

HANSER



Leseprobe

zu

Modelica – Mit GitHub-Tutorial

Objektorientierte Modellbildung von Drehfeldmaschinen

von Christian Kral

ISBN (Buch): 978-3-446-45551-1

ISBN (E-Book): 978-3-446-45733-1

Weitere Informationen und Bestellungen unter

<http://www.hanser-fachbuch.de/978-3-446-45551-1>

sowie im Buchhandel

© Carl Hanser Verlag, München

Vorwort

Das vorliegende Buch behandelt die objektorientierte Modellbildung elektrischer Maschinen mit Modelica. Dazu gehören viele unterschiedliche Aspekte: Einmal geht es um die zugrunde liegende Physik und die Gleichungen der elektrischen Maschinen, also um ein elektrotechnisches Verständnis. Dann geht es um Modelica, eine Modellierungssprache, mit der man physikalische Gleichungen auf akausale Weise formulieren kann. Weiters geht es um die verwendete Software, OpenModelica, die quelloffen zur Verfügung steht. Hier gilt es zu lernen, wie man Modelle entwickelt, simuliert und analysiert, wobei alle in diesem Buch verwendeten Modelica-Beispiele quelloffen und kostenfrei verfügbar sind. Zusätzlich geht es um die Organisation der Entwicklung von Modelica-Code. Dafür wird auf die Online-Plattform GitHub und die Software GitKraken zurückgegriffen, die bei nicht kommerzieller Nutzung ebenfalls kostenfrei sind. Dieses Buch ist damit nicht nur ein Buch über die Modellbildung elektrischer Maschinen, sondern eines, indem sehr unterschiedliche Themen ineinandergreifen und voneinander abhängig sind. Damit trägt dieses Buch auch dem Umstand Rechnung, dass es zunehmend wichtiger wird, ein systemisches Verständnis zu erlangen; dies beinhaltet das technische Fachverständnis ebenso wie das Verständnis für die Organisation der verwendeten Daten und der zugrunde liegenden Software.

Die Verwobenheit der Themen dieses Buchs spiegelt sich auch in seiner Sprache wider, in der viele englische Fachbegriffe aus der Informationstechnologie und der Semantik von Modelica Einzug gefunden haben. Diese Fachbegriffe wurden absichtlich nicht eingedeutscht, um möglichst unmissverständlich die originale Bedeutung wiederzugeben.

Dieses Buch richtet sich an Studierende, Ingenieure und Lehrende der Elektrotechnik, die an einer über das klassische Lehrbuchwissen hinausgehenden Modellierung elektrischer Maschinen interessiert sind. Es ist so strukturiert, dass es den Leserinnen und Lesern die Möglichkeit bietet, sich bezüglich unterschiedlicher Aspekte zu vertiefen oder sich stärker an den Anwendungen zu orientieren. Im Buch sind entsprechende Markierungen gesetzt, die zum Weiterlesen an unterschiedlichen Stellen einladen.

Leserinnen und Leser, die mit ihren elektrotechnischen Grundkenntnissen etwas über die Modellbildung und Simulation elektrotechnischer Systeme lernen wollen, erhalten in Kapitel 2 eine Einführung in die Modellierungssprache Modelica und die Software OpenModelica. Optional sind in Kapitel 1 die dafür erforderlichen physikalischen Grundlagen knapp zusammengefasst. Kapitel 3 vertieft das Basiswissen zu Modelica anhand konkreter Anwendungen. Theoretisches Hintergrundwissen und praktische Anwendungen werden dabei abwechselnd vermittelt. Hinsichtlich elektrischer Systeme werden transiente als auch eingeschwungene Modelle untersucht. Selbst einfache leistungselektronische Modelle werden präsentiert. Weiters werden auch physikalische Kopplungen elektrischer Systeme mit magnetischen, thermischen und mechanischen Teilsystemen entwickelt, erläutert und simuliert.

Wer sich in die objektorientierte, physikalische Modellierung elektrischer Maschinen vertiefen möchte, findet in den Kapiteln 4–6 das zugehörige Material. In diesen Kapiteln werden die Modelle der elektromagnetischen Kopplungen der Wicklungen, der unterschiedlichen Verlus-

te, des Luftspalts und der Besonderheiten der untersuchten Drehfeldmaschinen erläutert. Dabei werden Asynchronmaschinen mit Käfig- oder Schleifringläufer sowie Synchronmaschinen mit elektrischer oder Permanentmagneterregung als auch Synchronreluktanzmaschinen detailliert behandelt. Weiters wird die Verbindung zur klassischen Theorie der Raumzeiger hergestellt und die Parametrierung der Maschinenmodelle aufgezeigt. Für die Vertiefung in Kapitel 4 sind Grundkenntnisse elektrischer Maschinen erforderlich.

Für all jene, die das Betriebsverhalten elektrischer Drehfeldmaschinen anhand von konkreten Anwendungen besser verstehen wollen, bieten die Abschnitte 5.4 und 5.5 für Asynchronmaschinen sowie die Abschnitte 6.4–6.6 für Synchronmaschinen viele Simulationsbeispiele. In diesen Abschnitten werden sowohl industrielle Anwendungsfälle als auch Laborexperimente virtuell nachgestellt und analysiert. Damit lassen sich viele aus der klassischen Literatur elektrischer Maschinen bekannte Kennlinien in Simulationsmodellen generieren und nachvollziehen. Aufgrund der Objektorientierung von Modelica können die Leserinnen und Leser die zugrunde liegenden physikalischen Zusammenhänge und Variablen einfach visualisieren und analysieren.

Dieses Buch enthält außerdem ein Tutorial, das die versionierte Organisation, Entwicklung und Wartung von Modelica-Libraries mit GitHub erläutert. Da quelloffene Software auf GitHub kostenfrei verwaltet werden kann, bietet sich diese Plattform auch für quelloffene Modelica-Libraries an, wie sie auch in diesem Buch verwendet werden. Mit der für nicht kommerzielle Nutzung kostenfreien Software GitKraken lassen sich GitHub-Projekte auf dem eigenen Rechner einfach und übersichtlich verwalten. Das dafür erforderliche Handwerkszeug wird in seinen Grundzügen in Kapitel 7 vermittelt.

Mein ganz besonderes Augenmerk liegt auf der quelloffenen und kostenfreien Nutzung der verwendeten Modelica-Libraries und der dafür erforderlichen Simulationssoftware OpenModelica. In diesem Sinne sind alle in diesem Buch verwendeten Software-Programme und Simulationsmodelle kostenfrei nutzbar. So wie ich von quelloffenen Projekten profitiert habe, werden hoffentlich auch die Leserinnen und Leser profitieren. Mit etwas Glück liefert dieses Buch neue Ideen, neue Ansätze und neue Zugänge und ermöglicht vielleicht auch den Aufbau neuer Modelica-Libraries.

Mein großer Dank gilt Anton Haumer. Er hat die Ideen und Gedanken zu diesem Buch bis zur Fertigstellung begleitet. Gemeinsam haben wir viel diskutiert und viele Zeilen Modelica-Code entwickelt und durchdacht. Von ihm stammen auch viele der ursprünglich entwickelten Simulationsmodelle über elektrische Maschinen. Er ist mein bester Kritiker und guter Freund. Ich freue mich außerdem über die Beiträge von Michael Hochstöger und Beatrix Mastal, die zum guten Gelingen dieses Buches beigetragen haben. Bei allen, für die ich während der Entstehung dieses Buchs zu wenig Zeit hatte, bedanke ich mich für ihre Nachsicht. Meinem Lektor Manuel Leppert und Franziska Kaufmann von der Herstellung des Carl Hanser Verlags danke ich für die stets gute und konstruktive Zusammenarbeit.

Lichtenegg, im August 2018

Christian Kral

URL der Internetseite mit den Modelica-Beispielen zum Buch:

<https://github.com/christiankral/HanserModelica>

Gewidmet meinem Lehrer, Förderer und Freund Karl Haidinger, 16.04.1949 bis 21.09.1997.
Durch seine Inspiration und Begeisterung lebt fort sein guter Geist in diesem Buch.

