

Simscape™ Power Systems™

User's Guide (Simscape™ Components)



MATLAB® & SIMULINK®

R2018a



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

Simscape™ Power Systems™ User's Guide (Simscape™ Components)

© COPYRIGHT 2013–2018 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

September 2013	Online only	New for Version 6.0 (Release 2013b)
March 2014	Online only	Revised for Version 6.1 (Release 2014a) (Renamed from <i>SimPowerSystems™ User's Guide (Third Generation)</i>)
October 2014	Online only	Revised for Version 6.2 (Release 2014b)
March 2015	Online only	Revised for Version 6.3 (Release 2015a)
September 2015	Online only	Revised for Version 6.4 (Release 2015b)
March 2016	Online only	Revised for Version 6.5 (Release 2016a) (Renamed from <i>SimPowerSystems™ User's Guide (Simscape™ Components)</i>)
September 2016	Online only	Revised for Version 6.6 (Release 2016b)
March 2017	Online only	Revised for Version 6.7 (Release 2017a)
September 2017	Online only	Revised for Version 6.8 (Release 2017b)
March 2018	Online only	Revised for Version 6.9 (Release 2018a)

Simscape Power Systems Product Description	1-2
Key Features	1-2
Simscape Power Systems Block Libraries	1-3
Overview of Simscape Power Systems Libraries	1-3
Simscape Components Library	1-3
Specialized Technology Library	1-3
Using the Simulink Library Browser to Access the Block Libraries	1-4
Using the Command Prompt to Access the Block Libraries . . .	1-4
Comparison of Simscape Components and Specialized Technology	1-6
Per-Unit System of Units	1-7
What Is the Per-Unit System?	1-7
Example 1: Three-Phase Transformer	1-9
Example 2: Asynchronous Machine	1-10
Base Values for Instantaneous Voltage and Current Waveforms	1-11
Why Use the Per-Unit System Instead of the Standard SI Units?	1-11

Modeling Power Engineering Systems Using Simscape Power Systems Simscape Components Software	2-2
Simscape Power Systems Simscape Components Blocks and Ports	2-2
Machine and Transformer Source Code Examples	2-3
Plotting and Display Options for Asynchronous and Synchronous Machines	2-3
Choosing the Right Simscape Technology	2-3
Essential Power Engineering Modeling Techniques	2-4
Overview of Modeling Rules	2-4
Required Blocks	2-5
Three-Phase Ports	2-6
About Three-Phase Ports	2-6
Expand and Collapse Three-Phase Ports on a Block	2-7
Switch Between Physical Signal and Electrical Ports	2-8

Build and Simulate Composite Resistive and Reactive Three-Phase Models	3-2
Select System Component Blocks	3-2
Specify Simulation Parameters	3-4
Load Impedance Parameters	3-5
Specify Display Parameters	3-5
Save and Simulate the Model	3-6
Analyze the Resistive Three-Phase Model Simulation Results	3-6
Create and Simulate a Reactive Three-Phase Load	3-6
Analyze the Reactive Three-Phase Model Simulation Results	3-7

Create and Simulate Expanded Balanced and Unbalanced Three-Phase Models	3-9
Create an Expanded Balanced Three-Phase Model	3-9
Create an Expanded Unbalanced Three-Phase Model	3-10
Simulate the Models and Analyze Results	3-10

Modeling Machines

4

Machine Parameterization	4-2
Per-Unit Conversion for Machine Parameters	4-4
Impedance Conversion Equations	4-4
Magnetic Flux Linkage Conversion Equations	4-4
Machine Plotting and Display Options	4-6
Asynchronous Machine Options	4-6
Synchronous Machine Options	4-6
Machine Inertia Block Options	4-7
Initialize Synchronous Machines and Controllers	4-8

Customization

5

Build Custom Blocks Using the Three-Phase Electrical Domain	5-2
Customizing Machine Models	5-4
Custom Synchronous Machine	5-6

6

Tune an Electric Drive	6-2
Cascade Control Structure	6-2
Equations for PI Tuning Using the Pole Placement Method ...	6-2
Equations for DC Motor Controller Tuning	6-6
Tune the Electric Drive in the Example Model	6-8

Simulation and Analysis of Power Engineering Systems

7

Simulating Power Engineering Systems	7-2
Examine the Simulation Data-Logging Configuration of a Model	7-3
Simulate Thermal Losses in Semiconductors	7-5
Prerequisite	7-5
Thermal Variants	7-5
Thermal Blocks	7-5
Thermal Ports	7-6
Thermal-Modeling Parameters	7-7
Limitations	7-7
Model Thermal Losses for a Rectifier	7-8
Perform a Power-Loss Analysis	7-16
Prerequisite	7-16
Calculate Average Power Losses for the Simulation	7-16
Analyze Power Dissipation Differences Using Instantaneous Power Dissipation	7-18
Mitigate Transient Effects in Simulation Data	7-22
Choose a Simscape Power Systems Function for an Offline Harmonic Analysis	7-25
Harmonic Distortion	7-25
Harmonic Analysis Functions	7-25
Evaluate Relative Overall Harmonic Distortion	7-26

Compare Harmonic Distortion to Standard Limits	7-27
Minimize Harmonic Distortion with Passive Filters	7-27
Verify the Results of an Online Harmonic Analysis	7-28
Perform an Online Harmonic Analysis Using the Simscape	
Spectrum Analyzer Block	7-29
Harmonic Distortion	7-29
Prerequisite	7-29
Perform an Offline Harmonic Analysis	7-30
Perform an Online Harmonic Analysis	7-33
Optimize Block Settings for Simulations that Use the	
Partitioning Solver	7-38
Update Solver and Zero-Sequence Settings Using the	
pe_solverUpdate Function	7-39
Limitations of the pe_updateSolver Function	7-48
Prepare Simscape Power Systems Models for Real-Time	
Simulation Using Simscape Checks	7-49
Phasor-Mode Simulation in Simscape Components	7-51

Getting Started

- “Simscape Power Systems Product Description” on page 1-2
- “Simscape Power Systems Block Libraries” on page 1-3
- “Comparison of Simscape Components and Specialized Technology” on page 1-6
- “Per-Unit System of Units” on page 1-7

Simscape Power Systems Product Description

Model and simulate electrical power systems

Simscape Power Systems provides component libraries and analysis tools for modeling and simulating electrical power systems. It includes models of electrical power components, including three-phase machines, electric drives, and components for applications such as flexible AC transmission systems (FACTS) and renewable energy systems. Harmonic analysis, calculation of total harmonic distortion (THD), load flow, and other key electrical power system analyses are automated, helping you investigate the performance of your design.

Simscape Power Systems helps you develop control systems and test system-level performance. You can parameterize your models using MATLAB® variables and expressions, and design control systems for your electrical power system in Simulink®. You can integrate mechanical, hydraulic, thermal, and other physical systems into your model using components from the Simscape family of products. To deploy models to other simulation environments, including hardware-in-the-loop (HIL) systems, Simscape Power Systems supports C-code generation.

Simscape Power Systems was developed in collaboration with Hydro-Québec of Montreal.

Key Features

- Application-specific models, including common AC and DC electric drives, flexible AC transmission systems (FACTS), and renewable energy systems
- Discretization and phasor simulation modes for faster model execution
- Ideal switching algorithm for accelerated simulation of power electronic devices
- Analysis methods for obtaining state-space representations of circuits and computing machine load flow
- Basic models for developing key electrical technologies
- MATLAB based Simscape language for creating custom component models
- Support for C-code generation (with Simulink Coder™)

Simscape Power Systems Block Libraries

Overview of Simscape Power Systems Libraries

Simscape Power Systems contains two top-level block libraries, representing two different technologies.

Simscape Components Library

The Simscape Power Systems Simscape Components library contains Simscape blocks specifically developed for working with multiphase electrical domains. In addition to the Simscape Foundation domains, the product contains a three-phase electrical domain, and you can use this domain to develop your own custom three-phase blocks with Simscape language. You can also view and customize source for several machine and transformer example blocks provided with the product.

The main Simscape Components library is called **pe_lib**. It contains Simscape blocks with three-phase and single-phase electrical ports, as well as mechanical rotational and translational ports. These blocks are organized into sublibraries according to their function. You can connect these blocks directly to other Simscape blocks, from the Foundation library or from the other add-on products.

Three-phase electrical ports on Simscape Components blocks are collapsed by default, to support single-line diagrams. You can optionally expand them to separate the phases, for example, to inject a single-line-to-ground fault into your circuit.

Specialized Technology Library

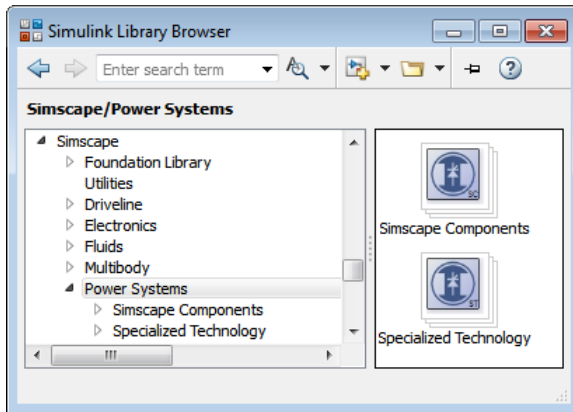
The Simscape Power Systems Specialized Technology library contains models of typical power equipment such as transformers, lines, machines, and power electronics. These models are proven ones coming from textbooks, and their validity is based on the experience of the Power Systems Testing and Simulation Laboratory of Hydro-Québec, a large North American utility located in Canada, and also on the experience of École de Technologie Supérieure and Université Laval. The capabilities of Simscape Power Systems software for modeling a typical electrical system are illustrated in example files. For users who want to refresh their knowledge of power system theory, there are also self-learning case studies.


The main Specialized Technology library is called **simscapepowersystems_ST**. It contains sublibraries of blocks that are organized according to their behavior. The

Fundamental Blocks sublibrary contains the Powergui block, which contains tools for the steady-state analysis of electrical circuits.

Using the Simulink Library Browser to Access the Block Libraries

You can access both Simscape Power Systems libraries through the Simulink Library Browser. To display the Simulink Library Browser, at the MATLAB command prompt, enter `sLibraryBrowser`. In the left pane of the library browser, scroll to the **Simscape** node. Expand the **Simscape** node and then the **Power Systems** node.



You can also use the library browser to access the Simscape Power Systems libraries from a Simulink model window. In the toolbar of the model window, click the **Library Browser** button .

Using the Command Prompt to Access the Block Libraries

To access the libraries that the table describes, enter the relevant command at the MATLAB command prompt.

Library	Description	Command
Simscape Power Systems	Top-level libraries in Simscape Power Systems	<code>simscapepowersystems</code>

Library	Description	Command
Simscape Power Systems Simscape Components	Simscape Components blocks	pe_lib
Simscape Power Systems Specialized Technology	Specialized Technology blocks	simscapepowersystems_ST
Simscape Power Systems Specialized Technology Fundamental Blocks	Powergui block, interface elements, and other fundamental Specialized Technology blocks	powerlib
Simscape	Simscape utility blocks, single-phase electrical sources, sensors, and other Foundation library blocks	simscape
Simulink	Generic Simulink blocks	slLibraryBrowser

Comparison of Simscape Components and Specialized Technology

Simscape Power Systems software includes two technologies and corresponding libraries to model and simulate electrical power systems. You can create and simulate systems using either library. A single Simscape Power Systems model can contain components from both libraries. You use electrical ports to connect the components within each technology and, ultimately, use Simulink signals to connect Simscape Components blocks to Specialized Technology blocks.

- Simscape Components libraries use the full range of Simscape technology, and the component models are written in the Simscape language. You can directly connect these models with the Simscape Foundation library components and with components from other add-on products. You can view and customize the Simscape language source code examples for machines and transformers. Simscape Components models are fully compatible with Simscape technology, including local solver and data logging.

Simscape Components also provides three-phase electrical connection ports, which you can individually expand into separate phases as needed. By default, these three-phase ports are collapsed, to support single-line diagrams.

- Specialized Technology libraries provide components and technology specifically developed for electrical power systems. The Specialized Technology library has been refined and tuned for more than a decade. Specialized Technology models use their own electrical domain. You connect Specialized Technology blocks to other Simscape elements ultimately through Simulink signals. The Specialized Technology software provides various simulation methods (continuous, discrete, phasor) and analysis tools.

Per-Unit System of Units

In this section...

“What Is the Per-Unit System?” on page 1-7

“Example 1: Three-Phase Transformer” on page 1-9

“Example 2: Asynchronous Machine” on page 1-10

“Base Values for Instantaneous Voltage and Current Waveforms” on page 1-11

“Why Use the Per-Unit System Instead of the Standard SI Units?” on page 1-11

What Is the Per-Unit System?

The per-unit system is widely used in the power system industry to express values of voltages, currents, powers, and impedances of various power equipment. It is typically used for transformers and AC machines.

For a given quantity (voltage, current, power, impedance, torque, etc.) the per-unit value is the value related to a base quantity.

$$\text{base value in p.u.} = \frac{\text{quantity expressed in SI units}}{\text{base value}}$$

Generally the following two base values are chosen:

- The base power = nominal power of the equipment
- The base voltage = nominal voltage of the equipment

All other base quantities are derived from these two base quantities. Once the base power and the base voltage are chosen, the base current and the base impedance are determined by the natural laws of electrical circuits.

$$\text{base current} = \frac{\text{base power}}{\text{base voltage}}$$

$$\text{base impedance} = \frac{\text{base voltage}}{\text{base current}} = \frac{(\text{base voltage})^2}{\text{base power}}$$

For a transformer with multiple windings, each having a different nominal voltage, the same base power is used for all windings (nominal power of the transformer). However,

according to the definitions, there are as many base values as windings for voltages, currents, and impedances.

The saturation characteristic of saturable transformer is given in the form of an instantaneous current versus instantaneous flux-linkage curve: [i_1 ϕ_1 ; i_2 ϕ_2 ; ..., in ϕ_{in}].

When the per-unit system is used to specify the transformer R L parameters, the flux linkage and current in the saturation characteristic must be also specified in pu. The corresponding base values are

$$\begin{aligned}\text{base instantaneous current} &= (\text{base rms current}) \times \sqrt{2} \\ \text{base flux linkage} &= \frac{(\text{base rms voltage}) \times \sqrt{2}}{2\pi \times (\text{base frequency})}\end{aligned}$$

where current, voltage, and flux linkage are expressed respectively in volts, amperes, and volt-seconds.

For AC machines, the torque and speed can be also expressed in pu. The following base quantities are chosen:

- The base speed = synchronous speed
- The base torque = torque corresponding at base power and synchronous speed

$$\text{base torque} = \frac{\text{base power (3 phases) in VA}}{\text{base speed in radians/second}}$$

Instead of specifying the rotor inertia in $\text{kg}\cdot\text{m}^2$, you would generally give the inertia constant H defined as

$$\begin{aligned}H &= \frac{\text{kinetic energy stored in the rotor at synchronous speed in joules}}{\text{machine nominal power in VA}} \\ H &= \frac{\frac{1}{2} \times J \cdot \omega^2}{P_{nom}}\end{aligned}$$

The inertia constant is expressed in seconds. For large machines, this constant is around 3–5 seconds. An inertia constant of 3 seconds means that the energy stored in the

rotating part could supply the nominal load during 3 seconds. For small machines, H is lower. For example, for a 3-HP motor, it can be 0.5–0.7 seconds.

Example 1: Three-Phase Transformer

Consider, for example, a three-phase two-winding transformer with these manufacturer-provided, typical parameters:

- Nominal power = 300 kVA total for three phases
- Nominal frequency = 60 Hz
- Winding 1: connected in wye, nominal voltage = 25 kV RMS line-to-line
resistance 0.01 pu, leakage reactance = 0.02 pu
- Winding 2: connected in delta, nominal voltage = 600 V RMS line-to-line
resistance 0.01 pu, leakage reactance = 0.02 pu
- Magnetizing losses at nominal voltage in % of nominal current:
Resistive 1%, Inductive 1%

The base values for each single-phase transformer are first calculated:

- For winding 1:

Base power	$300 \text{ kVA}/3 = 100\text{e}3 \text{ VA/phase}$
Base voltage	$25 \text{ kV}/\sqrt{3} = 14434 \text{ V RMS}$
Base current	$100\text{e}3/14434 = 6.928 \text{ A RMS}$
Base impedance	$14434/6.928 = 2083 \ \Omega$
Base resistance	$14434/6.928 = 2083 \ \Omega$
Base inductance	$2083/(2\pi*60) = 5.525 \text{ H}$

- For winding 2:

Base power	$300 \text{ kVA}/3 = 100\text{e}3 \text{ VA}$
Base voltage	600 V RMS
Base current	$100\text{e}3/600 = 166.7 \text{ A RMS}$
Base impedance	$600/166.7 = 3.60 \ \Omega$

Base resistance	$600/166.7 = 3.60 \Omega$
Base inductance	$3.60/(2\pi*60) = 0.009549 \text{ H}$

The values of the winding resistances and leakage inductances expressed in SI units are therefore

- For winding 1: $R_1 = 0.01 * 2083 = 20.83 \Omega$; $L_1 = 0.02*5.525 = 0.1105 \text{ H}$
- For winding 2: $R_2 = 0.01 * 3.60 = 0.0360 \Omega$; $L_2 = 0.02*0.009549 = 0.191 \text{ mH}$

For the magnetizing branch, magnetizing losses of 1% resistive and 1% inductive mean a magnetizing resistance R_m of 100 pu and a magnetizing inductance L_m of 100 pu. Therefore, the values expressed in SI units referred to winding 1 are

- $R_m = 100*2083 = 208.3 \text{ k}\Omega$
- $L_m = 100*5.525 = 552.5 \text{ H}$

Example 2: Asynchronous Machine

Now consider a three-phase, four-pole Asynchronous Machine block in SI units. It is rated 3 HP, 220 V RMS line-to-line, 60 Hz.

The stator and rotor resistance and inductance referred to stator are

- $R_s = 0.435 \Omega$; $L_s = 2 \text{ mH}$
- $R_r = 0.816 \Omega$; $L_r = 2 \text{ mH}$

The mutual inductance is $L_m = 69.31 \text{ mH}$. The rotor inertia is $J = 0.089 \text{ kg.m}^2$.

The base quantities for one phase are calculated as follows:

Base power	$3 \text{ HP}*746\text{VA}/3 = 746 \text{ VA/phase}$
Base voltage	$220 \text{ V}/\sqrt{3} = 127.0 \text{ V RMS}$
Base current	$746/127.0 = 5.874 \text{ A RMS}$
Base impedance	$127.0/5.874 = 21.62 \Omega$
Base resistance	$127.0/5.874 = 21.62 \Omega$
Base inductance	$21.62/(2\pi*60) = 0.05735 \text{ H} = 57.35 \text{ mH}$
Base speed	$1800 \text{ rpm} = 1800*(2\pi)/60 = 188.5 \text{ radians/second}$

Base torque (three-phase)	$746 \times 3 / 188.5 = 11.87$ newton-meters
---------------------------	--

Using the base values, you can compute the values in per-units.

$$R_s = 0.435 / 21.62 = 0.0201 \text{ pu} \quad L_s = 2 / 57.35 = 0.0349 \text{ pu}$$

$$R_r = 0.816 / 21.62 = 0.0377 \text{ pu} \quad L_r = 2 / 57.35 = 0.0349 \text{ pu}$$

$$L_m = 69.31 / 57.35 = 1.208 \text{ pu}$$

The inertia is calculated from inertia J , synchronous speed, and nominal power.

$$H = \frac{\frac{1}{2} \times J \cdot \omega^2}{P_{nom}} = \frac{\frac{1}{2} \times 0.089 \times (188.5)^2}{3 \times 746} = 0.7065 \text{ seconds}$$

If you open the dialog box of the Asynchronous Machine block in pu units provided in the Machines library of the Simscape Power Systems Specialized Technology Fundamental Blocks library, you find that the parameters in pu are the ones calculated.

Base Values for Instantaneous Voltage and Current Waveforms

When displaying instantaneous voltage and current waveforms on graphs or oscilloscopes, you normally consider the peak value of the nominal sinusoidal voltage as 1 pu. In other words, the base values used for voltage and currents are the RMS values given multiplied by $\sqrt{2}$.

Why Use the Per-Unit System Instead of the Standard SI Units?

Here are the main reasons for using the per-unit system:

- When values are expressed in pu, the comparison of electrical quantities with their "normal" values is straightforward.

For example, a transient voltage reaching a maximum of 1.42 pu indicates immediately that this voltage exceeds the nominal value by 42%.

- The values of impedances expressed in pu stay fairly constant whatever the power and voltage ratings.

For example, for all transformers in the 3-300 kVA power range, the leakage reactance varies approximately 0.01-0.03 pu, whereas the winding resistances vary between 0.01 pu and 0.005 pu, whatever the nominal voltage. For transformers in the 300 kVA to 300 MVA range, the leakage reactance varies approximately 0.03-0.12 pu, whereas the winding resistances vary between 0.005-0.002 pu.

Similarly, for salient pole synchronous machines, the synchronous reactance X_d is generally 0.60-1.50 pu, whereas the subtransient reactance X'_d is generally 0.20-0.50 pu.

It means that if you do not know the parameters for a 10-kVA transformer, you are not making a major error by assuming an average value of 0.02 pu for leakage reactances and 0.0075 pu for winding resistances.

The calculations using the per-unit system are simplified. When all impedances in a multivoltage power system are expressed on a common power base and on the nominal voltages of the different subnetworks, the total impedance in pu seen at one bus is obtained by simply adding all impedances in pu, without considering the transformer ratios.

Modeling Basics

- “Modeling Power Engineering Systems Using Simscape Power Systems Simscape Components Software” on page 2-2
- “Essential Power Engineering Modeling Techniques” on page 2-4
- “Three-Phase Ports” on page 2-6
- “Switch Between Physical Signal and Electrical Ports” on page 2-8

Modeling Power Engineering Systems Using Simscape Power Systems Simscape Components Software

In this section...
“Simscape Power Systems Simscape Components Blocks and Ports” on page 2-2
“Machine and Transformer Source Code Examples” on page 2-3
“Plotting and Display Options for Asynchronous and Synchronous Machines” on page 2-3
“Choosing the Right Simscape Technology” on page 2-3

Simscape Power Systems Simscape Components Blocks and Ports

Simscape Power Systems Simscape Components blocks are written in the Simscape language. The blocks are fully compatible with Simscape technology, including the local solver, code generation, and data logging.

Simscape Power Systems Simscape Components blocks have composite three-phase, electrical conserving, and mechanical rotational conserving ports. You can use composite three-phase ports to build models corresponding to single-line diagrams of three-phase electrical systems. Composite three-phase ports connect to other composite three-phase ports. Electrical and mechanical rotational conserving ports connect directly to Simscape Foundation library components and Simscape add-on products such as Simscape Electronics™. You can use a Phase Splitter block to split a composite three-phase port into individual electrical conserving ports.

Blocks in the **Semiconductors** library of Simscape Power Systems Simscape Components software have an option to switch certain ports between physical signal and electrical conserving ports. When you select electrical ports, the semiconductor block has the same ports as the equivalent semiconductor block in Simscape Electronics. Therefore, you can easily switch semiconductor blocks in your model between the two products. For example, suppose that you use Simscape Power Systems Simscape Components semiconductor blocks to model the electronic drive circuit for a three-phase machine but want to increase the drive circuit fidelity. You can directly replace the semiconductor blocks with higher-fidelity versions from the Simscape Electronics block library.

Machine and Transformer Source Code Examples

Simscape Power Systems Simscape Components software comes with Simscape language source code examples for machines and transformers, which you can view and customize. To access the example blocks, type `ThreePhaseExamples_Lib` at the MATLAB command prompt.

