einlass $F_i(t)$, die eingespritzte Kraftstoffmenge $W_{f,\tau}(t)$ und der verzögerte Massenfluss aus dem Einlass $W_{ie,\tau}(t)$ benötigt.

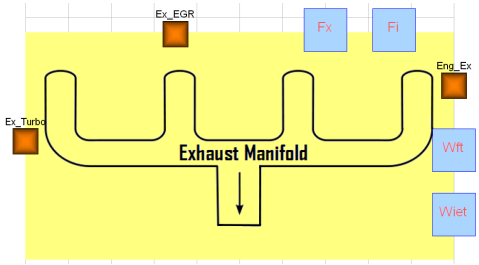

Abbildung 4.2.: Darstellung des Luftauslassvolumen in Dymola

4.5. Motorblock

In dieser Komponente sind keine Differentialgleichungen vorhanden, es werden lediglich algebraische Beziehungen hergestellt und Kennlinien bzw. Kennflächen hinterlegt. Sämtliche Kennlinien in dieser Arbeit werden durch *functions* in Dymola implementiert. In Listing 4.2 ist exemplarisch die Realisierung der Kennlinie für die Verbrennungstemperatur T_e in Dymola dargestellt. Diese Kennlinie ist quadratisch vom Luft-Kraftstoff Verhältnis und linear von der Motordrehzahl abhängig. Das Ergebnis ist in Abbildung 4.3 zu sehen. Die blauen Kreuze in diesem Diagramm stellen sämtliche Messwerte dar. Diese wurden als Grundlage für das Anpassen der Kennfläche herangezogen. Die grünen Linien repräsentieren die hinterlegte Kennfläche.

```
1  function temperatureEngine
2    input Real lambda_cyl;          // Air fuel ratio
3    input Real N_e;                 // Engine speed
4    output Real T_e;                // Temperature engine
5
6    parameter Real[4] K_T_e = {103.27, -682.63, 167.95, 1414.77};
7    Real[4] P_T_e;
8    Real lambda;
9
10 algorithm
11
12   lambda :=if (lambda_cyl > 3) then 3 else lambda_cyl;
13   P_T_e :={lambda^2,lambda,N_e/1000,1};
14   T_e := K_T_e * P_T_e;
15
16 end temperatureEngine;
```

Listing 4.2: Kennfläche der Verbrennungstemperatur T_e

Wie bereits erwähnt ist die eingespritzte Kraftstoffmenge m_f eine Eingangsgröße und sie ist unabhängig vom aktuellen Zustand des Luftpfades. Dadurch kann es vorkommen, dass bei einem

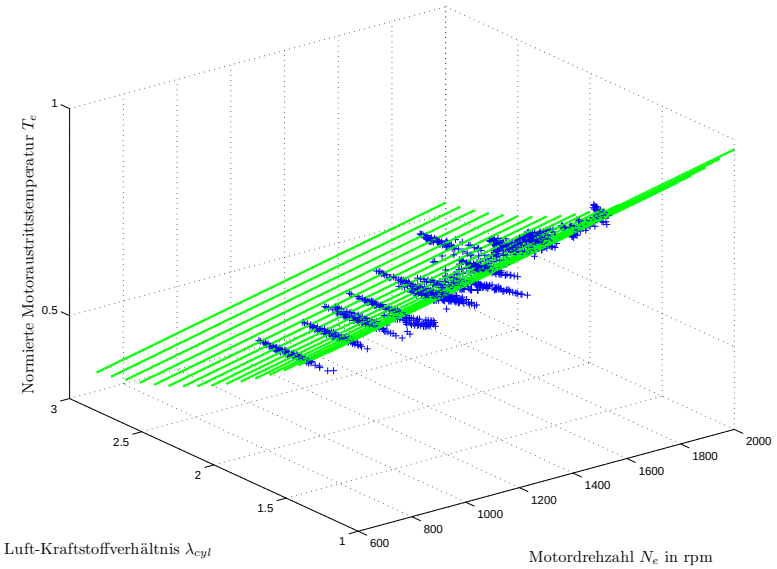

Abbildung 4.3.: Kennfläche der Brenntemperatur

hohen Motordrehmoment zu viel Kraftstoff zugeführt wird. Da dieser nicht mehr ausreichend Sauerstoff für die vollständige Verbrennung vorfindet, beginnt der Motor Ruß aus zustoßen. Durch eine Begrenzung des Kraftstoffes kann dieser Zustand verhindert werden, wobei die maximal zulässige Menge $W_{f,max}$ in Gleichung 4.8 berechnet wird:

$$W_{f,max} = \frac{W_{ie}(t) \cdot (1 - F_i(t))}{f_{st}} \tag{4.8}$$

Diese Begrenzung wird in Dymola durch den Befehl *result = min(a,b)* im Motorblock realisiert, welcher den jeweils kleineren Wert für die Zuweisung zur Größe auf der linken Seite heranzieht. In Gleichung 4.9 wird die zuvor beschriebene Begrenzung angewendet. Entweder wird genau soviel Kraftstoff zugeführt, wie durch die vorhandene Frischluft verbrannt werden kann, oder jene Menge, die sich über die aktuelle Eingangsgröße $m_f(t)$ mit Hilfe der aktuellen Motordrehzahl $N_e(t)$, der Anzahl der Zylinder N_{zyl} und der Umdrehungen pro Zyklus N_{re} ergibt:

$$W_f = min\left(W_{f,max}, \frac{m_f(t) \cdot N_e(t) \cdot N_{zyl}}{60 \cdot N_{re}}\right) \tag{4.9}$$

Die aus der Modellierungsart resultierenden Totzeiten im Motorblock können ebenfalls bequem mittels *delay(Ausrduck,Zeit)*, aus [Fritzson, 2003, S.203], umgesetzt werden. Für die beiden Massenflüsse $W_f(t)$ und $W_{ie}(t)$ tritt die Totzeit $\tau_{IEG}(t)$ auf, welche von der Einspritzung bis zum Ausstoß aus den Zylindern dauert:

$$\begin{aligned} W_{f,\tau}(t) &= W_f(t - \tau_{IEG(t)}) \\ W_{ie,\tau}(t) &= W_{ie}(t - \tau_{IEG(t)}) \\ \tau_{IEG}(t) &= \frac{90}{N_e(t)} \end{aligned} \tag{4.10}$$

Die Verzögerung $\tau_{IPS}(t)$ stellt die Zeit dar, die vom Einspritzen des Kraftstoffes bis zum Anliegen des korrespondierenden Motordrehmomentes $M_{e,\tau}(t)$ an der Antriebswelle vergeht:

$$\begin{aligned} M_{e,\tau}(t) &= M_e(t - \tau_{IPS}(t)) \\ \tau_{IPS}(t) &= \frac{60}{N_e(t)} \end{aligned} \tag{4.11}$$

Die Luftmasse $W_{ie}(t)$, welche durch den Motor gepumpt wird, resultiert aus Gleichung 3.10:

$$W_{ie}(t) = \eta_{vol} \cdot \frac{p_i(t) \cdot N_{zyl} \cdot V_d}{R \cdot T_i(t)} \cdot \frac{N_e(t)}{60 \cdot N_{re}} \tag{4.12}$$

Für die zugeführten thermodynamischen Größen wird der Ladedruck $p_i(t)$ und die Temperatur im Einlass $T_i(t)$ eingesetzt. Die Parametrierung der Kennlinien erfolgt durch das Verhältnis von Frischluft zu Kraftstoff λ_{cyl}, welches sich nachvollziehbar aus dem Frischluftanteil des Einlassmassenstroms $(1 - F_i)$ und der eingespritzten Kraftstoffmenge W_f berechnen lässt:

$$\lambda_{cyl} = \frac{W_{ie} \cdot (1 - F_i)}{f_{st} \cdot W_f} \tag{4.13}$$

Die Parametrierung der Motordrehmomentkennlinie ist in [Guzzella und Onder, 2004, S.65] detailliert ausgeführt. Dazu wird einerseits der mittlere Druck im Zylinder p_{me} definiert, und andererseits der durch den zugeführten Kraftstoff erzeugte mittlere Druck p_{mf} bei einer Effizienz von 1:

$$\begin{aligned} p_{me} &= \frac{M_e(t) \cdot 4 \cdot \pi}{V_d \cdot N_{cyl}} \\ p_{mf} &= \frac{HW_u \cdot m_f}{V_d \cdot N_{cyl}} \end{aligned} \tag{4.14}$$

HW_u symbolisiert den unteren Heizwert für Diesel. Durch die Kennfläche der thermodynamischen Effizienz kann nun die Beziehung für das Motordrehmoment in Gleichung 4.15 angegeben werden. In Abbildung 4.4 werden anhand der blauen Kreuze wieder alle Werte der thermodynamischen Effizienz, die aus den Messdaten des Prüfstandes kommen, dargestellt.

$$M_e(t) = \frac{V_d \cdot N_{cyl}}{N_{re} \cdot 2\pi} \left(p_{mf} \cdot \eta_{th} - p_{me0f} - p_{me0g}\right) \tag{4.15}$$

Die Repräsentation des Motorblocks in Dymola ist in Abbildung 4.5 dargestellt. Zuzüglich zu den beiden Rohrverbindungen werden die beiden Eingänge des Luftpfades, die Kraftstoffmenge $m_f(t)$ und die aktuelle Motordrehzahl $N_e(t)$ in dieser Komponente benötigt. Ebenfalls ist es notwendig, die zuvor berechneten verzögerten Werte Kraftstoffffluss $W_{f,\tau}(t)$, Massenfluss in den Motorblock $W_{ie,\tau}(t)$ und Motordrehmoment $M_{e,\tau}(t)$ den anderen Teilen der Modellierung zur Verfügung zu stellen.

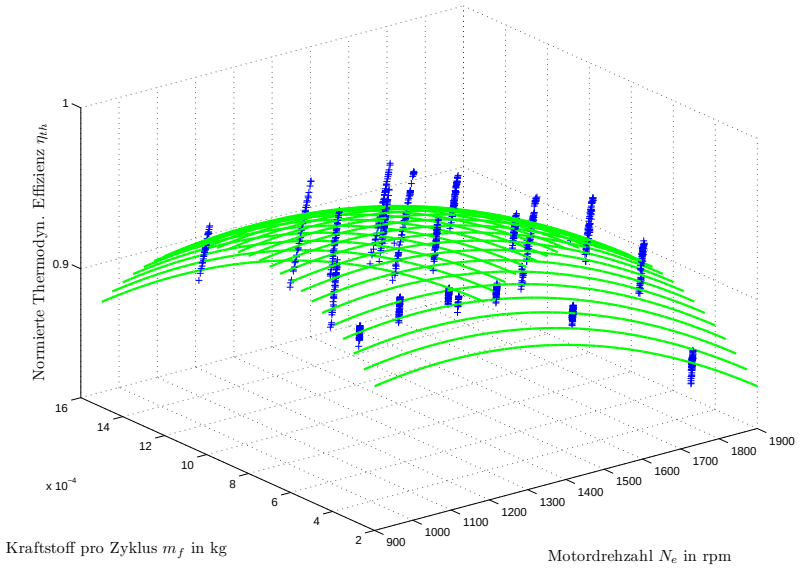

Abbildung 4.4.: Kennfläche der thermodynamischen Effizienz

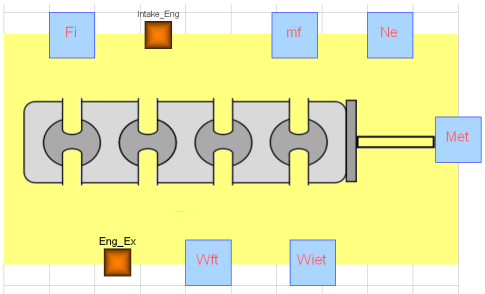

Abbildung 4.5.: Darstellung des Motorblocks in Dymola

4.6. Abgasrückführung

Die Abgasrückführung besteht im wesentlichen aus einem Kühler, der die hohe Temperatur des Abgases reduziert und einem Ventil. Durch die Ventilstellung $x_{egr}(t)$ wird die Luftmasse, welche rückgeführt wird, gesteuert. Das Ventil ist bei einem Wert von 0% geschlossen und lässt bei 100% die maximale Luftmasse durch. Der Effekt des Kühlers wird einfach aus Gleichung 3.11 übernommen und führt zu folgender Beziehung:

$$T_{xi}(t) = \eta_{egr} \cdot T_{KM,egr} + (1 - \eta_{egr}) \cdot T_x(t) \tag{4.16}$$

Die Effizienz des Kühlers η_{egr} und die Temperatur des Kühlmittels $T_{KM,egr}$ ist aus Tabelle 4.3 zu entnehmen. Die tatsächliche rückgeführte Luftmasse $W_{xi}(t)$ wird durch die Gleichung 3.8 für Rohrrestriktionen ermittelt. Dabei tritt jedoch das Problem auf, dass die Fläche, welche zum Durchströmen zur Verfügung steht, unbekannt ist. Diese ist zwar hauptsächlich von der Position der Klappe $x_{egr}(t)$ abhängig, weist jedoch in geöffneter Position auch eine starke Abhängigkeit vom Druckverhältnis $p_r(t)$ auf. Deshalb werden beide Größen für die Parametrierung herangezogen und resultieren in folgender Beziehung für die effektive Fläche:

$$A_{eff,egr}(t) = 5 \cdot 10^{-6} \cdot x_{egr}(t) + 3 \cdot 10^{-4} \cdot p_r(t)^{25} + e^{-1600 \cdot (1 - p_r(t) + 0.003)} \tag{4.17}$$

In den Abbildungen 4.7 und 4.6 sind die Abhängigkeiten von beiden Größen und die verwendete Kennlinie in zwei Ansichten ersichtlich. Beiden Abbildungen ist deutlich zu entnehmen, dass nicht alle Messpunkte, welche durch die blauen Kreuze repräsentiert werden, für die Ermittlung der jeweiligen Kennlinie herangezogen werden können.
Betrachtet man jene Messpunkte, die einer Klappenposition von $0\% - 95\%$ zugeordnet sind, ist eine deutliche lineare Abhängigkeit der effektiven Fläche $A_{eff,egr}(t)$ von der Klappenposition zu erkennen. Deshalb wird der lineare Term in der Kennlinie als sinnvoll erachtet. In diesem Bereich ist das Druckverhältnis $p_r(t)$ wesentlich kleiner als 1.
Betrachtet man die effektive Fläche in Abhängigkeit des an der Klappe anliegenden Druckverhältnis $p_r(t)$ lässt sich am ersten Blick keine eindeutige Beziehung erkennen. Nun erweist es sich als Vorteil, dass jene Messpunkte, an denen sich das Druckverhältnis deutlich von 1 unterscheidet, jene sind an denen die Klappe teilweise geschlossen ist. Dies ist auch nachvollziehbar, da nur wenn die Klappe zumindest teilweise geschlossen ist, auch ein Druckabfall auftreten kann. Deshalb ist es ausreichend, wenn der Term der die Abhängigkeit vom Druckabfall beschreibt nur jene Messpunkte ausreichend repräsentiert, die in der Nähe von $p_r(t) = 1$ sind. Diese Punkte werden durch eine geeignet angepasste Exponentialfunktion dargestellt. Nach ausgiebigen Variationen der Kennlinie stellte sich heraus, dass noch ein polynomialer Term hoher Ordnung notwendig ist, um die Messpunkte ausreichend zu beschreiben. In Abbildung 4.8 werden zu den Rohrverbindungen noch die Klappenposition $x_{egr}(t)$ und die Temperatur im Einlass $T_i(t)$ benötigt.

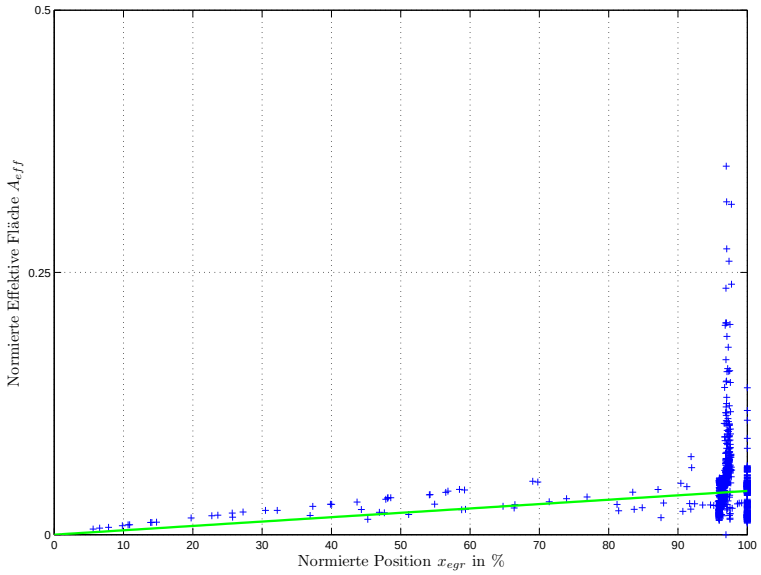

Abbildung 4.6.: Kennlinie der Rohrrestriktion in Abhängigkeit der Klappenposition

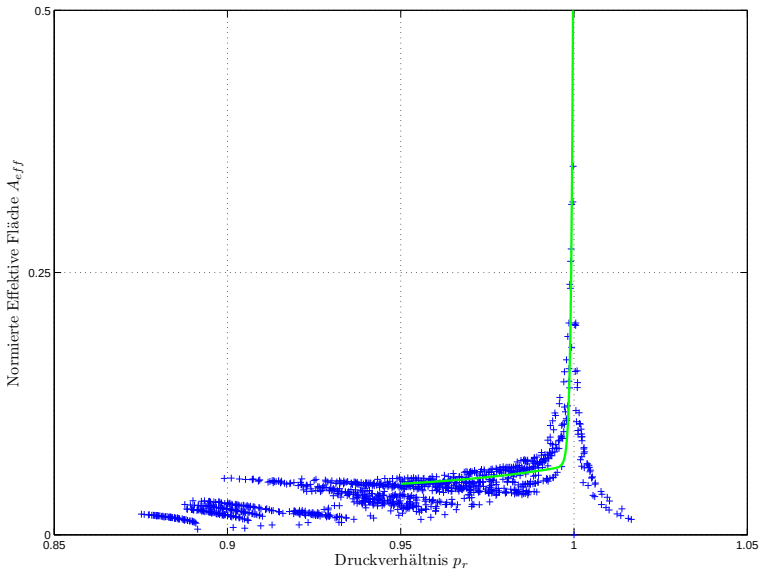

Abbildung 4.7.: Kennlinie der Rohrrestriktion in Abhängigkeit des Druckverhältnisses

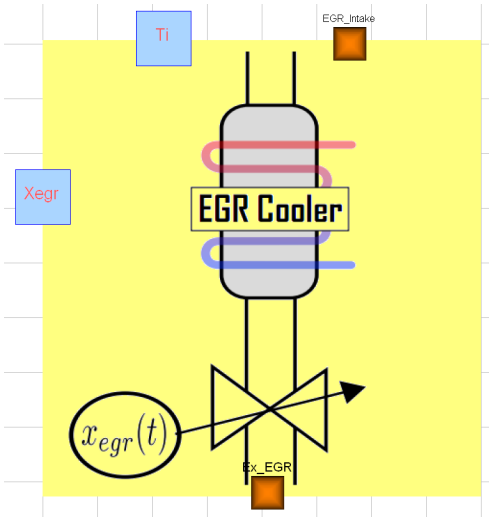

Abbildung 4.8.: Darstellung der Abgasrückführung in Dymola

4.7. Ladeluftkühler

Da die Größe dieser Komponente nicht unerheblich ist, wird ebenfalls ein Volumen V_{ic} modelliert, welches durch den Kompressor den Luftmassenstrom $W_c(t)$ erhält und diese an den Lufteinlass $W_{ci}(t)$ abgibt. Zusätzlich ist es vorgesehen, über eine Drosselklappenstellung $x_{air}(t)$ die Menge der Frischluft zu begrenzen. Anhand der bereits in den Gleichungen 3.5 dargestellten Beschreibung eines Volumen mit kompressiblem Medium werden folgende Beziehungen für das Volumen im Ladeluftkühler aufgestellt:

$$\begin{aligned} \frac{dp_{ic(}t)}{dt} &= \frac{\gamma \cdot R}{V_{ic}} \left[T_c(t) \cdot W_c(t) - \widetilde{T}_{ic}(t) \cdot W_{ci}(t) \right] \\ \frac{dm_{ic}(t)}{dt} &= W_c(t) - W_{ci}(t) \\ \widetilde{T}_{ic}(t) &= \frac{p_{ic}(t) \cdot V_{ic}}{R \cdot m_{ic}(t)} \end{aligned} \tag{4.18}$$

Neben dem Volumen wird natürlich auch die kühlende Wirkung modelliert. Dieser Effekt kann relativ einfach mathematisch dargestellt werden:

$$T_{ic}(t) = \eta_{ic} \cdot T_{KM,ic} + (1 - \eta_{ic}) \cdot \widetilde{T}_{ic}(t) \tag{4.19}$$

Die Effizienz η_{ic} der Kühleinheit und die Temperatur des Kühlmittels $T_{KM,ic}$ sind ebenfalls aus Tabelle 4.3 zu entnehmen. Im Volumen herrscht die Temperatur $\widetilde{T}_{ic}(t)$, welche sich beim Austritt aus dem Ladeluftkühler auf $T_{ic}(t)$ absenkt.
Die Darstellung der Drosselklappe stellte wiederum eine Herausforderung dar. Wie bereits bei der Modellierung der Klappe in der Abgasrückführung wird von einer Rohrrestriktion ausgegangen, die durch deren effektive Fläche $A_{eff,air}(t)$ beschrieben werden kann:

$$A_{eff,air}(t) = -2 \cdot 10^{-7} \cdot (x_{air}(t) - 100)^2 + 0.002 + e^{-450 \cdot (1 - p_r(t) + 0.0123)} \tag{4.20}$$

Diese Fläche ist nun wiederum von der aktuellen Klappenposition x_{air} und dem anliegenden Druckverhältnis p_r abhängig. Betrachtet man die beiden Abbildungen 4.9 und 4.10 kann wieder dasselbe Phänomen festgestellt werden, wie bei der Abgasrückführung. Bei teilweise geschlossener Drosselklappe, also alle Messpunkte deren Klappenposition zwischen 20% und 80% liegt, ist eine eindeutige Abhängigkeit von dieser Position $x_{air}(t)$ festzustellen. Mathematisch wird der zuvor beschriebene Zusammenhang durch den quadratischen Term in Gleichung 4.20 repräsentiert. Bei vollständig geöffneter Drosselklappe ist der Druckabfall sehr klein, das anliegende Druckverhältnis $p_r(t)$ also nahe 1. In diesem Fall wird, wie in Abbildung 4.10 ersichtlich eine exponentielle Funktion derart angepasst, dass sie die Messpunkte bestmöglich beschreibt. Als zusätzliche Größen werden neben den orangen Rohrverbindungen die Drosselklappenstellung $x_{air}(t)$, der Massenfluss in den Motoreinlass $W_{ci}(t)$ und die Temperatur im Einlass $T_i(t)$ benötigt.