Inhalt

Nomenklatur	15
1 Grundlagen	19
1.1 Elektrische Kreise	19
1.1.1 Kirchhoffsche Maschenregel	20
1.1.2 Kirchhoffsche Knotenregel	21
1.1.3 Zusammenschaltung von Komponenten	21
1.1.4 Elektrischer Widerstand	22
1.1.5 Spule	22
1.1.6 Kondensator	23
1.2 Magnetische Kreise	23
1.2.1 Magnetische Spannung	23
1.2.2 Magnetischer Fluss	24
1.2.3 Flussverkettung	24
1.2.4 Induktionsgesetz	25
1.2.5 Durchflutungssatz	26
1.2.6 Satz vom magnetischen Hüllenfluss	26
1.2.7 Magnetische Ersatzschaltbilder	27
1.2.8 Magnetischer Widerstand	27
1.2.9 Induktivität	27
1.2.10 Ideal gekoppelter Transformator	28
1.2.11 Permanentmagnet	29
1.2.12 Eisenverluste	30
1.3 Thermische Systeme	31
1.3.1 Wärmestrom und Temperatur	31
1.3.2 Thermischer Widerstand	31
1.3.3 Thermischer Kondensator	32
1.3.4 Aufbau thermischer Ersatzschaltbilder	32
1.4 Rotatorische mechanische Systeme	33
1.4.1 Verdrehwinkel und Drehmoment	33
1.4.2 Elektromagnetisches Drehmoment	33
1.4.3 Massenträgheitsmoment	34

2	Modelica	35
2.1	Simulationstools	38
2.1.1	Verfügbare Software	38
2.1.2	Simulation von Modellen	39
2.1.3	Kompatibilität und FMI	39
2.1.4	OpenModelica	40
2.2	Erste Schritte	41
2.2.1	Erste Implementierung	43
2.2.2	Datei-Handling	46
2.2.3	Zweite Implementierung	48
2.2.4	Simulationsergebnisse	49
2.2.5	Dritte Implementierung	51
2.2.6	Grafische Implementierung	51
2.2.7	Implementierung über Vererbung	57
2.3	Variablen und Datentypen	59
2.3.1	Instantiierung von Variablen	59
2.3.2	Konstante und Parameter	59
2.3.3	Komplexe Zahlen	62
2.3.4	Vektoren und Matrizen	62
2.3.5	Attribute von Variablen	66
2.3.6	Initialisierung von Parametern	68
2.4	Klassen und Konzepte	70
2.4.1	Aufbau von Klassen	70
2.4.2	Öffentliche und private Klassen	71
2.4.3	Initialisierung von Variablen	72
2.4.4	Dokumentation	73
2.4.5	partial	73
2.4.6	class	73
2.4.7	type	74
2.4.8	package	75
2.4.9	model	77
2.4.10	connector	78
2.4.11	Vererbung	80
2.4.12	Grafische Verbindungen	83
2.4.13	function	84
2.4.14	block	85
2.4.15	record	87

3	Modellierungskonzepte	89
3.1	Allgemeine Konzepte	89
3.1.1	Signale	89
3.1.2	Tables	91
3.1.3	Erstellung eigener Komponenten	95
3.1.4	Parameter-Records	98
3.1.5	Konditionale Gleichungen	101
3.1.6	Zusammengesetzte funktionale Zusammenhänge	104
3.1.7	Konditionale Komponenten und Verbindungen	105
3.2	Elektrische Systeme	107
3.2.1	Dioden	108
3.2.2	Drei- und mehrphasige Systeme in Stern- und Polygonschaltung	110
3.2.3	Dioden-Gleichrichter	115
3.2.4	Einphasige, quasistationäre Systeme	116
3.2.5	Ortskurve und Bodediagramm	120
3.2.6	Mehrphasige, quasistationäre Systeme	122
3.3	Magnetische Systeme	123
3.3.1	Magnetischer Konnektor	124
3.3.2	Elektromagnetische Kopplung	125
3.3.3	Transformator	126
3.4	Thermische Systeme	132
3.4.1	Thermischer Konnektor	132
3.4.2	Thermisches Netzwerk	133
3.4.3	Diskretisierte Wärmeleitung	134
3.4.4	Temperaturabhängiger Widerstand	137
3.4.5	Elektrisch-thermische Kopplung	140
3.5	Rotierende mechanische Systeme	140
3.5.1	Rotatorischer Konnektor	140
3.5.2	Einfacher mechanischer Antrieb	142
3.5.3	Elektromechanische Kopplung	143
3.5.4	Einfache Gleichstrommaschine	145
4	Drehfeldmaschinen	149
4.1	Voraussetzungen und Bezugssysteme	149
4.1.1	Zählpeilsysteme	150
4.1.2	Bezugsrichtungen	150
4.1.3	Grundwelle, Pole und Polpaarzahl	151
4.1.4	Räumliche und elektrische Winkel	152

4.2	Magnetische Größen des Grundwellenmodells	153
4.3	Spulen und Wicklungen	157
4.3.1	Spulenindizes im gültigen Wertebereich abbilden.....	161
4.3.2	Beschreibung einer Spulengruppe in Modelica	162
4.4	Elektromagnetische Kopplung	164
4.4.1	Magnetische Spannung der Spulen	164
4.4.2	Magnetische Spannung und Strangstrom.....	165
4.4.3	Berechnung komplexer Windungszahlen in Modelica	167
4.4.4	Magnetische Flussverkettung und induzierte Spannung	169
4.4.5	Elektromagnetisches Kopplungsmodell in Modelica.....	171
4.5	Koordinatensysteme	174
4.6	Luftspaltmodell	175
4.6.1	Magnetische Feldstärke und Flussdichte.....	175
4.6.2	Kehrwert der Luftspaltfunktion.....	176
4.6.3	Rotorfestes Luftspaltmodell bezüglich der Grundwelle	177
4.6.4	Reluktanzen des Luftspaltmodells	178
4.6.5	Drehmomentbildung im Luftspaltmodell.....	180
4.6.6	Luftspaltmodell in Modelica	181
4.7	Eisenverlustmodell	184
4.7.1	Magnetisches Modell der Wirbelstromverluste	184
4.7.2	Vergleich von elektrischem und magnetischem Verlustmodell	186
4.8	Wicklungsmodell	188
4.8.1	Phasenzahlen und Phasensymmetrie	188
4.8.2	Nullinduktivität	189
4.8.3	Wicklungsmodell in Modelica	190
4.9	Käfigmodell.....	193
4.9.1	Symmetrisches Käfigmodell	193
4.9.2	Achsiges Käfigmodell.....	194
4.10	Reibungsmodell	195
4.11	Zusatzverluste	196
4.11.1	Zusatzverluste in Anlehnung an DIN EN 60034-2:1998	197
4.11.2	Zusatzverluste in Anlehnung an IEEE Std 112-2004	198
4.11.3	Effektivwert des Zuleitungsstroms	199
4.11.4	Zusatzverluste in Modelica.....	199
4.12	Permanentmagnet.....	201
4.13	Bürstenübergangsverluste	203
4.14	Maschinenmodelle	204

5	Asynchronmaschinen	211
5.1	Objektorientierte Modelle	211
5.1.1	Asynchronmaschine mit Schleifringläufer	212
5.1.2	Asynchronmaschine mit Kurzschlussläufer	215
5.2	Raumzeigergleichungen	216
5.2.1	Raumzeiger der magnetischen Spannungen und Flüsse	217
5.2.2	Raumzeiger der elektrischen Spannungen und Ströme	219
5.2.3	Rücktransformation von Raumzeigern auf Stranggrößen	221
5.2.4	Induktivitäten	221
5.2.5	Äquivalente Wicklung des Rotors	222
5.2.6	Drehmoment	223
5.2.7	Quasistationäre Gleichungen bei sinusförmigem Betrieb	224
5.3	Parametrierung	224
5.3.1	Parametrierung des Schleifringläufers	225
5.3.2	Parametrierung des Kurzschlussläufers	225
5.4	Simulation von Asynchronmaschinen mit Kurzschlussläufern	226
5.4.1	Quasistationärer Betrieb am Netz	226
5.4.2	Direktanlauf am Netz	230
5.4.3	Stern-Dreieck-Anlauf am Netz	232
5.4.4	Anlauf mit Transformator am Netz	235
5.4.5	Hochlauf am Inverter	235
5.4.6	Vergleich von Simulations- mit Messdaten	238
5.4.7	Einphasiger Betrieb mit Steinmetz-Schaltung	240
5.5	Simulation von Asynchronmaschinen mit Schleifringläufern	243
5.5.1	Quasistationärer Betrieb am Netz	244
5.5.2	Direktanlauf am Netz mit externen Rotorwiderständen	246
6	Synchronmaschinen	249
6.1	Objektorientierte Modelle	250
6.1.1	Synchronmaschine mit elektrischer Erregung	250
6.1.2	Synchronmaschine mit Permanentmagneten	254
6.1.3	Synchronreluktanzmaschine	256
6.2	Raumzeigergleichungen	256
6.2.1	Herleitung	257
6.2.2	Äquivalenter Dämpferkäfig und äquivalenter Erregerkreis	258
6.2.3	Maschine mit Permanentmagneten	260
6.2.4	Quasistationäre Gleichungen bei sinusförmigem Betrieb	260
6.3	Parametrierung	261