Plotting and Display Options for Asynchronous and Synchronous Machines

For the Machine Inertia block and the asynchronous and synchronous machine blocks in Simscape Power Systems Simscape Components software, you can perform some useful plotting and display actions using the **Power Systems** menu on the block context menu. For example, to plot torque versus speed (both in SI units) for the Asynchronous Machine Wound Rotor (fundamental) block, right-click the block. From the block context menu, select **Power Systems > Plot Torque Speed (SI)**. The software plots the results in a figure window.

Using other options on the **Power Systems** menu, you can plot values in per-unit or display base parameter values in the MATLAB Command Window. These options enable you to tune the performance of your three-phase machine quickly.

Choosing the Right Simscape Technology

Simscape Power Systems software includes two different technologies and corresponding libraries. For a comparison of the two technologies, see “Comparison of Simscape Components and Specialized Technology” on page 1-6. Choose the Simscape Power Systems technology most appropriate for your modeling needs and if possible, build your model using blocks exclusively from that technology. However, if necessary, you can build a model that uses blocks from both technologies. To do so, use blocks from the **Simscape > Power Systems > Specialized Technology > Interface Elements** library to interface between them.

Essential Power Engineering Modeling Techniques

In this section...
“Overview of Modeling Rules” on page 2-4
“Required Blocks” on page 2-5

Overview of Modeling Rules

Simscape Power Systems Simscape Components models are essentially Simscape block diagrams refined for modeling three-phase electrical systems. Simscape Power Systems Simscape Components blocks feature these port types:

- Three-phase ports, which connect the phases of a three-phase electrical system between Simscape Power Systems Simscape Components blocks.

There are two three-phase port types in Simscape Power Systems Simscape Components blocks, composite and expanded. You can connect a composite three-phase port only to another composite three-phase port. You can connect the individual electrical conserving ports of an expanded three-phase port only to other electrical conserving ports. For more information, see “Three-Phase Ports” on page 2-6.

- Electrical and mechanical rotational conserving ports, which connect directly to Simscape foundation blocks.

Each port type has specific Across and Through variables associated with it. To learn about the rules to follow when building an electromechanical model, see “Basic Principles of Modeling Physical Networks” (Simscape).

- Physical signal ports, which connect to Simulink blocks through the Simulink-PS Converter and PS-Simulink Converter blocks from the Simscape Utilities library. These blocks convert physical signals to and from Simulink mathematical signals.

Keep these rules in mind when using each port type in Simscape Power Systems Simscape Components blocks.

- You can connect physical conserving ports only to other conserving ports of the same type. Electrical conserving ports in Simscape Power Systems Simscape Components blocks can connect directly to Simscape Electronics blocks and Simscape electrical components. Mechanical rotational conserving ports in Simscape Power Systems Simscape Components blocks can connect directly to Simscape mechanical rotational components.

- The physical connection lines that connect conserving ports are nondirectional lines that carry physical variables (Across and Through variables) rather than signals. You cannot connect physical conserving ports to Simulink ports or to physical signal ports.
- You can branch physical connection lines. When you do so, directly connected components share the same Across variables. The value of any Through variable (e.g., current or torque) transferred along the physical connection line is divided among the multiple components connected by the branches.

For each Through variable, the sum of the values flowing into a branch point equals the sum of the values flowing out.

- You can connect physical signal ports to other physical signal ports using regular connection lines, similar to Simulink signal connections. These connection lines carry physical signals between Simscape Power Systems Simscape Components blocks.
- You can connect physical signal ports to Simulink ports through converter blocks. Use the Simulink-PS Converter block to connect Simulink outputs to physical signal inports. Use the PS-Simulink Converter block to connect physical signal outputs to Simulink inports.
- Unlike Simulink signals, physical signals can have units. In Simscape Power Systems Simscape Components block dialog boxes, you can specify the units along with the parameter values, where appropriate. Use the converter blocks to associate units with an input signal and to specify the desired output signal units.

For an example of these rules applied to an electromechanical model, see *Three-Phase Asynchronous Machine Starting*.

Required Blocks

Each topologically distinct physical network in a diagram requires exactly one Solver Configuration block from the Simscape Utilities library. The Solver Configuration block specifies global environment information for simulation and provides parameters for the solver that your model needs for simulation.

Each electrical network requires an Electrical Reference block. This block establishes the electrical ground for the circuit. Networks with electromechanical blocks also require a Mechanical Rotational Reference block. For more information about using reference blocks, see “Grounding Rules” (Simscape).

Three-Phase Ports

In this section...
“About Three-Phase Ports” on page 2-6
“Expand and Collapse Three-Phase Ports on a Block” on page 2-7

About Three-Phase Ports

In Simscape Power Systems Simscape Components software, you can connect the phases of a three-phase system between blocks using two types of port.

- Composite three-phase port
- Expanded three-phase port

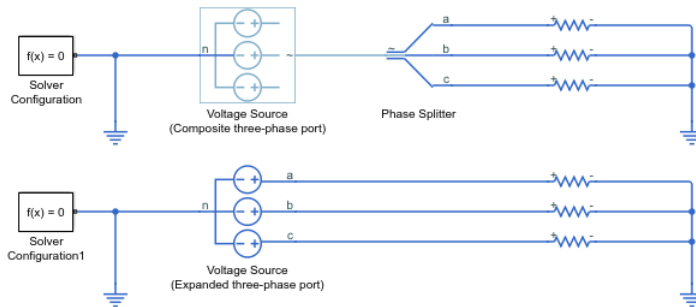
Composite three-phase ports represent three individual electrical conserving ports with a single block port. You can use composite three-phase ports to build models that correspond to single-line diagrams of three-phase electrical systems. Instead of explicitly connecting each phase of the three-phase system between blocks, you connect all three phases using a single port. You can connect composite three-phase ports only to other composite three-phase ports.

Expanded three-phase ports represent the individual phases of a three-phase system using three separate electrical conserving ports. You individually connect each phase of the three-phase system between blocks. Electrical conserving ports can connect directly to electrical components from the Simscape and Simscape Electronics libraries.

Composite three-phase ports produce results with the same fidelity as expanded three-phase ports. Both connection methods consider instantaneous phase voltages and currents and are suitable for modeling balanced and unbalanced three-phase electrical power systems. Each electrical conserving port in an expanded three-phase port has a Through variable of scalar current and an Across variable of scalar voltage. For a composite three-phase port, the Through variable is a three-element current, and the Across variable is a three-element voltage.

You can use the Phase Splitter block to expand a composite three-phase port into separate electrical conserving ports. The separate electrical ports can then connect to Simscape and Simscape Electronics electrical components.

The figure shows two simple circuits that contrast the composite and expanded connection methods. The two circuits produce the same results.



The top circuit uses a Voltage Source block with a composite three-phase port \sim . The bottom circuit uses a Voltage Source block with expanded electrical conserving ports a, b, and c. In each circuit, the instantaneous phase voltages and currents are the same.

Expand and Collapse Three-Phase Ports on a Block

Simscape Power Systems Simscape Components blocks that have composite three-phase ports have an option to switch between composite and expanded ports.

- Right-click the block. On the **Simscape block choices** context menu, select **Expanded three-phase ports** or **Composite three-phase ports**.

For blocks with a single composite port \sim , the expanded electrical ports are labeled a, b, and c. For blocks with more than one composite port ~ 1 and ~ 2 , the expanded electrical ports are labeled a1, b1, c1 and a2, b2, c2.

Switch Between Physical Signal and Electrical Ports

Some Simscape Power Systems Simscape Components blocks have an option to switch certain ports between physical signal and electrical conserving ports. An electrical conserving port is a Simscape physical conserving port that has a Through variable of current and an Across variable of voltage. For a comparison of Simscape physical signal and physical conserving ports, see “Connector Ports and Connection Lines” (Simscape).

- Right-click the block. On the **Simscape block choices** context menu, select a variant that includes an **Electrical control port** or **PS control port**.

The block connection port switches between (electrical conserving port) and (physical signal port).

Tutorials

- “Build and Simulate Composite Resistive and Reactive Three-Phase Models”
on page 3-2
- “Create and Simulate Expanded Balanced and Unbalanced Three-Phase Models”
on page 3-9

Build and Simulate Composite Resistive and Reactive Three-Phase Models

In this section...

“Select System Component Blocks” on page 3-2

“Specify Simulation Parameters” on page 3-4

“Load Impedance Parameters” on page 3-5

“Specify Display Parameters” on page 3-5

“Save and Simulate the Model” on page 3-6

“Analyze the Resistive Three-Phase Model Simulation Results” on page 3-6

“Create and Simulate a Reactive Three-Phase Load” on page 3-6

“Analyze the Reactive Three-Phase Model Simulation Results” on page 3-7

This tutorial shows how to build a Simscape Power Systems Simscape Components model. The model simulates the behavior of a three-phase AC voltage source driving a simple load.

To see the completed model, click Simple Three-Phase Model.

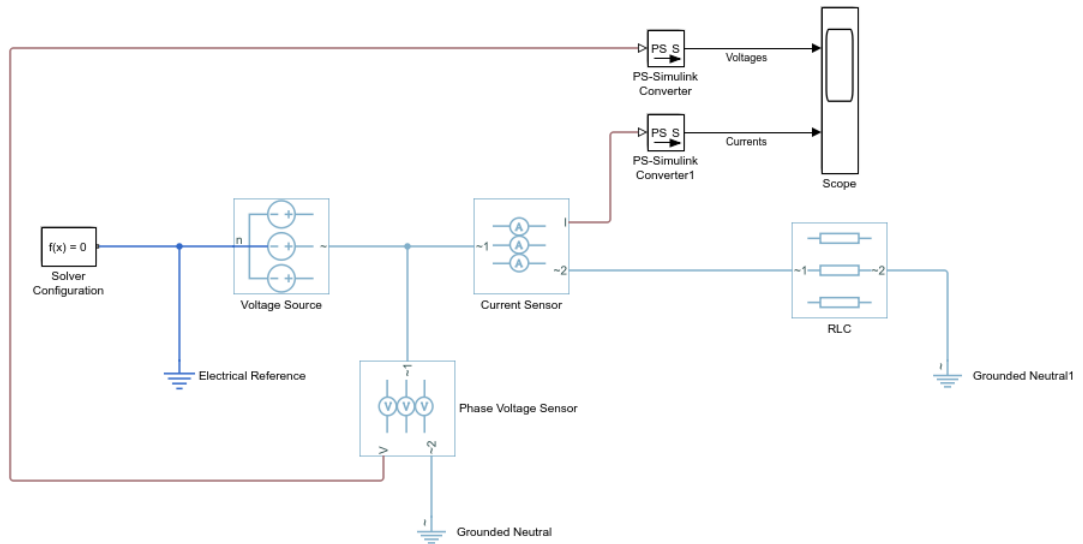
Select System Component Blocks

- 1 Open a blank model.
- 2 Add these blocks to the model.

Block	Purpose	Library Path	Quantity
Scope	Display phase voltages and currents for the three-phase system.	Simulink > Sinks	1
Electrical Reference	Provide the ground connection for electrical conserving ports.	Simscape > Foundation Library > Electrical > Electrical Elements	1
PS-Simulink Converter	Convert the physical signals to a Simulink signals.	Simscape > Utilities	2

Block	Purpose	Library Path	Quantity
Solver Configuration	Define solver settings that apply to all physical modeling blocks.	Simscape > Utilities	1
Grounded Neutral	Provide an electrical ground connection for each phase of the three-phase system.	Simscape > Power Systems > Simscape Components > Connections	2
RLC	Model the resistive, inductive, and capacitive properties of the three-phase load.	Simscape > Power Systems > Simscape Components > Passive Devices	1
Current Sensor	Convert the electrical current flowing in each phase of the three-phase load into a physical signal proportional to that current.	Simscape > Power Systems > Simscape Components > Sensors	1
Phase Voltage Sensor	Convert the voltage across each phase of the three-phase system into a physical signal proportional to that voltage.	Simscape > Power Systems > Simscape Components > Sensors	1
Voltage Source	Provide an ideal three-phase voltage source that maintains a sinusoidal voltage across its output terminals, regardless of the current flowing in the source.	Simscape > Power Systems > Simscape Components > Sources	1

- 3 Add a second input port to the Scope block.
 - a Right-click the Scope block.
 - b From the context menu, select **Signals & Ports > Number of Input Ports > 2**
- 4 Connect the blocks as shown.



- 5 Save the model using the name `simplethreephasemodel`.

The blocks in this model use composite three-phase ports. For more information, see “Three-Phase Ports” on page 2-6.

Specify Simulation Parameters

As with Simscape models, you must include a Solver Configuration block in each topologically distinct physical network. This model has a single physical network, so use one Solver Configuration block.

- 1 In the Solver Configuration block, select **Use local solver** and set **Sample time** to `0.0001`.

In Simscape-based models, the local solver is a sample-based solver that represents physical network states as discrete states. For most Simscape Power Systems Simscape Components models, the local solver is an appropriate first choice. The solver updates block states once per simulation time step, as determined by **Sample time**. For simulation of a 60-Hz AC system, an appropriate sample time is a value in the order of $1e-4$. For more information on solver options, see Solver Configuration.

If you prefer to use a continuous solver instead of a discrete solver, clear the **Use local solver** check box in the Solver Configuration block. The simulation then uses

the Simulink solver specified in the model configuration parameters (**Simulation > Model Configuration Parameters**). For Simscape Power Systems Simscape Components models, an appropriate solver choice is the moderately stiff solver ode23t. For a 60-Hz AC system, specify a value for **Max step size** in the order of $1e-4$. For more information, see “Variable-Step Continuous Explicit Solvers” (Simulink).

- 2 In the Simulink Editor, set the simulation **Stop time** to 0.1.

Load Impedance Parameters

The RLC block models resistive, inductive, and capacitive characteristics of the three-phase load. Using the **Component structure** parameter, you can specify a series or parallel combination of resistance, inductance, and capacitance.

In the RLC block, the defaults are:

- **Component structure** — R.
- **Resistance** — 1Ω .

Using the default **Component structure** value, R, models a three-phase load that is purely resistive in nature. The resistance in *each* phase is 1Ω .

Specify Display Parameters

Sensor blocks in the model convert the current and voltage in each phase of the three-phase system to proportional physical signals. PS-Simulink Converter blocks convert the physical signals into Simulink signals for the Scope block to display.


- 1 Of these three types of blocks, only the converter blocks have parameters. For this example:
 - Set **Output signal unit** of the PS-Simulink Converter block to A. This setting ensures that the block outputs a signal with the same magnitude as the ampere signal that enters it.
 - Set **Output signal unit** of the PS-Simulink Converter1 block to V. This setting ensures that the block outputs a signal with the same magnitude as the voltage signal that enters it.
- 2 Label the input signals to the Scope block. Double-click each line, and type the appropriate label, *Currents* or *Voltages*, as shown in the model graphic.

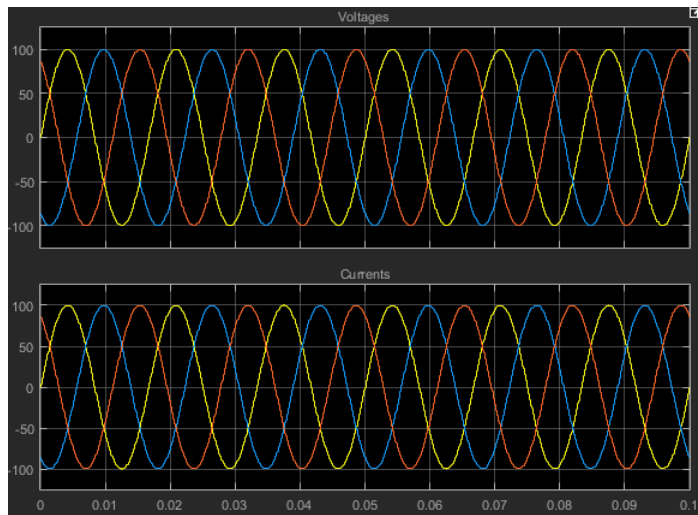
You are ready to simulate the model and analyze the results.

Save and Simulate the Model

- 1 Save the model.
- 2 Simulate the model. In the menu bar of the Simulink Explorer, click the **Run** button.

Analyze the Resistive Three-Phase Model Simulation Results

- 1 View the phase currents and voltages. Double-click the Scope block.
- 2 To scale the scope axes to the data, click the Autoscale button .




In this simulation, the **Component structure** parameter of the RLC block specifies that the electrical characteristics of the three-phase load are purely resistive. Therefore, for each phase of the three-phase system, the voltage and current remain in phase with each other. Because the resistance in each phase is 1Ω , the magnitude of the phase voltage is equal to the magnitude of the phase current.

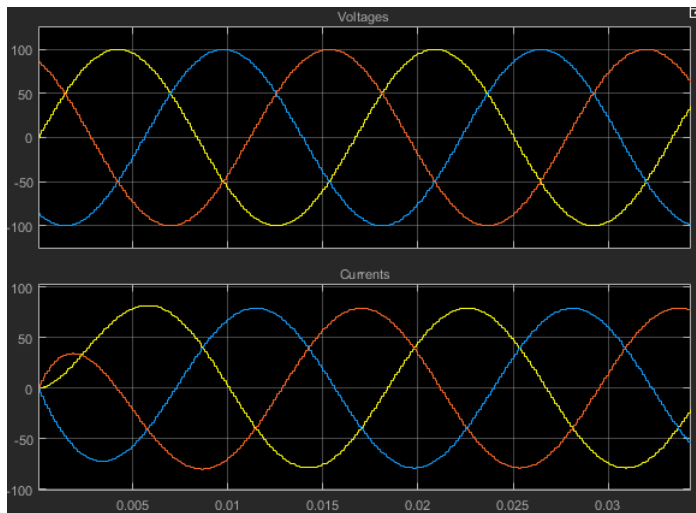
Create and Simulate a Reactive Three-Phase Load

This procedure shows you how to modify the model to create a reactive load. A reactive load has inductive or capacitive characteristics.

- 1 Save this version of the model using the name `simplethreephase_model_reactive`.
- 2 In the RLC block, set:
 - **Component structure** to Series RL
 - **Inductance** to 0.002
- 3 Simulate the model.

Analyze the Reactive Three-Phase Model Simulation Results

- 1 View the simulation results. Autoscale the scope axes.
- 2 Examine the results in closer detail. For example, click the Zoom button  and drag a box over the first third of one of the plots.



The electrical characteristics of three-phase load are no longer purely resistive. Because the load has an inductive characteristic, the current flowing in each phase lags the voltage.

See Also

Related Examples

- “Create and Simulate Expanded Balanced and Unbalanced Three-Phase Models” on page 3-9
- “Essential Power Engineering Modeling Techniques” on page 2-4

Create and Simulate Expanded Balanced and Unbalanced Three-Phase Models

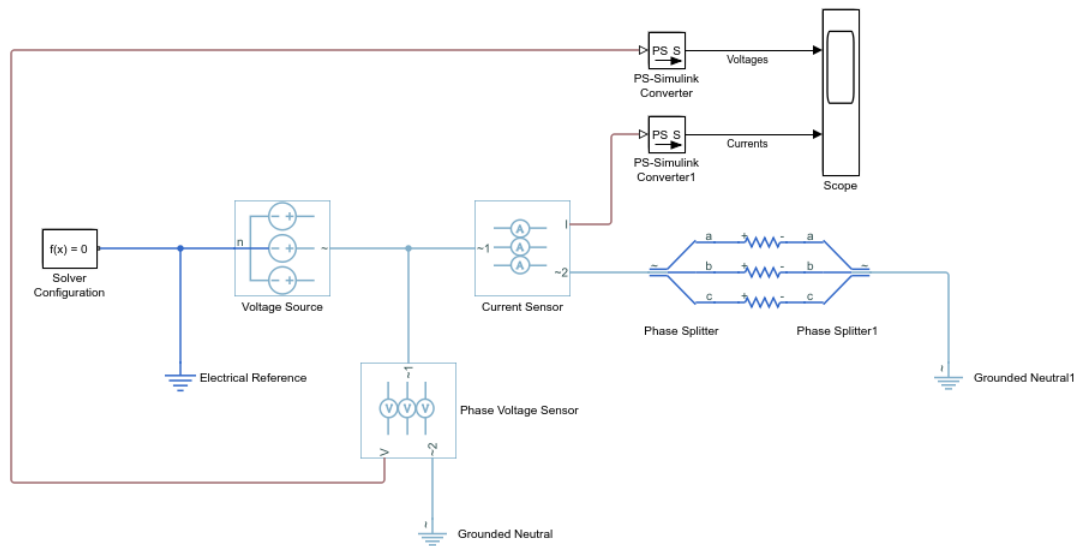
This procedure shows you how to modify a model that you built in “Build and Simulate Composite Resistive and Reactive Three-Phase Models” on page 3-2 to create:

- A three-phase load expanded into individual phases
- An expanded three-phase load that does not have equal resistance in each phase

In this procedure, you change the original model and save the changes as new models. You then simulate the new models and analyze the results.

Create an Expanded Balanced Three-Phase Model

- 1 Open Simple Three-Phase Model.
- 2 Delete the RLC block.
- 3 Drag two copies of the Phase Splitter block into the model from the **Simscape > Power Systems > Simscape Components > Connections** library.
- 4 Flip one of the Phase Splitter blocks horizontally. Right-click the block and select **Rotate & Flip > Flip Block > Left-Right**.
- 5 Drag a Resistor element into the model from the **Simscape > Foundation Library > Electrical > Electrical Elements** library.
- 6 To create space for more components, hide the Resistor element label. Right-click the resistor and select **Format > Show Block Name** to clear this option.
- 7 Make two more copies of the Resistor element.
- 8 Connect the components as shown.



- 9 Save this version of the modified model using the name `simplethreephase_model_expanded_balanced`.

This model name reflects that the load previously modeled by the RLC block is now expanded into individual phases. The load is still balanced, that is, there is equal resistance in each phase.


Create an Expanded Unbalanced Three-Phase Model

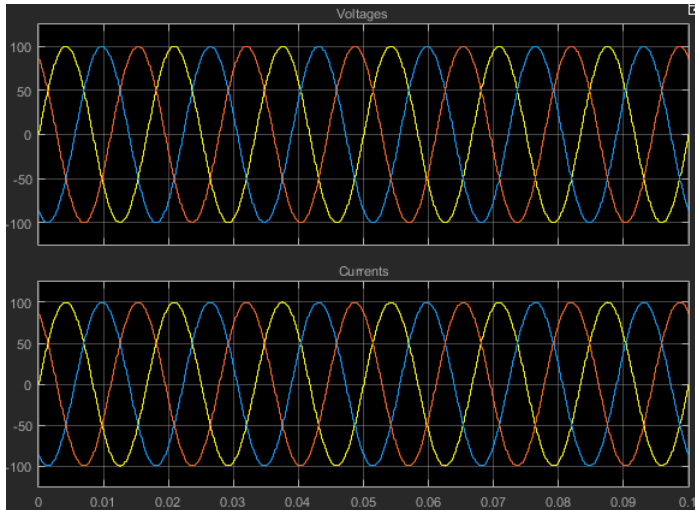
- 1 Unbalance the load by changing the resistance in one phase. Double-click one of the resistor elements. Change **Resistance** to 2.
- 2 Save this version of the modified model using the name `simplethreephase_model_expanded_unbalanced`.

This model name reflects that the three-phase load previously modeled by the RLC block is expanded into individual phases. The load is unbalanced, that is, the resistance in one of the phases is higher than in the other two.

Simulate the Models and Analyze Results

- 1 Simulate the `simplethreephase_model_expanded_balanced` model. In the menu bar of the Simulink Explorer, click the **Run** button.