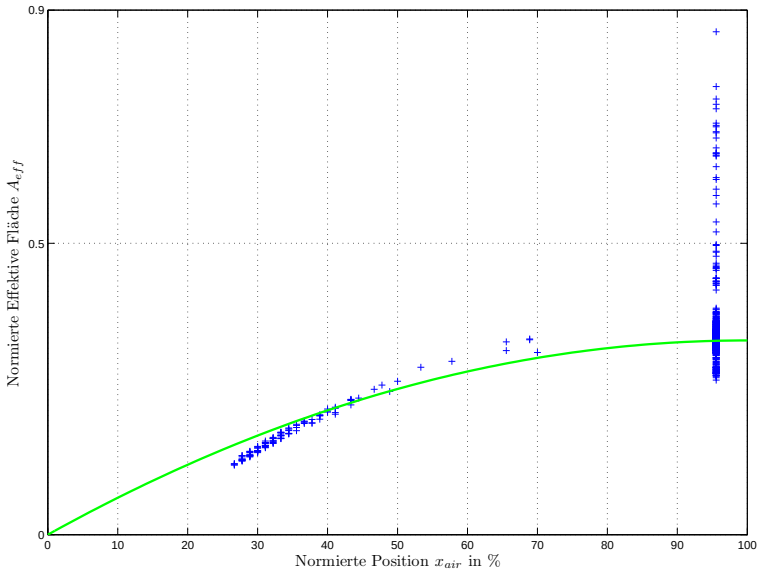

Abbildung 4.9.: Kennlinie der Drosselklappe in Abhängigkeit der Klappenposition

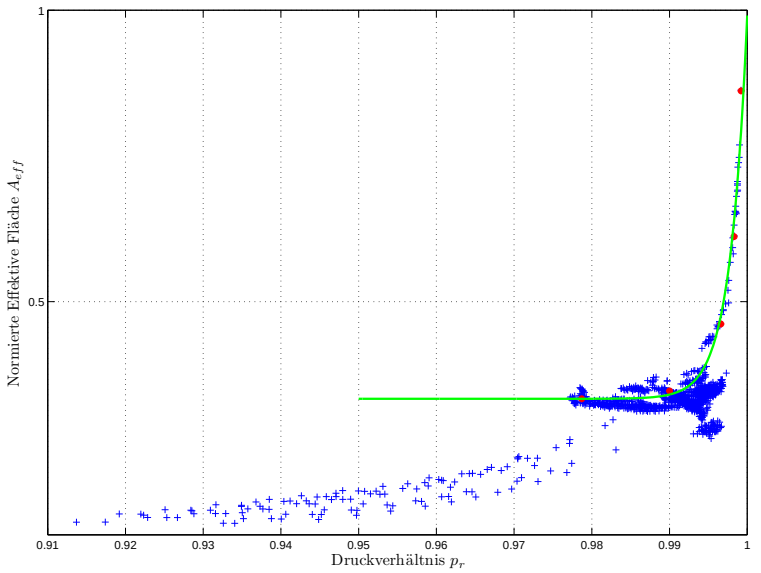

Abbildung 4.10.: Kennlinie der Drosselklappe in Abhängigkeit des Druckverhältnisses

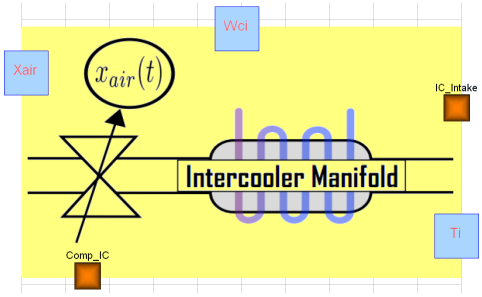

Abbildung 4.11.: Darstellung des Ladeluftkühlers in Dymola

4.8. Turboaufladung

Obwohl man die beiden Teile der Turboladerstufe, Aufladung und Komprimierung der Luft, trennen könnte, wurde dies nicht getan, da diese beiden Teile über die Dynamik der Drehzahl der Welle $N_t(t)$ eng miteinander gekoppelt sind. Deshalb gibt es nur eine Komponente in *Modelica*, die beide Funktionalitäten beinhaltet.

Kompressor

Die beschreibenden Gleichungen für den Kompressor leiten sich im Wesentlichen von jenen aus Kapitel 3 ab und werden zusammengefasst in den Gleichungen 4.21 und 4.22 angeführt:

$$T_c(t) = T_a + \frac{T_a}{\eta_c(t)} \left(\left(\frac{p_{ic}(t)}{p_a} \right)^{\frac{\gamma-1}{\gamma}} - 1 \right) \tag{4.21}$$

$$P_c(t) = W_c(t) \cdot c_p \cdot \frac{T_a}{\eta_c(t)} \left(\left(\frac{p_{ic}(t)}{p_a} \right)^{\frac{\gamma-1}{\gamma}} - 1 \right) \tag{4.22}$$

Maßgeblich für das Verhalten des Kompressors ist das Verhältnis des Drucks der Luft nach dem Kompressor $p_{ic}(t)$ zum Umgebungsdruck p_a am Eingang. Am Kompressoreingang herrschen, die für das Modell konstanten Werte Außentemperatur T_a und Umgebungsdruck p_a vor. In Gleichung 4.22 sind neben den bekannten Größen noch die beiden Unbekannten Kompressormassenfluss $W_c(t)$ und Kompressoreffizienz $\eta_c(t)$ enthalten. Um die Komplexität des erstellten Modells des Luftpfades in Grenzen zu halten, wurden an dieser Stelle für die beiden Größen Kennflächen erstellt. Es existieren nur wenige Modelle, die den Zusammenhang der beiden zuvor erwähnten Größen durch physikalische Beziehungen beschreiben, deshalb ist die Verwendung von Kennflächen die Standardvorgehensweise. Diese sind in den Abbildungen 4.12 und 4.13 dargestellt. Als Grundlage zur Erstellung der Kennlinien dienen einerseits mitgelieferte Herstellerdaten anderseits aus den Messwerten berechnete Punkte. Die Herstellerdaten werden durch die schwarzen Punkte repräsentiert. Die farbigen Kreuze stellen die aus den Messwerten berechneten Punkte dar. Die Kompressoreffizienz ist von dem Massenfluss $W_c(t)$ durch den Kompressor und der Turbinendrehzahl $N_t(t)$ abhängig. Die Kennfläche besitzt einen Grat, an dem die Effizienz ein Maximum aufweist. Dieser Grat ist als grüne Linie dargestellt. Der Kompressormassenfluss ist von der Turbinendrehzahl einerseits und von dem anliegenden Druckverhältnis anderseits abhängig. Die Modellierung des Kompressormassenflusses stellte eine Herausforderung dar, da die Kennfläche plötzlich sehr flach wird. Mit steigendem Druckverhältnis an dem Kompressor wird bei gleicher Turbinendrehzahl immer weniger Luftmasse

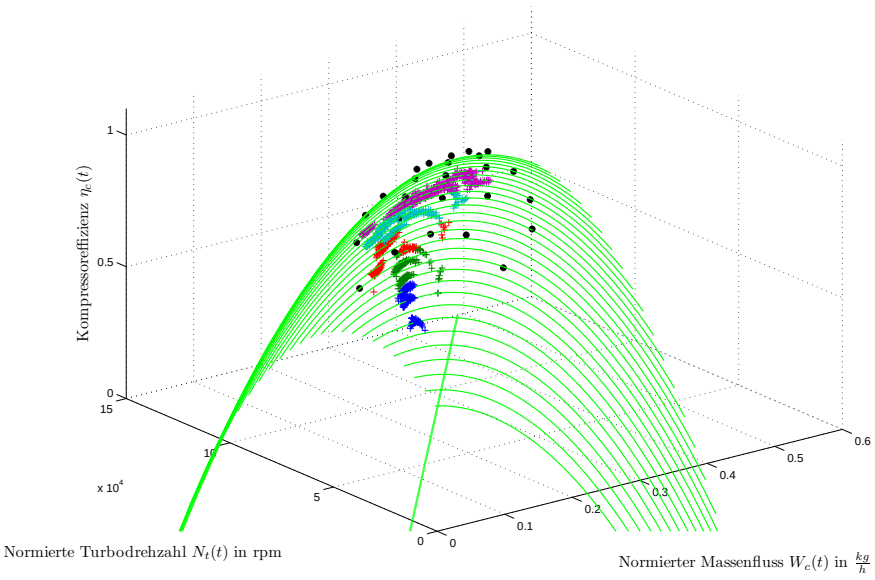

Abbildung 4.12.: Kennfläche der Kompressoreffizienz

transportiert. Ab einem gewissen Druckverhältnis sinkt die geförderte Luftmasse plötzlich auf 0 ab. Dies ist auf einen Strömungsabriss an den Kompressorschaufeln zurückzuführen, wie sie in [Guzzella und Onder, 2004] beschrieben wird. Zu beachten ist, dass die Daten des Herstellers korrigierte Werte darstellen. Das heisst, die beiden unabhängige Größen $W_c(t)$ und $N_t(t)$ werden um die Werte in Tabelle 4.4 nach den Gleichungen 4.23 korrigiert:

$$\begin{aligned} W_{c,corr}(t) &= W_c(t) \cdot \sqrt{\frac{T_a}{T_{c,ref}}} \cdot \frac{p_{c,ref}}{p_a} \\ N_{t,corr}(t) &= N_t(t) \cdot \sqrt{\frac{T_{c,ref}}{T_a}} \end{aligned} \tag{4.23}$$

Um die aus den Messwerten berechneten Punkte korrekt darzustellen, müssen diese ebenfalls normiert werden.

Benennung	Wert	Einheit	Erklärung
$T_{c,ref}$	298	K	Referenztemperatur Kompressor
$p_{c,ref}$	10^5	Pa	Referenzdruck Kompressor
$T_{t,ref}$	923	K	Referenztemperatur Turbolader

Tabelle 4.4.: Referenzwerte Turbo

Turbine

Für die Modellierung der Aufladung des Turbos werden ebenfalls die Gleichungen aus Kapitel 3 verwendet. Setzt man die in der Aufladung verwendeten Größen ein, erhält man folgende Beziehungen für die Temperatur nach der Turbine $T_t(t)$ und die von der Turbine aufgenommenen

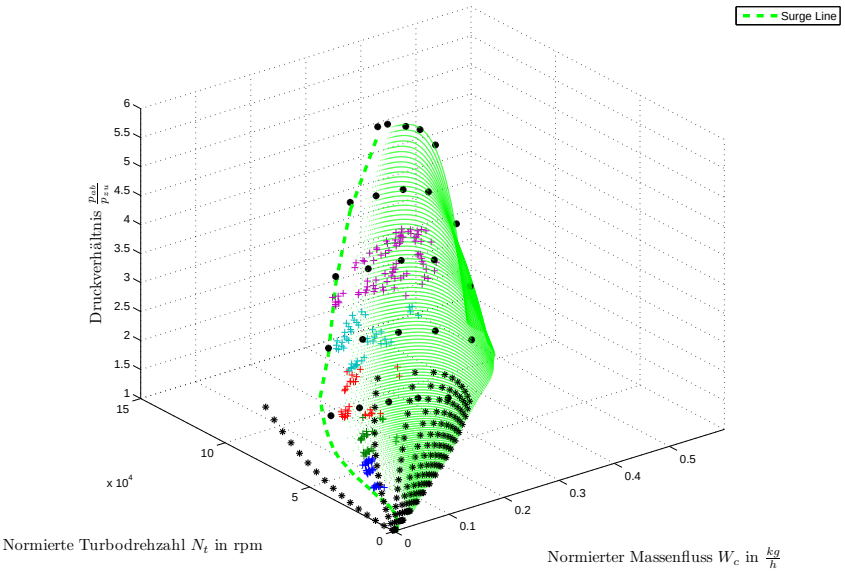

Abbildung 4.13.: Kennfläche des Kompressormassenflusses

Leistung $P_t(t)$:

$$P_t(t) = W_{xt}(t) \cdot c_p \cdot \eta_t(t) \cdot T_x(t) \left(1 - \left(\frac{p_t(t)}{p_x(t)}\right)^{\frac{\gamma-1}{\gamma}}\right) \tag{4.24}$$

$$T_t(t) = T_x(t) + \eta_t(t) \cdot T_x(t) \left(1 - \left(\frac{p_t(t)}{p_x(t)}\right)^{\frac{\gamma-1}{\gamma}}\right) \tag{4.25}$$

Der Druck nach dem Turbolader $p_t(t)$ ist nicht der Umgebungsdruck, da eine eventuell vorhandene Abgasnachbehandlung einen Gegendruck erzeugt. Dies kann durch einen erhöhten Druck $p_t(t)$ berücksichtigt werden. Für die Größen vor dem Turbolader wird die Temperatur des Auslasses $T_x(t)$ und der Druck im Auslass $p_x(t)$ eingesetzt. Wie bereits für den Kompressor werden auch für die Turboaufladung die beiden Größen Massenfluss $W_{xt}(t)$ und Effizienz $\eta_t(t)$ des Turbos über Kennflächen dargestellt. Die Korrektur der Herstellerangaben gestaltet sich jedoch etwas anders. Dies ist in den Gleichungen 4.26 beschrieben:

$$\begin{aligned} W_{xt,corr}(t) &= W_{xt}(t) \cdot \frac{\sqrt{T_x} \cdot 10^5}{p_x} \\ N_{t,corr} &= N_t \cdot \sqrt{\frac{T_{t,ref}}{T_x}} \end{aligned} \tag{4.26}$$

Die Erstellung der Turbineneffizienz wies gewisse Schwierigkeiten auf, da diese von mehreren Größen abhängig ist. Bei niedrigen Drehzahlen ist ein nicht zu vernachlässigender Wärmetransfer zu verzeichnen, wie in [Moraal und Kolmanovsky, 1999] hingewiesen wird. Deshalb teilt sich die Effizienz in 2 Teile auf, der aerodynamischen Effizienz $\eta_{t,aero}$ und jener, die aus dem Wärmetransfer resultiert $\eta_{t,heat}$. Die Effizienz aus dem Wärmetransfer (Abb. 4.14) wird durch folgende Beziehung angenähert:

$$\eta_{t,heat} = 6 \cdot e^{-2.2\frac{p_x(t)}{p_t(t)}} - 0.0022 \cdot x_{vgt}(t) + 0.27 \tag{4.27}$$

Die aerodynamische Effizienz, (Abb. 4.15) wird durch einen Ansatz über die Schaufelge-

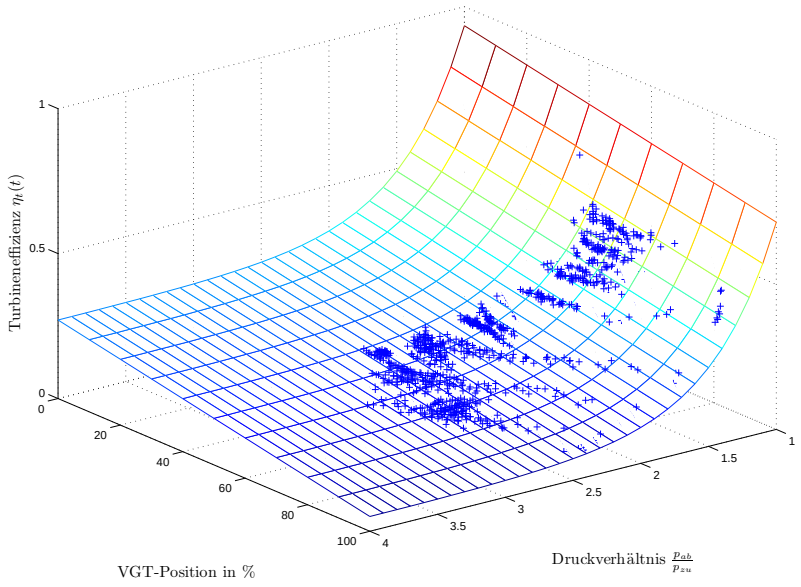

Abbildung 4.14.: Kennfläche der Turbineneffizienz aufgrund des Wärmetransfers

schwindigkeit (engl. „blade speed ratio“) $c_u(t)$ ermittelt [Jung, 2003],

$$c_u(t) = \frac{\pi \cdot d_t \cdot N_t(t)}{60\sqrt{2c_p \cdot T_x(t) \cdot \left(1 - \frac{p_t(t)}{p_x(t)}^{\frac{\gamma-1}{\gamma}}\right)}}, \tag{4.28}$$

wobei d_t den Durchmesser der Turbinenschaufel bezeichnet. Durch diesen Ansatz kann man die Anzahl der Größen, von denen die aerodynamische Effizienz abhängt, reduzieren. Nun kann ein quadratischer Ansatz in Abhängigkeit dieser Größe verwendet werden:

$$\eta_{t,aero}(t) = a \cdot (c_u(t) - c_{u,opt}(t))^2 + \eta_{t,max}(t) \tag{4.29}$$

Der Koeffizient $c_{u,opt}$ ist an dieser Stelle nur mehr von der korrigierten Turbinendrehzahl $N_{t,corr}$ abhängig und wird geeignet parametrisiert:

$$c_{u,opt}(t) = 2.17 \cdot 10^{-6} \cdot N_{t,corr}(t) + 0.348 \tag{4.30}$$

Im Vergleich zur Turbineneffizienz lässt sich die Kennfläche für den Turbinenmassenfluss $W_{xt}(t)$ einfach erstellen. Dazu wird zunächst die effektive Fläche der Turbinenschaufeln in Abhängigkeit des anliegenden Druckverhältnisses $p_r(t)$ und der Position der Turbinenschaufeln $x_{vgt}(t)$ modelliert:

$$A_{eff}(t) = 2.05 \cdot p_r^3 - 24.76 \cdot p_r^2 + 93.54 \cdot p_r + 0.0031 \cdot x_{vgt}^2 + 0.51 \cdot x_{vgt} - 80.19 \tag{4.31}$$

Anschließend kann durch Gleichung 3.6 der Turbinenmassenfluss berechnet werden. Diese Kennfläche ist in Abbildung 4.16 dargestellt.

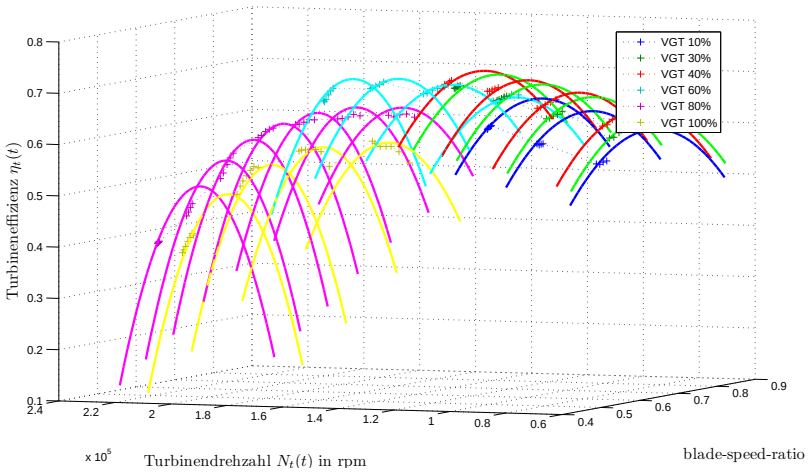

Abbildung 4.15.: Kennfläche der aerodynamischen Turbineneffizienz

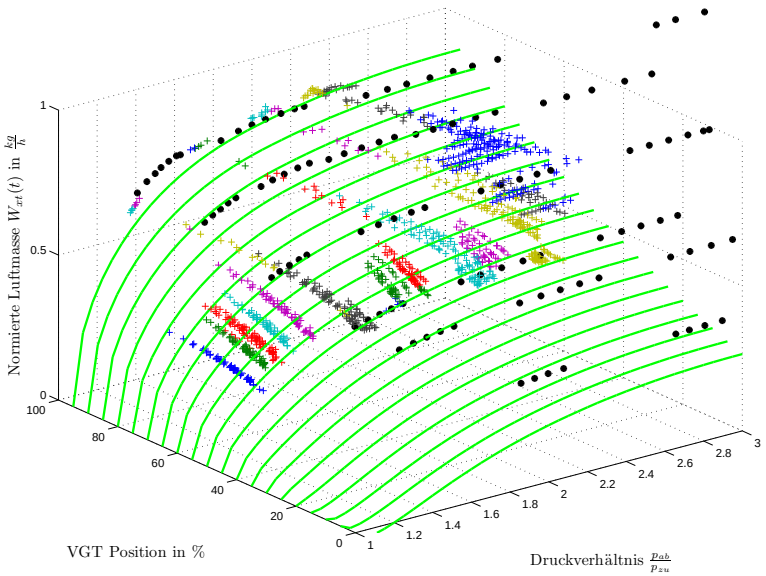

Abbildung 4.16.: Kennfläche des Turbinenmassenflusses

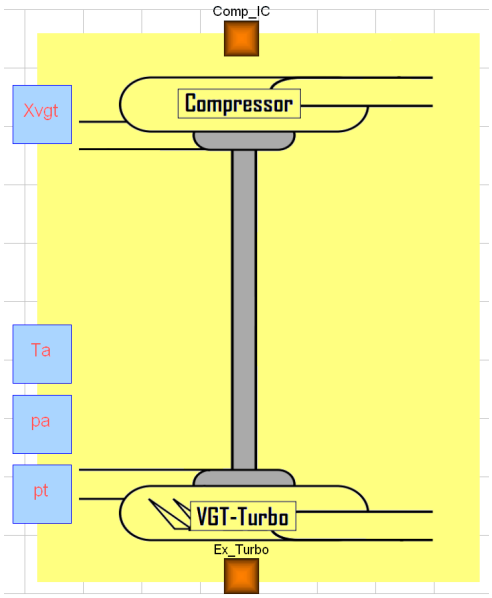

Abbildung 4.17.: Darstellung des Turboladers in Dymola

4.9. Initialisierung

Obwohl es der Initialisierungsroutine in Dymola gelingt sämtliche Größen des Modells konsistent zu initialisieren, ist es logischerweise notwendig, zumindest alle Zustandsgrößen am Beginn der Simulation vorzugeben. Dafür wird der Abschnitt *initial algorithm* in Modelica verwendet. Es wurde als sinnvoll erachtet, das System anhand eines Messpunktes zu initialisieren. Da sich auch der reale Motor am Prüfstand nach dem Startvorgang im Leerlauf befindet, wurde ein Messwert gewählt, der diesem Zustand am ehesten entspricht. Die Werte für die Zustandsgrößen sind in Tabelle 4.5 dargestellt.

