TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Modelica-Simulation einer feldorientiert geregelten Asynchronmaschine mit integrierter Nachführung der Maschinenparameter

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs

unter der Leitung von
Ao.Univ.Prof. Dipl.-Ing. Dr. Thomas Wolbank
Institut für Elektrische Antriebe und Maschinen (E372)

eingereicht an der
Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Emine Calis
Matr.Nr.0025259
Karajangasse 13/18, 1200 Wien

Wien, 30. September 2009 _____

Kurzfassung

Die Regelung der Asynchronmaschine ist heute noch ein hoch aktuelles Thema. Im Zuge zahlreicher Forschungsarbeiten auf diesem Gebiet entstehen stets neue Methoden. Diese werden meistens zum Zweck der Verbesserung der Regelungsqualität durchgeführt. Die Regelverfahren der Asynchronmaschine basieren auf verschiedenen Methoden. Mit dem Entkopplungsverfahren wird der Fluss gesteuert und das Drehmoment geregelt. Die feldorientierte Regelung regelt dagegen sowohl den Fluss als auch das Drehmoment und braucht deswegen die genaue Informationen zum Fluss in der Maschine. Das Parametertracking ist eine der Methoden, die für eine effiziente und genauere feldorientierte Regelung eingesetzt werden.

Mit Hilfe des Parametertrackings können die aktuellen Parameter in der Maschine erfasst, und zur Verbesserung der Regelung herangezogen werden. Die Methoden zur Bestimmung der Maschinenparameter werden entweder online (während des Betriebes) oder offline (im Stillstand) angewandt. Die untersuchte Tracking-Methode wird im Betrieb durchgeführt. Der Vorteil dieser Methode ist, dass sie keine komplexe Algorithmen beinhaltet, in jedem Betriebszustand (stationär und transient) durchführbar ist und die Parameternachführung unabhängig von der feldorientierten Regelung funktioniert. Eine Nachführung der Parameter kann also auch an einer am Netz betriebenen ungeregelten Asynchronmaschine durchgeführt werden.

Die Methode wird in der Modellierungssprache Modelica implementiert. Durch den verschachtelten Aufbau der Modelle in Modelica werden komplexe Systeme vereinfacht und übersichtlicher dargestellt. Die Sprache ist gleichungsbasiert, was zusätzlich zur Vereinfachung der Modelle beiträgt. Im Rahmen der Diplomarbeit werden die in Modelica Bibliotheken vorhandenen Modelle erweitert und zusätzlich neue Modelle entwickelt. Die Simulationsergebnisse sind mit dem Simulationstool Dymola erstellt.

Die Simulationsergebnisse bestätigen die erfolgreiche Nachführung der Parameter. Die aktuellen Parameter der Maschine werden ermittelt, weiters werden sogar absichtlich falsch angegebene Parameter auf richtige Werte nachgeführt. Die dadurch erhöhte Qualität der feldorientierten Regelung der Asynchronmaschine wird im Industriebereich und in weiteren Bereichen viele Vorteile bieten und die Asynchronmaschine noch attraktiver machen.

Schlagwörter: Asynchronmaschine, feldorientierte Regelung, Simulation, Parameternachführung, Modelica, Dymola

Abstract

The control of induction machines has been the main topic of numerous research studies. There are many new research papers about this topic, which are dedicated to improve the quality of the control system. The control systems of induction machines are based on different methods depending on the requested quality. Field Oriented Control (FOC) has emerged as the leading method. Parameter tracking is applied for quality improvement of field oriented control.

Parameter tracking facilitates to identify the instantaneous value of parameters in the machine, which upgrade the performance of the control system. The parameters can either be estimated (offline) or tracked (online). The investigated method works during the operation (online). The method is independent of the operation point, which means that tracking is accurate in every state of operation (steady state and transient). It works independent from control system of the machine. Therefore it is possible to track required parameters with or without FOC. In Addition the method has no complex algorithms that complicate to give a correct description of its process.

The investigated method of parameter tracking is implemented with the modeling language Modelica. In Modelica systems with a large number of components can be simplified by means of nested structures of models. The language system works with equations. This can also be seen as an advantage. Within this work two new models are developed. Two others are adjusted, which already existed in Library. The simulations are performed with Dymola, a simulation tool using Modelica.

The results of the performed simulations show that the parameters are tracked correctly to the accurate values. Although in examples for the parameters consciously false start values were given, they were tracked and adjusted to the correct values. The simulation results proof that the proposed method improves the quality of field oriented control system. It is shown that the parameter tracking accounts for improved control of the induction machine. Thus, the induction machine will be used in the industrial sector more often and will offer further advantages, which will make it more attractive.

Keywords: induction machine, field oriented control, simulation, parameter tracking, Modelica, Dymola

Danksagung

Ich möchte hier an dieser Stelle gerne allen jenen danken, die einen Beitrag zum erfolgreichen Abschluss dieser Arbeit geleistet haben.

Mein besonderer Dank gilt in diesem Zusammenhang meinem Betreuer Prof. Thomas Wolbank für die freundliche Betreuung bei der Ausarbeitung der Diplomarbeit, und das angenehme Gesprächsklima. Herrn Dr. Christian Kral möchte ich für die Unterstützung der Diplomarbeit bei der Firma Austrian Institute of Technology danken.

Weiters gilt mein Dank allen Mitarbeitern von Austrian Institute of Technology, die mich bei dieser Arbeit unterstützt haben. Besonders möchte ich mich beim Johannes Gragger MSc, Dipl.-Ing. Christian Niklas, Dipl.-Ing. Hansjörg Kapeller und Ing. Hubert Umschaden für ihre wertvolle Hilfe bedanken.

Ein herzliches Dankeschön geht an meine Eltern, die mich bei meiner Ausbildung unterstützt und weitergebracht haben.

Zu guter Letzt einen ganz besonderen Dank an meinen Ehemann, meine große Liebe, Zeyd Bilal, der mit viel Geduld meine Launen während des Studiums ertragen hat. Für das Verständnis und die Liebe, die er mir immer wieder entgegen gebracht hat, ein herzliches Danke, ohne ihn wäre diese Diplomarbeit nicht möglich gewesen.

Nomenklatur

Symbole	
$\frac{d}{dt}$, $\frac{d}{d\tau}$	zeitliche Ableitung, bezogene zeitliche Ableitung
f	Frequenz
I, i, $\underline{I}$	Strom, bezogener Strom, Stromraumzeiger
J	Trägheitsmoment
j	imaginäre Einheit
L, l	Induktivität, bezogene Induktivität
$\underline{L}$	komplexe Ersatzgröße
M, m	Drehmoment, bezogenes Drehmoment
n	Rotordrehzahl
p	Polpaarzahl
R, r	Ohm'scher Widerstand, bezogener Ohm'scher Widerstand
s	Schlupf
t, τ	Zeit, bezogene Zeit
T_r	Rotortemperatur
T_s	Statortemperatur
T_R	Rotorzeitkonstante
T_S	Statorzeitkonstante
T_M	Anlaufzeitkonstante
V, v, $\underline{V}$	Spannung, bezogene Spannung, Spannungsraumzeiger
V_m, v_m, $\underline{V}_m$	magnetische Spannung, bezogene magnetische Spannung, magnetischer Spannungsraumzeiger

Indizes	
$1, 2, 3, ...$	Wicklungsstränge der Drehfeldwicklung
a, b	Komponenten einer Zustandsgröße im statorfesten Koordinatensystem (KOS)
d, q	Komponenten einer Zustandsgröße im rotorfesten KOS
x, y	Komponenten einer Zustandsgröße im rotorflussfesten KOS
$BEZUG$	Referenzgröße für die Normierung
N	Nenngröße
f	magnetischer Rotorfluss
r	Rotor
s	Stator
Str	Strang-Scheitelwerte
v	Spannungsmodell
i	Strommodell
σ	Streuung als Bezugsgröße

Griechische Symbole

α	räumlicher Winkel der magnetischen Spannung im statorfesten KOS
γ_f	Rotorflusswinkel bezüglich des statorfesten KOS
γ_m	mechanischer Winkel bezüglich des statorfesten KOS
$\underline{\xi}$	Raumzeiger der Zustandsgröße
$\underline{\xi}^*$	konjugiert komplexer Raumzeiger zu $\underline{\xi}$
σ	Streukoeffizient
φ	räumlicher Winkel des Statorstromraumzeigers im statorfesten KOS
Ψ , ψ, $\underline{\Psi}$	magnetische Flussverkettung, bezogene magnetische Flussverkettung, magnetische Flussverkettungsraumzeiger
ω_f, $\omega_f^{'}$	Winkelgeschwindigkeit des Rotorfeldes, bezogene Winkelgeschwindigkeit des Rotorfeldes (statorfest)
ω_m, $\omega_m^{'}$	mechanische Winkelgeschwindigkeit des Rotors, bezogene mechanische Winkelgeschwindigkeit des Rotors
ω_r, $\omega_r^{'}$	elektrische Winkelgeschwindigkeit des Rotors, bezogene elektrische Winkelgeschwindigkeit des Rotors
ω_s, $\omega_s^{'}$	synchrone Winkelgeschwindigkeit des Drehfeldes, bezogene synchrone Winkelgeschwindigkeit des Drehfeldes

Inhaltsverzeichnis

1 Einleitung

Die Asynchronmaschine ist neben vielen Vorteilen die sie bietet, mit einem relativ großen Aufwand für die Regelung behaftet. Aufgrund der nicht übereinstimmenden Stator- und Rotordrehzahl wird die Erfassung des Flussraumzeigers erschwert. Die feldorientierte Regelung setzt die richtige Flusserfassung voraus. Mit dem Ziel die Qualität der Regelung zu verbessern werden Methoden eingesetzt, die die aktuellen Parameter an der Maschine nachführen.

Die der Diplomarbeit zugrunde liegende Aufgabe ist die Implementierung der Parameternachführung in Modelica. Es wird für das Parametertracking eine Methode untersucht, die auf Modellvergleichen während des Betriebes basiert. Die Flussmodelle bestimmen mit Hilfe der gemessenen Größen wie komplexer Statorstrom $\underline{I}_s$, komplexe Statorspannung $\underline{V}_s$ und mechanische Winkelgeschwindigkeit ω_m die magnetischen Flüsse im Rotor. Ausgehend von den magnetischen Flüssen werden die Parameter basierend auf der eingesetzten Methoden nachgeführt.

Modelica hat eine spezielle Bibliothek für elektrische Antriebe *(Smart Electric Drives Library)*. Diese beinhaltet Hilfskomponenten für die Untersuchung der Parameternachführung. Das für die Parameternachführung in Modelica entwickelte Modell heisst *Tracking*. Die Flussmodelle Spannungsmodell und Strommodell *(Currentmodel* und *Voltagemodel)* und das Modell der feldorientierten Regelung *(FOC)* waren in Modelica schon vorhanden. Das vorhandene Spannungsmodell und Strommodell haben aber nicht mit nachgeführten Parametern gearbeitet. Die beiden Modelle sind im Rahmen der Diplomarbeit so erweitert, dass sie mit aktuellen vom *Tracking* nachgeführten Maschinenparametern arbeiten. Die bei festen Parametern vernachlässigten Wirkungen, wie Temperaturänderungen und Sättigungseffekte, werden dadurch berücksichtigt.

Es soll ein kurzer Überblick der Kapitel der Diplomarbeit gegeben werden, in welchen im Detail auf die zu behandelten Themen eingegangen wird.

In Abschnitt 2 wird die Asynchronmaschine vorgestellt und das zugehörige Raumzeigerkalkül beschrieben. Die feldorientierte Regelung wird erklärt und die Bedeutung der zugehörigen Parameternachführung wird beschrieben. Am Schluss werden die unterschiedlichen Methoden der Parameternachführung bekanntgegeben, und die für die Diplomarbeit untersuchte Methode wird ausführlich erläutert.

Abschnitt 3 gibt einen Überblick über die Eigenschaften der Modellierungssprache Modelica. Die Bibliothek *Smart Electric Drives*; die die Modelle für die feldorientierte Regelung zur Verfügung stellt und für die die neuen Modelle entwickelt wurden bzw. Modellerweiterungen gemacht wurden, wird detailliert beschrieben. Abschliessend wird die vorhandene Antriebsstruktur, ohne Tracking der Parameter, vorgestellt und die notwendige Modellerweiterungen für das Parametertracking werden bekanntgegeben.

Abschnitt 4 stellt den Modellaufbau und die neu entwickelten Modelle vor. Es werden die neuen Komponenten und deren Aufgaben im Parametertracking der feldorientierten Regelung der Asynchronmaschine erläutert.

In Abschnitt 5 wird das Simulationstool Dymola vorgestellt. Die Funktionsfähigkeit der Modelle wird anhand von unterschiedlichen Simulationsbeispielen untersucht. Die Auswirkungen der falschen

Maschinenparameter auf die Regelung werden dokumentiert, mit dem Ziel die Notwendigkeit des Parametertrackings zu zeigen. Die Asynchronmaschine wird sowohl im normalen Netzbetrieb als auch im geregelten Betrieb untersucht. Anhand der Simulationsergebnisse wird der Verlauf des Parametertracking dargestellt.

Abschnitt 6 gibt eine Zusammenfassung der Arbeit wieder und erläutert die Vorteile der feldorientierten Regelung der Asynchronmaschine mit integrierter Nachführung der Maschinenparameter.

2 Grundlagen und Stand der Technik

Die Drehstrom-Asynchronmaschine spielt als Antriebsmotor eine führende Rolle. Ihre Wirkungsweise beruht auf der Entstehung eines Drehfeldes durch eine mehrsträngige Wicklung. Die Asynchronmaschine wird entweder mit Schleifring- oder Käfigläufer gefertigt. Letztere hat wegen des fehlenden Kommutators gegenüber der Gleichstrommaschine den Vorteil des wesentlich einfacheren und robusteren konstruktiven Aufbaus. Damit bedarf die Maschine geringer Wartung und ist preisgünstiger. Die Einsatzgebiete der Asynchronmaschine sind sehr vielfältig und in der Ausführung mit Käfigläufer ist sie der verbreitetste Antrieb weltweit. Die in dieser Arbeit untersuchte Asynchronmaschine ist mit Käfigläufer ausgeführt.

Die ausgeführte Leistung reicht in serienmäßigen Ausführungen von einigen Watt bis in den MW-Bereich. Die ausführbare Grenzleistung für die Asynchronmaschine steigt proportional mit der Polzahl und liegt bei Luftkühlung für vierpolige Motoren bei ca. 30 MW. Der Nachteil ist die enge Bindung der Betriebsdrehzahl an die Frequenz der Ständerspannung. Im 50 Hz-Netzbetrieb sind wegen der Abhängigkeit der Drehzahl von der Netzfrequenz und der Polpaarzahl $n = f/p$ nur eingeschränkte Drehzahlbereiche erreichbar [9]. Das macht die Maschine für Antriebe mit variablen Drehzahlen ungeeignet, solange sie ohne Frequenzumrichter eingesetzt ist. Die Frequenzumrichter ermöglichen Verfahren zur verlustarmen Drehzahlsteuerung. Die Fortschritte in der Leistungselektronik im Bereich abschaltbarer Leistungshalbleiter und in der Elektronik durch digitale Signalprozessoren ermöglichen die exakte Regelung der Drehzahl und des Drehmoments. Ausgehend vom allgemeinen Signalflussplan der Asynchronmaschine ist es möglich, die Drehmoment- und Drehzahlregelung durchzuführen. Die messtechnisch nicht zugänglichen internen Größen der Maschine können mit unterschiedlichen Maschinenmodellen berechnet werden. Zur Regelung der Asynchronmaschine existieren verschiedene Möglichkeiten. Nur den Fluss zu steuern und das Drehmoment zu regeln ist eine Möglichkeit (Entkopplungverfahren), wodurch eine aufwändige Schätzung der Orientierung des Flusses vermieden wird. Die feldorientierte Regelung braucht dagegen sowohl die Orientierung als auch die Amplitude des Flusses in der Maschine, und kann durch die Nachführung der Parameter effizienter arbeiten. Durch die feldorientierte Regelung kann somit die Asynchronmaschine so hochdynamisch wie eine Gleichstrommaschine betrieben werden [26]. Im Folgenden wird ein kurzer Überblick über die Asynchronmaschine und die feldorientierte Regelung gegeben. Abschließend wird die Parameternachführung der Asynchronmaschine erläutert.

2.1 Asynchronmaschine

Der Ständer (Stator) einer Asynchronmaschine besteht aus einem Ständergehäuse, in dem das Ständerblechpaket befestigt ist, in den Nuten des Blechpakets befindet sich die Drehstromwicklung. Die Ständerwicklungsstränge sind symmetrisch am Maschinenumfang verteilt und werden entweder in Stern oder in Dreieck geschaltet. Der Läufer trägt ebenfalls ein Blechpaket mit Nuten zur Aufnahme der Läuferwicklung. Der Schleifringläufer trägt wie der Ständer in seinen Nuten eine Drehstromwicklung. Die drei Wicklungsanfänge sind über Schleifringe herausgeführt und die Enden intern verbunden. Beim Käfigläufer ist die Läuferwicklung nicht zugänglich. Die Nuten sind mit

einem Profilstab aus Kupfer, Bronze oder Aluminium ausgefüllt und die Leiter auf beiden Läuferenden über sogenannte Kurzschlussringe miteinander verbunden. Das Ständerwicklungssystem soll mit einem symmetrischen Dreiphasen-Spannungssystem gespeist werden, damit sich ein Drehfeld im Luftspalt der Asynchronmaschine bildet. Die Abbildung 2.1 veranschaulicht eine Asynchronmaschine mit Käfigläufer mit gegossenen Aluminium-Profilstäben.

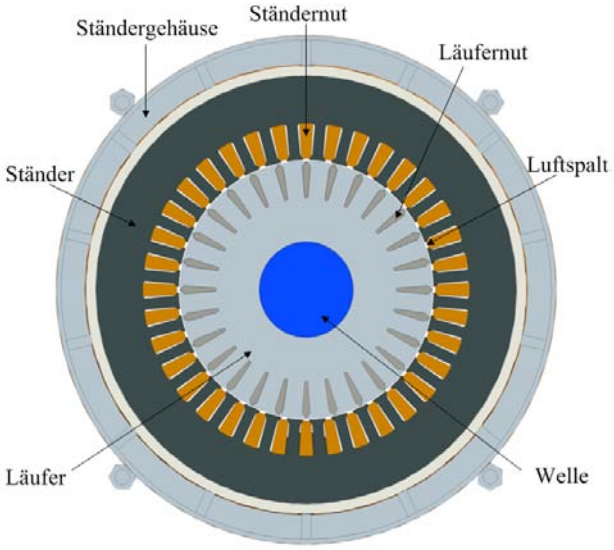

Abb. 2.1: Asynchronmaschine mit Käfigläufer

Werden die Statorwicklungen an ein Drehspannungssystem gelegt, bilden sich in den Strängen des Stators Ströme aus, die dann jeweils eine sowohl zeitlich als auch räumlich verschobene Durchflutung aufbauen. Diese Durchflutung bildet eine magnetische Flussverkettung bezüglich des Rotors mit der Synchrondrehzahl $n_s = f/p$ aus. Die zeitliche Veränderung der Flussverkettung induziert in den Leitern des Rotors eine Spannung, die im Rotor Reaktionsströme hervorruft, welche zusammen mit dem Drehfeld ein Drehmoment bewirken. Nach dem Lenz'schen Gesetzt läuft der Rotor in Drehfeldrichtung an, um der Ursache der Induktion entgegenzuwirken, indem er den relativen Unterschied zwischen Winkelgeschwindigkeit des Feldes $\omega_s = 2 \cdot \pi \cdot f_s$ und der mechanische Winkelgeschwindigkeit des Rotors $\omega_m = 2 \cdot \pi \cdot n \cdot p$ verringert. Im Falle $\omega_s = \omega_m$ sind die zeitliche Veränderung der Flussverkettung und somit die Reaktionsströme im Rotor, und das Drehmoment Null. Auf Grund von Reibungsverlusten in der Maschine kann dieser Zustand aber nicht erreicht werden. Der relative Unterschied zwischen der Winkelgeschwindigkeit der Flussverkettung ω_s und der mechanischen Winkelgeschwindigkeit des Rotors ω_m wird als Schlupf s bezeichnet und gemäß

$$s = \frac{\omega_s - \omega_m}{\omega_s} \tag{2.1}$$

berechnet. Die Beschreibung der physikalischen Vorgänge erfolgt durch Raumzeiger-Differentialgleichungen, die im Unterabschnitt 2.1.3 explizit erläutert werden.

2.1.1 Normierung

Die Normierung ist der Bezug einer physikalischen, dimensionsbehafteten Größe auf eine Referenzgröße gleicher Dimension. Die dadurch entstehende Größe hat somit die Dimension 1. Die Normierung der Größen bringt den Vorteil mit sich, dass danach der Zahlenwert dieser physikalischen Größe sich in einem sehr engen Bereich bewegt [27].

Die folgenden Referenzgrößen werden für die Normierung eingesetzt:

Die Normierung der Zeit wird mit Hilfe der Nenn- bzw. Bezugs-Kreisfrequenz

$$\tau = t \cdot \omega_{\mathrm{BEZUG}} \tag{2.2}$$

durchgeführt, wobei ω_{BEZUG} wie folgt definiert ist:

$$\omega_{\mathrm{BEZUG}} = 2 \cdot \pi \cdot f_N \tag{2.3}$$

Die mechanische Drehzahl wird unter Berücksichtigung der Polpaarzahl mit der Gleichung (2.4) normiert

$$\omega_m^{'} = \frac{\omega_m}{\frac{\omega_{\mathrm{BEZUG}}}{p}} \tag{2.4}$$

Spannungen und Ströme werden auf ihre Nennstrang-Scheitelwerte bezogen:

$$v(t) = \frac{V(t)}{V_{\mathrm{BEZUG}}} \tag{2.5}$$

$$i(t) = \frac{I(t)}{I_{\mathrm{BEZUG}}} \tag{2.6}$$

Dabei sind $V_{\mathrm{BEZUG}} = \sqrt{2} \cdot V_{N,Str}$ und $I_{\mathrm{BEZUG}} = \sqrt{2} \cdot I_{N,Str}$.

Flussverkettung:

$$\psi = \frac{\Psi}{\frac{V_{\mathrm{BEZUG}}}{\omega_{\mathrm{BEZUG}}}} \tag{2.7}$$

Impedanzen werden auf $Z_{\mathrm{BEZUG}} = \frac{V_{\mathrm{BEZUG}}}{I_{\mathrm{BEZUG}}}$ normiert, somit ist

$$z = \frac{Z}{Z_{\mathrm{BEZUG}}} \tag{2.8}$$

Das Moment wird auf das Nennmoment $M_{\mathrm{BEZUG}} = \frac{3 \cdot V_{\mathrm{BEZUG}} \cdot I_{\mathrm{BEZUG}}}{2 \cdot \omega_{\mathrm{BEZUG}}}$ bezogen, und das bezogene Moment ist

$$m = \frac{M}{M_{\mathrm{BEZUG}}} \tag{2.9}$$

2.1.2 Raumzeiger

Die komplexe Raumzeigerrechnung ermöglicht eine Beschreibung des transienten Verhaltens von Drehfeldmaschinen. Der Vorteil der Raumzeigerrechnung ist, dass im Gegensatz zur komplexen Zeitzeigerrechnung der zeitlichen Verlauf der Größen keine Rolle spielt. Die Lage und Größe der Raumzeiger charakterisiert den augenblicklichen, räumlichen elektromagnetischen Zustand der Maschine. Die Raumzeiger werden aus den Augenblickswerten der Stranggrößen berechnet [27]. Bezüglich der Raumzeigerrechnung werden folgende Annahmen getroffen:

- Die Statorwicklung ist dreisträngig
- Die Grundwellen der Strombeläge und der magnetischen Flussdichte im Luftspalt sind so dominant, dass die von Oberwellen verursachten Effekte vernachlässigt werden können. Daher können die elektromagnetischen Größen in der Maschine als räumlich sinusförmig verteilt angenommen werden.
- Die Maschine befindet sich in einem Arbeitspunkt, wo keine Sättigung des magnetischen Kreises vorliegt. Bei vorliegen stärkerer Sättigungen kann als Näherung ein durch den magnetischen Arbeitspunkt verlaufendes lineares Ersatzsystem angesetzt werden.
- Es wird ein rotationssymmetrisch aufgebauter Magnetkreis vorausgesetzt, um symmetrische Induktivitäten zu erhalten.
- Der Rotor wird als dreisträngige Ersatzwicklung modelliert.

Die Abbildung 2.2 veranschaulicht die Darstellung des Raumzeigers. In den geometrischen Mittelpunkt der Maschine wird ein komplexes Koordinatensystem gelegt. Die sinusförmig verteilten elektromagnetischen Größen in der Maschine ermöglichen eine Darstellung durch einen Zeiger in diesem Koordinatensystem. Der Raumzeiger zeigt in die Richtung, wo die elektromagnetische Größe ihren maximal Wert hat. Der Stromraumzeiger $\underline{I}$ repräsentiert die Richtung und Amplitude der magnetischen Spannung $V_m(\alpha)$ in der Maschine, die Spannungsraumzeiger $\underline{V}$ und Flussverkettungsraumzeiger $\underline{\Psi}$ können aber nicht physikalisch interpretiert werden. Diese Größen ergeben sich aus mathematischen Beziehungen.

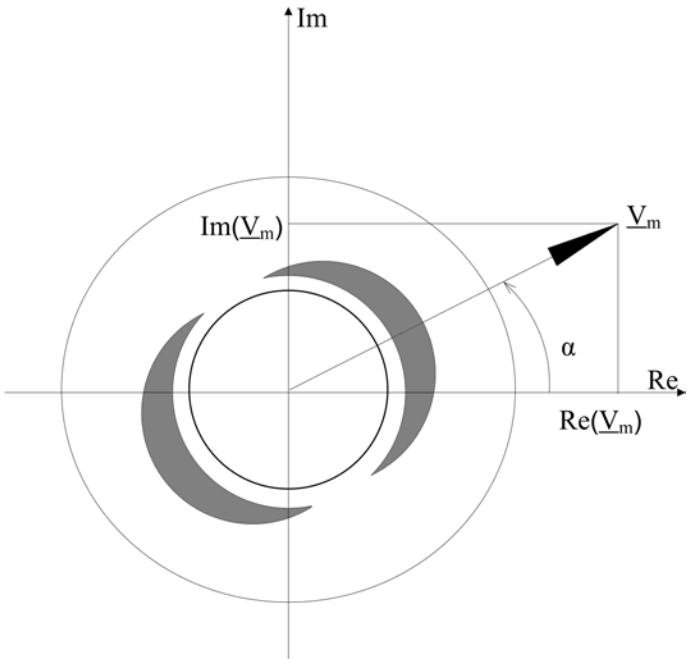

Abb. 2.2: Definition eines Raumzeigers

- **Stromraumzeiger:** Die Stränge der Maschine sind, unter der Berücksichtigung der Strangzahl, um $2\pi/3$ versetzt. Die Augenblickswerte der magnetischen Spannungen werden durch V_{m1}, V_{m2}, V_{m3} repräsentiert, die wie folgt berechnet werden:

$$V_{m1}(\alpha) = V_{m1} \cdot \cos(\alpha) = \mathrm{Re}(V_{m1} \cdot e^{j\alpha}) \tag{2.10}$$

$$V_{m2}(\alpha) = V_{m2} \cdot \cos(\alpha + \frac{2\pi}{3}) = \mathrm{Re}(V_{m2} \cdot e^{j\alpha} \cdot e^{j\frac{2\pi}{3}}) \tag{2.11}$$

$$V_{m3}(\alpha) = V_{m3} \cdot \cos(\alpha + \frac{4\pi}{3}) = \mathrm{Re}(V_{m3} \cdot e^{j\alpha} \cdot e^{j\frac{4\pi}{3}}) \tag{2.12}$$

Es ergibt sich für die magnetische Gesamtspannungsverteilung:

$$V_{m,\mathrm{ges}} = V_{m1}(\alpha) + V_{m2}(\alpha) + V_{m3}(\alpha) = \mathrm{Re}((V_{m1} + V_{m2} \cdot e^{j\frac{2\pi}{3}} + V_{m3} \cdot e^{j\frac{4\pi}{3}}) \cdot e^{j\alpha}) \tag{2.13}$$

Der komplexe Raumzeiger der magnetischen Spannung wird somit bestimmt:

$$\underline{V}_m = \frac{2}{3} \cdot (V_{m1} + V_{m2} \cdot e^{j\frac{2\pi}{3}} + V_{m3} \cdot e^{j\frac{4\pi}{3}}) \tag{2.14}$$

Die Gleichung der magnetischen Gesamtspannung (2.13) umgeformt mit dem Raumzeiger der magnetischen Spannung:

$$V_{m,\mathrm{ges}}(\alpha) = \mathrm{Re}(\frac{3}{2} \cdot \underline{V}_m \cdot e^{j\alpha}) \tag{2.15}$$

Als Spezialfall wird für die Speisung der Maschine ein symmetrisches Drehstromsystem betrachtet. Die magnetische Spannungen werden in diesem Fall wie folgt identifiziert:

$$V_{m1} = \hat{V} \cdot \cos(\omega t) \tag{2.16}$$

$$V_{m2} = \hat{V} \cdot \cos(\omega t - \frac{2\pi}{3}) \tag{2.17}$$

$$V_{m3} = \hat{V} \cdot \cos(\omega t - \frac{4\pi}{3}) \tag{2.18}$$

Das ermöglicht eine Darstellung des Raumzeigers ausgehend der Gleichung (2.14), mit Hilfe der obigen Gleichungen:

$$\underline{V}_m = \frac{2}{3} \cdot \widehat{V} \cdot (\cos(\omega t) + \cos(\omega t - \frac{2\pi}{3}) \cdot e^{j\frac{2\pi}{3}} + \cos(\omega t - \frac{4\pi}{3}) \cdot e^{j\frac{4\pi}{3}}) \tag{2.19}$$

Die Amplitude des Raumzeigers ist identisch mit der Amplitude von Phasengrößen. Analog zu Gleichung (2.15) kann eine Vereinfachung durchgeführt werden und somit wird für $\underline{V}_m$

$$\underline{V}_m = \frac{2}{3} \cdot (\frac{3}{2} \cdot \hat{V} \cdot e^{j\omega t}) = \hat{V} \cdot e^{j\omega t} \tag{2.20}$$

Ausgehend vom magnetischen Spannungsraumzeiger kann der Stromraumzeiger bestimmt werden. Der Strangstrom und der Scheitelwert der von ihm hervorgerufenen magnetischen Spannung sind zueinander proportional. Mit Hilfe dieser Beziehung und weiter folgender Information kann der Stromraumzeiger

$$V_{m_k} = const \cdot I_k \tag{2.21}$$

bestimmt werden. Bei der Erfassung von Zeigern muss immer ein Koordinatensystem angegeben werden, welches als Bezugssystem herangezogen wird. Das statorfeste Koordinatensystem ist mit dem Stator fest verbunden. Im statorfesten Koordinatensystem liegt die reelle Achse des Koordinatensystems in Richtung der Wicklungsachse des ersten Stranges. Die Raumzeiger des statorfesten Koordinatensystems werden mit hochgestelltem s indiziert, dessen Realteil mit dem Index a und Imaginärteil mit dem Index b bezeichnet wird. Die Achsen des rotorfesten Koordinatensystems sind mit dem Rotor der Maschine fest verbunden und werden mit d (reelle Achse) und q (imaginäre Achse) bezeichnet. Das rotorfeste Koordinatensystem wird mit hochgestelltem r indiziert. Beim flussfesten Koordinatensystemen, abhängig davon auf welchen Flussraumzeiger das Koordinatensystem bezogen wird, liegt die reelle Achse des Koordinatensystems in Richtung des jeweiligen Flusses. Ein hochgestelltes f wird als Indizierung des rotorflussfesten Koordinatensystems verwendet, wobei x (reelle Achse) und y (imaginäre Achse) die Indizes der Achsen sind. Die Gleichung (2.22) und die Abbildung 2.3 stellen den Statorstromraumzeiger in verschiedenen Koordinatensystemen dar:

$$\begin{aligned} \underline{I}_s^s &= I_a + j \cdot I_b = \underline{I}_s \cdot e^{j\varphi} \\ \underline{I}_s^r &= I_d + j \cdot I_q = \underline{I}_s^s \cdot e^{-j\gamma_m} \\ \underline{I}_s^f &= I_x + j \cdot I_y = \underline{I}_s^s \cdot e^{-j\gamma_f} \end{aligned} \tag{2.22}$$

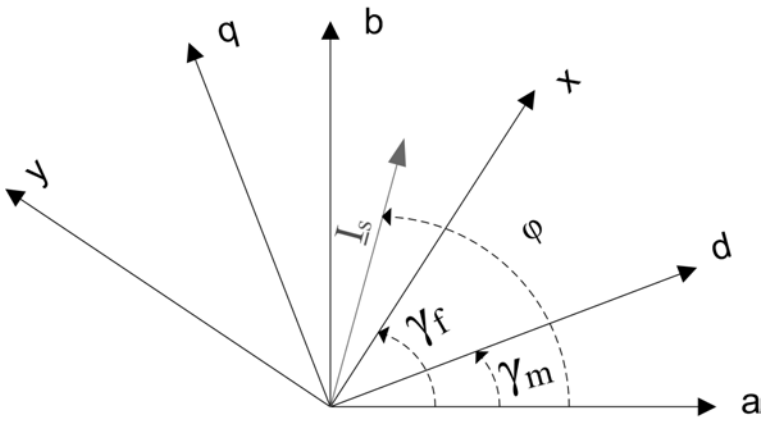

Abb. 2.3: Lage des Statorstromraumzeigers in unterschiedlichen Koordinatensystemen

Mit Hilfe der Beziehung (2.21) zwischen dem Strom und der magnetischen Spannung und der Gleichung (2.15) kann der Stator-Stromraumzeiger

$$\underline{I}_s^s = \frac{2}{3} \cdot (I_1 + I_2 \cdot e^{j\frac{2\pi}{3}} + I_3 \cdot e^{j\frac{4\pi}{3}}) \tag{2.23}$$

ausgerechnet werden.

- **Spannungsraumzeiger** $\underline{V}$ und **Flussverkettungsraumzeiger** $\underline{\Psi}$ werden rein formal definiert, weil diesen keine physikalisch anschauliche Interpretation zugedacht werden kann. Der Spannungsraumzeiger ergibt sich aus den Strangspannungen V_k zu

$$\underline{V}_s^s = \frac{2}{3} \cdot (V_1 + V_2 \cdot e^{j\frac{2\pi}{3}} + V_3 \cdot e^{j\frac{4\pi}{3}}) \tag{2.24}$$

Der Flussverkettungsraumzeiger ist nicht direkt zugänglich und wird durch Integration des korrespondierenden Spannungsraumzeigers berechnet.

$$\underline{\Psi}_s^s(t) = \Psi_s^s(0) + \int_0^t (V_s^s(t') - R_s \cdot I_s^s(t'))dt' \tag{2.25}$$

Nullgrößen haben keine Auswirkung auf die Bildung von Raumzeigern. Wenn zu jeder Strangspannung ein konstanter Wert addiert wird, dann verschiebt sich auch das Bezugspotential um diese Nullspannung gegenüber dem Sternpunkt. Bezüglich welchem Potential die Strangspannungen erfasst werden ist aber unwesentlich. Die Nullspannungen und Nullströme haben somit keine Auswirkung auf die Bildung des Raumzeigers. Dem Zufolge geht die Information über diese Nullgröße nach der Bildung des Raumzeigers verloren. Und es kann aus dem Raumzeiger nicht mehr auf Nullgrößen geschlossen werden.

Die Rückrechnung von Raumzeigern auf Strangwerte erfolgt durch die Projektion des Raumzeigers auf die betrachtete Strangachse. Der Raumzeiger als komplexer Vektor hat eine Länge und einen Winkel mit der reellen-Achse des statorfesten-Koordinatensystems. Die Strangströme werden dann wie folgt ausgerechnet:

$$I_1 = \mathrm{Re}(\underline{I}_s^s) \qquad I_2 = \mathrm{Re}(\underline{I}_s^s \cdot e^{-j\frac{2\pi}{3}}) \qquad I_3 = \mathrm{Re}(\underline{I}_s^s \cdot e^{-j\frac{4\pi}{3}}) \tag{2.26}$$

2.1.3 Mathematische Beschreibung der Asynchronmaschine

Das elektromagnetische Verhalten der Asynchronmaschine wird mit Raumzeiger-Differentialgleichungen beschrieben. Dies bietet den Vorteil, dass keine einschränkenden Voraussetzungen über den zeitlichen Verlauf der Größen bestehen. Die physikalischen Größen werden als Raumzeiger in unterschiedlichen Koordinatensystemen angegeben, je nachdem aus welchem System die Größen betrachtet werden. Ein wichtiger Begriff für die Asynchronmaschine ist die Streuung. Um die mathematischen Gleichungen der Maschine zu vereinfachen wird eine ungeteilte Gesamtstreuung der Maschine angenommen, welche dem Stator zugeordnet wird. Damit wird der Rotor streuungslos modelliert [19]. Eine charakteristische Größe für die Streuung in der Maschinen ist der Streukoeffizient σ, der zur Beschreibung der nicht idealen Kopplung zwischen Stator und Rotor eingeführt wird.

$$\sigma = 1 - \frac{L_m^{*2}}{(L_{s\sigma}^* + L_m^*) \cdot (L_{r\sigma}'^* + L_m^*)} \tag{2.27}$$

Das allgemeine Ersatzschaltbild der Asynchronmaschine ohne Zuordnung der Streuung auf die Statorseite ist in der Abbildung 2.4 dargestellt. Die rotorseitige Grössen sind auf eine äquivalente Statorwicklung umgerechnet.

