Power System Blockset

For Use with SIMULINK®

Hydro-Québec
TEQSIM International

Modeling

Simulation

Implementation

User's Guide
Version 1

How to Contact The MathWorks:

508-647-7000	Phone

508-647-7001	Fax

The MathWorks, Inc. 24 Prime Park Way Natick, MA 01760-1500	Mail

`http://www.mathworks.com`	Web
`ftp.mathworks.com`	Anonymous FTP server
`comp.soft-sys.matlab`	Newsgroup

`support@mathworks.com`	Technical support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`subscribe@mathworks.com`	Subscribing user registration
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

Power System Blockset User's Guide

Printing History: January 1998 First printing
January 1999 Revised for Version 1.1 (Release 11) (Online only)

Contents

Preface

1 Tutorial

2 Case Studies

3 Advanced Topics

4 Block Reference

Index

Preface

About the Authors

The Power System Blockset was developed by the following people and organizations:

- Hydro-Québec Research Institute (IREQ) - Varennes, Québec
- Teqsim International Inc. - St. Bruno, Québec
- Ecole de Technologie Superieure - Montreal
- Universite Laval - Québec City

This project was managed by Pierre Mercier. Gilbert Sybille was the technical coordinator and author of some sections. The technical supervision of the blockset was done by Kamal Al-Haddad, Hoang Le-Huy, Louis A. Dessaint, and Momcilo Gavrilovic. Patrice Brunelle, Roger Champagne, Christian Dufour, and Mohamed Tou programmed the functions and the models.

The Power System Blockset

Electrical power systems are combinations of electrical circuits and electro-mechanical devices, like motors and generators. Engineers working in this discipline are frequently tasked to improve the performance of the systems. Requirements for drastically increased efficiency have forced power system designers to use power electronic devices and sophisticated control system concepts that tax traditional analysis tools and techniques. Further complicating the analyst's role is the fact that the system is often so nonlinear that the only way to understand it is through simulation.

Land-based power generation from hydroelectric, steam, or other devices is not the only use of power systems. A common attribute of these systems is their use of power electronics and control systems to achieve their performance objectives.

The Power System Blockset was designed to provide a modern design tool that will allow scientists and engineers to rapidly and easily build models that simulate power systems. The blockset uses the Simulink® environment, allowing a model to be built using simple *click and drag* procedures. Not only can the circuit topology be drawn rapidly, but the analysis of the circuit can include its interactions with mechanical, thermal, control, and other disciplines. This is possible because all the electrical parts of the simulation interact with Simulink's extensive modeling library. Because Simulink uses MATLAB® as the computational engine, you can use MATLAB's toolboxes as you design your simulation.

You will find that the blockset can be put to work rapidly. The libraries contain 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. The capabilities of the blockset for modeling a typical electrical grid are illustrated in demonstration files. For users who want to refresh their knowledge of power system theory, there are also self-learning case studies.

The blockset fits well with other specialized analytical tools you use in the power system community.

1

Tutorial

Introduction

To master the Power Systems Blockset, you must learn how to build and simulate electrical circuits. The Power System Blockset operates in the Simulink environment. Before starting this training you should become familiar with Simulink.

The tutorial is organized in five different sessions. Sessions 1 through 3 are based on a simple power system. Session 4 illustrates power electronics, and Session 5 demonstrates three-phase power systems, electrical machinery, and load flow.

Session 1: Simulating a Simple Circuit

The Power System Blockset allows you to build and simulate electrical circuits containing linear and nonlinear elements. In Sessions 1 through 3 you will build, analyze, and simulate the circuit of Figure 1-1.

In this session, you will:

- Explore the powerlib library of the Power System Blockset
- Learn how to build a simple circuit from the powerlib library
- Interconnect Simulink blocks with your circuit

The circuit of Figure 1-1 represents an equivalent power system feeding a 300 km transmission line. The line is compensated by a shunt inductor at its receiving end. A circuit breaker allows energizing and de-energizing of the line. To simplify matters, only one of the three phases is represented. The parameters shown in the figure are typical of a 735 kV power system.