Zustand	Wert	Einheit
p_i	106233	Pa
m_i	0.0346	kg
F_i	0.579	1
p_x	116267	Pa
m_x	0.00221	kg
F_x	0.999	1
p_{ic}	106247	Pa
m_{ic}	0.0359	kg
N_t	34996	rpm

Tabelle 4.5.: Werte der Zustandsvariablen zum Startzeitpunkt

4.10. Gesamtmodell

Die zuvor beschriebenen Teilmodelle werden in Modelica zu einem neuen Modell zusammengeführt. Diese neue Einheit verfügt nun über die gesamte zu modellierende Funktionalität. Der

Aufbau des Luftpfades ist in Abbildung 4.18 dargestellt. Die selbst erstellten Schnittstellen, welche thermodynamische Verbindungen darstellen, sind orange eingefärbt. Jede Linie zwischen diesen Schnittstellen stellt am realen Motor eine Rohrverbindung dar. Die restlichen Linien repräsentieren ein einzelnes Signal, welches Größen zweier Komponenten verbindet. In Abbildung 4.18 sind 8 Eingangsgrößen für das System vorhanden. Jedoch sind nicht alle davon Stellgrößen des Modells. Die Umgebungstemperatur T_a, der Umgebungsdruck p_a und der Druck p_t im Abgasstrang nach der Turboaufladung stellen die Betriebsumgebung dar und können als Störgrößen betrachtet werden.
Durch die Motordrehzahl N_e und die Kraftstoffmasse m_f wird im Wesentlichen der Betriebs-

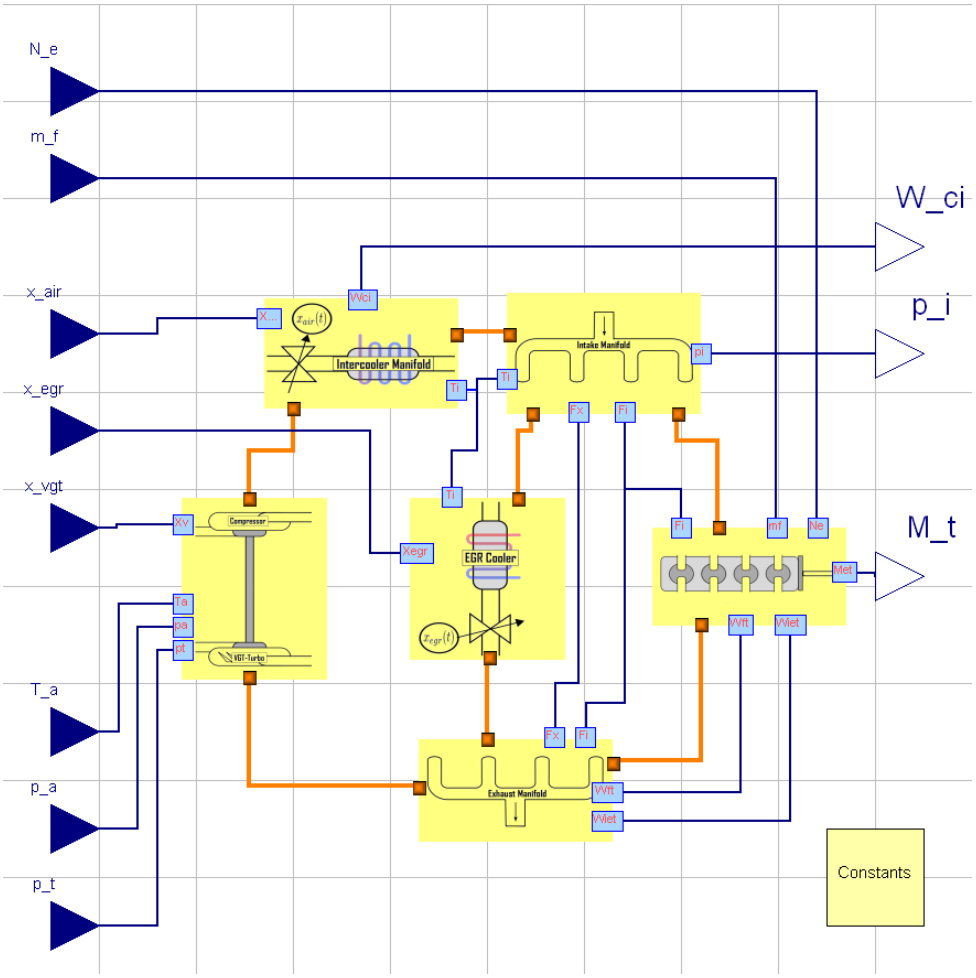

Abbildung 4.18.: Darstellung des gesamten Luftpfades in Dymola

punkt des Motors von außen vorgegeben. Somit verbleiben nur noch die 3 Stellgrößen Drosselklappe $x_{air}(t)$, Abgsrückführklappe $x_{egr}(t)$ und die Stellung der Turbinengeometrie $x_{vgt}(t)$, mit denen das Verhalten des Luftpfades geregelt werden kann.
Die Ausgangsgröße M_t stellt das vom Motor erzeugte Drehmoment dar. Es ist für die anschließende Simulation wesentlich. Demgegenüber stehen die Frischluftmasse W_{ci} und der Ladedruck p_i, welche als weitere Regelgrößen verwendet werden können.

Kapitel 5

Verifikation des Modells

Damit das in Kapitel 4 erstellte Modell als Simulationsgrundlage für den Reglerentwurf verwendet werden kann, muss dessen Übereinstimmung mit dem realen Motor nachgewiesen werden. Dazu werden die Messwerte, welche am Prüfstand ermittelt wurden, herangezogen. Der Motor wurde am Prüfstand durch die Motordrehzahl und die eingespritzte Kraftstoffmenge in einen gewünschten stationären Betriebszustand gebracht, um anschließend sämtliche Messwerte zu speichern. Dies geschah an rund 1100 Betriebspunkten.

5.1. Testumgebung in Dymola

Um dem Modell des Luftpfades die Motordrehzahl vorgeben zu können, muss diese aus dem erzeugten Motordrehmoment ermittelt werden. Dazu ist die Implementierung des Drallsatzes notwendig:

$$\ddot{\varphi}(t) = \frac{1}{J_e} \cdot \sum M(t) \tag{5.1}$$

Durch diesen mathematischen Zusammenhang wird die Winkelbeschleunigung $\ddot{\varphi}(t)$, welche sich aufgrund des konstanten Massenträgheitsmoments J_e unter Einwirkung der Summe der Momente $M(t)$ ergibt, beschrieben. Das Massenträgheitsmoment $J_e = 1.6\ kg \cdot m^2$ stellt die Trägheit der rotierenden Teile des Motors dar. Um die Simulationswerte für den Vergleich mit den Messdaten zu erhalten, ist es notwendig, den Motor in einer gewünschten Drehzahl $N_e(t)$ zu betreiben. Um diese Vorgabe erfüllen zu können wird ein Bremsmoment $M_b(t)$ vorgegeben. Durch eine Subtraktion dieses Bremsmomentes vom erzeugten Motordrehmoment $M_e(t)$ besteht die Möglichkeit, die Motorgeschwindigkeit zu beeinflussen. Dies führt zur Gleichung 5.2, welche als zusätzliche Komponente in Modelica implementiert wurde:

$$\frac{dN_e(t)}{dt} = \left(\frac{30}{\pi \cdot J_e}\right) \cdot (M_e(t) - M_b(t)) \tag{5.2}$$

Diese Komponente besitzt nur einen Eingang, die Differenz der beiden zuvor beschriebenen Drehmomente, und ermittelt die aktuelle Motordrehzahl, damit diese für die Simulation an den Luftpfad weitergegeben werden kann.

5.2. Simulation

Zur Verifikation dienten die rund 1100 stationären Betriebspunkte, welche als Messwerte zur Verfügung stehen. Dazu wurden sämtliche Eingangsgrößen aus den Messwerten ausgelesen und Dymola zur Verfügung gestellt. Nach dem Aufschalten der Eingangsgrößen auf das Luftpfadsystem wird eine Einschwingzeit von $5s$ gewartet, bevor die eingeschwungenen Werte der Zustände in eine Tabelle gespeichert werden. Daraus resultiert eine Simulationszeit von $5500s$ zum Erfassen der korrespondierenden simulierten Werte des erstellten Modells. Anschließend wird ein Vergleich des transienten Verhaltens der beiden Modelle in Matlab und Dymola durchgeführt. Zur Simulation wurde als Löser der standardmäßig eingestellte DASSL verwendet, da mit diesem positive Erfahrungen gemacht wurden. Die Toleranz wurde ebenfalls mit 0.0001 auf dem Standardwert belassen. Da die Simulationsdauer erheblich erhöht wurde, war es notwendig auch die Zeitschritte auf 50000 stark zu erhöhen, um zufriedenstellende Ergebnisse zu erhalten.

5.2.1. Statischer Vergleich des Modelica Modells mit den Messwerten

Bei der an dieser Stelle beschriebenen Validierung handelt es sich ausschließlich um einen statischen Vergleich des realen Motors mit dem neu erstellten Modell. Sowohl die Messwerte vom Prüfstand als auch die Simulationswerte aus dem Dymola Modell stammen von eingeschwungenen Zuständen. In Abbildung 5.1 werden ausgewählte Größen aus dem Gesamtmodell den entsprechenden Messwerten gegenübergestellt. Bei der Wahl der zu vergleichenden Größen liegt das Hauptaugenmerk auf die in den großen Volumina vorherrschenden Drücke und den Massenflüssen zwischen den Komponenten. Es wird der Druck am Motoreinlass $p_i(t)$ und jener am Motorauslass $p_x(t)$ dargestellt. Vor allem der Druck am Einlass, welcher auch Ladedruck genannt wird, ist für die nachfolgende Betrachtung von großer Bedeutung. Weiters wird der Massenfluss vom Einlass in den Verbrennungsraum $W_{ie}(t)$, jener vom Auslass in die Turbine $W_{xt}(t)$ und jener der aus dem Kompressor kommt $W_c(t)$, als interessant erachtet. Schließlich wird noch die Drehzahl der Turboladerstufe $N_t(t)$ betrachtet, da diese den Zustand der Turboladerstufe gut wiedergibt. In den angeführten Diagrammen sind jeweils auf der Abszisse die Simulationswerte und auf der Ordinate die zum Betriebspunkt zugehörigen Messwerte aufgetragen. Idealerweise sollten alle Punkte auf der blau eingezeichneten 45^o Linie liegen. Da bei der Modellierung einige Abstraktionen vorgenommen wurden, ist es einsichtig, dass das entworfene Modell nicht vollständig mit den Messdaten übereinstimmt. Die Ursachen für die Abweichungen von den Messwerten sind sehr unterschiedlich. Einerseits treten, wie bei jeder Messung von physikalischen Größen, Messungenauigkeiten auf. Solche Messfehler, wie zum Beispiel aus dem Luftmassensensor, können nur in Grenzen gehalten werden, aber nicht vollständig vermieden werden. Eine weitere Ursache für die auftretenden Abweichungen liegt mit Sicherheit in den Approximationen der zahlreichen Kennlinien und Kennflächen. Obwohl die Kennlinien optimal im Sinne der kleinsten Fehlerquadrate angepasst wurden, sind nicht unerhebliche Abweichungen von den Messwerten vorhanden. Schlussendlich treten bei einem realen Motor auch betriebspunktabhängige Schwankungen der Parameter auf, die in der Modellierung nicht berücksichtigt wurden. Da zwischen den Messwerten, die am Prüfstand aufgezeichnet wurden, nur eine begrenzte Zeit abgewartet werden kann, tritt eine Beeinflussung eines Messpunktes durch den vorangegangenen Messpunkt ein. Diese Beeinflussung kann über Motorteile erfolgen, die eine höhere Temperatur aufweisen, als sie durch den aktuellen Betriebspunkt hätten.

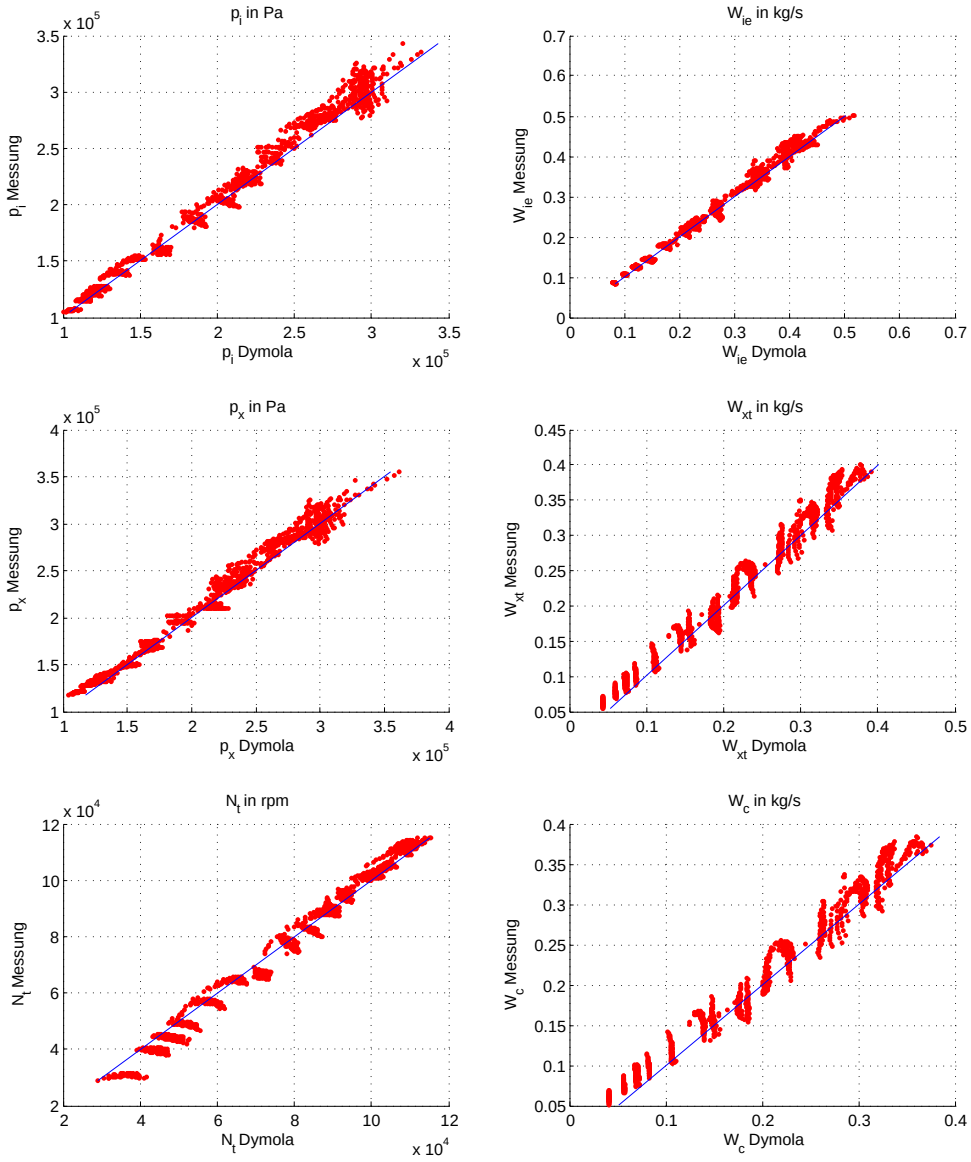

Abbildung 5.1.: Vergleich repräsentativer Größen der Simulation mit den Messwerten

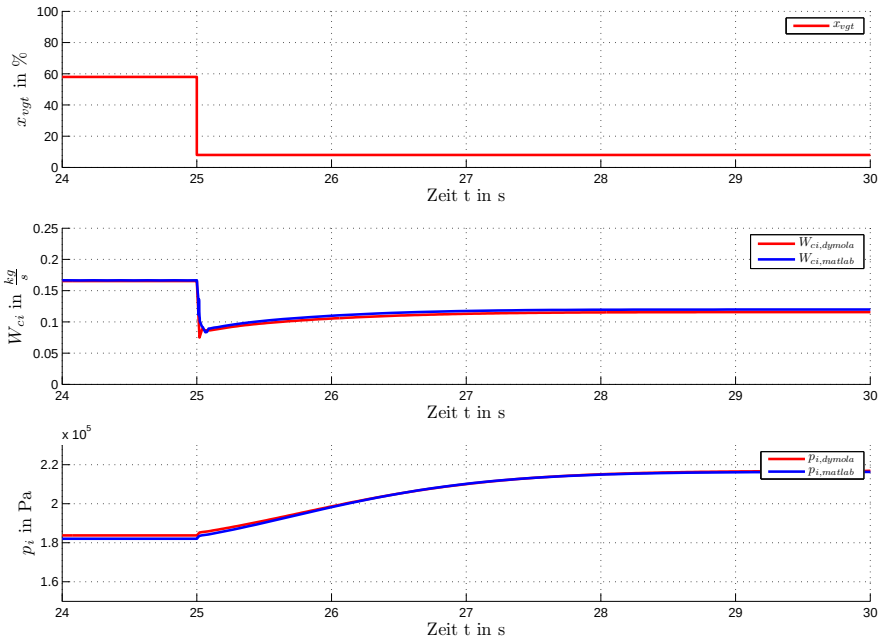

Abbildung 5.2.: Vergleich des Matlab und des Dymola Modells bei sprungförmiger Anregung

Das vorliegende Ergebnis kann jedoch als ausreichend genau für den Reglerentwurf und die darauf folgende Simulation betrachtet werden.

5.2.2. Vergleich des transienten Verhaltens

Für die Validierung der Dynamiken aus dem realen Motor am Prüfstand wären neben den vorhandenen Daten aus eingeschwungenen Betriebspunkten auch zeitabhängige Messwerte über Betriebspunktwechsel notwendig. Da diese Messwerte jedoch nicht zur Verfügung standen, stellte sich die Validierung des transienten Verhaltens als schwierig heraus. Aus den Gleichungen der Modellierung der Volumina sticht der Einfluss der drei Volumina, jenes am Einlass V_i, jenes am Auslass V_x und jenes des Ladeluftkühlers V_{ic} auf die Dynamik des Luftpfads heraus. Diese Größen bestimmen maßgeblich das dynamische Verhalten des modellierten Motors und wurden an die realen Verhältnisse am Motor angenähert. In Abbildung 5.2 wird das Verhalten der Modelle von Dymola und Matlab auf einen Sprung der Stellung der Turbinenschaufel dargestellt. Die Motordrehzahl beträgt $1400\ U/min$, und die zugeführte Kraftstoffmenge $6.06 \cdot 10^{-4}\ kg$ pro Zyklus. Die Eingangsgröße $x_{vgt}(t)$ springt bei einer Simulationszeit $25\ s$ um 50% nach unten. In den beiden unteren Diagrammen in Abbildung 5.2 ist einerseits der Luftmassenstrom vom Ladeluftkühler in den Einlass $W_{ci}(t)$, andererseits der Ladedruck $p_i(t)$ dargestellt. Es werden jeweils die Verläufe der zuvor beschriebenen Größen aus dem Matlab Modell und aus dem Dymola Modell angezeigt. Es ist eine gute Übereinstimmmung der beiden Modelle zu erkennen. Jedoch sind die Verläufe nicht identisch. Besonders kurz nach dem Sprung der Eingangsgröße ist eine größere Abweichung zu erkennen. Dies kann auf die unterschiedlichen Lösungsmethoden der beiden Programme zurückgeführt werden. Die jeweiligen

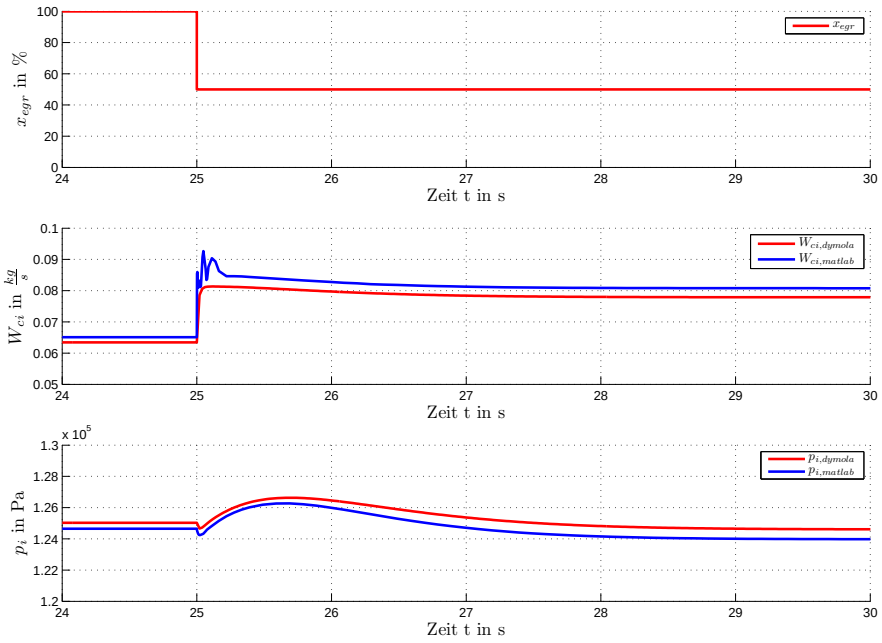

Abbildung 5.3.: Vergleich des Matlab und des Dymola Modells bei sprungförmiger Anregung

Endwerte stimmen jedoch gut überein.