6.3.1	Parametrierung der Hauptfeldinduktivität	262
6.3.2	Parametrierung des Dämpferkäfigs	262
6.3.3	Parametrierung der Erregerwicklung	262
6.3.4	Parametrierung der Permanentmagnet-Erregung	263
6.3.5	Parametrierung aus Kenngrößen eines Datenblatts	263
6.4	Simulation von Synchronmaschinen mit elektrischer Erregung.....	267
6.4.1	Direktanlauf am Netz	267
6.4.2	Synchronisation mit dem Netz	270
6.4.3	Variabler Polradwinkel am Netz	271
6.4.4	Regulierkennlinien am Netz	276
6.4.5	Belastungskennlinien im Inselbetrieb	278
6.4.6	Lastabwurf im Inselbetrieb mit Spannungsregelung	279
6.4.7	Inselbetrieb mit Gleichrichter und Spannungsregelung	279
6.4.8	Stoßkurzschluss	283
6.5	Simulation von Synchronmaschinen mit Permanentmagneten	284
6.5.1	Betrieb bei variablem Stromwinkel	285
6.5.2	Ideale Speisung mit Stromquelle	289
6.6	Simulation von Synchronreluktanzmaschinen	290
6.6.1	Hochlauf am Umrichter mit Dämpferkäfig	291
6.6.2	Betrieb bei variablem Stromwinkel	294
7	GitHub-Tutorial	295
7.1	Erstellung eines Repositories	296
7.2	GitKraken	300
7.3	Klonen eines Repositories	301
7.4	Entwicklung und Wartung einer Modelica-Library	303
7.4.1	Hinzufügen von Dateien	303
7.4.2	Push und Pull	306
7.4.3	Löschen von Dateien	307
7.4.4	Löschen von Verzeichnissen	308
7.4.5	Wiederherstellen von versehentlich gelöschten Dateien	308
7.4.6	Konsistenz der Commitments	309
7.5	Arbeiten mit Branches	309
7.5.1	Branch erstellen und zusammenführen	310
7.5.2	Entwicklungsstand wiederherstellen	311
7.6	Issue-Tracking.....	311
7.7	Erstellung einer Release-Version	313
7.7.1	Anpassung der Modelica-Library.....	314

7.7.2	Version im Repository.....	315
7.8	Zusammenarbeit und Weiterentwicklung	316
7.8.1	Einladung zur Zusammenarbeit.....	316
7.8.2	Fork.....	317
7.8.3	Pull-Request.....	317
A	Modelica Standard Library	319
A.1	 Modelica.Blocks.....	319
A.2	 Modelica.Electrical.Analog.....	320
A.3	 Modelica.Electrical.Machines.....	321
A.4	 Modelica.Electrical.MultiPhase	322
A.5	 Modelica.Electrical.QuasiStationary	322
A.6	 Modelica.Magnetic.FluxTubes	323
A.7	 Modelica.Magnetic.FundamentalWave.....	323
A.8	 Modelica.Magnetic.QuasiStatic.FundamentalWave	324
A.9	 Modelica.Mechanics.Rotational	324
A.10	 Modelica.Mechanics.Translational	325
A.11	 Modelica.Thermal.FluidHeatFlow	325
A.12	 Modelica.Thermal.HeatTransfer	326
B	Formelzeichen	327
B.1	Variablen.....	327
B.2	Erster Index.....	329
B.3	Zweiter und dritter Index.....	330
B.4	Hochgestellte Symbole	331
	Literatur.....	333
	Glossar.....	337
	Index.....	341

2

Modelica

Modelica ist eine objektorientierte Modellierungssprache für physikalische Systeme, die über algebraische und gewöhnliche Differenzialgleichungen beschrieben werden können. Durch die Objektorientierung ist man in der Lage, erstellte Modelle – oder genauer gesagt Klassen – modular und übersichtlich wiederzuverwenden sowie redundanten Code zu vermeiden. Dadurch wird eine strukturierte und wenig fehleranfällige Entwicklung von physikalischen Modellen ermöglicht. Beispielsweise wird das Modell eines ohmschen Widerstands nur einmal erstellt. Beim Aufbau von verschiedenen Experimenten lassen sich diese ohmschen Widerstände dann entsprechend wiederverwenden ohne das ohmsche Gesetz wiederholt formulieren zu müssen.

Im Vergleich zu konventionellen Programmiersprachen zeichnet sich Modelica durch akausale Modellierung von Systemen aus. Die entwickelten Modelle werden also nicht nach dem Prinzip von Ursache und Wirkung modelliert, sondern auf Basis physikalischer Gesetze und Zusammenhänge. Erst die Verwendung von Modellen in einem Experiment legt aufgrund der Randbedingungen fest, welche Variablen bekannt und welche unbekannt sind. So ist die Gleichung $v = R * i$ etwa gleichwertig zu $v / R = i$ oder $R = v / i$. Das Gleichheitszeichen beschreibt in Modelica daher eine mathematische Gleichheit und keine Zuweisung. Klassische Zuweisungen im Geiste konventioneller Programmiersprachen mit geordneter Abfolge von Anweisungen werden in Modelica ausschließlich in sogenannten Algorithmen verwendet. Derartige Algorithmen kommen bei der Modellierung nur in Ausnahmefällen vor, etwa bei der Bestimmung von Anfangsbedingungen oder der Formulierung von Funktionen. Eine Zuweisung in einem Algorithmus wird beispielsweise durch `s := a + 3` durchgeführt, wobei `s` bestimmt wird und `a` an dieser Stelle bereits bekannt sein muss.

Das gemeinsame Gleichungssystem, bestehend aus algebraischen Gleichungen und Differenzialgleichungssystemen, wird als *differential algebraic equations* (DAE) bezeichnet. Gleichungen können einerseits direkt in textueller Form als algebraische und gewöhnliche Differenzialgleichungen angeschrieben werden. Andererseits lassen sich Modelle auch grafisch miteinander verschalten, indem die entsprechenden grafischen Anschlüsse von Komponenten miteinander verbunden werden. Zu diesem Zweck stellt ein Simulationstool in der Regel einen grafischen Editor zur Verfügung, der die entsprechende grafische Verschaltung der Komponenten ermöglicht.

Modelica ist objektorientiert. Jedes physikalische Modell in Modelica ist ein Objekt. Jedes Objekt wird entsprechend einer in Modelica verfügbaren Klasse definiert. Es gibt unterschiedliche Klassen wie `model` (Modell), `function` (Funktion) oder `package` (Paket, fortan als Package bezeichnet). Jede Klasse hat bestimmte Einschränkungen und Erweiterungen gegenüber der allgemeinsten Klasse `class`. Ein Objekt der Klasse `model` darf etwa Gleichungen enthalten. Im Gegensatz dazu sind in der Klasse `package` beispielsweise keine Gleichungen erlaubt. Dafür dürfen in einem Package weitere Klassen, also auch weitere Packages enthalten sein. Ein Package ist vergleichbar mit einem Verzeichnis in einem Dateisystem, das als Container für weitere Packages oder Modelle dient. Befindet sich beispielsweise das Objekt eines ohmschen Widerstands

Resistor in einem Package `Basic`, welches sich wiederum in einem Package `Analog` befindet, so wird im hierarchischen Sinne auf das Objekt des Widerstands mittels `Analog.Basic.Resistor` Bezug genommen.

Eine abgeschlossene Sammlung von physikalischen Objekten, die allesamt in einem Package enthalten sind, bezeichnet man in Modelica als Library. Der Begriff Library wird in diesem Buch einheitlich anstelle des deutschen Begriffs Bibliothek verwendet.



Die *Modelica Association* ist der Trägerverein von Modelica, der sich im Wesentlichen zu zwei Aufgaben verpflichtet hat:

- Wartung und Weiterentwicklung des Modelica-Sprachstandards
- Wartung und Weiterentwicklung der Modelica Standard Library

Der Modelica-Sprachstandard ist mit Versionsnummern versehen. Alle Sprachelemente dieses Buchs beziehen sich auf den Modelica-Sprachstandard 3.2, Revision 2 [Mod13]. Zu jedem Sprachstandard passend gibt es auch eine sogenannte Modelica Standard Library, oft mit MSL abgekürzt. Es handelt sich dabei um eine Basis-Library von grundlegenden physikalischen Datentypen und Modellen für unterschiedliche physikalische Bereiche. Dieses Buch bezieht sich auf die Version 3.2.2 in der Revision 3 der Modelica Standard Library.

Die Modelica Standard Library ist als Package  `Modelica` verfügbar. Das Symbol  kennzeichnet dabei, dass nachstehend ein Klassenname angeführt ist. Über Punkte ist wiederum die hierarchische Struktur der Packages innerhalb einer Library gegliedert. Nachfolgend sollen einige ausgewählte Packages der Modelica Standard Library kurz beschrieben werden, damit die Leserinnen und der Leser einen Eindruck über die Vielfalt dieser Library bekommen. Speichert man die Library  `Modelica` als eine zusammenhängende Datei ab, erhält man insgesamt mehr als eine viertel Million Code-Zeilen.