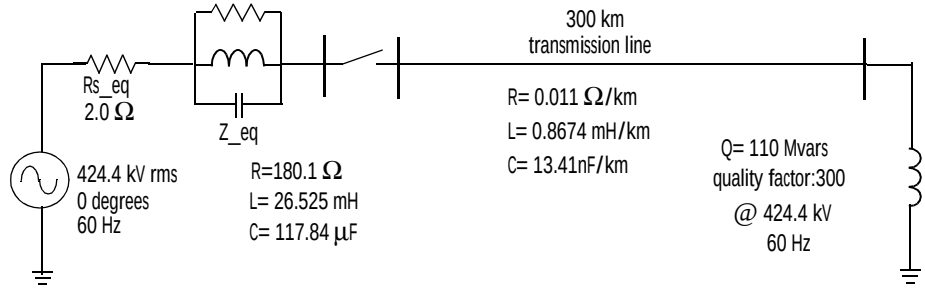


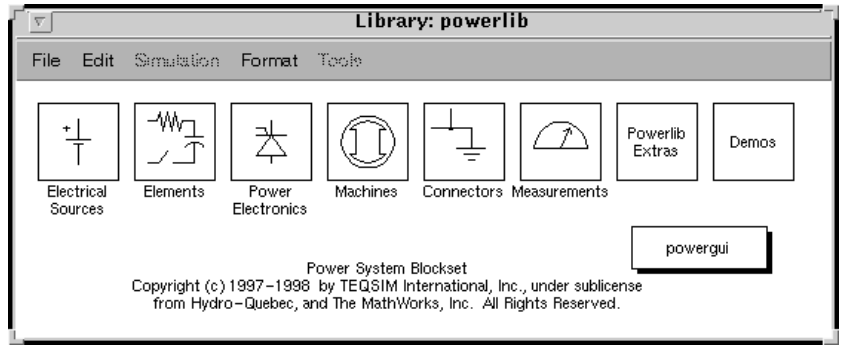
Figure 1-1 Circuit To Be Modeled With the Power System Blockset

The graphical interface makes use of the Simulink functionality to interconnect various electrical components. The electrical components are grouped in a special library called powerlib.

Open the Power System library by entering the following command at the MATLAB prompt:

```
powerlib
```

This command displays a Simulink window showing icons of different subsystem blocks.



These subsystems can be opened to produce the windows containing the blocks to be copied into your circuit. Each component is represented by a special icon having one or several inputs and outputs corresponding to the different terminals of the component.

1. From the **File** menu of the **powerlib** window, select New – Model, which will contain your first circuit and save it as `circuit1`.

2. Open the Electrical Sources library and copy the AC Voltage Source block into the **circuit1** window.

3. Open the **AC Voltage Source** dialog box by double clicking on the icon and enter the Amplitude, Phase, and Frequency parameters according to the values shown in Figure 1-1.

   Note that the amplitude to be specified for a sinusoidal source is its peak value (424.4e3*sqrt(2) volts in this case).

4. Change the name of this block from Voltage Source to `Vs`.

5. Copy the Parallel RLC Branch block, which can be found in the Elements library of powerlib, set its parameters as shown in Figure 1-1, and name it `Z_eq`.

   The resistance block cannot be found directly in the Elements subsystem. However it can be obtained from the Parallel RLC Branch block.

6 Duplicate the RLC block, which is already in your **circuit1** window, set the R parameter according to Figure 1-1, and set the L and C parameters respectively to infinite (`inf`) and zero values.

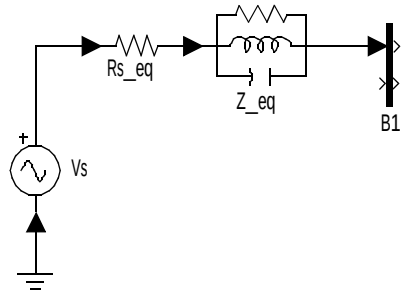
   When the dialog box is closed, you will notice that the L and C components have disappeared so that the icon now shows a single resistor. The same result would have been obtained with the Series RLC Branch block by setting L and C respectively at zero and inf values.

7 Name this resistance `Rs_eq`.

8 Open the Connectors subsystem of powerlib and copy a Bus Bar.

9 Open the **Bus Bar** dialog box and set its parameters at 2 inputs and 2 outputs and name it `B1`. Also copy the Ground block (select the block with an output connection).

Resize the various components and interconnect blocks by dragging lines from outputs to inputs of appropriate blocks.



You need a voltage measurement to measure the voltage at bus `B1`. This voltage measurement can be found in the Measurements subsystem. Copy it and name it `U1`.