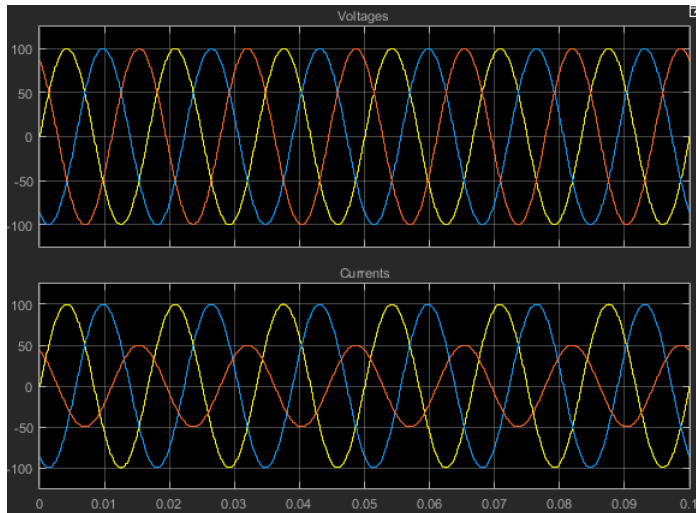
- 2 View the simulation results. Double-click the Scope block.
- 3 To scale the scope axes to the data, click the Autoscale button .



In “Build and Simulate Composite Resistive and Reactive Three-Phase Models” on page 3-2, the **Component structure** parameter of the RLC block specifies that the three-phase load is purely resistive. In this version of the model, the load is expanded into an individual resistive element for each phase, but the resistance in each phase is unchanged. For each phase of the three-phase system, the voltage and current remain in phase with each other. Because the resistance in each phase is 1Ω , the magnitude of the phase voltage is equal to the magnitude of the phase current.

Comparing these results with the results for the three-phase resistive model shows that a block with composite three-phase ports (the RLC block in the original model), produces results with the same fidelity as that of expanded phases.

- 4 Open the `simplethreephase_model_expanded_unbalanced` model.
- 5 Simulate the model. Autoscale the scope axes.



In this version of the model, one phase of the three-phase load has twice the resistance of the other two. Therefore, half as much current flows in that phase, as the second plot shows. However, because the load remains purely resistive, the voltage and current remain in phase with each other.

See Also

Related Examples

- “Build and Simulate Composite Resistive and Reactive Three-Phase Models” on page 3-2
- “Essential Power Engineering Modeling Techniques” on page 2-4

Modeling Machines

- “Machine Parameterization” on page 4-2
- “Per-Unit Conversion for Machine Parameters” on page 4-4
- “Machine Plotting and Display Options” on page 4-6
- “Initialize Synchronous Machines and Controllers” on page 4-8

Machine Parameterization

In Simscape Power Systems Simscape Components software, asynchronous machines are parameterized using fundamental parameters. Each synchronous machine is parameterized using standard or fundamental parameters.

Machine fundamental parameters include the values of inductances and resistances of the stator and rotor d - and q -axis equivalent circuits. These parameters fully specify the electrical characteristics of the machine, but you cannot determine them directly from machine test responses. Hence, it is more common to parameterize a synchronous machine using a standard parameter set. You can obtain the standard parameters by observing responses at the machine terminals with suitable tests scenarios.

You can tell the parameter set a block uses because the block name includes the parameter set name, e.g. Asynchronous Machine Squirrel Cage (fundamental). The parameters you can set in the block dialog box correspond to the parameterization type.

If a machine block has standard and fundamental variants, base your block choice on the parameters you are most familiar with or you have available. Standard block variants use classical equations to convert standard parameter values that you enter to fundamental parameter values for use at run time.

If a machine block has an SI and a per-unit variant, base your block choice on the parameters you have available. For machine blocks that are SI variants, you enter the number of pole pairs and the SI values for the nominal voltage, power, and frequency on the main tab of the dialog box. You also enter SI values for the resistance and reactance parameters on the impedance tab, and for the magnetic flux linkage parameters on the initial condition tab. The block uses classical equations to calculate per-unit base values from the parameters on the main tab. It expresses the resistance, inductance, and magnetic flux linkage parameters as per-unit ratios of the SI values (resistance, reactance, and magnetic flux linkage) and the base values for use at run time.

The field circuit and rotational ports of machine blocks use SI units. However, the pu measurement port of machine blocks outputs a vector of physical signals in per-unit.

See Also

More About

- “Per-Unit System of Units” on page 1-7
- “Per-Unit Conversion for Machine Parameters” on page 4-4

Per-Unit Conversion for Machine Parameters

In this section...
“Impedance Conversion Equations” on page 4-4
“Magnetic Flux Linkage Conversion Equations” on page 4-4

Impedance Conversion Equations

For machine impedance parameters (resistance, inductance, and reactance), the relationships between SI and per-unit values are defined by these equations:

$$R = \frac{R_{(SI)}}{R_{base}}$$

$$L = X = \frac{X_{(SI)}}{X_{base}}$$

where:

- $R_{(SI)}$ is the resistance, expressed in Ω .
- R_{base} is the per-unit base resistance, expressed in Ω .
- R is the per-unit resistance.
- $X_{(SI)}$ is the reactance, expressed in Ω .
- X_{base} is the per-unit base reactance, expressed in Ω .
- X is the per-unit reactance.
- L is the per-unit inductance.

Magnetic Flux Linkage Conversion Equations

For machine magnetic flux linkage parameters, the relationship between SI and per-unit values is defined by

$$\psi = \frac{\psi_{(SI)}}{\psi_{base}}$$

where:

- $\psi_{(SI)}$ is the magnetic flux linkage, expressed in Wb.
- ψ_{base} is the per-unit base magnetic flux linkage, expressed in Wb.
- ψ is the per-unit magnetic flux linkage.

See Also

More About

- “Per-Unit System of Units” on page 1-7

Machine Plotting and Display Options

Use the **Power Systems** menu on the block context menu to perform plotting and display actions for certain blocks in the Simscape Power Systems Simscape Components Machines sublibrary. For example, you can plot torque versus speed for the Asynchronous Machine Wound Rotor block, either in SI or per-unit units.

Using other options on the **Power Systems** menu, you can display values in per-unit or display base parameter values in the MATLAB Command Window. These options enable you to initialize and tune your three-phase machine quickly.

Asynchronous Machine Options

- **Display Base Values** displays the machine per-unit base values in the MATLAB Command Window.
- **Plot Torque Speed (SI)** plots torque versus speed (both measured in SI units) in a MATLAB figure window using the present machine parameters.
- **Plot Torque Speed (pu)** plots torque versus speed (both measured in per-unit) in a MATLAB figure window using the present machine parameters.

Synchronous Machine Options

All synchronous machine block context menus contain these display options:

- **Display Base Values** displays the machine per-unit base values in the MATLAB Command Window.
- **Display Associated Base Values** displays associated per-unit base values in the MATLAB Command Window.
- **Display Associated Initial Conditions** displays associated initial conditions in the MATLAB Command Window.

The round rotor and salient pole synchronous machine block context menus also contain these plotting options;

- **Plot Open-Circuit Saturation (pu)** plots air-gap voltage, V_{ag} , versus field current, i_{fd} , both measured in per-unit, in a MATLAB figure window. The plot contains three traces:
 - Unsaturated: **Stator d-axis mutual inductance (unsaturated), L_{ad}** you specify

- Saturated: **Per-unit open-circuit lookup table (Vag versus ifd)** you specify
- Derived: Open-circuit lookup table (per-unit) derived from the **Per-unit open-circuit lookup table (Vag versus ifd)** you specify. This data is used to calculate the saturation factor, K_s , versus magnetic flux linkage, ψ_{at} , characteristic.
- **Plot Saturation Factor (pu)** plots saturation factor, K_s , versus magnetic flux linkage, ψ_{at} , both measured in per-unit, in a MATLAB figure window using the present machine parameters. This value is derived from parameters you specify:
 - **Stator d-axis mutual inductance (unsaturated), Ladu**
 - **Per-unit field current saturation data, ifd**
 - **Per-unit air-gap voltage saturation data, Vag**

Machine Inertia Block Options

For the Machine Inertia block, you can display the inertia parameters and base values using the **Power Systems** menu on the block context menu. The block displays parameter values in the MATLAB Command Window.

Initialize Synchronous Machines and Controllers

In Simscape Power Systems Simscape Components software, you can specify steady-state power and voltage values for a synchronous machine. Based on the values you specify, the machine block calculates the initial field circuit and rotational input values required to achieve this steady state. Starting a machine at steady state prevents undesired transient effects in your simulation.

- 1 Calculate the required power and voltage characteristics of your load circuit.
- 2 In the **Initial Conditions** tab of the dialog box, set **Specify initialization by** to **Electrical power and voltage output**.
- 3 Enter the required power and voltage values and click **OK**.
- 4 Right-click the machine block and select **Power Systems > Display Associated Initial Conditions**.

Simscape Power Systems calculates the field circuit and rotational port values required to start the machine in steady state and displays them in the MATLAB Command Window.

- 5 Use these values to input parameters to the blocks connected to the field circuit and rotational ports of the synchronous machine.

Note If you set **Specify initialization by** to **Mechanical** and **magnetic states**, Simulink does not calculate the associated initial conditions for the machine.

Customization

- “Build Custom Blocks Using the Three-Phase Electrical Domain” on page 5-2
- “Customizing Machine Models” on page 5-4
- “Custom Synchronous Machine” on page 5-6

Build Custom Blocks Using the Three-Phase Electrical Domain

In addition to the Simscape Foundation domains, Simscape Power Systems Simscape Components software contains a three-phase electrical domain. You can use this domain to develop your own custom three-phase blocks using Simscape language.

The three-phase electrical domain declaration is shown.

```
domain electrical
    % Three-Phase Electrical Domain

    % Copyright 2012-2013 The MathWorks, Inc.

    parameters
        Temperature = { 300.15 , 'K'      }; % Circuit temperature
        GMIN         = { 1e-12  , '1/Ohm' }; % Minimum conductance, GMIN
    end

    variables
        V = { [ 0 0 0 ], 'V' };
    end

    variables(Balancing = true)
        I = { [ 0 0 0 ], 'A' };
    end

end
```

It contains the following variables and parameters:

- Across variable V (voltage), declared as a three-element row vector, in volts
- Through variable I (current), declared as a three-element row vector, in amperes
- Parameter $Temperature$, specifying the circuit temperature
- Parameter $GMIN$, specifying minimum conductance

To refer to this domain in your custom component declarations, use the following syntax:

```
pe.electrical.three_phase.electrical
```

See Also

Related Examples

- “Custom Synchronous Machine” on page 5-6

More About

- “Customizing Machine Models” on page 5-4
- “Custom Components” (Simscape)
- “Foundation Domains” (Simscape)

Customizing Machine Models

The ThreePhaseExamples library, included in the product examples, contains the following custom three-phase components:

- Permanent Magnet Synchronous Motor
- Synchronous Machine
- Synchronous Machine (simplified)
- Zigzag Transformer

You can use these simplified example models to write your own machine and transformer component files.

To open the custom library, at the MATLAB command prompt, type `ThreePhaseExamples_lib`. Double-click any block in the library to open its dialog box, and then click the **Source code** link in the block dialog box to open the Simscape source file for this block in the MATLAB Editor.

To customize the block for your application, edit the source file and save it in a package directory.

For example, you can create a folder called `+MyMachines` and save the source files for your customized machines in this folder. Create this folder in your working directory, or in another directory that is on the MATLAB path. Running the `ssc_build` command on this package generates the `MyMachines_lib` library model. This library contains all your custom machine blocks and is located in the same directory where you have created the `+MyMachines` folder. Open the `MyMachines_lib` library by double-clicking it or by typing its name at the MATLAB command prompt.

For more information on packaging and deploying Simscape component files, see “Building Custom Block Libraries” (Simscape).

Things to keep in mind when writing component files:

- If you create a custom component by modifying an existing one, do not forget to change the name of the component and the name of the resulting block.
- The component name must be the same as the name of the Simscape file. For example, if you plan to save your component in a file called `MyComponent.ssc`, change the declaration line in the file:

```
component MyComponent
```

- The comment line immediately following the component declaration (that is, the first line beginning with the % character) defines the name of the block, as it appears in the custom library next to the block icon and at the top of the block dialog box. If you do not specify this comment, then the component name serves as the block name. The block name must be unique within the subpackage (sublibrary) where it resides.
- Additional comments, below the line specifying the block name, are interpreted as the block description. You do not have to modify them when copying an existing file, but if you change the way the component works, it makes sense to reflect the change in the block description. The block description is for informational purposes only.
- When modifying component equations, if you introduce additional terms, make sure to add the appropriate variables or parameters to the component declaration section. For example, if you add zero-sequence dynamics to the component equations, declare an additional parameter for stator zero-sequence inductance, L_0 , and an additional variable for the initial stator zero-sequence magnetic flux linkage.

The “Custom Synchronous Machine” on page 5-6 tutorial shows how you can modify the Synchronous Machine component file and customize it for use in your applications. For more information on writing customized component files, see “Custom Components” (Simscape).

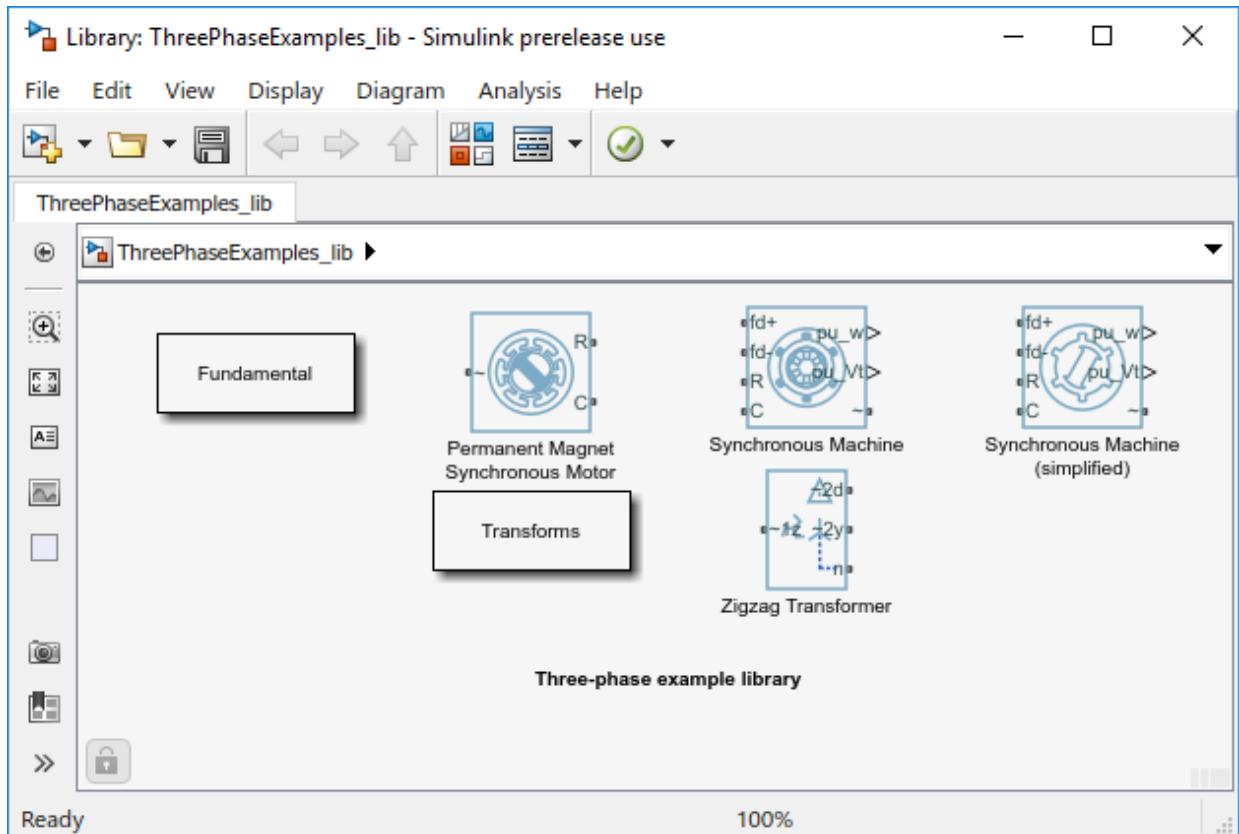
Custom Synchronous Machine

The ThreePhaseExamples library, included in the product examples, contains simplified example models that you can use to write your own machine and transformer component files. The Synchronous Machine component in the ThreePhaseExamples library is similar to the Synchronous Machine Round Rotor (fundamental) block, but its equations have been simplified to omit zero-sequence dynamics. The Synchronous Machine block is therefore suitable for balanced operation only.

This example shows how you can further simplify the component file and make a custom machine block that does not account for the stator rate of change of flux.

- 1** In your working directory, create a folder called `+MyMachines`. This folder will contain the source files for your customized machines.
- 2** To open the library of simplified component examples, at the MATLAB command prompt, type:

```
ThreePhaseExamples_lib
```

- 3 Double-click the Synchronous Machine block.
- 4 In the block dialog box, click the **Source code** link.

The Simscape source file for this block opens in the MATLAB Editor.

- 5 Change the name of the component, the name of the block, and the block description by replacing these lines of the file:

```

component sm
% Synchronous Machine
% Synchronous machine (SM) with a round rotor parameterized using
% fundamental per-unit parameters. The SM model includes field and
% damper windings on the d-axis and two damper windings on the q-axis.
% The stator d.psi/dt terms are included, as is the effect of speed
% variation on the stator voltages. The defining equations are

```

```
% simplified by omitting the zero-sequence dynamics: the model is  
% suitable for balanced operation.
```

```
% Copyright 2012-2014 The MathWorks, Inc.
```

with:

```
component sm1  
% Simplified Synchronous Machine  
% This synchronous machine does not include the stator d.psi/dt terms.
```

- 6 To remove the stator rate of change of flux terms, scroll down to the equations section and modify the stator voltage equations from:

```
% Per unit stator voltage equations  
pu_ed == oneOver0mega*pu_psid.der - pu_psiq*pu_velocity - Ra*pu_id;  
pu_eq == oneOver0mega*pu_psiq.der + pu_psid*pu_velocity - Ra*pu_iq;
```

to:

```
% Per unit stator voltage equations  
pu_ed == -pu_psiq*pu_velocity - Ra*pu_id;  
pu_eq == pu_psid*pu_velocity - Ra*pu_iq;
```

- 7 Save the file in the +MyMachines folder as sm1.ssc. The name of the Simscape file must match the component name.
- 8 To generate the custom library containing the new block, at the MATLAB command prompt, type:

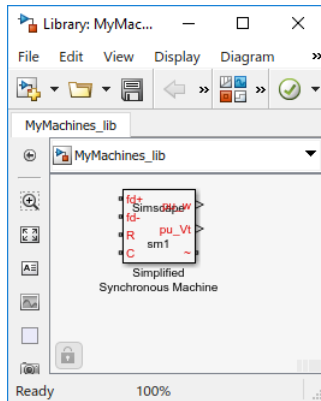
```
ssc_build
```

This command generates the MyMachines_lib library model in your working directory.

- 9 To open the custom library, at the MATLAB command prompt, type:

```
MyMachines_lib
```

The library contains the Simplified Synchronous Machine block, which you can now use in your models.



See Also

`ssc_build`

Related Examples

- “Building Custom Block Libraries” (Simscape)

More About

- “Customizing Machine Models” on page 5-4
- “Build Custom Blocks Using the Three-Phase Electrical Domain” on page 5-2
- “Custom Components” (Simscape)
- “Customizing the Block Name and Appearance” (Simscape)
- “Component Equations” (Simscape)

Control

Tune an Electric Drive

In this section...

“Cascade Control Structure” on page 6-2

“Equations for PI Tuning Using the Pole Placement Method” on page 6-2

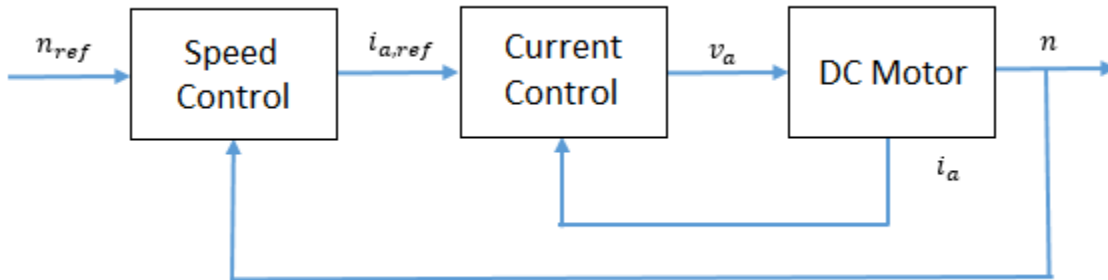
“Equations for DC Motor Controller Tuning” on page 6-6

“Tune the Electric Drive in the Example Model” on page 6-8

This example shows how to tune an electric drive using a cascade control structure.

Cascade Control Structure

The figure shows a feedback control loop that uses a cascade control structure. The outer speed-control loop is slower acting than the inner current-control loop.



Equations for PI Tuning Using the Pole Placement Method

To satisfy the required control performance for a simple discrete plant model, $G_f(z^{-1})$, use a closed loop PI control system $G_{PI}(z^{-1})$. The transient performance can be expressed in terms of the overshoot. The overshoot decreases relative to the damping factor:

$$\sigma = e^{\frac{-\pi\xi}{\sqrt{1-\xi^2}}}$$

where,

- σ is overshoot.
- ξ is the damping factor.

The response time, t_r , depends on the damping and the natural frequency, ω_n , such that:

- If $\xi < 0.7$,

$$t_r \cong \frac{4}{\omega_n \xi}.$$

- If $\xi \geq 0.7$,

$$t_r \cong \frac{6\xi}{\omega_n}.$$

The general workflow for designing a PI controller for a first-order system is:

- 1 Discretize the plant model using the zero-order hold (ZOH) discretization method. That is, given that the first-order equation representing the plant is

$$G(s) = \frac{K_m}{T_m s + 1},$$

where,

- K_m is the first-order gain.
- T_m is time constant of the first-order system.

Setting

$$s = \frac{1 - z^{-1}}{z^{-1} T_s},$$

yields the discrete plant model,

$$G(z^{-1}) = \frac{K_m \left(\frac{T_s}{T_m} \right) z^{-1}}{1 + \left(\frac{T_s - T_m}{T_m} \right) z^{-1}} = \frac{b_1 z^{-1}}{1 + a_1 z^{-1}},$$

where T_s is sample time for the discrete-time controller.