 `Modelica.Blocks` Signalorientierte Blöcke einschließlich Regelungstechnik

 `Modelica.ComplexBlocks` Blöcke mit komplexwertigen Signalen

 `Modelica.StateGraph` Zustandsgraphen

 `Modelica.Electrical.Analog` Einphasige elektrische Systeme

 `Modelica.Electrical.Digital` Digitalschaltungen

 `Modelica.Electrical.Machines` Elektrische Maschinen

 `Modelica.Electrical.MultiPhase` Mehrphasige elektrische Systeme

 `Modelica.Electrical.PowerConverters` Leistungselektronik

 `Modelica.Electrical.QuasiStationary` Eingeschwungene sinusförmige elektrische Systeme die über komplexe Zeitzeigerrechnung behandelt werden

 `Modelica.Electrical.Spice3` Spice3-Modelle der Elektronik

 `Modelica.Magnetic.FluxTubes` Magnetische Kreise basierend auf der Vorstellung von Flussröhren

 `Modelica.Magnetic.FundamentalWave` Mehrphasige, rotierende elektrische Maschinen

 `Modelica.Magnetic.QuasiStatic.FundamentalWave` Eingeschwungene, sinusförmige, mehrphasige, rotierende elektrische Maschinen, die über komplexe Zeitzeigerrechnung behandelt werden

- ☞ `Modelica.Mechanics.MultiBody` Dreidimensionale Mehrkörper-Systeme
- ☞ `Modelica.Mechanics.Rotational` Mechanische Systeme mit einer Rotationsachse
- ☞ `Modelica.Mechanics.Translational` Mechanische Systeme mit einer translatorischen Bewegungsrichtung
- ☞ `Modelica.Fluid` Thermodynamik
- ☞ `Modelica.Media` Medienmodelle für die Thermodynamik
- ☞ `Modelica.Thermal.FluidHeatFlow` Vereinfachte thermodynamische Modelle für die Kühlung von Leistungselektronik und elektrischen Maschinen
- ☞ `Modelica.Thermal.HeatTransfer` Wärmeleitung und Konvektion
- ☞ `Modelica.Math` Mathematische Funktionen einschließlich, Vektoren, Matrizen, Signalverarbeitung
- ☞ `Modelica.ComplexMath` Mathematische Funktionen mit komplexwertigen Argumenten
- ☞ `Modelica.Utilities` Hilfsmittel für die Behandlung von Dateien und das Skripting
- ☞ `Modelica.Constants` Mathematische und naturwissenschaftliche Konstanten
- ☞ `Modelica.Icons` Häufig gebrauchte grafische Icons
- ☞ `Modelica.SIunits` Wichtigste Definition von reell- und komplexwertigen SI-Einheiten physikalischer Größen

Eine detailliertere Beschreibung der Modelica Standard Library ist im Anhang A angegeben, wo die für dieses Buch relevanten Packages etwas genauer beschrieben sind.

Die Modelica Standard Library ist quelloffen (Open Source) unter einer Lizenz veröffentlicht, die die Modifikation und weitere Verwendung in praktisch unbeschränktem Ausmaß gestattet. Als weiterführende Literatur bezüglich Modelica seien [Fri15] sowie das freie Online-Buch [Til17] empfohlen, welche eine gute Einführung in Modelica bieten. Der Sprachstandard selbst und weitere freie Libraries sind auf der Homepage der Modelica Association unter <https://www.modelica.org> zu finden.



Die wesentlichen Vorteile von Modelica gegenüber anderen Modellierungssprachen lassen sich wie folgt zusammenfassen:

Offener Standard. Die Sprachspezifikation von Modelica ist frei zugänglich und wird in einem demokratischen Prozess gewartet und weiterentwickelt.

Physikalische Größen. Modelica rechnet nicht nur mit Zahlenwerten, sondern auch mit physikalischen Einheiten.

Akausalität. In Modelica formuliert man Zusammenhänge über physikalische Gleichungen, nicht über Berechnungsalgorithmen; was bekannte und unbekannte Größen in einem physikalischen System sind, wird daher durch die Randbedingungen eines durchgeführten Experiments bestimmt.

Objektorientierung. Damit sind die entwickelten Komponenten in hohem Grade wiederverwendbar und universell einsetzbar.

Gleichungsbehandlung. Ein Simulationstool führt in der Regel eine analytische Vorbehandlung von Gleichungen aus; mögliche analytische Vereinfachungen und Lösungen von algebraischen Gleichungssystemen werden automatisch im Hintergrund durchgeführt.

Modelica Standard Library. Eine umfangreiche Sammlung von Modellen, die quelloffen zur Verfügung steht.

Simulationstools. Es gibt kommerzielle und freie Simulationstools für Modelica.

Interoperabilität. Modelica-Modelle können über sogenannte *Functional Mock-up Interfaces* mit anderen Tools ausgetauscht werden (siehe Abschnitt 2.1.3).

■ 2.1 Simulationstools

Modelica ist eine Sprache für die Simulation von physikalischen Systemen. Ein Modelica-Modell ist letztlich nichts anderes als ein Stück Code, das mit einem Editor bearbeitet werden kann. Für die Simulation eines Experiments benötigt man jedoch ein Modelica-Simulationstool, also Software. Eine Liste verfügbarer Simulationstools ist auf der Homepage von Modelica <https://www.modelica.org/tools> verfügbar.

2.1.1 Verfügbare Software

Es gibt kommerzielle und quelloffene (Open-Source) Simulationstools. Auf die kommerziellen Tools wird im Rahmen dieses Buchs nicht eingegangen. Bezüglich der quelloffenen Software sollen hier kurz die beiden am weitesten entwickelten Simulationstools beschrieben werden:

OpenModelica wird von der Linköping University (Schweden) und dem Open Source Modelica Consortium entwickelt und ist unter <https://openmodelica.org/> verfügbar. OpenModelica verfügt über ein grafisches Benutzerinterface, das die textuelle und grafische Entwicklung von Modellen und Simulationsexperimenten unterstützt. Zusätzlich können Simulationsergebnisse grafisch analysiert und exportiert werden. Die meisten Simulationsexperimente der Modelica Standard Library können mit OpenModelica simuliert werden. Das ist ein Indikator dafür, dass OpenModelica den aktuellen Modelica-Sprachstandard 3.2 gut implementiert hat. Daher ist diese Software für viele Analysen im schulischen, akademischen und industriellen Umfeld gut geeignet. Konkrete Arbeitsschritte für die Modellierung und Simulation werden in diesem Buch anhand von OpenModelica erläutert. Alle für dieses Buch simulierten Modelle laufen mit OpenModelica.

JModelica wird von Modelon AB (Schweden) entwickelt und kann auf <http://www.jmodelica.org/> heruntergeladen werden. Diese Software verfügt über kein eigenes grafisches Benutzerinterface. Derzeit gibt es zumindest ein kommerzielles Simulationstool, das auf JModelica basiert. Mit JModelica kann man Modelica-Modelle numerisch simulieren und grafisch analysieren. JModelica ist im Wesentlichen in die Programmiersprache Python eingebunden, was JModelica zu einem insgesamt sehr mächtigen Werkzeug macht, da die numerischen Fähigkeiten von Python in vollem Umfang zusätzlich genutzt werden können. JModelica ist daher sehr gut geeignet, um Simulationen eingebettet in Python-Code laufen zu lassen.

2.1.2 Simulation von Modellen

Ein simulierbares Modell muss gleich viele Gleichungen wie Unbekannte aufweisen. Ein solches Modell wird in Modelica als *Simulationsmodell* bezeichnet. Um eine Simulation durchführen zu können, muss der Modelica-Code des Simulationsmodells übersetzt werden. Dazu wird vom verwendeten Tool zunächst *Flat Modelica Code* erzeugt. Das bedeutet, dass alle Objekte durch den vollen ihnen zugrunde liegenden Modelica-Code ersetzt werden. Im nächsten Schritt werden dann alle Gleichungen auf Basis der Kausalitäten sortiert und triviale Gleichungen eliminiert. Solche trivialen Gleichungen eines Simulationsmodells sind beispielsweise Gleichungen, die sich aus der Gleichheit zweier Variablen ergeben. Der Verweis auf die Gleichung bleibt für das Ergebnis erhalten, während die Gleichung hinsichtlich der Lösung des Systems eliminiert wird.

Die Gleichungen selbst werden, sofern es möglich ist, vom Simulationstool analytisch umgeformt und vereinfacht. Gleichungsstrukturen, die keine explizite Lösung ermöglichen, werden so aufbereitet, dass sie während der Laufzeit numerisch gelöst werden können. Ziel ist es in diesem Zusammenhang, die Anzahl der numerischen Operationen und Iterationen während der Laufzeit so gering wie möglich zu halten. Zusätzlich dazu findet eine sogenannte Index-Reduktion statt. Da nicht jede Variable eines Differenzialgleichungssystems notwendigerweise auch eine Zustandsgröße sein muss, können bestimmte Zustandsgrößen eliminiert werden. Die Serienschaltung von zwei Spulen kann so beispielsweise durch Index-Reduktion zu einer äquivalenten Ersatzspule zusammengefasst werden. Dieser Vorgang findet automatisiert im Hintergrund statt.