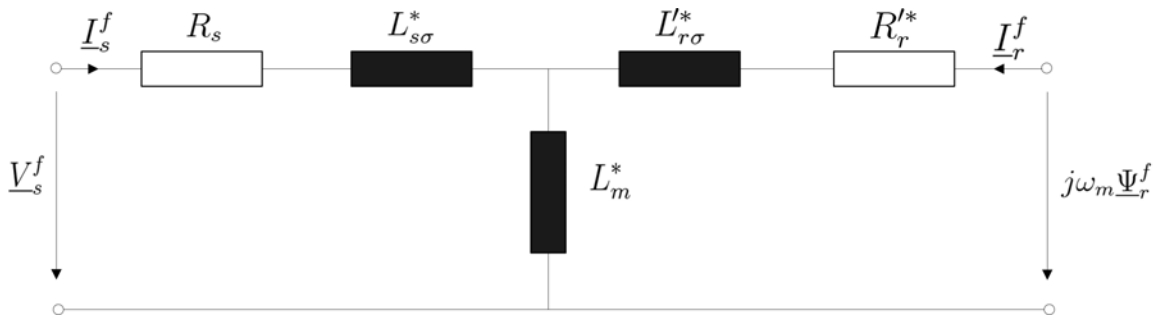

Abb. 2.4: Allgemeines Ersatzschaltbild der Asynchronmaschine im stationären Zustand

Die Zuordnung der Rotorstreuung auf die Statorseite, erfolgt mit Hilfe mathematischer Beziehungen [20], die im weiteren erläutert werden:

Die statorseitige Gesamtinduktivität wird durch Addition der Statorstreuinduktivität und Hauptfeldinduktivität berechnet und von der Zuordnung der Streuung nicht beeinflusst.

$$L_s^* = L_{s\sigma}^* + L_m^* = L_s \tag{2.28}$$

Die Streuinduktivität vom Stator nach der Zuordnung der Rotorstreung auf die Statorseite ist L_σ und wird wie folgt berechnet:

$$L_{s\sigma} = \sigma \cdot L_s^* \tag{2.29}$$

$$L_\sigma = L_{s\sigma} \tag{2.30}$$

Nach der Zuordnung ist die Rotorseite streuungslos und somit befindet sich die gesamte rotorseitige Induktivität im Hauptfeld der Maschine.

$$L_{r\sigma} = 0 \tag{2.31}$$

Mit der statorseitigen Induktivität aus (2.28), ergibt sich mit der Gleichung (2.27) der Streukoeffizient σ für die Hauptfeldinduktivität bzw. rotorseitige Induktivität:

$$L_m \quad = \quad L_r = L_s \cdot (1 - \sigma) \tag{2.32}$$

Der Statorwiderstand R_s wird von der Transformation nicht beeinflusst:

$$R_s = R_s^* \tag{2.33}$$

Es wird weiters angenommen, dass die Rotorgrößen auf eine äquivalente Statorwicklung umgerechnet werden. Die rotorseitige Induktivität vor der Zuordnung der Rotorstreuung auf die Statorseite wird wie folgt definiert:

$$L_r^* = L_{r\sigma}^{'*} + L_m^* \tag{2.34}$$

Ausgehend von der Transformation des Rotorstroms

$$\underline{I}_r = \underline{I}_r^* \cdot \frac{L_r^*}{L_m^*} \tag{2.35}$$

kann der Rotorwiderstand wie folgt berechnet werden:

$$R_r = R_r^{'*} \cdot \frac{L_m^{*2}}{L_r^{*2}} \tag{2.36}$$

Die vereinfachte Ersatzschaltung der Asynchronmaschine ist in Abb 2.5 dargestellt.

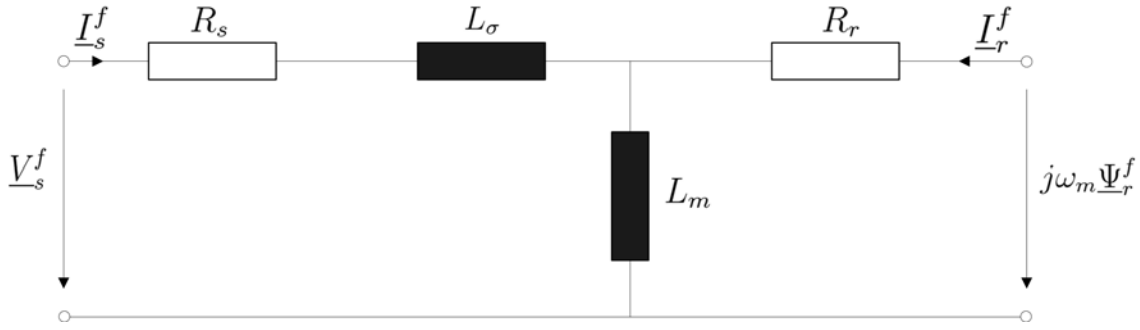

Abb. 2.5: Vereinfachtes Ersatzschaltbild der Asynchronmaschine im stationären Zustand

Die grundlegenden Gleichungen der Asynchronmaschine können wie folgt angegeben werden:

- **Rotorspannungsgleichung:** Im rotorfesten Koordinatensystem lautet die Rotorspannungsgleichung

$$\underline{V}_r^r = \underline{I}_r^r \cdot R_r + \frac{d\underline{\Psi}_r^r}{dt} \,. \tag{2.37}$$

  Beim Kurzschlussläufer wird die Rotorspannung wegen den kurzgeschlossenen Läuferenden zu Null $\underline{V}_r^r = 0$. Damit reduziert sich die Gleichung (2.37) auf:

$$\frac{d\underline{\Psi}_r^r}{dt} = -\underline{I}_r^r \cdot R_r \tag{2.38}$$

  Bei der Transformation der Rotorspannungsgleichung vom rotor- in das statorfeste Koordinatensystem, tritt zufolge der Winkeldifferenz zwischen Stator- und Rotorkoordinatensystem und des Differentialoperators in der Gleichung (2.37) der Term $-j\omega_m \cdot \underline{\Psi}_r^s$

$$\underline{V}_r^s = \underline{I}_r^s \cdot R_r + \frac{d\underline{\Psi}_r^s}{dt} - j\omega_m \cdot \underline{\Psi}_r^s \,, \tag{2.39}$$

wobei:

$$\omega_m = d\gamma_m/dt \tag{2.40}$$

ist.
Analog dazu kann die Gleichung in weitere Koordinatensysteme transformiert werden.

- **Statorspannungsgleichung:** Im statorfesten Koordinatensystem gilt für die Statorspannung

$$\underline{V}_s^s = \underline{I}_s^s \cdot R_s + \frac{d\underline{\Psi}_s^s}{dt} \tag{2.41}$$

  Die Statorspannungsgleichung in das rotorfeste Koordinatensystem transformiert ergibt:

$$\underline{V}_s^r = \underline{I}_s^r \cdot R_s + \frac{d\underline{\Psi}_s^r}{dt} + j\omega_m \cdot \underline{\Psi}_s^r \tag{2.42}$$

- **Flussverkettungsgleichungen:** Das betrachtete Koordinatensystem ist bei der Berechnung der Flussverkettungen, wegen des fehlenden Differentialanteils, ohne Bedeutung. Die folgenden Gleichungen werden im rotorflussfesten Koordinatensystem angegeben. Die Flussverkettungsraumzeiger des Stators und Rotors lauten:

$$\underline{\Psi}_s^f = L_s \cdot \underline{I}_s^f + L_m \cdot \underline{I}_r^f \tag{2.43}$$

$$\underline{\Psi}_r^f = L_m \cdot \underline{I}_s^f + L_r \cdot \underline{I}_r^f \tag{2.44}$$

  Die Gleichung (2.32) verhilft zur Vereinfachung der beiden Flussverkettungsgleichungen:

$$\underline{\Psi}_s^f = L_s \cdot (\underline{I}_s^f + (1-\sigma) \cdot \underline{I}_r^f) \tag{2.45}$$

$$\underline{\Psi}_r^f = L_s \cdot (1-\sigma) \cdot (\underline{I}_r^f + \underline{I}_s^f) \tag{2.46}$$

Die beiden Flussverkettungen unterscheiden sich somit um den Faktor $\sigma L_s \cdot \underline{I}_s^f$:

$$\underline{\Psi}_s^f = \underline{\Psi}_r^f + \sigma L_s \cdot \underline{I}_s^f = \underline{\Psi}_r^f + L_\sigma \cdot \underline{I}_s^f \tag{2.47}$$

- **Das Drehmoment:** Der Statorstromraumzeiger und Rotorflussraumzeiger spannen eine Fläche (Abbildung 2.6), die proportional zum inneren Drehmoment ist. Die Abbildung 2.6 veranschaulicht diesen Zusammenhang aber indirekt, indem die aufgespannte Fläche nicht proportional zu $\underline{\Psi}_r \cdot \underline{I}_s$ sondern zu $\underline{\Psi}_r \cdot \underline{I}_{sy}$ angegeben wird, wobei die beiden Flächen mathematisch identisch sind. Das größte Moment würde erzielt werden, wenn der Statorstromraumzeiger normal zum Flussverkettungsraumzeiger läge. Welche Flussverkettungsraumzeiger zur Berechnung herangezogen wird ist unerheblich, weil der Raumzeiger des Streuflusses parallel zum Statorstromraumzeiger liegt. In welchem Koordinatensystem das Drehmoment berechnet wird macht ebenfalls keinen Unterschied, weshalb die Gleichung ohne Bezugskoordinatensystem angegeben wird

$$M_i = -\frac{3}{2} \cdot p \cdot \mathrm{Im}(\underline{I}_s^* \cdot \underline{\Psi}_r) \tag{2.48}$$

Die Differenz zwischen innerem und aussen angreifendem Moment verursacht eine Änderung der mechanischen Winkelgeschwindigkeit ω_m.

$$M_i - M_{Last} = J \cdot \frac{d\omega_m}{dt} \tag{2.49}$$

Die Gleichung 2.49 dient zur Beschreibung des mechanischen Systems der Maschine. Der Rotor erreicht seine Nenndrehzahl nach der Anlaufzeitkonstante T_M.

$$T_M = \frac{J \cdot \omega_N}{M_N} \tag{2.50}$$

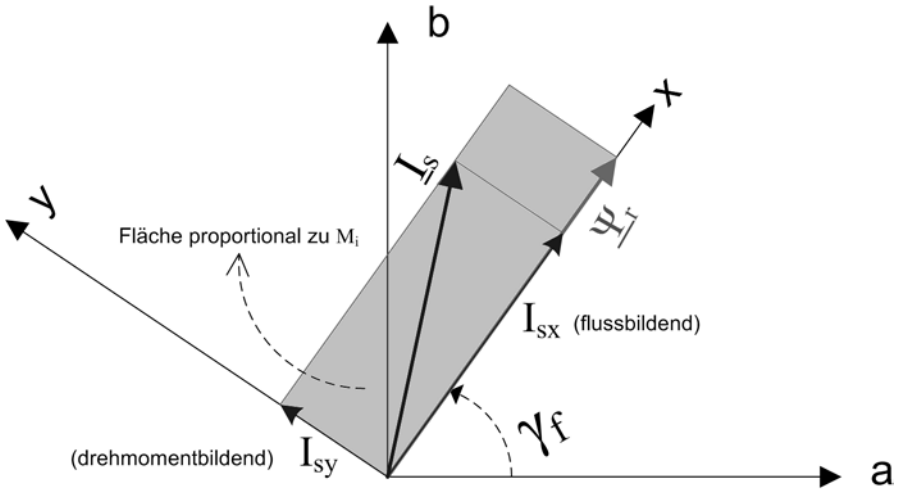

Abb. 2.6: Drehmomentbildung in der Asynchronmaschine

2.2 Feldorientierte Regelung

Die feldorientierte Regelung ermöglicht einen hochdynamischen Betrieb der Asynchronmaschine. Die Grundidee dabei ist, ein gefordertes inneres Moment durch einen an einem Flussverkettungsraumzeiger orientierten Stromraumzeiger einzustellen. Das Problem ist aber, dass dieser Flussverkettungsraumzeiger unbekannt ist, weil der Rotor nicht synchron mit dem Stator läuft und deshalb der Flussraumzeiger nicht mit der Rotorlage zusammenhängt. Außerdem kann der Rotor nicht von sich aus magnetisch erregt werden. Es muss dazu ein Fluss vom Stator her aufgebaut werden. Die Abbildung 2.7 gibt das Strukturbild einer feldorientierten Regelung der Asynchronmaschine an [27].

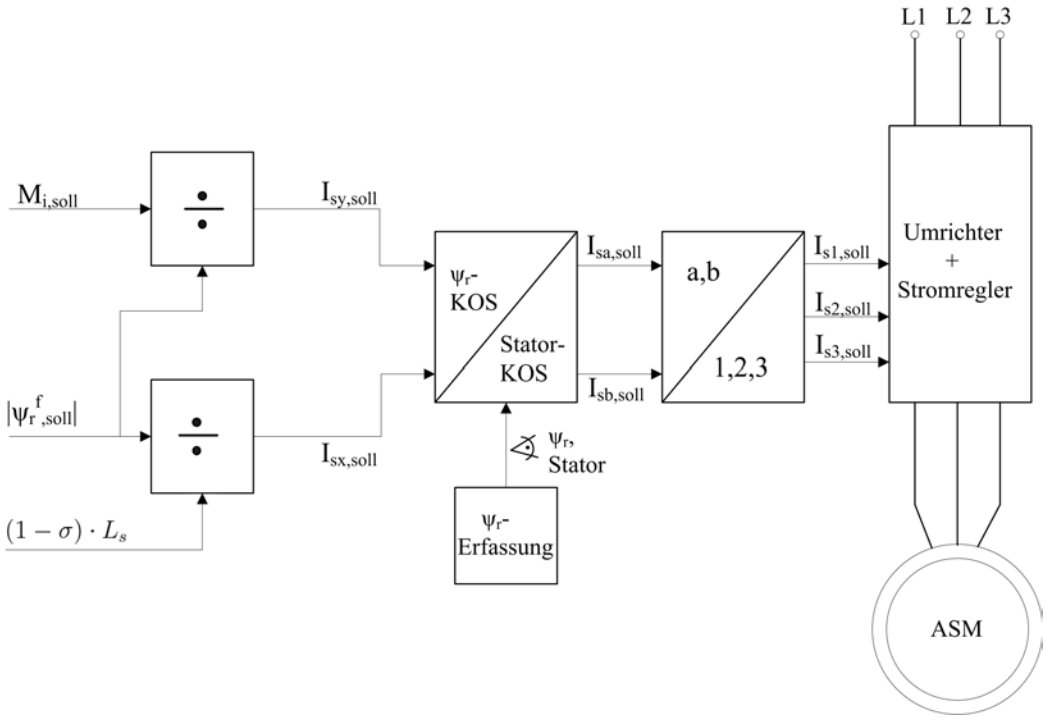

Abb. 2.7: Struktur der feldorientierten Regelung

Wenn die Gleichung (2.39) für die Rotorspannung in das rotorflussfeste Koordinatensystem transformiert wird, ergibt sich die Gleichung:

$$0 = \underline{I}_r^f \cdot R_r + \frac{d\underline{\Psi}_r^f}{dt} + j(\omega_f - \omega_m) \cdot \underline{\Psi}_r^f \tag{2.51}$$

wobei:

$$\omega_f = d\gamma_f/dt \tag{2.52}$$

Nach Elimination des Rotorstroms mit Hilfe der Gleichung (2.46), und Festlegung der Winkelgeschwindigkeit, die in diesem Fall die elektrische Rotorwinkelgeschwindigkeit des Rotorflussverkettungsraumzeigers im rotorflussfesten Koordinatensystem

$$\omega_r = \omega_f - \omega_m = s \cdot \omega_s \tag{2.53}$$

ist, stellt sich die endgültige Formulierung des Gleichungssystems für die feldorientierte Regelung heraus:

$$0 = \frac{1}{T_R} \cdot (\underline{\Psi}_r^f - (1-\sigma) \cdot L_s \cdot \underline{I}_s^f) + \frac{d\underline{\Psi}_r^f}{dt} + j\omega_r \cdot \underline{\Psi}_r^f \tag{2.54}$$

Die Rotorzeitkonstante $T_R = L_r/R_r$ ist ein Maß dafür, wie schnell der Rotorfluss in den Rotor eindringt. Der Fluss kann im ersten Moment nicht in den Rotor eindringen, weil wegen der Lenz'schen Regel im Rotor einen Gegenstrom generiert wird, der das eindringen des Rotorflusses verzögert. Der Rotorwiderstand R_r ist aber nicht beliebig klein, deswegen bildet sich der Fluss mit einer Verzögerung im Rotor. Bei einem supraleitenden Rotor könnte der Fluss, wegen $R_r = 0$, nie eindringen.

Der Realteil (2.57) und der Imaginärteil (2.58) der Gleichung (2.54) werden getrennt angegeben, um das Verhalten des Rotorflusses zu beschreiben und die nötige Information über die Beziehungen zwischen Statorstrom und Rotorfluss bzw. Statorstrom und Drehmoment bestimmen zu können. Wie im Unterabschnitt 2.1.2 angegeben, liegt die reelle Achse des rotorflussfesten Koordinatensystems in Richtung des Rotorflusses, damit ist

$$\underline{\Psi}_r = \Psi_{rx} \tag{2.55}$$

und

$$\Psi_{ry} = 0\ . \tag{2.56}$$

Die Gleichungen werden somit:

$$0 = \frac{1}{T_R} \cdot (\Psi_{rx} - (1-\sigma) \cdot L_s \cdot I_{sx}) + \frac{d\Psi_{rx}}{dt} \tag{2.57}$$

$$0 = -\frac{1}{T_R} \cdot (1-\sigma) \cdot L_s \cdot I_{sy} + \omega_r \cdot \Psi_{rx} \tag{2.58}$$

Der Flussverkettungsraumzeiger ist betragsmäßig nur von der parallelen Stromkomponente I_{sx} abhängig. Die Gleichung (2.57) ist eine lineare Differentialgleichung erster Ordnung mit der rotorflussparallelen Stromkomponente I_{sx} als Steuergröße, welche die Rotorflussverkettung aufbaut. Die Gleichung (2.58) gibt Auskunft über das innere Drehmoment der Maschine und die Rotorwinkelgeschwindigkeit ω_r des Rotorflussraumzeigers bezüglich des Rotors , der sich wiederum mit der mechanischen Winkelgeschwindigkeit ω_m dreht. Das innere Drehmoment wird mit der rotorfluss-normalen Stromkomponente eingestellt.

$$\omega_r = \frac{(1-\sigma) \cdot L_s \cdot I_{sy}}{T_R \cdot \Psi_{rx}} \tag{2.59}$$

$$M_i = -\mathrm{Im}(\underline{I}_s^f \cdot \underline{\Psi}_r^f) =\mid \underline{\Psi}_r^f \mid \cdot I_{sy} = \Psi_{rx} \cdot I_{sy} \tag{2.60}$$

Unter der Annahme, dass die Maschine in einem konstanten Arbeitspunkt betrieben wird, also die Beträge der Rotor und Statorflussraumzeiger konstant sind, kann eine Vereinfachung der Gleichungen (2.57) und (2.58) durchgeführt werden. Die grundlegenden Gleichungen der feldorientierten Regelung (2.61) und (2.62) für den stationären Betrieb werden sich dadurch herausstellen. Die Information

über die Lage des Rotorflussverkettungsraumzeigers $\underline{\Psi}_r^f$ wird dabei als bekannt vorausgesetzt:

$$I_{sx,\text{soll}} = \frac{\mid \underline{\Psi}_{r,soll}^f \mid}{(1-\sigma) \cdot L_s} = \frac{\Psi_{rx,soll}}{(1-\sigma) \cdot L_s} \tag{2.61}$$

$$I_{sy,\text{soll}} = \frac{M_{i,soll}}{\mid \underline{\Psi}_r^f \mid} = \frac{M_{i,soll}}{\Psi_{rx,soll}} \tag{2.62}$$

In der Abbildung 2.7 wird der Fluss für die Berechnung der Stromkomponente $I_{sx,soll}$ gesteuert vorgegeben.

2.2.1 Erfassung des magnetischen Flusses

Die feldorientierte Regelung braucht die Information über die Lage und den Betrag des Flussverkettungsraumzeigers. Die Methoden zur direkten Messung des Flusses, wie mittels Messspulen oder Hallsonden ermöglichen die Erfassung des Flusses, sind aber mit dem Nachteil verbunden, dass ein konstruktiver Eingriff in die Maschine ausgeführt werden muss. Außerdem ist die Fehlerfreiheit einer solchen Einrichtung auch nicht unbestreitbar. Diese Methoden werden daher nicht betrachtet. Die indirekte Methode benützt ein mathematisches Modell der Maschine, um den Fluss ausgehend vom leicht messbaren Größen wie Strom, Spannung und Drehzahl zu bestimmen. Es können dafür die Messgrößen in unterschiedlichen Kombinationen verwendet, und die Strukturen in verschiedenen Koordinatensystemen dargestellt werden, was eine Vielfalt an Modellen für die Messung mit sich bringt. Es werden im Folgenden zwei Methoden vorgestellt, die eine Flussmessung ausgehend von der Rotor- bzw. Statorgleichung ermöglichen.

2.2.2 Spannungsmodell

Die Gleichungen (2.41) und (2.47) geben die Rotorflussverkettung $\underline{\Psi}_r$ berechnet aus Statorstrom- und Statorspannungsraumzeiger an. Es ergibt sich im statorfesten Koordinatensystem nach dem Eliminieren der Statorflussverkettung und des Rotorstromes die folgende Gleichung:

$$\frac{d\underline{\Psi}_r^{s^v}}{dt} = \underline{V}_s^s - R_s \cdot \underline{I}_s^s - \sigma \cdot L_s \cdot \frac{d\underline{I}_s^s}{dt} \tag{2.63}$$

Die Gleichungen können wie folgt in Real und Imaginärteil aufgeteilt werden:

$$\frac{d\Psi_{ra}^v}{dt} = V_{sa} - R_s \cdot I_{sa} - \sigma \cdot L_s \cdot \frac{dI_{sa}}{dt} \tag{2.64}$$

$$\frac{d\Psi_{rb}^v}{dt} = V_{sb} - R_s \cdot I_{sb} - \sigma \cdot L_s \cdot \frac{dI_{sb}}{dt} \tag{2.65}$$

Der Vorteil bei dieser Methode ist, dass die Flussverkettung des Rotors allein aus elektrischen Größen berechnet wird und keine Drehzahlmessung notwendig ist. Nachteilig wirkt der Term mit dem Spannungabfall an Statorwiderstand. Bei kleinen Drehzahlen wirkt ein Fehler in der Berechnung des Spannungsabfallthermes sehr störend auf das Modell, weil der Term so dominiert, dass der Einfluss

restlichen Komponenten nicht mehr richtig ermittelt werden kann. Außerdem versagt das Modell bei geringeren Drehzahlen wegen der offenen Integration in der Gleichung.

2.2.3 Strommodell

Die Basisgleichung für das Strommodell ist (2.54). Die Gleichung hat eine zusätzliche mechanische Eingangsgröße. Nachteilig ist diese Eingangsgröße, wenn Fehler bei der Bestimmung der mechanischen Drehzahl oder des Drehwinkels auftreten. Das Modell funktioniert im Gegensatz zum Spannungsmodell bei allen Drehzahlen gleich gut, weil die Integration in der Gleichung stabil rückgekoppelt ist. Im rotorflussfesten Koordinatensystem ergibt sich eine Struktur des Modells, bei der der Real- und Imaginärteil des Flusses nicht miteinander gekoppelt sind. Diese werden im Rotorflusskoordinatensystem über ein Verzögerungsglied 1. Ordnung mit der temperaturabhängigen Rotorzeitkonstante T_R und den entsprechenden Stromkomponenten aufgebaut. Die Qualität des Modells hängt sehr stark von der Rotorzeitkonstante T_R ab. Die Parameterempfindlichkeit macht sich diesbezüglich bei größeren Belastungen bemerkbar, weil der Parameter, um die Modellgenauigkeit sicher zu stellen, nachgeführt werden muss. Die Methoden der Parameternachführung werden im Abschnitt 2.3 ausführlich beschrieben.

Die Gleichung für das Strommodell im rotorfesten Koordinatensystem ergebt sich aus der Rotorspannungsgleichung (2.38) mit (2.46):

$$\frac{d\underline{\Psi}_r^{r^i}}{dt} = -\frac{1}{T_R} \cdot (\underline{\Psi}_r^r - (1-\sigma) \cdot L_s \cdot \underline{I}_s^r) \tag{2.66}$$

Die Aufteilung vom Real und Imaginärteil der Gleichung ist wie folgt gegeben:

$$\frac{d\Psi_{rd}^i}{dt} = -\frac{1}{T_R} \cdot (\Psi_{rd} - (1-\sigma) \cdot L_s \cdot I_{sd}) \tag{2.67}$$

$$\frac{d\Psi_{rq}^i}{dt} = -\frac{1}{T_R} \cdot (\Psi_{rq} - (1-\sigma) \cdot L_s \cdot I_{sq}) \tag{2.68}$$

2.3 Parameter und Parameternachführung

Die Parameter im Ersatzschaltbild der Maschine (Abb. 2.5) werden von bestimmten Faktoren und Größen in der Maschine beeinflusst. Die Temperaturschwankungen wirken sich hauptsächlich auf den Statorwiderstand R_s und den Rotorwiderstand R_r aus. Während des Betriebes können sich der Ständer- und der Läuferwiderstand erwärmungsbedingt um bis zu 50% ändern. Es ergibt sich abhängig von der Größe des eingestellten Maschinenflusses eine unterschiedliche Sättigung, die dann einen Einfluss auf die Hauptfeldinduktivität L_m und die Statorinduktivität L_s hat. Im Gegensatz zur Synchronmaschine muss die magnetische Durchflutung in der Asynchronmaschine erst durch den Strom in der Ständerwicklung aufgebaut werden. Änderungen der Hauptfeldinduktivität L_m sind zu berücksichtigen, wenn die Maschine auch im Feldschwächbereich betrieben wird. Die Streuinduktivität der Maschine L_σ ist von der Sättigung abhängig. Die sättigungsbedingten Änderungen können rasch ablaufen, die temperaturbedingten Änderungen laufen dagegen sehr langsam ab. Bei der feld-

orientierten Regelung bleibt die Läuferfrequenz klein, in den Läuferstäben tritt nur vernachlässigbare Stromverdrängung auf. Deshalb kann im weiteren eine stromverdrängungsfreie Asynchronmaschine vorausgesetzt werden.

Wie bereits erwähnt sind die Parameter, die für die feldorientierte Regelung einer Asynchronmaschine benötigt werden, nicht immer einer direkten Messung zugänglich. Sie werden üblicherweise mit Hilfe eines parallelen Maschinenmodells oder eines Beobachters nachgeführt. Der Zustand der Maschine wird aber nur dann richtig festgelegt, wenn die Modellparameter mit den Maschinenparametern übereinstimmen und damit für die Struktur der Regelung notwendige Komponenten vom Statorstrom, die moment- bzw. flussbildend wirken richtig entkoppelt werden können [24], [32]. Da sich die Parameter während des laufenden Betriebes ändern können und verstimmte Parameter das dynamische Verhalten der feldorientierten Regelung verschlechtern, ist die genaue Kenntnis der Maschinenparameter sehr wichtig. Es existiert eine Vielzahl von Methoden für die Identifikation bzw. Nachführung der Parameter. Sie werden im Folgenden kurz erläutert und abschließend wird eine detaillierte Beschreibung der in der Arbeit eingesetzter Methode durchgeführt.

2.4 Methoden der Parameternachführung

Die Parameter werden entweder offline identifiziert oder online nachgeführt. Weil die feldorientiert geregelte Asynchronmaschine immer mit einem Umrichter gespeist wird, sind zahlreiche Verfahren basierend auf Umrichterspeisung entwickelt worden. Einige dieser Vefahren werden als Methoden mit "Offline-Parameteridentifikation" bezeichnet, da erfolgt die Bestimmung der Parameter im Stillstand der Maschine, wobei die Rechenzeit eine untergeordnete Rolle spielt. Zusätzlich gibt es heute zahlreiche Verfahren zur Nachführung der Parameter während des Betriebes. Diese Verfahren werden als Methoden mit "Online-Parameternachführung" bezeichnet. Die Bestimmung der durch Erwärmung oder Sättigung veränderlichen Parameter erfolgt während des normalen Betriebes. Für die Identifikation steht nur eine begrenzte Rechenzeit zu Verfügung. Außerdem soll die Identifikation der Parameter das Regelverhalten nicht stören bzw. beeinflussen. Die eingesetzte Methode entscheidet darüber, welche Größen mit welcher Genauigkeit für die Berechnung der Rotorflussverkettung notwendig sind [34], [32]. Besonders wichtig für die Parameternachführung sind die Anregungssignale. Einige Verfahren funktionieren mit künstlich erzeugten Testsignalen. Es gibt Identifikationsverfahren, die mit binären Zufallssignalen, oder rechteckförmigen Signalen arbeiten. Es ist auch zu erwähnen, dass bei einer Identifikation im Stillstand mit relativ großen Testsignalen gearbeitet werden kann, dagegen aber im laufenden Betrieb kleine Testsignale eingesetzt werden sollen, um die Auswirkung der zusätzlichen Anregung auf die Momentenentwicklung zu minimieren [16].

2.4.1 Methoden zur Offline Parameter Identifikation

Häufig kommt es vor, dass der Antrieb mit der anzutreibenden Anlage schon verbunden ist. Es ist dann erforderlich die Identifikation der Parameter im Stillstand zu machen damit sich kein Drehmoment in der Maschine entwickelt. Die drehmomentbildende Stromkomponente I_{sy} ist dann Null, die flussbildende Stromkomponente I_{sx} kann eingespeist werden. Dazu wird die Asynchronmaschi-

ne im Stillstand einachsig entweder mit sinusförmigen Testsignalen oder mit einer Gleichspannung gespeist. Messungen werden entweder beim drehbarem, unbelasteten Rotor oder bei blockiertem (abgebremstem) Rotor ausgeführt. Wenn bei unbelastetem Rotor ein Testsignal in die Maschine geschickt wird, fließt es abhängig von Frequenz als Blindstrom durch die Hauptfeldinduktivität L_m, und eine Messung von L_m wird dadurch ermöglicht. Der Rotorwiderstand R_r kann aber mit so einer Messung nicht gemessen bzw. nachgeführt werden. Wenn die Messung bei blockiertem Rotor ausgeführt wird, dann fließt der Teststrom auch durch den Rotor und ermöglicht die Messung des Rotorwiderstandes R_r [33]. Weiters werden verschiedene Methoden vorgestellt, die im Stillstand der Maschine die Parameteridentifikation durchführen.

Als erstes wird die Identifikation mit der Gleichspannungseinspeisung betrachtet. Bei dieser wird aus der angelegten Spannung und dem resultierenden stationären Strom der Statorwiderstand R_s berechnet. Die restlichen Parameter werden aus der Sprungantwort des Stromes, als Reaktion der angelegten Spannung gewonnen [17], [21], [11]. Es wird mit unterschiedlich gepulsten Gleichspannungen der gesamte Frequenzbereich der Asynchronmaschine angeregt.

Neben dieser Methode der Parameteridentifikation im Stillstand der Maschine wird eine Adaption der Parameter während des Betriebes vorgestellt. Die Parameter R_s, L_s, und σ werden aus den Zeitkonstanten der Übertragungsfunktion über Koeffizientenvergleich berechnet. Das Verfahren der größten Wahrscheinlichkeit "Maximum-Likelihood-Schätzung" erlaubt dagegen mit Hilfe eines Gleichspannungssprungs die Parameter der Übertragungsfunktion zu ermitteln. An eine Achse der Maschine wird ein Gleischspannungssprung gelegt. Mit der Kenntnis des Statorwiderstandes R_s werden dann schrittweise die Parameter des Ersatzschaltbildes in der Abb. 2.5 mit der Maximum-Likelihood-Methode identifiziert. Es wird dazu die gemessene Sprungantwort des Stromes mit der durch das Modell beschriebenen verglichen, die im Stillstand ermittelte Frequenzkennlinie mit der durch das Modell beschriebenen verglichen und der gemessene Beschleunigungsvorgang mit dem berechneten verglichen [23].

Eine andere Art von Anregung ist ein Spannungsprung in negativer Richtung. Die Identifikation erfolgt mit der rekursiven Methode der kleinsten Fehlerquadrate "Least Squares Algorithm" [25]. Dabei wird im Stillstand in die $x-$Achse in Abb. 2.3 eine Gleichspannung eingespeist, sodass sich ein flussbildender Strom einstellt. Als erster Schritt wird R_s ermittelt, in dem die Spannung solange erhöht wird, bis sich der Nennhauptfeldstrom I_m einstellt. Um die Statorzeitkonstante

$$T_S = (L_m + L_\sigma)/R_s \tag{2.69}$$

zu bestimmen, wird die Sprungantwort des Stromes auf eine hochfrequente Gleichspannung, eine zufällige binäre Sequenzspannung, aufgenommen. Sie wird hauptsächlich von der schnelleren Zeitkonstante (Statorzeitkonstante T_S) bestimmt. Der Einfluss der Rotorzeitkonstante T_R kann dabei vernachlässigt werden. Die Parameter lassen sich aus der Übertragungsfunktion berechnen [5], [31].

Eine Erweiterung der Methode erfolgt mit einer direkten zeitkontinuierlichen Identifikation. Bei der Methode muss R_s nicht vorher bestimmt werden. Die Grundlage ist ein adaptiver Beobachter. Mit Hilfe des Least Squares Algorithm werden offline aus dem Beobachtungsfehler die gesuchten

Parameter, R_s, T_R, σ bestimmt. Es können verschiedene Anregungssignale wie Spannungssprünge verschiedener Amplituden oder Binäre Zufallsrauschsignalen, angelegt werden [6]. Um eine Identifikation ohne Strangspannungserfassung zu ermöglichen, werden verschiedene Testsignale angelegt, was zuerst die Berechnung der Spannung, dann R_s, L_σ, R_r und die Identifikation der Magnetisierungskennlinie ermöglicht. Die berechneten und gemessenen Stromantworten werden verglichen um zu sehen, wie gut die Parameter identifiziert wurden.

Bei der sinusförmigen Speisung müssen verschiedene Frequenzen durchfahren werden um das System anzuregen. Die Auswertung erfolgt dann über eine Frequenzgangsanalyse. Die Asynchronmaschine wird dazu im Stillstand mit sinusförmigen Testsignalen verschiedener Frequenz und konstanter Amplitude gespeist. Im [3] wird angegeben, dass bei diesem Verfahren aus dem Imaginärteil der komplexen Impedanz der Maschine, der Rotorwiderstand R_r und die Streuinduktivität L_σ bestimmt werden, aber der Statorwiderstand R_s, der aus dem Realteil der komplexen Impedanz bestimmt wird, nur fehlerbehaftet ermittelt werden kann.

Ein anderes Verfahren direkter Selbstregelung mit sinusförmigen Testsignalen ermittelt R_s in einem Gleichspannungstest und in einem Zweifrequenzverfahren iterativ, und die restlichen Parameter L_m, L_σ, R_r unter Berücksichtigung des Betriebspunktes sehr genau [18]. Die Magnetisierungskurve der Asynchronmaschine kann durch einphasige sinusförmige Testsignale, die einachsig eingespeist werden, mit nur einem Testverfahren bestimmt werden. Der Rotorwiderstand R_r wird auch gleichzeitig mit dem selben Verfahren ermittelt [2]. Normalerweise benötigt die Erfassung der Magnetisierungskurve mehrere Versuche für die richtige Identifikation der Kurve.

Eine weitere spezielle Identifikationsfunktion ermöglicht eine genaue Erfassung der Magnetisierungskurve die robust gegenüber Änderungen des Statorwiderstands R_s ist. Die Methode braucht kein Testsignal. Die Messung wird an einer unbelasteten Maschine bei der Drehzahl von ca. 100 Umdrehungen pro Minute durchgeführt. Die niedrige Drehzahl ermöglicht eine Messung ohne Berücksichtigung der Eisen- oder Reibungsverluste, was eine relativ gute Flusserfassung mit sich bringt [22].

Es muss betont werden, dass die Genauigkeit aller offline durchgeführten Identifikationsmethoden von der Abtastgeschwindigkeit, der Auflösung und der Genauigkeit der Sensoren abhängig ist. Die Probleme bei einer offline Identifikation sind die Nichtlinearität von Umrichter, die eine exakte Parameteridentifikation auf Basis vom rekonstruierten Spannungsabfall ohne vorherige Kenntnis des Spannunsverhaltens des Umrichters erschwert. Und die Aussperrezeit des Inverters, in der keine Messungen durchgeführt werden können.

2.4.2 Methoden zur Online Parameternachführung

Es existieren eine Vielzahl von Verfahren die die Läuferzeitkonstante und besonders relevante Parameter während des Betriebes identifizieren und ihren Wert nachführen. Sie werden weiters in drei Gruppen aufgeteilt.