- 2 Write a discrete-time representation for the PI controller using the same transform. For

$$G_{PI}(s) = K_P + K_I \left(\frac{1}{s} \right),$$

setting

$$s = \frac{1 - z^{-1}}{z^{-1} T_s},$$

yields the discrete controller model,

$$G_{PI}(z^{-1}) = \frac{K_P + (K_I T_s - K_P) z^{-1}}{1 - z^{-1}} = \frac{q_0 + q_1 z^{-1}}{1 - z^{-1}}.$$

Combining the discrete equations for the plant and the controller yields the closed loop transfer function for the system,

$$G_0(z^{-1}) = \frac{q_0 b_1 z^{-1} + q_1 b_1 z^{-2}}{1 + (a_1 - 1 + q_0 b_1) z^{-1} + (-a_1 + q_1 b_1) z^{-2}},$$

The denominator of the transfer function is the characteristic polynomial. That is,

$$P_{c0}(z^{-1}) = 1 + (a_1 - 1 + q_0 b_1) z^{-1} + (-a_1 + q_1 b_1) z^{-2}.$$

- 3 The characteristic polynomial for achieving the required performance is defined as

$$P_{cd}(z^{-1}) = 1 + \alpha_1 z^{-1} + \alpha_2 z^{-2},$$

where,

-
- $$\alpha_1 = -2e^{-\xi\omega_n T_s} \cos\left(\omega_n T_s \sqrt{1-\xi^2}\right).$$
- $$\alpha_2 = e^{-2\xi\omega_n T_s}.$$

- 4 To determine the controller parameters, set the characteristic polynomial for the system equal to the characteristic polynomial for the required performance. If

$$P_{c0}(z^{-1}) = P_{cd}(z^{-1}),$$

then

$$\alpha_1 = a_1 - 1 + q_0 b_1$$

and

$$\alpha_2 = -a_1 + q_1 b_1.$$

Solving for q_0 and q_1 yields

$$q_0 = \frac{\alpha_1 - a_1 + 1}{b_1}$$

and

$$q_1 = \frac{\alpha_2 + a_1}{b_1}.$$

Therefore, the general equations for the proportional and integral control parameters for the first-order system are

$$K_P = q_0$$

and

$$K_I = \frac{q_1 + K_P}{T_s}.$$

Equations for DC Motor Controller Tuning

Assuming that, for the system in the example model, $K_b = K_t$, the simplified mathematical equations for voltage and torque of the DC motor are

$$v_a = L_a \frac{di_a}{dt} + R_a i_a + K_b \omega$$

and

$$T_e = J_m \frac{d\omega}{dt} + B_m \omega + T_{load} = K_b i_a,$$

where:

- v_a is the armature voltage.
- i_a is the armature current.
- L_a is the armature inductance.
- R_a is the armature resistance.
- ω is the rotor angular velocity
- T_e is the motor torque.
- T_{load} is the load torque.
- J_m is the rotor moment of inertia.
- B_m is the viscous friction coefficient.
- K_b is a constant of proportionality.

To tune the current controller, assume that the model is linear, that is, that the back electromotive force, as represented by $K_b \omega$, is negligible. This assumption allows for an approximation of the plant model using this first-order Laplace equation:

$$G_I(s) = \frac{1}{R_a} \frac{1}{\left(\frac{L_a}{R_a}\right)s + 1}.$$

Given the system requirements, you can now solve for K_P and K_I . The requirements for the current controller in the example model are:

- Sample time, $T_s = 1$ ms.
- Overshoot, $\sigma = 5\%$.
- Response time, $t_r = 0.11$ s.

Therefore, the proportional and integral parameters for the current controller are:

- $K_P = 7.7099$.
- $K_I = 455.1491$.

To tune the speed controller, approximate the plant model with a simple model. First assume that the inner loop is much faster than the outer loop. Also assume that there is no steady-state error. These assumptions allow for the use a first-order system by considering a transfer function of 1 for the inner current loop.

To output rotational velocity in revolutions per minute, the transfer function is multiplied by a factor of $30/\pi$. To take as control input the armature current instead of the motor torque, the transfer function is multiplied by the proportionality constant, K_b . The resulting approximation for the outer-loop plant model is

$$G_n(s) = \frac{\frac{30K_b}{\pi B_m}}{\left(\frac{J_m}{B_m}\right)s + 1}.$$

The speed controller has the same sample time and overshoot requirements as the current controller, but the response time is slower, such that:

- Sample time $T_s = 1$ ms.
- Overshoot $\sigma = 5\%$.
- Response time $t_r = 0.50$ s.

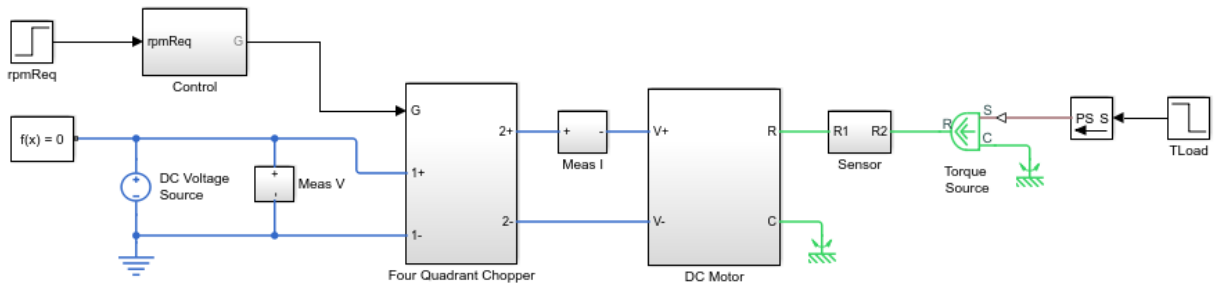
Therefore, the proportional and integral parameters for the speed controller are:

- $K_P = 0.0045$
- $K_I = 0.0405$

Tune the Electric Drive in the Example Model

- 1 Explore the models of the DC motor and the cascaded controller.
 - a Open the model. At the MATLAB command prompt, enter

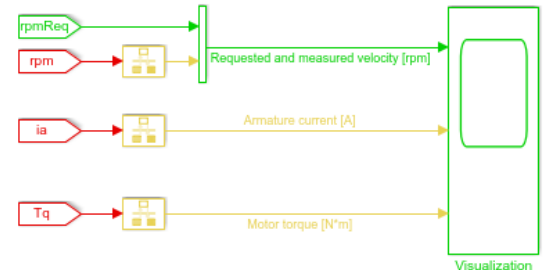
```
model = 'pe_dc_motor_control'
open_system(model)
```



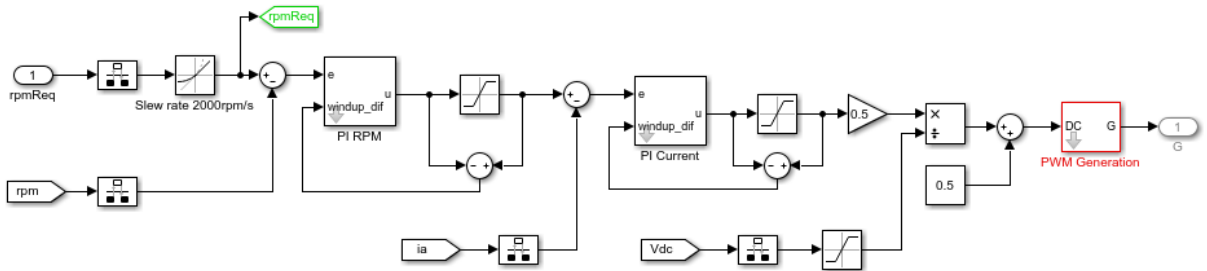
DC Motor Control

This example shows a cascade speed-control structure for a DC motor. A PWM controlled four-quadrant Chopper is used to feed the DC motor. The Control subsystem includes the outer speed-control loop, the inner current-control loop, and the PWM generation. The total simulation time (t) is 4 seconds. At $t = 1.5$ seconds, the load torque increases. At $t = 2.5$ seconds, the reference speed is changed from 1000 rpm to 2000 rpm.

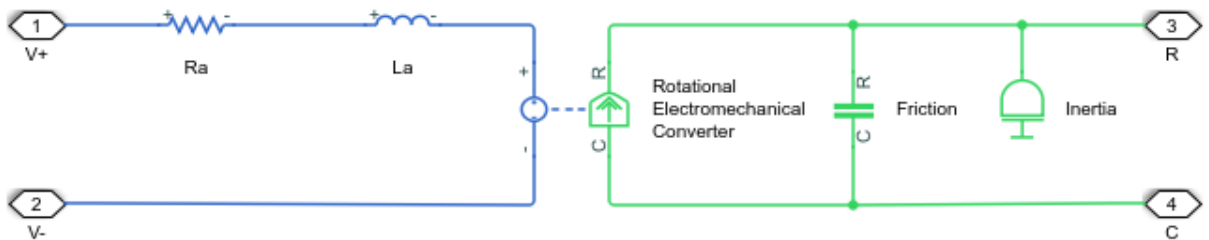
1. [Edit associated parameterization script.](#)
2. [Explore simulation results](#) using `sscexplore`.
3. [Learn more](#) about this example.



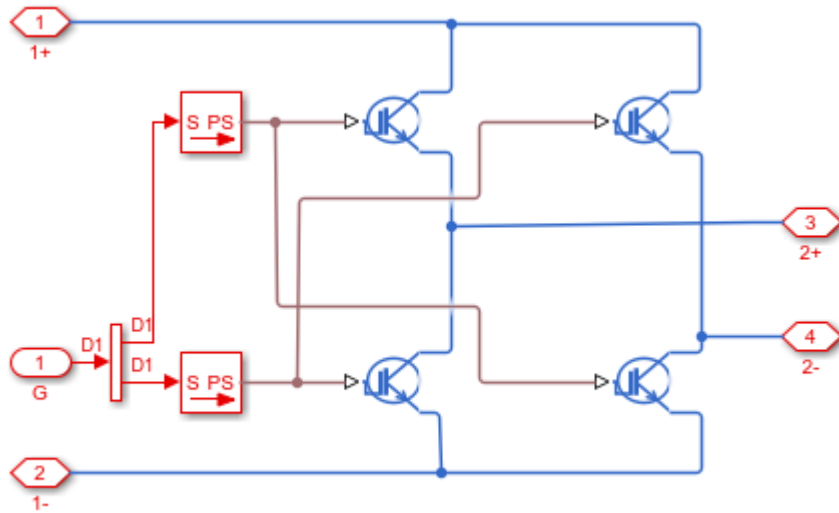
- b The **Control** subsystem contains the model of the cascaded control system built using blocks from the Simulink library.



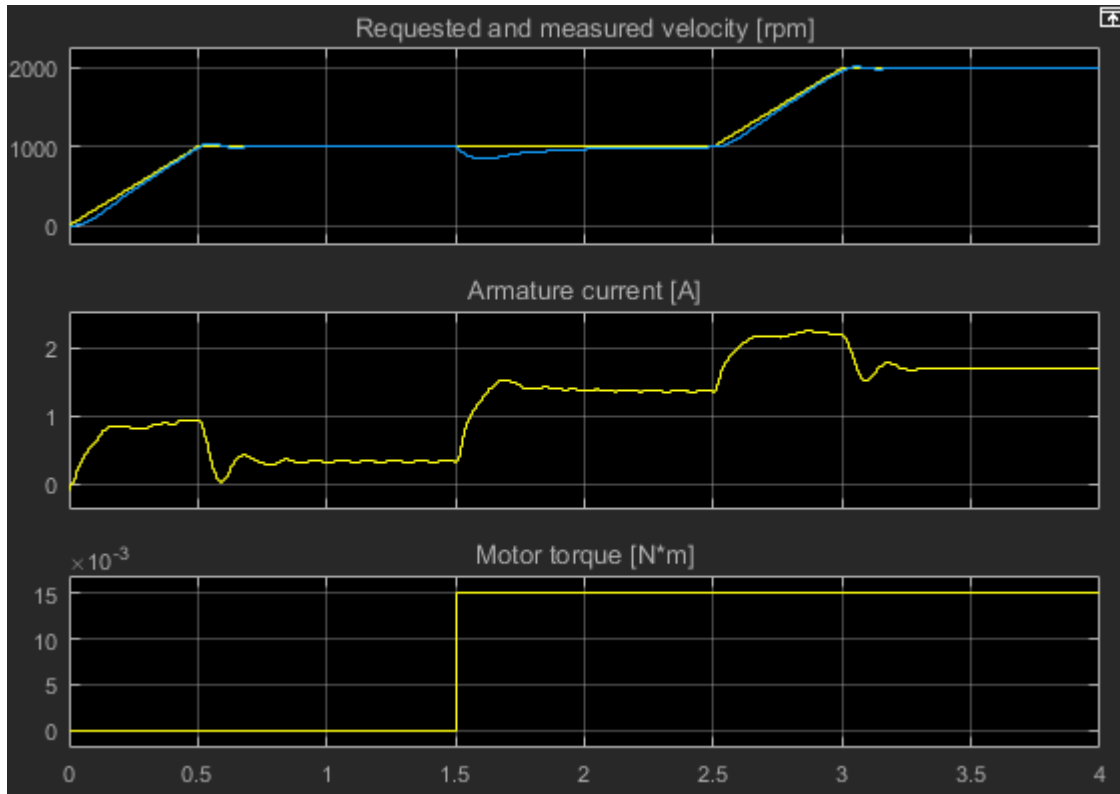
- c The **DC Motor** subsystem contains a simple DC motor model built using blocks from the Simscape library.



- d The Four Quadrant Chopper subsystem contains four IGBT blocks. When the input voltage exceeds the threshold of 0.5 V , the IGBT blocks behave like linear diodes with a forward-voltage of 0.8 V and a resistance of $1e-4 \text{ ohm}$. When the threshold voltage is not exceeded, the IGBT blocks act like linear resistors with an off-state conductance of $1e-5 \text{ 1/ohm}$.



- 2 Simulate the model.
`sim(model)`
- 3 View the results. Open the **Scope** block.



At 1.5 seconds, there is a load torque that results in a steady-state error.

- 4 Tune the DC motor controller. The `pe_getDCMotorFirstOrderPIParams` function calculates the proportional gain, K_p , and the integral gain, K_i , for the first-order system in this example.

The function syntax is `[Kp, Ki] = getParamPI(Km,Tm,Ts,sigma,tr)`.

The input arguments for the function are the system parameters and the requirements for the controller:

- K_m is the first-order gain.
- T_m is the time constant of the first-order system.
- T_s is the sample time for the discrete-time controller.

- σ is the desired maximum overshoot, σ .
 - t_r is the desired response time.
- a** To examine the equations in the function, enter
- edit `pe_getDCMotorFirstOrderPIParams`
- b** To calculate the controller parameters using the function, save these system parameters to the workspace:

```
Ra=4.67;           % [Ohm]
La=170e-3;        % [H]
Bm=47.3e-6;       % [N*m/(rad/s)]
Jm=42.6e-6;       % [Kg*m^2]
Kb=14.7e-3;       % [V/(rad/s)]
Tsc=1e-3;         % [s]
```

- c** Calculate the parameters for tuning the current controller as a function of the parameters and requirements for the inner controller:

- $K_m = 1/R_a$.
- $T_m = L_a/R_a$.
- $T_s = T_{sc}$.
- $\sigma = 0.05$.
- $T_r = 0.11$.

```
[Kp_i, Ki_i] = pe_getDCMotorFirstOrderPIParams(1/Ra,La/Ra,Tsc,0.05,0.11)
```

```
Kp_i =
```

```
7.7099
```

```
Ki_i =
```

```
455.1491
```

The gain parameters for the current controller are saved to the workspace.

- d** Calculate the parameters for tuning the speed controller based on the parameters and requirements for the outer controller:
- $K_m = K_b \cdot (30/\pi)$.

- $T_m = J_m/R_a$.
- $T_s = T_{sc}$.
- $\sigma = 0.05$.
- $T_r = 0.5$.

```
[Kp_n, Ki_n] = pe_getDCMotorFirstOrderPIParams((Kb*(30/pi))/Bm, Jm/Bm, Tsc, 0.05, 0.5)
```

```
Kp_n =
```

```
0.0045
```

```
Ki_n =
```

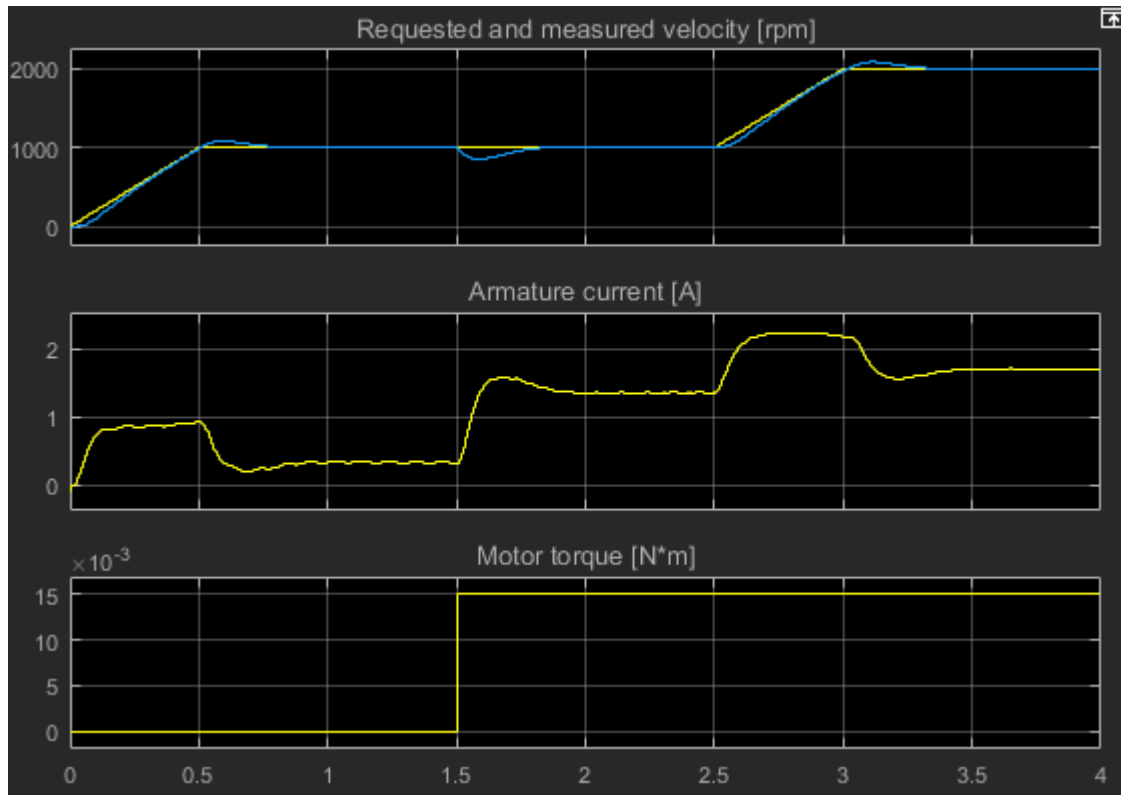
```
0.0405
```

The gain parameters for the speed controller are saved to the workspace.

- 5 Simulate the model using the saved gain parameters for the speed and controllers.

```
sim(model)
```

- 6 View the results. Open the **Scope** block.



There is slightly more overshoot, however, the controller responds much faster to the load torque change.

See Also

Inertia | Rotational Electromechanical Converter | Rotational Friction

Related Examples

- "DC Motor Control"

Simulation and Analysis of Power Engineering Systems

- “Simulating Power Engineering Systems” on page 7-2
- “Examine the Simulation Data-Logging Configuration of a Model” on page 7-3
- “Simulate Thermal Losses in Semiconductors” on page 7-5
- “Perform a Power-Loss Analysis” on page 7-16
- “Choose a Simscape Power Systems Function for an Offline Harmonic Analysis” on page 7-25
- “Perform an Online Harmonic Analysis Using the Simscape Spectrum Analyzer Block” on page 7-29
- “Optimize Block Settings for Simulations that Use the Partitioning Solver” on page 7-38
- “Prepare Simscape Power Systems Models for Real-Time Simulation Using Simscape Checks” on page 7-49
- “Phasor-Mode Simulation in Simscape Components” on page 7-51

Simulating Power Engineering Systems

Simscape Power Systems Simscape Components models are Simscape block diagrams refined for modeling and simulating three-phase electrical power systems. Therefore, Simscape Power Systems Simscape Components and Simscape simulations behave in the same way. In addition, Simscape techniques for simulation setup and troubleshooting apply to Simscape Power Systems Simscape Components models.

To learn about:

- The simulation behavior of Simscape models, see “How Simscape Simulation Works” (Simscape).
- Techniques for finding system operating points and linearizing the response of Simscape models, see “Trimming and Linearization” (Simscape).
- Troubleshooting Simscape simulations, see “Troubleshooting” (Simscape).

Examine the Simulation Data-Logging Configuration of a Model

Many analyses that you can perform using Simscape Power Systems require a simulation log variable in your MATLAB workspace. The model in this example is configured to log Simscape data for the whole model for the entire simulation time. To examine the data-logging configuration of a model:

- 1 Open the model. At the MATLAB command prompt, enter

```
model = 'pe_rectifier_power_dissipated';  
open(model)
```

- 2 Open the model configuration parameters and then, in the left pane, select **Simscape**. Relevant parameters are:

- **Log simulation data** — Data logging is enabled for the whole model because this parameter is set to All so you can calculate the power dissipated by any of the semiconductors in the model.
- **Workspace variable name** — This parameter, which is also referred to as the name of the simulation log variable, is specified as `simlog_pe_rectifier_power_dissipated`.
- **Limit data points** — You can calculate the power dissipated for the entire simulation time because the option is not selected.

Alternatively, you can determine the Simscape data-logging configuration without opening the model configuration parameters, by using the `get_param` function. For example, for the `pe_rectifier_power_dissipated` model, to determine:

- If all, some, or no data is logged, at the MATLAB command prompt, enter

```
get_param(model, 'SimscapeLogType')
```

```
ans =
```

```
    'all'
```

- The name of the Simscape logging variable

```
get_param(model, 'SimscapeLogName')
```

```
ans =  
    'simlog_pe_rectifier_power_dissipated'  
• If the option to limit data-points is on or off  
get_param(model, 'SimscapeLogLimitData')  
ans =  
    'off'
```

See Also

Functions

get_param

Related Examples

- “Data Logging” (Simscape)

Simulate Thermal Losses in Semiconductors

In this section...

“Prerequisite” on page 7-5

“Thermal Variants” on page 7-5

“Thermal Blocks” on page 7-5

“Thermal Ports” on page 7-6

“Thermal-Modeling Parameters” on page 7-7

“Limitations” on page 7-7

“Model Thermal Losses for a Rectifier” on page 7-8

Prerequisite

This example requires a simulation log variable in your MATLAB workspace. The model in this example is configured to log Simscape data for the whole model for the entire simulation time.

To learn how to determine if a model is configured to log simulation data, see “Examine the Simulation Data-Logging Configuration of a Model” on page 7-3.

Thermal Variants

Thermal modeling provides data that helps you to estimate cooling requirements for your system. The nonideal blocks in the Simscape Power Systems Simscape Components Semiconductors library have thermal variants that allow you to determine device temperatures by simulating heat generation. For example, the IGBT block, which models a three-terminal semiconductor device, has thermal variants that can simulate the heat generated by switching events and conduction losses. Selecting a thermal variant for a block adds a thermal port to the block and enables the associated thermal-modeling parameters.

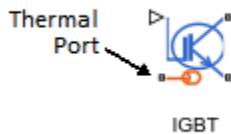
Thermal Blocks

In the Semiconductors library, the Fundamental Components sublibrary includes a Thermal sublibrary of blocks that allow you to model heat transfer using thermal variants:

- Cauer Thermal Model Element — A thermal component that, in a series connection, models heat transfer as a function of the thermal characteristics of the individual physical components and materials, for example, a chip, solder, and base that make up a semiconductor.
- Foster Thermal Model — A thermal component that models heat transfer as a function of the thermal characteristics of a semiconductor.
- Thermal Resistor — A thermal interface resistance component that models conductive heat transfer through a layer of material. Use the Thermal Resistor block to parameterize heat transfer using the thermal resistance value of the material.

Thermal Ports

Thermal ports are physical conserving ports in the Simscape thermal domain. Thermal ports on Simscape Power Systems Simscape Components semiconductors are associated with temperature and heat flow. The figure shows a thermal port on a thermal variant of the IGBT block.



Thermal ports are associated with temperature and heat flow which are the Across and Through variables of the Simscape thermal domain. To measure thermal variables, you can use one or both of these methods:

- 1 Log simulation data using a Simscape logging node. View the data using the `sscexplore` function.
- 2 Add a sensor from the **Simscape > Foundation Library > Thermal > Thermal Sensors** library to your model. To measure temperature, use a parallel-connected Ideal Temperature Sensor block. To measure heat flow, use a series-connected Ideal Heat Flow Sensor block.