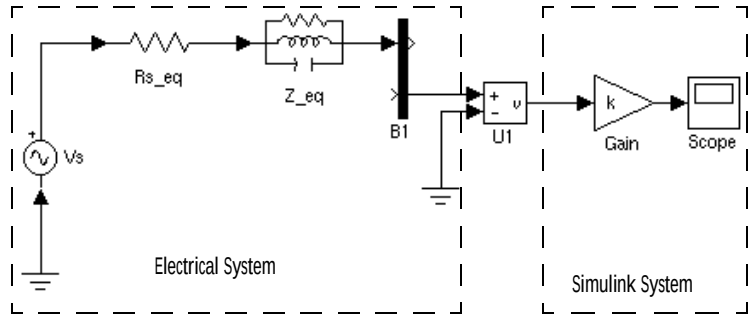
Connect its positive input to the second output of bus bar `B1` and its negative input to a new ground block.

Thus far you have used only blocks from the electrical library. To observe the voltage measured by the voltage measurement block, you need a display system. This could be any device found in the Sinks Subsystem block of Simulink. Use the standard Simulink scope.

Open the Simulink library and copy the scope in your **circuit1** window. If the scope were connected directly at the output of the voltage measurement, it would display the voltage in volts. However, electrical engineers in power systems are used to working with normalized quantities (per unit system). You can normalize the voltage by dividing the value in volts by a base voltage corresponding to the peak value of the system nominal voltage. In this case, the scaling factor *K* is:

$$K = \frac{1}{424.4 \times 10^3 \times \sqrt{2}}$$

Copy a Gain block from the Simulink library and set its gain as above. Connect the measurement system to the voltage measurement as shown below.



The measurement therefore appears as an interface between the electrical blocks and the Simulink blocks. For the simple system shown above, you do the communication in one direction only (from the electrical system to Simulink). However, for electrical systems requiring control, such as power electronic devices, you would perform the communication in both directions.

To complete the circuit of Figure 1-1, you need to add a transmission line, and a shunt reactor. You will add the circuit breaker later in “Session 3: Simulating Transients”.

The model of a line with uniformly distributed R, L, and C parameters normally consists of a delay equal to the wave propagation time along the line. This model cannot be simulated as a linear system because a delay corresponds to an infinite number of states. However, you can obtain a good approximation of the line with a finite number of states by cascading several pi circuits, each representing a small section of the line. A pi section consists of a series R-L

branch and two shunt C branches. The model accuracy depends on the number of pi sections used for the model.

Copy the PI Section Line block from the Elements library into `circuit1`. Open its dialog box. Set its parameters as shown in Figure 1-1, and specify one line section.

You can model the shunt reactor with a series RL branch. Use a Series RLC Branch block or a Series RLC Load block for the model. For the Branch block, set the R and L values corresponding to the active and reactive power specified in Figure 1-1 (`Q=110 Mvar; P=110/300=0.37 MW` at `V=424.4 kV rms` and `f= 60 Hz`).

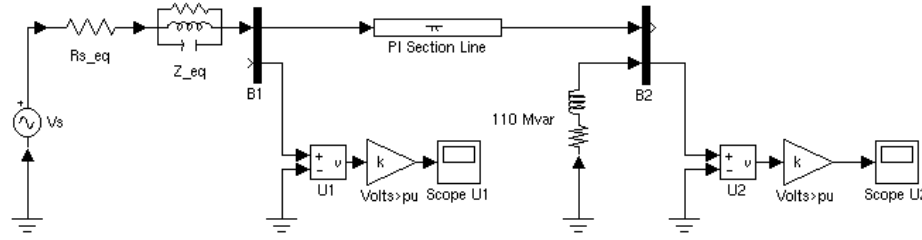
You will find it more convenient to use a Series RLC Load block which allows you to specify the active and reactive powers absorbed by the shunt reactor directly.

Copy the Series RLC Load block, which can be found in the Elements subsystem of powerlib. Open the dialog box and set its parameters as follows: `Vn=424.4e3V; fn=60 Hz; P= 110e6/300 W (quality factor=300); QL=110e6` vars and `Qc=0`.

Note that, as no reactive capacitive power has been specified, the capacitor disappears on the block icon when the dialog box is closed.

Add a receiving end bus B2 by duplicating B1. Duplicate the voltage measurement system (blocks U1, Volts>pu, and Scope U1).