1. Methoden mit Spektralanalyse

Diese Methoden umfassen alle Messungen basierend auf den eingespeisten Testsignalen oder die existierende charakteristische Oberschwingungen im Strom- oder Spannungsspektrum. Bei der Durchführung der Methode werden Statorströme/-spannungen abgetastet. Die Parameter werden aus der danach durchgeführten Spektralanalyse gewonnen. Es wird ein Störsignal gebraucht, weil im Leerlauf keinen Strom im Rotor induziert wird und der Schlupfferquenz ebenfalls Null ist. Dem zufolge wäre es ohne ein Testsignal nicht möglich, die Rotorparameter zu bestimmen.

Eine Methode verwendet die statistische Korrelation. Ausgehend vom Strommodell im statorfesten Koordinatensystem wird die Rotorzeitkonstante T_R adaptiert. Falls T_R fehlerhaft eingestellt worden ist, sind die Stromkomponenten, die fluss- bzw. drehmomentbildend wirken, nicht mehr entkoppelt. Wenn die flussbildende Statorstromkomponente I_{sx} mit einem Testsignal beaufschlagt wird, wirkt es sich auf den Differenz des gemessenen und berechneten Moments aus und der Fehler ist deutlich zu erkennen. Es muss dazu die Amplitude des Störsignals klein gehalten werden, um die Momentenschwankung bei einer fehlerhaft eingestellten T_R gering zu halten. Dazu kann im Mikrorechner ein hochfrequentes statistisches Zufallssignal als Rauschen erzeugt werden.

Eine weitere Methode basiert auf der Online-Adaption der Rotorzeitkonstante, dabei wird die Kurvenschar des gemessenen Statorstromes mit der Kurvenschar des gerechneten über ein Gradientenverfahren abgeglichen, bis der minimale Fehler gefunden wird. Erst dann stimmt die vorhandene Rotorzeitkonstante mit der im Modell eingestellten überein. Für die genaue Bestimmung muss aber der Vergleichsvorgang über einen längeren Zeitraum stattfinden. Das ermöglicht aber nur die Adaption der langsam veränderlichen Parameter wie den temperaturabhängigen Rotorwiderstand R_r udgl. [14].

Es kann auch als Testsignal ein binäres Zufallssignal eingespeist werden. Wie oben schon erläutert, wird die flussbildende Achse des Stromes I_{sx} mit dem binären Signal beaufschlagt und mit der drehmomentbildender Komponente I_{sy} korreliert. Das Vorzeichen der Korrelation gibt an, in welche Richtung die Nachführung der Rotorzeitkonstante T_R gehen soll. Die Methode arbeitet aber nicht zufriedenstellend mit schwacher Last. Als Alternative kann ein sinusförmiges Signal als Störsignal in die flussbildende Stromachse gespeist werden. Diese Methode arbeitet zwar gut mit allen Belastungsarten und Geschwindigkeiten, ist aber wegen der erforderlichen Prüfspulen nicht kostengünstig [30].

2. Beobachterbasierte Methoden

Die Methode funktioniert während des normalen Betriebs der Asynchronmaschine und benötigt daher kein Testsignal. Es wird für die Parameternachführung ein erweitertes Kalman-Filter (EKF) eingesetzt, bei dem die Werte der Rückführmatrix so berechnet werden, dass bei verrauschten Signalen eine optimale Zustandsschätzung erreicht wird. Ziel ist es den Zustandsvektor so zu rekonstruieren, dass der quadratische Mittelwert, die Kovarianz des Rekonstruktionsfehlers minimal wird. EKF wird zur Schätzung des Rotorflusses Ψ_r und der elektrischen Rotorwinkelgeschwindigkeit ω_r eingesetzt. Es werden dazu die Strangströme der Maschine als Zustandsgrößen und Störungen, die als weißes

Rauschen angenommen werden, in das Filter angegeben und daraus Ψ_r und ω_r geschätzt.

Als Beispiel kann folgende Methode angegeben werden: für die Nachführung der Rotorzeitkonstante T_R, wird $\frac{1}{T_R}$ als fünfte Zustandsgröße mit den restlichen Stator- und Rotorströmen in das Filter gegeben. Das ist ähnlich wie in das System ein Störsignal zu schicken. In diesem Fall wird es aber nicht extern erzeugt. Die Oberschwingungen der Ausgangsspannung des PWM Inverters sorgen für die Anregung des Systems. Der Grundgedanke dahinter ist, wenn die Winkelgeschwindigkeit der Maschine sich zeitlich ändert, dann wird die Maschine sich wie ein zeitabhängiges "2 Input/ 2 Output"-System mit überlagertem Störgeräusch-Input verhalten [36]. Bei der Verwendung des Kalman-Filters ist es wichtig, die Hauptfeldinduktivität $L_m = L_r$ richtig zu schätzen weil die Rotorzeitkonstante von dieser Größe abhängig ist:

$$T_R = \frac{L_r}{R_r} \tag{2.70}$$

In [15] wurden ausgehend von "2 Input/ 2 Output"-Systemverhalten der Maschine Messungen durchgeführt, in denen von der Abhängigkeit der Hauptfeldinduktivität vom Fluss Gebrauch gemacht wird. Die Rotorinduktivität L_r kann dann mit Hilfe der Gleichung (2.70) nachgeführt werden. Das größte Problem bei den Methoden mit Kalman-Filter ist der große Rechenaufwand. Außerdem werden alle Induktivitäten in Gleichungen als Konstanten behandelt, was dem Grundgedanken der Parameternachführung widerspricht.

3. Adaption mit Modellrechnungen

Die Methoden dieser Gruppe erregen, dank ihrer einfachen Stuktur und Realisierung, die größte Aufmerksamkeit. Die Methoden bauen nicht auf sehr umfangreichen und komplexen Modellen auf. Ein Beispiel dazu ist das Aufstellen einer Ausgangsfehlerfunktion, die mit Hilfe eines Identifikationsverfahrens vereinfacht wird. Dafür wird die Differenz zwischen einer im Modell berechneten Maschinengröße und deren gemessener Wert ermittelt und dann die Korrektur des Modells durchgeführt. Ausgehend vom Strommodell im statorfesten Koordinatensystem kann mit dem gemessenen Statorspannungsraumzeiger $\underline{V}_s$ als Vergleichsgröße der Rotorwiderstand R_r bestimmt werden. Der Spannungsraumzeiger wird in Real- und Imaginärteil zerlegt. Der Einfluss des Statorwiderstands im Imaginärteil kann durch Transformation der Spannung in ein Koordinatensystem, das mit dem Statorstromraumzeiger umläuft, eliminiert werden. Es ergeben sich aber zusätzliche Fehler zwischen dem gemessenen und dem berechneten Wert wegen der Transformation. Das führt zu einer weiteren Methode, bei der aus den Amplituden der Statorspannung von Modell bzw. Maschine der Ausgangsfehler gebildet wird. In dieser Methode ist aber die Fehlerfunktion mit Statorkupferverlusten behaftet und bei größeren Drehzahlen verschlechtert die Phasendifferenz zwischen der gemessenen und berechneten Spannung die Genauigkeit der Parameternachführung [16].

Eine andere Methode stellt eine Fehlerfunktion aus dem stationären Gleichungssystem als Ergebnis einer Blindleistungsbilanz für das Strommodell im rotorfesten Koordinatensystem auf. Die Stromkomponenten werden aus den Strangströmen ermittelt und die Spannungskomponenten werden aus dem Regelalgorithmus entnommen. Nach der Bestimmung der Lage der Welle, können mit

Hilfe von Least Squares Algorithmen die Parameter T_R, L_s und σ in einem Berechnungsverfahren identifiziert werden. Es existieren verschiedene Varianten für das Aufstellen der Ausgangsfehlerfunktion bezüglich des im rotorflussfesten Koordinatensystem erfassten Strommodells.

Es können weitere Methoden erwähnt werden; die Verwendung von künstlicher Intelligenz "Artificial Intelligence", wo neuronale Netze für die Nachführung der Rotorzeitkonstante T_R bzw. Rotorwiderstand R_r eingesetzt werden [35]. Eine andere Variante der Methoden mit künstlichen Intelligenz ist die Fuzzy-Logik für die Nachführung der Rotorzeitkonstante T_R [8].

2.4.3 Untersuchte Methode

Die in der Diplomarbeit untersuchte Methode basiert auf Adaption mit Modellrechnungen. Mit dem Ziel, die Rotorzeitkonstante T_R und die Rotorinduktivität L_r nachzuführen. Wie oben erwähnt hat die Methode eine einfach realisierbare Struktur. Der gesamte Vorgang funktioniert auch ohne Regelung der Maschine und ist somit von den Größen der Regelung völlig unabhängig. Der Grundgedanke dahinter ist, die Rotorinduktivität L_r , die Statorinduktivität L_s und die Rotorzeitkonstante T_R aus dem Vergleich zweier Modelle zu ermitteln [4], [7]. Die Modelle, die zum Vergleichsprozess verwendet werden, sind das Spannungsmodell aus Unterabschnitt 2.2.2 und das Strommodell aus Unterabschnitt 2.2.3. In beiden Modellen wird ausgehend von (2.64), (2.65) für das Spannungsmodell und von (2.67), (2.68) für das Strommodell jeweils ein magnetischer Flussverkettungsraumzeiger ermittelt. Das Spannungsmodell arbeitet mit Statorstromraumzeiger und Statorspannungsraumzeiger und ermittelt aus diesen Größen einen Statorflussraumzeiger mit der Annahme, dass die Streuinduktivität L_σ bekannt sind. Deswegen ist das Spannungsmodell das Referenzmodell der Methode. Das Strommodell braucht für die Ermittlung des Flussverkettungsraumzeigers die nachgeführten Parameter T_R und L_r und die Lage des Rotors γ_m als Eingangsgrößen. Die berechneten Flussverkettungsraumzeiger werden abschließend durch den Statorstromraumzeiger dividiert, woraus eine komplexe Größe erhält wird, die dann als Hilfsgröße für die Parameternachführung herangezogen wird. Diese Größe wird weiters als "komplexe Ersatzgröße" benannt.

Die Statorinduktivität wird ausgehend der komplexen Ersatzgröße $\underline{L}_s$ festgelegt,

$$\underline{L}_s = \frac{\underline{\Psi}_s^s}{\underline{I}_s^s} \tag{2.71}$$

wobei die Statorinduktivität L_s in der Gleichung (2.28) angegeben ist. Wenn der stationäre Zustand der Maschine betrachtet wird, wird $\underline{L}_s$ ausgehend folgenden Spannungs- und Flussverekttungsgleichungen der Asynchronmaschine (2.37), (2.41), (2.46), (2.47) bestimmt:

$$\underline{L}_s = \frac{L_m}{1 + j(\omega_s - \omega_m)T_R} + L_{s\sigma} \tag{2.72}$$

Für die Nachführung der Rotorinduktivität L_r wird die komplexe Ersatzgröße ausgewertet, die wieder aus dem komplexen Rotorflussraumzeiger und dem Statorstromraumzeiger berechnet wird

$$\underline{L}_r = \frac{\underline{\Psi}_r}{\underline{I}_s} \tag{2.73}$$

Für jedes Modell wird die oben angegebene komplexe Ersatzgröße berechnet und abschliessend werden die beiden abgeglichen. Die Nachführung erfolgt über den Modellvergleich mit dem Spannungsmodell als Referenzmodell.

$$\underline{L}_r^v = \frac{\underline{\Psi}_r^{s^v}}{\underline{I}_s^s} \tag{2.74}$$

$$\underline{L}_r^i = \frac{\underline{\Psi}_r^{r^i}}{\underline{I}_s^r} \tag{2.75}$$

Die Methode basiert auf der Anpassung der beiden komplexen Ersatzgrößen aneinander. Die komplexe Ersatzgröße des Strommodells $\underline{L}_r^i$ wird aus der komplexen Ersatzgröße des Spannungsmodells $\underline{L}_r^v$, das als Referenzmodell gilt, adaptiert. Wenn die beiden Ersatzgrößen gleich sind, sind daraus folgend die Flussverkettungen der Modelle gleich groß und es stimmen auch die Drehmomenten des Strom- und Spannungsmodells überein. Die komplexe Ersatzgröße des Rotors $\underline{L}_r$ wird wie die komplexe Ersatzgröße des Stators $\underline{L}_s$ aus den Spannungs- und Flussverkettungsgleichungen gewonnen:

$$\underline{L}_r = \frac{L_m}{1 + j(\omega_f - \omega_m)T_R} \tag{2.76}$$

Für die Anpassung betrachtet man einen Kreis, auf dem sich die komplexe Ersatzgröße des Rotors, in Abhängigkeit des Betriebspunktes der Maschine bewegt. Wenn die komplexe Ebene betrachtet wird, liegt der Mittelpunkt dieses Kreises auf der positiven reellen Achse, ein Punkt des Kreises geht durch den Ursprung. Der Durchmesser dieses Kreises ist die Rotorinduktivität , weil $\underset{(\omega_f-\omega_m)\to\infty}{\ell im}\ \underline{L}_r = L_r$ ist. Die aktuelle Lage von $\underline{L}_r$ auf der Ortskurve (Abb. 2.8) ist durch den Term $(\omega_f - \omega_m) \cdot T_R$ bestimmt.

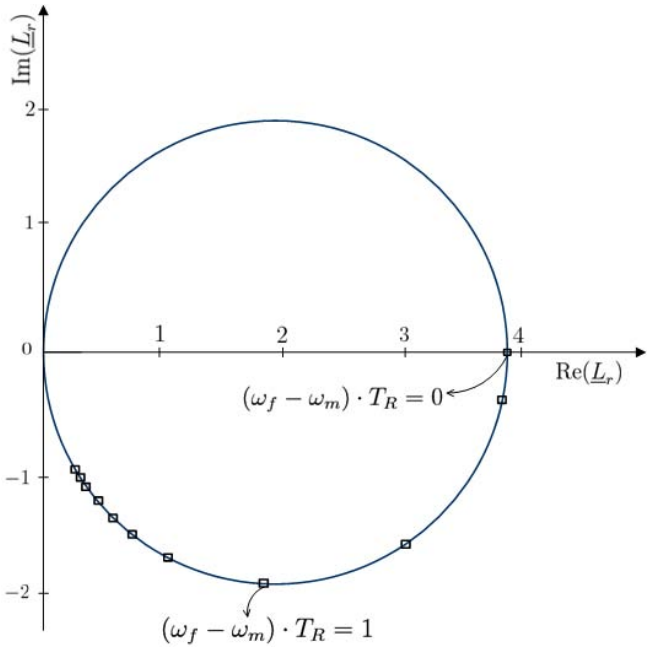

Abb. 2.8: Ortskurve der komplexen Ersatzgröße des Rotors

Für die Parameternachführung müssen vom Kreis der zugehörige Durchmesser und die Lage von $\underline{L}_r$ des Strommodells bestimmt und mit der Lage und dem Durchmesser von $\underline{L}_r$ des Spannungsmodells abgeglichen werden[4].

Die Rotorinduktivität L_r wird aus den Durchmessern beider Modelle wie folgt bestimmt:

$$L_r^v = \frac{|\,\underline{L}_r^v\,|^2}{\mathrm{Re}(\underline{L}_r^v)} \tag{2.77}$$

$$L_r^i = \frac{|\,\underline{L}_r^i\,|^2}{\mathrm{Re}(\underline{L}_r^i)} \tag{2.78}$$

Die Differenz von Rotorinduktivitäten beider Modelle

$$\Delta L = L_r^v - L_r^i \tag{2.79}$$

wird einem Regler zur Nachführung zugeführt. Nach der Anpassung der Durchmesser des Kreises wird die Lage von $\underline{L}_r$ von beiden Modellen bestimmt und über den Beträgen der beiden Ersatzgrößen kann T_R abgeglichen werden. Die Differenz der Beträge

$$\Delta L = |\,\underline{L}_r^i\,| - |\,\underline{L}_r^v\,| \tag{2.80}$$

wird auch zu einem Regler geführt, der dann T_R nachführt. Die Abbildung 2.9 gibt das Strukturschaltbild der untersuchten Methode an.

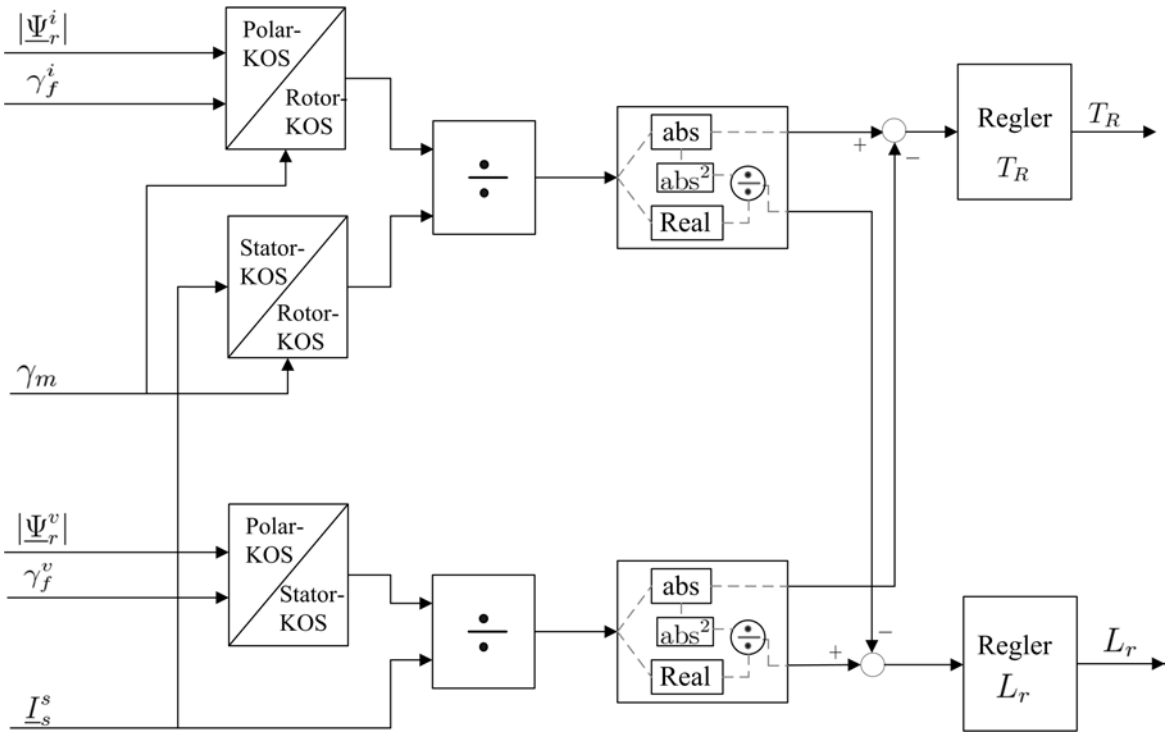

Abb. 2.9: Strukturschaltbild der untersuchten Methode

Die Temperaturbestimmung wird mit Kenntnis der beiden nachgeführten Größen ermöglicht. Der aktuelle Rotorwiderstand R_r wird ausgehend von der Gleichung (2.70) bestimmt. Der temperaturabhängige Rotorwiderstand R_r kann mit Hilfe der bekannten Referenzgrößen $R_{r,\mathrm{ref}}$ und $T_{r,\mathrm{ref}}$ wie folgt angegeben werden:

$$R_r = R_{r,\mathrm{ref}} \cdot (1 + \alpha_{\mathrm{ref}} \cdot (T_r - T_{r,\mathrm{ref}})) \tag{2.81}$$

wobei weiters als Referenztemperatur 20°C angenommen wird und $\alpha_{20} = 0,0040$ [1/K] die Temperaturkoeffizient des Rotors bei dieser Referenztemperatur ist. Die ausgehend von der Gleichung (2.81) bestimmte Rotortemperatur T_r wird zur Bestimmung der Statortemperatur herangezogen. Basierend auf einer vereinfachten Modellvorstellung können die Änderungen der Statortemperatur den Änderungen der Rotortemperatur mit einer bestimmten Proportionalität k gleichgesetzt werden, wobei der Proportionalitätsfaktor k einen manuell bestimmbaren Wert hat und sich zeitlich nicht verändert.

$$\Delta T_s = \mathrm{k} \cdot \Delta T_r \tag{2.82}$$

2.5 Zusammenfassung

Die Asynchronmaschine mit feldorientierter Regelung ist so hochdynamisch wie eine Gleichstrommaschine, und viel robuster durch die verbesserte Qualität der Regelung mit der Parameternachführung. Deswegen ist das wesentliche, was die feldorientierte Regelung attraktiver macht, die Parameternachführung. Mit der genauen Kenntnis der Parameter wird das dynamische Verhalten des gesamten Antriebes verbessert. Es existieren zahlreiche Methoden der Parameternachführung bzw. Identifikation, die entweder während des normalen Betriebes oder im Stillstand der Maschine durchgeführt werden. In der Diplomarbeit wird eine Methode untersucht, die basierend auf Modellrechnungen im normalen Betrieb die Parameternachführung durchführt. Bei der Methode wird der magnetische Rotorflussverkettungsraumzeiger jeweils von einem Spannungsmodell und einem Strommodell berechnet. Der Flussverkettungsraumzeiger wird durch den Statorstromraumzeiger dividiert um die komplexe Ersatzgröße als Hilfsgröße zu bekommen. Durch den Vergleich und die Regelung dieser beiden Größen werden die Rotorinduktivität L_r und Rotorzeitkonstante T_R nachgeführt. Die durch die Nachführung ermittelten Parameter werden wieder zum Strommodell zugeführt. Die Methode ist im Vergleich zu anderen Methoden relativ robust, und benötigt keine aufwendige Algorithmen.

3 Analyse und Design

3.1 Aufgabenstellung

Die der Arbeit zugrunde liegende Aufgabe ist die Implementierung der Parameternachführung in Modelica nach der im Unterabschnitt 2.4.3 erklärten Methode. In Modelica waren das Strommodell, das Spannungsmodell, und das Modell der feldorietierten Regelung schon vorhanden. Das Spannungsmodell und das Strommodell haben aber mit konstanten Parametern gearbeitet. Die beiden Modelle sind im Rahmen der Diplomarbeit so umstrukturiert, dass sie mit nachgeführten Parametern arbeiten, die mit einer Tracking Box geliefert werden. Die Wirkungen wie Temperaturänderungen und die Sättigungseffekte auf die Paramater werden dadurch berücksichtigt. Die Tracking Box beschreibt die Methode der Parameternachführung (Unterabschnitt 2.4.3) in Modelica. Die Parameter, die im aktuellen Zustand der Maschine wirksam sind, werden durch die Tracking-Box erfasst.

3.2 Modelica

Die Modellierung und Simulation von technischen Systemen ist eine der wichtigsten Grundvoraussetzungen bei den heutigen Entwicklungsprozessen, um frühzeitig Aussagen über das Verhalten von Systemen machen zu können. Modelica als objektorientierte Modellierungssprache ermöglicht im Vergleich zu anderen Modellierungssprachen, wie MATLAB, einfachere Lösungen von komplexen Systemen. Das wird durch die modulare Modellierung von komplexen Systemen realisiert. Die Eigenschaften wie z.B. die objekt-orientierte Modellierung und die Möglichkeit der Wiederverwendung und Austausch von Modellen ermöglichen es, intelligente Modelle zu erstellen. Als offene Sprache unterstützt Modelica die Weiterentwicklung von Bibliotheken sowie die Erweiterung von Modellen. Modelica ist eine gleichungsorientierte Sprache für technische und physikalische Systeme. Die Modelle werden mit Gleichungen beschrieben und orientieren sich somit an einer bestimmten Modellvorstellung von realen Systemen. Es bietet zudem die Möglichkeit, kontinuierliche und diskrete Systeme zu beschreiben [10] (Die weiter folgenden Informationen über Modelica beziehen sich auf die Literatur [10]). Die Grundlagen von gleichungsorientierten Modellierungssprachen können wie folgt zusammengefasst werden:

- Jedes Symbol in einem graphischen Layer repräsentiert eine physikalische Komponente: ohmscher Widerstand, Pumpe, elektrische Maschine, mechanisches Getriebe usw.
- Verbindungen von Konnektoren repräsentieren ideale physikalische Verbindungen; elektrische Leitung, mechanische Verbindung, thermische Verbindung
- Die Wechselwirkung der Komponenten wird in Bezug auf Variablen in den Interfaces beschrieben.
- Die Gleichungen beschreiben das physikalische Verhalten der Komponenten.
- Hierarchische Zerlegung der Komponenten unterstützt die Übersichtlichkeit.

Die Beschreibung der Modelle erfolgt in Modelica durch gewöhnliche Differentialgleichungen sowie diskrete und algebraische Gleichungen. Die Programmierung der Gleichungen erfolgt akausal. Dies bedeutet, dass im Gegensatz zu herkömmlichen Programmiersprachen keine Zuweisungen verwendet werden. Die Gleichung wird somit nicht manuell nach der unbekannten aufgelöst, sondern automatisch beim Übersetzen des Modells aufgrund der Modellkausalitäten. Der im Simulationsprogramm enthaltenen Modellübersetzer transformiert die Gleichung symbolisch in eine lösbare Form. Weiterhin können Differential-algebraische-Gleichungssysteme gelöst werden, womit eine dynamische Simulation des Modellverhaltens ermöglicht wird.

Zum Beispiel wird $x = 2+y$ in Programmiersprachen wie Java oder C++ etc. als eine Zuweisung durchgeführt, während genau der selbe Ausdruck in Modelica nach dem unbekannten x oder y gelöst wird, je nach dem welche die unbekannte Variable ist. Der Modellübersetzer manipuliert die Gleichung um die unbekannte Variable zu bestimmen.

Die akausale Programmierung vereinfacht die Entwicklung der Modelle beträchtlich. Eine einzelne Klasse kann Variablen (auch Instanzen anderer Klassen), Gleichungen und lokale Klassendefinitionen enthalten. Mit Hilfe gleichungsbasierter, akausaler Modellierung wird die Wiederverwendung von Komponenten und Modellen erleichtert. Es werden zur Beschreibung von Vorgängen Konnektoren genutzt, wobei die Flussgrößen eine feste Bezugsrichtung aufweisen. An den Konnektoren existieren Potential- und Fluss-Variablen, die in allen Modellen verwendet werden. Somit ist der Aufbau und das Zusammenfügen der physikalischen Komponenten möglich. Damit unterstützt Modelica die Modellbildung mehrerer physikalischer Effekte. So können etwa elektrische, thermische, mechanische, chemische etc. gleichzeitig simuliert werden. Diese multiphysikalische Berechnungsmöglichkeit wird "Multi Domain" Fähigkeit genannt.

Zum Beispiel, wenn Konnektoren verschiedener Teilmodelle miteinander verbundenen werden, wird beim Kompilieren des Gesamtmodells ein Gleichungssystem generiert, welches die Potenziale der miteinander verbundenen Konnektoren einander gleichsetzt und die Summe der Flussgrößen zu Null addiert (Knotensatz). Auf diesem Konzept beruht die Affinität zwischen Modelica und der Physik der dynamischen Systeme.

Eine weitere wichtige Eigenschaft ist die Vererbung, "inheritance". Die Daten und die Eigenschaften wie Deklarationen, Gleichungen und andere gewisse Inhalte werden dabei in die Unterklasse, die so genannte "subclass", kopiert. Die Vererbung ermöglicht die Wiederverwendung der Komponenten und eine hierarchische Modellstruktur. Modelle werden von Basisklassen abgeleitet und erben deren Daten und Verhalten. Das ermöglicht eine besser strukturierte Organisation gleichartiger Komponenten.

Zusammenfassend zeichnet sich Modelica durch folgende Vorteile aus:

1. Die akausale Modellbildung auf der Basis von Differential-algebraischen Gleichungen

2. Multidomain-Modellierung (Multiphysikalische-Modellierung),

3. Eine allgemeine übersichtliche Struktur, die Objektorientierung, Mehrfachvererbung und Templates in einer einzigen Klassenstruktur vereinigt

3.2.1 Grundlagen der Modellierungssprache Modelica

Die Syntax und Sprachelemente von Modelica orientieren sich an anderen objektorientierten Programmiersprachen. die wichtigsten Sprachelemente sind

- **Variablen, und Konstanten:** Die Variablen sind Behälter für Rechengrößen, die im Verlauf eines Rechenprozesses auftreten. Sie formulieren die Gleichungen und können entweder zeitlich diskret oder kontinuierlich veränderlich sein. Die Bezeichnungen *public* und *protected* werden wie in anderen Objekt-orientierten Sprachen eingesetzt und verwendet. Das Präfix *parameter* gibt an, dass eine Variable während der Simulation konstant ist, sie kann vor oder zwischen Simulationsschritten initialisiert werden. Mit dem Präfix *constant* können Konstanten angegeben werden. Die Definition der bekannten Konstante π kann in Modelica wie gefolgt dargestellt werden;
  constant Real PI = 3.141592653589793;
  Es existiert eine Modelica Bibliothek *Modelica.Constants*, in der wichtige Konstanten wie π ,die Boltzmann-Konstante k etc. zu finden sind.
  Grundlegende Datentypen in Modelica sind in der Tabelle 3.1 dargestellt.

Tab. 3.1: Grundlegende Datentypen in Modelica

Datentypen	
Boolean	*true / false*
Integer	Ganzzahliger Wert : 42 oder –3
Real	Fließkommazahl : 2,4e-6
String	String : "Hello World"
Enumeration	Aufzählungsliteral : Farben.rot

- **Kommentare:** Die Kommentare sind wie in herkömmlichen Programmiersprachen angewendet. In Modelica existieren drei Arten der Kommentare. *"Kommentar 1"* ist ein Definitionskommentar und tritt nach der Variablendeklaration, oder am Beginn der Klassendeklaration auf, wird beim Kompilieren nicht ignoriert und ist nachher sichtbar. */* Kommentar 2*/*ist ein gewöhnlicher Kommentar im Programmcode und wird für lange Beschreibungen die über eine Zeile hinaus gehen verwendet. *//Kommentar3* wird für einzeilige, kurze Beschreibungen im Programmcode genommen , die letzten zwei Kommentare werden beim kompilieren ignoriert.

- **Klasse (Class):** Das Grundelement der Modellierung in Modelica ist die Klasse. Die Klassen bieten die Struktur für die Objekten an. Mit anderen Worten; jedes Objekt in Modelica hat eine Klasse, die seine Daten und Verhältnisse definiert. Das Wort *class* kann durch Wörtern; *model*, *record*, *block*, *connector*, *function* ersetzt werden. Diese können aber nicht unbegrenzt und überall verwendet werden. Die Begrenzungen basieren auf dem Inhalt von Klassen. Zum Beispiel : *model* kann nicht als *connector* class verwendet werden, oder als *record*, weil es

nur Daten enthält und keine Gleichungen, *block* ist dagegen mit fixer input-output Kausalität beschrieben. Jedes Icon repräsentiert ein physikalisches oder logisches Objekt, dieses wird hierarchisch erstellt oder durch Gleichungen beschrieben.

- **Software Component Model:** Wiederverwendbarkeit und Standardisierung der Komponenten verlangt ein effizientes Simulationsmodell. Der Schlüssel zu dem Weg der Entwicklung des Modells ist, dass die Klassen in Modelica basierend auf Gleichungen gebaut werden. Dadurch wird die Flexibilität wesentlich erhöht. Ein Software Component Modell hat folgende Bausteine:

1. Komponente: Komponenten sind Instanzen von Modelica Klassen. Die Komponenten müssen unabhängig von der Umgebung definiert werden, was für die Wiederverwendbarkeit notwendig ist. Sie können intern aus weiteren Komponenten bestehen; "hierarchische Modellierung". Dadurch werden komplexe Systeme übersichtlicher dargestellt. Die komplexen Systeme bestehen im Allgemeinen aus einer großen Anzahl von verbundenen Komponenten. Die Verbindung und Kommunikation der Komponenten wird durch Konnektoren ermöglicht.

2. Verbindungsdiagramm: Es ist typisch, dass große Systeme eine große Anzahl von verbundenen Komponenten haben, die wieder hierarchisch auf weitere verbundene Unterkomponenten zerlegt werden können. Um diese Komplexität zu vereinfachen kann eine graphische Darstellung von den Komponenten und Konnektoren zu Hilfe genommen werden. Diese werden in Modelica Verbindungsdiagramme benannt (Abb 3.1). Jedes Rechteck im Diagramm repräsentiert eine physikalische Komponente wie einen Widerstand, eine Kapazität, ein Ventil etc. Die Verbindungen, die durch Linien zwischen Komponenten dargestellt sind, repräsentieren die ideale physikalische Verbindungen, wie elektrische, mechanische, thermische Verbindungen.

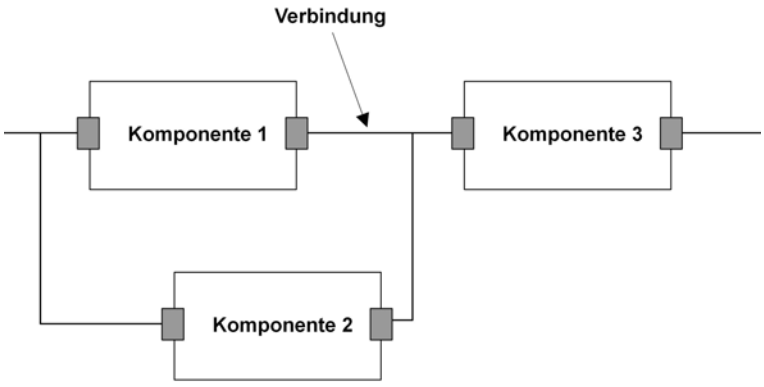

Abb. 3.1: Schematische Darstellung eines Verbindungsdiagrammes

3. Konnektor: Die Modelica Standard Bibliothek stellt Konnektoren zu Verfügung um die Kommunikation der Komponenten zu ermöglichen. Die Schnittstellen von Komponenten werden mit der Konnektor-Klasse definiert. In jedem Einsatzgebiet sind meistens zwei Konnektoren

vorhanden. Die Variablendeklarationen sind vom Syntax her ident nur das zugehörige Symbol wird anders dargestellt um zwischen den Konnektoren unterscheiden zu können, wenn sie an dem selben Komponent eingebaut sind. Modelica unterstützt auch bei den Konnektoren den Aufbau der hierarchischen Struktur der Komponenten, damit die Zusammenstellung der grundlegenden Konnektoren ermöglicht wird. Die Konnektoren definieren Variablen, die wesentlich für die Kommunikation sind. Zwei Arten von Variablen stehen in Konnektoren zur Verfügung. Potentialvariablen wie das elektrische Potential, und mit dem Attribut *flow* definierten Flussvariablen wie Strom.
Pin ist zum Beispiel ein Konnektor für die Verbindung von elektrischen Komponenten, und ist in Abb3.2 dargestellt. Die Variablen v und i sind jeweils ein Beispiel für Potential- und Flussgröße.

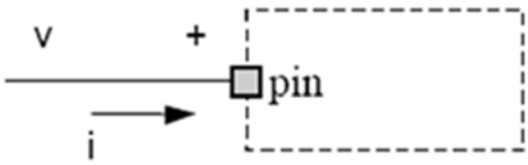

Abb. 3.2: Komponente mit elektrischen Konnektor (Pin)

Tabelle 3.2 zeigt elementare Konnektoren, die in Modelica-Bibliotheken zur Verfügung gestellt werden

Tab. 3.2: Konnektoren in Modelica

Bereiche	Potentialvariable(n)	Flussvariable(n)	Konnektor(en)	Symbol(e)
elektrisch (analog)	elektrische Potential	elektrischer Strom	Pin (positive , negativ)	
elektrisch (mehrphasig)	Vektoren von elektrischen Pins	elektrischer Strom	Plug (positive , negativ)	
elektrisch (raumzeiger)	Zwei elektrische Potential-Größen	Zwei elektrische Ströme	Space Phasor	
mechanisch (rotatorisch)	Winkel	Drehmoment	Flange (Flange_a, Flange_b)	
mechanisch (translatorisch)	Position	Kraft	Flange (Flange_a, Flange_b)	
Wärmeaustausch	Temperatur	Wärmestrom-geschwindigkeit	HeatPort (HeatPort_a, HeatPort_b)	
Block diagramm	Real Variable Integer Variable Boolean Variable		RealSignal (RealInput, RealOutput) IntegerSignal (IntegerInput, IntegerOutput) BooleanSignal (BooleanInput, BooleanOutput)	

4. Verbindung (Connection): Die Verbindungen entstehen zwischen Konnektoren äquivalenter Typen. Modelica unterstützt gleichungsbasierte akausale Verbindungen, dadurch werden die Verbindungen als Gleichungen berücksichtigt. Wegen der Akausalität muss die Flussrichtung im Konnektor bekannt sein. Zum Beispiel bei einer elektrischen Konnektor *Pin* sind die Verbindungseigenschaften für Potential- und Flussvariablen unterschiedlich.
   Die Flüsse summieren sich zu Null; die zweite Kirchhoffsche Regel.
   Die Potentiale haben den gleichen Wert; erste Kirchhoffsche Regel.
   Die Verbindung zwischen zwei elektrischen Konnektoren (*Pin*) wird wie folgt berücksichtigt:

   $pin1.v = pin2.v$ //Setzt die Spannungen im Verbindungspunkt gleich Null
   $pin1.i + pin2.i = 0$ //Summiert die Ströme im Verbindungspunkt zu Null

5. Teilklasse (Partial Class): Die Bezeichnung *partial class* steht für Klassen, die keinen vollständigen Gleichungssatz enthalten, die ihr Verhalten beschreiben. Zum Beispiel eine Teilklasse Namens $TwoPin$, welche zwei elektrische Konnektoren (p und n) und deren Verlauf ($p.v = n.v$ und $p.i + n.i = 0$) enthält, kann von anderen elektrischen Komponenten durch Vererbung verwendet werden.