An dieser Stelle wird ein weiterer transienter Verlauf bestimmter Modellgrößen aufgrund einer Änderung einer Eingangsgröße dargestellt. Die Stellung der Klappe an der Abgasrückführung $x_{egr}(t)$ springt nach $25\ s$ um 50% nach unten. Der Motor dreht sich vor und nach dem Sprung mit 1000 Umdrehungen in der Minute. Außerdem wird die Kraftstroffmenge $4.54 \cdot 10^{-4}\ kg$ pro Zyklus zugeführt. Die beiden unteren Diagramme stellen wieder dieselben Größen dar, wie bereits in Abbildung 5.2. Zu erwähnen ist, dass nach einem Sprung der Abgasrückführklappe die beiden Modelle nicht mehr in so hohem Maße übereinstimmen, wie dies nach dem Sprung der Turbinenschaufelstellung der Fall war. Außerdem ist beim Verlauf des Luftmassenstroms vom Kompressor in den Einlass zu erkennen, dass das Matlab Modell starke Schwankungen aufweist. Dies lässt vermuten, dass der Löser nahe einer Instabilität ist. Der Verlauf aus dem Dymola Modell weist hingegen keine derartige Schwingungen auf. Die beiden Verläufe aus dem Sprung der Turbinenschaufelstellung führen zu einer größeren stationären Abweichung. Das kann auf die relativ komplexen hinterlegten Kennlinien der Komponenten Kompressor und Turbine zurückgeführt werden, da die beiden unterschiedlichen Löser zu unterschiedlichen Gleichgewichtszuständen kommen.

Kapitel 6

Aufbau des Regelkreises

An dieser Stelle wird schrittweise der Weg zum vollständigen Regelkreis erläutert. Dabei wird besonders darauf geachtet, dass die Bedingungen für das Modell mit jenen des realen Motors am Prüfstand übereinstimmen. Dadurch soll gewährleistet werden, dass der Regler auch am Motor angewendet werden kann. Aus diesem Grund wird die Implementierung des Prüfstandes und des Reglers in Matlab/Simulink durchgeführt. Dadurch sind die beiden Teilsysteme, bestehend aus Regler und Luftpfadmodell, getrennt voneinander verwendbar.

6.1. Prüfstandssimulation in Simulink

Um die Bedingungen, welche am Prüfstand vorliegen herzustellen, ist es notwendig, den Zustand des Luftpfadmodells in geeigneter Weise vorzugeben. Die Wahl der Größen, welche den Betriebszustand vorgeben, erfordert die Betrachtung des gesamten Motors. Hieraus ergibt sich als erste logische Größe die Motordrehzahl N_e. Aus Sicht des Luftpfades stellt die Drehzahl eine Eingangsgröße dar, die theoretisch unabhängig vom Modell vorgegeben werden kann. In der Realität ergibt sich die Drehzahl jedoch aus dem Antriebs- und dem Bremsmoment des Fahrzeugs. Deshalb wird die Differenz dieser Momente mittels des Drallsatzes benutzt, um die Drehzahl zu bestimmen. Die Drehzahl stellt nun keine Eingangsgröße des gesamten Regelkreises dar, sie muss lediglich einen Anfangswert besitzen, damit der Startzustand des Motors festgelegt werden kann.
Für die Festlegung des Motorstartzustandes fehlt zusätzlich noch eine weitere Größe. Die Brennstoffmasse $m_f(t)$ steht in einem direkt proportionalem Zusammenhang mit dem erzeugtem Motordrehmoment $M_e(t)$, [Rueckert, 2008, S.68]. Deshalb wird das Motordrehmoment als zweite Größe gewählt, um den Zustand zu definieren.

6.1.1. Regelkreise für das Motordrehmoment und die Motordrehzahl

In einer Simulation auf dem Prüfstand wird in der Leerlaufdrehzahl gestartet und dem Gesamtsystem, bestehend aus Verbrennungsmotor und Prüfstand, das erforderliche Moment und die gewünschte Motordrehzahl vorgegeben. Diese beiden Größen können durch zwei unabhängige PI-Regler eingeregelt werden. Die Realisierung in Simulink ist in Abbildung 6.1 dargestellt.
Der untere Regelkreis für das Drehmoment wird an dieser Stelle näher erläutert. Um ein besse-

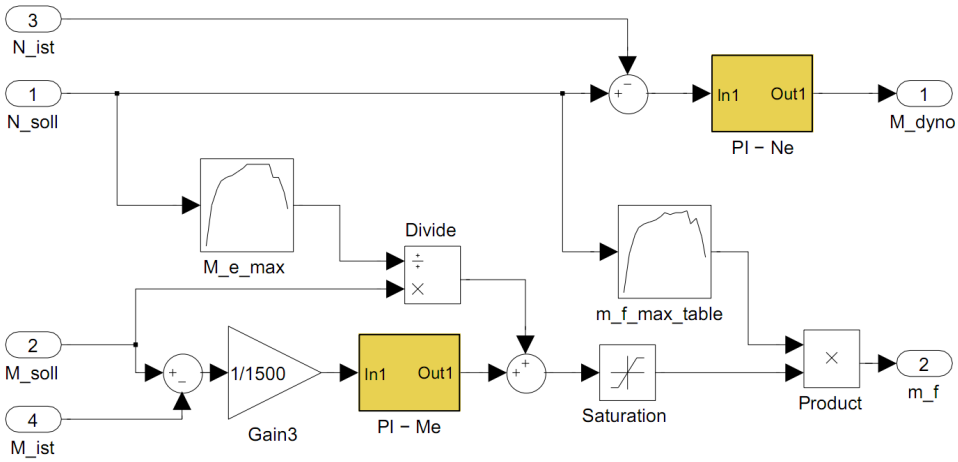

Abbildung 6.1.: PI-Regelkreise für Drehzahl und Drehmoment

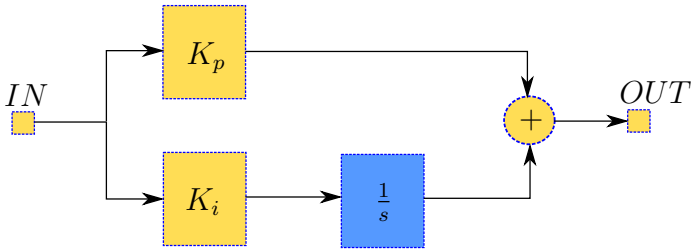

Abbildung 6.2.: Aufbau der PI-Regler

res Regelverhalten zu erreichen, wird eine Vorsteuerung verwendet, welche aus dem *M_e_max* Block und einer Division besteht. Dieser Block enthält die Kennlinie des maximal zulässigen Motordrehmoments und ist im unteren Diagramm in Abbildung 6.3 dargestellt. Diese Vorsteuerung übernimmt den Hauptteil der Stellarbeit. Das Signal nach der Division repräsentiert den Anteil des aktuell geforderten Drehmoments $M_{e,soll}$ zu dem maximal möglichen Drehmoment. Dieser Wert sollte immer zwischen 0 und 1 liegen, ansonsten wird ein unmöglicher Betriebszustand gefordert. Lediglich die noch bestehende Regelabweichung wird als Eingangsgröße für den PI-Regler verwendet. Damit die beiden Anteile an der Stellgröße in der selben Größenordnung sind wird der Teil des Reglers mit $\frac{1}{1500}$ normiert. Die Anteile der Stellgröße werden addiert, und danach auf 1 begrenzt, damit keinesfalls ein größerer Wert für die Kraftstoffmenge vorgegeben wird als $m_{f,max}$. Die Stellgröße kann deswegen auf 1 begrenzt werden, da nur der Anteil an der maximal möglichen Kraftstoffmenge geregelt wird.
Vor allem bei Dieselmotoren können im niedrigen Drehzahlbereich sehr hohe Motordrehmomente $M_e(t)$ auftreten, welche zur mechanischen Zerstörung mancher Bauteile führen. Um dies zu verhindern, ist eine Drehmomentbegrenzung erforderlich. Diese wird durch eine Begrenzung der Kraftstoffmenge $m_f(t)$ realisiert. Die approximierte Kennlinie aus den Messdaten ist im oberen Diagramm der Abbildung 6.3 dargestellt. Die zuvor beschriebene begrenzte Stellgröße wird mit der Begrenzungskennlinie multipliziert. Dadurch kann eine Überschreitung der Kraftstoffbegrenzung ausgeschlossen werden. Die PI-Regler wurden aus Standardelementen aus der Simulinkbibliothek nach Abbildung 6.2 aufgebaut. Die Werte der beiden verwendeten PI-Regler sind aus Tabelle 6.1 zu entnehmen: Diese Werte wurden experimentell ermittelt.

In den Abbildungen 6.4 und 6.5 sind die Ergebnisse der Regelungen dargestellt. Es befinden sich im jeweils oberen Plot die Sollwerte mit der Regelgröße, im jeweils unteren hingegen die

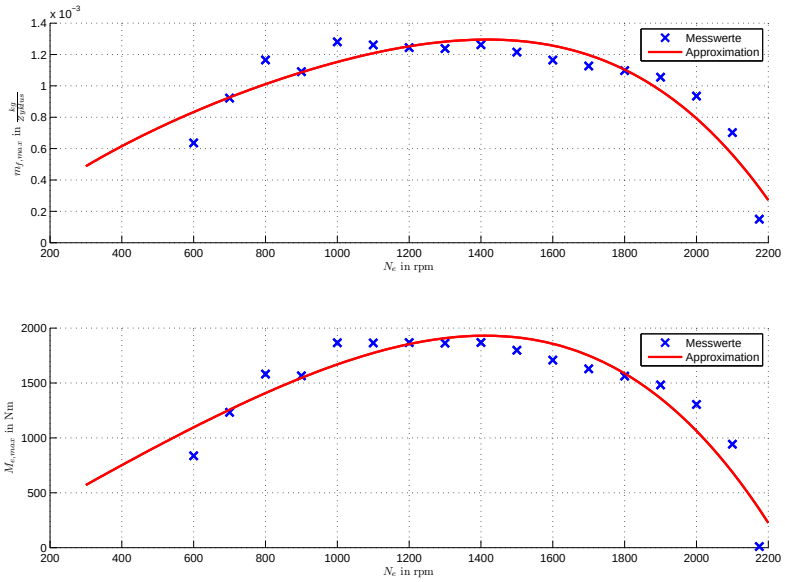

Abbildung 6.3.: Drehzahlabhängige Drehmoment- und Kraftstoffbegrenzung im PI-Regelkreis

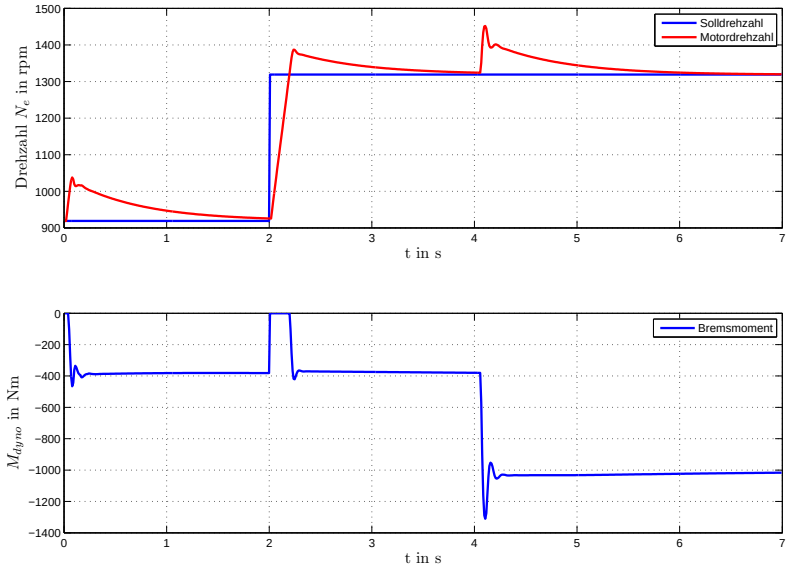

Abbildung 6.4.: Drehzahlregelung als Prüfstandersatz

Regler	K_p	K_i
$PI - N_e$	7	10
$PI - M_e$	0.2	0.3

Tabelle 6.1.: Werte für PI-Regler

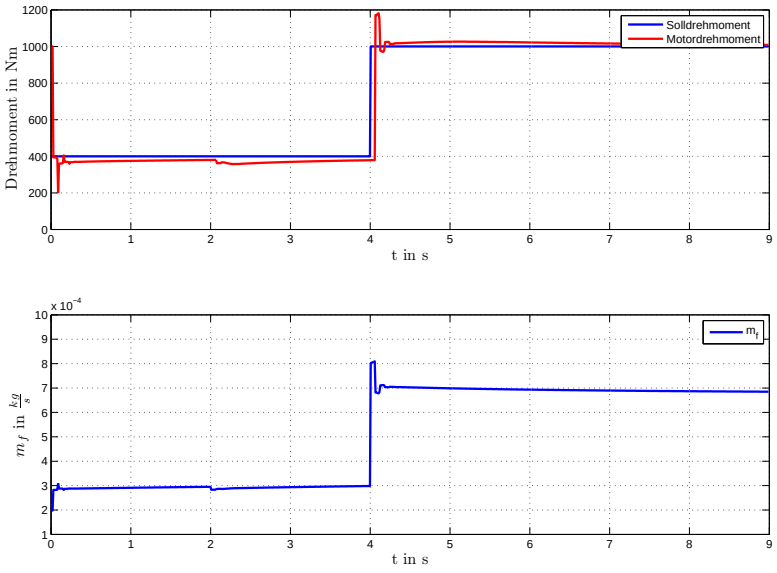

Abbildung 6.5.: Drehmomentregelung als Prüfstandersatz

zugehörigen Stellgrößen der beiden Regelkreise. Nach dem Start der Simulation pendeln sich beide Werte ein. Das Drehmoment weist eine hohe Dynamik auf und folgt der Sollgröße rasch, die Drehzahl benötigt etwas mehr Zeit, um den Sollwert zu erreichen. Nach $2s$ erhöht sich die Motordrehzahl um $400rpm$ und nach $4s$ das Motordrehmoment um $600Nm$. Die Stellgrößen sind einerseits das Bremsmoment des Prüfstandes $M_b(t)$, um die Motordrehzahl zu regeln und andererseits die Kraftstoffmenge $m_f(t)$, um das Drehmoment einstellen zu können.

6.1.2. Vorgabe der Solltrajektorien

Aus dem aktuellen Zustand des Motors ist es möglich, die für die Regelung des Luftpfades notwendigen Sollwerte der beiden Regelgrößen Ladedruck $p_i(t)$ und Frischluftmasse $W_{ci}(t)$ vorzugeben. Am Prüfstand und in der realen Verwendung des Motors wird die Vorgabe jener Solltrajektorien durch das Motorsteuergerät durchgeführt. Da auch diese Werte in den Messdaten vorhanden sind, ist es relativ einfach möglich, die beiden Regelgrößen mit Hilfe einer Kennfläche darzustellen. In den Abbildungen 6.6 und 6.7 sind die Kennflächen, und die gemittelten Werte an den Linearisierungspunkten der beiden Regelgrößen dargestellt. Wie in Kapitel 6.3 erklärt wird, wird allen Messpunkten ein zugehöriger Linearisierungspunkt zugeordnet. Die Mittelwerte ergeben sich aus allen Messpunkten eines Linearisierungspunktes. Beide Größen steigen mit der Drehzahl und der zugeführten Kraftstoffmenge an. Diese Kennflächen werden nun verwendet, um ausgehend vom aktuellen Betriebspunkt (N_e,m_f) die korrespondierenden Sollwerte

(W_{ci},p_i) für die Luftpfadregelung vorzugeben. Abschließend wird in Abbildung 6.8 schema-

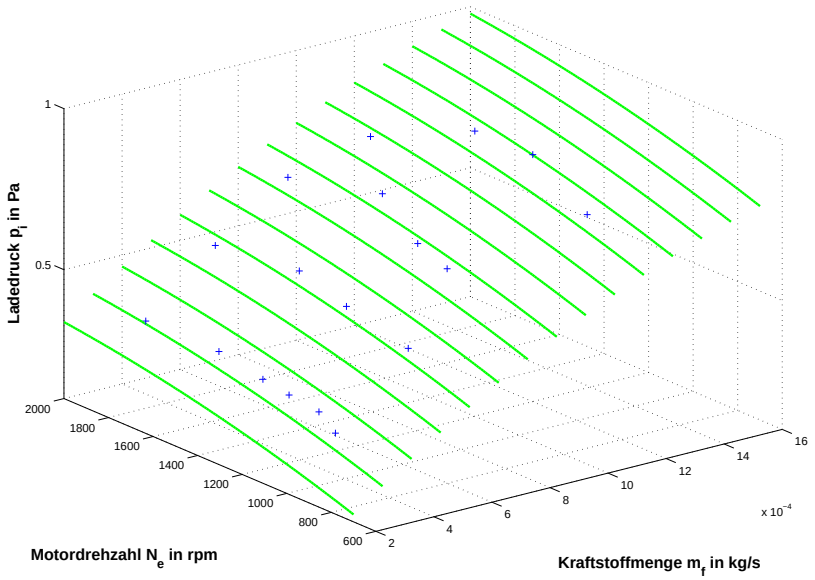

Abbildung 6.6.: Kennlinien für die Vorgabe des Ladedrucks

tisch der Aufbau des entwickelten Regelkreises dargestellt. Dabei sind die zuvor beschriebenen Komponenten *PI-Regler* und die *Kennlinienfelder* für die Solltrajektorien enthalten. Der *MPR* Block repräsentiert den verwendeten Regler. In diesem ist eine modellbasierte prädiktive Regelung (engl. „model predictive control“) enthalten. Der Entwurf wird in Kapitel 7 erklärt. Als Ausgang besitzt er die beiden Stellgrößen, Stellung der Klappe der Abgasrückführung $x_{egr}(t)$ und die Geometrie der Turbinenstellung $x_{vgt}(t)$.
Zusätzlich wurde bereits ein Block abgebildet, in dem der Index jenes Linearisierungspunktes ermittelt wird, der den kleinsten Abstand zum gegebenen Zustand besitzt. Der Abstand d_{AP}, der den aktuellen Motorzustand zu einem Linearisierungspunkt darstellt, wird in Gleichung 6.1 berechnet. Es kann nun jenes Modell zur Regelung verwendet werden, dessen Linearisierungspunkt den geringsten Abstand d_{AP} aufweist.

$$d_{AP} = (m_{f,AP} - m_f)^2 + (N_{e,AP} - N_e)^2 \tag{6.1}$$

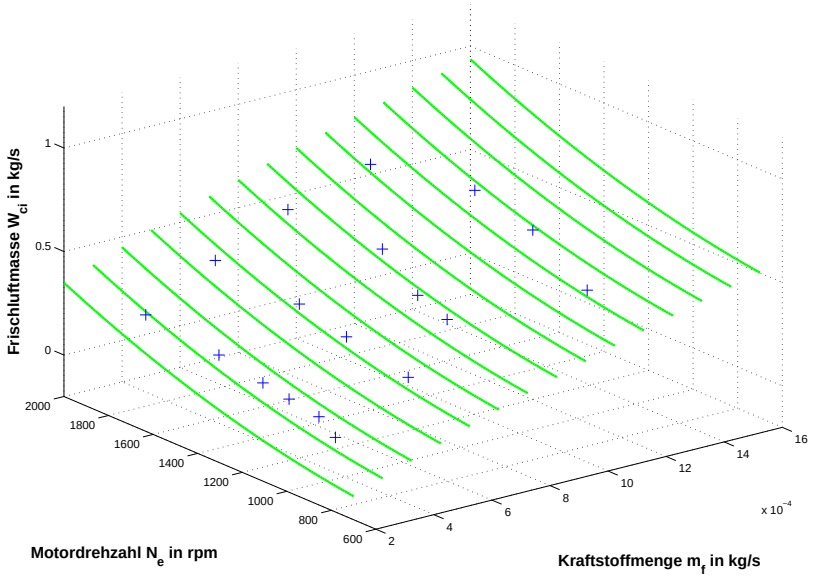

Abbildung 6.7.: Kennlinien für die Vorgabe der Frischluftmasse

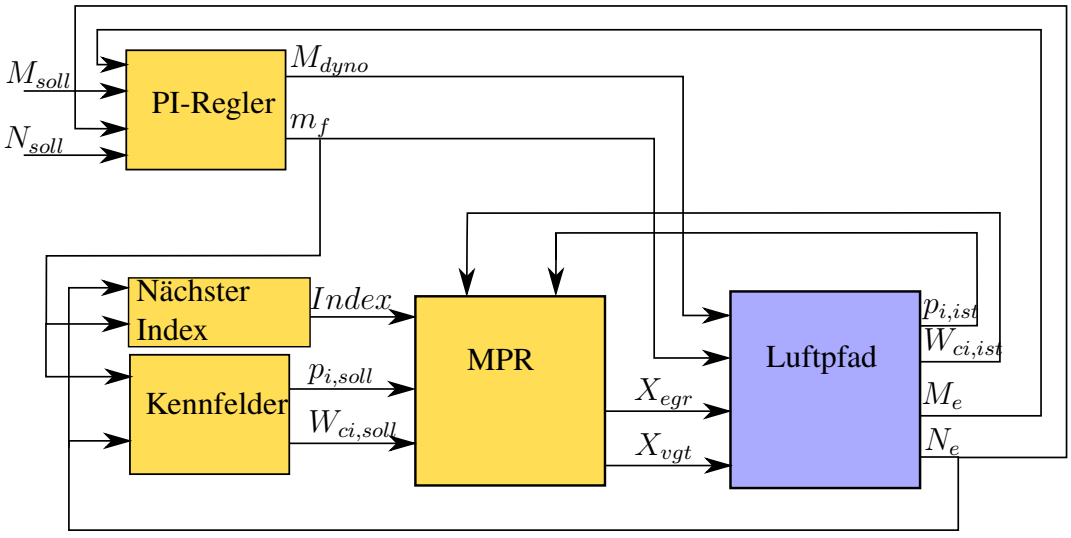

Abbildung 6.8.: Aufbau des Regelkreises für den Luftpfad

6.2. Einbindung des Co-Simulationswerkzeuges ICOS

Für die Simulation des Reglers mit dem Lufpfadmodell wird das Co-Simulationswerzeug (engl. „independent CO-Simulation“) verwendet. Dabei handelt es sich um eine Software, mit deren Hilfe es möglich ist, Simulationen von Teilsystemen, die aus unterschiedlichen Domänen stammen miteinander zu verbinden. Dies wird in technischen Entwicklungen zunehmend wichtiger, da immer komplexere Modelle zur Vorhersage des Verhaltens eines Produktes benötigt werden. Da sich diese Implementierungen aber auf unterschiedlichen Abstrahierungsebenen befinden, können nicht alle in der selben Softwareumgebung simuliert werden. ICOS besitzt eine flexible Sever/Client Struktur, wie in [ViF, 2011] beschrieben wird. Das heißt, die Berechnungen können im Netzwerk verteilt durchgeführt werden. Von den Clients wird die Simulation konfiguriert und gestartet. In Abbildung 6.9 ist ein selbst arrangierendes Netzwerk dargestellt. In die-

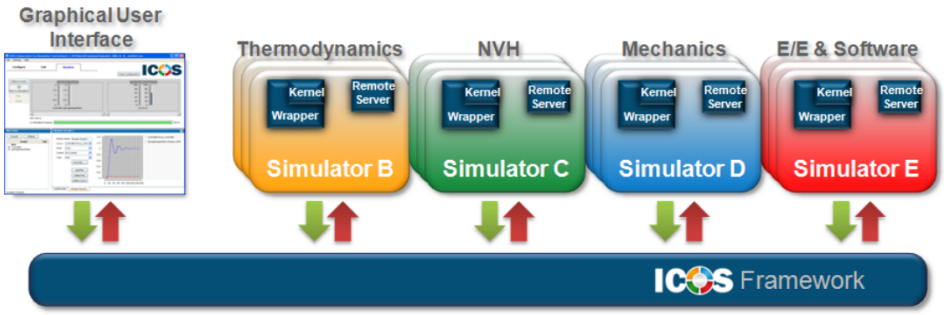

Abbildung 6.9.: Koppelung von Simulationswerkzeugen,[ViF, 2011]

ser Arbeit werden die beiden Simulationswerkzeuge Matlab/Simulink und Dymola/Modelica gekoppelt. Die Kopplung erfolgt dabei noch nach dem sogenannten „nicht-iterativem“Prinzip. ICOS lässt jeweils eine Software einen Zeitschritt berechnen, übergibt die zuvor definierten Ausgangsgrößen an das jeweils andere Programm und lässt dieses dann berechnen. Im Falle einer geschlossenen Schleife ist diese sequentielle Vorgehensweise problematisch, da die Eingangsgrößen des zuerst simulierten Teilsystems nicht zur Verfügung stehen. Einen Ausweg bietet hier eine Extrapolation von Koppelgrößen. Dadurch entsteht ein Resultat, bei dem die beiden Umgebungen gekoppelt sind. Wichtig für die Qualität der Simulation ist selbstverständlich der gewählte Zeitschritt. Da in dieser Arbeit eine modellbasierte prädiktive Regelung realisiert wird, welche zeitdiskret ist, kann als Zeitschritt die Abtastzeit des Reglers gewählt werden. Da das Modell als erstes Teilsystem gelöst wird, ist eine Extrapolation 0-ter ordnung sinnvoll.