Finally, interconnect all these new blocks as shown in the following figure.



Now you can start the simulation from the **Simulation** menu. As expected, voltages are sinusoidal with peak value of 1 pu. While the simulation is running open the **Vs block** dialog box and modify the amplitude. Observe the effect on the two scopes. Remember that you may zoom in on the waveforms in

the scope windows by rubberbanding (using the left mouse button) around the region of interest. You can also modify the frequency and the phase. Use one of the Simulink stiff solvers (like `ode15s`) for increased simulation.

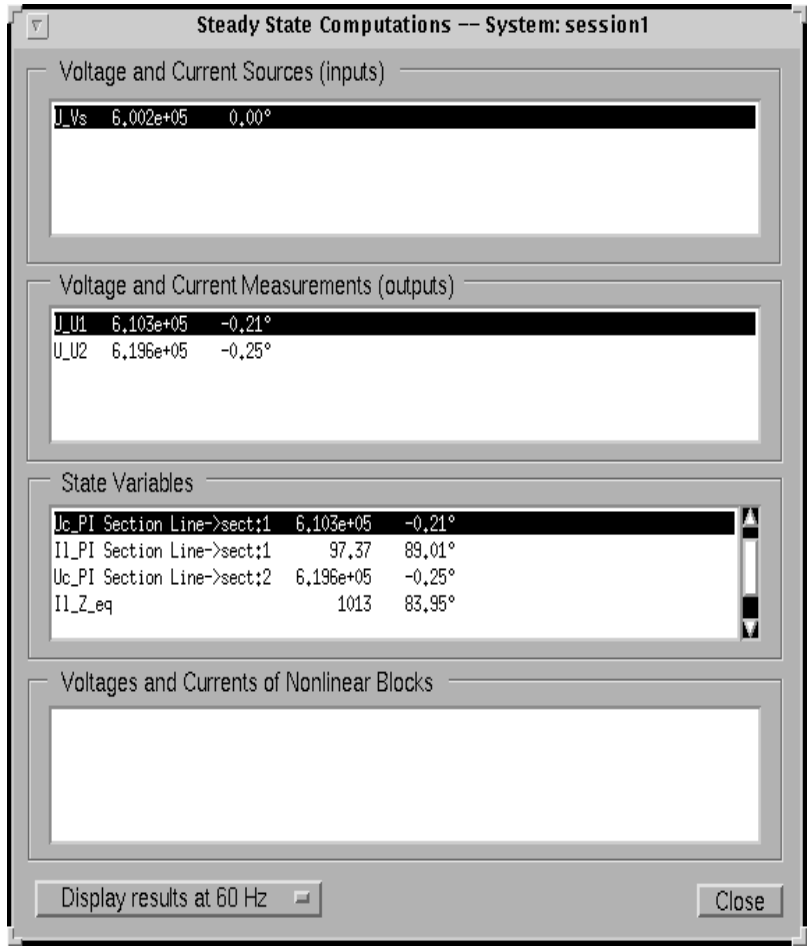
Session 2: Analyzing a Simple Circuit

In this session you will:

- Use the **Powergui** graphical user interface
- Obtain the steady-state outputs of the system
- Analyze your circuit with the `power2sys` function
- Analyze an electrical circuit in the frequency domain

To facilitate the steady-state analysis of your circuit, a graphical user interface (GUI) is provided in the powerlib library. Copy the **Powergui** interface block into your **circuit1** window and double click on the block icon to open it.

Select the **Steady state** button of **Powergui**. This opens the **Steady state** window where the steady-state phasor corresponding to the Vs electrical source is displayed in polar form. Each measurement output is identified by a string corresponding to the measurement block name, preceded by the prefix `U_` for a voltage measurement. The six state variables are also displayed in phasors corresponding to the steady states values of inductor currents and capacitor voltages. The variable names contain the name of the block where the inductor or capacitor is found, preceded by the `Il_` or `Uc_` prefix.