- **Gleichungen:** Die Hauptverwendung der Gleichungen in Modelica ist die Repräsentation von Zusammenhängen von physikalischen Größen.
  Attributzuweisungen repräsentieren Gleichungen
  Verbindungen zwischen Objekten generieren Gleichungen.
  Die Gleichungen können entsprechend ihrer Syntax in vier unformale Kategorien gegliedert werden:

1. Normale Gleichungen: $x = 2 + y$
   Die Verbindungsgleichung *connect*, und die anderen syntaktischen Gleichungen gehören auch zu dieser Gruppe.

2. Deklarationgleichungen: Real $x = 2.0$
   Deklaration von Variablen, Parameter oder Konstanten

3. Modifikationsgleichungen: $(x(start = 0.2))$
   Modifikation der Attributen in Klassen

4. Initialisierungsgleichungen (Initial Equations) : $xstart = 0.2$
   when initial() $x = xstart$; *end when*;
   Diese Gleichungen werden angegeben, um Initialisierungsprobleme zu lösen.

- **Algorithmen und Funktionen:** Obwohl die Gleichungen genau passend für die physikalische Modellierung sind, treten immer wieder Fälle auf, wo Algorithmen (Befehlsfolgen) statt Gleichungen zum Einsatz kommen. Diese Algorithmen befinden sich dann im *equation* (Gleichung) Teil des Modells. Ein Algorithmus wird mit; *equation, public, protected, algorithm, initial, oder end* abgeschlossen. Die Algorithmen ermöglichen es die Zuweisungen abzuarbeiten. Alle Zuweisungen innerhalb derselben *algorithm* Sektion werden als eine Menge von n Gleichungen betrachtet, wobei n die Anzahl der unterschiedlichen Variablen ist, die auf der linken Seite der Zuweisungen auftreten [26].
  Funktionen sind teilweise in Modelica verfügbar, aber es ist auch möglich selbst definierte Funktionen zu bilden. Modelica Funktionen sind deklarative mathematische Funktionen, sie liefern als Rückgabewert immer den selben Wert, solange die Eingabewerte gleich angegeben sind. Die Funktion hält also ihre Semantik oder Bedeutung unabhängig davon, wo sie referenziert oder aufgerufen wird. Das deklarative Verhalten der Funktionen bedeutet, dass die Funktionen keine Daten speichern. Dieses Verhalten ist die Voraussetzung um kontinuierliche und differentielle Bedingungen sicher zu stellen, die für die Lösung vom kontinuierlichen Teil des Gesamtsystems nötig ist. Trotz Funktionen, sind die Schleifen (if, *when*, *etc.*) innerhalb von Funktionsblöcken nicht deklarativ.

- **Bibliotheken:** Mit einem in der Sprache Modelica enthaltenen Package-Konzept ist es möglich, ähnlich dem Filesystem Modelle geordnet zu verwalten. Jeder Anwender kann das entsprechend seinen Erfordernissen einrichten. Gleichzeitig mit der Sprachentwicklung von Modelica wird an umfassenden Bibliotheken gearbeitet. Diese Bibliotheken verfügen über Spezialbibliotheken für entsprechende Fachgebiete. Die Smart Electric Drives Library ist eine von diesen Bibliotheken, und wurde zur Simulationen von elektrischen Antriebe mit der Sprache Modelica entworfen.
  Die Modelica Standard Library gliedert sich in allgemeine Pakete, die in der Tabelle 3.3 angegeben sind. Am weitesten ausgebaut und verfügbar sind die Teilpakete der Standardbibliothek für translatorische und rotatorische Mechanik, für analoge elektrische Bauelemente und Maschinen sowie die regelungstechnischen Blöcke.

Tab. 3.3: Modelica Standard Library

Modelica Standard Library	
Blocks	Grundlegende Kontrollblöcke für input/output (kontinuierliche, diskrete, logische, Tabellenblöcke)
Constants	mathematische und physikalische Konstanten wie; π, ϵ, σ, k
Electrical	elektrische und elektronische Modelle (analog, digital, el. maschinen, mehrphasig)
Icons	Grundelemente für graphische Darstellung
Math	mathematische Funktionen wie; sin, cos und Funktionen auf Matrizen und Vektoren
Mechanics	mechanische Komponenten (rotatorisch, translatorisch)
SIunits	Definition der SI-Einheiten

3.2.2 Smart Electric Drives Library (SED)

Smart Electric Drives (SED) Library ist eine Bibliothek für die Simulation elektrischer Antriebe in komplexen elektromechanischen Systemen. Neben zahlreiche Komponenten für die spezielle Modellbau existieren in SED fertige "ready-to-use" Antriebmodelle. Diese Modelle beinhalten alle nötigen Ausstattungen und Eigenschaften von modernen elektrischen Antrieben. Die Beispiele der Anwendungen, für die SED zum Einsatz kommt, sind: Robotik, Traktionsysteme und Hilfskomponenten in Fahrzeugen bzw. Hybridfahrzeugen [28],[29],[13].

Alle für einen modernen elektrischen Antrieb nötigen Komponenten sind in der SED Library vorhanden. Die Tabelle 3.4 gibt die Struktur der Library an.

Tab. 3.4: Modelica SED Library

Modelica SED Library	
QuasiStationaryDrives	Antriebe mit dem integriertem Umrichter und Regelung inklusive Strom- und Spannungsabgrenzung
TransientDrives	Komponenten für die Regelung der Antriebe
Converters	Umrichter mit Spannungsversorgung
Sources	Batterien, Brennstoffzellen, Doppelschicht-Kondensatoren
Process Controllers	Komponenten für die Geschwindigkeits- und Positionskontrolle
Loads	Ladungsysteme
Sensors	Sensoren
Interfaces	Buskonnektoren für elektrische Signalübertragung, und weitere Schnittstellen
AuxiliaryComponents	Allgemeine Reglerkomponenten
Icons	Symbolen

- Quasistationäre Maschinen: Die Maschinen sind drehmomentgeregelt. Alle transienten Vorgänge ausser mechanische sind bei diesen Modellen vernachlässigt. Deswegen arbeiten diese Modelle schneller als die Maschinen, bei denen transiente Effekten berücksichtigt werden. Diese Eigenschaft ist bei der Betrachtung des Energieverbrauchs oder Effizienz der Maschine sehr vorteilhaft. Ein weiterer Vorteil dieser Modellen ist, dass die Regelungen nicht parametriert werden müssen, weil die Parameter der Regelung in den stationären Gleichungen des Antriebes nicht vorkommen. Die zugehörige Komponenten sind: 1) quasistationäre Asynchronmaschine, 2) quasistationäre Permanent-Magnet-Synchronmaschine, 3) quasistationäre Permanent-Magnet-Gleichstrommaschine, 4) quasistationäre fremderregte Gleichstrommaschine
  Alle Maschinen werden mit Gleichstrom versorgt. Bei den Drehfeldmaschinen ist das der Anschluss für die Zwischenkreisspannung.

- Transiente Maschinen: Die Maschinen enthalten alle Antriebskomponenten die nötig sind für eine Antriebsstruktur mit Berücksichtigung der transienten Effekten. Es existieren vier Maschinen, die als ready-to-use Modell zur Verfügung stehen. Diese enthalten DC/DC, oder DC/AC Konverter, eine Maschine, und grundlegende Antriebselemente wie ein Messbox, Buskonnektor etc.

  Als Beispiel für ein ready-to-use Modell kann eine transiente Asynchronmaschine mit integrierter Regelung und Umrichter als fertiger Antrieb in der Abbildung 3.3 mit Gleichspannungsversorgung angegeben werden. Das hellblaue Symbol repräsentiert den Antrieb mit allen nötigen Antriebselementen.

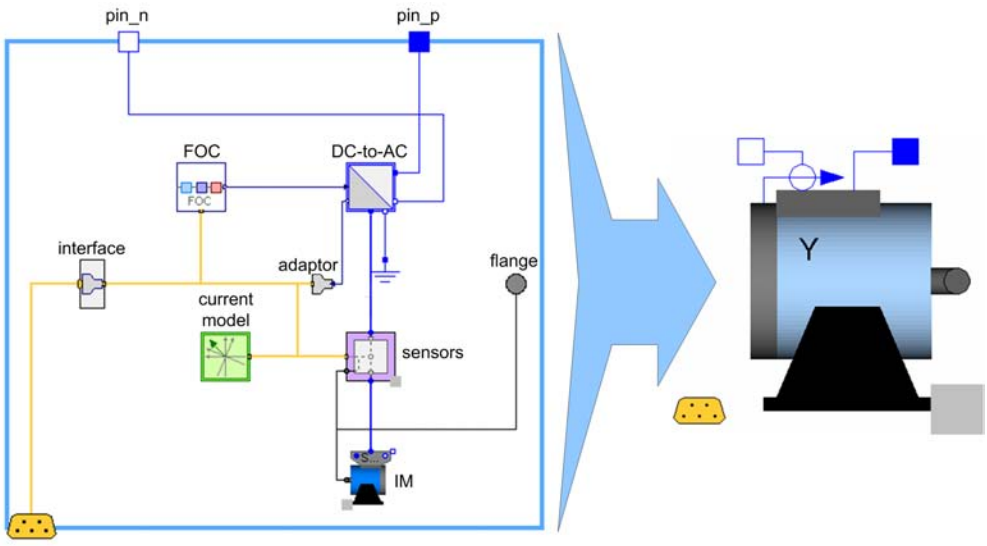

Abb. 3.3: Transiente Asynchronmaschine mit Gleichstromversorgung [1]

- Allgemeine Reglerkomponenten (*AuxiliaryComponents*): Die wichtigsten Transformationen wie von Stranggrößen auf Raumzeigergrößen (mit Nullsystem) oder von kartesischen Koordinatensystem auf Polarkoordinaten werden mit Komponenten dieses Packages ermöglicht. Zusätzlich existieren spezielle Funktionen für die Parameternachführung der Regelung.

- Buskonnektoren: Mit dem Ziel mehrere Kontrollsignale zu gruppieren bzw. sammeln und diese zu messen sind die meisten SED Library Komponenten mit einem Buskonnektor zusammengefasst. Aufgrund unterschiedlicher Reglerparameter und Variablen der Maschinen, existiert ein internes Bussystem angepasst an jede Maschine für interne Anwendungen. Es gibt außerdem einen allgemeinen Buskonnektor im Package Interfaces: *ControlBus* für externen Signalaustausch. Mit Hilfe des *ControlBus* wird die Verbindung zwischen dem drehmomentgesteuerten Antriebssystem und des Geschwindigkeit-, Lagereglers udgl. ermöglicht. Dieses externe Bussytem wird über ein Interface mit dem internen Bussytem verbunden [12]. Die Abbildung 3.4 veranschaulicht dieses Verhalten der Buskonnektoren. Der größte Vorteil der Buskonnektoren ist, dass die Modelle übersichtlicher bleiben während der Modellaufbau immer komplexer wird, weil die Anzahl der Verbindungen mit Hilfe des Busses extrem reduziert werden. Ein weiterer Vorteil kommt im Simulationsschritt vor. Alle Variablen, die auf dem Bus liegen, werden in einer eigenen Gruppe gesammelt und sind besser zu verfolgen bzw. zu vergleichen. Die Busse sind so modelliert, dass die wichtigsten Variablen für das Antriebsystem im Variablenbrowser sichtbar sind und ausgewählt werden können.

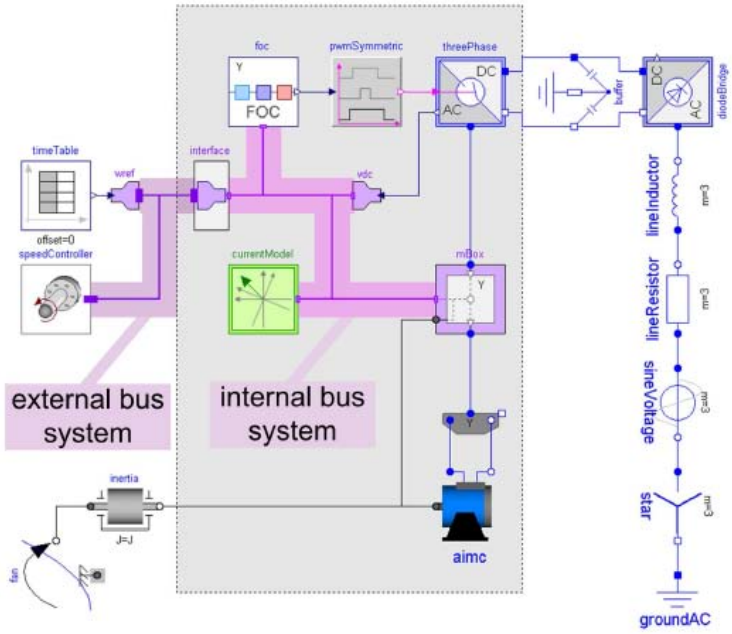

Abb. 3.4: SED Simulation einer drehzahlgeregelten ASM mit externen und internen Bussystem [12]

Ausserdem befinden sich zwei Batterietypen im Package Sources. Die Umrichter sind auch in zwei unterschiedliche Varianten zu finden. Die erste Variante enthält *power balance converter*, bei denen das Schalten der Halbleiter nicht modelliert ist und dadurch die Arbeitsgeschwindigkeit hoch ist. In der zweiten Variante *ideal switching converter* werden die Umrichter mit ideal abschaltbaren Ventilen berücksichtigt. Diese sind aufgrund der auftretenden Iterationen rund um die Schaltaugenblicke im Vergleich zur vorherigen Variante langsamer.

3.2.3 Antriebsstruktur in der SED

Die in der Diplomarbeit untersuchte Antriebsstruktur ist eine drehmomentgeregelte Asynchronmaschine mit feldorientierter Regelung (Abbildung 3.3). Das Modell besteht aus einem Inverter, einer feldorientierten Regelung mit Strom- und Spannungsbegrenzung und Flussschwächung, einem sogennanten Flussmodell: Strommodell für die Flussbestimmung, einer Messbox für die Erfassung der Spannungen, Ströme und der Winkellage des Rotors. Die Modelle kommunizieren miteinander über den internen Bus. Der Umrichter speist die Asynchronmaschine basierend auf dem, aus feldorientierter Regelung gelieferten, Referenzspannungssignal $vRef$. Die Messbox $mBox$ liefert die nötigen Maschinengrößen für die feldorientierte Regelung. Das Strommodell ermittelt den Betrag, und den Winkel des Rotorflussraumzeigers. Die Welle ist über ein Massenträgheitsmoment *inertia* mit einem Lastmoment belastet. Mit dem *speedController* und *timeTable* werden jeweils das Referenzdrehmoment τ_{ref} und die Referenzdrehzahl ω_{ref} an dem externen Bus gelegt, die dann über dem Interface vom internen Bus empfangen werden. Weiters werden die einzelnen Modelle des Antriebes beschrieben:

Das Modell *FOC*, feldorientierte Regelung, befindet sich in der Bibliothek im Package *TransientDrives*, und beinhaltet alle nötigen Komponenten für die Durchführung der feldorientierten Regelung der käfigläufer Asynchronmaschine. Die Abbildung 3.5 veranschaulicht die Struktur des Modells. Wie in der Abbildung ersichtlich enthält das Modell FOC weitere Modelle. Der Drehmoment- /Flussregler *TorqueFluxController*, der Stromlimiter *CurrentLimiter*, der Stromregler *CurrentController*. Diese werden weiters kurz erläutert:

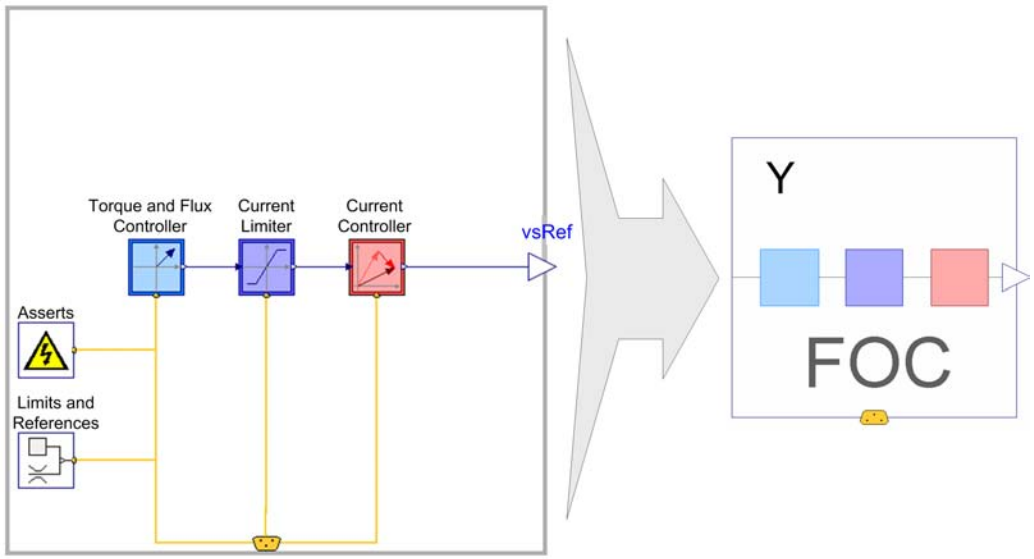

Abb. 3.5: Feldorientierte Regelung System [1]

- Das Modell *TorqueFluxController* regelt den Statorstrom mit Hilfe des Referenzdrehmomentes und des Referenzbetrages der magnetischen Rotorflussverkettung. Das Modell hat Komponenten, die am Schluss ein Referenzstromsignal $isRef$ für den *CurrentLimiter* erzeugen. Die Komponenten des *TorqueFluxController* sind: 1) Das Modell $MaxTorqueVoltageCondition$ regelt I_{sy} so, dass während der Flussschwächung das geforderte Drehmoment der Maschine erhalten bleibt. 2) $FluxWeakening$ arbeitet mit dem vom $LimitsAndReferences$ abgeholten Signal $fluxLevel$ und stellt den Referenzfluss entsprechend ein. Wenn die erforderliche Statorspannung mit größer werdender Frequenz den Maximalwert der erlaubten Maschinenspannung oder Inverterspannung überschreitet, bringt das Modell die Maschine in den Feldschwächbereich in dem es den Fluss kleiner macht damit größere Frequenzen ohne Spannungserhöhungen erreicht werden. Die Maschine kann mit Flussschwächung in jedem Drehzahlbereich arbeiten ohne dass die Spannungen unzulässig groß werden. 3) $FluxController$: mit Hilfe der Information über die Feldschwächung, die vom Modell $FluxWeakening$ geholt wird, ermittelt das Modell die flussbildende Komponente des Statorstromes I_{sx}.

  Das Modell erlaubt dem User die Flussschwächung ein/auszuschalten. Falls diese eingeschaltet ist, dann wird mit größer werdende Drehzahl und bei der maximalen Spannung kontinuierlich das Fluss geschwächt.

- Das Modell *CurrentLimiter* limitiert den Statorstrom im Bezug auf dem maximal erlaubten

Statorstrom $iMachineMax$, welcher vom Bus abgeholt wird.

- Das Modell *CurrentController* regelt den Statorstrom und erzeugt die Referenzstatorspannung $vsRef$. Ein Entkopplungmechanismus *DecouplingNetwork* eliminiert die Verkopplung der beiden Achsen und ermöglicht dadurch eine effiziente schnell laufende Regelung. Die benötigten Signale wie Rotorflusswinkelgeschwindigkeit ω_f, magnetischer Rotorfluss Ψ_r, und der aktuelle Statorstrom I_s werden vom Bus geholt. Der Statorstrom und der Referenzstrom $iMachineMax$, der als Limit für Maschinenstrom angegeben wird, werden abgeglichen und abschliessend mit einem PI Regler geregelt. Mit dem Entkopplungmechanismus und zusätzlich vom Bus abgeholten Signalen erzeugt das Modell die Referenzspannung $vsRef$.

- Das Modell $LimitsAndReferences$ informiert das interne Bussystem über den Grenzwert des Statorstroms, Statorspannungs und gibt das Signal $fluxLevel$ als Referenzwert für den Rotorfluss an.

Das Strommodell $CurrentModel$ ist die Realisierung des im Unterabschnitt 2.2.3 erläuterten Strommodells in Modelica und bezieht sich auf die Gleichungen (2.67) und (2.68). Das vorhandene Strommodell wird wie in der Abb. 3.6 gegeben implementiert.

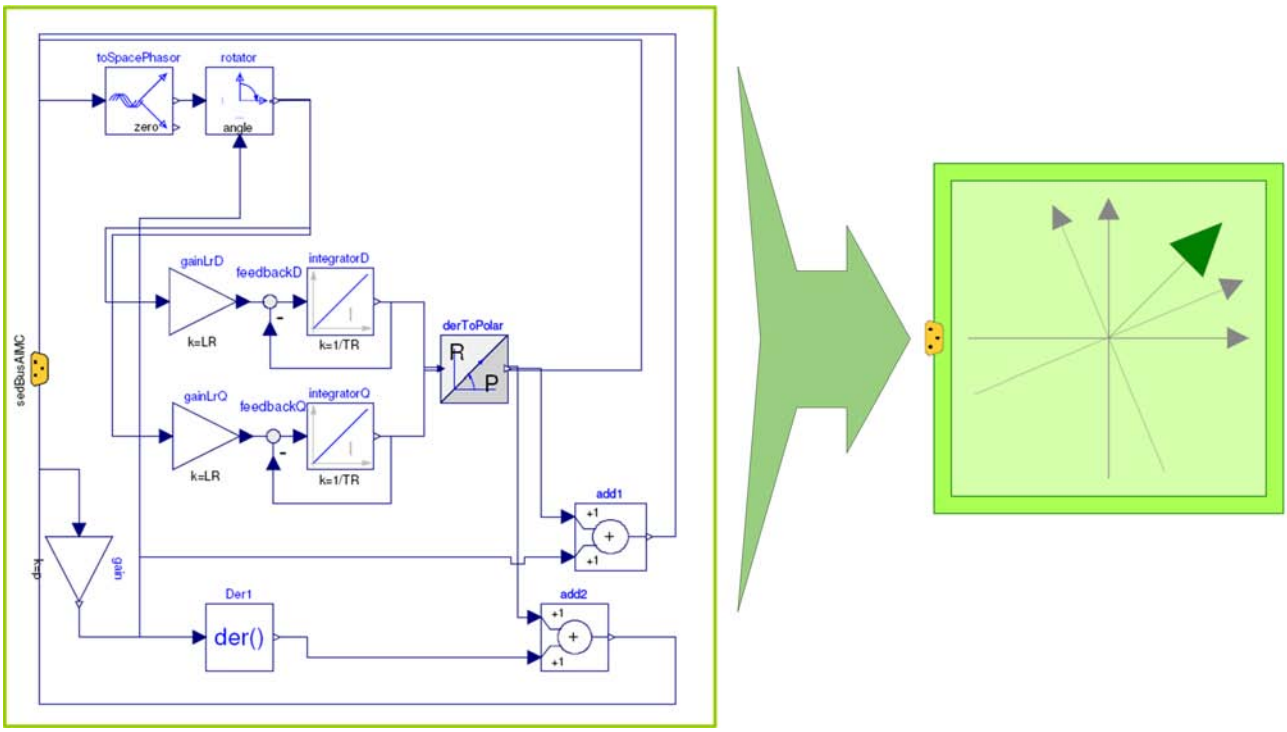

Abb. 3.6: Strommodell in Modelica

Mit Hilfe des Strommodells wird der Betrag, der Winkel und die Winkelgeschwindigkeit des Flussverkettungsraumzeigers im rotorfesten Koordinatensystem festgelegt und abschließend in das statorfeste Koordinatensystem transformiert. Der Statorstrom wird vom Bus geholt und mit Hilfe der Transformationsbox als Raumzeigergröße weiterverarbeitet. Abschließend wird er mit der

ebenfalls vom Bus abgeholten Information über die Winkellage des Rotors γ_m in das rotorfeste Koordinatensystem transformiert. Diese Größe wird dann mit Rotorwiderstand L_r multipliziert. Die Differenz zwischen dem multiplizierten Term und dem Term mit dem Flussverkettungsraumzeiger $\frac{1}{T_R} \cdot \Psi_r$ entspricht der zeitlichen Ableitung des Flussverkettungsraumzeigers. In der Realisierung des Modells wird dem Integrator eine rückgekoppelte Integration zugeführt, um den Rotorflussverkettungsraumzeiger zu bekommen. Nach der Transformation dieser komplexen Größe ins Polarkoordinatensystem, wird der Betrag $|\underline{\Psi}_r|$, der Winkel γ_r und die elektrische Rotorwinkelgeschwindigkeit ω_r des Rotorflussverkettungsraumzeigers ermittelt. Zum Winkel γ_r wird der mechanische Winkel γ_m addiert um den Winkel im statorfesten Koordinatensystem γ_f zu bekommen, weiters wird zur elektrischen Rotorwinkelgeschwindigkeit ω_r die mechanische Winkelgeschwindigkeit ω_m addiert um die Rotorflusswinkelgeschwindigkeit ω_f der Maschine zu bekommen. Diese Größen werden dann auf den Bus gelegt.

Das Spannungsmodell *VoltageModel* ist die Realisierung des im Unterabschnitt 2.2.2 erläuterten Spannungsmodells in Modelica und basiert auf der Gleichung (2.63). Die Implementierung des Spannungsmodells in Modelica wird in der Abb. 3.7 veranschaulicht.

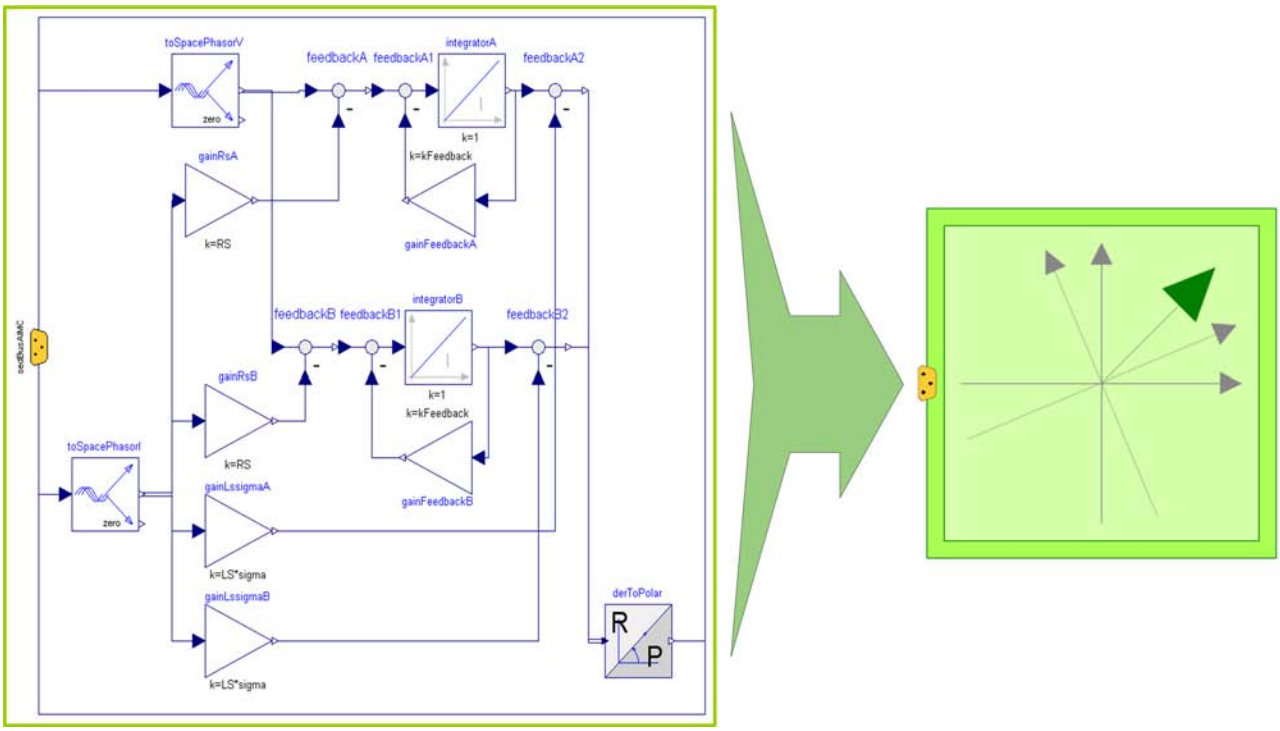

Abb. 3.7: Spannungsmodell in Modelica

Das System ist im statorfesten Koordinatensystem implementiert, deswegen ist keine Information über die Winkellage oder Winkelgeschwindigkeit des Rotors notwendig. Die vom Bus abgeholte Statorspannung wird wie im Strommodell mit der Transformationsbox in das Raumzeigersystem transformiert. Der ebenfalls transformierte und mit dem Statorwiderstand R_s multiplizierte Statorstromraumzeiger wird von dem Spannungsraumzeiger abgezogen. Nach erfolgreicher Integration mit Hilfe der Rückkopplung wird der Flussverkettungsraumzeiger ermittelt. Die Rückkopplung ist

hier deshalb notwendig, um die Neigung zur Instabilität der offenen Integration zu unterdrücken. Der so ermittelte Flussverkettungsraumzeiger wird dann mit der Transformationsbox in das Polar-Koordinatensystem transformiert und somit als Betrag $|\Psi_r|$, Winkel γ_f und Winkelgeschwindigkeit ω_f auf den Bus gelegt. Das Modell arbeitet wie bereits im Unterabschnitt 2.2.2 erwähnt nur mit großen Drehzahlen zufriedenstellend.

3.2.4 Modellerweiterung

Die Erweiterung des Spannungsmodells $VoltageModel$ und des Strommodells $CurrentModel$ basiert auf der Idee, die derzeit in den Modellen vorhandenen konstanten Parameter mit einer Tracking-Box $Tracking$ nachzuführen. Die Tracking-Box übernimmt die Aufgabe der Implementierung der in der Arbeit untersuchten Trackingmethode in Modelica, die in der Abbildung 2.9 beschrieben wurde. Es existieren zwei Modellen für die Flusserfassung. Im vorhanden Modell wird aber nur das Strommodell verwendet und das Spannungsmodell wegen seines beschränktes Arbeitsbereich weggelassen. Das Prinzip der untersuchten Methode für die Parameternachführung basiert aber auf Modellvergleichen, was das Zusammenarbeiten beider Modelle verlangt. Die Auswirkungen der Probleme des Spannungsmodells bei niedrigen Drehzahlen können mit Hilfe eines von der Drehzahl und vom Strom abhängigem Modells, das das Spannungsmodell steuert, unterdrückt werden. Das ermöglicht den erforderlichen parallel-Betrieb der Modelle.

3.3 Zusammenfassung

Die objektorientierte Modellierungssprache Modelica ermöglicht hoch genaue und einfache Lösungen von komplexen Systemen. Modelica ist eine gleichungsorientierte Sprache. Diese Eigenschaft ermöglicht die Implementierung aller technischen und physikalischen Systemen, die sich mit algebraischen und gewöhnlichen Differentialgleichungen beschreiben lassen. Die Modelle werden mit Gleichungen beschrieben und orientieren sich somit am eigentlichen Verhalten von realen Systemen. Diese enge Verbindung macht Modelica besonders attraktiv gegenüber anderen Simulationsprogrammen. Eine besonders vorteilhafte Eigenschaft ist die Weiterentwicklung von Bibliotheken sowie die Erweiterung von Modellen. Die entwickelten Bibliotheken verfügen über Spezialbibliotheken für entsprechende Fachgebiete. Die *Smart Electric Drives* (SED) Library ist eine von diesen Bibliotheken, und wird zur Simulationen von elektrischen Antrieben eingesetzt.Die SED Library verfügt über alle für einen modernen elektrischen Antrieb nötigen Komponenten. Die in der Diplomarbeit untersuchte Antriebsstruktur ist eine drehmomentgeregelte Asynchronmaschine mit feldorientierter Regelung und wurde mit Komponenten der SED Library implementiert. Sie besteht aus einem Inverter, einer feldorientierten Regelung mit Strom- und Spannungsbegrenzung und Flussschwächung, einem "Strommodell" für die Flussbestimmung, einer Messbox für die Erfassung der Spannungen Ströme und Winkellage des Rotors. Die vorhandenen Flussmodelle arbeiten mit konstanten Parametern, und die Temperaturschwankungen der Maschine werden auch nicht berücksichtigt. Für die Optimierung werden diese beiden Modelle so umgestellt, dass sie statt konstanten Parametern mit vom Bus abgeholten und nachgeführten Parameter arbeiten. Diese Aufgabe des Parametertrackings nach der im Unterab-

schnitt 2.4.3 untersuchten Methode wird durch die Tracking-Box erfüllt. Das Spannungsmodell wird mit Hilfskomponenten stabilisiert, womit ein parallel-Betrieb beider Modelle und der erforderliche Modellvergleich ermöglicht werden.

4 Implementierung der Parameternachführung in Modelica

Die Parameternachführung wird in der Tracking Library mit Hilfe einer Tracking-Box *Tracking* ermöglicht. Wie in Unterabschnitt 3.2.4 angegeben, basiert die Parameternachführung auf Modellvergleichen. Das Spannungsmodell, das bei dem Antrieb ohne Parameternachführung wegen seines beschränkten Arbeitsbereichs nicht eingesetzt wurde, kann mit Hilfe der entsprechenden Entwicklungen seine Aufgabe als Referenzmodell in der Parameternachführung unter bestimmten Einschränkungen erfüllen. Die Temperaturschwankungen im Rotor bzw. Stator werden mit einer Temperatur-Box *Temperature* berücksichtigt. Die Kommunikation der Modelle erfolgt nach wie vor über Busverbindungen.

4.1 Struktur und Implementierung

Die Antriebsstruktur der SED Library wurde im Unterabschnitt 3.2.3 beschrieben. Die neue Antriebsstruktur (Abbildung 4.1) besteht wiederum aus der feldorientierten Regelung *FOC*, dem Inverter, der Messbox , aber statt *CurrentModel*, aus den beiden für die Parameternachführung erweiterten Flussmodellen sowie zusätzlich aus der Tracking-Box und der Temperatur-Box. Das Arbeitsprinzip des Antriebes ist großteils wie das Arbeitsprinzip des im Unterabschnitt 3.2.3 beschriebenen Antriebes. Die zwei für die Parameternachführung neu entwickelten Flussmodelle; Spannungsmodell und Strommodell arbeiten für die Flusserfassung und durch die Modellvergleiche für die Parameternachführung. Auf der anderen Seite benötigen die neuen Flussmodelle die nachgeführten Parameter bzw. die aktuellen Temperaturen der Maschine um die benötigte Flusserfassung zu liefern. Die aktuelle Rotor- bzw. Statortemperatur wird mit der Temperatur-Box bestimmt, die wiederum mit den von der Tracking-Box abgeholten nachgeführten Parametern arbeitet.

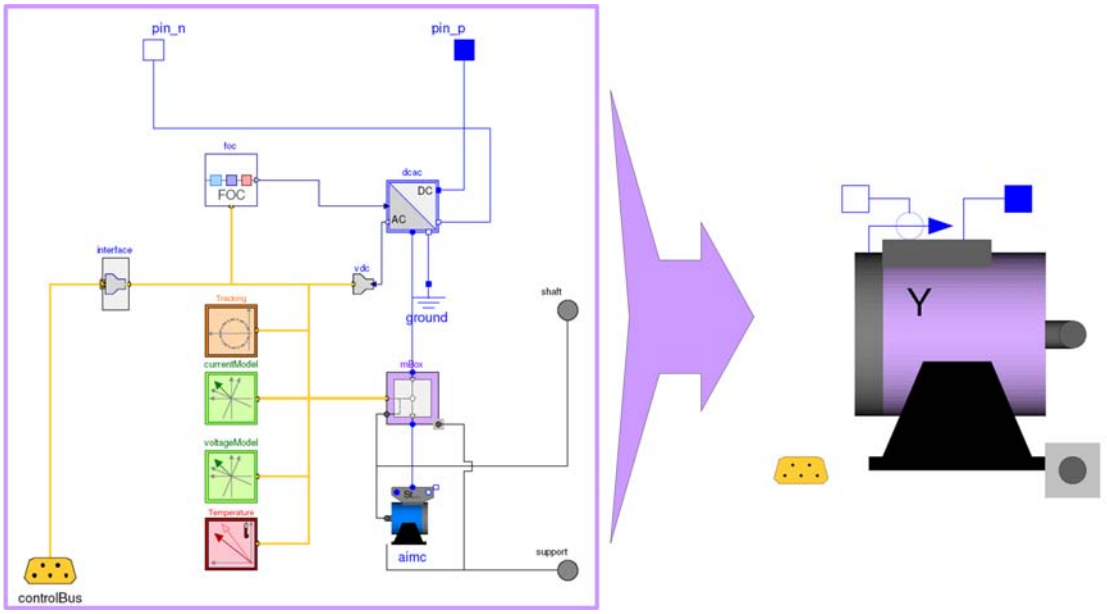

Abb. 4.1: Die Antriebsstruktur mit Parameternachführung

4.2 Modellaufbau

Die Entwicklung des Antriebes basiert hauptsächlich auf der Tracking-Box. Die *Tracking-Box* beinhaltet die in Modelica implementierte Methode, die im Unterabschnitt 2.4.3 (Abbildung 2.9) erläutert wurde. Die Abbildung 4.2 veranschaulicht die Struktur der *Tracking-Box* in der Tracking Library.