6.2.1. Schnittstellen zwischen Simulink und Dymola

In ICOS ist es notwendig, die Richtung jeder gekoppelten Größe vorzugeben, das heißt, Eing- und Ausgangsgrößen müssen spezifiziert werden. In Tabelle 6.2 sind diese für die vorliegende Aufgabenstellung zusammengefasst angeführt. Obwohl für die Regelung des Luftpfades nur die beiden Regelgrößen $p_i(t)$ und $W_{ci}(t)$ notwendig sind, müssen die Größen $N_e(t)$ und $M_e(t)$ ebenfalls über ICOS verbunden werden, um den Zustand des Motors zu definieren.

Benennung	Richtung in Dymola	Erklärung
p_i	Ausgangsgröße	Ladedruck
W_{ci}	Ausgangsgröße	Frischluftmasse
N_e	Ausgangsgröße	Motordrehzahl
M_e	Ausgangsgröße	Motordrehmoment
x_{egr}	Eingangsgröße	EGR-Klappe
x_{vgt}	Eingangsgröße	Turbinenstellung
M_{dyno}	Eingangsgröße	Bremsmoment des Prüfstandes
m_f	Eingangsgröße	Kraftstoffmenge

Tabelle 6.2.: Gekoppelte Größen bei der Co-Simulation

6.3. Linearisierung in Dymola

In Kapitel 7 wird eine lineare modellbasierte prädiktive Regelung entworfen. Um das dafür notwendige Modellwissen in die Regelung einbringen zu können, wird ein lineares Zustandsraummodell benötigt. Anhand dieser linearen Modelle wird in Simulink eine MPR erarbeitet.
Das Softwarepaket Dymola stellt über einen Menüeintrag einen Befehl zur Verfügung, mit dem die Möglichkeit besteht, das aktuelle Modell zu linearisieren. Es werden die im Modell definierten Ein- und Ausgänge für das linearisierte Zustandsraummodell verwendet. Wichtig ist dabei, dass vor der Linearisierung das gesamte Modell initialisiert werden muss. Deshalb müssen die Zustände und die Eingänge derart vorgegeben werden, damit die Linearisierung in der gewünschten Ruhelage durchgeführt wird. Dymola erzeugt dabei die Datei *dslin.mat*, in welcher sich die Matrizen für das lineare Zustandsraummodell A, B, C, D befinden.
Über den Befehl *[A,B,C,D,xName,uName,yName] = tloadlin('dslin')* kann dieses Modell in Matlab importiert werden, um für den Reglerentwurf herangezogen werden zu können.
Da sich der Luftpfad des Dieselmotors in unterschiedlichen Zuständen befinden kann, wurde diese Linearisierung anhand des in Abbildung 6.10 dargestellten Rasters mehrmals durchgeführt. Das Raster wird über die beiden Größen Drehzahl N_e und Kraftsoffmenge m_f aufgespannt, da über diese der Zustand des Motors definiert wird. Die Wahl der konkreten Punkte ergab sich aus den vorhandenen Messwerten des realen Motors. Sie sind in Abbildung 6.10 durch Kreise dargestellt. An den schwarzen Kreisen erkennt man, dass bei den Messungen nur bestimmte Drehzahlen angefahren wurden. Sinnvollerweise werden diese für die Linearisierung verwendet.

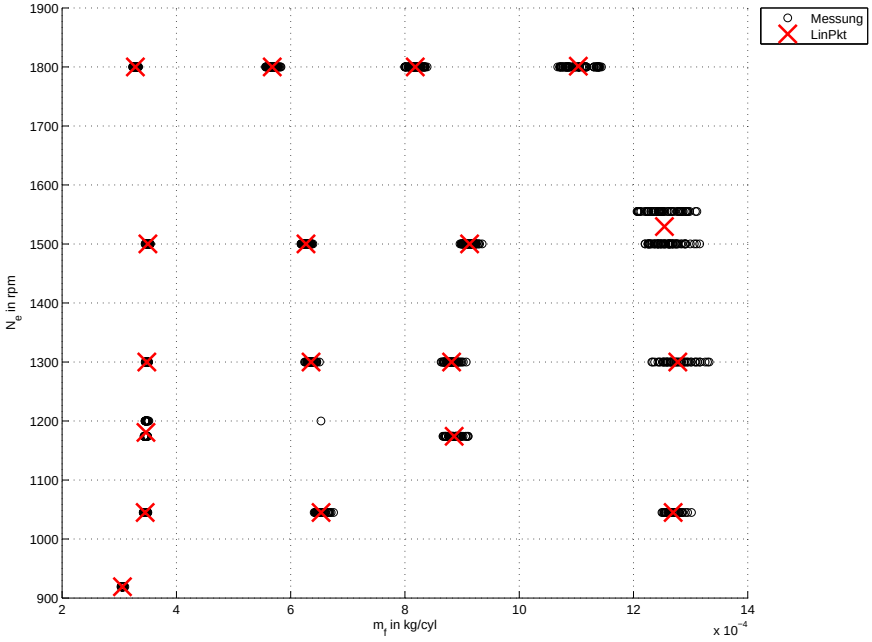

Abbildung 6.10.: Betriebspunkte zur Linearisierung

Kapitel 7

Modellbasierte prädiktive Regelung

In diesem Kapitel werden grundlegende Eigenschaften, sowie Vor- und Nachteile, einer modellbasierten Regelung angeführt. Da der Kostendruck in der industriellen Fertigung stetig zunimmt, ist es notwendig, die Produktionsprozesse möglichst ressourcenschonend ablaufen zu lassen. Dazu trägt selbstverständlich eine passende Regelung einen großen Teil bei. Vor allem bei langsam ablaufenden Prozessen, wie sie in der chemischen Industrie vorkommen, fand die prädiktive Regelung schnell Verbreitung [Maciejowski, 2002].
Die folgende Aufzählung enthält die wesentlichsten Vorteile einer modellbasierten prädiktiven Regelung.

- Für regelungstechnische Aufgaben, in denen die zu regelnde Strecke einen ausgeprägten Mehrgrößencharakter aufweist, stellt die MPR eine hervorragende Wahl dar. Unter ausgeprägtem Mehrgrößencharakter versteht man Mehrgrößensysteme, bei denen die Koppelung einzelner Eingangsgrößen zu mehreren Ausgangsgrößen, und umgekehrt stark ist. Es ist offensichtlich, dass bei derartigen Strecken getrennte Eingrößenregelungen schwierig zu realisieren sind. Um dem Ziel, sämtlichen Regelgrößen ein gewünschtes Verhalten aufprägen zu können, näher zu kommen, birgt die Einbeziehung des Modellwissens immense Vorteile.
- Jede reale Regelstrecke ist einer Beschränkung der Stellgrößen unterworfen. Herkömmliche Regler müssen so dimensioniert werden, dass sie unter normalen Bedingungen nicht in die Beschränkungen kommen, da dies ohne zusätzliche Maßnahmen zu einem nicht prognostizierbaren Verhalten des Regelkreises führen kann. Der Vorteil in einer modellbasierten prädiktiven Regelung ist, dass jene Stellgrößenbeschränkungen beim Entwurf des Reglers angegeben werden können. Dadurch wird bei der Optimierung im laufenden Betrieb automatisch auf die Einhaltung der Beschränkungen geachtet.
- Vor allem im verfahrenstechnischen Bereich kommt es oft vor, dass sich geänderte Kostenstrukturen auf den optimalen Arbeitspunkt eines Prozesses auswirken. Geschieht dies, muss die für den Prozess verantwortliche Regelung ebenfalls geändert werden. Die dafür notwendige Strukturflexibilität wird durch eine modellbasierte prädiktive Regelung gewährleistet. Das bedeutet, dass ein ausgefallener Sensor, oder die Wartung eines Anlagenteils nicht zu einem völligen Versagen der Regelung führt.

Für die Beschreibung der Funktionsweise der prädiktiven Regelung wird die Symbolik aus Tabelle 7.1 verwendet. Diese Form der Regelung von Systemen kann aufgrund der Vorausberechnungen praktisch nur als diskreter Regler realisiert werden. Deshalb sind die vom Regler generierten Größen, wie Stellgrößen u_k und prädizierte Regelgrößen $\hat{y}_k$, Zahlenfolgen, deren

Benennung	Erklärung
u_k	Stellgröße
$y_{ref}(t)$	Referenztrajektorie der Regelgröße
$y_{ref,k}$	Referenztrajektorie der Regelgröße zum Zeitpunkt k
$\hat{y}_k$	prädizierte Regelgröße
Q	Ausgangsgewichtungsmatrix
R	Stellgrößengewichtungsmatrix
T_d	Abtastperiode
H_p	Prädiktionshorizont
H_u	Steuerungshorizont

Tabelle 7.1.: Größen für die modellbasierte prädiktive Regelung

Werte zum Abtastzeitpunkt $t = k \cdot T_d$ durch den Index k im Symbol gekennzeichnet wird. D.h. es gilt:

$$\begin{aligned} u_k &= u(k \cdot T_d) \\ \hat{y}_k &= \hat{y}(k \cdot T_d) \end{aligned} \tag{7.1}$$

usw.

7.1. Prinzip der modellbasierten prädiktiven Regelung

Im Wesentlichen wird bei der MPR versucht, durch ein Vorausberechnen des Modells eine optimale Regelung, im Sinne des gewählten Gütekritieriums, zu erreichen. Deshalb ist auch das gesamte Modellwissen notwendig, um diese Vorausberechnung durchführen zu können. Um von einer optimalen Regelung sprechen zu können, muss zuerst das zuvor erwähnte Gütekriterium J formuliert werden, welches es zu minimieren gilt. Auf die Wahl dieses Kriteriums und der dazugehörigen Parametrierung wird später näher eingegangen.
Der Regler versucht ausgehend vom aktuellen Zustand x_k der Regelstrecke durch eine geeignete Wahl der zukünftigen Elemente der Stellgröße u_k bis u_{k+H_u} das gewählte Gütekriterium zu minimieren. Dazu werden mit Hilfe des Modells und der gewählten Stellgröße die zukünftigen Ausgänge $\hat{y}_k$ des realen Systems errechnet und mit den Werten der vorgegebenen Referenztrajektorie $y_{ref,k}$ zu den Zeitpunkten k der Regelgröße verglichen. In Abbildung 7.1 wird dieser Mechanismus anhand einer Eingrößenregelung grafisch erläutert. Es ist anzumerken, dass der Regler die zukünftigen Werte der Referenztrajektorie $y_{ref}(t)$ nicht kennt. Deshalb gilt für die Optimierung im Zeitschritt k der aktuelle Wert $y_{ref,k}$ als der Anzustrebende. Da nur die Stellgrößen bis zum Steuerungshorizont H_u ermittelt werden, ist es notwendig für die Differenz zum Prädiktionshorizont H_p Werte anzunehmen, damit die Verläufe der Ausgangsgrößen berechnet werden können. Es wird angenommen, dass die Stellgrößen konstant verlaufen. Für die Optimierung im nächsten Zeitschritt wird der aktualisierte Wert herangezogen. Dabei hat der Regler das Ziel, die prognostizierten Werte für die Regelgrößen mit möglichst geringen Stellgrößenänderungen rasch an die Solltrajektorie heranzuführen. Da es sich bei der Prädiktion um eine rechenintensive Aufgabe handelt, wird diese durch einen Prädiktionshorizont H_p auf eine bestimmte Zeit eingeschränkt. In dieser sollte sich jedoch ein Großteil der Streckendynamik befinden, damit der Regler sinnvolle Vorhersagen treffen kann.

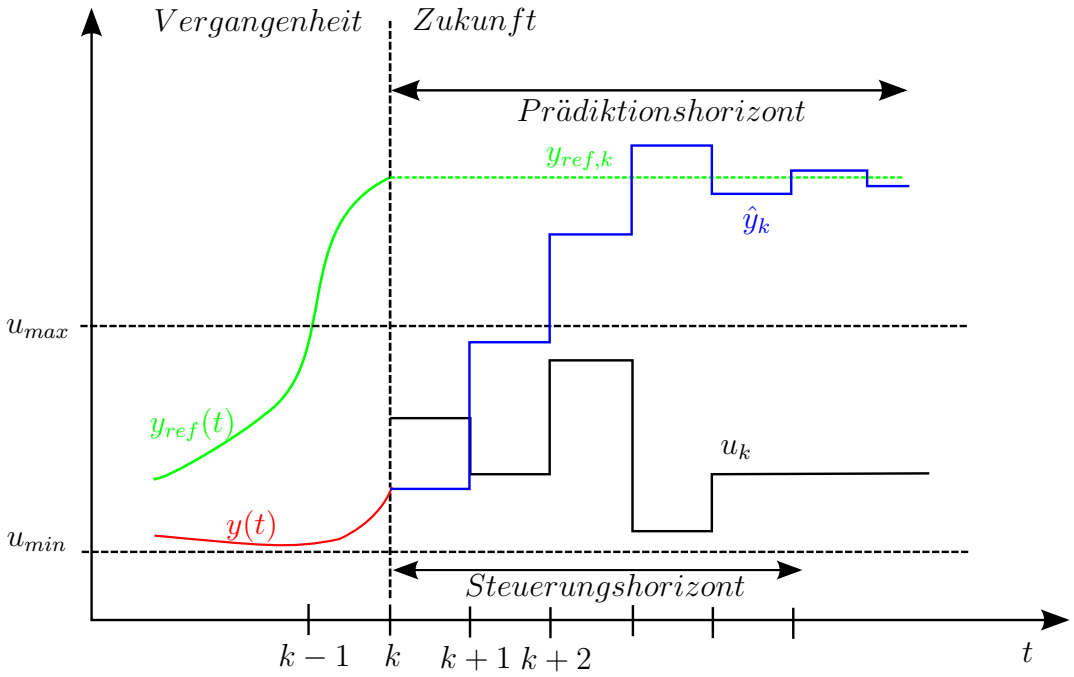

Abbildung 7.1.: Schema einer modellbasierten prädiktiven Regelung

7.1.1. Prinzip des gleitenden Horizonts

Obwohl, die in Abbildung 7.1 dargestellte optimale Stellfolge bis zum Steuerungshorizont H_u ermittlet wird, werden nicht die gesamten Werte für die Regelung herangezogen. Es wird nur der erste Wert der Stellfolge als aktueller Stellgrößenwert verwendet. Einen Zeitschritt später wird die selbe Optimierung mit den aktuellen Messwerten wieder durchgeführt, und von der neu ermittelten Stellfolge wieder nur der erste Wert für die Regelung herangezogen. Diese Vorgehensweise erlaubt es, die auf das System einwirkenden Störgrößen in jedem Optimierungsschritt, aufgrund der aktuellen Messwerte, einfließen zu lassen. Obwohl in jedem Iterationsschritt eigentlich nur eine optimale Steuerung berechnet wird, ergibt sich aufgrund der gewählten Strategie insgesamt eine Regelung.

7.1.2. Kostenfunktion

Entscheidend für die Qualität des Regelkreises ist die Wahl einer geeigneten Kostenfunktion, die bezüglich der Stellfolgenwerte innerhalb des Steuerungshorizontes zu minimieren ist. Allgemein soll selbstverständlich die Abweichung der Regelgrößen von den Solltrajektorien bestraft werden. Zusätzlich ist es sinnvoll, den Aufwand der Stellglieder in Grenzen zu halten. Deshalb wird, wie in Gleichung 7.2 zu sehen ist, diese Änderung ebenfalls bestraft. Für die prädizierten Regelgrößen $\hat{y}_k$ und die berechneten Stellfolgen u_k ist es notwendig mit Hilfe eines weiteren Index den Zeitpunkt der Berechnung anzugeben. Das heißt $\hat{y}_{k+i|k}$ symbolisiert einen zum Zeitpunkt k berechneten zukünftigen Wert der Regelgrößen für den Zeitpunkt $k+i$ und $\Delta u_{k+i|k}$ den zum Zeitpunkt k berechneten zukünftigen Wert der Stellgrößenänderungen für den Zeitpunkt $k+i$:

$$J(k) = \sum_{i=1}^{H_p} \left\| \hat{y}_{k+i|k} - y_{ref,k} \right\|^2 + \sum_{i=0}^{H_u} \left\| \Delta u_{k+i|k} \right\|^2 \tag{7.2}$$

Die prognostizierte Abweichung der Regelgrößen $\hat{y}_{k+i|k}$ von der Referenztrajektorie $y_{ref,k}$ zum Zeitpunkt k wird zum Regelfehler $\hat{e}_{k+i|k}$ in Gleichung 7.3 zusammengefasst:

$$\hat{e}_{k+i|k} = \hat{y}_{k+i|k} - y_{ref,k} \tag{7.3}$$

In der Literatur wird üblicherweise die Matrizenschreibweise verwendet, damit die Übersichtlichkeit erhalten bleibt und weitere Rechenschritte nachvollziehbar durchgeführt werden können. Der Regelfehler und die Stellgrößenänderung werden zu folgenden Vektoren zusammengefasst:

$$\vec{\hat{e}_k} = \begin{pmatrix} \hat{y}_{k+1} - y_{ref,k} \\ \hat{y}_{k+2} - y_{ref,k} \\ \vdots \\ \hat{y}_{k+H_p} - y_{ref,k} \end{pmatrix} \quad \vec{\Delta u_k} = \begin{pmatrix} u_{k+1} - u_k \\ u_{k+2} - u_{k+1} \\ \vdots \\ u_{k+H_u} - u_{k+H_u-1} \end{pmatrix} \tag{7.4}$$

Dazu werden in Gleichung 7.5 die beiden Summen aus Gleichung 7.2 zu Produkten von Vektoren zusammengefasst, [Dittmar, 2004, S.140]. Es wird die Euklidische-Norm verwendet, wodurch sich ein quadratisches Gütekriterium ergibt. Damit unnötige Rechenschritte erspart bleiben, wird die Quadratwurzel, welche normalerweise gebildet wird um die Euklidische-Norm zu ermitteln, durch das Quadrieren aufgehoben.

$$\begin{aligned} \vec{\hat{e}_k}^{\,T} \cdot \vec{\hat{e}_k} &= \sum_{i=1}^{H_p} \left(\hat{y}_{k+i|k} - y_{ref,k}\right)^2 \\ \Delta\vec{u}_k^{\,T} \cdot \Delta\vec{u}_k &= \sum_{i=0}^{H_u} \left(\Delta u_{k+i|k}\right)^2 \end{aligned} \tag{7.5}$$

Um eine Gewichtung der Strafterme anhand der Regelgrößen, und der Stellgrößen vornehmen zu können, werden zusätzlich die beiden Gewichtungsmatrizen Q und R eingeführt. Damit die Minimierung der Kostenfunktion $J(k)$ für die Regelung sinnvoll ist, müssen die Summanden stets nicht negative Werte annehmen. Ist dies nicht der Fall, könnte ein großer positiver Summand, der sich selbstverständlich schlecht auf das Regelverhalten auswirkt, durch einen ebenso großen negativen Wert aufgehoben werden. Deshalb müssen die beiden Matrizen Q und R *positiv semidefinit* sein, [Fischer und Kaul, 2001, S.372]. Praktisch wählt man in den meisten Fällen eine Diagonalmatrix mit positiven Diagonalelementen, wie in Gleichung 7.6 dargestellt, [Dittmar, 2004, S.140]. Die Elemente q_1 bis q_{H_p} repräsentieren die Gewichtungselemente für jeden Zeitschritt im Optimierungsintervall.

$$Q = \begin{pmatrix} q_1 & 0 & \dots & 0 \\ 0 & q_2 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & q_{H_p} \end{pmatrix} \tag{7.6}$$

Dasselbe gilt für die Gewichtungsmatrix R, um die Stellgrößenänderungen zu bewerten.

$$R = \begin{pmatrix} r_1 & 0 & \dots & 0 \\ 0 & r_2 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & r_{H_u} \end{pmatrix} \tag{7.7}$$

Mit dieser Form der Kostenfunktion stehen dem Anwender nun die beiden Horizonte H_p, H_u und die Gewichtungsmatrizen Q, R zur Parametrierung zur Verfügung:

$$J(k) = \vec{\hat{e}}_k{}^T \cdot Q \cdot \vec{\hat{e}}_k + \Delta\vec{u}_k{}^T \cdot R \cdot \Delta\vec{u}_k \qquad Q \geq 0\ ,\ R \geq 0 \tag{7.8}$$

Es ist offensichtlich, dass es für die Regelungseigenschaften vorteilhafter ist, je größer die beiden Horizonte sind, da das Verhalten des Modells weiter prädiziert wird. Außerdem kann nachgewiesen werden, dass bei einem unendlichen Prädiktionshorizont der Regelkreis garantiert stabil ist [Maciejowski, 2002, S.172]. Jedoch ist bei einem großen Prädiktionshorizont zu beachten, dass die zukünftige Solltrajektorie y_{ref} unbekannt ist. Es wird lediglich der zum aktuellen Zeitpunkt bekannte Wert für den gesamten Prädiktionshorizont als Sollwert herangezogen. Das führt zunehmend zu Stellfolgen, die auf einer wahrscheinlich falschen Solltrajektorie beruhen. Dasselbe Problem tritt aufgrund der unbekannten Störgrößen auf. Hinzu kommt der deutlich zunehmende Rechenaufwand, den der Regelungsalgorithmus benötigt. Sinnvollerweise sollte der Prädiktionshorizont so gewählt werden, dass auch die langsamste Streckendynamik ausreichend berücksichtigt werden kann.