Nachdem der Übersetzungsvorgang abgeschlossen ist, kann der Solver die numerische Lösung eines Simulationsmodells bestimmen. Diese numerische Lösung beinhaltet sowohl die Lösung des algebraischen Teils als auch des Differenzialgleichungssystems. Bezüglich der numerischen Lösung müssen vorher jedoch bestimmte Bedingungen spezifiziert werden; diese sind in der Regel:

- Anfangszeit der Simulation, meist Null
- Endzeit der Simulation
- Integrationsverfahren
- Explizite Angabe der Schrittweite des Integrationsverfahrens, wenn keine automatische Schrittweitensteuerung vorgesehen ist
- Genauigkeitsschranke des Integrationsverfahrens
- Anzahl der Ausgabepunkte oder das Ausgaberaaster der Ergebnisdateien
- Variablen, die in einer Ergebnisdatei für eine Analyse gespeichert werden sollen

2.1.3 Kompatibilität und FMI

Jedes Modelica-Modell läuft auf jedem beliebigen Modelica-Tool, sofern

- das Modelica-Modell den Richtlinien der Modelica-Spezifikation entspricht und
- das Simulationstool die Modelica-Spezifikation implementiert hat.

Da die Simulationstools in der Regel keine inkompatiblen Features und keine absichtlich inkompatible Code-Generierung implementiert haben, ist der Austausch von Modelica-Code

meist kein Problem. Mitunter sind jedoch einige besondere Sprachkonstrukte der Modelica-Spezifikation nicht in allen Tools umgesetzt. In einfacheren Fällen gibt es in Modelica dann eine alternative Formulierung eines Modells, die dann vom jeweiligen Tool verarbeitet werden kann.

Modelica-Dateien sind nach UTF-8 kodiert und im Wesentlichen Text-Dateien, die mit einem Editor geöffnet und verändert werden können. Alle externen Ressourcen wie etwa Bilder werden über Hyperlinks in HTML in den Modelica-Code eingebunden.

Eine Liste freier Modelica-Editoren findet man auf <https://modelica.org/tools/>. Aus Sicht des Autors dieses Buchs ist der Editor Atom sehr empfehlenswert, für den ein Modelica-Plugin zur Syntaxhervorhebung installiert werden kann. Atom ist unter <https://atom.io/> für Linux, Mac OS X und Windows verfügbar.

Modelica alleine löst nicht alle Probleme. Für spezialisierte Aufgabenstellungen benötigt man spezialisierte Tools. Daher wurden *Functional Mock-up Interfaces* (FMIs) als toolunabhängiger Standard geschaffen, um sowohl den Austausch über Co-Simulation als auch den Austausch von Modellen (engl. *model exchange*) für transiente Simulationen zu ermöglichen. Das erlaubt einerseits die Kopplung unterschiedlicher Softwaretools. Man erzeugt in einem Tool sogenannte *Functional Mock-up Units* (FMUs), die dann den entsprechenden Austausch ermöglichen. Bei der *Co-Simulation* werden zu festen Zeitschritten Informationen zwischen den unterschiedlichen Tools ausgetauscht, beim Austausch von Modellen werden ganze Modelle – einschließlich Solver – ausgetauscht. Andererseits ermöglichen es die *Functional Mock-up Units* eigene Entwicklungen zu schützen, da diese in binärer Form ausgetauscht werden, wodurch die Implementierung der Modelle nicht einsehbar ist. Durch die *Functional Mock-up Interfaces* eröffnet sich ein sehr breites Feld von Anwendungsmöglichkeiten. Derzeit sind auf <http://fmi-standard.org> weit mehr als 100 Tools gelistet, die diesen Standard bereits unterstützen oder deren Entwickler angekündigt haben, ihn zu unterstützen.

2.1.4 OpenModelica

OpenModelica ist ein quelloffenes Simulationstool für Modelica, der auf <https://openmodelica.org/> für Linux, Mac OS X und Windows verfügbar ist. Die Installationsanleitungen befinden sich direkt auf der Homepage. Im vorliegenden Buch wird auf die Version 1.13.0 von OpenModelica (engl. *nightly build*) mit englischem Benutzerinterface Bezug genommen, siehe Bild 2.1, da die englische Version im Hinblick auf den Gebrauch von Modelica konsistenter und insgesamt logischer als die deutsche Implementierung ist. Die Umstellung auf Englisch erfolgt über das Menü ▶ Tools ▶ Optionen ▶ Allgemeine ▶ Sprache ▶ English (en).

In den nachfolgenden Abschnitten werden zunächst die ersten Schritte mit OpenModelica erklärt. Dabei wird auch konkret auf die erforderlichen Arbeitsschritte Bezug genommen, die notwendig sind, um Modelle zu erstellen, zu organisieren, zu übersetzen und zu analysieren. Weiterführende Erläuterungen zu OpenModelica finden sich im User's Guide auf <https://www.openmodelica.org/doc/OpenModelicaUsersGuide/v1.13.0/>.

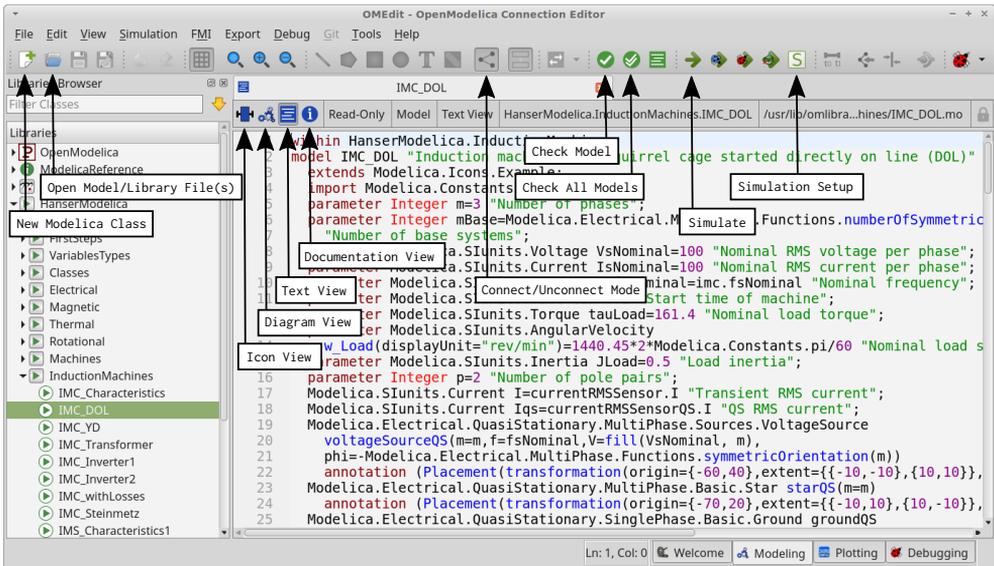


Bild 2.1 OpenModelica Editor

2.2 Erste Schritte

Nach dem Start von OpenModelica werden einige Libraries automatisch geladen. Die für die praktische Arbeit relevanten Libraries sind:

- Modelica Standard Library:  Modelica
- Kurzreferenz der Syntax bzw. Befehlsreferenz zur Sprache Modelica:  ModelicaReference
- Komplexe Zahlen in Modelica (in manchen Versionen von OpenModelica wird diese Library ausgeblendet):  Complex

Zusätzlich wird die Library  OpenModelica geladen. Dabei handelt es sich um eine interne Library, die für die Anwenderinnen und Anwender in der Regel nicht relevant ist.

Alle in diesem Buch erläuterten Beispiele stammen entweder aus der Modelica Standard Library, deren Modelle auszugsweise im Anhang A detaillierter erläutert sind, oder aus der Library  HanserModelica, die auf <https://github.com/christiankral/HanserModelica> quelloffen zur Verfügung steht. Die Library  HanserModelica besteht aus vielen eigens für dieses Buch entwickelten Simulationsmodellen, übernimmt und modifiziert jedoch auch viele Beispiele aus der Modelica Standard Library. Die Library  HanserModelica ist in OpenModelica als System-Library integriert, siehe Bild 2.2, und kann durch einen Klick auf **File** **System Libraries** **HanserModelica** geladen werden.