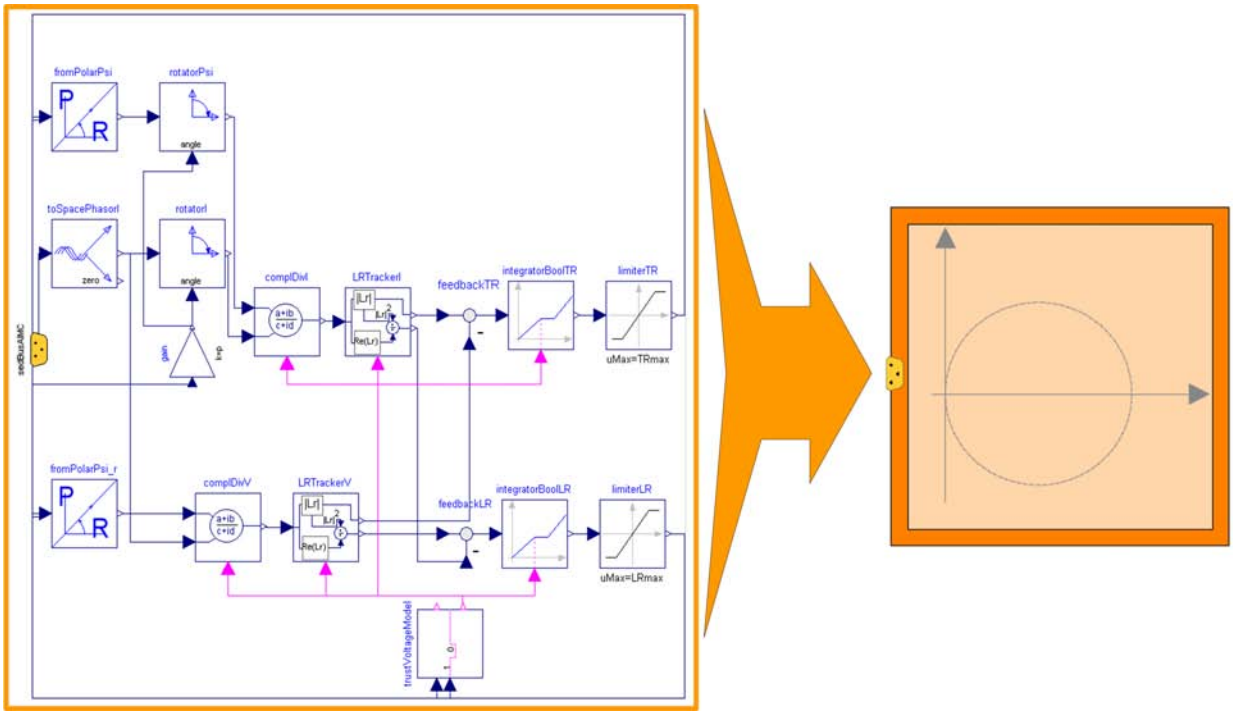

Abb. 4.2: Tracking-Box

Die Flussverkettungsraumzeiger des Strommodells $\underline{\Psi}_r^{r^i}$ und Spannungsmodells $\underline{\Psi}_r^{s^v}$, sowie der Statorstrom $\underline{I}_s^s$ und die Winkellage des Rotors γ_r werden vom Bus geholt. Mit Hilfe der Transformationsbox wird von jeder Größe die benötigte Transformation in einem Raumzeiger durchgeführt. Weil das System des Strommodells im rotorfesten Koordinatensystem untersucht wird, müssen zusätzlich vor der Division des vom Strommodell gelieferten Flussverkettungsraumzeigers $\underline{\Psi}_r^s$ durch den Statorstromraumzeiger $\underline{I}_s^s$, beide Größen noch in das rotorfeste Koordinatensystem transformiert werden. Es wird dann für jeden Flussverkettungsraumzeiger die entsprechende Division durch den entsprechenden Statorstrom $\underline{I}_s$ berechnet. Die Division erfolgt mit einer speziellen Divisions-Box *ComplDiv*, die für die Tracking entwickelt wurde, um die Division komplexer Größen durchzuführen. Die Erfassung des Durchmessers jedes Kreises sowie der Beträge der komplexen Ersatzgrößen $\underline{L}_r$ erfolgt mit *LRTracker*. Die Differenzen werden entsprechend der Gleichungen im Unterabschnitt 2.4.3 gebildet und jeweils zum zugehörigen I-Regler geführt, um die Nachführung abzuschließen. Der Limiter im Modell dient dazu, dass das Modell niemals die Zahl "0", zu kleine oder zu große Zahlenwerte als Parameter auf dem Bus legt.

Die Modelle in der *Tracking-Box* sind mit Booleschen Eingängen gesteuert. Dieser Eingang wird vom *TrustVoltageModel* erzeugt, das die Modelle abhängig von der elektrischen Winkelgeschwindigkeit des Rotors ω_r und vom Statorstrom $\underline{I}_s$ steuert. Das hilft bei den Rechnungen mit komplexen

Größen wie Strom und Fluss dazu, dass das Ergebnis in einem vernünftigen Bereich bleibt. Weil das Modell erst dann ein *"true"* Signal liefert und die gesteuerten Modelle zu arbeiten beginnen, wenn diese Größen das vorgegebene Limit überschreiten und sich in einem vertrauenswürdigen Bereich befinden (siehe Unterabschnitt 4.3). *TrustVoltageModel* wird auch im Spannungsmodell eingesetzt, um die Probleme bei niedrigen Drehzahlen zu beheben.

4.3 Komponenten

Die vorhandenen Flussmodelle in der SED Library arbeiten mit konstanten Parametern. Damit sie mit nachgeführten Parametern arbeiten, wurden sie in der Tracking Library wie folgt erweitert:

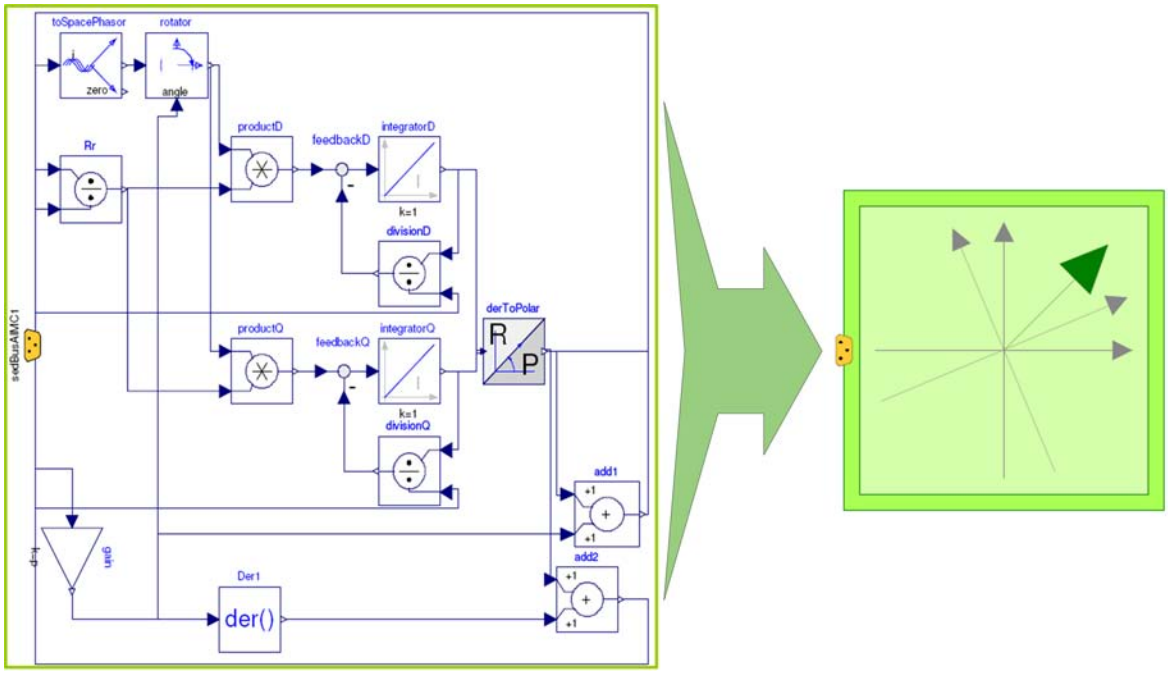

Abb. 4.3: Strommodell in Tracking Library

Das Strommodell *CurrentModel* (Abbildung 4.3, Unterabschnitt 2.2.3) holt vom Bus, wie in Unterabschnitt 2.4.3 erläutert, die nachgeführten Parameter L_r und T_R ab, um den Rotorwiderstand R_r zu bestimmen. Diese Größe wird dann sowohl mit dem Realteil als auch mit dem Imaginärteil des Statorstroms $\underline{I}_s$ multipliziert und der Term $R_r \cdot \underline{I}_s$ wird gebildet. Anschließend wird wieder mit Hilfe der vom Bus abgeholten Rotorzeitkonstante T_R der Term $\frac{1}{T_R} \cdot \Psi_r$ gebildet. Die Differenz dieser Terme ist wiederum die zeitliche Ableitung des Rotorflussverkettungsraumzeigers $\frac{d\underline{\Psi}_r}{dt}$. Wie im vorhandenen Strommodell strukturiert wurde, wird bei dem entwickelten Modell auch eine rückgekoppelte Integration verwendet, um den Rotorflussverkettungsraumzeiger $\underline{\Psi}_r$ zu bekommen.

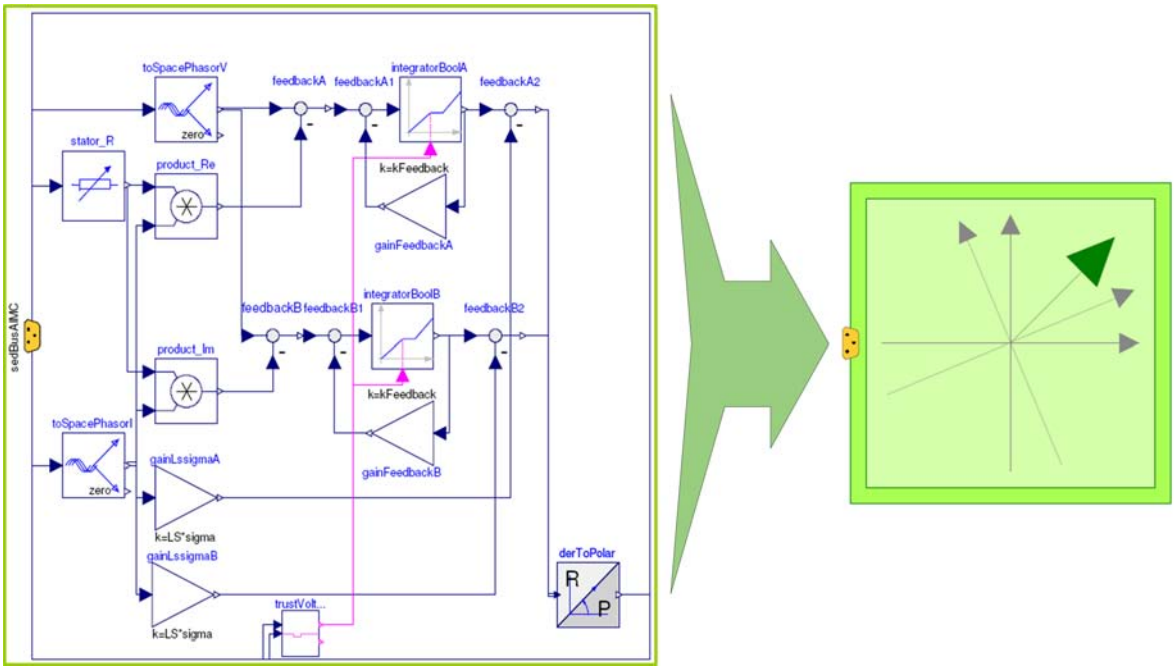

Abb. 4.4: Spannungsmodell in Tracking Library

Das Spannungsmodell *VoltageModel* (Abbildung 4.4) arbeitet im Gegensatz zum alten Spannungsmodell nicht mit dem fixen Statorwiderstand R_s. Es holt vom Bus die aktuelle Statortemperatur T_s, um den aktuellen Statorwiderstand R_s zu bestimmen. Die Bestimmung erfolgt ausgehend von der Temperaturabhängigkeit des Statorwiderstandes R_s, wie vorher in der Gleichung (2.81) formuliert wurde. Der so bestimmte Statorwiderstand R_s wird dann mit dem Statorstrom $\underline{I}_s$ multipliziert und der Term $I_s \cdot R_s$ gebildet. Weil die statorseitigen Streuinduktivitäten kaum einen Einfluss auf die Regelung haben, werden sie nicht nachgeführt. Die Integratoren des vorherigen Modells wurden durch Integratoren mit Booleschen Eingängen ersetzt, mit dem Ziel, den Integrator mit Hilfe eines Steuersignals im niedrigen oder hohen Drehzahlbereich abzuschalten. Das Boolesche-Signal wird aus dem *TrustVoltageModel* geliefert. Das ermöglicht die Behebung der Probleme, die das Modell in niedrigen Drehzahlbereichen hat. Der Integrator arbeitet somit nur im erlaubten Arbeitsbereich, der für den Parallelbetrieb vorgesehen ist.

Das *TrustVoltageModel* (Abbildung 4.5) hat die Aufgabe *Tracking* und *VoltageModel* mit Booleschen Signalen zu steuern. *VoltageModel* wird mit dem Signal *useIntegrator* und *Tracking* wird mit dem Signal *trustVoltage* gesteuert. Der einzige Unterschied zwischen den beiden Signalen ist eine Zeitverzögerung: *delayTime*. Das Signal *trustVoltage* wird damit erst nach *delayTime* auf *true* geschaltet. Mit Hilfe dieser Zeitverzögerung fängt *Tracking* erst dann an zu arbeiten, wenn das Spannungsmodell die transiente Einschaltphase schon überschritten hat und den richtigen Fluss liefert. Das Modell liefert nur dann den Wert *"true"*, wenn sich die Drehzahl der Maschine nicht in dem Bereich zwischen *wLimit* und *-wLimit* befindet, und der Strom größer als vorgegebenen *iLimit* wird. Die Begrenzung *iLimit* des Stromes liegt etwas oberhalb des Magnetisierungsstromes der Maschine, weil das Modell mit Null Last oder schwacher Last nicht zufriedenstellend arbeitet. Es kann mit Hilfe des Limits sichergestellt werden, dass die Parameter ihre vorhandenen Werte behalten und

nicht auf falsche Werte nachgeführt werden.

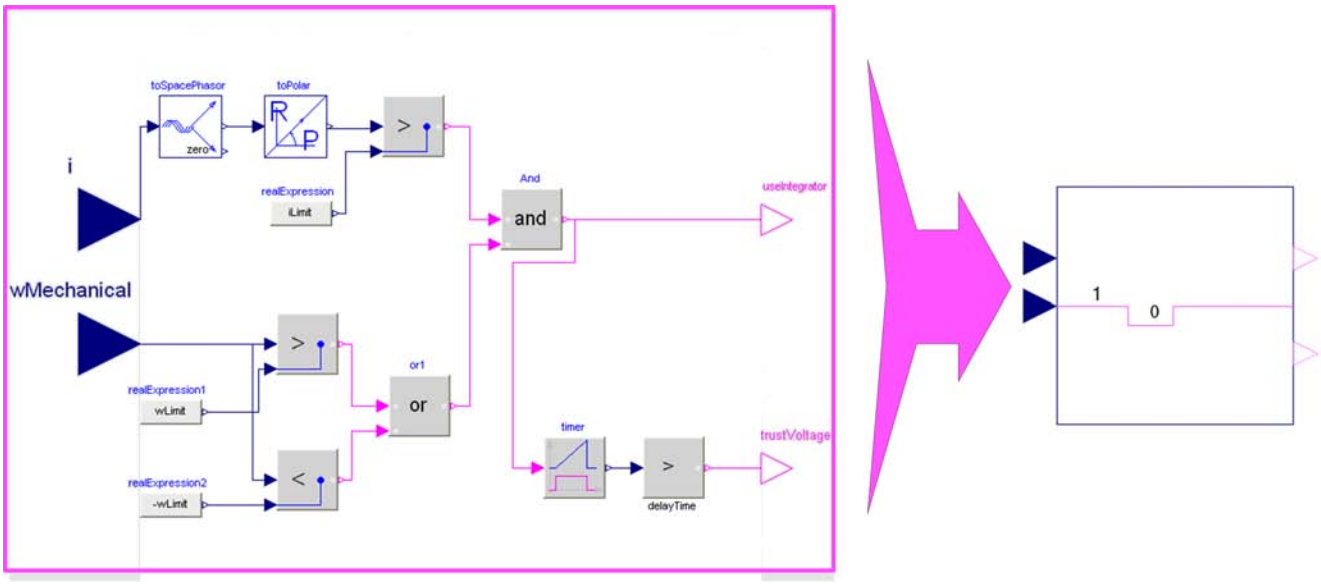

Abb. 4.5: Trust-Voltage-Model

Die Entwicklung des Temperaturmodells wird in der Abbildung 4.6 gezeigt und basiert auf der Temperaturabhängigkeit des Rotorwiderstands R_r, die im Unterabschnitt 2.4.3 mit der Gleichung (2.81) angegeben wurde. Für die Bestimmung des aktuellen Rotorwiderstands R_r werden die Parameter L_r und T_R vom Bus geholt und die Division $R_r = \frac{L_r}{T_R}$ durchgeführt. Ausgehend von der Gleichung (2.81) wird die Rotortemperatur T_r bestimmt und auf den Bus gelegt. Unter Berücksichtigung der Umgebungstemperatur T_{amb}, die als Parameter bekanntgegeben wird, wird die Temperaturänderung des Rotors bestimmt, welche über die Proportionalität k mit der des Statortemperatur verknüpft ist. Dadurch wird die Rotortemperatur T_r zur Bestimmung der Statortemperatur herangezogen.

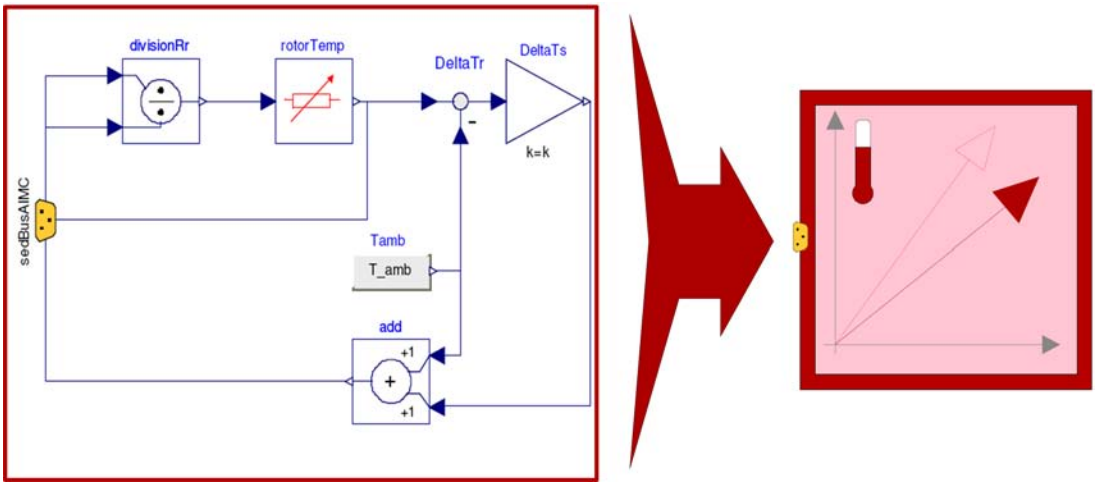

Abb. 4.6: Temperatur-Box

4.4 Zusammenfassung

Die Antriebsstruktur mit der Parameternachführung der Tracking Library (Abbildung 4.1) hat dasselbe Arbeitsprinzip wie der alte Antrieb ohne Nachführung der Parameter, aber unterscheidet sich durch neu entwickelte Modelle, wie *Tracking* und *Temperature* für die Durchführung der Parameternachführung und die erweiterten Flussmodelle *CurrentModel* und *VoltageModel*. *Tracking* ist die Beschreibung der in der Diplomarbeit eingesetzten Parameternachführungmethode, die im Unterabschnitt 2.4.3 erläutert wurde. Das Modell *Temperature* bestimmt die Rotor- bzw. Statortemperatur ausgehend der nachgeführten Parameter. Die alten Flussmodelle wurden so umstrukturiert, dass sie statt konstanten Parametern die nachgeführten Parameter für die Flusserfassung heranziehen. Die Probleme des Spannungsmodells *VoltageModel* bei niedrigen Drehzahlen wurden mit Hilfe eines vom *TrustVoltageModel* gelieferten Booleschen Signal gelöst, das den Integrator im Modell steuert. *VoltageModel* funktioniert somit nur innerhalb des vorgegebenen Limits: *wLimit*, *iLimit*. *TrustVoltageModel* steuert auch das Modell *Tracking*, um das Modell erst nach Entstehung des magnetischen Flusses in der Maschine einzuschalten. Die restlichen Komponenten unterscheiden sich nicht von derjenigen im alten Antrieb. Mit Hilfe der eingeführten Erweiterungen und der neu entwickelten Modelle, entspricht der neu entstandene Antrieb dem erwünschten Modell der feldorientierten Regelung der Asynchronmaschine mit integrierter Nachführung der Parameter.

5 Simulation

Die Simulationen der Modelle wurden mit Dymola *Dynamic Modeling Laboratory* durchgeführt. Dymola ist eine Simulationsumgebung für die Modellierungssprache Modelica. Auf Basis von Modelica, ist Dymola das zur Zeit leistungsfähigste und flexibelste Werkzeug zur Simulation und Optimierung komplexer Systeme. Dymola bietet den Anwendern durch die Simulation ein besseres Bild der Wirklichkeit und einen schnellen und einfachen Modellaufbau. Mit Dymola wurden die Funktionstests der Modelle gemacht und Simulationen für die Modellverifikation anhand von mehreren Beispielen durchgeführt. Im weiteren wird Dymola näher beschrieben und abschliessend werden die Simulationen erläutert.

5.1 Dymola

Modelica ist in der Simulationsumgebung Dymola implementiert. Dymola ist ein komplettes Tool für die Modellierung und Simulation von komplexen physikalischen Systemen. Mit "offener" Modellierungssprache Modelica ermöglicht Dymola die Erweiterung der vorhandenen oder die Entwicklung von neuen Modellen bzw. Bibliotheken. Das macht es besonders attraktiv für die Modellierung und Simulation neuer Technologien und spezieller Anforderungen. Die multi-engineering Eigenschaft erleichtert die Analyse des dynamischen Verhaltens und Lösungen mehrerer komplexe Systeme, wie elektrische, mechanische, thermodynamische etc. Diese werden in Dymola hierarchisch gegliedert [10].

Dymola hat einen Editor für die grafische Zusammensetzung von Modellen. Dabei vereinfacht es die Modellbildung mit Hilfe des Einbindens von Bibliotheken und den Aufbau anhand grafischer Modelle. Zudem vereinfacht und reduziert es die Gleichungssysteme, so dass eine schnellere Simulationsgeschwindigkeit erreicht werden kann. Es entsteht dadurch eine graphisch-textuelle Entwicklungsumgebung für Modelica Modelle. Das Modellierungsfenster ermöglicht die Modellerstellung inklusive Quelltexteingabe. Eine übersichtliche Modellentwicklung wird durch verschiedene Sichten auf das Modell (Quelltexte, Icons, Gleichungen) ermöglicht [1]. Die Abbildung 5.1 stellt ein typisches Dymola Modellierungsfenster dar.

Nach erfolgreicher Modellierungsphase wird das Modell übersetzt. Bei der Übersetzung werden die mathematischen Gleichungen und algebraischen Schleifen aufgelöst. Das Modell wird in C-Code übersetzt, woraus dann ein lauffähiges Programm erzeugt wird. Dymola hat leistungsfähige numerische Integrationsalgorithmen für die Simulationsphase, die die korrekte und robuste Verarbeitung der Modelle gewährleisten. Die am häufigsten verwendeten sind: DASSL, Euler-,Radau, Lsodar, Runge-Kutta-Verfahren [1]. Das Simulationsfenster ermöglicht die grafische Darstellung der Berechnungsergebnisse mittels konfigurierbarer Diagramme. Die Abbildung 5.2 stellt ein typisches Simulationsfenster in Dymola dar.

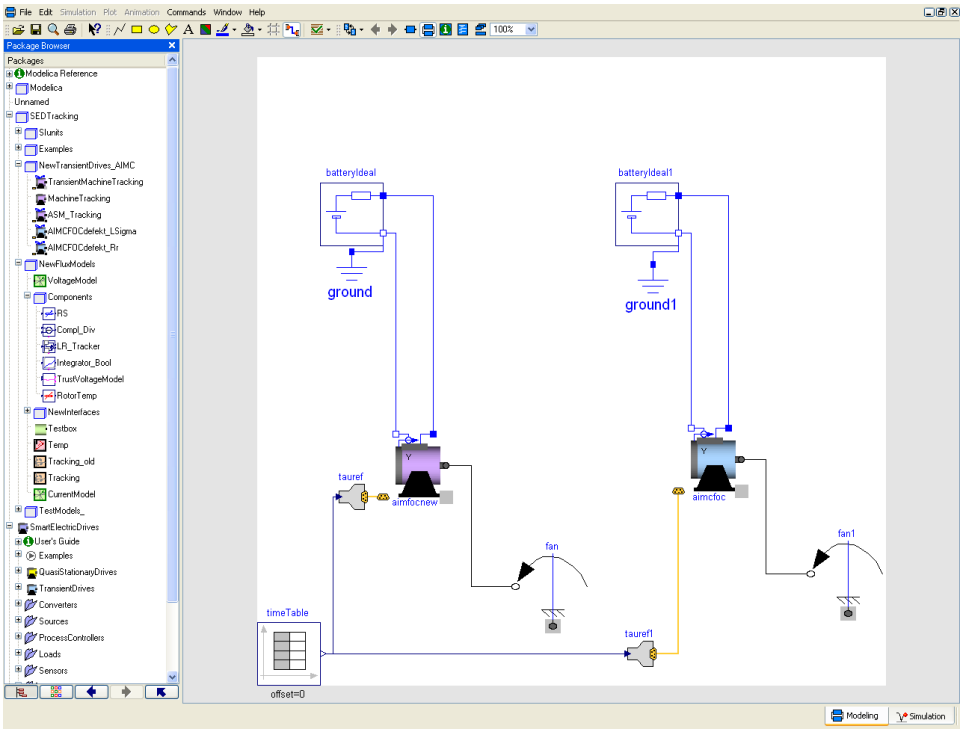

Abb. 5.1: Dymola Modellierungsfenster

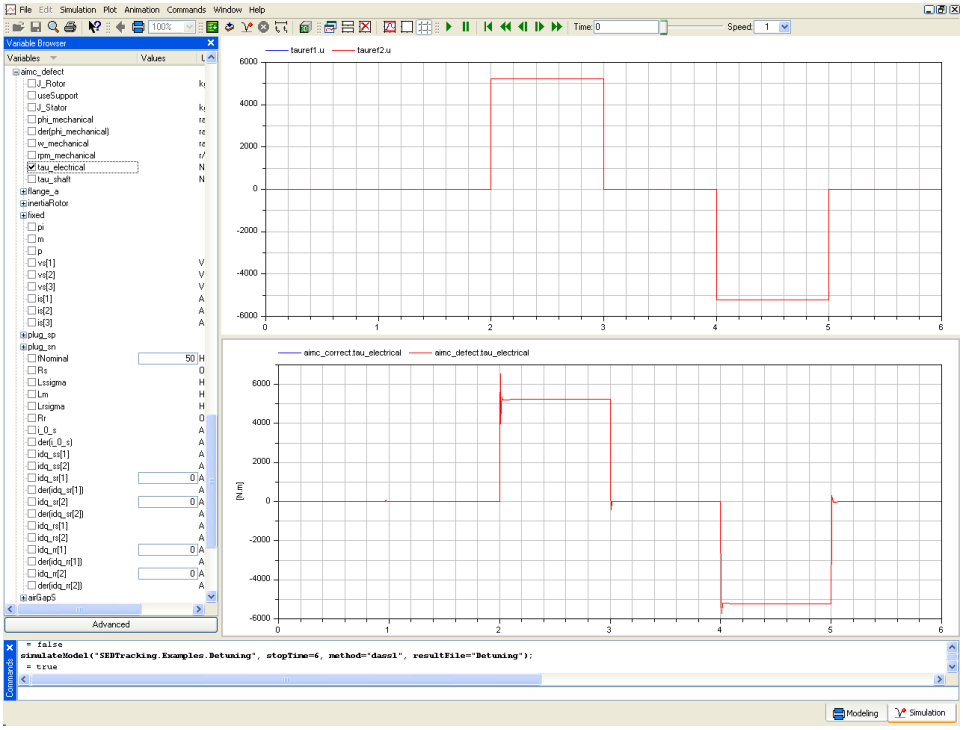

Abb. 5.2: Dymola Simulationsfenster

5.2 Modellverifikation

Die Parameter der Asynchronmaschine können sich während des laufenden Betriebes ändern und falsche Parameter verschlechtern das dynamische Verhalten der feldorientierten Regelung. Das Parametertracking ist mit dem Ziel eingesetzt, dass die verstimmten oder falsch angegebenen Parameter das Systemverhalten nicht verschlechtern, in dem sie auf den richtigen Wert nachgeführt werden. Es wird folglich anhand von Simulationsergebnissen gezeigt:

1. Welche Einflüsse die falschen Parameter auf die Regelung haben (Verstimmung/Detuning)
2. Wie gut die Parameternachführung im ungeregelten Betrieb der Asynchronmaschine funktioniert (Funktionstest im ungeregelten Betrieb)
3. Wie gut die Parameternachführung einer feldorientiert geregelten Asynchronmaschine funktioniert (Funktionstest mit FOC)

Weil die Methode der Parameternachführung mit und ohne feldorientierten Regelung arbeiten kann, wurde sowohl der Antrieb ohne feldorientierte Regelung als auch der Antrieb mit feldorientierten Regelung simuliert.

5.2.1 Verstimmung (Detuning)

Wie bereits erwähnt, wird der Zustand der Maschine nur dann richtig festgelegt, wenn die Modellparameter mit den Maschinenparametern übereinstimmen und damit die notwendigen Komponenten vom Statorstrom, die moment- bzw. flussbildend wirken, richtig entkoppelt werden können. Wenn die Parameter verstimmt sind, bildet sich demzufolge ein falsches Drehmoment in der Maschine. Welche Auswirkungen ein falsch angegebener Parameter hat, wird folglich anhand eines Simulationsbeispiels gezeigt.

Das Beispiel basiert auf dem im Unterabschnitt 3.2.2 erläuterten und in der Abbildung 3.3 dargestellten Antrieb mit der feldorientierten Regelung ohne Parameternachführung (Abbildung 5.3). Es werden dafür zwei Modelle miteinander verglichen. Das erste Modell arbeitet mit den richtigen Parametern. Bei dem zweiten Modell werden die Parameter der Regelung und Flussmodellen mit unterschiedlichen Faktoren multipliziert, um den Effekt der Verstimmung (Detuning) zu zeigen.

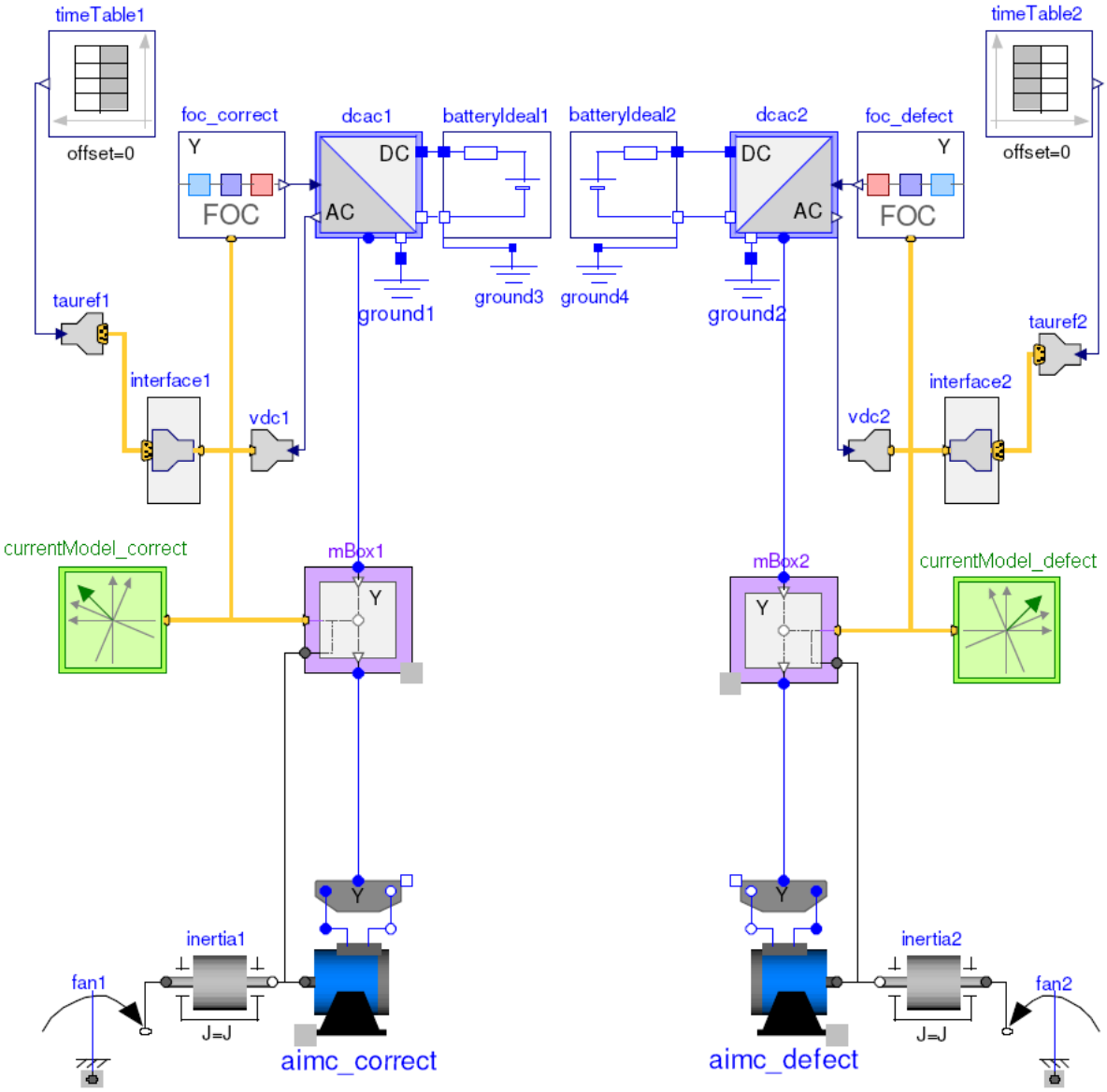

Abb. 5.3: Example Detuning

Erstens wird gezeigt, welches Drehmoment sich in der Maschine bildet, wenn die Parameter nicht verstimmt sind. Also, wenn die beiden Modelle die richtigen Parameter haben. Mit Hilfe der Busverbindungen *tauref* und *tauref1* werden in beiden Maschinen das in der Abbildung 5.4 dargestellte Drehmoment bei konstanter Drehzahl eingespeist. Die Ergebnisse in der Abbildung 5.4 zeigen, dass die Drehmomente beider Maschinen übereinstimmen und sich an das Referenzdrehmomentvelauf anpassen, solange die Maschinenparameter mit den in der Regelung und im Flussmodell propagierten Parametern übereinstimmen.