7.2. Anwendung der *MPC Toolbox* aus Simulink

In der verwendeten Entwicklungsumgebung Matlab/Simulink wird die sogenannte *Model Predictive Control Toolbox* für die Implementierung einer modellbasierten prädiktiven Regelung zur Verfügung gestellt. Mit Hilfe des *MPC (engl. modellbased predictive control)* Blocks aus dieser Toolbox wird der gewünschte Regler erstellt. Ein solcher Block, wie in Abbildung 7.2

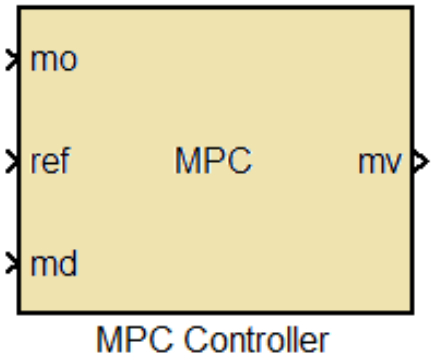

Abbildung 7.2.: MPR Block in Simulink

dargestellt, enthält den bereits konfigurierten Regler. Die Konfiguration des Reglers wird anschließend erläutert. An den Ports *mo* und *ref* sind die Ausgänge des simulierten Modells und die gewünschten Referenztrajektorien anzulegen. Die Dimension entspricht der Anzahl der Regelgrößen, in diesem Fall also 2. Der einzige Ausgangsport enthält als Vektor die beiden Stellgrößen, mit denen das System geregelt werden kann. Der Eingang *md* steht für *measured disturbance* und steht optional zur Verfügung, um messbare Störungen für die Modellvorhersage mit einfließen zu lassen [Mathworks, 2012].
Da in dieser Arbeit die nichtlineare Strecke an mehreren Ruhelagen linearisiert wurde, ist es möglich, für die Regelung einen MPC Regler zu entwerfen, der mehrer linearisierte Modelle

enthält. Diese müssen anhand des aktuellen Zustands des Modells zur Laufzeit passend ausgewählt werden, um die Stellgrößen vorzugeben. Damit das Umschalten nicht manuell durchgeführt werden muss, stellt die verwendete Toolbox einen *Multiple MPC Controller* zur Verfügung. Dieser besitzt neben den bereits beschriebenen Ein- und Ausgänge noch einen weiteren Eingang. An diesem wird ein ganzzahliger Index übergeben, wodurch das zu verwendende Modell vorgegeben werden kann. Das Umschalten zwischen den unterschiedlichen Vorgaben der Stellgrößen wird geglättet, um Sprünge zu vermeiden. Dies kann problematisch sein, da es dadurch zu sehr großen Belastungen der Stellglieder kommen kann.

7.3. Parametrierung des Reglers

In diesem Abschnitt wird die Wahl der Reglerparameter anhand von Simulationsergebnissen begründet. Grundlegend ist zu erwähnen, dass zur Lösung des Optimierungsproblems, die *Standard Form* aus [Mathworks, 2012, 2-5] gewählt wurde. In dieser werden nicht die gesamten, im vorherigen Abschnitt beschriebenen Matrizen gewählt, sondern lediglich die Gewichtung der einzelnen Ausgänge zueinander. In gleicher Weise wird für die Stellgrößenänderung vorgegangen. Für die Gewichtungsmatrizen des i-ten Ausgangs und der i-ten Stellgröße ergibt sich somit folgende Form:

$$Q_i = \begin{pmatrix} q_i & 0 & \dots & 0 \\ 0 & q_i & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & q_i \end{pmatrix} \quad R_i = \begin{pmatrix} r_i & 0 & \dots & 0 \\ 0 & r_i & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & r_i \end{pmatrix} \tag{7.9}$$

Wird ein System mit mehreren Ein- und Ausgängen geregelt ist es notwendig sämtliche Regelabweichungen und Stellgrößenänderungen in die Kostenfunktion einfließen zu lassen. Dazu muss über die Anzahl dieser Größen aufsummiert werden. Für das vorliegende Modell mit 2 Ausgänge und 2 Stellgrößen ergibt sich, die in Gleichungen 7.10 dargestellte Kostenfunktion:

$$J(k) = \sum_{i=1}^{2} \vec{\hat{e}}_k{}^T \cdot Q_i \cdot \vec{\hat{e}}_k + \sum_{i=1}^{2} \Delta\vec{u}_k{}^T \cdot R_i \cdot \Delta\vec{u}_k \tag{7.10}$$

Diese Kostenfunktion erlaubt nur mehr die Wahl eines Koeffizienten für jeden Ausgang q_i und für jede Stellgröße r_i. Die beiden Stellgrößen, Position der Abgasrückführklappe x_{egr} und Stellung der Turbinengeometrie x_{vgt} müssen aufgrund der Konstruktion auf folgende Bedingung beschränkt werden:

$$\begin{aligned} 0 \leq x_{egr} \leq 100 \\ 0 \leq x_{vgt} \leq 100 \end{aligned} \tag{7.11}$$

Um den Einfluss der Änderung einzelner Parameter am Luftpfadmodell darstellen zu können, wurde ein Sprung der Sollwerte im gesamten Regelkreis simuliert. In Abbildung 7.3 ist ein Drehmomentsprung bei unterschiedlichen Prädiktionshorizonten H_p dargestellt. Das Modell des Luftpfades ist in dieser Simulation in Dymola ausgeführt worden, und der Regler mit der Prüfstandssimulation in Simulink. Diese beiden Umgebungen werden, wie zuvor beschrieben mit der Co-Simulationsumgebung ICOS ausgeführt. Es ist deutlich zu erkennen, dass bei kleineren Horizonten ($H_p = 50$) die Abweichung der Regelgrößen vom Sollwert die ersten $2s$

nach dem Sollwertsprung deutlich geringer ist. Jedoch ist am Regelfehler zu erkennen, dass die Bewertung des Endwerts nicht mehr zufriedenstellend einfließen kann. Deshalb wurde für den entworfenen Regler $H_p = 100$ gewählt. Weiters ist der Regelungshorizont H_u zu bestimmen,

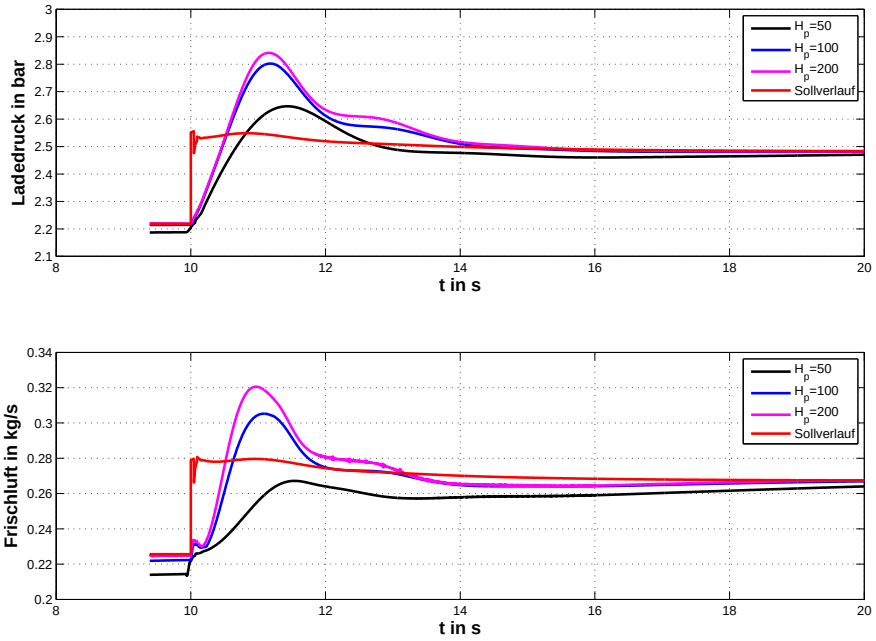

Abbildung 7.3.: Drehmomentsprung bei unterschiedlichen Prädiktionshorizonten

welcher die Anzahl der Zeitschritte festlegt, in der die Elemente der Stellfolge in den Strafterm einfließen. In Abbildung 7.4 ist diese Variation dargestellt. Die Unterschiede bei den Verläufen sind eher gering. Bei der Frischluftmasse ist etwas deutlicher zu sehen, dass ein höherer Regelungshorizont ein etwas geringeres Überschwingen zur Folge hat. Deshalb wurde $H_u = 90$ gewählt. Dieser Werte könnte aber auch geringer gewählt werden, falls nicht ausreichend Rechenleistung vorhanden ist.

Im Fall des Modells für den Luftpfad kann die Gewichtung der Änderung der beiden Stellgrößen $x_{vgt}(t)$ und $x_{egr}(t)$ zueinander unverändert bleiben, da sich beide mit einer ähnlichen Dynamik auf das Modell auswirken. Es wurde für beide 0.1 gewählt, um den Einfluss der Änderung der Stellgrößen nicht zu hoch anzusetzen. Die Gewichtung für den Ladedruck $p_i(t)$ wurde mit 0.7 kleiner gewählt als jene der Frischluftmasse W_{ci} mit 1, da der Ladedruck höhere Werte als die Frischluftmasse aufweist.

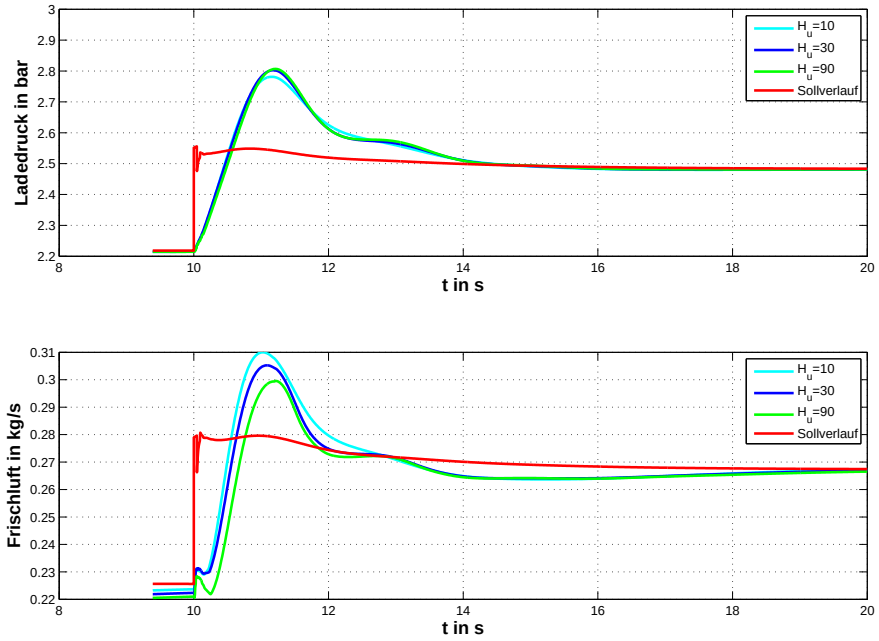

Abbildung 7.4.: Drehmomentsprung bei unterschiedlichen Regelungshorizonten

7.4. Simulationsergebnisse

Abschießend werden 2 Verläufe der gesamten Reglung abgebildet. In den jeweiligen Abbildungen wurde das Einschwingen der übergeordneten PI-Regelkreise ausgeblendet. Diese bringen den gesamten Motor in den gewünschten Zustand. In der Simulation für Abbildung 7.5 springt der Sollwert für die Drehzahl nach $5s$ um $300rpm$ nach oben, und nach $10s$ der des Drehmoments um $200Nm$. Diese Sprünge verursachen die Änderungen der beiden Solltrajektorien für die Regelgrößen Ladedruck $p_i(t)$ und Frischluftmasse W_{ci} des Luftpfads, wie in den beiden Abbildungen zu sehen ist. Zusätzlich sind im unteren Plot die Stellgrößen $x_{vgt}(t)$ und $x_{egr}(t)$ abgebildet, um die Regleraktivität darzustellen. Dabei werden die untere Grenze 0 und die obere Begrenzung von 100 für beide Stellgrößen eingehalten. Diese Grenzen stellen für den Regler harte Grenzen dar, das heißt, sie werden unter keinen Umständen überschritten.

Die Änderung des Motorzustandes in Abbildung 7.5 ist zu klein, um einen Wechsel des MPC-Reglers durch den *Multiple MPC*-Block zu bewirken. Deshalb wird in Abbildung 7.6 ein Drehmomentsprung um $800Nm$ bei $10s$ simuliert. Daraufhin ändert sich der Index, welcher dem *Multiple MPC*-Block den aktuellen Zustand des Motors angibt. Die beiden Solltrajektorien Ladedruck $p_i(t)$ und Frischluftmasse W_{ci} werden, wie in Kap. 6 erläutert über die Kennlinien aus den Abbildungen 6.6 und 6.7 ermittelt. Bei einer entsprechend starken Änderung des Motorzustandes ändern sich diese Sollwerte ebenfalls mit, wie bei $10s$ zu erkennen ist. Bei $14s$ findet das Umschalten zu einer anderen MPC Regelung statt, ersichtlich ist dies an den Stellgrößen. Diese weisen an besagter Stelle einen Knick auf, ändern sich jedoch nicht sprunghaft.

Somit kann mit dem entworfenen Regelungskonzept eine stark nichtlineare Strecke durch Linearisierung an mehreren Punkten mittels einer ebenso großen Anzahl an linearen Streckenmodellen für den MPC Regler zufriedenstellend geregelt werden.

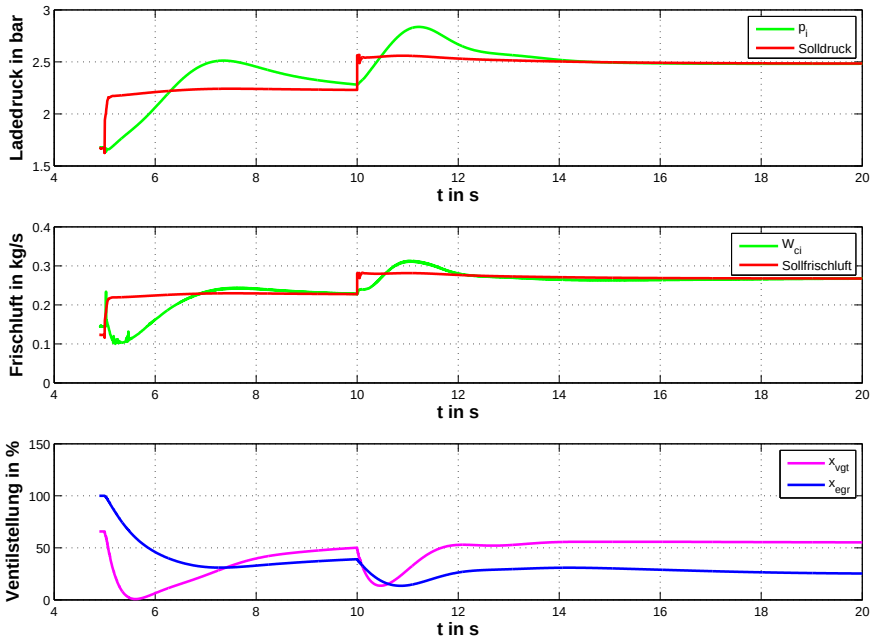

Abbildung 7.5.: Regelungsverlauf mit einem Regler

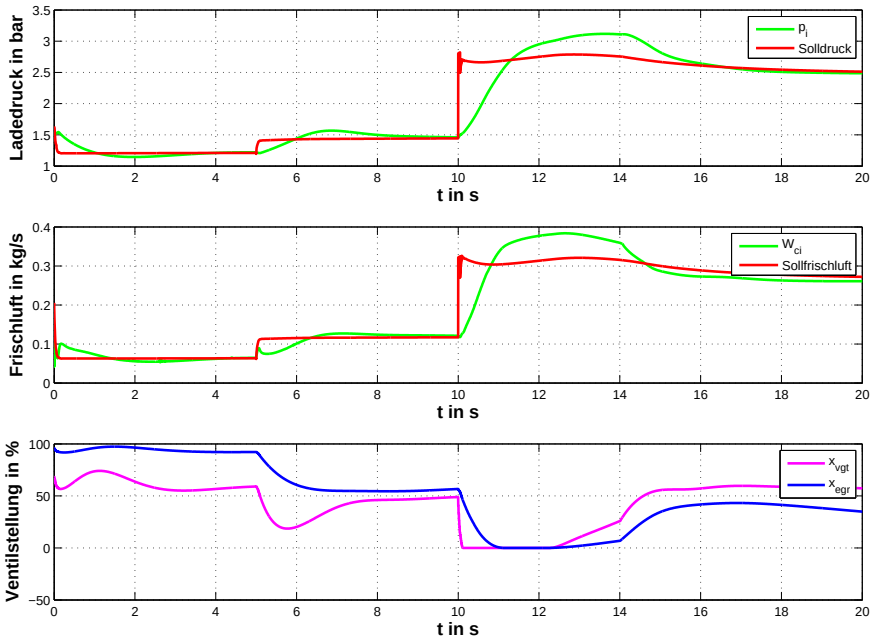

Abbildung 7.6.: Regelungsverlauf mit Wechsel des Linearisierungspunktes

Die Stabilität kann, da der Prädiktionshorizont nicht unendlich gewählt wurde, nicht einfach nachgewiesen werden. Es existieren in der Literatur Verfahren, um die Stabilität bei endlichem Prädiktionshorizont nachzuweisen, diese werden jedoch in dieser Arbeit nicht behandelt. Jedoch mussten, aufgrund einer kleinen Abtastzeit von $0.01s$, im Hinblick auf die Streckendynamiken ein relativ großer Prädiktionshorizont H_p von 100 gewählt werden. Obwohl keine expliziten Maßnahmen zur Einhaltung der Stabilität getroffen wurden und auch keine explizite Bewertung des Endzustandes vorgenommen wurde, traten in den zahlreich durchgeführten Simulationsabläufen keine Instabilitäten auf. Allgemein gilt, je höher der Prädiktionshorizont ist, desto stabiler ist das Verhalten eines Regelkreises mit modellbasierter prädiktiver Regelung. Zusätzlich ist zu beachten, dass der gesamte Luftpfad von den beiden äußeren PI-Regelkreisen „festgehalten“wird und mit den beiden Stellgrößen nur in einem relativ kleinen Bereich geregelt werden kann. Eine detailliertere Betrachtung der Eigenschaften einer modellbasierten prädiktiven Regelung eines Luftpfades soll Gegenstand zukünftiger Arbeiten sein. Dabei kann vor allem auf eine Modellreduktion vertiefend eingegangen werden. Im Rahmen dieser Arbeit wurde die Möglichkeit einer derartigen Regelung mit kleinen Zeitschritten aufgezeigt.

Kapitel 8

Schlussbemerkung und Ausblick

Durch das in dieser Arbeit erstellte Modell des Luftpfades eines LKW Dieselmotors ist es möglich, weitere geeignete Regelungsstrategien zu simulieren. Dabei kann auf eine gleichungsbasierte stabile Implementierung in Modelica zurückgegriffen werden. Die Durchführung in Dymola kann abschließend als sehr zufriedenstellend betrachtet werden. Durch den modularen Aufbau des Modells besteht zukünftig die Möglichkeit, aus diesem Gerüst auch andere Verbrennungsmotoren mit Turboaufladung zu modellieren oder auch zum Beispiel eine zweistufige Turboaufladung zu realisieren. Dadurch ist man in der Lage, ein Regelungskonzept für Motoren zu entwickeln, die sich noch in der Entwurfsphase befinden. Um dem Ziel, der Verkürzung der Entwicklungszyklen von einzelnen Komponenten näher zu kommen, stellt dies einen wichtigen Schritt dar. Weiterführend wäre es möglich, die Modellierung noch weniger auf Kennlinien zu stützen, damit sich Änderungen in der Konstruktion des Motors bequemer in das Modell übernehmen lassen.
Da das Gesamtmodell dem Aufbau eines Motorenprüfstandes nachempfunden wurde, sollte es einen sehr geringen Aufwand benötigen, den entworfenen Regler am realen Motor zu testen. Die Anbindung an das Co-Simulationswerkzeug ICOS ermöglicht es außerdem, das Modell in einer Fahrzeugsimulation einzubinden. Damit können umfangreichere Situationen simuliert werden, wie zum Beispiel modellübergreifende Zusammenhänge mit anderen Teilen des Antriebsstrangs.