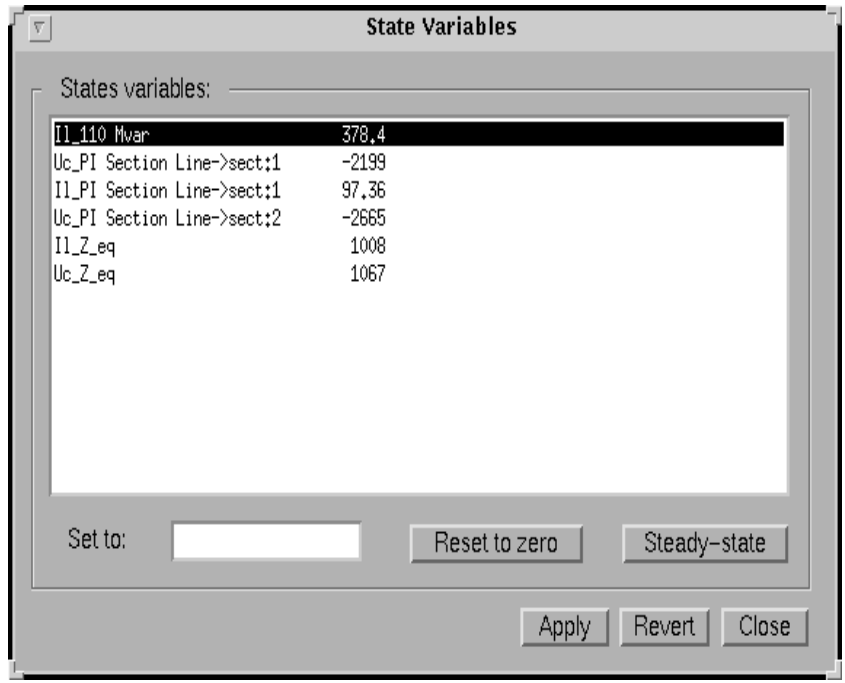


The sign conventions used for the voltages and currents of sources and state variables are determined by the orientation of the blocks:

- Inductor currents flowing in the arrow direction are considered positive
- Capacitor voltages are $V_{block\ output}$-$V_{block\ input}$

Note: Depending on the exact position of the various blocks in your `circuit1` diagram, the state variables may not be ordered the same way as in the figure above.

Now, select the **Set state variables** button to display the initial values of the state variables. These initial values are set in order to start the simulation in steady-state.



Frequency Analysis

To measure the impedance versus frequency at bus `B2`, you will need a current source to provide a second input to the state-space model. Open the Electrical Sources subsystem and copy the AC Current Source block in your model. Connect this source at bus `B2` as shown in Figure 1-2. Set its source amplitude to `0`, its frequency to `60 Hz`, and ground the negative port.

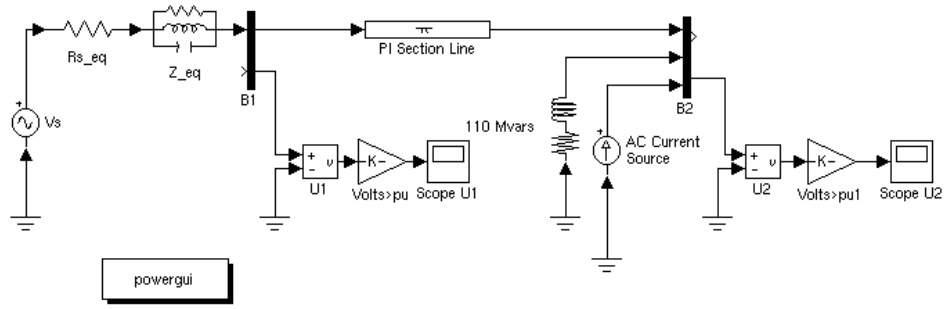


Figure 1-2 AC Current Source at the B2 Bus Bar

Now compute the state-space representation of the model `circuit1` with the `power2sys` function. Enter the following command at the MATLAB prompt.

```
[A,B,C,D,x0,states,inputs,outputs]=power2sys('circuitl');
```

The `power2sys` function returns the state-space model of your circuit in the four matrices A, B, C and D. `x0` is the vector of initial conditions that you have just displayed with the **Powergui**.The names of the state variables, inputs, and outputs are returned in three string matrices.

```
states

states =

Il_110 Mvars
Uc_PI Section Line->sect:1
Il_PI Section Line->sect:1
Uc_PI Section Line->sect:2
Il_Z_eq
Uc_Z_eq

inputs

inputs =

U_Vs
I_AC Current Source

outputs

outputs =

U_U1
U_U2
```

Note that you could have obtained the names and ordering of the states, inputs and outputs directly from the **Powergui**.