There are several advantages to using data logging for desktop simulation. Data logging is less computationally costly than using a sensor block and it allows you to:

- View post-simulation results easily using the Simscape Results Explorer.
- Output data easily to the MATLAB Workspace for post-processing analysis.

However, if you use only data logging to measure a variable, you cannot output a feedback signal for that variable to a control system during simulation as you can when you use only a sensor to measure the variable. Also, because data logging is not supported for code generation, you cannot use Simscape data logging when you perform real-time simulation on target hardware.

Thermal-Modeling Parameters

Thermal-modeling parameters are device-specific characteristics that determine how much heat a block generates during simulation. When you select a thermal variant for a Diode or Commutation Diode block, no additional parameters are enabled because the default variant includes all parameters necessary to model conduction loss. When you select a thermal variant for a three-terminal semiconductor block, additional thermal-modeling parameters are enabled because the default variant does not include parameters necessary to model switching losses.

Three-terminal semiconductors allow you to parameterize thermal losses based on **Voltage and current** or on **Voltage, current, and temperature**. If you parameterize thermal characteristics based only on voltage and current, use scalar values to specify these parameters:

- Output current
- Switch-on loss
- Switch-off loss
- On-state voltage

If you parameterize thermal losses based on **Voltage, current, and temperature**, use vectors to specify the temperature, output current, switching losses, and on-state voltage.

Limitations

Even though simulating thermal losses generates information about the thermal state of a block, thermal dynamics do not affect the electrical behavior of Simscape Power Systems blocks during simulation.

Model Thermal Losses for a Rectifier

Model Heat Transfer for a Single Rectifier Diode

To model and measure heat transfer as a function of the thermal characteristics of a semiconductor, connect a Foster model-based thermal network and a temperature sensor to the thermal port on **Diode1**.

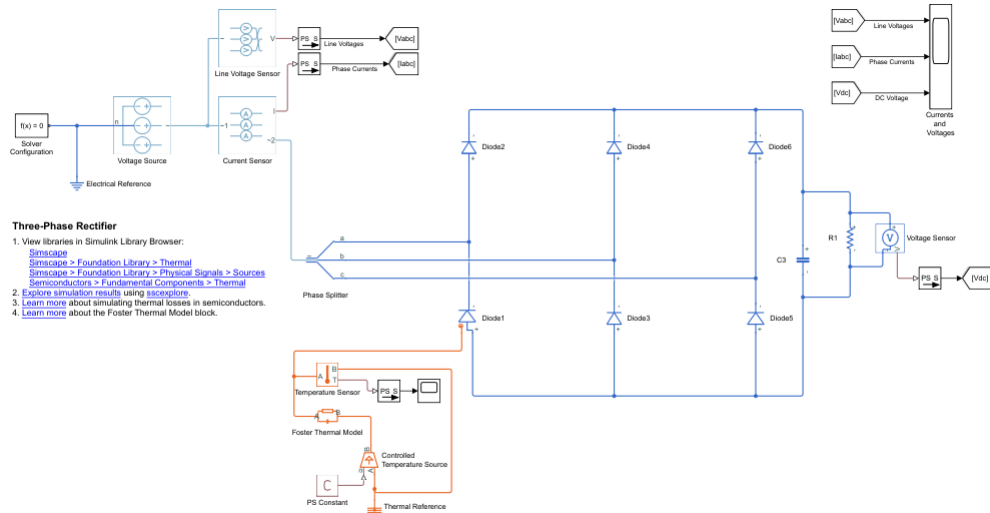
- 1 Open the model, at the MATLAB command prompt, enter

```
pe_rectifier_diodes
```

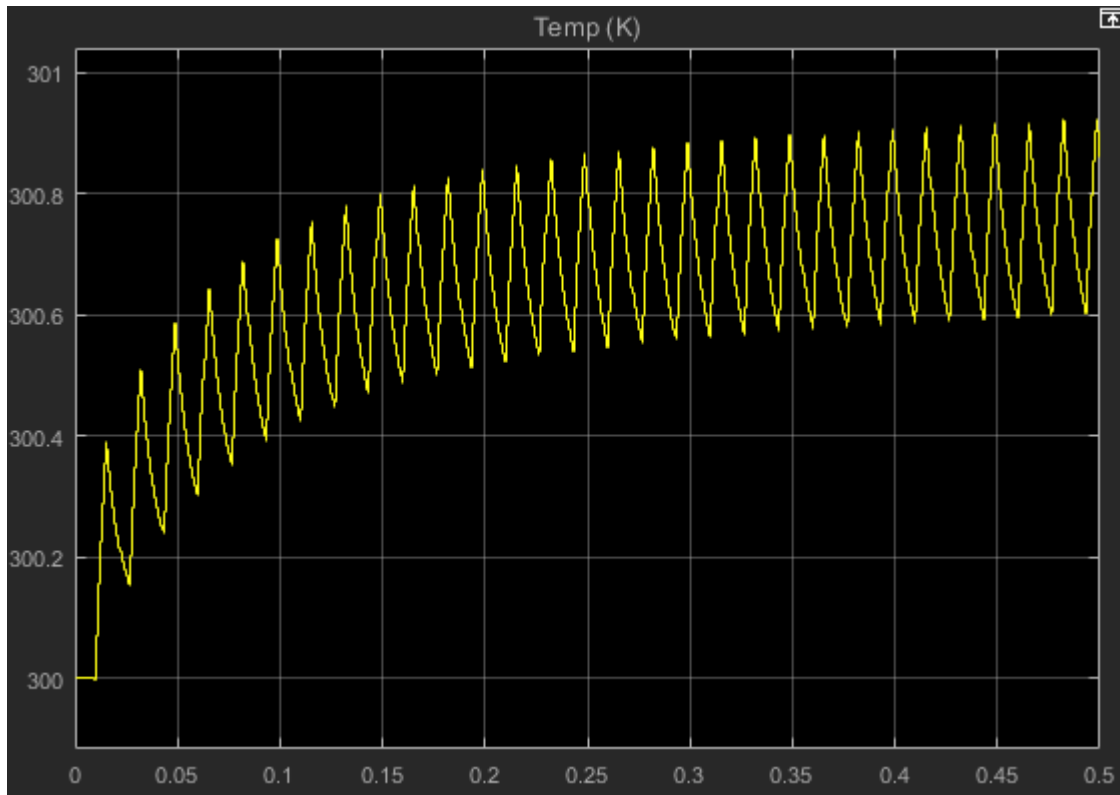
The model contains a three-phase rectifier that includes six Diode blocks.

- 2 Select a thermal variant for the **Diode1** block, right-click the block and, from the context menu, select **Simscape > Block choices**. Select **Show thermal port**.
- 3 Add a Simscape Power Systems Simscape Components block that represents heat flow between the diode and the environment. In the model window, the text on the right, contains links that open the Simulink Library browser. Click **Semiconductors > Fundamental Components > Thermal** and add a Foster Thermal Model block to the model.
- 4 Modify these Foster Thermal Model block parameters:
 - a **Thermal resistance data** — specify [0.00311 0.008493 0.00252 0.00288] K/W.
 - b **Thermal time constant data** — specify [0.0068 0.0642 0.3209 2.0212] s.
- 5 Represent the ambient temperature as constant using an ideal temperature source.
 - a From the Simulink Library browser, open the **Simscape > Foundation Library > Thermal > Thermal Sources** library and add an Ideal Temperature Source block.
 - b From the **Simscape > Foundation Library > Thermal > Thermal Elements** library, add a Thermal Reference block.
 - c From the **Simscape > Foundation Library > Physical Signals > Sources** library, add a PS Constant block. For the **Constant** parameter, specify a value of 300.
- 6 Measure and display the temperature of **Diode1**:
 - a From the Simulink Library browser, open the **Simscape > Foundation Library > Thermal > Thermal Sensors** library, add an Ideal Temperature Sensor block.

- b Make a copy of one of the PS-Simulink Converter blocks in the model window. For the **Output signal unit** parameter, select K.
 - c From the Simulink Library browser, open the **Simulink** > **Sinks** library and add a Scope block.
- 7 Arrange and connect the blocks as shown in the figure.



- 8 Label the signal from the PS-Simulink Converter block to the Scope block, double-click the line between the blocks and at the prompt, enter Temp (K).
- 9 Simulate the model.
- 10 To see the temperature data, open the Scope block.



The temperature of **Diode1** fluctuates over a temperature range of 0.3 K as it increases from the initial value of 300 K to a settling point of 300.6–300.9 K toward the end of the simulation.

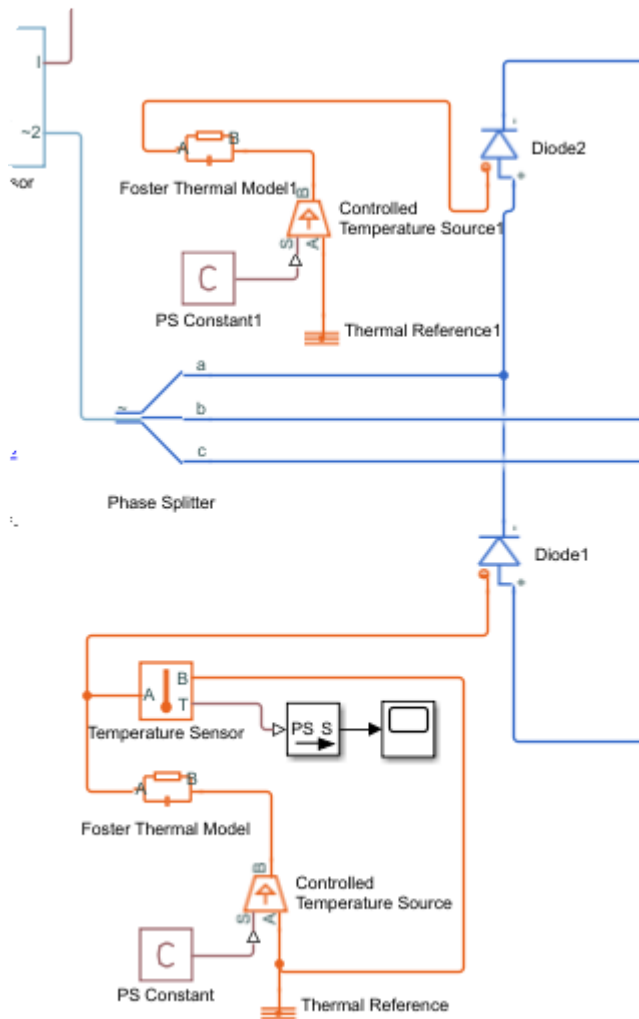
Model Heat Transfer for All Rectifier Diodes

To see the total heat generated by all the semiconductors in the rectifier, use data logging and the Simscape Results Explorer.

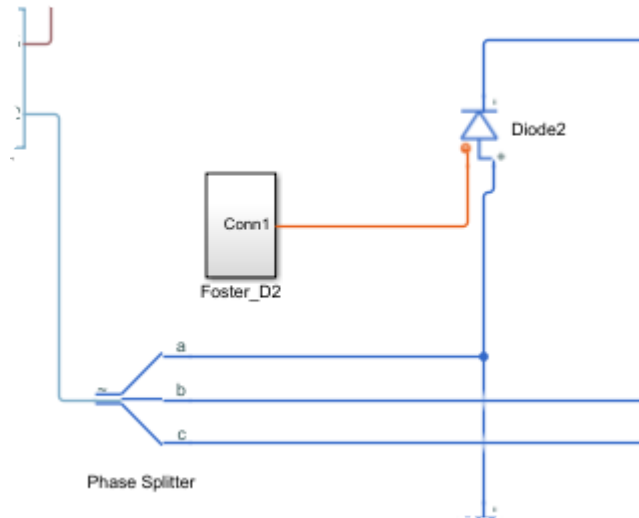
- 1 To enable the thermal ports on all the rectifier diodes, select thermal variants for the **Diode2**, **Diode3**, **Diode4**, **Diode5**, and **Diode6** blocks.
- 2 To measure heat transfer for each diode, create a Foster thermal model subsystem:
 - a Make a copy of this group of blocks:


- Foster Thermal Model
- Ideal Temperature Source
- PS Constant
- Thermal Reference

b Arrange and connect the copied blocks as shown in the figure.

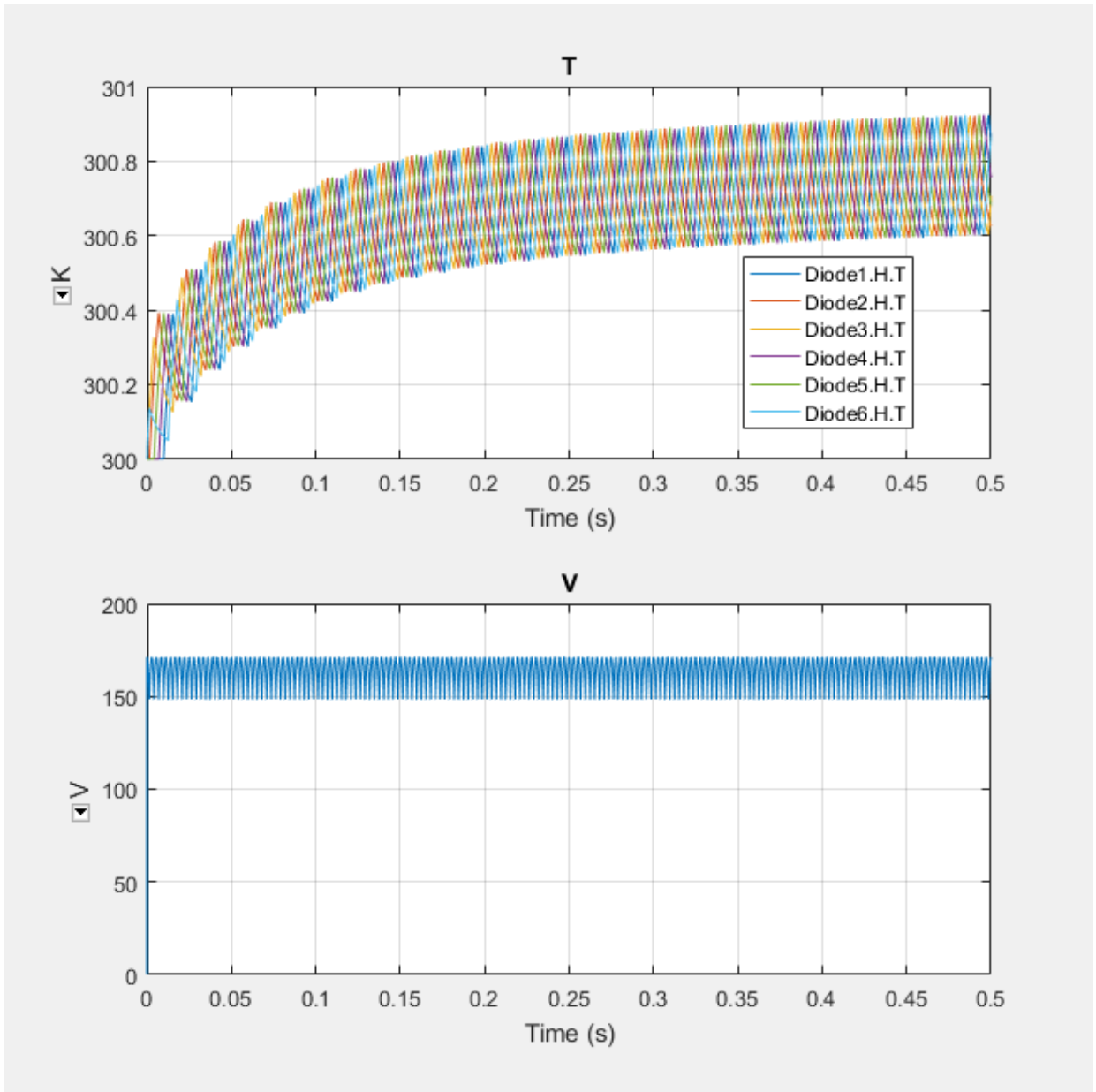


- c Create a subsystem from the copied blocks and rename the subsystem as **Foster_D2**. For information see, “Create a Subsystem” (Simulink).
- d Open the **Foster_D2** subsystem. For the **Conn1** block, for the **Port location on the parent subsystem** parameter, select Right.



- e Make four copies of the **Foster_D2** subsystem. Attach one subsystem to each of the remaining Diode blocks and rename the subsystems as **Foster_D3** through **Foster_D6** to match the **Diode3** through **Diode6** block names.
- 3 Simulate the model.
 - 4 View the results using the Simscape Results Explorer:
 - a In the model window, in the text under **Three-Phase Rectifier**, click **Explore simulation results**.
 - b To display the temperature data for **Diode1**, in the Simscape Results Explorer window, expand the **Diode1 > H** node and click **T**.
 - c To display the DC voltage in a separate plot, expand the **Voltage_Sensor** node and **CTRL**+click **V**.
 - d To display the temperature data for all the diodes, expand the **Diode2 > H** node and **CTRL**+click **T**. Repeat the process for **Diode3** through **Diode6**.
 - e To overlay the temperature data in single plot, in the Simscape Results Explorer window, above the tree-node window, click the options  button. In the Options

dialog box, for **Plot signals**, select **Overlay**. To accept the change, click **OK**. Click and drag the legend down to see the temperature data clearly.



The temperature profile for each diode lags, in succession, behind the temperature profile of **Diode1**. For each diode, the temperature also rises and settles along the same values as the temperature profile for **Diode1**. The data indicate that, because of the lagging behavior of the individual diode temperatures, the temperature of the rectifier rises and settles along the same temperature profile as the diodes, but with less fluctuation.

References

- [1] Schütze, T. *AN2008-03: Thermal equivalent circuit models*. Application Note. V1.0. Germany: Infineon Technologies AG, 2008.

See Also

Cauer Thermal Model Element | Commutation Diode | Diode | Foster Thermal Model | GTO | IGBT | MOSFET | Thermal Resistor | Thyristor

Related Examples

- “Quantifying IGBT Thermal Losses”

Perform a Power-Loss Analysis

In this section...

“Prerequisite” on page 7-16

“Calculate Average Power Losses for the Simulation” on page 7-16

“Analyze Power Dissipation Differences Using Instantaneous Power Dissipation” on page 7-18

“Mitigate Transient Effects in Simulation Data” on page 7-22

This example shows how to analyze power loss and how to mitigate transient power dissipation behavior. Analyzing power loss, with and without transients, is useful for determining if components are operating within safety and efficiency guidelines.

Prerequisite

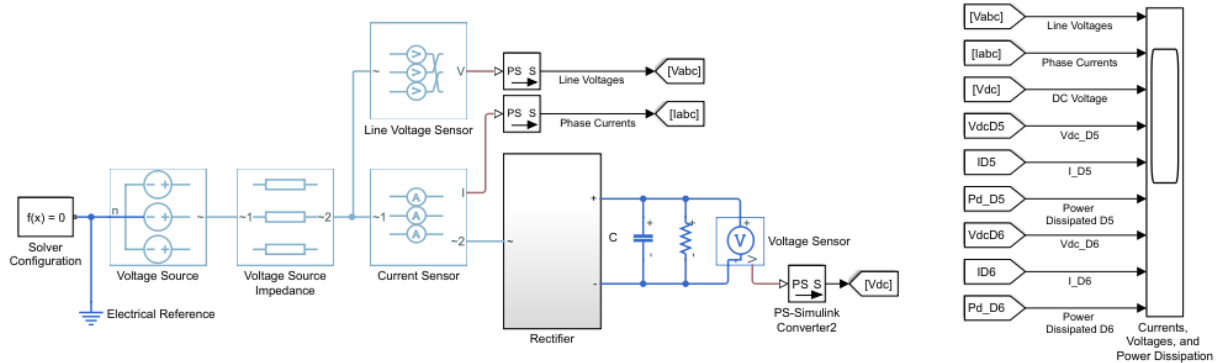
This example requires a simulation log variable in your MATLAB workspace. The model in this example is configured to log Simscape data for the whole model for the entire simulation time.

To learn how to determine if a model is configured to log simulation data, see “Examine the Simulation Data-Logging Configuration of a Model” on page 7-3.

Calculate Average Power Losses for the Simulation

- 1 Open the model. At the MATLAB command prompt, enter

```
model = 'pe_rectifier_power_dissipated';  
open(model)
```



Power-Loss Analysis of a Three-Phase Rectifier

- 2 Simulate the model.

```
sim(model)
```

The simulation log variable, which is named `simlog_pe_rectifier_power_dissipated`, appears in the workspace.

- 3 Calculate the average losses for the entire simulation for each of the diodes in the model.

```
rectifierLosses = pe_getPowerLossSummary(simlog_pe_rectifier_power_dissipated.Rectifier)
```

```
rectifierLosses =
```

6x2 table

LoggingNode	Power
'pe_rectifier_power_dissipated.Rectifier.D6'	52.222
'pe_rectifier_power_dissipated.Rectifier.D3'	52.222
'pe_rectifier_power_dissipated.Rectifier.D4'	52.194
'pe_rectifier_power_dissipated.Rectifier.D5'	52.194
'pe_rectifier_power_dissipated.Rectifier.D1'	52.194
'pe_rectifier_power_dissipated.Rectifier.D2'	52.194

On average, diodes D3 and D6 dissipate more power than the other diodes in the rectifier.

Analyze Power Dissipation Differences Using Instantaneous Power Dissipation

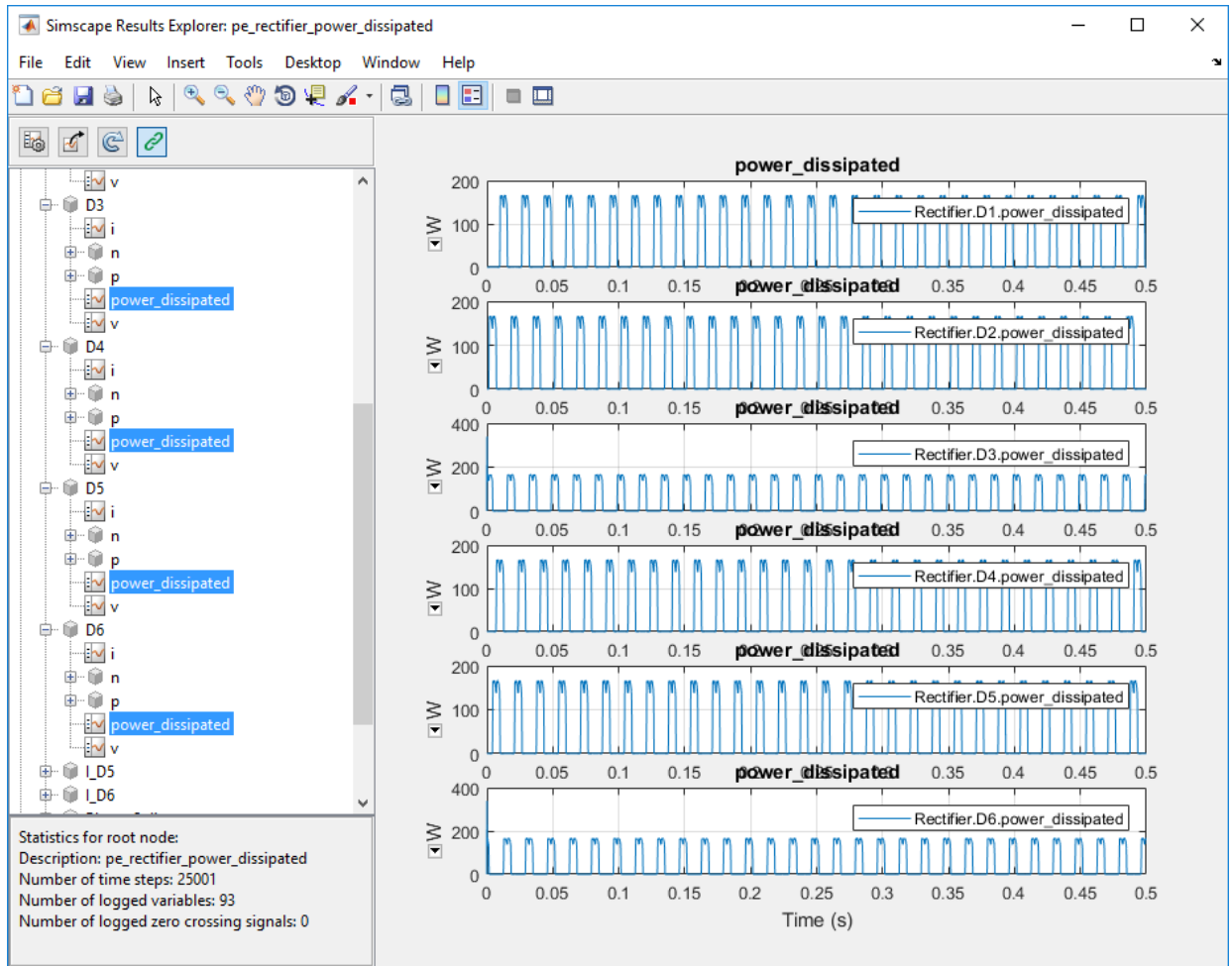
The Diode blocks each have a *power_dissipated* variable, which measures instantaneous power dissipation. To investigate the differences in the average power dissipated by the diodes, view the simulation data using the Simscape Results Explorer.