Anhang A

Abkürzungsverzeichnis

DASSL Differential Algebraic System Solver
ICOS Independent Co-Simulation
MPR modellbasierte prädiktive Regelung
BDFs Backward Differentiation Formulas
ViF Virtuelles Fahrzeug
VGT Variable Turbinengeometrie

Anhang B

Modelica Quellcode des Luftpfadmodells

```
1  package AirPathCombustionEngineLib
2    "Air path of a combustion diesel engine with controller library"
3  connector Pipe
4    Modelica.SIunits.Pressure p;
5    Modelica.SIunits.MassFlowRate W;
6    Modelica.SIunits.Temperature T;
7  end Pipe;
8
9  connector Signal
10   replaceable type SignalType = Real;
11
12   extends SignalType;
13
14 end Signal;
15
16 function psi
17   input Real p_r;
18   output Real value;
19
20   import AirPathCombustionEngineLib.Constants.gamma;
21
22 algorithm
23   value :=if (p_r > (2/(gamma + 1))^(gamma/(gamma - 1))) then sqrt
       (2*gamma/(
24     gamma - 1)*(p_r^(2/gamma) - p_r^((gamma + 1)/gamma))) else sqrt
         (gamma)*(2
25     /(gamma + 1))^((gamma + 1)/(2*gamma - 2));
26 end psi;
27
28   function orificeFlow
29     input Real A_eff;
30     input Real p_in;
31     input Real p_out;
32     input Real T_in;
33     input Real T_out;
34
```

```
35      output Real m;
36
37      import AirPathCombustionEngineLib.Constants.gamma;
38      import AirPathCombustionEngineLib.Constants.R;
39      constant Real C_D = 1;  // Discharge or Flow coefficient [Hey88
           ] (page 906)
40
41      Real a;
42      Real b;
43    algorithm
44      a :=  C_D*A_eff*p_in/sqrt(R*T_in)*psi(p_out/p_in);
45      b :=  -C_D*A_eff*p_out/sqrt(R*T_out)*psi(p_in/p_out);
46
47      m := if (p_out < p_in) then a else  b;
48      m := if (p_out == p_in) then 0 else m;
49    end orificeFlow;
50
51    model Engine
52
53      parameter Modelica.SIunits.Efficiency eta_vol =
           0.92012104387268
54        "Volume efficiency of the Engine";
55      parameter Modelica.SIunits.Volume V_d =  0.00175 "Volume per
           Cylinder";
56      parameter Integer N_cyl =                        6 "Number of
           Cylinders";
57      parameter Integer N_re =                         2 "rounds per
           cycles";
58      parameter Modelica.SIunits.Length L= 0.155 "Length of stroke";
59      import AirPathCombustionEngineLib.Constants.f_st;
60      import AirPathCombustionEngineLib.Constants.R;
61
62      Real W_fmax;
63      Real help_min;
64      Real lambda_cyl;
65      Real W_f;
66
67      Real Tau_ips;
68      Real Tau_ieg;
69      Real max_delay = 0.9;
70
71    function temperatureEngine
72        "Estimated tempeature of gas in the engine as a function of
             air-fuel-ratio and engine speed"
73      input Real lambda_cyl;          // Air fuel ratio
74      input Real N_e;          // Engine speed
75      output Modelica.SIunits.Temperature T_e;      // Temperature
           engine
76
```

```
77      parameter Modelica.SIunits.Temperature[4] K_T_e =
           {103.277173908392, -682.630937178668, 167.950020428914,
           1414.777748765431};
78      Real[4] P_T_e;
79      Real lambda;
80
81    algorithm
82      lambda :=if (lambda_cyl > 3) then 3 else lambda_cyl;
83
84      P_T_e :={lambda^2,lambda,N_e/1000,1};
85      T_e := K_T_e * P_T_e;
86
87    end temperatureEngine;
88
89    function TorqueEngine
90      input Real m_f;
91      input Real N_e;
92      input Real p_i;
93      input Real p_x;
94      input Real V_d;
95      input Real N_cyl;
96      input Real N_re;
97      input Modelica.SIunits.Length L;
98      output Real M_e;
99
100     Real P_me0g;
101     Real P_me0f;
102     Real P_mf;
103     Real P_me;
104     Real eta_th;
105
106     parameter Real[5] K_eta_th = {-64827.71053700243,
           101.29548314877, -0.04936271175, 0.10786159455,
           0.36179106990};
107     parameter Real[5] P_eta_th = {m_f^2, m_f, (N_e/1000)^2, N_e
           /1000, 1};
108
109     import AirPathCombustionEngineLib.Constants.HV_low;
110
111   algorithm
112     P_me0g :=p_x - p_i;
113     P_me0f :=75000 + 48*(N_e/1000) + 0.4*(2*L*N_e/60)^2;
114     P_mf :=(HV_low*m_f)/(V_d*N_cyl);
115
116     eta_th := K_eta_th * P_eta_th;
117     P_me :=P_mf*eta_th - P_me0f - P_me0g;
118     M_e :=(P_me*V_d*N_cyl)/(2*Modelica.Constants.pi*N_re);
119   end TorqueEngine;
120
121     Real M_e;
```

```
122     Real p_x;
123     Real W_ex;
124     Real T_e;
125     Real p_i;
126     Real W_ie;
127     Real T_i;
128
129     Pipe Eng_Ex annotation (extent=[-60,-58; -40,-38]);
130     Pipe Intake_Eng annotation (extent=[-42,42; -22,62]);
131     Signal Fi annotation (extent=[-80,40; -60,60]);
132     Signal mf annotation (extent=[18,40; 38,60]);
133     Signal Ne annotation (extent=[60,40; 80,60]);
134     Signal Met annotation (extent=[86,-6; 110,14]);
135     Signal Wft annotation (extent=[-22,-60; 4,-40]);
136     Signal Wiet annotation (extent=[26,-60; 58,-40]);
137   equation
138
139     Eng_Ex.p = p_x;
140     Eng_Ex.W = W_ex;
141     Eng_Ex.T = T_e;
142
143     Intake_Eng.p = p_i;
144     Intake_Eng.W = W_ie;
145     Intake_Eng.T = T_i;
146
147     Tau_ips = 60/Ne;
148     Tau_ieg = 90/Ne;
149     W_ie = eta_vol * (p_i*N_cyl*V_d*Ne)/(60*N_re*R*T_i);
150     Wiet = delay(W_ie,Tau_ieg,max_delay);
151     W_fmax = (W_ie*(1-Fi))/f_st;
152     help_min = (mf*Ne)/(60*N_re);
153     W_f = min(help_min,W_fmax);
154     Wft = delay(W_f,Tau_ieg,max_delay);
155     W_ex = Wiet + Wft;
156
157     lambda_cyl = (W_ie*(1 - Fi))/(f_st*W_f);
158     T_e  = temperatureEngine(lambda_cyl, Ne);
159     M_e  = TorqueEngine(mf,Ne,p_i,p_x,V_d,N_cyl,N_re,L);
160     Met = delay(M_e,Tau_ips,max_delay);
161
162   end Engine;
163
164   model CompressorTurbo
165
166       parameter Modelica.SIunits.Temperature T_cref=298
167       "Input reference temperature compressor";
168     parameter Modelica.SIunits.Temperature T_tref=923
169       "Input referenz temperature turbo";
170     parameter Modelica.SIunits.Diameter d_c=0.08
171       "Diameter of the compressor wheel";
```

```
172     parameter Modelica.SIunits.Diameter d_t=0.07 "Diameter of the
           turbo wheel";
173     parameter Modelica.SIunits.Pressure p_cref=100000
174       "Input reference pressure ambient";
175     parameter Modelica.SIunits.MomentOfInertia J_t =  0.00016
176       "[kg*m^2] turbocharger inertia";
177
178     function cubicSpline
179       input Real Nt_low;
180       output Real c;
181
182       parameter Real[19] basepoints = {2.1, 2.6, 3.1, 3.6, 4.1,
             4.6, 5.1, 5.6, 6.1, 6.6, 7.1, 7.6, 8.836,
183                                        9.5, 10, 10.56, 11.5, 12.05,
                                              13.39};
184       parameter Real[18,4] K=
185       1000*  {{ -1.39200812230389,
             4.48801218345583,-5.49600406115194, 3.00000000000000},
186       {  -1.39200812230389, 2.40000000000000,-2.05199796942403,
             1.20000000000000},
187       {   0.08004061151945, 0.31198781654417,-0.69600406115194,
             0.60000000000000},
188       {  -0.28815432377389, 0.43204873382333,-0.32398578596819,
             0.34000000000000},
189       {   0.03257668357613,-0.00018275183750,-0.10805279497528,
             0.25000000000000},
190       {  -0.00215241053062, 0.04868227352669,-0.08380303413069,
             0.20000000000000},
191       {  -0.02396704145366, 0.04545365773076,-0.03673506850197,
             0.17000000000000},
192       {  -0.01397942365475, 0.00950309555027,-0.00925669186145,
             0.16000000000000},
193       {   0.01588473607268,-0.01146603993186,-0.01023816405224,
             0.15600000000000},
194       {  -0.00155952063595, 0.01236106417716,-0.00979065192959,
             0.15000000000000},
195       {  -0.00964665352889, 0.01002178322324, 0.00140077177060,
             0.14800000000000},
196       {   0.00880170469882,-0.00444819707010, 0.00418756484717,
             0.15000000000000},
197       {   0.00105025809721, 0.02818852395313, 0.03353060887459,
             0.16500000000000},
198       {  -0.02997780603122, 0.03028063808277, 0.07235413246642,
             0.20000000000000},
199       {   0.11229477314084,-0.01468607096407, 0.08015141602577,
             0.24000000000000},
200       {  -0.07573957825507, 0.17396914791254, 0.16934993911691,
             0.30000000000000},
201       {   0.59733238318429,-0.03961646276677, 0.29564146315393,
             0.55000000000000},
```

```
202         {   0.59733238318430, 0.94598196948732, 0.79414249185023,
               0.80000000000000}};
203         Real Nt_low_shift;
204         Real[4] P;
205
206       algorithm
207         for i in (1:(size(basepoints,1)-1)) loop
208           if
209             (basepoints[i] <= Nt_low and Nt_low < basepoints[i+1])
                  then
210               Nt_low_shift :=Nt_low - basepoints[i];
211               P :={Nt_low_shift^3,Nt_low_shift^2,Nt_low_shift,1};
212               c :=K[i, :]*P;
213           end if;
214         end for;
215
216         if
217           ( Nt_low <= basepoints[1]) then
218           Nt_low_shift :=Nt_low - basepoints[1];
219           P :={Nt_low_shift^3,Nt_low_shift^2,Nt_low_shift,1};
220           c :=K[1, :]*P;
221         end if;
222         if
223           ( Nt_low >= basepoints[size(basepoints,1)]) then
224               Nt_low_shift := Nt_low - basepoints[size(basepoints, 1)
                      -1];
225               P :={Nt_low_shift^3,Nt_low_shift^2,Nt_low_shift,1};
226             c :=K[size(basepoints,1)-1, :]*P;
227         end if;
228
229       end cubicSpline;
230
231       function compressorMassFlow "Mass Flow in the compressor"
232         input Real N_t_norm;
233         input Real p_r;
234         output Real W_c;
235         output Real surge;
236
237         parameter Real Ntcorr =    N_t_norm/10000;
238         parameter Real[6] Kmcsurge = {0.000006865796007,
              -0.000263815322517, 0.003820595771073,
239                                       -0.022602000124399,
                                             0.058740800948434,
                                             -0.001831646618993};
240         parameter Real[6] POLYmcsurge = {(Ntcorr)^5, (Ntcorr)^4, (
              Ntcorr)^3, (Ntcorr)^2, (Ntcorr), 1};
241         parameter Real offset = 0.1;
242         parameter Real gain =              10;
243         Real c = cubicSpline(Ntcorr);
244         Real radikand;
```

```
245       Real p_rmax;
246
247       parameter Real[5] K_p_rratio = {-0.000096023957919,
             0.003254269815725, -0.009280807913107, 0.057558490596182,
             0.969705413821149};
248       parameter Real[5] P_p_rratio = {Ntcorr^4, Ntcorr^3, Ntcorr^2,
              Ntcorr, 1};
249
250     algorithm
251         p_rmax := K_p_rratio * P_p_rratio;
252         surge :=Kmcsurge*POLYmcsurge;
253          radikand := p_rmax - p_r - 0.012*Ntcorr;
254          if ((radikand-offset) >= 0) then
255            W_c := sqrt(sqrt(radikand/c)) + surge;
256          else
257            W_c := -gain*sqrt(sqrt((-radikand+2*offset)/c)) + surge
                + sqrt(sqrt(offset/c)) + gain*sqrt(sqrt(offset/c));
258          end if;
259
260     end compressorMassFlow;
261
262     function compressorEfficiency
263       input Real N_t_norm;          // revolution speed of the turbo
264       input Real W_c;      // mass flow
265       output Real eta_c;        //compressor efficiency
266
267       import AirPathCombustionEngineLib.Constants.R;
268
269        parameter Real[2] K_eta_c = {3.554724041159963e-006,
             -0.042344246959775};
270        parameter Real[2] P_eta_c = {N_t_norm, 1};
271
272     algorithm
273        eta_c := ( -7*(W_c- K_eta_c * P_eta_c)^2 + (-4*10^(-11))*(
             N_t_norm-85000)^2 + 0.765);
274
275     end compressorEfficiency;
276
277    function turboMassFlow
278
279     input Real x_vgt;          // variable geometry turbine position
280     input Real p_r;           // pressure ratio
281     output Real W_xt;      // turbo mass flow
282
283       import AirPathCombustionEngineLib.Constants.gamma;
284       import AirPathCombustionEngineLib.Constants.R;
285
286     parameter Real[6] K_A_vgt = {2.047755332684998,
          -24.763494833185771, 93.539919978535167,
```

```
287                                    0.003090620145594,
                                          0.509151944794677,
                                          -80.194847012283191};
288     Real[6] P_A_vgt;
289     Real A_vgt;
290     Real p_rsat;
291
292   algorithm
293     p_rsat := if (p_r > 3.5) then 3.5 else p_r;
294     P_A_vgt := {p_rsat^3,p_rsat^2,p_rsat,x_vgt^2,x_vgt,1};
295     A_vgt := K_A_vgt * P_A_vgt;
296
297     W_xt := if (1/p_rsat > (2/(gamma+1))^(gamma/(gamma-1))) then
298               A_vgt/sqrt(R) * sqrt( 2*gamma/(gamma-1) * ((1/p_rsat)
                     ^(2/gamma) - (1/p_rsat)^((gamma+1)/gamma))) else
299               A_vgt/sqrt(R) * sqrt(gamma) * (2/(gamma+1))^((gamma
                     +1)/(2*gamma-2));
300   end turboMassFlow;
301
302     function turboEfficiencyHeat
303       input Real x_vgt;          // variable geometry turbine
             position
304       input Real p_r;          // pressure ratio
305       output Real eta_theat;        // turbo heat efficiency
306
307       Real p_ratio;
308     algorithm
309       p_ratio := if (p_r < 1) then 2-p_r else p_r;
310
311      // eta_theat := 12*exp(-2.8*p_ratio) - 0.004*x_vgt;// +0.1;//+
             0.27;
312       eta_theat := 3*exp(-2.2*p_ratio) - 0.0022*x_vgt +0.27-0.16;
313     end turboEfficiencyHeat;
314
315     function turboAeroEfficiency
316       input Real x_vgt;          // variable geometry turbine
             position
317       input Real p_r;          // pressure ratio
318       input Real N_t_norm;          // turbo speed
319       input Real T_x;          // exhaust gas temperature
320       input Real d_t;
321       input Real T_tref;
322       output Real eta_t;        // turbo aero efficiency
323
324       import AirPathCombustionEngineLib.Constants.gamma;
325       import AirPathCombustionEngineLib.Constants.c_p;
326
327       parameter Real[9] K_N_toffset = {-0.0000000008143,
             0.0000002542801, -0.0000235860220, -0.0001019528993,
             0.1238445350996,
```

```
328                                  -5.9165447278170, 100.4384603950022,
                                        -582.0382277768065,
                                     609.08828932608920};
329       parameter Real[9] P_N_toffset = {x_vgt^8, x_vgt^7, x_vgt^6,
             x_vgt^5, x_vgt^4, x_vgt^3, x_vgt^2, x_vgt, 1};
330
331       parameter Real[8] K_eta_toffset = {0.000001196635417,
             -0.000043897811406, 0.000619568317803, -0.004140804241259,
332                                        0.012821399119294,
                                           -0.018063912185653,
                                           0.024935445967421,
                                           0.688737161018410};
333       parameter Real[8] P_eta_toffset = {(x_vgt/10)^7, (x_vgt/10)
             ^6, (x_vgt/10)^5, (x_vgt/10)^4, (x_vgt/10)^3, (x_vgt/10)
             ^2, (x_vgt/10), 1};
334
335       parameter Real[7] K_eta_max = {0.000000015386207,
             -0.000001337808276, 0.000038924715416, -0.000515562405951,
336                                     0.002811758897781,
                                        0.012741890307293,
                                        -0.217253321703842};
337       Real[7] P_eta_max;
338
339       parameter Real[10] K_a = {0.000000003306992,
             -0.000000387900754,   0.000018715489704,
             -0.000479366109257,
340                               0.007036160715956,
                                     -0.059830430229080,
                                     0.284150562094412,
                                     -0.683974800538081,
341                               0.671497562210304,
                                     -3.146418287829716};
342       Real[10] P_a;
343
344       Real c_uopt;
345       Real c_u;
346       Real N_tnew;
347       Real eta_toffset;
348       Real eta_max;
349       Real p_ratio;
350     algorithm
351
352       p_ratio :=  if (p_r < 1) then 2-p_r else p_r;
353
354       // Ermitteln des Blade-Speed-Ratios C/U
355       c_uopt :=(0.13/60000)*N_t_norm + 0.3483;
356       c_u := (Modelica.Constants.pi*d_t*N_t_norm/60) / sqrt(2*c_p*
             T_x * (1-(1/p_ratio)^((gamma-1)/gamma)));
357
358       // Ermitteln der Koeff. der quad. Funktion
```

```
359         N_tnew := (N_t_norm + K_N_toffset * P_N_toffset)/10000;
360         P_eta_max :={N_tnew^6,N_tnew^5,N_tnew^4,N_tnew^3,N_tnew^2,
               N_tnew,1};
361         P_a := {N_tnew^9,N_tnew^8,N_tnew^7,N_tnew^6,N_tnew^5,N_tnew
               ^4,N_tnew^3,N_tnew^2,N_tnew,1};
362
363         eta_toffset :=K_eta_toffset*P_eta_toffset;
364         eta_max :=K_eta_max*P_eta_max + eta_toffset;
365         eta_t :=(eta_max*(2*c_u/c_uopt - (c_u/c_uopt)^2));
366
367       end turboAeroEfficiency;
368
369       function readTurboState
370
371         input Integer APIndex;
372         input String Matrix;
373         output Real N_t;
374       //  Real mat[1136,1];
375         Real mat[18,1];
376       algorithm
377          mat :=DataFiles.readMATmatrix("states_AP.mat",Matrix);
378          N_t :=mat[APIndex, 1];
379        annotation (Icon);
380       end readTurboState;
381
382       import AirPathCombustionEngineLib.Constants.c_p;
383       import AirPathCombustionEngineLib.Constants.gamma;
384       import Modelica.Constants.pi;
385       import AirPathCombustionEngineLib.Constants.c_p;
386       import AirPathCombustionEngineLib.Constants.gamma;
387
388       Real N_t;
389       Real W_c_norm;
390       Real W_xt_norm;
391       Real eta_c;
392       Real P_t;
393       Real P_c;
394       Real eta_t;
395       Real T_t;
396       Real surge;
397       Real eta_t_aero;
398       Real eta_t_heat;
399
400       outer parameter Integer APIndex;
401       Real p_ic;
402       Real W_c;
403       Real T_c;
404       Real p_x;
405       Real W_xt;
406       Real T_x;
```

```
407       Pipe Comp_IC annotation (extent=[-16,82; 4,102]);
408       Pipe Ex_Turbo annotation (extent=[-16,-102; 4,-82]);
409       Signal Xv   annotation (extent=[-84,54; -64,74]);
410       Signal Ta annotation (extent=[-84,-28; -64,-8]);
411       Signal pa annotation (extent=[-84,-52; -64,-32]);
412       Signal pt annotation (extent=[-84,-76; -64,-56]);
413     equation
414        eta_t_aero = turboAeroEfficiency(
415         Xv,
416         p_x/pt,
417         N_t*sqrt(T_tref/T_x),
418         T_x,d_t,T_tref);
419       eta_t_heat = turboEfficiencyHeat(Xv,p_x/pt);
420       eta_t = eta_t_aero - eta_t_heat;
421       W_xt_norm = turboMassFlow(Xv,p_x/pt);
422       W_xt = W_xt_norm*p_x/(sqrt(T_x)*100000);
423       // Compressor equations
424       eta_c = compressorEfficiency(N_t * sqrt(T_cref/Ta), W_c_norm);
425       P_c = W_c * c_p * Ta /eta_c *((p_ic/pa)^((gamma-1)/gamma) - 1);
426       T_c = Ta + Ta*((p_ic/pa)^((gamma-1)/gamma) - 1)/eta_c;
427       (W_c_norm,surge) = compressorMassFlow(N_t * sqrt(T_cref/Ta),
             p_ic/pa);
428       W_c = W_c_norm * sqrt(T_cref/Ta) * pa/p_cref;
429
430       // Turbo equations
431       der(N_t) = (60/(2*pi))^2 * (P_t-P_c)/(J_t*N_t);
432       P_t = W_xt*c_p*T_x*eta_t * (1 - (pt/p_x)^((gamma-1)/gamma));
433       T_t = T_x- eta_t *T_x*(1 - (pt/p_x)^((gamma-1)/gamma));
434
435       Comp_IC.p = p_ic;
436       Comp_IC.W = W_c;
437       Comp_IC.T = T_c;
438
439       Ex_Turbo.p = p_x;
440       Ex_Turbo.W = W_xt;
441       Ex_Turbo.T = T_x;
442
443     equation
444
445     initial algorithm
446     // N_t := readTurboState(APIndex,"StatesTurbo");
447
448     // initialisation at idle speed
449     N_t :=34996;
450     end CompressorTurbo;
451
452     model IntercoolerManifold
453
454       function readIntercoolerState
455
```

```
456       input Integer APIndex;
457       output Real p_ic;
458       output Real m_ic;
459     //  Real mat[1136,2];
460       Real mat[18,2];
461     algorithm
462        mat :=DataFiles.readMATmatrix("states_AP.mat","
              StatesICManifold");
463        p_ic :=mat[APIndex, 1];
464        m_ic :=mat[APIndex, 2];
465       annotation (Icon);
466     end readIntercoolerState;
467
468     function effectiveAreaIC
469       input Real x_air;          // variable geometry turbine
            position
470       input Real p_r;          // pressure ratio
471       output Real A_effic;        // effective area IC
472
473       Real p_ratio;
474     algorithm
475        p_ratio := if (p_r > 1) then 2-p_r else p_r;
476        A_effic :=exp(-450*(1 - p_ratio + 0.0123)) + (-0.0000002*(
             x_air - 100)^2 +    0.002);
477     end effectiveAreaIC;
478
479     import AirPathCombustionEngineLib.Constants.R;
480     import AirPathCombustionEngineLib.Constants.gamma;
481
482     outer parameter Integer APIndex;
483     parameter Modelica.SIunits.Efficiency eta_ic=0.96
484       "Efficiency of the intercooler";
485     parameter Modelica.SIunits.Temperature T_coolic=297
486       "Tempretaure of the cooling fluid in the intercooler";
487     parameter Modelica.SIunits.Volume V_ic=0.03
488       "Volume of the intercooler manifold";
489
490     Real Ttilde_ic;
491     Real A_IC;
492
493     Real p_i;
494     Real T_ic;
495
496     Real m_ic;
497
498     Real p_ic;
499     Real W_c;
500     Real T_c;
501     Pipe IC_Intake
502   equation
```

```
503
504       der(p_ic) = gamma*R*(T_c*W_c - Ttilde_ic*Wci)/V_ic;
505       der(m_ic) = W_c - Wci;
506       Ttilde_ic = (V_ic*p_ic)/(R*m_ic);
507       T_ic = eta_ic*T_coolic + (1-eta_ic)*Ttilde_ic;
508       A_IC = effectiveAreaIC(Xair, p_i/p_ic);
509       Wci = orificeFlow(A_IC, p_ic, p_i, T_ic, Ti);
510
511       IC_Intake.p = p_i;
512       IC_Intake.W = Wci;
513       IC_Intake.T = T_ic;
514
515       Comp_IC.p = p_ic;
516       Comp_IC.W = W_c;
517       Comp_IC.T = T_c;
518
519     initial algorithm
520      //(p_ic,m_ic) := readIntercoolerState(APIndex);
521
522     //initialisation at idle speed
523     p_ic :=106247.203125;
524     m_ic :=0.0359601341;
525
526     end IntercoolerManifold;
527
528     model IntakeManifold
529
530       function readIntakeManifoldState
531
532         input Integer APIndex;
533         output Real p_i;
534         output Real m_i;
535         output Real F_i;
536       //  Real mat[1136,3];
537         Real mat[18,3];
538
539       algorithm
540          mat :=DataFiles.readMATmatrix("states_AP.mat","
               StatesInManifold");
541          p_i :=mat[APIndex, 1];
542          m_i :=mat[APIndex, 2];
543          F_i :=mat[APIndex, 3];
544         annotation (Icon);
545       end readIntakeManifoldState;
546
547       import AirPathCombustionEngineLib.Constants.R;
548       import AirPathCombustionEngineLib.Constants.gamma;
549
550       outer parameter Integer APIndex;
551       parameter Modelica.SIunits.Volume V_i=0.0324
```

```
552         "Volume of the intake manifold";
553
554       Real m_i;
555       Real T_ic;
556       Real W_ci;
557
558       Real W_xi;
559       Real T_xi;
560
561       Real W_ie;
562       Pipe IC_Intake
563               annotation (extent=[-108,12; -88,32]);
564       Pipe EGR_Intake annotation (extent=[-84,-70; -64,-50]);
565       Pipe Intake_Eng annotation (extent=[66,-70; 86,-50]);
566       Signal Ti annotation (extent=[-110,-36; -90,-16]);
567       Signal Fx annotation (extent=[-36,-70; -16,-50]);
568       Signal Fi annotation (extent=[12,-70; 32,-50]);
569       Signal pi annotation (extent=[90,-10; 110,10]);
570     equation
571
572       IC_Intake.p = pi;
573       IC_Intake.W = W_ci;
574       IC_Intake.T = T_ic;
575
576       EGR_Intake.p = pi;
577       EGR_Intake.W = W_xi;
578       EGR_Intake.T = T_xi;
579
580       Intake_Eng.p = pi;
581       Intake_Eng.W = W_ie;
582       Intake_Eng.T = Ti;
583
584       der(pi) = gamma*R*(T_ic*W_ci + T_xi*W_xi - Ti*W_ie)/V_i;
585       der(m_i) = W_ci + W_xi - W_ie;
586       Ti = V_i*pi/(R*m_i);
587       der(Fi) = ((Fx - Fi)*W_xi - Fi*W_ci)/(m_i);
588
589     initial algorithm
590
591     // initialisation at idle speed
592     pi :=106233.6;
593     m_i :=0.034642;
594     Fi :=0.579837;
595     end IntakeManifold;
596
597     model ExhaustManifold
598
599     function readExhaustManifoldState
600
601         input Integer APIndex;
```

```
602         output Real p_x;
603         output Real m_x;
604         output Real F_x;
605     //  Real mat[1136,3];
606       Real mat[18,3];
607
608     algorithm
609         mat :=DataFiles.readMATmatrix("states_AP.mat","
              StatesExManifold");
610         p_x :=mat[APIndex, 1];
611         m_x :=mat[APIndex, 2];
612         F_x :=mat[APIndex, 3];
613        annotation (Icon);
614     end readExhaustManifoldState;
615
616       import AirPathCombustionEngineLib.Constants.R;
617       import AirPathCombustionEngineLib.Constants.gamma;
618       import AirPathCombustionEngineLib.Constants.f_st;
619
620       outer parameter Integer APIndex;
621       parameter Modelica.SIunits.Volume V_x=0.0054;//0.004 "Volume of
            the exhaust manifold";
622       Real m_x;
623       Real p_x;
624       Real W_xt;
625       Real T_x;
626       Real W_xi;
627       Real W_ex;
628       Real T_e;
629       Pipe Ex_Turbo
630       Pipe Ex_EGR
631       Pipe Eng_Ex
632       Signal Fx
633       Signal Fi
634       Signal Wft
635       Signal Wiet
636     equation
637       der(p_x) = gamma*R*(T_e*W_ex - T_x*W_xi - T_x*W_xt)/V_x;
638       der(m_x) = W_ex - W_xi - W_xt;
639       T_x = V_x*p_x/(R*m_x);
640       der(Fx) = (Fi*Wiet + Wft*(1+f_st) - Fx*W_ex)/m_x;
641
642       Ex_Turbo.p = p_x;
643       Ex_Turbo.W = W_xt;
644       Ex_Turbo.T = T_x;
645
646       Ex_EGR.p = p_x;
647       Ex_EGR.W = W_xi;
648       Ex_EGR.T = T_x;
649
```

```
650       Eng_Ex.p = p_x;
651       Eng_Ex.W = W_ex;
652       Eng_Ex.T = T_e;
653
654     initial algorithm
655     // (p_x, m_x, Fx) := readExhaustManifoldState(APIndex);
656
657     // initialisation at idle speed
658     p_x :=116267;
659     m_x :=0.0022097;
660     Fx :=0.9999590;
661     end ExhaustManifold;
662
663     model ExhausGasRecirculation
664         parameter Modelica.SIunits.Temperature T_coolegr=356
665         "Temperature of the cooling fluid in the EGR cooler";
666       parameter Modelica.SIunits.Efficiency eta_egr=0.96 "Efficiency
            of the EGR";
667
668       function effectiveAreaEGR
669         input Real x_egr;          //
670         input Real p_r;          // pressure ratio
671         output Real A_effegr;        // effective area EGR
672
673         output Real p_ratio;
674       algorithm
675         p_ratio := if (p_r > 1) then 2-p_r else p_r;
676
677         A_effegr :=0.0003*p_ratio^25 + exp(-1600*(1-p_ratio+0.003)) +
              0.000005*x_egr; //0.000005*x_egr;
678       end effectiveAreaEGR;
679
680       Real A_EGR;
681       Real p_x;
682       Real W_xi;
683       Real T_x;
684       Real p_i;
685       Real T_xi;
686       Pipe Ex_EGR;
687       Pipe EGR_Intake;
688       Signal Ti;
689       Signal Xegr;
690     equation
691       Ex_EGR.p = p_x;
692       Ex_EGR.W = W_xi;
693       Ex_EGR.T = T_x;
694
695       EGR_Intake.p = p_i;
696       EGR_Intake.W = W_xi;
697       EGR_Intake.T = T_xi;
```

```
698
699       A_EGR = effectiveAreaEGR(Xegr,p_i/p_x);
700       W_xi = orificeFlow(A_EGR,p_x,p_i,T_x,Ti);
701       T_xi = eta_egr*T_coolegr + (1-eta_egr)*T_x;
702
703     end ExhausGasRecirculation;
704
705     model InertiaEngine
706
707       Modelica.Blocks.Interfaces.RealInput M_t annotation (extent
             =[-128,-12; -88,28]);
708       Modelica.Blocks.Interfaces.RealOutput N_e annotation (extent
             =[92,-10; 132,30]);
709        parameter Modelica.SIunits.MomentOfInertia J_e =  1.6
710         "[kg*m^2] engine inertia";
711       import Modelica.Constants.pi;
712
713       function readInertiaState
714
715           input Integer APIndex;
716           input String Matrix;
717           output Real N_e;
718
719       //     Real mat[1136,8];
720           Real mat[18,5];
721
722       algorithm
723       //      mat :=DataFiles.readMATmatrix("InOutValues.mat",Matrix);
724           mat :=DataFiles.readMATmatrix("InValuesLin.mat",Matrix);
725           N_e :=mat[APIndex, 2];
726           annotation (Icon);
727       end readInertiaState;
728
729     //  outer parameter Integer APIndex;
730       Modelica.Blocks.Interfaces.RealSignal realSignal
731         annotation (extent=[-46,62; -26,82]);
732     equation
733       der(N_e) = (60*M_t)/(2*pi*(J_e));
734
735     initial algorithm
736     // N_e := readInertiaState(APIndex,"InOutValues");
737      N_e := 919; //readInertiaState(APIndex,"InValuesLin");
738     end InertiaEngine;
739
740     record Constants "Natural constants "
741       constant Modelica.SIunits.
             SpecificHeatCapacityAtConstantPressure c_p=1014.4
742         "specific heat coeficient at constant pressure";
743       constant Modelica.SIunits.SpecificHeatCapacityAtConstantVolume
             c_v=727.4
```

```
744         "specific heat coeficient at constant volume";
745       constant Modelica.SIunits.RatioOfSpecificHeatCapacities gamma
            =1.39455595270828
746         "specific heat relation";
747       constant Modelica.SIunits.SpecificEntropy R=287.058 "constant
            for air";
748       constant Real f_st=14.6 "stoichiometric faktor for diesel";
749       constant Modelica.SIunits.SpecificEnergy HV_low=43000000
750         "lower heating value";
751     end Constants;
752
753     model Airpath "airpath of a diesel combustion engine"
754
755       Modelica.Blocks.Interfaces.RealInput N_e annotation (extent
            =[-88,72; -68,92]);
756       Modelica.Blocks.Interfaces.RealInput m_f annotation (extent
            =[-88,54; -68,74]);
757       Modelica.Blocks.Interfaces.RealInput x_egr annotation (extent
            =[-88,2; -68,
758             22]);
759       Modelica.Blocks.Interfaces.RealInput x_air annotation (extent
            =[-88,22; -68,
760             42]);
761       Modelica.Blocks.Interfaces.RealInput x_vgt annotation (extent
            =[-88,-18; -68,
762             2]);
763       Modelica.Blocks.Interfaces.RealOutput M_t annotation (extent
            =[70,-28; 90,-8],
764                 rotation=0);
765       IntercoolerManifold intercoolerManifold
766       IntakeManifold intakeManifold annotation (extent=[4,8; 44,48]);
767       ExhaustManifold exhaustManifold annotation (extent=[-14,-82;
            26,-42]);
768       Engine engine annotation (extent=[34,-38; 74,2]);
769       Constants constants(c_p) annotation (extent=[70,-90; 90,-70]);
770
771       outer parameter Integer APIndex;
772
773       ExhausGasRecirculation exhausGasRecirculation;
774       Modelica.Blocks.Interfaces.RealInput p_a;
775       Modelica.Blocks.Interfaces.RealInput T_a;
776       Modelica.Blocks.Interfaces.RealInput p_t;
777       Modelica.Blocks.Interfaces.RealOutput p_i;
778       Modelica.Blocks.Interfaces.RealOutput W_ci;
779       CompressorTurbo compressorTurbo;
780     equation
781
782       connect(compressorTurbo.Comp_IC, intercoolerManifold.Comp_IC);
783       connect(intakeManifold.EGR_Intake, exhausGasRecirculation.
            EGR_Intake);
```

```
784       connect(exhaustManifold.Ex_Turbo, compressorTurbo.Ex_Turbo);
785       connect(exhaustManifold.Ex_EGR, exhausGasRecirculation.Ex_EGR);
786       connect(exhaustManifold.Eng_Ex, engine.Eng_Ex);
787       connect(engine.Intake_Eng, intakeManifold.Intake_Eng);
788       connect(x_air, intercoolerManifold.Xair);
789       connect(W_ci, intercoolerManifold.Wci);
790       connect(x_egr, exhausGasRecirculation.Xegr);
791       connect(intakeManifold.pi, p_i);
792       connect(intakeManifold.Ti, intercoolerManifold.Ti);
793       connect(T_a, compressorTurbo.Ta)bo.pa);
794       connect(p_t, compressorTurbo.pt);
795       connect(N_e, engine.Ne);
796       connect(engine.Met, M_t);
797       connect(exhaustManifold.Wiet, engine.Wiet);
798       connect(m_f, engine.mf));
799       connect(engine.Fi, intakeManifold.Fi));
800       connect(exhaustManifold.Fi, intakeManifold.Fi);
801       connect(exhaustManifold.Fx, intakeManifold.Fx));
802       connect(exhausGasRecirculation.Ti, intakeManifold.Ti);
803       connect(engine.Wft, exhaustManifold.Wft);
804       connect(intakeManifold.IC_Intake, intercoolerManifold.IC_Intake
             );
805       connect(x_vgt, compressorTurbo.Xv);
806     end Airpath;
807
808     model Airpath_ICOS_model
809       ICOS.Controller controller(
810         Port=3333,
811         OutputParameter=5,
812         TSample=0.01,
813         InputParameter=4)  annotation (extent=[-4,-2; 26,38]);
814       AirpathICOSextract airpathICOSextract annotation (extent
             =[-64,8; -22,36]);
815     equation
816       connect(airpathICOSextract.y, controller.u) annotation (points
             =[-22.84,
817           17.24; -12.37,17.24; -12.37,18; -2.5,18],
818                                                 style(color=74,
                                                       rgbcolor={0,0,127}))
                                                       ;
819       connect(controller.y, airpathICOSextract.u) annotation (points
             =[24.5,18; 38,
820           18; 38,48; -76,48; -76,24.24; -63.58,24.24],
821                                                       style(color=74,
                                                             rgbcolor={0,0,
822             127}));
823     end Airpath_ICOS_model;
824
825     model AirPathTest
826       "Testbed for the airpath with all measument points "
```

```
827
828       inner parameter Integer APIndex = 1;
829       Integer i;
830       Integer setuptime = 0;
831       Integer holdtime = 5;
832       Integer iter = 50;
833
834     function readInOutValues
835
836         input Integer APIndex;
837         input String Matrix;
838         output Real m_f;
839         output Real x_air;
840         output Real x_egr;
841         output Real x_vgt;
842         output Real p_t;
843         output Real p_a;
844         output Real T_a;
845
846         Real mat[1136,8];
847
848     algorithm
849          mat :=DataFiles.readMATmatrix("InOutValues.mat",Matrix);
850          m_f :=mat[APIndex, 2];
851          x_air :=mat[APIndex, 3];
852          x_egr :=mat[APIndex, 4];
853          x_vgt :=mat[APIndex, 5];
854          p_t :=mat[APIndex, 6];
855          p_a :=mat[APIndex, 7];
856          T_a :=mat[APIndex, 8];
857        annotation (Icon);
858     end readInOutValues;
859
860     function readNe
861
862         input Integer APIndex;
863         input String Matrix;
864         output Real N_e;
865
866         Real mat[1136,8];
867
868     algorithm
869          mat :=DataFiles.readMATmatrix("InOutValues.mat",Matrix);
870          N_e :=mat[APIndex, 1];
871        annotation (Icon);
872     equation
873
874     end readNe;
875
876       InertiaEngine inertiaEngine;
```

```
877     Modelica.Blocks.Math.Add add4(k2=+1);
878     Real m_f0;
879     Real x_air0;
880     Real x_egr0;
881     Real x_vgt0;
882     Real p_t0;
883     Real p_a0;
884     Real T_a0;
885     Real m_f1;
886     Real x_air1;
887     Real x_egr1;
888     Real x_vgt1;
889     Real p_t1;
890     Real p_a1;
891     Real T_a1;
892     Modelica.Blocks.Continuous.PI PI(T=2, k=100);
893   equation
894     connect(add4.y, inertiaEngine.M_t);
895
896   algorithm
897     i :=integer(time/(setuptime+holdtime));
898     if (i==0) then
899       (m_f0,x_air0,x_egr0,x_vgt0,p_t0,p_a0,T_a0) := readInOutValues
          (APIndex, "InOutValues");
900     else
901       (m_f0,x_air0,x_egr0,x_vgt0,p_t0,p_a0,T_a0) := readInOutValues
          (APIndex+(i-1)*iter, "InOutValues");
902     end if;
903
904     (m_f1,x_air1,x_egr1,x_vgt1,p_t1,p_a1,T_a1) := readInOutValues(
        APIndex + i*iter, "InOutValues");
905
906       add1.u1 :=readNe(APIndex + i*iter, "InOutValues");
907
908       airpath.m_f :=m_f1;
909       airpath.x_air :=x_air1;
910       airpath.x_egr := 50*sin(time) + 50;//x_egr1;
911       airpath.x_vgt :=x_vgt1;
912       airpath.p_t := p_t1;
913       airpath.p_a :=p_a1;
914       airpath.T_a :=T_a1;
915
916     annotation (Diagram);
917   equation
918
919     connect(PI.y, add4.u2);
920     connect(add1.y, PI.u);
921     connect(add1.u2, inertiaEngine.N_e);
922     connect(airpath.M_t, add4.u1);
923     connect(airpath.N_e, inertiaEngine.N_e);
```

```
924     end AirPathTest;
925   end AirPathCombustionEngineLib;
```