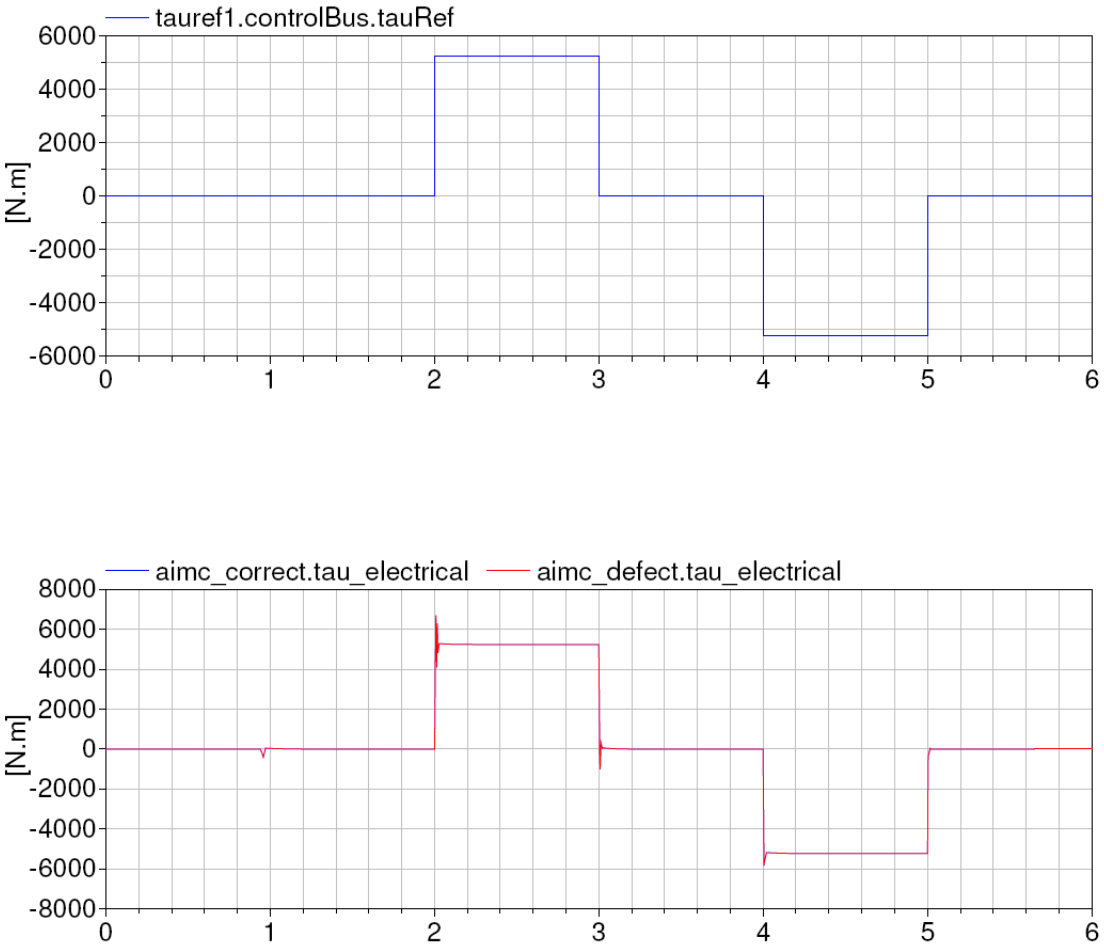

Abb. 5.4: Simulationsergebnis ohne Detuning

Als zweiter Schritt wird die Streuinduktivität L_σ der *aimc_ defect* mit dem Faktor k_1 multipliziert $k_1 \cdot L_\sigma$ und dieser Wert wird im *foc_ defect* und *currentModel_ defect* propagiert. In dem Beispiel wurde für Multiplikationsfaktor $k_1 = 1.1$ gewählt. Das Ergebnis (Abbildung 5.5) zeigt, dass die verstimmte Streuinduktivität kaum eine Änderung des Drehmoments verursacht. Die Abweichung am Beginn des Drehmomentsprungs stört die Regelung nicht und hat nur einen Einfluss auf die Dynamik.

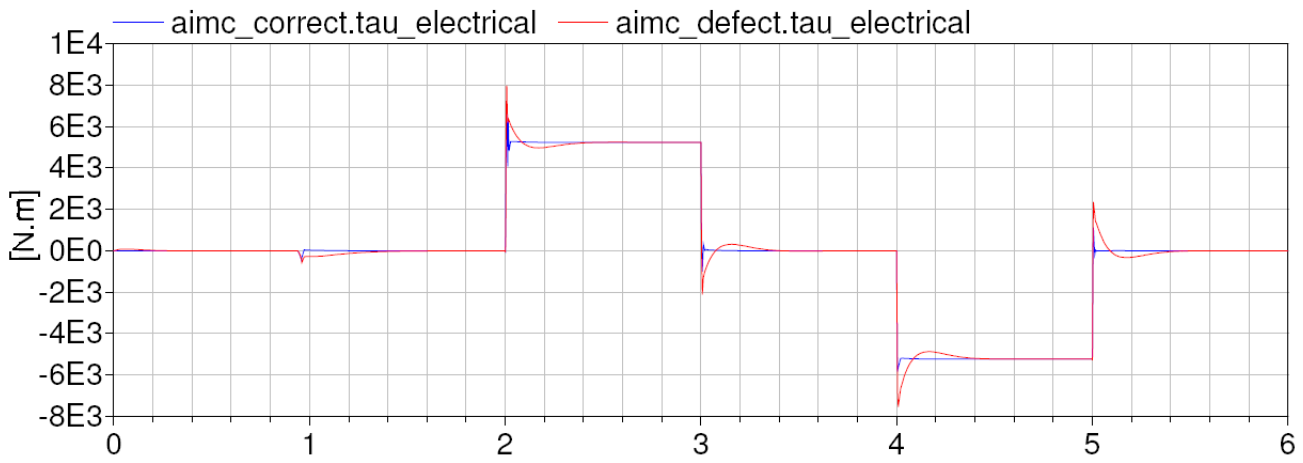

Abb. 5.5: Simulationsergebnis für verstimmte L_σ

Als dritter Schritt wird die Hauptfeldinduktivität L_m mit dem Faktor k_2 multipliziert und $k_2 \cdot L_m$ im *foc_ defect* und *currentModel_ defect* propagiert. Es wurde als Multiplikationsfaktor $k_2 = 1.1$ gewählt. In der Abbildung 5.6 ist ersichtlich, dass eine Änderung der Hauptfeldinduktivität bei der Regelung und beim Flussmodell eine Abweichung des Drehmomentes vom Referenzdrehmoment verursacht.

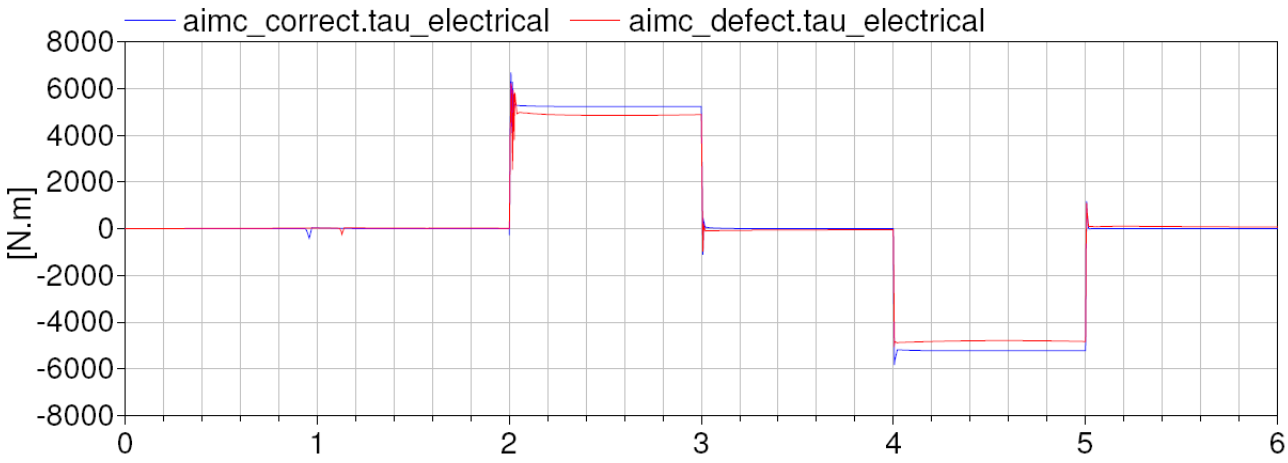

Abb. 5.6: Simulationsergebnis für verstimmte L_m

Als letztes wird zusätzlich zur verstimmten Hauptfeldinduktivität $k_2 \cdot L_m$ auch der Rotorwiderstand R_r mit dem Faktor k_3 multipliziert $k_3 \cdot R_r$ und diese werden im *foc_ defect* und *currentModel_ defect* propagiert. Es wurde als Multiplikationsfaktor wieder $k_3 = 1.1$ gewählt. Eine falsche Angabe für $R_r$wirkt auf dem Schlupf s der Maschine. Das Ergebnis (Abbildung 5.7) zeigt uns, dass die zusätzliche Verstimmung des Rotorwiderstandes R_r eine deutliche Abweichung des Drehmomen-

tes verursacht. Beim Drehmoment Null hat der Term $k_3 \cdot R_r$ keine Auswirkung, weil da der Rotor stromlos ist.

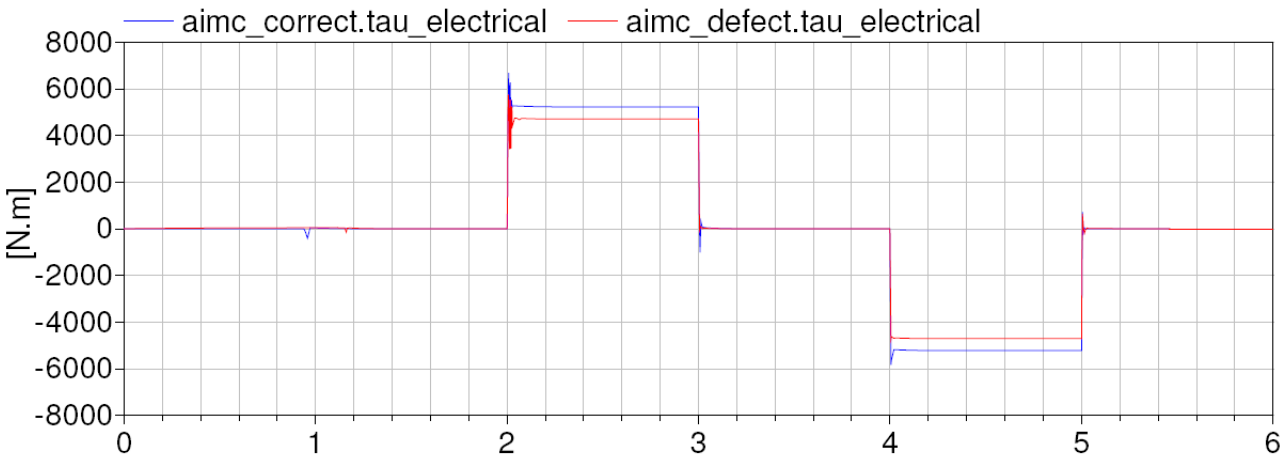

Abb. 5.7: Simulationsergebnis für verstimmte R_r

5.2.2 Funktionstest im ungeregelten Betrieb

Mit dem Ziel die Funktionsfähigkeit der Modelle im ungeregelten Betrieb zu testen, wird folglich ein Simulationsbeispiel präsentiert in dem zwei Asynchronmaschinen ohne feldorientierte Regelung am Netz betrieben werden. Die Abbildung 5.8 stellt das untersuchte Beispiel dar. Bei der Maschine *AIMC* werden die Parameter L_r, T_R vom *TEST-BOX* , die Parameter T_r, T_s vom *TEMP* als fixe Parameter auf dem Bus gelegt. Bei der Maschine *AIMC1* werden die Parameter mit *Tracking* und *Temperature* nachgeführt. In beiden Maschinen stellt sich das Drehmoment anhand eines Lastmomentes ein.

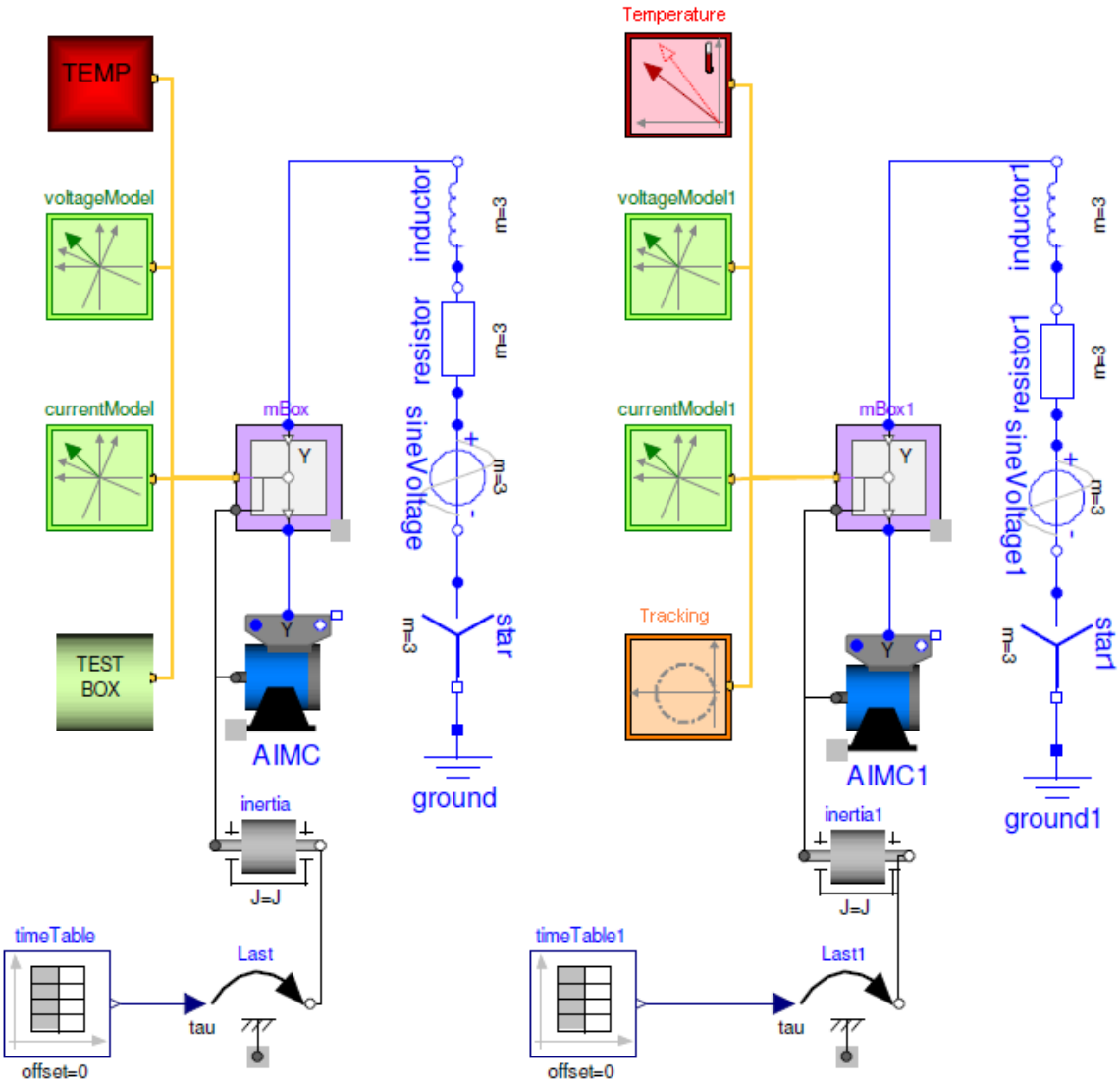

Abb. 5.8: Example Test Netz

Innerhalb 10 Sekunden ist der Einschaltvorgang der Maschine vollständig abgeschlossen. Als Last wird nach 10 Sekunden das negative Nenndrehmoment angelegt, demzufolge bildet sich in beiden Maschinen das positive Nenndrehmoment (Abbildung 5.9). Der Momentverlauf beider Modelle und das dazu gehörige Tracking der Parameter werden nachfolgend dargestellt.

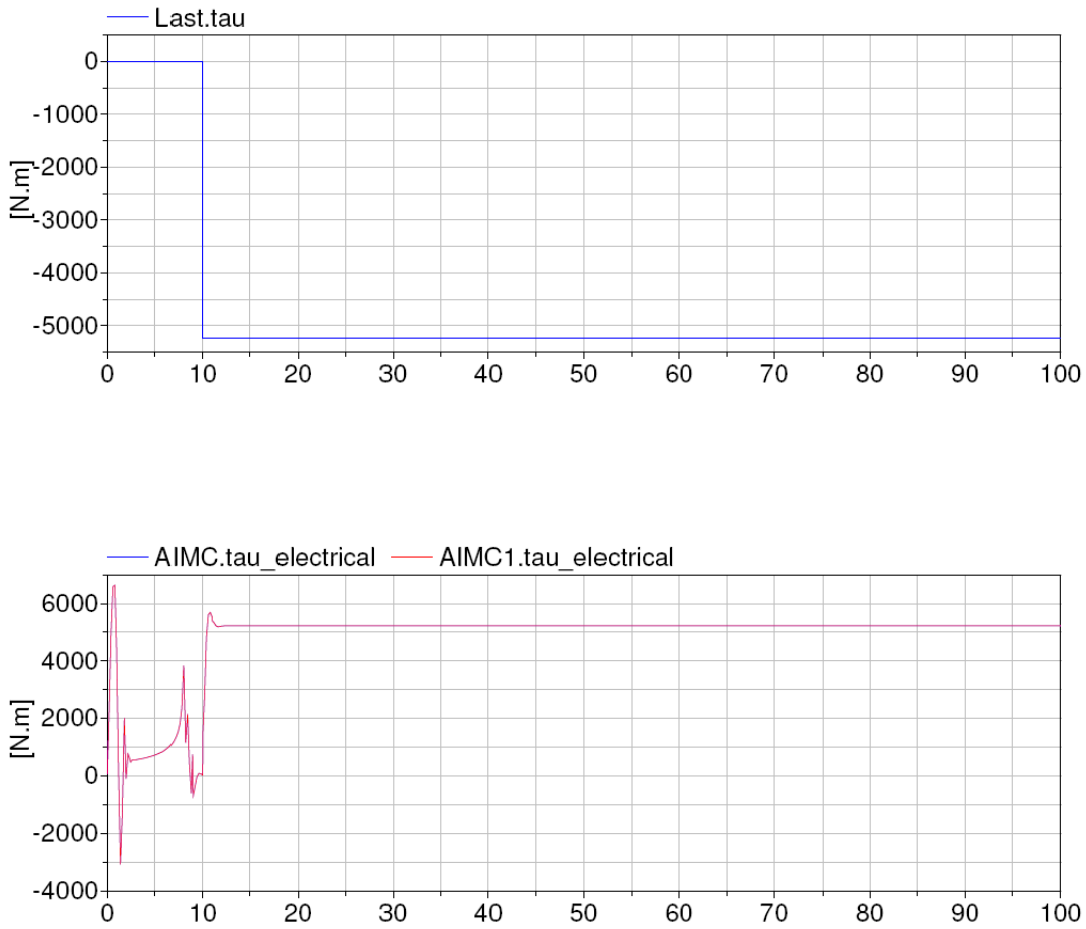

Abb. 5.9: Drehmomentverläufe für den Betrieb am Netz

Die Simulation wird mit falsch angegebenen Startwerten für die Rotorinduktivität L_r und für die Rotorzeitkonstante T_R der *currentModel1* gestartet, um zu testen ob sie mit Hilfe der Nachführung auf die richtigen Werten nachgeführt werden. Beide Maschinenmodelle im Beispiel berücksichtigen keine Temperaturänderungen. Die Rotor- und Statortemperatur der *AIMC* ist im *TEMP* als 393.15 [K] angegeben. Bei der *AIMC1* wird die Temperatur des Rotors in *Temperature* ausgehend von Referenzwerten im warmen Zustand (393.15 [K]) gerechnet und abschliessend mit der Statortemperatur gleichgesetzt $T_r = T_s$ $(k = 1)$. Weil die Temperaturen beider Maschinen in diesem Beispiel gleich angegeben sind (393.15 [K]) und keine Sättigungseffekte berücksichtigt wurden, sollen am Ende der Regelung die nachgeführten Parameter der *AIMC1* mit den fixen Parametern der *AIMC* übereinstimmen.

Die Rotorinduktivität L_r der *currentModel1* (Abbildung 5.10 rote Linie) beginnt mit dem falschen Startwert. L_r schwingt am Beginn aufgrund der Einschaltvorgängen und des vorgegebenen Drehmomentsprungs. Es dauert circa 30 Sekunden bis die Rotorinduktivität vollständig nachgeführt wird. Die Rotorinduktivität L_r passt sich an die Änderungen schneller als die Rotorzeitkonstante T_R an, aus diesem Grund wurde die Reglerzeitkonstante der Rotorinduktivität auf 1 s eingestellt (Abbildung 5.10).

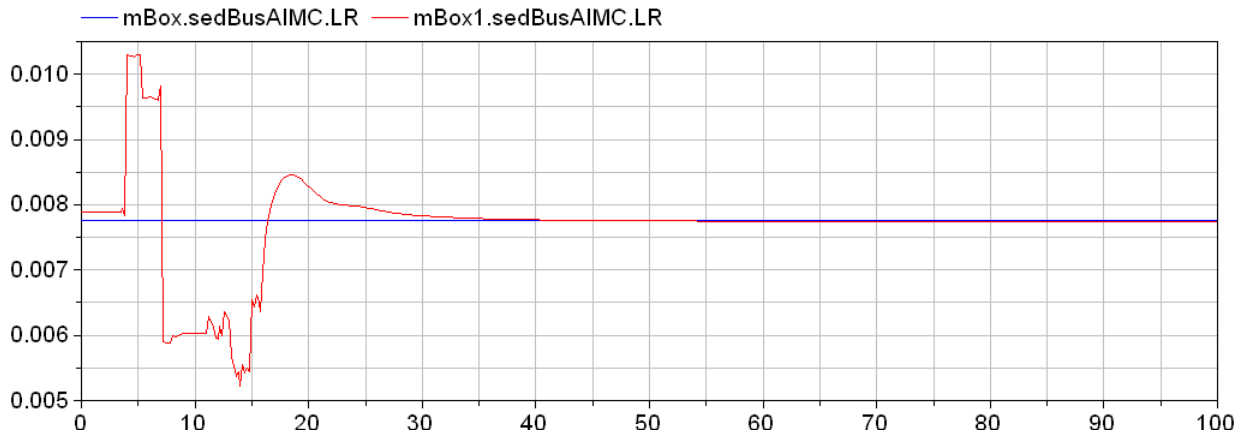

Abb. 5.10: Simulationsergebnis für Lr (Betrieb am Netz)

Die Rotorzeitkonstante T_R der *currentModel1* (Abbildung 5.11 rote Linie) beginnt auch mit einem falsch angegebenen Startwert und innerhalb 40 Sekunden nach dem Drehmomentsprung (t = 50) wird T_R auch auf den richtigen Wert nachgeführt. T_R passt sich dagegen langsamer an die Änderungen an, als die Rotorinduktivität L_r. Die Reglerzeitkonstante der Rotorinduktivität wird auf 10 ms eingestellt, um die Regelung zu beschleunigen. Die Abbildung 5.11 stellt die nachgeführte T_R der *currentModel1* und die konstante T_R der *currentModel* dar.

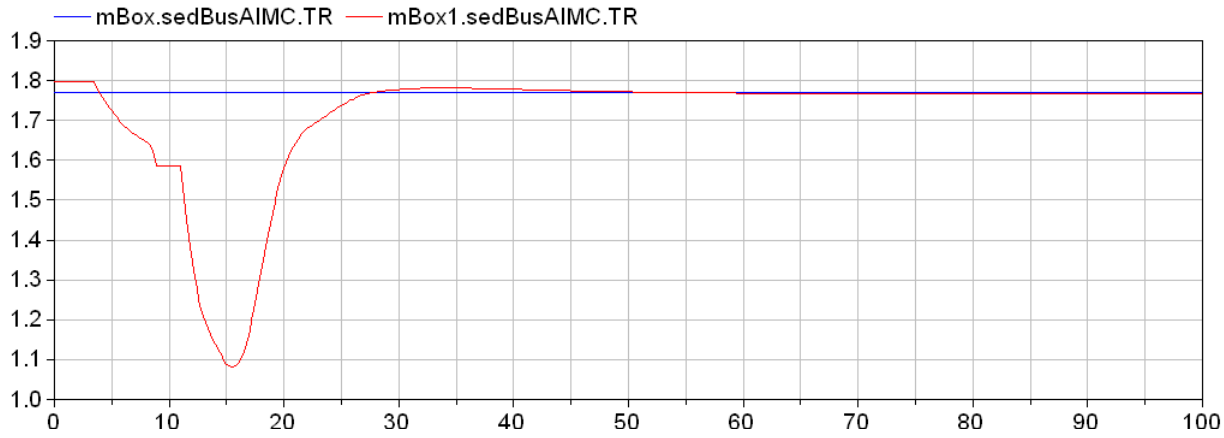

Abb. 5.11: Simulationsergebnis für TR (Betrieb am Netz)

Die Temperaturen des Rotors und des Stators in der *AIMC1* zeigen beim Drehmomentsprung einen leichten Anstieg. Die Temperatur in der Maschine mit dem Parametertracking (*AIMC1*) soll am Ende der Nachführung mit der Temperatur der Maschine ohne Parametertracking (*AIMC*) übereinstimmen, weil das Maschinenmodell *AIMC1* wie das Maschinenmodell *AIMC* im warmen Zustand arbeitet und im Beispiel keine Temperaturänderungen vom Maschinenmodell berücksichtigt werden. Die Abbildung 5.12 zeigt den Verlauf beider Temperaturen. Die Rotortemperatur T_r und die Statortemperatur T_s der *AIMC1* wird im Modell *Temperature* durch $k = 1$ gleich gesetzt (Abbildung 5.12 violette Linie). Die Rotor und Statortemperaturen der *AIMC* wurden als fixe Parameter ent-

sprechend der warmen Zustand der Maschine angegeben (Abbildung 5.12 rote Linie).

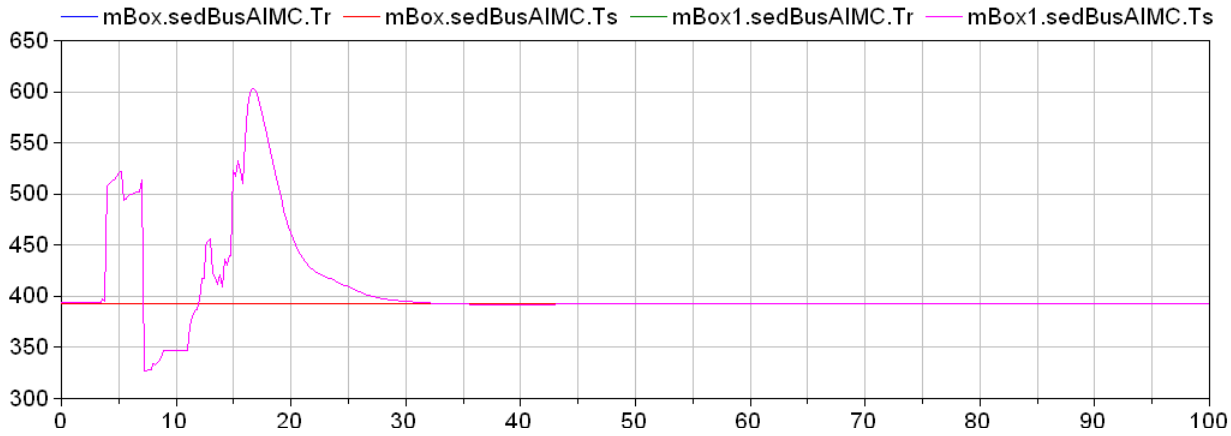

Abb. 5.12: Temperaturverlauf im Rotor und Stator (Betrieb am Netz)

5.2.3 Funktionstest mit FOC

Die Asynchronmaschine mit feldorientierter Regelung wird als letzter Schritt untersucht. In diesem Beispiel werden die zwei in der Diplomarbeit vorgestellten Antriebe getestet (Abbildung 3.3, Abbildung 4.1). Es werden für die Parameter wieder falsche Startwerte angegeben. Daraus folgend wird das Verhalten des Modells aus der Sicht der Parameter betrachtet. An der Maschine wird ein konstantes Drehmoment angelegt und eine konstante Drehzahl wird als Last angeschlossen. Anhand des Simulationsergebnisses wird gezeigt, wie die falsch angegebenen Parameter auf den richtigen Wert geregelt werden.

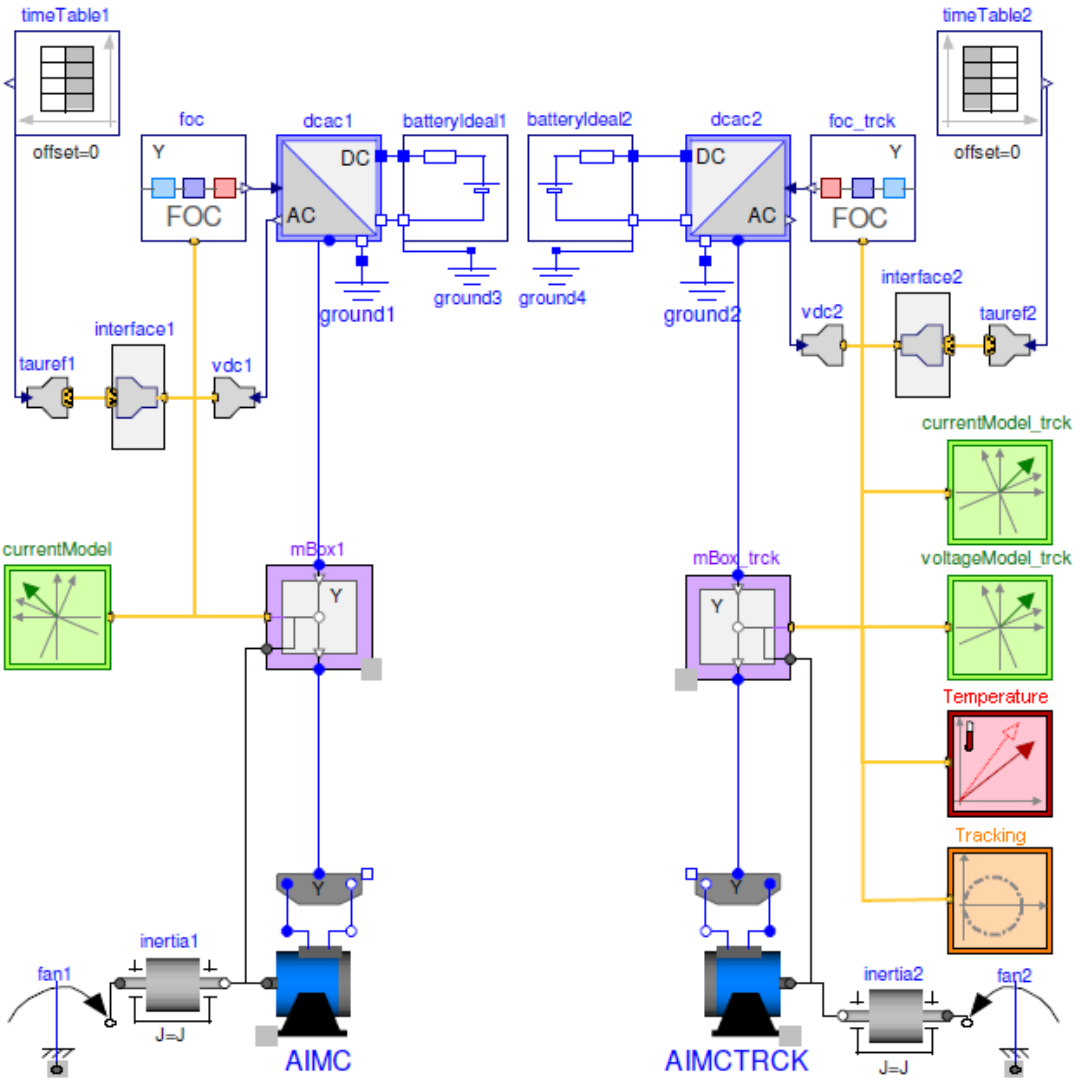

Abb. 5.13: Example Test FOC

Die Maschinen werden mit dem Nenndrehmoment betrieben, das als Referenzdrehmoment auf den Bus gelegt wird. Es bilden sich in den Maschinen die in der Abbildung 5.14 dargestellten Drehmomentverläufe. Die Maschinen sind wie im vorherigen Simulationsbeispiel im warmen Zustand. Mit

dem Ziel noch einmal den einfachsten Fall zu untersuchen werden bei der *AIMCTRCK* keine Temperaturänderungen angegeben und keine Sättigungseffekte werden berücksichtigt. Das hat wieder zur Folge, dass die Parameter der *AIMCTRCK* nach der vollständigen Nachführung mit den Parameter der *AIMC* übereinstimmen sollen. Die Maschine mit Parametertracking *AIMCTRCK* hat aufgrund der Parameter mit falschen Werten am Anfang eine Abweichung vom Referenzdrehmoment (Abbildung 5.14 rote Linie), die sich mit der Nachführung der Parameter auf zugehörige Werte nach 40 Sekunden aufhebt.

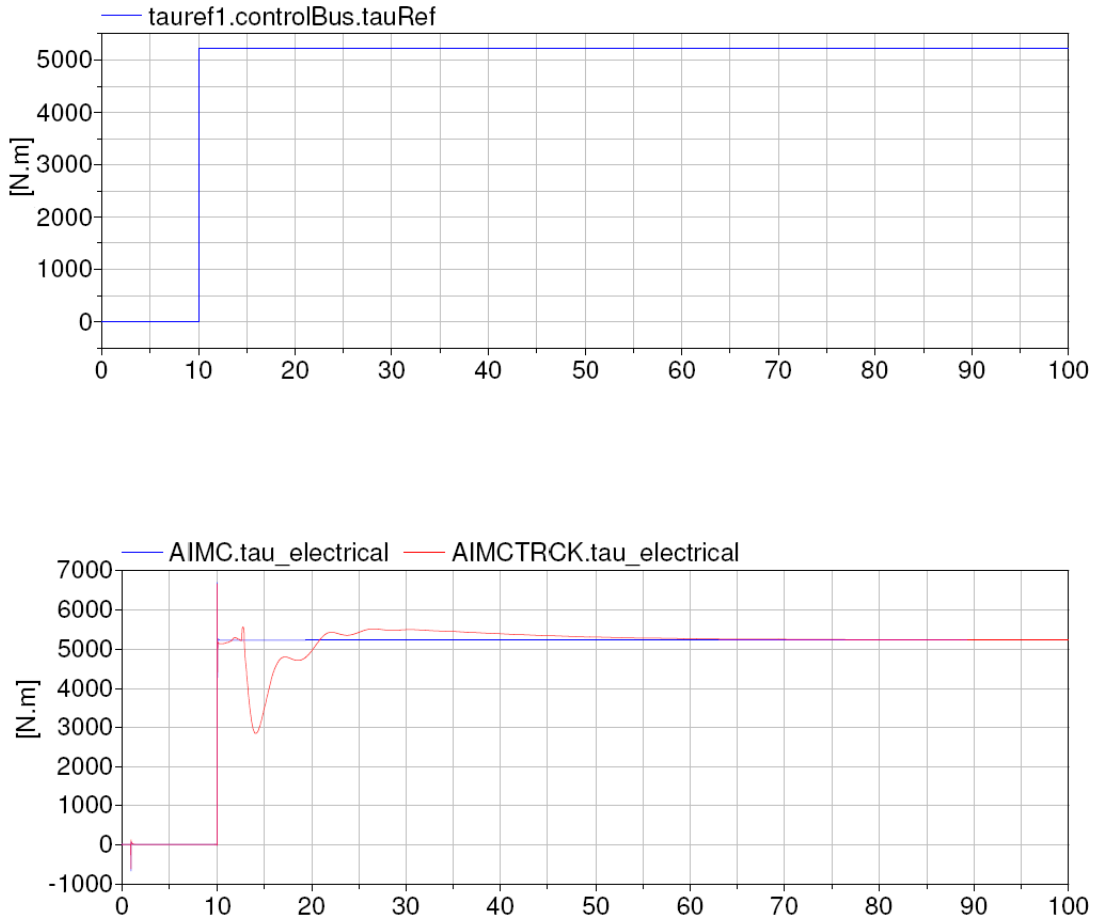

Abb. 5.14: Drehmomentverläufe für den Betrieb mit FOC

Die Rotorinduktivität L_r der *currentModel_ trck* (Abbildung 5.15 rote Linie) wird beginnend mit dem falschen Startwert auf den richtigen Wert nachgeführt. Die Nachführung der Rotorinduktivität fängt nach 10 Sekunden mit dem Drehmomentsprung an und dauert circa 30 bis 40 Sekunden. Der Verlauf der Rotorinduktivität wird in der Abbildung 5.15 dargestellt.

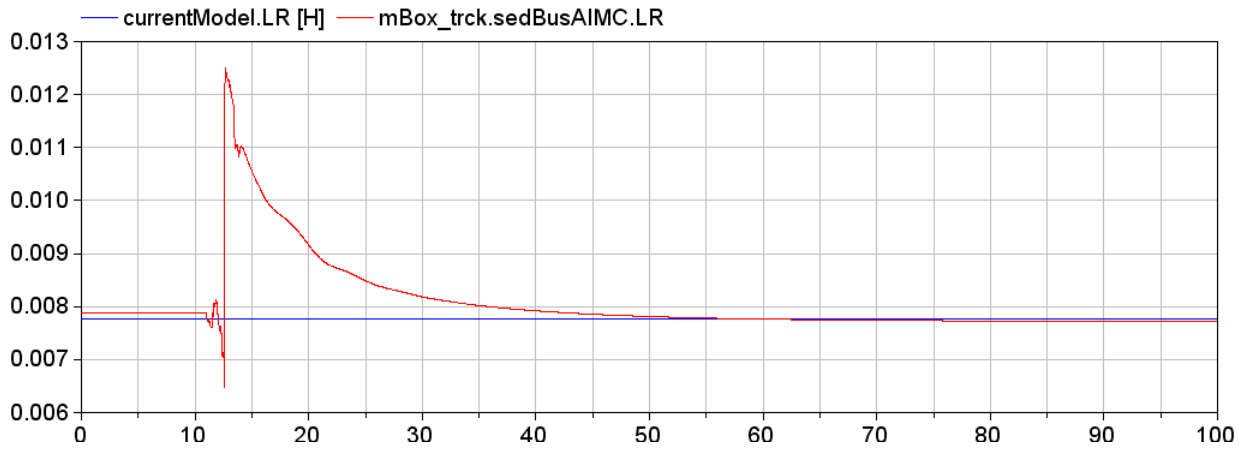

Abb. 5.15: Simulationsergebnis für Lr (Betrieb mit FOC)

Die Rotorzeitkonstante T_R der *currentModel_ trck* wird auch entsprechend der Rotorinduktivität L_r nachgeführt, wobei die Rotorzeitkonstante 60 bis 70 Sekunden für die vollständige Nachführung braucht (Abbildung 5.16).