- 1 Open the simulation data using the Results Explorer.


```
sscexplore(simlog_pe_rectifier_power_dissipated)
```

- 2 View the instantaneous power dissipated by the diodes.

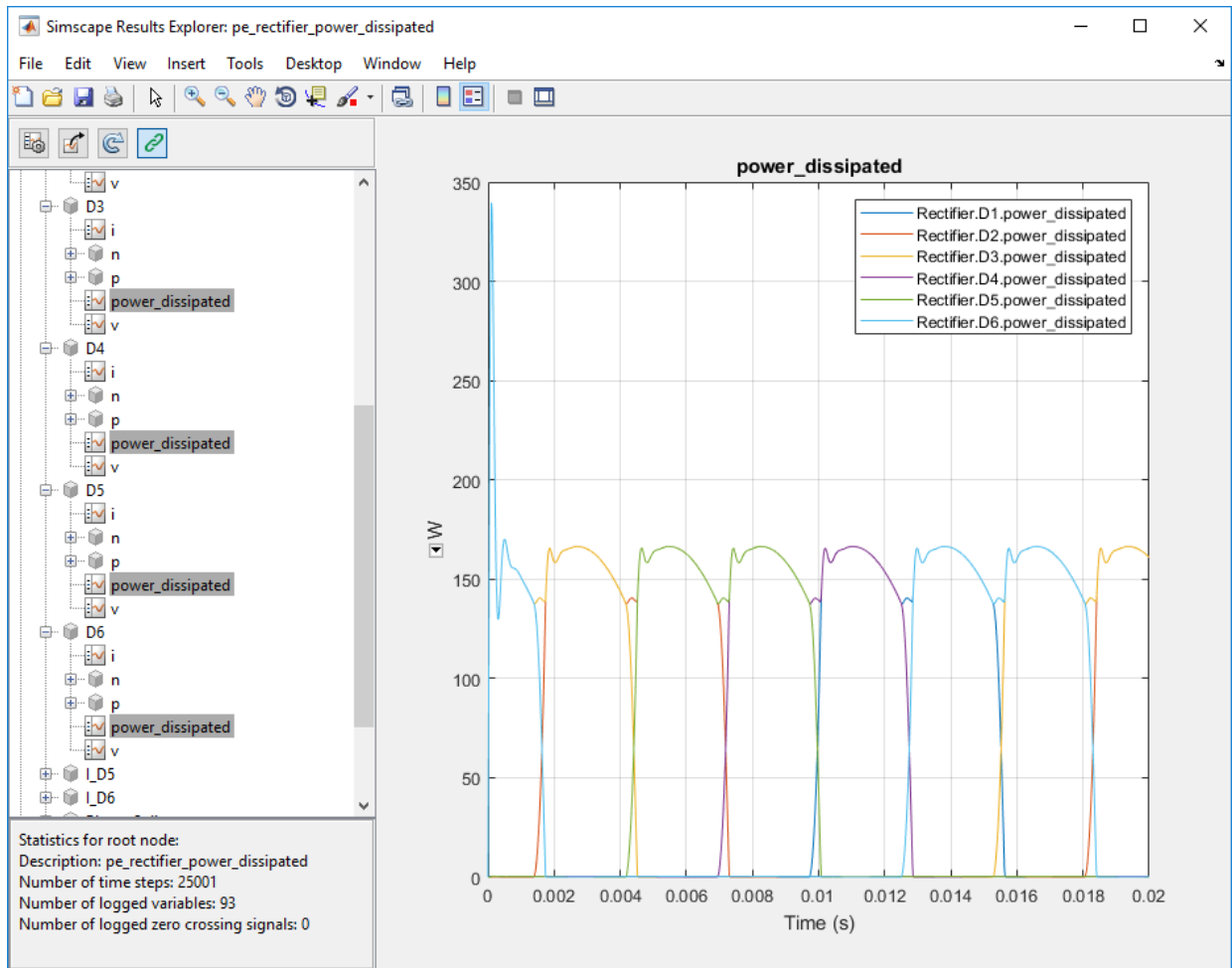
- a Expand the **Rectifier** node
- b Expand the **D1** through **D6** nodes
- c Click the *power_dissipated* nodes for diode **D1**, and then **Ctrl+click** the *power_dissipated* nodes for the other five diodes.



At the beginning of the simulation, there is a difference in the power dissipation for each diode.

- 3 Take a closer look at the differences. Overlay the plots and zoom to the beginning of the simulation.
 - a In the Results Explorer window, click the plot options  button.
 - b Enable the **Limit time axis** option.

- c For **Stop time**, specify 0.02.
- d Set **Plot signals** to Overlay.
- e Click **OK**.



The variation in power dissipation is due to transient behavior at the beginning of the simulation. The model reaches steady state at simulation time, $t \approx 0.001$ seconds.

- 4 Determine the average power dissipation for only the diodes during the interval that contains transient behavior.

```
rectifierLosses = pe_getPowerLossSummary(simlog_pe_rectifier_power_dissipated.Rectifier,0,1e-3)
```

```
rectifierLosses =
```

```
6x2 table
```

LoggingNode	Power
'Rectifier.D3'	174.88
'Rectifier.D6'	174.88
'Rectifier.D4'	0.27539
'Rectifier.D5'	0.27539
'Rectifier.D1'	0.12482
'Rectifier.D2'	0.032017

The average power dissipated by diodes D3 and D6 exceeds the average for the other diodes.

- 5** Output a table of the maximum power dissipation for each diode, for the entire simulation time.

```
pd_D1_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D1.power_dissipated.series.values);
pd_D2_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D2.power_dissipated.series.values);
pd_D3_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D3.power_dissipated.series.values);
pd_D4_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D4.power_dissipated.series.values);
pd_D5_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D5.power_dissipated.series.values);
pd_D6_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D6.power_dissipated.series.values);
```

```
diodes = {'D1'; 'D2'; 'D3'; 'D4'; 'D5'; 'D6'};
PowerMax = [pd_D1_max;pd_D2_max;pd_D3_max;pd_D4_max;pd_D5_max;pd_D6_max];
```

```
T = table(PowerMax, 'RowNames', diodes)
```

```
T =
```

```
6x1 table
```

	PowerMax
D1	166.45
D2	166.45
D3	339.54
D4	166.45
D5	166.45
D6	339.54

The maximum instantaneous power dissipation for diodes D3 and D6 is almost double the maximum instantaneous power dissipation for the other diodes.


Mitigate Transient Effects in Simulation Data

To mitigate the transient power dissipation at the beginning of the simulation, use the final simulation state to initialize a new simulation at steady-state conditions.

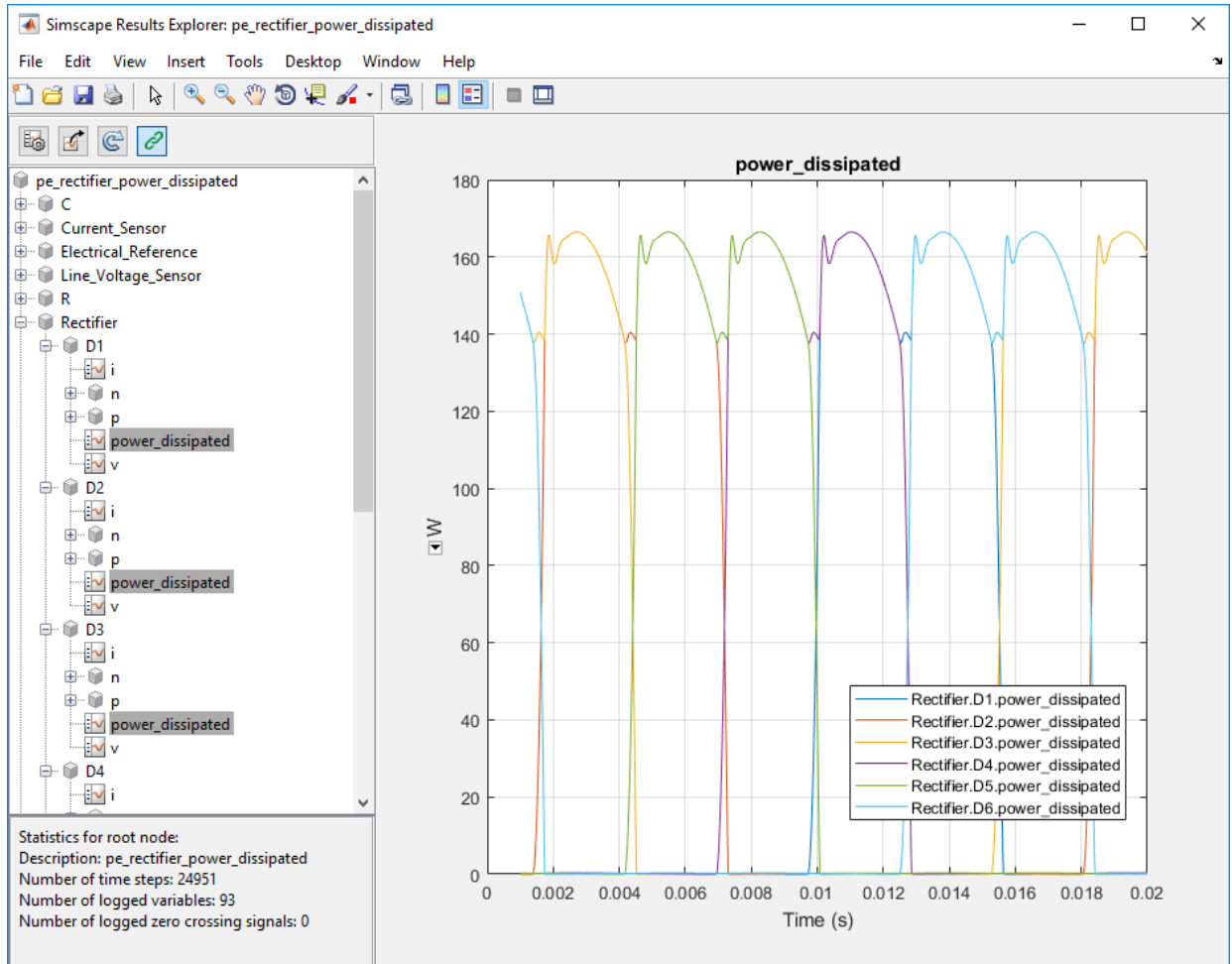
- 1 Configure the model to save the final state.
 - a Open the model configuration parameters.
 - b In the **Solver** pane, change the **Stop time** from 0.5 to 1e-3.
 - c In the **Data Import/Export** pane, select these options:
 - **Final States**
 - **Save complete SimState in final state**
 - d Click **Apply**.

- 2 Run the simulation.

The final state is saved as the variable `xFinal` in the MATLAB workspace.

- 3 Configure the model to initialize using `xFinal`, in the model configuration parameters.
 - a In the Data Import/Export pane:
 - Select the **Initial state** option.
 - Change the **Initial state** parameter value from `xInitial` to `xFinal`.
 - Clear the **Final states** option.
 - b In the **Solver** pane, change the **Stop time** to 0.5.
 - c Click **OK**.
- 4 Run the simulation.
- 5 View the data from the new simulation.
 - a Click the **Reload logged data**  button in the Simscape Results Explorer.
 - b Click **OK** to confirm that `simlog_pe_rectifier_power_dissipated` is the variable name that contains the logged data.

- c To see the data more clearly, click and drag the legend away from the peak amplitudes.



The plot shows that the simulation no longer contains the transient.

- 6 Output a table of the maximum power dissipation for each diode, for the modified simulation.

```
pd_D1_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D1.power_dissipated.series.values);
pd_D2_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D2.power_dissipated.series.values);
pd_D3_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D3.power_dissipated.series.values);
```

```
pd_D4_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D4.power_dissipated.series.values);
pd_D5_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D5.power_dissipated.series.values);
pd_D6_max = max(simlog_pe_rectifier_power_dissipated.Rectifier.D6.power_dissipated.series.values);

diodes = {'D1'; 'D2'; 'D3'; 'D4'; 'D5'; 'D6'};
PowerMax = [pd_D1_max; pd_D2_max; pd_D3_max; pd_D4_max; pd_D5_max; pd_D6_max];

T = table(PowerMax, 'RowNames', diodes)

T =
```

6×1 table

	PowerMax
D1	166.45
D2	166.45
D3	166.45
D4	166.45
D5	166.45
D6	166.45

The maximum instantaneous power dissipation for diodes D3 and D6 is the same as the maximum instantaneous power dissipation for the other diodes.

See Also

Functions

[pe_getEfficiency](#) | [pe_getPowerLossSummary](#) | [pe_getPowerLossTimeSeries](#)

Related Examples

- “Power-Loss Analysis of a Three-Phase Rectifier”
- “Examine the Simulation Data-Logging Configuration of a Model” on page 7-3
- “Data Logging” (Simscape)
- “About the Simscape Results Explorer” (Simscape)

Choose a Simscape Power Systems Function for an Offline Harmonic Analysis

In this section...

“Harmonic Distortion” on page 7-25

“Harmonic Analysis Functions” on page 7-25

“Evaluate Relative Overall Harmonic Distortion” on page 7-26

“Compare Harmonic Distortion to Standard Limits” on page 7-27

“Minimize Harmonic Distortion with Passive Filters” on page 7-27

“Verify the Results of an Online Harmonic Analysis” on page 7-28

Harmonic Distortion

Nonlinear loads create power distortion in the form of harmonics, that is, voltages and currents that are multiples of the fundamental frequency. Harmonic waveforms can result in energy losses through heat dissipation and in reduced power quality. They can also cause equipment to malfunction or to become damaged. Standards development organizations such as the Institute of Electrical and Electronics Engineers (IEEE) and the International Electrotechnical Commission (IEC) define the recommended limits for harmonic content in electric power systems.

Harmonic Analysis Functions

You can use the simulation and analysis functions in Simscape Power Systems Simscape Components to perform an offline, that is post-simulation, analysis to examine harmonic distortion in your model. The `pe_plotHarmonics` function generates a bar chart. The `pe_getHarmonics` and `pe_calculateThdPercent` functions provide harmonic data in numerical form.

To decide which functions and workflows to use for your harmonic analysis, consider your goals. The table cross-references the harmonic functions with common harmonic analysis according to the data the function outputs and the task requires.

Goal	pe_plotHarmonics	pe_getHarmonics	pe_calculateThdPercent
Evaluate the relative overall harmonic distortion	<ul style="list-style-type: none"> • Bar chart of the percentage of fundamental magnitude • Fundamental peak value • Total harmonic distortion (THD) percentage 		
Compare the harmonic distortion to standard limits		<ul style="list-style-type: none"> • Fundamental frequency • Harmonic orders • Harmonic magnitudes 	Total harmonic distortion (THD) percentage
Determine the parameters for filtering harmonic distortion		<ul style="list-style-type: none"> • Fundamental frequency • Harmonic orders • Harmonic magnitudes 	

Evaluate Relative Overall Harmonic Distortion

Use this workflow for a high-level understanding of the waveform distortion in your power system.

- 1** Enable Simscape data logging.
- 2** Save the logged voltage or current data to a variable.
- 3** Use the `pe_plotHarmonics` function to generate a bar chart of harmonic percentages with the peak fundamental magnitude and the total harmonic distortion (THD) percentage displayed in the plot title.

Compare Harmonic Distortion to Standard Limits

Use this workflow to obtain values for evaluating the IEEE or IEC suitability of your power system.

- 1 Enable Simscape data logging.
- 2 Save the logged voltage or current data to a variable.
- 3 Use the `pe_getHarmonics` function to obtain the harmonic orders, the magnitude for each order, and the fundamental frequency.
- 4 Save the fundamental peak to a new variable.
- 5 Calculate the RMS voltage or current for each order.
- 6 Calculate the harmonic distortion percentage for individual harmonics.
- 7 Use the `pe_calculateThdPercent` function to obtain the total harmonic distortion (THD).
- 8 Compare the percentage data for each order and the THD percentage to the standard limits.

Minimize Harmonic Distortion with Passive Filters

Use this workflow to determine the parameters for filtering the distorted waveforms with passive filters. Use individual, series-tuned filters for specific harmonic orders. Use a single high-pass filter to filter higher orders.

- 1 Enable Simscape data logging.
- 2 Save the logged voltage or current data in a variable.
- 3 Use the `pe_getHarmonics` function to obtain the harmonic orders, the magnitude for each order, and the fundamental frequency.
- 4 Identify the harmonic orders that you want to filter.
- 5 For each filter:
 - a Specify the filter size, in terms of reactive power compensation, and specify the filter quality.
 - b Calculate the capacitor reactance at the tuned harmonic order.
 - c Calculate the filter capacitance.
 - d Calculate the inductor reactance at the tuned harmonic order.

- e Calculate the filter inductance.
- f Calculate the filter resistance.

Verify the Results of an Online Harmonic Analysis

You can examine harmonic distortion in your model online, that is during simulation, using the Simscape Spectrum Analyzer block. To verify the results from the Spectrum Analyzer block:

- 1 To determine the THD in your model, perform an online analysis. For information, see “Perform an Online Harmonic Analysis Using the Simscape Spectrum Analyzer Block” on page 7-29.
- 2 Use the `pe_getHarmonics` and `pe_calculateThdPercent` functions to determine the THD in your model.
- 3 Compare the THD values for the online and offline analyses. If the results differ, reconfigure the Spectrum Analyzer block.

See Also

Blocks

Spectrum Analyzer

Functions

`pe_calculateThdPercent` | `pe_getHarmonics` | `pe_plotHarmonics`

Related Examples

- “Harmonic Analysis of a Three-Phase Rectifier”
- “Perform an Online Harmonic Analysis Using the Simscape Spectrum Analyzer Block” on page 7-29
- “Data Logging” (Simscape)

Perform an Online Harmonic Analysis Using the Simscape Spectrum Analyzer Block

In this section...

“Harmonic Distortion” on page 7-29

“Prerequisite” on page 7-29

“Perform an Offline Harmonic Analysis” on page 7-30

“Perform an Online Harmonic Analysis” on page 7-33

Harmonic Distortion

Nonlinear loads create power distortion in the form of harmonics, that is, voltages and currents that are multiples of the fundamental frequency. Harmonic waveforms can result in energy losses through heat dissipation and in reduced power quality. They can also cause equipment to malfunction or to become damaged. Standards development organizations such as the Institute of Electrical and Electronics Engineers (IEEE) and the International Electrotechnical Commission (IEC) define the recommended limits for harmonic content in electric power systems.

This example shows how to examine harmonic distortion in your model using offline, that is after simulation, and online, that is during simulation, analyses. The offline analysis uses the Simscape Power Systems harmonic analysis functions and helps you to determine configuration settings for, and verify the results of, the online analysis. The online analysis uses the Simscape Spectrum Analyzer block.

Prerequisite

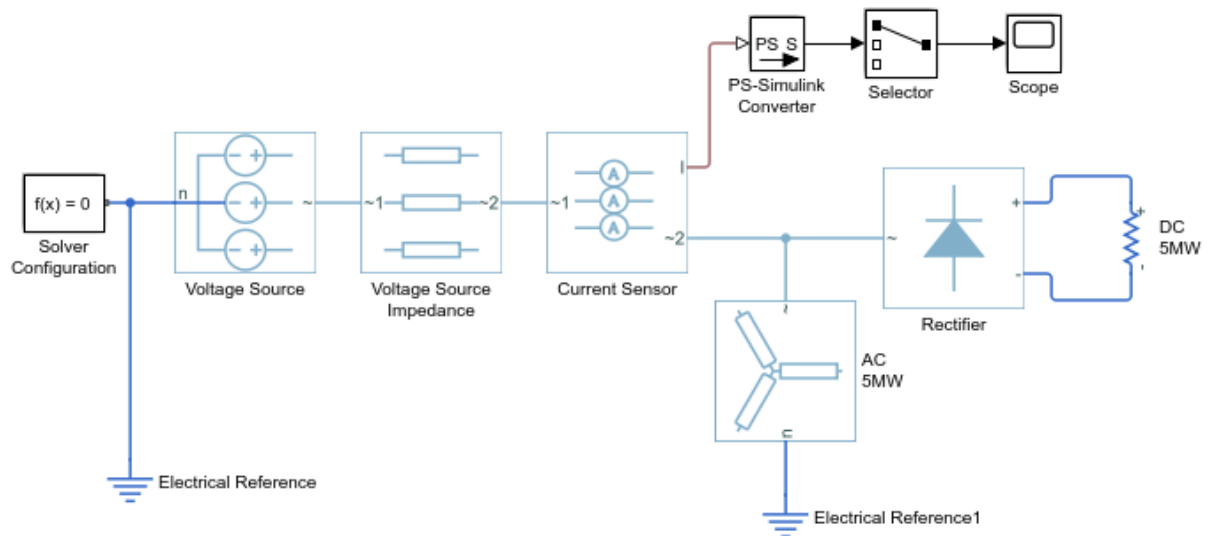
This example requires a simulation log variable in your MATLAB workspace. The model in this example is configured to log Simscape data for the whole model for the entire simulation time.

To learn how to determine if a model is configured to log simulation data, see “Examine the Simulation Data-Logging Configuration of a Model” on page 7-3.