Once the state-space model of the system is known, it can be analyzed in the frequency domain. For example, the modes of this circuit can be found from the eigenvalues of matrix A (use the MATLAB `eig` command).

```
eig(A)
ans =

1.0e+05 *
-2.4972
-0.0001 + 0.0144i <---229 Hz
-0.0001 - 0.0144i
-0.0002 + 0.0056i <---89 Hz
-0.0002 - 0.0056i
-0.0000
```

This system has two oscillatory modes at 89 Hz and 229 Hz. The 89 Hz mode is due to the equivalent source, which is modeled by a single pole equivalent. The 229 Hz mode is the first mode of the line modeled by a single pi section.

If you have the Control System Toolbox, you can compute the impedance of the network as a function of frequency by using the `bode` function. In the Laplace domain, the impedance `Z2` at bus `B2` is defined as the transfer function between the current injected at bus `B2` (input 2 of the system) and the voltage measured at bus `B2` (output 2 of the system).

$$Z2(s) = \frac{U2(s)}{I2(s)}$$

The impedance at bus `B2` for the 0-1500Hz range can be calculated and visualized as follows.

```
freq=0:1500;

w=2*pi*freq;
[mag1,phase1]=bode(A,B,C,D,2,w);
semilogy(freq,mag1(:,2));
```

Repeat the same process to get the frequency response with a 10 line section model. Open the **PI Section Line** dialog box and change the number of sections

from 1 to 10. To calculate the new frequency response and superimpose it with the one obtained with a single line section, enter the following commands.

```
[A,B,C,D]=power2sys('circuit1');
[mag10,phase10]=bode(A,B,C,D,2,w);
hold on
semilogy(freq,mag1(:,2),freq,mag10(:,2));
```

The result is shown in Figure 1-3.

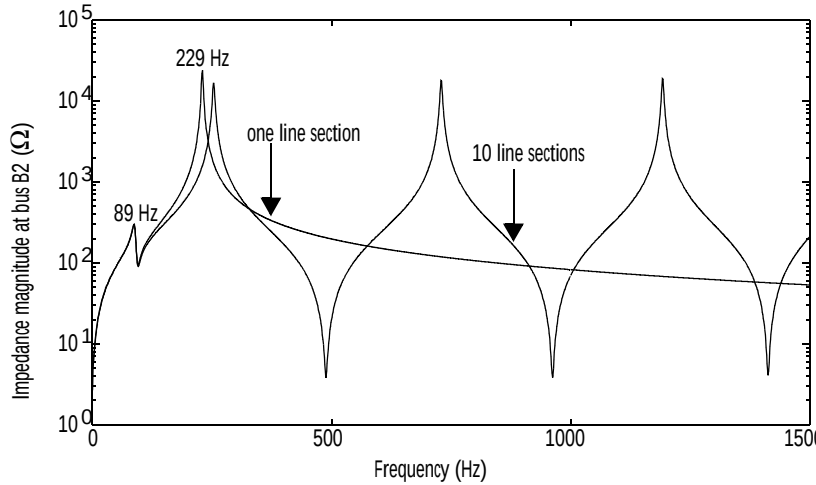


Figure 1-3 Impedance at Bus B2 as Function of Frequency

This graph indicates that the frequency range represented by the single line section model is limited to approximately 150 Hz. For higher frequencies, the 10 line section model is a better approximation.

For a distributed parameter line model, the propagation speed is:

$$v = \frac{1}{\sqrt{L \cdot C}} = 293\,208 \text{ km/s}$$

The propagation time for 300 km is therefore T=300/293 208=1.023 ms and the frequency of the first line mode is f1=1/4T=244 Hz. A distributed parameter line would have an infinite number of modes every 244+n*488 Hz (n=1,2, 3...).

The 10 line section model simulates the first 10 modes. The first three line modes can be seen on Figure 1-3, (244Hz, 732Hz and 1220 Hz).

Now open the **Simulation/Parameters** menu of your `circuit1` model. In the Solver section, select the `ode15s` integration algorithm. Keep the default parameters except set the stop time to `0.1`. Open the scopes and start the simulation. Look at the waveforms of the sending and receiving end voltages on Scope `U1` and Scope `U2`. As the state variables have been automatically initialized, the system starts in steady state and sinusoidal waveforms are observed.