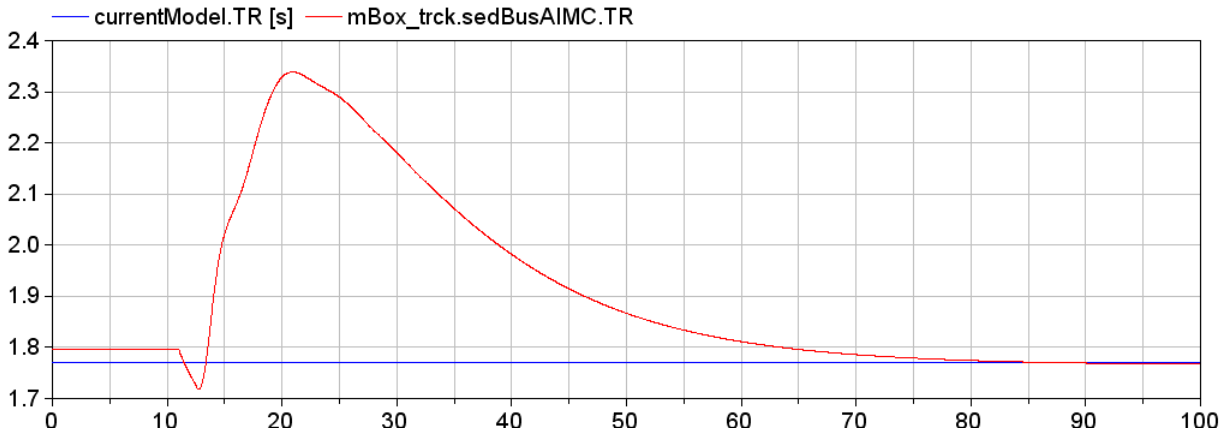

Abb. 5.16: Simulationsergebnis für TR (Betrieb mit FOC)

Der Verlauf der Temperaturen wird in der Abbildung 5.17 angegeben. Die Temperaturen des Stators und des Rotors in der *Temperature* wurden mit $k = 1$ gleich gesetzt und liegen deshalb übereinander. Der Wert, den die nachgeführten Temperaturen nach 40 Sekunden erreichen, entspricht der Temperatur der Maschine im warmen Zustand.

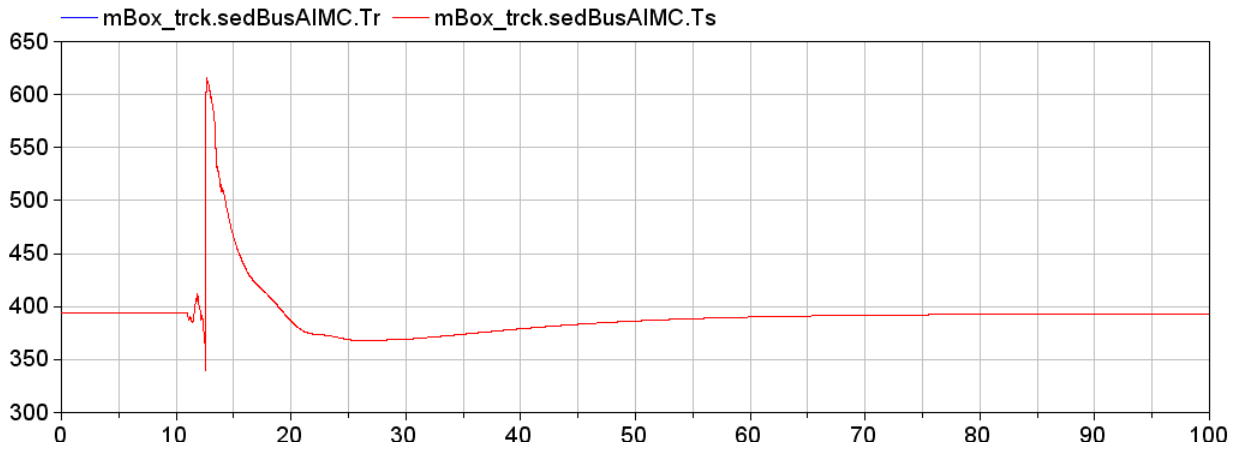

Abb. 5.17: Temperaturverlauf im Rotor und Stator (Betrieb mit FOC)

5.3 Zusammenfassung

Als Simulationsumgebung für die Modellierungssprache Modelica ermöglicht Dymola die Erweiterung der vorhandenen oder die Entwicklung der neuen Modelle bzw. Bibliotheken. Die Modellierung und Simulationen komplexer Systeme werden dank der multi-engineering Eigenschaft und der hierarchischen Gliederung erleichtert. Mit Hilfe der multi-engineering Eigenschaft werden mehrere Systeme, wie elektrische, mechanische, thermodynamische etc. untersucht. Die Dymola-Simulationen werden durchgeführt, um die Funktionsfähigkeit der neuen Modelle zu untersuchen. Die für die Parameternachführung entwickelten und erweiterten Modelle werden in den Beispielen eingesetzt. Es wird anhand von Simulationsbeispielen untersucht:

1. Welche Einflüsse die falschen Parameter auf die Regelung haben (Verstimmung/Detuning)
2. Wie gut die Parameternachführung im ungeregelten Betrieb der Asynchronmaschine funktioniert (Funktionstest im ungeregelten Betrieb)
3. Wie gut die Parameternachführung einer feldorientiert geregelten Asynchronmaschine funktioniert (Funktionstest mit FOC)

Die Simulationsbeispiele sind um den Testvorgang unkompliziert zu halten im einfachsten Fall untersucht, in dem bei der Maschine keine Temperaturänderungen oder Sättigungseffekte berücksichtigt werden. Die Ergebnisse haben gezeigt, dass die Modelle ihre Aufgabe erfolgreich erfüllen, und die Parameter sich am Zustand der Maschine orientieren.

6 Zusammenfassung

Die Drehstrom-Asynchronmaschine mit Käfigläufer ist wegen des fehlenden Kommutators relativ robust im Vergleich zur Gleichstrommaschine und findet deswegen ein verbreitetes Einsatzgebiet. Die Asynchronmaschine wird mit einem Umrichter und geregelt betrieben. Die feldorientierte Regelung ist eine der möglichen Regelungsmethoden. Das Parametertracking der Asynchronmaschine verbessert die Qualität der feldorientierten Regelung. Das dynamische Verhalten der Maschine wird mit Hilfe der aktualisierten Parameter der Maschine verbessert. Es existieren zahlreiche Methoden für die Erfassung bzw. Nachführung der Parameter. Diese werden in der Diplomarbeit kurz erläutert.

Die in der Diplomarbeit untersuchte Methode basiert auf Modellvergleichen während des Betriebes. Die Flussmodelle der Asynchronmaschine werden für die Erfassung des magnetischen Flusses eingesetzt. Abschliessend wird die Abweichung zwischen den Flüssen zur Nachführung der Parameter herangezogen. Mit Hilfe des gemessenen Statorstroms und des berechneten Flusses werden die komplexen Ersatzgrößen $\underline{L}_r$ für beide Flussmodelle ausgerechnet. Ausgehend von der komplexen Ersatzgröße des Strommodells $\underline{L}_r^i$ und des Spannungsmodells $\underline{L}_r^v$ werden die Rotorinduktivität L_r und die Rotorzeitkonstante T_R nachgeführt. Diese beiden Größen werden zur Gewinnung des Rotorwiderstandes R_r herangezogen. Mit Hilfe des festgelegten Rotorwiderstands kann die aktuelle Rotortemperatur T_r und die Statortemperatur T_s festgelegt werden. Die Statortemperatur T_s wird dann zur Bestimmung des Statorwiderstandes R_s herangezogen.

Die Methode des Parametertracking ist in der Modellierungssprache Modelica modelliert. Modelica als objektorientierte Sprache ermöglicht einfache Lösungen für komplexe Systeme durch einen gleichungsbasierten und hierarchischen Modellaufbau. Ausserdem ermöglichen Techniken wie Austausch und Wiederverwendung von Modellen den Aufbau verschachtelter Modelle. Die *Smart Electric Drives* (SED) Library als spezielle Bibliothek für Modellentwicklungen im Bereich elektrischer Antriebe verfügt über alle nötigen Komponenten, die für die Erstellung neuer Modelle benötigt werden. Für die Implementierung der untersuchten Methode wurde ein in der SED Library schon vorhandener Antrieb der Asynchronmaschine mit feldorientierten Regelung entsprechend der Parameternachführung erweitert. Zu diesem Zweck wurden zuerst zwei neue Modelle entwickelt. Das erste Modell; *Tracking* wird für das Tracking der Rotorinduktivität L_r und der Rotorzeitkonstante T_R eingesetzt. Die Aufgabe des zweiten Modells *(Temperature)* ist die Temperaturerfassung der Maschine. Weiters werden die vorhandenen Flussmodelle in der SED Library *(Currentmodel* und *Voltagemodel)* entsprechend der Parameternachführung erweitert, damit sie anstelle von fixen Parametern mit den nachgeführten Parametern arbeiten. Ausserdem werden die Probleme des Spannungsmodells bei niedrigen Drehzahlen mit dem Modell *TrustVoltageModel* gelöst, welches ein Boolesches-Signal als Steuersignal liefert und somit das Spannungsmodell ein- bzw. ausschaltet. Das ermöglicht die für das Tracking benötigte parallele Arbeit beider Flussmodelle.

Der neu entstandene Antrieb wird anhand von Simulationsbeispielen mit dem Simulationstool Dymola getestet. Dymola ermöglicht auf Basis von Modelica die Modellierung und Simulation komplexer Systeme. Die Simulationsbeispiele zeigen, welche Einflüsse die Maschinenparameter auf die Regelung haben wenn falsch angegebene Parameter bei der Regelung propagiert werden. Mit dem

Ziel, das Verhalten der neuen Modelle nachzuvollziehen, sind Tests durchgeführt worden. Zuerst wurde der Antrieb mit Parametertracking ohne feldorientierte Regelung direkt am Netz getestet. Dafür wird das Modell einmal mit den entwickelten Tracking-Modellen betrieben und einmal mit Hilfskomponenten eingespeist, die anstatt der nachgeführten die fixen Parameter im warmen Zustand der Maschine liefern. Als letztes wird der neue Antrieb der Tracking Library mit dem alten Antrieb der SED Library verglichen. Die Simulationsergebnisse haben eine richtige Nachführung der Parameter bestätigt, und diese werden im Abschnitt 5.2 angegeben. Die detaillierte Darstellung der Ergebnisse zeigt, dass im Falle einer falschen Angabe für die Parameterwerte, diese mit Hilfe des Tracking auf den richtigen Wert nachgeführt werden.

Abschließend lässt sich zusammenfassen: Es wird der Einfluss der Parameternachführung für die feldorientiert geregelte Asynchronmaschine untersucht. Die untersuchte Methode ist zuerst theoretisch, und dann als Modelica Modell vorgestellt. Die notwendigen Modellerweiterungen bzw. die Entwicklung neuer Modelle sind detailliert angegeben. Anhand von Simulationsbeispielen wird die Funktionsfähigkeit der entwickelten Modelle untersucht. Somit wird gezeigt, dass sich die Parameter mit einer erfolgreichen Nachführung am Zustand der Maschine orientieren und zur Verbesserung der feldorientierten Regelung beitragen.

Literaturverzeichnis

[1] AR. *Dymola Multi-Engineering Modeling and Simulation Smart Electric Drives Library.* Dynasim, Wien, 2005-2008.

[2] M. Bertoluzzo, G. Buja, and R. Menis. Self-commissioning of IM drives: One-test identification of the magnetization characteristic of the motor. *Conference Proceedings of the IEEE Symposium on Electrical Machines, Power Electronics and Drives, SDEMPED*, pages 509–514, 1999.

[3] A. Buente, H. Grotstollen, and P. Krafka. Field weakening of induction motors in a very wide region with regard to parameter uncertainties. *Power Electronics Specialists Conference*, 1:944–950, 1996.

[4] G. Pascoli C. Kral, F. Pirker and H. Kapeller. Robust Rotor Fault Detection by Means of the Vienna Monitoring Method and a Parameter Tracking Technique. *IEEE Trans. Ind. Applicat.*, 55, 2008.

[5] P. J. Chrzan, G. Champenois, P. Coirault, and J. C. Trigeassou. Diagnosis of induction machine in stillstand conditions. *Conference Proceedings of the International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives, SDEMPED*, pages 497–501, 1999.

[6] P. J. Chrzan and R. Szczesny. Fault diagnosis of voltage-fed inverter for induction motor drive. *Proceedings of the IEEE International Symposium on Industrial Electronics, ISIE*, pages 1011–1016, 1996.

[7] Patentauftrag A1903/2001 C.Kral. Verfahren zur Bestimmung und Nachfuehrung der Rotorreaktanz und der Rotorzeitkonstante bei Asynchronmaschinen durch Modellrechnungen.

[8] F. Filippetti, G. Franceschini, C. Tassoni, and P. Vas. A fuzzy logic approach to on-line induction motor diagnostics based on stator current monitoring. *Conference Proceedings of the Stockholm Power Tech*, pages 156–161, 1995.

[9] Rolf Fischer. *Elektrische Maschinen.* Carl Hanser-Verlag, München/Wien, 1989.

[10] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1.* IEEE Press, Piscataway, NJ, 2004.

[11] M. Globevnik. Induction motor parameters measurement at stand still. *Proceeding of the 24th Annual Conference of the IEEE Indutrial Electronic Society. IECON '98*, 1:280–285, September 1998.

[12] J. V. Gragger, H. Giuliani, C. Kral, T. Baeuml, H. Kapeller, and F. Pirker. The SmartElectricDrives Library - powerful models for fast simulations of electric drives. *International Modelica Conference 5th Vienna Austria*, pages 571–577, 2006.

[13] J. V. Gragger, D. Simic, C. Kral, H. Giuliani, V. Conte, and F. Pirker. A simulation tool for electric auxiliary drives in HEVs - the SmartElectricDrives library. *World Automotive Congress, FISITA '06, Yokohama, Japan*, 2006.

[14] J. Holtz and T. Thimm. dentification of the machine parameters in a vector-controlled induction motor drive. *IEEE Transactions on Industry Applications*, 27(6):1111–1118, Nov.-Dec. 1991.

[15] T. Kataoka, S. Toda, and Y. Sato. On-line estimation of induction motor parameters by extended kalman filter. *Fifth European Conference on Power Electronic and Application*, pages 325–329, September 1993.

[16] Jana Kertzscher. *Ein Verfahren zur Identifikation der elektrischen Parameter von Asynchronmaschinen.* Tenea Verlag, Berlin, 2003.

[17] A. Khambadkone and J. Holtz. Vector controlled induction motor drive with a self-commissioning scheme. *16th Annual Conference of IEEE Indutrial Electronic Society. IECON '90*, pages 927–932, November 1990.

[18] N. R. Klaes. Parameter identification of an induction machine with regard to dependencies on saturation. *IEEE Transactions on Industry Applications*, 29(6):1135–1140, Nov.-Dec. 1993.

[19] H. Kleinrath. Ersatzschaltbilder fuer Transformatoren und Asynchronmaschinen. *e&i*, 110:68–74, 1993.

[20] C. Kral. Modellbildung und Simulation des Betriebsverhaltens einer umrichtergespeisten Asynchronmaschine mit defektem Rotorstab. Master's thesis, Technische Universitaet Wien, Vienna, January 1997.

[21] W. H. Kwon, C. H. Lee, K. S. Youn, and G. H. Cho. Measurement of rotor time constant taking into account magnetizing flux in the induction motor. *Conference Record of the 1994 IEEE Industry Applications Society Annual Meeting, 1994.*, pages 88–92, 1994.

[22] E. Levi and S. N. Vukosavic. Identification of the magnetising curve during commissioning of a rotor flux oriented induction machine. *IEE Proceedings - Electric Power Applications*, 146, 6:685–693, 1999.

[23] S. I. Moon and A. Keyhani. Estimation of induction machine parameters from standstill time-domain data. *IEEE Trans. Ind. Applicat.*, 30, 1994.

[24] M. Raina and H. A. Toliyat. Parameter estimation of induction motors - a review and status report. *2001. IECON '01. The 27th Annual Conference of the IEEE Industrial Electronics Society*, 2:1327–1332, 2001.

[25] M. Ruff and H. Grotstollen. Identification of the saturated mutual inductance of asynchronous motor at standstill by recursive least squares algorithm. *European Conference on Power Electronics and Applications, 5th*, pages 103–108, 1993.

[26] D. Schroeder. *Elektrische Antriebe – Regelung von Antriebssystemen.* Springer, Berlin-Heidelberg, 2 edition, 2001.

[27] Manfred Schroedl. *Drehstromantriebe mit Mikrorechnern.* Technische Universität Wien, Wien, 1988.

[28] D. Simic, H. Giuliani, C. Kral, and F. Pirker. Simulation of conventional and hybrid vehicle including auxiliaries with respect to fuel consumption and exhaust emissions. *SAE World Congress 2006*, April 2006.

[29] D. Simic, C. Kral, and H. Lacher. Optimization of a cooling circuit with a parameterized water pump model. *International Modelica Conference, 5th, Vienna, Austria*, 2006.

[30] H. Sugimoto and S. Tamai. Secondary resistance identification of an induction-motor applied model reference adaptiev system and its characteristics. *IEEE Transactions on Industry Applications*, IA-23(2):296–303, March/April 1987.

[31] M. Sumner and G. M. Asher. Self-commissioning for voltage-referenced voltage-fed vector controlled induction motor drives. *0-7803-0695-3 / IEEE*, pages 139–144, 1992.

[32] H. A. Toliyat, E. Levi, and M. Raina. A review of rfo induction motor parameter estimation techniques. *IEEE Transactions on Energy Conversion*, 18(2):271–283, June 2003.

[33] Peter Vas. *Parameter Estimation, Condition Monitoring, and Diagnosis of Electrical Machines.* Oxford University Press Inc., New York, 1979.

[34] J. Weidauer. Verfahren zur Identifikation von Parametern einer Asynchronmaschine. *Patent, DE 41 10 716 A1*, 1992.

[35] M. Wlas, Z. Krzeminski, and H. A. Toliyat. Neural-network-based parameter estimations of induction motors. *IEEE Transactions on Industrial Electronics*, 55(4):1783–1794, April 2008.

[36] L. C. Zai, C. L. DeMarco, and T.A. Lipo. An extended Kalman filter approach to rotor time constant measurement in PWM induction motor drives. *IEEE Trans. Ind. Applicat.*, 28, 1992.

Abbildungsverzeichnis

Tabellenverzeichnis

A Anhang

A.1 Programmlisting Tracking Library

Der Modelica-Quellcode für das Strommodell *CurrentModel* und für das Spannungsmodell *Voltage-Model* der Tracking Library wird folglich angegeben:

```
model CurrentModel
"Calculates the rotor flux space phasor related to the stator
fixed reference frame with updated parameter"
extends SmartElectricDrives.Icons.FluxModel;

// parameters of asm
parameter Integer p(min=1) "Number of pole pairs";

protected
Modelica.Blocks.Math.Gain gain(finalk=p);
Modelica.Electrical.Machines.SpacePhasors.Blocks.Rotator rotator;
Modelica.Electrical.Machines.SpacePhasors.Blocks.
ToSpacePhasor toSpacePhasor;
Modelica.Blocks.Math.Feedback feedbackD;
Modelica.Blocks.Math.Feedback feedbackQ;
Modelica.Blocks.Continuous.Integrator integratorQ(finalk=1;
SmartElectricDrives.AuxiliaryComponents.Transformations.
DerToPolar derToPolar;
Modelica.Blocks.Continuous.Der Der1;
Modelica.Blocks.Math.Add add2;
Modelica.Blocks.Math.Add add1;
public
Modelica.Blocks.Math.Product productD;
public
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.SEDBusAIMC sedBusAIMC1;
Modelica.Blocks.Math.Product productQ;
protected
Modelica.Blocks.Continuous.Integrator integratorD(final k=1);
public
Modelica.Blocks.Math.Division divisionD;
Modelica.Blocks.Math.Division divisionQ;
public
Modelica.Blocks.Math.Division Rr;

equation
connect(toSpacePhasor.y, rotator.u);
connect(gain.y, rotator.angle);
connect(integratorQ.y, derToPolar.u[2]);
connect(gain.y, Der1.u);
connect(Der1.y, add2.u2);
connect(derToPolar.y[3], add2.u1);
connect(derToPolar.y[2], add1.u1);
```

```
connect(gain.y, add1.u2);
connect(gain.u, sedBusAIMC1.phiMechanical);
connect(toSpacePhasor.u, sedBusAIMC1.i);
connect(rotator.y[1], productD.u1);
connect(productQ.u1, rotator.y[2]);
connect(integratorD.y, derToPolar.u[1]);
connect(derToPolar.y[1], sedBusAIMC1.psi);
connect(add1.y, sedBusAIMC1.gammaPsi);
connect(add2.y, sedBusAIMC1.omegaPsi);
connect(feedbackD.y, integratorD.u);
connect(divisionD.u1, integratorD.y);
connect(Rr.u1, sedBusAIMC1.LR);
connect(Rr.u2, sedBusAIMC1.TR);
connect(integratorQ.y, divisionQ.u1);
connect(divisionQ.u2, sedBusAIMC1.TR);
connect(divisionQ.y, feedbackQ.u2);
connect(feedbackQ.y, integratorQ.u);
connect(productD.u2, Rr.y);
connect(feedbackD.u2, divisionD.y);
connect(divisionD.u2, sedBusAIMC1.TR);
connect(Rr.y, productQ.u2);
connect(productD.y, feedbackD.u1);
connect(feedbackQ.u1, productQ.y);

end CurrentModel;



model VoltageModel
"Calculates the rotor flux space phasor related to the stator
fixed reference frame with updated parameter"

extends SmartElectricDrives.Icons.FluxModel;
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.SEDBusAIMC  sedBusAIMC
"Bus for asynchronous induction machines with squirrel cage rotor";
parameter Integer p(min=1) "Number of pole pairs";
constant Integer m=3 "Number of phases";
parameter Modelica.SIunits.Frequency fNominal;
parameter Modelica.SIunits.Inductance Lssigma(
final min=Modelica.Constants.small) "Stator stray inductance";
parameter Modelica.SIunits.Inductance Lm(
final min=Modelica.Constants.small) "Main field inductance";
parameter Modelica.SIunits.Inductance Lrsigma(
final min=Modelica.Constants.small)
"Rotor stray inductance referred to stator";
parameter Real kFeedback(min=0)=0.01
"Gain of the integrator feedback for the output
stabilisation at low input offset";
protected
```

```
parameter Real sigma=1-Lm^2/((Lssigma+Lm)*(Lrsigma+Lm))
"Stray coefficient";
parameter Modelica.SIunits.Inductance LS=(Lm+Lssigma)
"Stator inductance per phase";
//-------------------------------------------------------------------------
// parameters of trust voltage
public
parameter Modelica.SIunits.Time delayTime
"Comparison with respect to useIntegrator Signal";
parameter Modelica.SIunits.Current iLimit;
parameter Modelica.SIunits.AngularVelocity wLimit;
//-------------------------------------------------------------------------
// parameters of integrator
public
parameter Real y_start;
//-------------------------------------------------------------------------
//parameters of Rs
public
parameter SEDTracking.SIunits.LinearTemperatureCoefficient alfa20_s;
parameter Modelica.SIunits.Resistance Rs_ref;
parameter Modelica.SIunits.Temp_K Ts_ref;
protected
Modelica.Electrical.Machines.SpacePhasors.Blocks.
ToSpacePhasor toSpacePhasorV;
Modelica.Electrical.Machines.SpacePhasors.Blocks.
ToSpacePhasor toSpacePhasorI;
Modelica.Blocks.Math.Feedback feedbackB;
Modelica.Blocks.Math.Feedback feedbackA;
Modelica.Blocks.Math.Feedback feedbackA2;
Modelica.Blocks.Math.Feedback feedbackB2;
protected
Modelica.Blocks.Math.Gain gainLssigmaA(final k=LS*sigma);
Modelica.Blocks.Math.Gain gainLssigmaB(final k=LS*sigma);
SmartElectricDrives.AuxiliaryComponents.Transformations.
DerToPolar derToPolar;
Modelica.Blocks.Math.Feedback feedbackA1;
Modelica.Blocks.Math.Feedback feedbackB1;
Components.Integrator_Bool integratorBoolA(
final y_start=y_start, k=1);
Components.Integrator_Bool integratorBoolB(
final y_start=y_start, k=1);
Modelica.Blocks.Math.Gain gainFeedbackA(final k=kFeedback);
Modelica.Blocks.Math.Gain gainFeedbackB(final k=kFeedback);
public
SEDTracking.NewFluxModels.Components.RS stator_R(
final alfa20_s=alfa20_s,
final Rs_ref=Rs_ref,
final Ts_ref=Ts_ref) ;
```

```
Modelica.Blocks.Math.Product product_Re;
Modelica.Blocks.Math.Product product_Im;
SEDTracking.NewFluxModels.Components.TrustVoltageModel
trustVoltageModel( final fNominal=fNominal,
final p=p,
final m=m,
final wLimit=wLimit,
final iLimit=iLimit,
final delayTime=delayTime);

equation
connect(gainLssigmaA.u, toSpacePhasorI.y[1]);
connect(gainLssigmaB.u, toSpacePhasorI.y[2]);
connect(gainLssigmaA.y, feedbackA2.u2);
connect(gainLssigmaB.y, feedbackB2.u2);
connect(toSpacePhasorV.y[1],feedbackA. u1);
connect(toSpacePhasorV.y[2],feedbackB. u1);
connect(sedBusAIMC.v,toSpacePhasorV. u);
connect(sedBusAIMC.i,toSpacePhasorI. u);
connect(feedbackA2.y, derToPolar.u[1]);
connect(feedbackB2.y, derToPolar.u[2]);
connect(feedbackA.y, feedbackA1.u1);
connect(feedbackB.y, feedbackB1.u1);
connect(feedbackA1.y, integratorBoolA.u);
connect(integratorBoolA.y, feedbackA2.u1);
connect(feedbackB1.y, integratorBoolB.u);
connect(integratorBoolB.y, feedbackB2.u1);
connect(integratorBoolA.y, gainFeedbackA.u);
connect(gainFeedbackB.u, integratorBoolB.y);
connect(gainFeedbackB.y, feedbackB1.u2);
connect(stator_R.RS, product_Re.u1);
connect(toSpacePhasorI.y[1], product_Re.u2);
connect(toSpacePhasorI.y[2], product_Im.u2);
connect(feedbackA.u2, product_Re.y);
connect(product_Im.y, feedbackB.u2);
connect(derToPolar.y[1], sedBusAIMC.psi_r);
connect(derToPolar.y[2], sedBusAIMC.gammaPsi_r);
connect(derToPolar.y[3], sedBusAIMC.omegaPsi_r);
connect(trustVoltageModel.wMechanical, sedBusAIMC.wMechanical);
connect(stator_R.Ts, sedBusAIMC.Ts);
connect(stator_R.RS, product_Im.u1);
connect(trustVoltageModel.i, sedBusAIMC.i);
connect(integratorBoolA.enable, trustVoltageModel.useIntegrator);
connect(integratorBoolB.enable, trustVoltageModel.useIntegrator);
connect(gainFeedbackA.y, feedbackA1.u2);

end VoltageModel;
```

Das Modell *Tracking* in der Tracking Library hat folgenden Modelica -Quellcode:

```
model Tracking
"Calculates the rotor inductance and rotor time constant"

// parameters of asm

parameter Integer m;
parameter Integer p;
constant Real pi=Modelica.Constants.pi;
parameter Modelica.SIunits.Voltage VNominal;
parameter Modelica.SIunits.Current INominal;
parameter Modelica.SIunits.Frequency fNominal;
parameter Modelica.SIunits.Inertia Jr;
parameter Modelica.SIunits.Torque tauNominal;
parameter Modelica.SIunits.Resistance Rs;
parameter Modelica.SIunits.Inductance Lssigma;
parameter Modelica.SIunits.Inductance Lm;
parameter Modelica.SIunits.Resistance Rr;
parameter Modelica.SIunits.Inductance Lrsigma;
//-----------------------------------------------------------------------
// parameters of integrator
parameter Real k1 "LR Controller gain factor";
parameter Real k2 "TR Controller gain factor";
parameter Modelica.SIunits.Inductance LRstart
"Starting value of rotor inductance";
parameter Modelica.SIunits.Time TRstart  "
Starting value of rotor time constant";
//-----------------------------------------------------------------------
//parameters of trust voltage
parameter Modelica.SIunits.AngularVelocity wLimit;
parameter Modelica.SIunits.Current iLimit;
parameter Modelica.SIunits.Time delayTime
"Delay with respect to useIntegrator Signal";
//-----------------------------------------------------------------------
// parameters of limiter
parameter Real LRmax "Upper limits of LR signals";
parameter Real LRmin "Lower limits of LR signals";
parameter Real TRmax "Upper limits of TR signals";
parameter Real TRmin "Lower limits of TR signals";
public  SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.SEDBusAIMC  sedBusAIMC ;
protected
Modelica.Electrical.Machines.SpacePhasors.Blocks.
Rotator rotatorI;
Modelica.Electrical.Machines.SpacePhasors.Blocks.
ToSpacePhasor toSpacePhasorI;
public
Modelica.Blocks.Math.Gain gain(final k=p);
Components.Compl_Div complDivI;
```

```
Components.Compl_Div complDivV;
Components.LR_Tracker LRTrackerI;
Components.LR_Tracker LRTrackerV;
Modelica.Blocks.Math.Feedback feedbackLR;
Modelica.Blocks.Math.Feedback feedbackTR;
Modelica.Electrical.Machines.SpacePhasors.Blocks.
FromPolar fromPolarPsi;
protected
Modelica.Electrical.Machines.SpacePhasors.Blocks.Rotator rotatorPsi;
public
Modelica.Electrical.Machines.SpacePhasors.Blocks.
FromPolar fromPolarPsi_r;
Components.Integrator_Bool integratorBoolLR(
final y_start=LRstart,
final k= k1);
Components.TrustVoltageModel trustVoltageModel(
final fNominal=fNominal,
final p=p,
final m=3,
final wLimit=wLimit,
final iLimit=iLimit,
final delayTime=delayTime);
Modelica.Blocks.Nonlinear.Limiter limiterLR(
final uMax=LRmax,
final uMin=LRmin);
Modelica.Blocks.Nonlinear.Limiter limiterTR(
final uMax=TRmax,
final uMin=TRmin);
Components.Integrator_Bool integratorBoolTR(
final y_start=TRstart,
final k=  k2);

equation
connect(complDivV.LR, LRTrackerV.LR);
connect(LRTrackerV.LRout, feedbackLR.u1);
connect(LRTrackerI.LRout, feedbackLR.u2);
connect(LRTrackerI.MagnitudeLR, feedbackTR.u1);
connect(LRTrackerV.MagnitudeLR, feedbackTR.u2);
connect(feedbackTR.y,integratorBoolTR. u);
connect(feedbackLR.y,integratorBoolLR. u);
connect(integratorBoolTR.y,limiterTR. u);
connect(integratorBoolLR.y,limiterLR. u);
connect(fromPolarPsi.u[1], sedBusAIMC.psi);
connect(fromPolarPsi.y,rotatorPsi. u);
connect(toSpacePhasorI.y,rotatorI. u);
connect(gain.u, sedBusAIMC.phiMechanical);
connect(gain.y,rotatorI. angle);
connect(rotatorPsi.angle, gain.y);
connect(fromPolarPsi.u[2], sedBusAIMC.gammaPsi);
```

```
connect(toSpacePhasorI.u, sedBusAIMC.i);
connect(rotatorPsi.y,complDivI. psi);
connect(rotatorI.y,complDivI. i);
connect(trustVoltageModel.wMechanical, sedBusAIMC.wMechanical);
connect(trustVoltageModel.i, sedBusAIMC.i);
connect(fromPolarPsi_r.u[2], sedBusAIMC.gammaPsi_r);
connect(fromPolarPsi_r.u[1], sedBusAIMC.psi_r);
connect(fromPolarPsi_r.y,complDivV. psi);
connect(toSpacePhasorI.y,complDivV. i);
connect(complDivI.LR, LRTrackerI.LR);
connect(trustVoltageModel.trustVoltage,integratorBoolLR. enable);
connect(trustVoltageModel.trustVoltage, LRTrackerV.enable);
connect(trustVoltageModel.trustVoltage,complDivV. enable);
connect(LRTrackerI.enable, trustVoltageModel.trustVoltage);
connect(complDivI.enable, trustVoltageModel.trustVoltage);
connect(integratorBoolTR.enable, trustVoltageModel.trustVoltage);
connect(limiterTR.y, sedBusAIMC.TR);
connect(limiterLR.y, sedBusAIMC.LR);

end Tracking;
```

Das Modell *Temperature* der Tracking Library besitzt folgenden Modelica-Quellcode:

```
model Temp
"Calculates the rotor  and stator tempreture"

SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.SEDBusAIMC  sedBusAIMC ;

parameter Real k;
parameter SEDTracking.SIunits.LinearTemperatureCoefficient alfa20_r;
parameter Modelica.SIunits.Resistance RR_ref;
parameter Modelica.SIunits.Temp_K Tr_ref;
parameter Modelica.SIunits.Temp_K T_amb;
Components.RotorTemp rotorTemp(
final alfa20_r=alfa20_r,
final Tr_ref=Tr_ref,
final Rr_ref=RR_ref);
Modelica.Blocks.Math.Division divisionRr;
Modelica.Blocks.Sources.RealExpression Tamb(final y=T_amb);
Modelica.Blocks.Math.Feedback DeltaTr;
Modelica.Blocks.Math.Gain DeltaTs(final k=k);
Modelica.Blocks.Math.Add add;

equation
connect(divisionRr.u1, sedBusAIMC.LR);
connect(divisionRr.u2, sedBusAIMC.TR);
connect(divisionRr.y, rotorTemp.R_r);
connect(rotorTemp.Tr, sedBusAIMC.Tr);
connect(rotorTemp.Tr, DeltaTr.u1);
```

```
connect(DeltaTr.y, DeltaTs.u);
connect(Tamb.y, DeltaTr.u2);
connect(Tamb.y, add.u1);
connect(DeltaTs.y, add.u2);
connect(add.y, sedBusAIMC.Ts);

end Temp;
```

Das *TrustVoltageModel* in der Tracking Library hat folgenden Modelica-Quellcode:

```
model TrustVoltageModel
"Sets  the limits of angular velocity and current"

extends Modelica.Blocks.Interfaces.BlockIcon;
parameter Modelica.SIunits.Frequency fNominal(
final min=Modelica.Constants.small);
constant Real pi = Modelica.Constants.pi;
parameter Integer p( min= 1);
constant Integer m=3 "Number of phases";
final parameter Integer nin = m;
Modelica.Blocks.Interfaces.RealInput i[nin];
parameter Modelica.SIunits.AngularVelocity wLimit;
parameter Modelica.SIunits.Current iLimit;
parameter Modelica.SIunits.Time delayTime
"Delay with respect to useIntegrator Signal";
Modelica.Blocks.Interfaces.RealInput wMechanical;
Modelica.Blocks.Interfaces.BooleanOutput useIntegrator;
Modelica.Blocks.Interfaces.BooleanOutput trustVoltage;
Modelica.Blocks.Logical.Timer timer;
Modelica.Blocks.Sources.RealExpression realExpression1(
final y=wLimit);
Modelica.Blocks.Logical.Greater greater;
Modelica.Blocks.Sources.RealExpression realExpression2(
final y=-wLimit);
Modelica.Blocks.Logical.Less less;
Modelica.Blocks.Logical.Or or1;
Modelica.Electrical.Machines.SpacePhasors.Blocks.
ToPolar toPolar;
Modelica.Electrical.Machines.SpacePhasors.Blocks.
ToSpacePhasor toSpacePhasor;
Modelica.Blocks.Sources.RealExpression realExpression(
final y=iLimit);
Modelica.Blocks.Logical.Greater greater1;
Modelica.Blocks.Logical.And And;
Modelica.Blocks.Logical.GreaterThreshold greaterThreshold(
final threshold=  delayTime);

equation
connect(wMechanical,greater. u1);
connect(realExpression1.y, greater.u2);
```

```
connect(wMechanical,less. u1);
connect(realExpression2.y,less. u2);
connect(greater.y, or1.u1);
connect(or1.u2, less.y);
connect(i, toSpacePhasor.u);
connect(toSpacePhasor.y, toPolar.u);
connect(realExpression.y, greater1.u2);
connect(toPolar.y[1], greater1.u1);
connect(greater1.y, And.u1);
connect(or1.y, And.u2);
connect(And.y, timer.u);
connect(And.y, useIntegrator);
connect(timer.y, greaterThreshold.u);
connect(greaterThreshold.y, trustVoltage);

end TrustVoltageModel;
```