Listing B.1: Luftpfad

Literaturverzeichnis

Baehr, Hans Dieter und Stephan Kabelac [2006]. *Thermodynamik Grundlagen und technische Anwendungen*. Springer. ISBN 3540325131. (zitiert auf Seite 9.)

Becher, Stefan und Torsten Eggert [2004]. *Dieselmotormanagment*. 4. Auflage. Robert Bosch GmbH. ISBN 3528238739. (zitiert auf Seite 2.)

Chauvin, Corde und Petit [2007]. *Transient control of a Diesel engine airpath*. *Proceedings of the 2007 American Control Conference*, Seiten 4394 – 4400. (zitiert auf Seite 13.)

Claytex []. *Symbolic manipulation*. `http://www.claytex.com/`. (zitiert auf Seite 5.)

DieselNet [2012]. *Emission Standards*. `http://www.dieselnet.com/standards/eu/hd.php`. (zitiert auf Seite 2.)

Dittmar, Rainer [2004]. *Modellbasierte prädiktive Regelung*. Oldenburg Verlag München Wien. ISBN 3486275232. (zitiert auf Seite 53.)

Felgner []. *Vergleich numerischer Löser zur Simulation steifer und hybrider Systeme*. `http://www.aut.uni-saarland.de/uploads/media/FF_LL_GF_AUTOMATION_JUNE_2011.pdf`. (zitiert auf Seite 6.)

Fischer, Helmut und Helmut Kaul [2001]. *Mathematik fuer Physiker Band 1 Grundkurs*. B.G.Teubner. ISBN 3519320797. (zitiert auf Seite 53.)

Fritzson, Peter [2003]. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Computer Society Pr. ISBN 0471471631. (zitiert auf den Seiten 3, 5 and 21.)

Guzzella, Lino und Christopher H. Onder [2004]. *Introduction to Modeling and Control of Internal Combustion Engine Systems*. Springer. ISBN 354022274X. (zitiert auf den Seiten 8, 9, 11, 12, 16, 22 and 30.)

Heywood [1988]. *Internal Combustion Engine Fundamentals*. McGraw-Hill. ISBN 0071004998. (zitiert auf Seite 10.)

Jankovic und Kolmanivski [2000]. *Constructive Lyapunov Control Design for Turbocharged Diesel Engines*. *IEEE Transaction on Control System Technology*, 8(2), Seiten 288 – 299. (zitiert auf Seite 10.)

Jung [2003]. *Mean-Value Modelling and Robust Control of the Airpath of a Turbocharged Diesel Engine*. Dissertation, Department of Engineering University Cambirdge, Sidney Sussex College. (zitiert auf den Seiten 12, 14 and 32.)

Maciejowski, Jan [2002]. *Predictive Control with Constraints*. Person Education Limited. ISBN 0201398230. (zitiert auf den Seiten 50 and 54.)

Mathworks [2012]. *Model Predictive Control Toolbox User Guide*. `http://www.mathworks.fr/help/pdf_doc/mpc/mpc_ug.pdf`. (zitiert auf den Seiten 54 and 55.)

Moraal und Kolmanovsky [1999]. *Thurbocharger Modeling for Automotive Control Applications*. *SAE International*, Seite 15. (zitiert auf den Seiten 14 and 31.)

Rueckert, Joachim [2008]. *Modellgestützte Regelung von Ladedruck und Abgasrückführrate beim Dieselmotor*. *Institut für Regelungstechnik RWTH Aachen*. (zitiert auf Seite 41.)

ViF [2011]. *ICOS user manual*. Http://www.vif.tugraz.at. (zitiert auf Seite 47.)