Perform an Offline Harmonic Analysis

- 1 Open the model. At the MATLAB command prompt, enter:

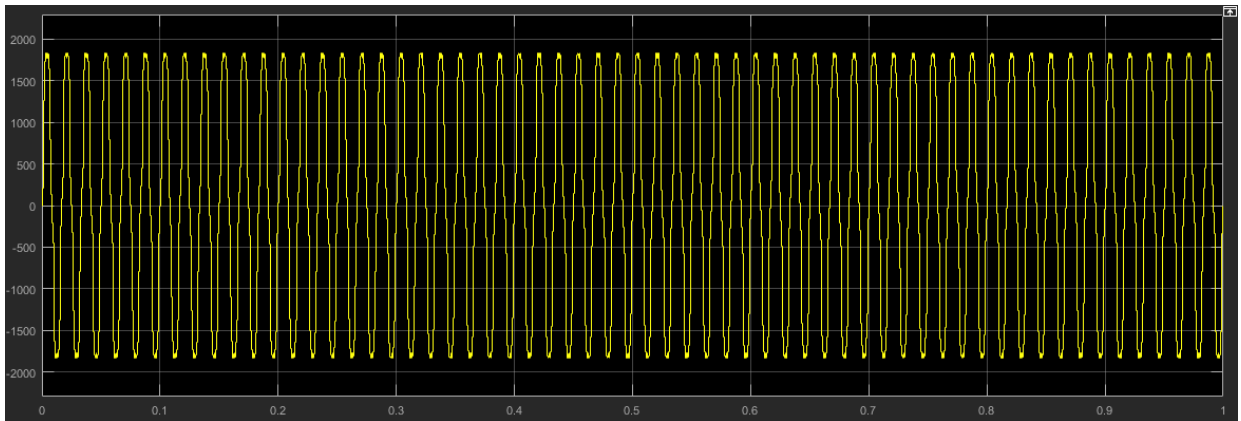
```
model = 'pe_composite_rectifier';
open_system(model)
```



The example model contains a three-phase rectifier. The model also contains a Selector block that outputs only the a -phase from three-phase current signal that it receives from the PS-Simulink Converter block.

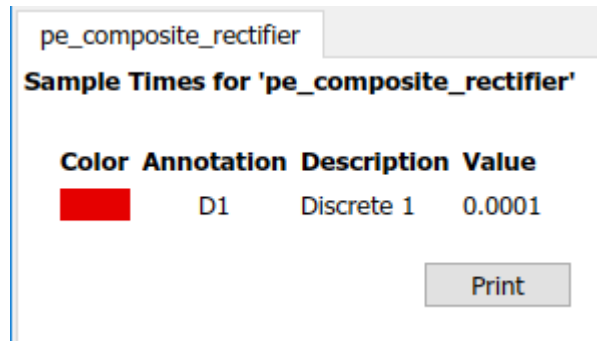
- 2 Simulate the model.


```
sim(model)
```
- 3 View the time-domain results. Open the Scope block.



The time domain analysis shows that the rectifier is converting the voltage, but it does not include any information about the frequencies in the signal.

- 4 Determine configuration settings and calculate the expected results for an online harmonic analysis. Perform an offline harmonic analysis.
 - a The Simscape Power Systems harmonic analysis functions require that you use a fixed-step solver. Determine the solver type and sample time for the model. To turn on sample-time highlighting, in the Simulink editor menu bar, select **Display > Sample Time > All**.



The model is running at a discrete rate, therefore it is using a fixed-step solver, with a sample time of $1e-4$ s.

- b** Use the `pe_getHarmonics` function to calculate the harmonic order, the harmonic magnitude, and the fundamental frequency based on the voltage source currents.

```
[harmonicOrder,harmonicMagnitude,fundamentalFrequency] = ...
pe_getHarmonics(simlog_composite_rectifier.Voltage_Source.I);
```

- c** Performing an online harmonic analysis using the Spectrum Analyzer block requires that you specify a value for maximum harmonic order and the resolution bandwidth (RBW). The RBW depends on the fundamental frequency.

Extract and display the maximum harmonic order and the fundamental frequency:

```
disp(['Maximum Harmonic Order = ', num2str(max(harmonicOrder))])
disp(['Fundamental Frequency = ', num2str(fundamentalFrequency)])
```

```
Maximum Harmonic Order = 30
Fundamental Frequency = 60
```

- d** Determine the peak value of the fundamental frequency. This value is useful for filtering out negligible harmonics and for verifying the results of the offline analyses.

```
fundamentalPeak = harmonicMagnitude(harmonicOrder==1);
disp(['Peak value of fundamental = ', num2str(fundamentalPeak), ' A']);
```

```
Peak value of fundamental = 1945.806 A
```

- e** Filter out small harmonics by identifying and keeping harmonics that are greater than one thousandth of the fundamental peak frequency.

```
threshold = fundamentalPeak ./ 1e3;
aboveThresold = harmonicMagnitude > threshold;
harmonicOrder = harmonicOrder(aboveThresold)';
harmonicMagnitude = harmonicMagnitude(aboveThresold)';
```

- f** Display the harmonic data in a MATLAB table.

```
harmonicRms = harmonicMagnitude./sqrt(2);
harmonicPct = 100.*harmonicMagnitude./harmonicMagnitude(harmonicOrder == 1);
harmonicTable = table(harmonicOrder,...
    harmonicMagnitude,...
    harmonicRms,...
    harmonicPct,...
    'VariableNames',{'Order','Magnitude','RMS','Percentage'});
display(harmonicTable);
```

```
harmonicTable =
```

10×4 table

Order	Magnitude	RMS	Percentage
1	1945.8	1375.9	100
5	218.86	154.75	11.248
7	105.83	74.835	5.439
11	85.135	60.2	4.3753
13	57.599	40.729	2.9602
17	50.417	35.65	2.5911
19	37.612	26.596	1.933
23	33.859	23.942	1.7401
25	26.507	18.743	1.3622
29	23.979	16.955	1.2323

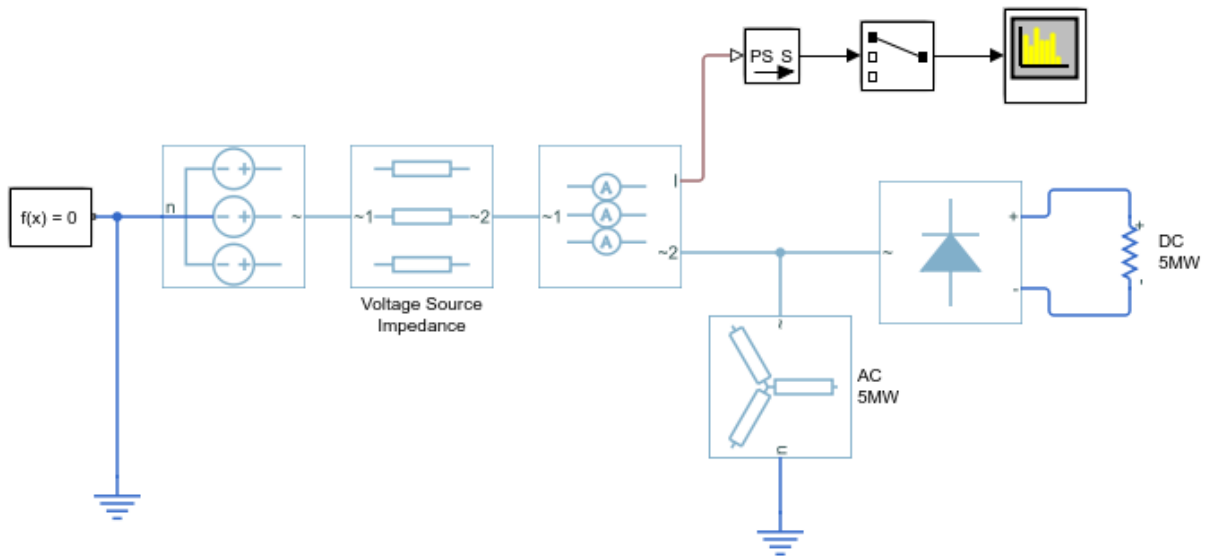
- g** Calculate the total harmonic distortion (THD) percentage using the `pe_calculate_ThdPercent` function.


```
thdPercent = pe_calculateThdPercent(harmonicOrder,harmonicMagnitude);
disp(['Total Harmonic Distortion Percentage = ' num2str(thdPercent), '%']);
```

Total Harmonic Distortion percentage = 14.1721 %

Perform an Online Harmonic Analysis

- 1** In the Simulink editor that contains the `pe_composite_rectifier` model, replace the Scope block with a Spectrum Analyzer block from the Simscape Utilities Library:
 - a** Delete the Scope block.
 - b** Left-click within the block diagram.
 - c** After the search icon appears, type `spec`, and then from the list, select the Spectrum Analyzer from the Utilities library.
 - d** Connect the Spectrum Analyzer block to the output signal from the Selector block.



- 2 Configure the Spectrum Analyzer block using the Spectrum Settings panel.
 - a Open the Spectrum Analyzer.
 - b Open the Spectrum Settings panel. On the Spectrum Analyzer toolbar, click the **Spectrum Settings**  button.
 - c Configure the parameters on the **Main Options** pane.
 - i Configure the block to display the root mean square (RMS) of the frequency. From the **Type** dropdown menu, select RMS.
 - ii Determine the value to specify for the resolution bandwidth (RBW) using this equation:

$$RBW = \frac{NENBW * f}{N},$$

where,



- *NENBW* is the normalized effective noise bandwidth, a factor of the windowing method used. The Hanning (Hann) window has an *NENBW* value of approximately 1.5.

- f is the fundamental frequency.
- N is the number of periods.
- RBW is the resolution bandwidth in Hz.

For a fundamental frequency of 60 Hz over 10 periods, using a Hann window,

$$RBW = \frac{1.5 * 60Hz}{10} = 9Hz$$

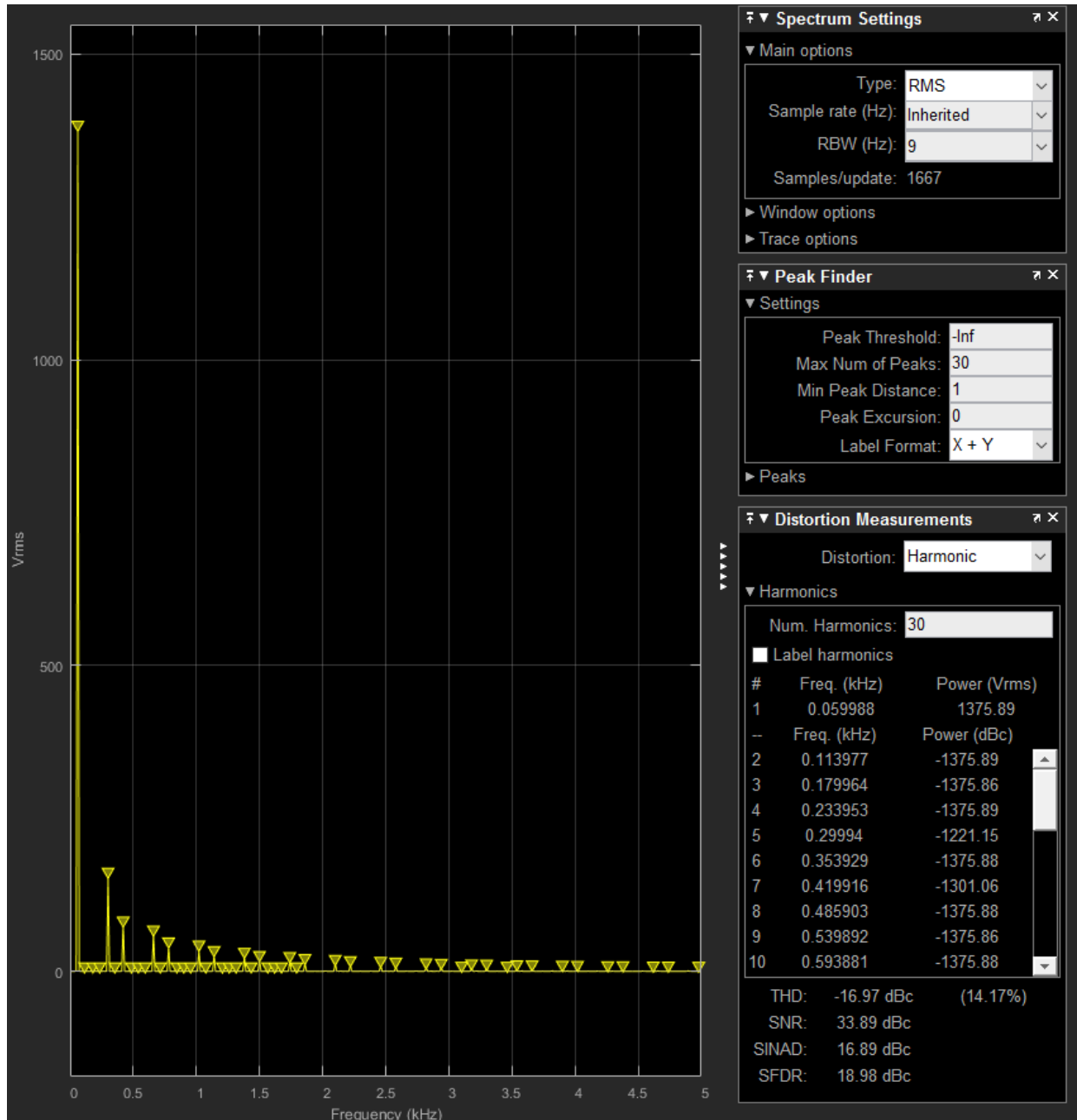
For **RBW (Hz)**, specify 9.

- Expand the **Windows Options** pane and specify an **Overlap (%)** of 90.
- Specify the maximum number of peaks for the analyzer to display. In the menu bar, select **Tools > Measurements > Peak Finder**. Alternatively, in the Spectrum Analyzer toolbar, select the Peak Finder  button. In the **Peakfinder** pane, in the **Settings** section, for **Max Num of Peaks**, enter 30. This value is based on the maximum harmonic order as indicated by the offline analysis.
- Set the number of harmonics to use for measuring harmonic distortion. Specify a number that captures the largest harmonic order that the offline analysis captures. In the menu bar, select **Tools > Measurements > Distortion Measurements**. Alternatively, in the Scope toolbar, click the Distortion Measurements  button. Scroll as required to see the **Distortion Measurements** pane.

In the **Distortion Measurements** pane, for **Num Harmonics**, again enter 30.

- 3 Simulate the model.

```
sim(model)
```



The THD percentage is 14.17% and the fundamental peak power is 1375.89 Vrms at 0.06 kHz (60 Hz). These results agree with the results from the offline harmonic analysis.

See Also

Blocks

PS-Simulink Converter | Selector | Spectrum Analyzer

Functions

pe_calculateThdPercent | pe_getHarmonics | pe_plotHarmonics

Related Examples

- “Harmonic Analysis of a Three-Phase Rectifier”
- “Choose a Simscape Power Systems Function for an Offline Harmonic Analysis” on page 7-25
- “Data Logging” (Simscape)

Optimize Block Settings for Simulations that Use the Partitioning Solver

In this section...

“Update Solver and Zero-Sequence Settings Using the `pe_solverUpdate` Function” on page 7-39

“Limitations of the `pe_updateSolver` Function” on page 7-48

The Partitioning solver is a Simscape fixed-step local solver that improves performance for certain models. However, not all networks can simulate with the Partitioning solver. Some models that use the Partitioning solver can produce errors and fail to initialize due to numerical difficulties. To resolve numerical difficulties preventing initialization with asynchronous, synchronous, and permanent magnet rotor machine blocks, you can exclude zero-sequence terms. Excluding parasitic conductance resolves numerical difficulties with the Floating Neutral and Neutral Connection block, which include such conductance by default.

To determine the best solver choice for your model, use the `pe_updateSolver` helper function, which is useful for iterating with various solvers. The function updates certain parameter values for every instance of these blocks in your model:

- Solver Configuration blocks
- Machine blocks that have a **Zero sequence** parameter
- Connection blocks that have a **Parasitic conductance to ground** parameter

The function syntax is `pe_updateSolver(solver, system)`. Specify both input arguments using character vectors. The table shows how the function updates the values, depending on the solver that you specify.

Input Argument	Solver Configuration Block (<i>Solver type</i>)	Solver Configuration Block (<i>Use local solver and Use fixed-cost runtime consistency iterations</i>)	Asynchronous, Synchronous, and Permanent Magnet Rotor Machine Blocks (<i>Zero sequence</i>)	Floating Neutral Block and Neutral Connection Block (<i>Parasitic conductance to ground</i>)
'Partitioning'	Partitioning	Selected	Exclude	0
'Backward Euler' or 'BackwardEuler'	Backward Euler	Selected	Include	1e-12
'Trapezoidal'	Trapezoidal	Selected	Include	1e-12
'Global' or 'Nonlocal'	No change	Cleared	Include	1e-12

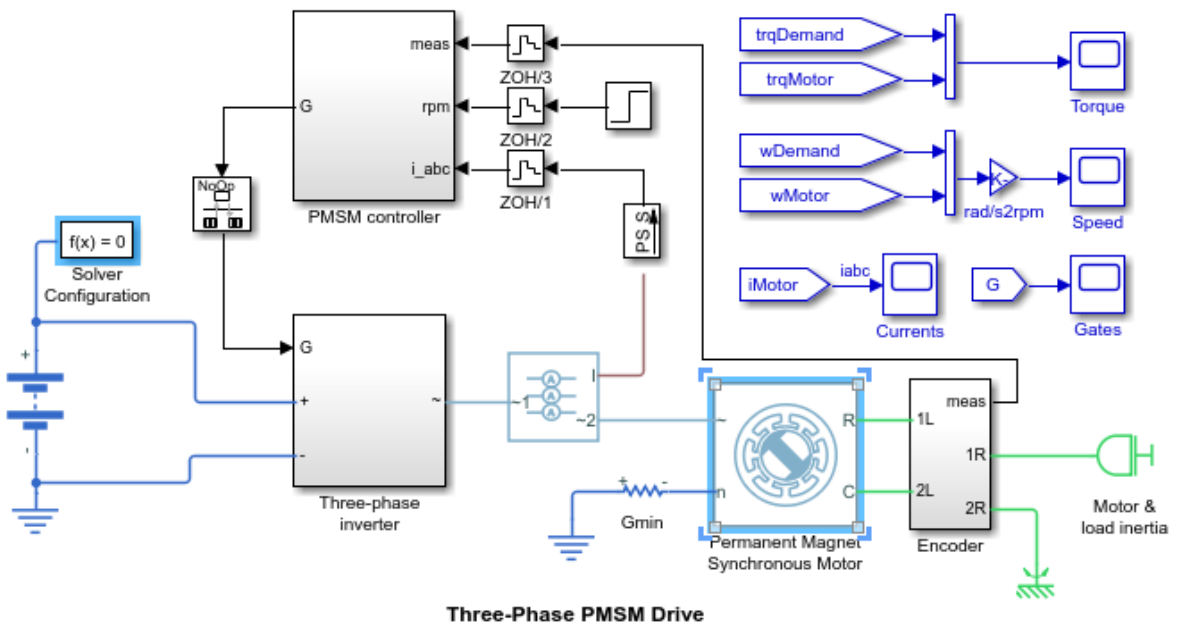
Update Solver and Zero-Sequence Settings Using the `pe_solverUpdate` Function

This example shows how to use the `pe_solverUpdate` function to configure the Solver Configuration and a Permanent Magnet Synchronous Motor blocks in a model for simulation with the Partitioning solver and the Backward Euler solver. It also shows how to compare the simulation duration times and the results.

- 1 Open the model. At the MATLAB command prompt, enter this code.

See Code

```
model = 'pe_pmsm_drive';
open_system(model)
```



This example shows a Permanent Magnet Synchronous Machine (PMSM) and inverter sized for use in a typical hybrid vehicle. Here the inverter is connected directly to the vehicle battery, but often there is also a DC-DC converter stage in between. The model can be used to design the PMSM controller, selecting architecture and gains to achieve desired performance. To check the timing of IGBT turn-on and turn-off, the IGBT devices can be directly replaced by more detailed device models from Simscape(TM) Electronics(TM). For complete vehicle modeling, the Simscape Electronics Servomotor block can be used to abstract the PMSM, inverter and controller with an energy-based model. The Gmin resistor provides a very small conductance to ground that improves the numerical properties of the model when using a variable-step solver.

Two blocks that the `pe_solverUpdate` function can update are the Solver Configuration block and Permanent Magnet Synchronous Motor (PMSM) block.

- 2 Save the parameter settings for the two blocks.

See Code

```
% Define the Solver Configuration block and the path
%   to it as variables
solvConfig = 'Solver Configuration';
solvConfigPath = [model,'/',solvConfig];

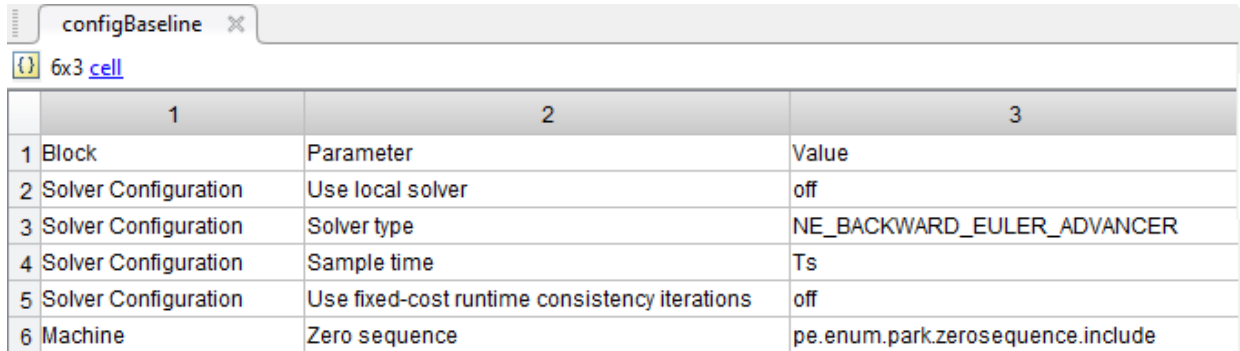
% Define the machine block and the path
%   to it as variables
```

```

machine = 'Permanent Magnet Synchronous Motor';
machinePath = [model, '/', machine];

% Create a cell array that contains configuration data
configBaseline = {'Block', 'Parameter', 'Value';
    'Solver Configuration', 'Use local solver', ...
    get_param(solvConfigPath, 'UseLocalSolver');
    'Solver Configuration', 'Solver type', ...
    get_param(solvConfigPath, 'LocalSolverChoice');
    'Solver Configuration', 'Sample time', ...
    get_param(solvConfigPath, 'LocalSolverSampleTime');
    'Solver Configuration', ...
    'Use fixed-cost runtime consistency iterations', ...
    get_param(solvConfigPath, 'DoFixedCost');
    'Machine', 'Zero sequence', ...
    get_param(machinePath, 'zero_sequence')};
    
```

The settings are saved to configBaseline array in the MATLAB workspace.



	1	2	3
1	Block	Parameter	Value
2	Solver Configuration	Use local solver	off
3	Solver Configuration	Solver type	NE_BACKWARD_EULER_ADVANCER
4	Solver Configuration	Sample time	Ts
5	Solver Configuration	Use fixed-cost runtime consistency iterations	off
6	Machine	Zero sequence	pe.enum.park.zerosequence.include

The settings of interest for the Solver Configuration block are:

- **Use local solver** — The option to use a local Simscape solver is cleared.
- **Solver type** — Backward Euler, a Simscape local fixed- cost solver, is specified. However, if you open the block dialog box, you can see that it is not enabled because the option to use a local solver is cleared.
- **Use fixed-cost runtime consistency iterations** — The option to use fixed-cost is cleared. This option is also disabled when the option to use a local solver is cleared.

For the machine, the **Zero sequence** parameter is set to `Include`. Zero-sequence equations can cause numerical difficulty when you simulate with the Partitioning solver.

- 3 Mark the rotor torque signal, which connects the **trqMotor** From block to a Mux block, for Simulink data logging and viewing with the Simulation Data Inspector (SDI).

See Code

```
% Define the trqMotor From block and the path
%   to it as variables
torqueSensor = 'From6';
torqueSensorPath = [model, '/', torqueSensor];

% Mark the output signal from the trqMotor From block
%   for Simulink(R) data logging
phTorqueSensor = get_param(torqueSensorPath, 'PortHandles');
set_param(phTorqueSensor.Outport(1), 'DataLogging', 'on')
```

The logging badge  marks the signal in the model.

- 4 Determine the results and how long it takes to simulate with the baseline settings.

