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

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

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 Ausgaberaster 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/*.

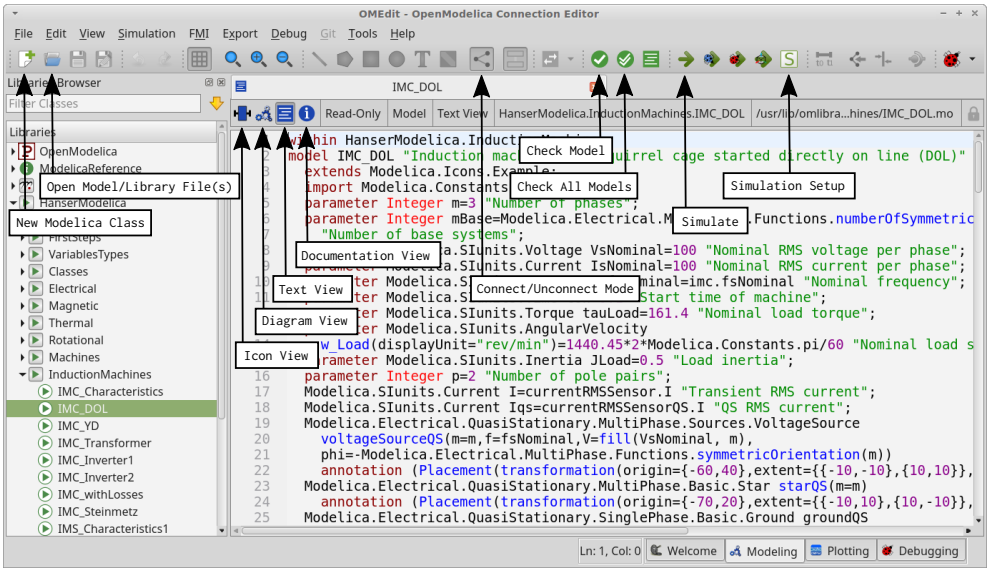


Bild 2.1 OpenModelica Editor

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

$$v_R = R \cdot i. \tag{2.1}$$

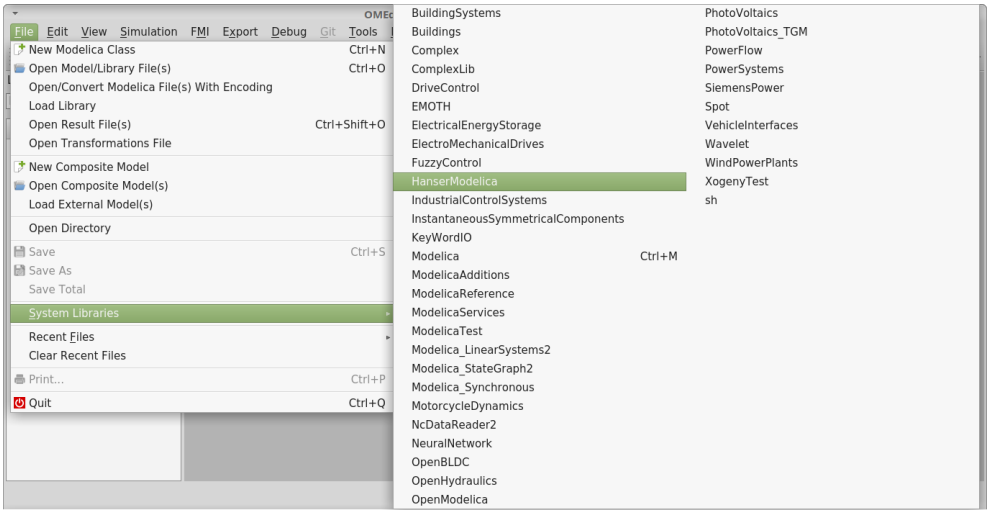
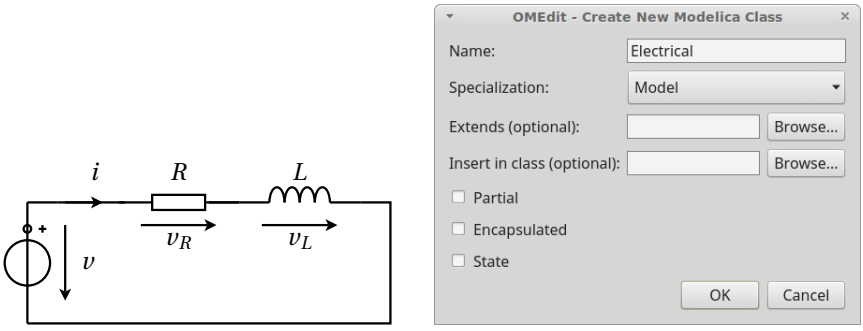


Bild 2.2 Die Library `HanserModelica` ist in OpenModelica als System-Library integriert



(a) **(b)**

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

$$v_L = L \cdot \frac{\mathrm{d}i}{\mathrm{d}t} \tag{2.2}$$

zusammen. Zusätzlich gilt die Maschenregel

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

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

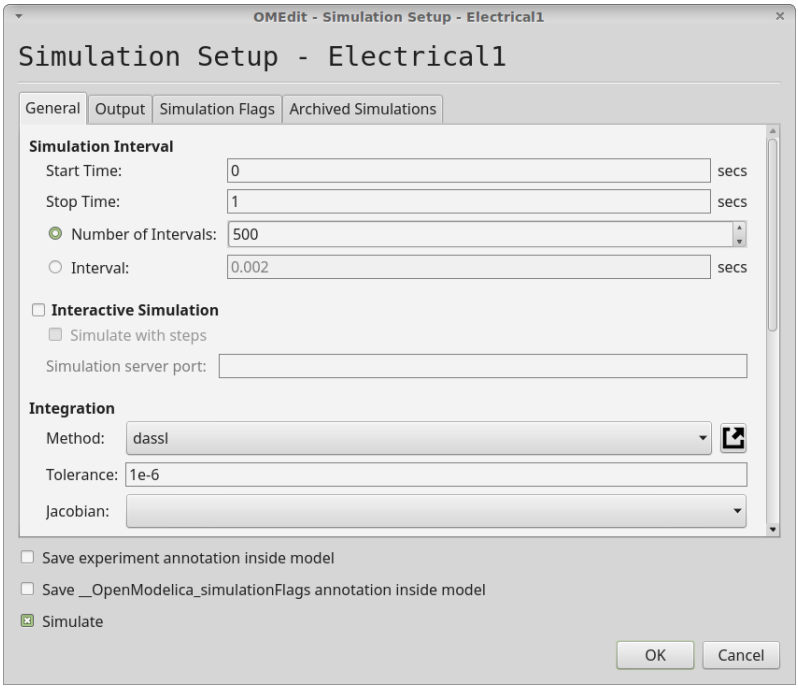


Bild 2.4 Simulationseinstellungen des Modells `Electrical1`

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 erstelle 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 einzeiliger 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

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

U

V

W

Z