In diesem Abschnitt sollen mehrere Varianten eines ersten Simulationsmodells untersucht werden. Als Beispiel dient die Serienschaltung eines ohmschen Widerstands und einer Spule, die an einer Gleichspannung eingeschaltet werden soll, siehe Bild 2.3a. Aufgrund der Serienschaltung ist der Strom i durch alle Komponenten gleich groß. Für den ohmschen Widerstand R gilt die algebraische Gleichung

$$v_R = R \cdot i. \quad (2.1)$$

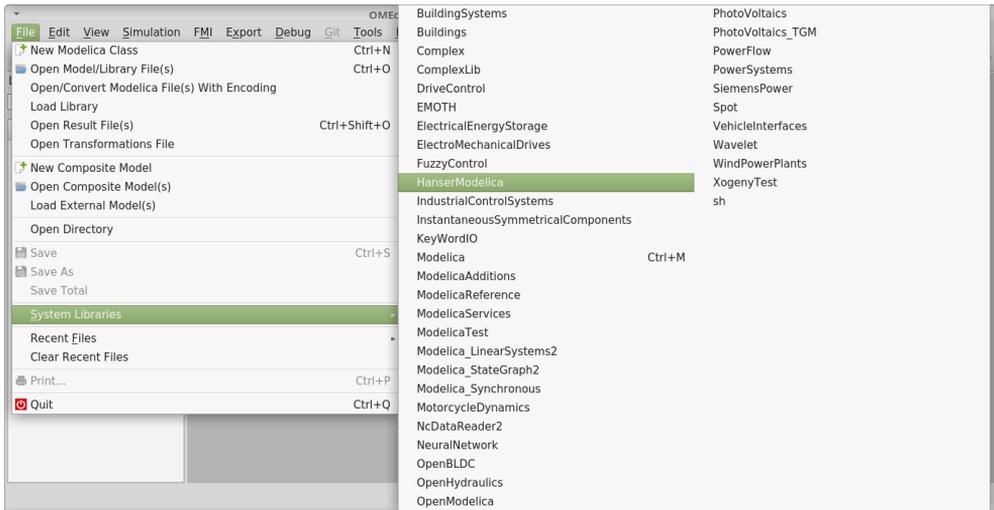


Bild 2.2 Die Library  HanserModelica ist in OpenModelica als System-Library integriert

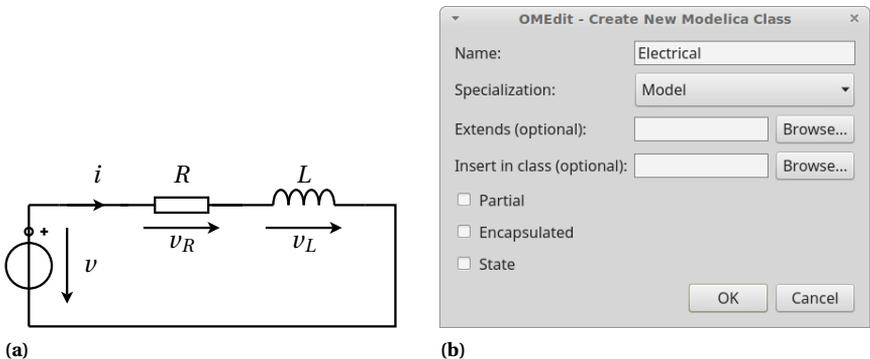


Bild 2.3 (a) Serienschaltung eines R-L-Kreises an einer Gleichspannung; (b) Erstellung eines neuen Modells `Electrical1` über das Menü  File  New Modelica Class

Der Spannungsabfall an der Spule mit der Induktivität L hängt mit dem Strom i über die gewöhnliche Differenzialgleichung

$$v_L = L \cdot \frac{di}{dt} \tag{2.2}$$

zusammen. Zusätzlich gilt die Maschenregel

$$v = v_R + v_L. \tag{2.3}$$

Zum Zeitpunkt $t = 0$ soll der Strom $i = 0$ sein.

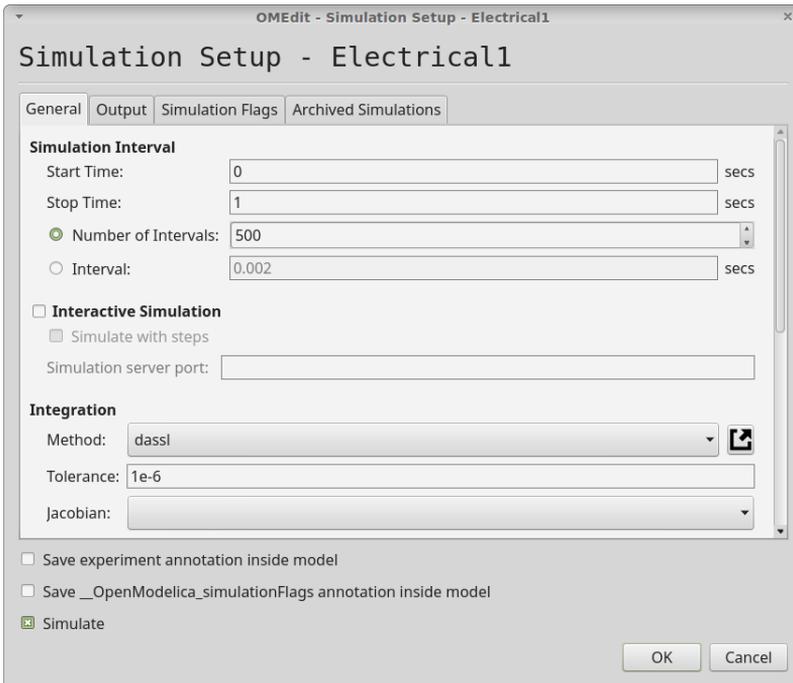


Bild 2.4 Simulationseinstellungen des Modells `Electrical1`

2.2.1 Erste Implementierung

Die Serienschaltung soll in einer ersten Implementierung als Modelica-Modell umgesetzt werden. Zunächst wird ein neues Modell in OpenModelica über das Menü **File** **▶** **New Modelica Class** erstellt, wobei als Klassenname `Electrical1` angegeben werden soll, wie das in Bild 2.3b dargestellt ist. Die Spezialisierung `Model` wird übernommen. Der Quelltext aus Listing 2.1 kann über einen Klick auf **Text View** (Bild 2.1) eingegeben werden. Alternativ zur Eingabe des Codes kann das Modell `HanserModelica.FirstSteps.Electrical1` geöffnet werden.

Ganz allgemein spricht man in Modelica von Klassen, die im Detail in Abschnitt 2.4 behandelt werden. Eine spezialisierte Klasse ist `model`, die sowohl für physikalische Modelle von Komponenten als auch für Simulationsmodelle verwendet wird. Das neu erstellte Modell beginnt mit dem Schlüsselwort `model`, gefolgt vom Modell-Namen `Electrical1` und einer optionalen, unter Anführungszeichen angegebenen Beschreibung. Die erste Zeile wird, im Gegensatz zu Deklarationen und Gleichungen, nicht mit einem Semikolon abgeschlossen. Das Ende des Modells wird mit dem Schlüsselwort `end`, dem Modell-Namen `Electrical1` sowie einem Semikolon abgeschlossen.

Reellwertige Variablen werden mit dem Schlüsselwort `Real` deklariert, danach wird der Variablenname angegeben, gefolgt von einer optionalen Beschreibung, die unter Anführungszeichen gesetzt wird. Die Deklaration wird mit einem Semikolon abgeschlossen. Die Eindeutigkeit aller Klassen- und Variablennamen sowie der Modelica-Schlüsselwörter hängt von der genauen Groß- und Kleinschreibung ab. Die Variable `vR` und die Variable `vr` sind also nicht identisch. Jede verwendete Variable muss in einem Modell einmal deklariert werden. Die in Anfüh-

Listing 2.1 Erste Implementierung `Electrical1` einer R-L-Serienschaltung (siehe auch  `HanserModelica.FirstSteps.Electrical1`)

```

model Electrical1 "First example"
  // Parameters are constant variables
  parameter Real R = 10 "Resistance";
  parameter Real L = 2 "Inductance";
  parameter Real v = 20 "Total DC voltage";
  Real vR "Voltage drop of resistor";
  Real vL "Voltage drop of inductor";
  Real i "Current";
  initial equation
    i = 0;
  equation
    /*
    3 equation
    3 unknowns v,vR,vL
    */
    v = vR + vL;
    vR = R*i;
    vL = L*der(i);
end Electrical1;

```

rungszeichen angegebene Beschreibung dient der Dokumentation der Variable und wird beispielsweise auch bei der Auswahl der Variablen für die grafische Darstellung wieder mit angezeigt. Es empfiehlt sich also, die Beschreibung in Anführungszeichen immer anzugeben, auch wenn nur kleine Modelle erstellt werden. Die Beschreibung in Anführungszeichen ist nicht viel Aufwand, erhöht jedoch die Lesbarkeit des entwickelten Modelica-Codes und erleichtert die Analyse.

Eine mit dem Präfix `parameter` versehene Variable ist während der Laufzeit einer Simulation konstant. Es können jedoch unterschiedliche Simulationsläufe mit unterschiedlichen Parameterwerten durchgeführt werden, ohne dass das Simulationsmodell neu übersetzt werden müsste. Den Parametern `R`, `L` und `v` werden dabei Werte über ein Gleichheitszeichen zugewiesen. Diese Werte nennt man Defaultwerte oder Standardwerte, da eine Simulation diese Werte verwendet, wenn diese nicht explizit geändert werden.