See Code

```
% Run a timed simulation using the Baseline solver configuration
tic;
sim(model);
tBaseline = toc;
```

- 5 Use `pe_updateSolver` function to change to the Backward Euler solver configuration. Save the configuration settings, and compare the settings to the baseline settings.

See Code

```
% Configure for Backward Euler solver simulation
pe_updateSolver('Backward Euler', model)

% Save the new parameter settings and compare them to the baseline
% configuration.

% Create a cell array that contains configuration data
configBackEuler = {'Block', 'Parameter', 'Value';
                  'Solver Configuration', 'Use local solver', ...
                  get_param(solvConfigPath, 'UseLocalSolver');
                  'Solver Configuration', 'Solver type', ...
```

```

get_param(solvConfigPath,'LocalSolverChoice');
'Solver Configuration','Sample time',...
get_param(solvConfigPath,'LocalSolverSampleTime');
'Solver Configuration',...
'Use fixed-cost runtime consistency iterations',...
get_param(solvConfigPath,'DoFixedCost');
'Machine','Zero sequence',...
get_param(machinePath,'zero_sequence'));

% Compare the Partitioning solver block settings to the Baseline settings
configDiff = setdiff(configBackEuler,configBaseline)

configDiff =

    1x1 cell array

    {'on'}
    
```

	1	2	3
1	Block	Parameter	Value
2	Solver Configuration	Use local solver	on
3	Solver Configuration	Solver type	NE_BACKWARD_EULER_ADVANCER
4	Solver Configuration	Sample time	Ts
5	Solver Configuration	Use fixed-cost runtime consistency iterations	on
6	Machine	Zero sequence	pe.enum.park.zerosequence.include

The option to use the local solver, which is set to Backward Euler by default, and the option to use fixed-cost runtime consistency iterations are now both selected.

- 6 Run a timed simulation using the Backward Euler solver.

See Code

```

tic;
sim(model)
tBackEuler = toc;
    
```

- 7 If you change the local solver to the Partitioning solver and simulate the model now, an error occurs because of the zero-sequence terms. Use the `pe_updateSolver` function to configure the model for simulating with the Partitioning solver without generating an error. Save the configuration settings, compare the settings to baseline settings, and run a timed simulation.

See Code

```
% Configure for Partitioning solver simulation
pe_updateSolver('Partitioning', model)

% Create a cell array that contains configuration data
configPartitioning = {'Block','Parameter','Value';
    'Solver Configuration','Use local solver',...
    get_param(solvConfigPath,'UseLocalSolver');
    'Solver Configuration','Solver type',...
    get_param(solvConfigPath,'LocalSolverChoice');
    'Solver Configuration','Sample time',...
    get_param(solvConfigPath,'LocalSolverSampleTime');
    'Solver Configuration',...
    'Use fixed-cost runtime consistency iterations',...
    get_param(solvConfigPath,'DoFixedCost');
    'Machine','Zero sequence',...
    get_param(machinePath,'zero_sequence')};

% Compare the Partitioning solver block settings to the Baseline settings
configDiff = setdiff(configPartitioning,configBaseline)

% Run a timed simulation using the Partitioning solver
tic;
sim(model)
tPartitioning = toc;

configDiff =

3x1 cell array

    {'NE_PARTITIONING_ADVANCER'      }
    {'on'                            }
    {'pe.enum.park.zerosequence.exclude'}
```

Warning: Initial conditions for nondifferential variables not supported. The following states may deviate from requested initial conditions

```
['pe_pmsm_drive/Battery']
    In elec.sources.battery_base
['pe_pmsm_drive/Permanent Magnet Synchronous Motor']
```

configPartitioning			
6x3 cell			
	1	2	3
1	Block	Parameter	Value
2	Solver Configuration	Use local solver	on
3	Solver Configuration	Solver type	NE_PARTITIONING_ADVANCER
4	Solver Configuration	Sample time	Ts
5	Solver Configuration	Use fixed-cost runtime consistency iterations	on
6	Machine	Zero sequence	pe.enum.park.zerosequence.exclude

The solver type is now set to the Partitioning solver and the machine is configured to exclude zero-sequence terms.

The simulation runs without generating an error. It does generate a warning because initial conditions for nondifferential variables are not supported for the Partitioning solver.

8 Print tables that show:

- Simulation time for each solver
- Percent differences in speed for the local solvers versus the baseline global solver.

See Code

```
% Display the simulation times
compTimeDiffTable = table({'Baseline';...
    'Backward Euler';...
    'Partitioning'},...
    {tBaseline;tBackEuler;tPartitioning},...
    'VariableNames', {'Solver','Sim_Duration'});

display(compTimeDiffTable);

% Compute and display the percent difference for the simulation times
spdBackEulerVsBaseline = 100*(tBaseline - tBackEuler)/tBaseline;
spdPartitionVsBaseline = 100*(tBaseline - tPartitioning)/tBaseline;

compPctDiffTable = table({'Backward Euler versus Baseline';...
    'Partitioning versus Baseline'},...
    {spdBackEulerVsBaseline;...
    spdPartitionVsBaseline},...
```

```
'VariableNames', {'Comparison','Percent_Difference'});
display(compPctDiffTable);
compTimeDiffTable =
    3x2 table
           Solver      Sim_Duration
    _____  _____
    'Baseline'      [36.4557]
    'Backward Euler' [22.9982]
    'Partitioning'  [ 9.7051]
compPctDiffTable =
    2x2 table
           Comparison      Percent_Difference
    _____  _____
    'Backward Euler versus Baseline' [36.9147]
    'Partitioning versus Baseline'   [73.3783]
```

Simulation time on your machine may differ because simulation speed depends on machine processing power and the computational cost of concurrent processes. The local fixed-step Partitioning and Backward Euler solvers are faster than the baseline solver, which is a global, variable-step solver. The Partitioning solver is faster than the Backward Euler solver.

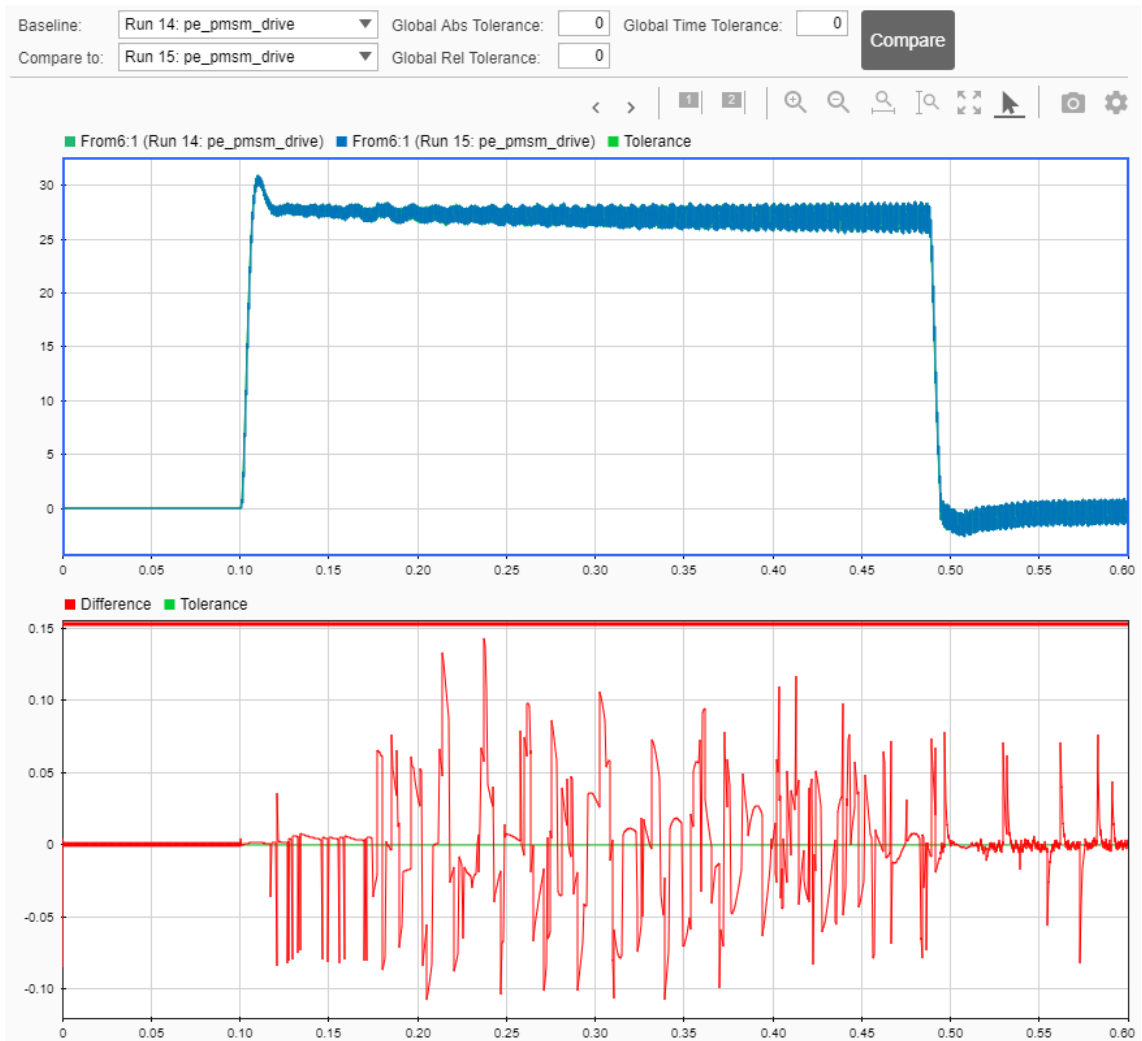
- 9 Compare the results using the Simulation Data Inspector.

See Code

```
% Get Simulink Data Inspector run IDs for
%   the last three runs
runIDs = Simulink.sdi.getAllRunIDs;
runBackEuler = runIDs(end - 1);
runPartition = runIDs(end);

% Open the Simulink Data Inspector
Simulink.sdi.view

compBaselinePartition = Simulink.sdi.compareRuns(runBackEuler,...
    runPartition);
```

The first plot shows the overlay of the Backward Euler and Partitioning solver simulation results. The second plot shows how they differ. The default tolerance for differences is 0. To determine if the accuracy of the results meets your requirements, you can adjust the relative, absolute, and time tolerances. For more information, see “Compare Simulation Data” (Simulink).

You can also use the `pe_updateSolver` function to reset the model for simulation with a global solver.

See Code

```
% Configure for Global/Nonlocal solver simulation  
pe_updateSolver('Global',model)
```

Limitations of the `pe_updateSolver` Function

Using the `pe_updateSolver` function does not guarantee that a simulation runs does not generate an error or that a simulation produces accurate results. To ensure that simulation accuracy meets your requirements, it is a recommended practice to compare simulation results to baseline results whenever you change model or block settings.

See Also

Permanent Magnet Synchronous Motor | Solver Configuration

Related Examples

- “Increase Simulation Speed Using the Partitioning Solver” (Simscape)

Prepare Simscape Power Systems Models for Real-Time Simulation Using Simscape Checks

If you have a Simulink Real-Time license, you can optimize your model for real-time execution using the `Execute real-time application` activity mode in the Simulink Performance Advisor. This mode includes several checks specific to physical models. For example, the Simulink Performance Advisor identifies Simscape Solver Configuration blocks with settings that are suboptimal for real-time simulation. For optimal results, Solver Configuration blocks should have the **Use local solver** and **Use fixed-cost runtime consistency iterations** options selected.

The checks are organized into folders. You can use the checks in the **Simscape checks** folder for all physical models. Subfolders contain checks that target blocks from Simscape Power Systems and other add-on products such as Simscape Driveline™ and Simscape Electronics.

Before you run the checks, use the processes described in “Real-Time Model Preparation Workflow” (Simscape), “Real-Time Simulation Workflow” (Simscape), and “Hardware-In-The-Loop Simulation Workflow” (Simscape).

To run the Simulink Real-Time Performance Advisor Checks:

- 1 In the Simulink Editor menu bar, select **Analysis > Performance Tools > Performance Advisor**.
- 2 In the Performance Advisor window, under **Activity**, select `Execute real-time application`.
- 3 In the left pane, expand the **Real-Time** folder, and then the **Simscape checks** folder.
- 4 Run the top-level Simscape checks and the Simscape Power Systems checks. If your model contains blocks from other add-on products, also run the checks in the subfolder corresponding to that product.

See Also

More About

- “Model Preparation Objectives” (Simscape)
- “Real-Time Model Preparation Workflow” (Simscape)

- “Real-Time Simulation Workflow” (Simscape)
- “Use Performance Advisor to Improve Simulation Efficiency” (Simulink)

Phasor-Mode Simulation in Simscape Components

You can run your model in phasor mode to speed up simulation. In Simscape, phasor mode is known as frequency-time equation formulation. In general, this formulation leads to accurate simulation of AC models using larger time steps than the traditional time formulation.

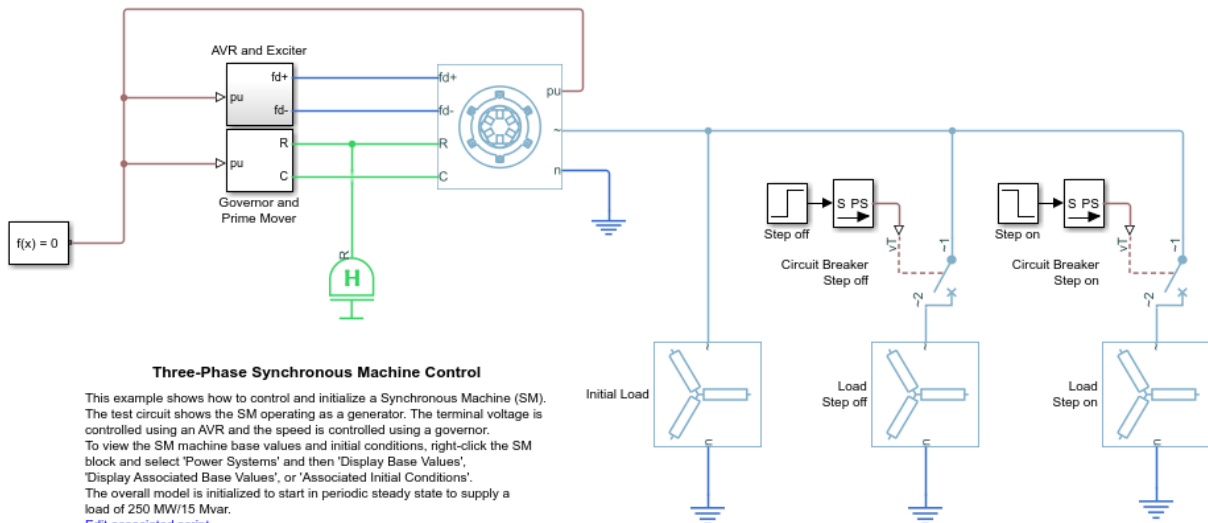
Use frequency-time equation formulation to speed up your simulation when:

- Your simulation contains periodic AC signals with a common fundamental frequency
- You are interested in the slow-moving AC-related quantities, such as amplitude or phase, and the DC output signals

Set up the model

To measure the time required to run a simulation, open the model *pe_sm_control* and create a model callback.

```
mdl = load_system('pe_sm_control');
open_system(mdl);
set_param(mdl, 'StartFcn', 'tic;');
set_param(mdl, 'StopFcn', 'tsim=toc;');
```



Run a time-based simulation

Double-click the Solver Configuration block and apply the following configuration:

- Enable the local solver by checking the **Use local solver** check box
- Set the `Sample time` parameter to `1e-3`
- Set the `Equation formulation` parameter to `Time`

You can also run this code to configure the block.

```
blk = find_system mdl, 'MaskType', 'Solver Configuration');  
set_param(blk, 'UseLocalSolver', 'on');  
set_param(blk, 'LocalSolverSampleTime', '1e-3');  
set_param(blk, 'EquationFormulation', 'NE_TIME_EF');
```

Simulate the model and save the run time and logging variable.

```
sim(get_param mdl, 'Name');  
tsim_time = round(tsim, 2);  
pe_sm_control_simlog_time = pe_sm_control_simlog;
```

Run a phasor-mode simulation

Double-click the Solver Configuration block and apply the following configuration:

- Enable the local solver by checking the `Use local solver` check box
- Set the `Sample time` parameter to `1e-2`
- Set the `Equation formulation` parameter to `Frequency and time`

You can also run this code to configure the block.

```
blk = find_system mdl, 'name', 'Solver Configuration');  
set_param(blk, 'UseLocalSolver', 'on');  
set_param(blk, 'LocalSolverSampleTime', '1e-2');  
set_param(blk, 'EquationFormulation', 'NE_FREQUENCY_TIME_EF');
```

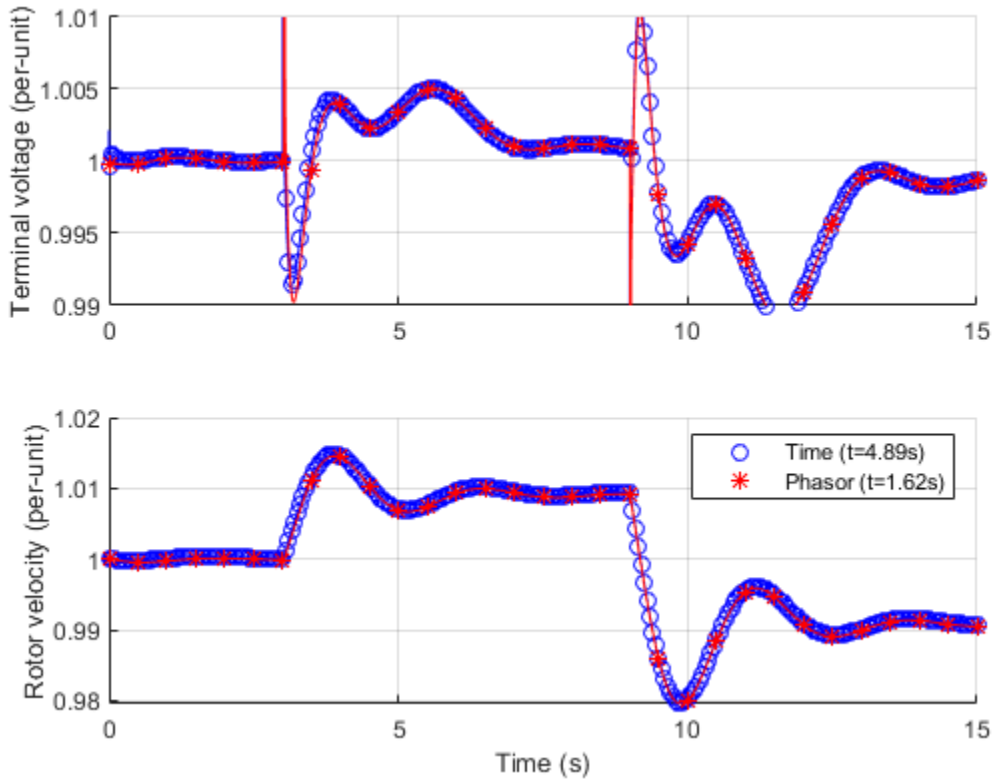
Simulate the model and save the run time and logging variable.

```
sim(get_param mdl, 'Name');  
tsim_phasor = round(tsim, 2);  
pe_sm_control_simlog_phasor = pe_sm_control_simlog;
```

Compare DC results

Plot the field voltage and rotor speed for both the time and frequency-time simulations. For each simulation mode, display markers at every 50 data points.

```
[hTime,hPhasor]=setup_figure(pe_sm_control_simlog_time,pe_sm_control_simlog_phasor,'dc');
legend([hTime,hPhasor],{'Time (t=',num2str(tsim_time),'s)'},{'Phasor (t=',num2str(tsim_time),'s)'});
```

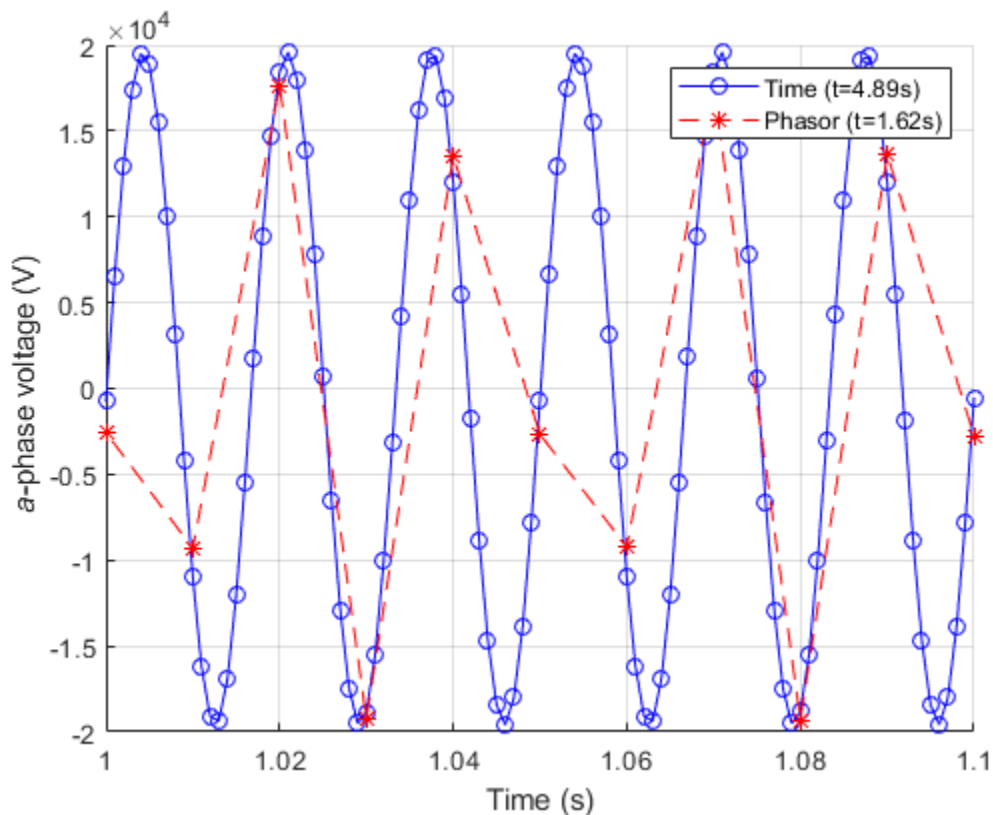


The phasor simulation reproduces near-identical results as the time-based simulation, despite using a time step that is 10 times larger. The measured simulation time is also shown for each of the simulation modes in the plot legend. This performance indicator is different on different machines, but the frequency-time simulation should be about two times faster than the time simulation. Note that the actual time required per step is higher in the frequency-time case, but the overall time is reduced.

Compare AC results

Plot the a -phase voltage of the synchronous machine over the time period 1s to 1.1s. Because of the larger time steps in the frequency-time formulation, the resolution of the AC quantity is too small to make out the sine wave. The points that are available are undersampled, but still accurate.

```
[hTime,hPhasor]=setup_figure(pe_sm_control_simlog_time,pe_sm_control_simlog_phasor,'ac');
legend([hTime,hPhasor],{'Time (t=',num2str(tsim_time),'s)'},{'Phasor (t=',num2str(tsim_time),'s)'}
```



In general, use frequency-time formulation to speed up simulations where the outputs of interest are DC or slow-moving AC quantities. You can use periodic sensors to measure slow-moving properties of AC signals such as amplitude and phase in both time and frequency time formulations. For more information, see the Harmonic Estimator block.

Sometimes there are small phase offsets between time- and frequency-time-generated AC signals. This difference is caused by the accumulated integration error of a slightly different signal frequency over time.

See Also

Solver Configuration

More About

- “Frequency and Time Simulation Mode” (Simscape)