A.2 Programmlisting Examples

Das Beispiel Detuning mit Parameterverstimmung hat folgenden Modelica-Quellcode.

```
model Detuning

constant Real pi=Modelica.Constants.pi;
//parameters of the machine
constant Integer m=3 "Number of phases";
parameter Integer p=4;
parameter Modelica.SIunits.Voltage VNominal=400;
parameter Modelica.SIunits.Current INominal=415.8;
parameter Modelica.SIunits.Frequency fNominal=50;
parameter Modelica.SIunits.Inertia Jr=35;
parameter Modelica.SIunits.AngularVelocity wNominal=2*pi*fNominal/p;
parameter Modelica.SIunits.Torque tauNominal=5226.79;
parameter Modelica.SIunits.Resistance Rs=8.086e-3;
parameter Modelica.SIunits.Inductance Lssigma=3.001e-4;
parameter Modelica.SIunits.Inductance Lm=8.231e-3;
parameter Modelica.SIunits.Resistance Rr=4.934e-3;
parameter Modelica.SIunits.Inductance Lrsigma=5.020e-4;
//------------------------------------------------------------------------
// parameters of the foc
parameter Real k1= 1 "Stray Inductance Detuning";
parameter Real k2= 1 "Main Field Inductance Detuning";
parameter Real k3= 1 "Rotor Resistance Detuning";
//---------------Detuned parameter-----------------------------------------------------
parameter  Real sigma=1 -Lm^2/((Lssigma + Lm)*(Lrsigma + Lm))
"Stray coefficient";
parameter Modelica.SIunits.Resistance RS=Rs
"Warm stator resistance per phase";
parameter Modelica.SIunits.Resistance RR=k3*Rr*(Lm/(Lm + Lrsigma))^2
"Warm rotor resistance per phase";
parameter Modelica.SIunits.Inductance LS=Lm + Lssigma;
parameter Modelica.SIunits.Inductance Lsigma=k1*sigma*LS
"Stator stray inductance per phase";
parameter Modelica.SIunits.Inductance LM= k2*(1-sigma)*LS;

//-------------------------------------------------------------------------------------
// parameters of the load
parameter Modelica.SIunits.Inertia J=50;
//-------------------------------------------------------------------------------------
// references and limits
parameter Modelica.SIunits.Voltage vMachineMax=VNominal
"Maximum phase voltage of the machine (RMS value)";
parameter Modelica.SIunits.Current iMachineMax=500
"Maximum phase current of the machine (RMS value)";
parameter Modelica.SIunits.Current IConverterMax=800
"Maximum admissible converter DC supply current";
parameter Real fluxLevel=0.95 "Per unit reference flux level";
```

```
//-------------------------------------------------------------------------------------
// parameters of the converter
parameter Modelica.SIunits.Time TiConverter=1e-3;
//-------------------------------------------------------------------------------------
// parameters of the controller
parameter Real kpxCurrent=0.4988 "Gain of x-component";
parameter Modelica.SIunits.Time TixCurrent=0.0620
"Integral time constant of x-component";
parameter Real kpyCurrent=0.4988 "Gain of y-component";
parameter Modelica.SIunits.Time TiyCurrent=0.0620
"Integral time constant of y-component";
parameter Real kpFlux=73581.7 "Gain";
parameter Modelica.SIunits.Time TiFlux=0.0062 "Integral time constant" ;
parameter Real kpFluxWeak=0.0013 "Gain" ;
parameter Modelica.SIunits.Time TiFluxWeak=0.0062 "Integral time constant";
parameter Real kpSpeed=24595.2 "Gain";
parameter Modelica.SIunits.Time TiSpeed=0.0062 "Integral time constant";
//-------------------------------------------------------------------------------------
// parameters of battery
parameter Modelica.SIunits.Voltage VCellNominal = 1000 "Nominal cell voltage";
parameter Modelica.SIunits.Current ICellMax = 800
"Maximum admissable cell current";
parameter Modelica.SIunits.Resistance RsCell = 6e-3 "Internal cell resistor";
parameter Integer nsBatt(min=1) = 1 "Number of series connected cells";
parameter Integer npBatt(min=1) = 1 "Number of parallel connected cells";
SmartElectricDrives.Converters.PowerBalance.DCAC.ThreePhase dcac1(
VNominal=VNominal,
INominal=INominal,
TiConverter=TiConverter,
IConverterMax=IConverterMax);
Modelica.Electrical.Analog.Basic.Ground ground1;
SmartElectricDrives.Interfaces.BusAdaptors.TauRefIn tauref1;
Modelica.Mechanics.Rotational.Inertia inertia1(J=J);
SmartElectricDrives.Sources.Batteries.BatteryIdeal batteryIdeal1(
VCellNominal=VCellNominal,
ICellMax=ICellMax,
RsCell=RsCell,
ns=nsBatt,
np=npBatt);
public
Modelica.Electrical.Machines.BasicMachines.AsynchronousInductionMachines
.AIM_SquirrelCage  aimc_correct( final p=p,
final Rs=Rs,
final Lssigma=Lssigma,
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr,
final J_Rotor=Jr);
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
```

```
Sensors.MBox  mBox1(StarDelta="Y");
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.FluxModels.
CurrentModel  currentModel_correct( final p=p,
final Lssigma=Lssigma,
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr);
Modelica.Electrical.Machines.Examples.Utilities.TerminalBox terminalBox(
StarDelta="Y");
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.BusAdaptors.VDCIn  vdc1;
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
FieldOrientedControl.FOC  foc_correct(final p=p,
final VNominal=VNominal,
final fNominal=fNominal,
final Lssigma=Lssigma,
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr,
final Rs=Rs,
vMachineMax=vMachineMax,
iMachineMax=iMachineMax,
fluxLevel=fluxLevel,
final kpFlux=kpFlux,
final TiFlux=TiFlux,
final TixCurrent=TixCurrent,
final kpyCurrent=kpyCurrent,
final TiyCurrent=TiyCurrent,
final kpxCurrent=kpxCurrent,
final kpFluxWeak=kpFluxWeak,
final TiFluxWeak=TiFluxWeak,
StarDelta="Y",
externalVoltageMachineMax=false,
externalCurrentMachineMax=false,
externalFluxLevel=false);

SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.Interface  interface1;
public
Modelica.Electrical.Analog.Basic.Ground ground3;
Modelica.Blocks.Sources.TimeTable timeTable1(
table=[0,0;
       2,0;
       2,tauNominal;
       3,tauNominal;
       3,0;
       4,0;
       4,-tauNominal;
       5,-tauNominal;
```

```
       5,0]) ;
SmartElectricDrives.Converters.PowerBalance.DCAC.ThreePhase dcac2(
VNominal=VNominal,
INominal=INominal,
TiConverter=TiConverter,
IConverterMax=IConverterMax;
Modelica.Electrical.Analog.Basic.Ground ground2;
SmartElectricDrives.Interfaces.BusAdaptors.TauRefIn tauref2;
Modelica.Mechanics.Rotational.Inertia inertia2(J=J);
SmartElectricDrives.Sources.Batteries.BatteryIdeal batteryIdeal2(
VCellNominal=VCellNominal,
ICellMax=ICellMax,
RsCell=RsCell,
ns=nsBatt,
np=npBatt);
public
Modelica.Electrical.Machines.BasicMachines.AsynchronousInductionMachines.
AIM_SquirrelCage aimc_defect( final p=p,
final Rs=Rs,
final Lssigma=Lssigma,
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr,
final J_Rotor=Jr);
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Sensors.MBox mBox2(StarDelta="Y");
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.FluxModels.
CurrentModel currentModel_defect( final p=p,
final Lrsigma=Lrsigma,
final Lm=1/((1 -sigma))*LM -Lssigma,
final Rr=RR*(Lm + Lrsigma)^2/Lm^2,
final Lssigma=1/(sigma)*Lsigma -Lm);

Modelica.Electrical.Machines.Examples.Utilities.TerminalBox
terminalBox1(StarDelta="Y") ;
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.BusAdaptors.VDCIn vdc2 ;
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
FieldOrientedControl.FOC foc_defect( final p=p,
final VNominal=VNominal,
final fNominal=fNominal,
final Rs=Rs,
vMachineMax=vMachineMax,
iMachineMax=iMachineMax,
fluxLevel=fluxLevel,
final kpFlux=kpFlux,
final TiFlux=TiFlux,
final TixCurrent=TixCurrent,
final kpyCurrent=kpyCurrent,
```

```
final TiyCurrent=TiyCurrent,
final kpxCurrent=kpxCurrent,
final kpFluxWeak=kpFluxWeak,
final TiFluxWeak=TiFluxWeak,
StarDelta="Y",
externalVoltageMachineMax=false,
externalCurrentMachineMax=false,
externalFluxLevel=false,
final Lrsigma=Lrsigma,
final Lm=1/((1 -sigma))*LM -Lssigma,
final Rr=RR*(Lm + Lrsigma)^2/Lm^2,
final Lssigma=1/(sigma)*Lsigma -Lm);
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.Interface interface2 ;
public
Modelica.Electrical.Analog.Basic.Ground ground4;
Modelica.Mechanics.Rotational.ConstantSpeed fan1(w_fixed=wNominal);
Modelica.Mechanics.Rotational.ConstantSpeed fan2(w_fixed=wNominal);
Modelica.Blocks.Sources.TimeTable timeTable2(
table=[0,0;
       2,0;
       2,tauNominal;
       3,tauNominal;
       3,0;
       4,0;
       4,-tauNominal;
       5,-tauNominal;
       5,0]) ;

equation
connect(dcac1.ground, ground1.p);
connect(dcac1.pSupply, batteryIdeal1.p);
connect(dcac1.nSupply, batteryIdeal1.n);
connect(currentModel_correct.sedBusAIMC, mBox1.sedBusAIMC);
connect(mBox1.pMachine, terminalBox.plugToGrid);
connect(terminalBox.positiveMachinePlug, aimc_correct.plug_sp);
connect(mBox1.sedBusAIMC, vdc1.sedBusAIMC);
connect(mBox1.sedBusAIMC, foc_correct.sedBusAIMC);
connect(interface1.sedBusAIMC, vdc1.sedBusAIMC);
connect(terminalBox.negativeMachinePlug, aimc_correct.plug_sn);
connect(dcac1.vDC, vdc1.u);
connect(foc_correct.vRef, dcac1.vRef);
connect(tauref1.controlBus, interface1.controlBus);
connect(inertia1.flange_b, aimc_correct.flange_a);
connect(mBox1.pConverter, dcac1.pLoad);
connect(mBox1.encoder, aimc_correct.flange_a);
connect(ground3.p, batteryIdeal1.n);
connect(timeTable1.y, tauref1.u);
connect(dcac2.ground, ground2.p);
```

```
connect(dcac2.pSupply,batteryIdeal2. p);
connect(dcac2.nSupply,batteryIdeal2. n);
connect(currentModel_defect.sedBusAIMC, mBox2.sedBusAIMC);
connect(mBox2.pMachine, terminalBox1.plugToGrid);
connect(terminalBox1.positiveMachinePlug, aimc_defect.plug_sp);
connect(mBox2.sedBusAIMC,vdc2. sedBusAIMC);
connect(mBox2.sedBusAIMC, foc_defect.sedBusAIMC);
connect(interface2.sedBusAIMC,vdc2. sedBusAIMC);
connect(terminalBox1.negativeMachinePlug, aimc_defect.plug_sn);
connect(dcac2.vDC,vdc2. u);
connect(foc_defect.vRef, dcac2.vRef);
connect(tauref2.controlBus,interface2. controlBus);
connect(inertia2.flange_b, aimc_defect.flange_a);
connect(mBox2.pConverter,dcac2. pLoad);
connect(mBox2.encoder, aimc_defect.flange_a);
connect(ground4.p,batteryIdeal2. n);
connect(fan1.flange, inertia1.flange_a);
connect(inertia2.flange_a,fan2. flange);
connect(timeTable2.y,tauref2. u);

end Detuning;
```

Die am Netz betriebene Asynchronmaschine wird im Beispiel *TrackingTest_ Netz* untersucht, der Modelica-Quellcode zu diesem Beispiel wird folglich angegeben.

```
model TrackingTest_Netz

//parameters of asm
constant Integer m=3 "Number of phases";
parameter Integer p=4;
parameter Modelica.SIunits.Voltage VNominal=400;
parameter Modelica.SIunits.Current INominal=415.8;
parameter Modelica.SIunits.Frequency fNominal=50;
parameter Modelica.SIunits.Inertia Jr=35;
parameter Modelica.SIunits.AngularVelocity
wNominal=2*pi*fNominal/p;
parameter Modelica.SIunits.Torque tauNominal=5226.79;
parameter Modelica.SIunits.Resistance Rs=8.086e-3;
parameter Modelica.SIunits.Inductance Lssigma=3.001e-4;
parameter Modelica.SIunits.Inductance Lm=8.231e-3;
parameter Modelica.SIunits.Resistance Rr=4.934e-3;
parameter Modelica.SIunits.Inductance Lrsigma=5.020e-4;
//-------------------------------------------------------------------------------
//parameters of temp-box
parameter Modelica.SIunits.Temp_K Tr_ref = 293.15;
parameter Modelica.SIunits.Resistance Rr_ref = Rr/(1+alfa20_r*(Tr-Tr_ref));
parameter SEDTracking.SIunits.LinearTemperatureCoefficient alfa20_r=0.004;
parameter Modelica.SIunits.Temp_K T_amb = 293.15;
//-------------------------------------------------------------------------------
//parameters of trust voltage
```

```
parameter Modelica.SIunits.AngularVelocity wLimit = 0.1* 2*pi*fNominal/p;
parameter Modelica.SIunits.Current iLimit=270;
parameter Modelica.SIunits.Time delayTime=1
"Comparison with respect to useIntegrator Signal";
//-------------------------------------------------------------------------------
//parameters of integrator_bool
parameter Real k1 = 0.1;
parameter Real k2= 100;
parameter Real y_start = 1E-6;
//-------------------------------------------------------------------------------
//parameters of Rs
parameter Modelica.SIunits.Temp_K Ts_ref = 293.15;
parameter Modelica.SIunits.Resistance Rs_ref = Rs/(1+alfa20_s*(Ts-Ts_ref));
//Formel gilt nur fuer Ts_ref = 293.15 Kelvin
parameter SEDTracking.SIunits.LinearTemperatureCoefficient alfa20_s=0.0039;
//-------------------------------------------------------------------------------
//parameters of limiter
parameter Real LRmax=2*LRstart "Upper limits of LR signals";
parameter Real LRmin= LRstart/2 "Lower limits of LR signals";
parameter Real TRmax=2*TRstart "Upper limits of TR signals";
parameter Real TRmin= TRstart/2 "Lower limits of TR signals";
parameter Boolean limitsAtInit = true
"= false, if limits are ignored during initializiation ";
parameter Modelica.SIunits.Inductance LRstart = 7.88396e-3;
parameter Modelica.SIunits.Time TRstart = 1.7969964;
constant Real pi=Modelica.Constants.pi;
//-------------------------------------------------------------------------------
//parameters of source
parameter Modelica.SIunits.Voltage VSupply=660;
parameter Modelica.SIunits.Frequency fSupply=50;
parameter Modelica.SIunits.Resistance RLine=1e-3;
parameter Modelica.SIunits.Inductance LLine=1e-4;
parameter Modelica.SIunits.Inertia J=50;
//-------------------------------------------------------------------------------
//parameters of temp
parameter Modelica.SIunits.Temp_K Ts = 393.15;
parameter Modelica.SIunits.Temp_K Tr = Ts;
parameter Real k = 1;
parameter Real sigma=1-Lm^2/((Lssigma+Lm)*(Lrsigma+Lm))
"Stray coefficient";
parameter Modelica.SIunits.Inductance LS=(Lm+Lssigma)
"Stator inductance per phase";
parameter Modelica.SIunits.Inductance LR=(1-sigma)*LS
"Rotor inductance per phase";
parameter Modelica.SIunits.Time TR=(Lm+Lrsigma)/Rr
"Rotor time constant";
parameter Modelica.SIunits.Resistance RR = LR/TR;
parameter Modelica.SIunits.Resistance RR_ref = RR/(1+alfa20_r*(Tr-Tr_ref);
Modelica.Electrical.Machines.BasicMachines.AsynchronousInductionMachines.
```

```
AIM_SquirrelCage AIMC(final J_Rotor=Jr,
final p=p,
final fNominal=fNominal,
final Rs=Rs,
final Lssigma=Lssigma,
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr);
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Sensors.MBox mBox(StarDelta="Y");
Modelica.Electrical.Machines.Examples.Utilities.TerminalBox terminalBox;
SEDTracking.NewFluxModels.VoltageModel voltageModel(p=p,
Lm=Lm,
Lrsigma=Lrsigma,
Lssigma=Lssigma,
final m=m,
final fNominal=fNominal,
final wLimit=wLimit,
final iLimit=iLimit,
final delayTime=delayTime,
final y_start=y_start,
final alfa20_s=alfa20_s,
final Rs_ref=Rs_ref,
final Ts_ref=Ts_ref,
kFeedback=0.5) ;
Modelica.Electrical.MultiPhase.Basic.Star star;
Modelica.Electrical.MultiPhase.Basic.Resistor resistor(
R=fill(RLine, 3), m=3);
Modelica.Electrical.MultiPhase.Basic.Inductor inductor(
m=3, L=fill(LLine, 3));
Modelica.Electrical.Analog.Basic.Ground ground;
Modelica.Electrical.MultiPhase.Sources.SineVoltage sineVoltage(
V=fill(VSupply,3),
freqHz=fill(fSupply, 3));
SEDTracking.NewFluxModels.CurrentModel currentModel(final p=p);
SEDTracking.NewFluxModels.Testbox test-box(
Lssigma=Lssigma,
Lm=Lm,
Lrsigma=Lrsigma,
Rr=Rr);
Models_.NewFluxModels.Temp_old temp_1( Tr=Tr, Ts=Tr);
Modelica.Mechanics.Rotational.Inertia inertia(J=J);
Modelica.Mechanics.Rotational.Torque Last;
Modelica.Electrical.Machines.BasicMachines.
AsynchronousInductionMachines.AIM_SquirrelCage AIMC1(final J_Rotor=Jr,
final p=p,
final fNominal=fNominal,
final Rs=Rs,
final Lssigma=Lssigma,
```

```
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr);

SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Sensors.MBox  mBox1(StarDelta="Y");
Modelica.Electrical.Machines.Examples.Utilities.TerminalBox terminalBox1;
SEDTracking.NewFluxModels.VoltageModel voltageModel1(p=p,
Lm=Lm,
Lrsigma=Lrsigma,
Lssigma=Lssigma,
final m=m,
final fNominal=fNominal,
final wLimit=wLimit,
final iLimit=iLimit,
final delayTime=delayTime,
final y_start=y_start,
final alfa20_s=alfa20_s,
final Rs_ref=Rs_ref,
final Ts_ref=Ts_ref,
kFeedback=0.5);
Modelica.Electrical.MultiPhase.Basic.Star star1;
Modelica.Electrical.MultiPhase.Basic.Resistor resistor1(
R=fill(RLine, 3), m=3);
Modelica.Electrical.MultiPhase.Basic.Inductor inductor1(
m=3, L=fill(LLine, 3));
Modelica.Electrical.Analog.Basic.Ground ground1;

Modelica.Electrical.MultiPhase.Sources.SineVoltage sineVoltage1(
V=fill(VSupply,  3), freqHz=fill(fSupply, 3));
NewFluxModels.Tracking Trck(m=m,
p=p,
VNominal=VNominal,
INominal=INominal,
fNominal=fNominal,
tauNominal=tauNominal,
Rs=Rs,
Lssigma=Lssigma,
Lm=Lm,
Rr=Rr,
Lrsigma=Lrsigma,
k1=k1,
k2=k2,
LRstart=LRstart,
TRstart=TRstart,
wLimit=wLimit,
iLimit=iLimit,
delayTime=delayTime,
LRmax=LRmax,
```

```
LRmin=LRmin,
TRmax=TRmax,
TRmin=TRmin,
Jr=Jr);
SEDTracking.NewFluxModels.CurrentModel currentModel1(p=p);
Modelica.Mechanics.Rotational.Inertia inertia1(  J=J);
SEDTracking.NewFluxModels.Temp temp(
alfa20_r=alfa20_r,
RR_ref=RR_ref,
Tr_ref=Tr_ref,
T_amb=T_amb,
k=1);
Modelica.Mechanics.Rotational.Torque Last1;
Modelica.Blocks.Sources.TimeTable timeTable1(
table=[ 0,0;
       10,0;
       10,tauNominal;
      150,tauNominal]);
Modelica.Blocks.Sources.TimeTable timeTable(
table=[ 0,0;
       10,0;
       10,tauNominal;
      150,tauNominal]);

equation
connect(voltageModel.sedBusAIMC, mBox.sedBusAIMC);
connect(ground.p,star. pin_n);
connect(resistor.plug_n,inductor. plug_p);
connect(sineVoltage.plug_n,star. plug_p);
connect(sineVoltage.plug_p,resistor. plug_p);
connect(mBox.pConverter,inductor. plug_n);
connect(mBox.encoder,AIMC. flange_a);
connect(AIMC.plug_sp,terminalBox. positiveMachinePlug);
connect(AIMC.plug_sn,terminalBox. negativeMachinePlug);
connect(terminalBox.plugToGrid,mBox. pMachine);
connect(currentModel.sedBusAIMC1, mBox.sedBusAIMC);
connect(testtbox.sedBusAIMC, currentModel.sedBusAIMC1);
connect(temp_1.sedBusAIMC, voltageModel.sedBusAIMC);
connect(Last.flange_b,inertia. flange_b);
connect(inertia.flange_a,AIMC. flange_a);
connect(ground1.p, star1.pin_n);
connect(resistor1.plug_n, inductor1.plug_p);
connect(sineVoltage1.plug_n, star1.plug_p);
connect(sineVoltage1.plug_p, resistor1.plug_p);
connect(mBox1.pConverter, inductor1.plug_n);
connect(AIMC1.plug_sp, terminalBox1.positiveMachinePlug);
connect(AIMC1.plug_sn, terminalBox1.negativeMachinePlug);
connect(terminalBox1.plugToGrid,mBox1. pMachine);
connect(mBox1.encoder, AIMC1.flange_a);
```

```
connect(voltageModel1.sedBusAIMC,mBox1. sedBusAIMC);
connect(Trck.sedBusAIMC,mBox1. sedBusAIMC);
connect(currentModel1.sedBusAIMC1,mBox1. sedBusAIMC);
connect(AIMC1.flange_a, inertia1.flange_a);
connect(temp.sedBusAIMC, voltageModel1.sedBusAIMC);
connect(Last1.flange_b, inertia1.flange_b);
connect(timeTable1.y, Last1.tau) ;
connect(timeTable.y, Last.tau);

end TrackingTest_Netz;
```

Die feldorientiert geregelte Asynchronmaschine mit und ohne Parametertracking wird anhand des *TrackingTest_FOC* Beispiels untersucht. Das Beispiel hat folgenden Modelica-Quellcode.

```
model TrackingTest_FOC

constant Real pi=Modelica.Constants.pi;
//parameters of asm
constant Integer m=3 "Number of phases";
parameter Integer p=4;
parameter Modelica.SIunits.Voltage VNominal=400;
parameter Modelica.SIunits.Current INominal=415.8;
parameter Modelica.SIunits.Frequency fNominal=50;
parameter Modelica.SIunits.Inertia Jr=35;
parameter Modelica.SIunits.AngularVelocity wNominal=2*pi*fNominal/p;
parameter Modelica.SIunits.Torque tauNominal=5226.79;
parameter Modelica.SIunits.Resistance Rs=8.086e-3;
parameter Modelica.SIunits.Inductance Lssigma=3.001e-4;
parameter Modelica.SIunits.Inductance Lm=8.231e-3;
parameter Modelica.SIunits.Resistance Rr=4.934e-3;
parameter Modelica.SIunits.Inductance Lrsigma=5.020e-4;
//-------------------------------------------------------------------------------
//parameters of temp-box
parameter Modelica.SIunits.Temp_K Tr_ref = 293.15;
parameter Modelica.SIunits.Resistance Rr_ref = Rr/(1+alfa20_r*(Tr-Tr_ref));
parameter SEDTracking.SIunits.LinearTemperatureCoefficient alfa20_r=0.004;
parameter Modelica.SIunits.Temp_K T_amb = 293.15;
//-------------------------------------------------------------------------------
// parameters of rotorTemp
parameter Modelica.SIunits.Temp_K Ts = 393.15;
parameter Modelica.SIunits.Temp_K Tr = Ts;
parameter Real k = 1;
parameter Real sigma=1-Lm^2/((Lssigma+Lm)*(Lrsigma+Lm)) "Stray coefficient";
parameter Modelica.SIunits.Inductance LS=(Lm+Lssigma)
"Stator inductance per phase";
parameter Modelica.SIunits.Inductance LR=(1-sigma)*LS
"Rotor inductance per phase";
parameter Modelica.SIunits.Time TR=(Lm+Lrsigma)/Rr "Rotor time constant";
parameter Modelica.SIunits.Resistance RR = LR/TR;
parameter Modelica.SIunits.Resistance RR_ref = RR/(1+alfa20_r*(Tr-Tr_ref));
```

```
//-------------------------------------------------------------------------------------
//parameters of trustvoltage
parameter Modelica.SIunits.AngularVelocity wLimit = 0.1* 2*pi*fNominal/p;
parameter Modelica.SIunits.Current iLimit=270;
parameter Modelica.SIunits.Time delayTime=1
"Comparison with respect to useIntegrator Signal";
//-------------------------------------------------------------------------------------
//parameters of Integrator_bool
parameter Real k1 = 0.1;
parameter Real k2= 100;
parameter Real y_start = 1E-6;
//-------------------------------------------------------------------------------------
//parameters of Rs
parameter Modelica.SIunits.Temp_K Ts_ref = 293.15;
parameter Modelica.SIunits.Resistance Rs_ref = Rs/(1+alfa20_s*(Ts-Ts_ref));
//Formel gilt nur fuer Ts_ref = 293.15 Kelvin
parameter SEDTracking.SIunits.LinearTemperatureCoefficient alfa20_s=0.0039;
//-------------------------------------------------------------------------------------
//parameters of limiter
parameter Real LRmax=2*LRstart "Upper limits of LR signals";
parameter Real LRmin= LRstart/2 "Lower limits of LR signals";
parameter Real TRmax=2*TRstart "Upper limits of TR signals";
parameter Real TRmin= TRstart/2 "Lower limits of TR signals";
parameter Boolean limitsAtInit = true
"= false, if limits are ignored during initializiation ";
parameter Modelica.SIunits.Inductance LRstart = 7.88396e-3;
parameter Modelica.SIunits.Time TRstart= 1.7969964;
//-------------------------------------------------------------------------------------
//  parameters of load
parameter Modelica.SIunits.Inertia J=50;
//-------------------------------------------------------------------------------------
// references and limits
parameter Modelica.SIunits.Voltage vMachineMax=VNominal
"Maximum phase voltage of the machine (RMS value)";
parameter Modelica.SIunits.Current iMachineMax=500
"Maximum phase current of the machine (RMS value)";
parameter Modelica.SIunits.Current IConverterMax=800
"Maximum admissible converter DC supply current";
parameter Real fluxLevel=1 "Per unit reference flux level";
//-------------------------------------------------------------------------------------
// parameters of the converter
parameter Modelica.SIunits.Time TiConverter=1e-3;
//-------------------------------------------------------------------------------------
// parameters of the controller
parameter Real kpxCurrent=0.4988 "Gain of x-component";
parameter Modelica.SIunits.Time TixCurrent=0.0620
"Integral time constant of x-component";
parameter Real kpyCurrent=0.4988 "Gain of y-component";
parameter Modelica.SIunits.Time TiyCurrent=0.0620
```

```
"Integral time constant of y-component";
parameter Real kpFlux=73581.7 "Gain";
parameter Modelica.SIunits.Time TiFlux=0.0062 "Integral time constant";
parameter Real kpFluxWeak=0.0013 "Gain";
parameter Modelica.SIunits.Time TiFluxWeak=0.0062 "Integral time constant";
parameter Real kpSpeed=24595.2 "Gain";
parameter Modelica.SIunits.Time TiSpeed=0.0062 "Integral time constant";
//-------------------------------------------------------------------------------
// parameters of Battery
parameter Modelica.SIunits.Voltage VCellNominal = 1000 "Nominal cell voltage";
parameter Modelica.SIunits.Current ICellMax = 800
"Maximum admissable cell current";
parameter Modelica.SIunits.Resistance RsCell = 6e-3 "Internal cell resistor";
parameter Integer nsBatt(min=1) = 1 "Number of series connected cells";
parameter Integer npBatt(min=1) = 1 "Number of parallel connected cells";

SmartElectricDrives.Converters.PowerBalance.DCAC.
ThreePhase dcac1(VNominal=VNominal,
INominal=INominal,
TiConverter=TiConverter,
IConverterMax=IConverterMax);
Modelica.Electrical.Analog.Basic.Ground ground1;
SmartElectricDrives.Interfaces.BusAdaptors.TauRefIn tauref1;
Modelica.Mechanics.Rotational.Inertia inertia1(J=J);
SmartElectricDrives.Sources.Batteries.BatteryIdeal batteryIdeal1(
VCellNominal=VCellNominal,
ICellMax=ICellMax,
RsCell=RsCell,
ns=nsBatt,
np=npBatt);
public
Modelica.Electrical.Machines.BasicMachines.AsynchronousInductionMachines.
AIM_SquirrelCage  AIMC(final p=p,
final Rs=Rs,
final Lssigma=Lssigma,
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr,
final J_Rotor=Jr) ;

SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.Sensors.
MBox  mBox1(StarDelta="Y");
CurrentModel currentModel(final p=p,
final Lssigma=Lssigma,
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr);

Modelica.Electrical.Machines.Examples.Utilities.TerminalBox
```

```
terminalBox(  StarDelta="Y");
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.BusAdaptors.VDCIn  vdc1;
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
FieldOrientedControl.FOC foc(final p=p,
final VNominal=VNominal,
final fNominal=fNominal,
final Lssigma=Lssigma,
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr,
final Rs=Rs,
vMachineMax=vMachineMax,
iMachineMax=iMachineMax,
fluxLevel=fluxLevel,
final kpFlux=kpFlux,
final TiFlux=TiFlux,
final TixCurrent=TixCurrent,
final kpyCurrent=kpyCurrent,
final TiyCurrent=TiyCurrent,
final kpxCurrent=kpxCurrent,
final kpFluxWeak=kpFluxWeak,
final TiFluxWeak=TiFluxWeak,
StarDelta="Y",
externalVoltageMachineMax=false,
externalCurrentMachineMax=false,
externalFluxLevel=false);
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.Interface  interface1;
public
Modelica.Electrical.Analog.Basic.Ground ground3;
SmartElectricDrives.Converters.PowerBalance.
DCAC.ThreePhase dcac2(VNominal=VNominal,
INominal=INominal,
TiConverter=TiConverter,
IConverterMax=IConverterMax);
Modelica.Electrical.Analog.Basic.Ground ground2;
SmartElectricDrives.Interfaces.BusAdaptors.TauRefIn tauref2;
Modelica.Mechanics.Rotational.Inertia inertia2(J=J);
SmartElectricDrives.Sources.Batteries.BatteryIdeal batteryIdeal2(
VCellNominal=VCellNominal,
ICellMax=ICellMax,
RsCell=RsCell,
ns=nsBatt,
np=npBatt);
public
Modelica.Electrical.Machines.BasicMachines.AsynchronousInductionMachines.
AIM_SquirrelCage  AIMCTRCK(final p=p,
final Rs=Rs,
```

```
final Lssigma=Lssigma,
final Lm=Lm,
final Lrsigma=Lrsigma,
final Rr=Rr,
final J_Rotor=Jr);

SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Sensors.MBox  mBox_trck(StarDelta="Y");
Modelica.Electrical.Machines.Examples.Utilities.TerminalBox terminalBox1(
 StarDelta="Y");
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.BusAdaptors.VDCIn  vdc2;
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
FieldOrientedControl.FOC  foc_trck(final p=p,
final VNominal=VNominal,
final fNominal=fNominal,
final Rs=Rs,
vMachineMax=vMachineMax,
iMachineMax=iMachineMax,
fluxLevel=fluxLevel,
final kpFlux=kpFlux,
final TiFlux=TiFlux,
final TixCurrent=TixCurrent,
final kpyCurrent=kpyCurrent,
final TiyCurrent=TiyCurrent,
final kpxCurrent=kpxCurrent,
final kpFluxWeak=kpFluxWeak,
final TiFluxWeak=TiFluxWeak,
StarDelta="Y",
externalVoltageMachineMax=false,
externalCurrentMachineMax=false,
externalFluxLevel=false,
final Lrsigma=Lrsigma,
final Lssigma=Lssigma,
final Lm=Lm,
final Rr=Rr) ;
SmartElectricDrives.TransientDrives.AsynchronousInductionMachines.
Interfaces.Interface interface2;
public
Modelica.Electrical.Analog.Basic.Ground ground4;
Modelica.Mechanics.Rotational.ConstantSpeed fan1(w_fixed=wNominal);
Modelica.Mechanics.Rotational.ConstantSpeed fan2(w_fixed=wNominal);
SEDTracking.NewFluxModels.VoltageModel voltageModel_trck(
fNominal=fNominal,
delayTime=delayTime,
iLimit=iLimit,
wLimit=wLimit,
y_start=y_start,
alfa20_s=alfa20_s,
```

```
Rs_ref=Rs_ref,
Ts_ref=Ts_ref,
p=p,
Lssigma=Lssigma,
Lm=Lm,
Lrsigma=Lrsigma,
final kFeedback=0.5);
SEDTracking.NewFluxModels.CurrentModel currentModel_trck(p=p);
SEDTracking.NewFluxModels.Temp temp(
alfa20_r=alfa20_r,
RR_ref=RR_ref,
Tr_ref=Tr_ref,
T_amb=T_amb,
k=k);
SEDTracking.NewFluxModels.Tracking tracking( m=m,
p=p,
VNominal=VNominal,
INominal=INominal,
fNominal=fNominal,
Jr=Jr,
tauNominal=tauNominal,
Rs=Rs,
Lssigma=Lssigma,
Lm=Lm,
Rr=Rr,
Lrsigma=Lrsigma,
k1=k1,
k2=k2,
LRstart=LRstart,
TRstart=TRstart,
wLimit=wLimit,
iLimit=iLimit,
delayTime=delayTime,
LRmax=LRmax,
LRmin=LRmin,
TRmax=TRmax,
TRmin=TRmin);
Modelica.Blocks.Sources.TimeTable timeTable1(
table=[0,0;
      10,0;
      10,tauNominal;
     150,tauNominal]);
Modelica.Blocks.Sources.TimeTable timeTable2(
table=[0,0;
      10,0;
      10,tauNominal;
     150,tauNominal]);

equation
```

```
connect(dcac1.ground, ground1.p);
connect(dcac1.pSupply, batteryIdeal1.p);
connect(dcac1.nSupply, batteryIdeal1.n);
connect(currentModel.sedBusAIMC, mBox1.sedBusAIMC);
connect(mBox1.pMachine, terminalBox.plugToGrid);
connect(terminalBox.positiveMachinePlug,AIMC. plug_sp);
connect(mBox1.sedBusAIMC, vdc1.sedBusAIMC);
connect(mBox1.sedBusAIMC, foc.sedBusAIMC);
connect(interface1.sedBusAIMC, vdc1.sedBusAIMC);
connect(terminalBox.negativeMachinePlug,AIMC. plug_sn);
connect(dcac1.vDC, vdc1.u);
connect(foc.vRef, dcac1.vRef);
connect(tauref1.controlBus, interface1.controlBus);
connect(inertia1.flange_b,AIMC. flange_a);
connect(mBox1.pConverter, dcac1.pLoad);
connect(mBox1.encoder,AIMC. flange_a);
connect(ground3.p, batteryIdeal1.n);
connect(dcac2.ground, ground2.p);
connect(dcac2.pSupply,batteryIdeal2. p);
connect(dcac2.nSupply,batteryIdeal2. n);
connect(mBox_trck.pMachine, terminalBox1.plugToGrid);
connect(terminalBox1.positiveMachinePlug, AIMCTRCK.plug_sp);
connect(mBox_trck.sedBusAIMC, vdc2.sedBusAIMC);
connect(mBox_trck.sedBusAIMC, foc_trck.sedBusAIMC);
connect(interface2.sedBusAIMC,vdc2. sedBusAIMC);
connect(terminalBox1.negativeMachinePlug, AIMCTRCK.plug_sn);
connect(dcac2.vDC,vdc2. u);
connect(foc_trck.vRef, dcac2.vRef);
connect(tauref2.controlBus,interface2. controlBus);
connect(inertia2.flange_b, AIMCTRCK.flange_a);
connect(mBox_trck.pConverter, dcac2.pLoad);
connect(mBox_trck.encoder, AIMCTRCK.flange_a);
connect(ground4.p,batteryIdeal2. n);
connect(fan1.flange, inertia1.flange_a);
connect(inertia2.flange_a,fan2. flange);
connect(mBox_trck.sedBusAIMC, currentModel_trck.sedBusAIMC1);
connect(voltageModel_trck.sedBusAIMC, mBox_trck.sedBusAIMC);
connect(temp.sedBusAIMC, mBox_trck.sedBusAIMC);
connect(tracking.sedBusAIMC, mBox_trck.sedBusAIMC);
connect(timeTable1.y, tauref1.u);
connect(timeTable2.y, tauref2.u);

end TrackingTest_FOC;
```