Zusätzlich zu den in Anführungszeichen angegebenen Beschreibungen können in einer Modelica-Klasse auch noch zusätzliche Kommentare eingefügt werden. Ein einzelner Kommentar wird mit zwei Schrägstrichen `//` eingeleitet. Ein mehrzeiliger Kommentar beginnt mit `/*` und endet mit `*/`.

Die Variablen und Parameter, die in Anführungszeichen gesetzten Beschreibungen wie auch die Kommentare und der Modell-Name sind in englischer Sprache gehalten. Für die Entwicklung von Modelica-Modellen ist es vorteilhaft, wenn alles in englischer Sprache formuliert ist. Das mag im Privatbereich vielleicht nicht zwingend erscheinen, doch sobald man in einem produktiven industriellen oder universitären Umfeld tätig ist, gibt es früher oder später immer jemanden, der nicht Deutsch kann. Weiters kann es zu einem bestimmten Zeitpunkt einmal passieren, dass Teile einer Entwicklung Open Source gestellt werden. All diese Argumente sprechen dafür, Entwicklungen mit Modelica und anderer Software in einem technisch-wissenschaftlichen Umfeld stets in englischer Sprache zu erstellen und zu dokumentieren.

Index

A

- abs 61
- Achsigkeit 177, 179, 194
- acos 61
- Akausalität 37
- algorithm 69, 87, 98
- Alias-Variable 55
- Aliasing 77
- Anlauf → Asynchronmaschine, → Synchronmaschine
- annotation 54
 - checkBox 137
 - defaultComponentName 100, 145
 - defaultComponentPrefixes 100
 - Dialog 209
 - Evaluate 137
 - experiment 77
 - HideResult 137
 - uses 315
 - version 314
 - versionBuild 315
 - versionDate 315
- Arbeitsverzeichnis 302
- Array 62
 - Dimension 66
- asin 61
- assert 139
- Asynchronmaschine 211, 321
 - Anlasstransformator 235
 - Anlaufpunkt 228
 - Bremse 228
 - Direktanlauf 230, 246
 - Einphasiger Betrieb 240
 - Generator 228
 - Inverter 235
 - Kippmoment 226, 244
 - Kreisdiagramm 228
 - Netzbetrieb 226, 244
 - Ortskurve 228

- Parametrierung 224
- Stern-Dreieck-Anlauf 232
- Vergleich mit Messdaten 238
- atan, atan2 61
- Attribut 66
- Auflager 141
- auschecken 310
- äußeres Produkt 66
- Avatar 296, 304

B

- Backup 307
- Basissystem 188
- Befehlsreferenz 16, 41
- Bezier 98
- Bezugsimpedanz 263
- Bodediagramm 120
- Branch 309
- Bruchlochwicklung → Wicklung
- Bugfix 315
- Bürsten 105, 203, 250

C

- ceil 61
- checkout 310
- class 73
- clone → Repository
- commit 304
 - Konsistenz 309
- connect 21, 51, 71, 83
 - konditional 105
- connector 51, 78
 - magnetisch 124, 154
 - quasistationär, einphasig 118
 - quasistationär, mehrphasig 122
 - rotatorisch 140, 207
 - thermisch 132, 208, 214

– transient, mehrphasig 110
 constant 59
 cos, cosh 61
 cross 66
 CSV → Dateiformat

D

Dämpferkäftig → Kurzschlusskäftig
 Datei 92
 – hinzufügen 303
 Dateierweiterung 46, 307
 Dateiformat 46, 49, 94
 Datentyp 59
 Defaultwert 44, 59, 60, 77, 84
 Deklarationsabschnitt 55
 Dezimaltrennzeichen 60
 Diagram View 51
 Differenzialgleichung → Gleichung
 Diode 108, 320, 322
 Diskretisierung 134
 displayUnit 67, 74
 div 61
 Division 65
 Documentation View 73
 Dokumentation 73
 Drehfeldleistung 123
 Drehfeldmaschine 149, 204
 Drehmoment 33, 145, 180, 209
 – Asynchronmaschine 223, 224
 – Synchronmaschine 258, 260, 285, 293
 Drehzahl 141
 – synchrone 226
 Dreieckschaltung 112, 157, 189
 dreiphasiges System → mehrphasiges System
 Duplikat erstellen 47
 Durchflutungssatz 26, 125
 Durchmesserwicklung 158

E

each 113
 Effektivwert 113, 199
 Einheit → SI-Einheit
 einphasiges System

– quasistationär 116
 – transient 89
 Eisenverluste 30, 126, 184, 323
 else 101
 elseif 101
 Endring 193, 225
 equation 45, 87, 98
 Erdung → Masse
 Erregerstrom 252, 262, 276
 Erregung 274
 Ersatzmaschine zweipolig 152, 164, 249
 Ersatzschaltbild
 – elektrisch 19
 – magnetisch 27
 – thermisch 32
 Event 104
 exp 61
 Exponent 60
 extends 57, 80

F

Felderregerkurve 158
 feldorientierte Regelung 285
 Feldschwächung 235, 286
 Feldstärke → magnetische Feldstärke
 fill 65
 final 60, 67
 fixed 68
 Fliehkraftschalter 242
 floor 61
 Fluss → magnetischer Fluss
 Flussdichte → magnetische Flussdichte
 Flussgröße 20, 23, 31, 33
 Flussröhre 26
 Flussverkettung → magnetische Flussverkettung
 for 65
 Fork 317
 function 84
 Functional Mock-up
 – Interface 39
 – Unit 39

G

Getriebe 324
 Git 296, 300
 GitHub 295
 .gitignore 298
 GitKraken 300
 Gleichrichter 115, 281
 Gleichstrommaschine 145, 321
 Gleichung 35
 – Differenzialgleichung 35, 45, 70, 134
 – implizit 68
 – konditional 101
 – redundant 119
 Gleitkommazahl 60
 globale Größe 19
 Grundwelle 151, 176

H

HanserModelica 41, 295
 Hash 306
 Hochlauf → Asynchronmaschine, →
 Synchronmaschine
 Host 296
 HTML 73
 Hüllenfluss → magnetischer Hüllenfluss
 Hystereseverluste → Eisenverluste

I

Icon View 56
 identity 65
 if 101
 Impedanz 277
 import 76
 Index 303
 index reduction 72
 Induktionsgesetz 25, 125, 169
 Induktivität 27
 – Asynchronmaschine 220, 221
 – einer Spule 211
 – gegenseitige 218
 – Hauptfeldinduktivität 126, 212, 222
 – Nullinduktivität 213
 – Spule 179, 262
 – Streuinduktivität 126, 190, 212, 215
 – Synchronmaschine 257

initial algorithm 69, 87, 98
 initial equation 45, 87, 98
 Initialisierung 68, 72
 inneres Produkt 65
 input 84, 85, 89
 Inselbetrieb 277, 281
 Instantiierung → Instanz
 Instanz 51, 59, 75, 80
 – vektorisiert 113
 integer 61
 Integrationsverfahren 46
 Interfaces 319
 Inversion
 – Matrix → Matrix
 – Modellinversion 90
 Issue-Tracking 296, 311
 – Klassifizierung 313
 Iteration 69

J

JModelica 38
 Julia 49

K

Käfig → Kurzschlusskäfig
 Kippmoment
 – Asynchronmaschine 226, 244
 – Synchronmaschine 273
 Klasse 70
 Klassendefinition 52
 klonen → Repository
 Koerzitivfeldstärke 29
 Kommentar 44
 Kommutierung 115, 281
 Komplexe Zahl 62
 Kondensator
 – elektrisch 23, 320, 322
 – thermisch 32, 326
 Konnektor → connector
 Konsistenz 309
 Konstante → constant
 Konvektion 326
 Koordinatensystem 174
 – rotorfest 152

- statorfest 152
- statorspannungsfest 224
- Zylinderkoordinatensystem 150
- Kopie → Duplikat erstellen
- Kopplung 320, 323
 - elektrisch-thermisch 137, 140
 - elektromagnetisch 125, 164, 171
 - elektromechanisch 143
 - magnetisches Feld 221
- Kreisdiagramm 120, 123, 228
- Kreisfrequenz 117
- Kurve 98
- Kurzschluss
 - Synchronmaschine 281
- Kurzschlusskäfig 193
 - Asynchronmaschine 215, 225, 226
 - Synchronmaschine 258, 262

L

- Lastabwurf 279
- Lastimpedanz 277
- Leerlaufspannung 254
- Leitwert 322
- Library 36, 46
- Linie 98
- Linienleiter 160
- linspace 65
- Lizenz 298, 314
- Lochzahl 151, 217
- log, log₁₀ 61
- lokale Größe 19
- Luftpalt 175
- Luftpaltfunktion 176
- Luftpaltmodell 177, 181, 205

M

- Magnet → Permanentmagnet
- magnetische Feldstärke 23
- magnetische Flussdichte 23, 156
- magnetische Flussverkettung 24
 - Asynchronmaschine 220
 - Synchronmaschine 258
- magnetische Spannung 23, 153
- magnetischer Fluss 23, 24, 154

- Mantisse 60
- Markdown 311
- Masse 53, 111, 320, 323
- Massenträgheitsmoment 34, 142, 324
- master 304
- Matlab → Dateiformat
- Matrix 62
 - Erzeugung 65
 - Größe 66
 - Inversion 69
 - transponiert 66
- max 66
- mehrphasiges System 157, 188
 - quasistationär 122
 - transient 110
- min 66
- mod 61, 161
- model 77
 - balanced 73
 - erstellen 43
- model exchange 40
- Modelica 35
 - Sprachstandard 36
 - Syntax 16, 41
- Modelica Association 36
- Modelica Standard Library 36, 51, 319
- Modelica-Spezifikation 16
- modelica:// 93
- Modifier 57
- Moment → Drehmoment
- MTPA 285, 293

N

- Namenskonvention 70
- ndims 66
- Nennspannung 213, 252
- Nullinduktivität → Induktivität
- Nullstrom → Strom
- Nuten 151

O

- Oberschicht 157
- Oberwelle 151, 176
- Objekt 51, 70

ones 65
 Open Source 37, 41, 295
 OpenModelica 38, 40
 Operator 65
 origin 315
 Ortskurve 120, 123, 274
 – Asynchronmaschine 228
 output 84, 85, 89, 209

P

package 75
 – erstellen 46
 package.mo 48
 package.order 48
 parallele Zweige 159
 parameter 44, 57, 59, 68, 95, 98
 – Startwert 60
 Parametrierung
 – Asynchronmaschine 224
 – Synchronmaschine 261, 263
 partial 73
 Permanentmagnet 29, 201, 249
 Permeabilität 27, 29
 Permeanz 27
 Phasenverschiebung 113, 188
 Phasenwinkel → Winkel
 Phasenzahl 110, 188
 – Rotor 194, 212
 Plotfenster 49
 Pollücke 249
 Polpaarzahl 151, 249
 Polradspannung → Spannung
 Polradwinkel → Winkel
 Polygonschaltung 157, 189
 Potenzialgröße 20, 23, 31, 33
 Potenzierung 65
 Prüfsumme SHA-1 306, 312
 product 66
 Produkt 65
 Propagation 55
 protected 71
 public 71
 Pull 306
 Pull-Request 317
 Push 306

Python 38, 49

Q

quantity 48, 67, 74
 Quasi-Effektivwert → Effektivwert
 quasistationär 116, 122
 – Raumzeiger 224, 260
 Quelle 89–91, 319

R

Raumzeiger 15, 154, 256
 – Asynchronmaschine 216
 – Rücktransformation 221
 – Synchronmaschine 256
 – Transformation 174, 224
 README 298
 Reaktanz 120, 273, 281
 – bezogen 264
 Reaktionsmoment → Reluktanzmoment
 record 62, 87, 98, 162, 169
 redeclare 180
 Regulierkennlinie → V-Kurve
 Reibung 195, 324
 Reihenfolge Modelica-Klassen 310
 Release 313
 Reluktanz 27, 323
 – Luftspaltmodell 178
 Reluktanzmoment 34, 249, 258
 rem 61
 Remanenzflussdichte 29
 replaceable 179, 208, 214
 Repository 295
 – Branch 309
 – commit 304
 – erstellen 296
 – Fork 317
 – klonen 301
 – Konsistenz 309
 – Pull 306
 – Push 306
 – Tag 315
 – Version erstellen 314
 – Wiederherstellung 311
 – Zusammenarbeit 316

- zusammenführen 306, 310
- Rotorkäfig → Kurzschlusskäfig
- Rotorstab 193, 225
- Rotorwiderstand → Widerstand

S

- Sättigung 149
- Schalter 106, 279, 320, 322
- Schenkelläufer 249
- Schleifringläufer 212, 244
- Schrägung 219
- Sehnung 158
- Sehnungsfaktor → Wicklungsfaktor
- Sensor 89, 90, 319
- SHA-1 → Prüfsumme SHA-1
- SI-Einheit 49, 74
- sign 61
- Signal 89
- Simulation
 - Co-Simulation 40
 - Integrationsverfahren 46
- Simulationseinstellung 45, 49, 77
- Simulationsmodell 39, 77
- sin, sinh 61
- size 66
- Skalar 62
- Skalarprodukt 65
- smooth 105
- Spannung 20
 - induziert 145, 213, 254
 - Leerlauf 254
 - Polradspannung 261
- Spannungs-Frequenz-Kennlinie 235
- Spannungsgleichung
 - Asynchronmaschine 220
 - Synchronmaschine 257
- Spannungsregelung 279
- speichern 46, 77
- Spule 157, 164, 320, 322
- Spulengruppe 159, 162
- Spulenschritt 157, 161
- stage 303
- start 67
- Startwert 45, 68
- state → Zustandsvariable

- Steinmetz-Schaltung 240
- Stern-Dreieck-Anlauf 232
- Sternschaltung 111, 157, 189
- Streuinduktivität → Induktivität
- Streukoeffizient 252, 269
- String 92
- Strom 20
 - Nullstrom 190
 - Strangstrom 159, 166
- Stromwinkel → Winkel
- sum 66
- Symmetrie 149, 172
- Symmetrische Komponenten der
 - Momentanwerte 221
- Synchronisation 270
- Synchronmaschine 321
 - Direktanlauf 267
 - Drehmoment 285
 - elektrisch erregt 249, 250, 262, 267
 - Inselbetrieb 279, 281
 - MTPA 289, 293
 - Netzbetrieb 270
 - Parametrisierung 263
 - Permanentmagnet 249, 254, 260, 263, 285
 - Stoßkurzschluss 281
 - stromgespeist 285, 289
 - V-Kurve 276
 - variabler Polradwinkel 271
- Synchronreluktanzmaschine 249, 256
 - MTPA 293
 - Umrichter 291
- System-Library 41

T

- Tabelle 91
- Tag 315
- tan, tanh 61
- Temperatur 31
 - Rotor 213, 215, 252
- Temperaturkoeffizient 22, 212
- Text View 43, 54
- Textfeld 98
- Thyristor 320, 322
- Transformation
 - Raumzeiger → Raumzeiger

- Wicklung 222, 225, 258
- Transformator 28, 126, 320, 321
- Anlasstransformator 235
- Transistor 320
- transpose 66
- Turboläufer 249
- type 74

U

- übererregt 274
- umbenennen 52
- Umlaufspannung 25
- unit 48, 67, 74
- Unload 47
- unstage 303, 304
- untererregt 274
- Unterschicht 157
- URI 93

V

- V-Kurve 276
- Variable 59
- Vektor 62, 113
 - Länge 66
- Vektorprodukt 66
- Verbindung → connect
- Verdrehwinkel 33
- Vererbung 57, 80
- Vergleich 62
- Verkettungsfluss → magnetische Flussverkettung
- Verluste 149, 209, 228
 - Eisenverluste 30, 126, 184, 323
 - Reibung 195, 324
 - Zusatzverluste 196
- Version 313
 - erstellen 314
- Versionierung 295, 311
- Voraussetzungen der Modellbildung 149

W

- Wicklung 157
 - Bruchlochwicklung 151
 - Einschichtwicklung 157

- Ganzlochwicklung 151
- Modelica 190
- Sechszonenwicklung 217
- Transformation 222
- Zweischichtwicklung 157
- Wicklungsachse 160, 169
- Wicklungsfaktor
 - Sehnungsfaktor 165, 217
 - Zonenfaktor 217
- Widerstand
 - bezogen 264
 - elektrisch 22, 137, 320, 322
 - elektrisch, temperaturabhängig 22
 - Erregerwicklung 252, 264
 - magnetisch → Reluktanz
 - Rotor 213, 215, 247
 - thermisch 31, 326
- Wiederherstellung → Repository
- Wiki 299
- Windungszahl
 - effektiv 217
 - komplex 167, 217
 - Spule 159, 165
 - Verhältnis 213
- Winkel 152
 - Phasenwinkel 113, 261, 289, 294
 - Polradwinkel 261, 271, 289, 294
 - Stromwinkel 261, 285, 293
 - Verdrehwinkel 217
- Winkelgeschwindigkeit 33, 141
- Wirbelstromverluste → Eisenverluste
- within 48

Z

- Zählpfeilsystem 20, 150
- Zeit
 - Simulation → Simulationseinstellung
 - Variable time 59
- Zeitkonstante 267
- Zeitzeiger 15, 116
- zeros 65
- Zonenfaktor → Wicklungsfaktor
- Zusammenarbeit → Repository
- Zusatzverluste 196
- Zustandsvariable 71, 72, 104
- Zweig → if