Finally, open the **Powergui** interface, select the **Set State Variables** button, and reset all the states to `0` by selecting the **Reset to zero** button and then **Apply** button. Restart the simulation and observe the transient when the line is energized from zero.

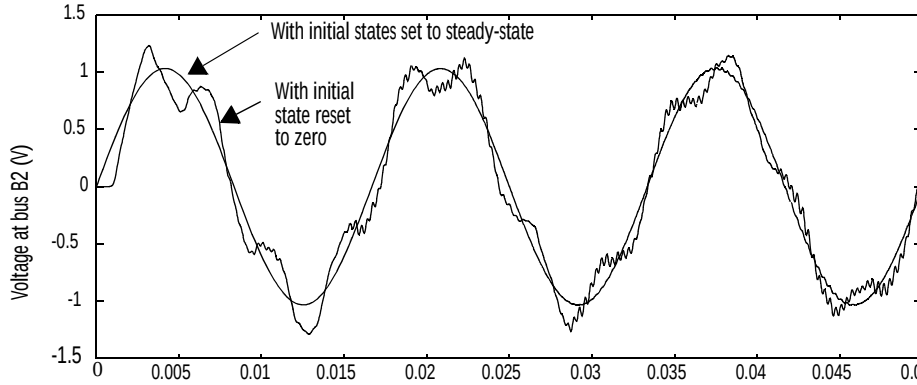


Figure 1-4 Receiving End Voltage U2 With 10 pi Section Line

Session 3: Simulating Transients

In this session you will:

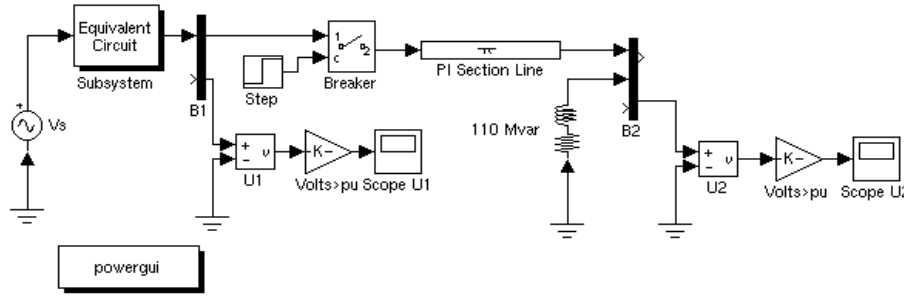
- Learn how to create an electrical subsystem
- Simulate transients with a circuit breaker
- Compare time domain simulation results with different line models

One of the main uses of the Power System Blockset is to simulate transients in electrical circuits. This can be done with either mechanical switches (circuit breakers) or switches using power electronic devices.

First open your circuit1 system and delete the current source connected at bus `B1`. Save this new system as `circuit2`. Before connecting a circuit breaker, you will modify the schematic diagram of `circuit2`. As with Simulink, the Power System Blockset allows you to group several components into a subsystem. This feature is useful to simplify complex schematic diagrams.

Use this feature to transform the source impedance into a subsystem by:

1. Select the two blocks identified Rs_eq and Z_eq by surrounding them with a bounding box (left mouse button) and use the **Edit/Create Subsystem** menu. The two blocks now form a new block called Subsystem.



2. Use the **Edit/Mask Subsystem** menu to change the icon of that subsystem. In the Icon section of the mask editor, type the following command:

   ```
   disp('Equivalent\nCircuit')
   ```

3. Select Apply.

4 Use the **Format/Show Drop shadow** menu to get the appearance as shown in the figure. You can now double click on the Subsystem block and look at its content.

5 Insert a circuit breaker in your circuit to simulate a line energization by opening the Elements subsystem of the powerlib library. Copy the Breaker block into your `circuit2` window.

The circuit breaker is a nonlinear element consisting of an ideal switch in series with a series R-L circuit. Because of modeling constraints, the switch inductance cannot be set to `0`. However, this inductance can be set to a very small value, say 10 μH, that does not affect the performance of the circuit:

1 Open the breaker dialog box and set its parameters as follows: `Ron=0 Ω; Lon=10e-6 H; and Initial state=0 (open)`.

2 Insert the circuit breaker in series with the sending end of the line, then rearrange the circuit as shown in the figure in step 1.

3 Copy the Step block from the Simulink Sources library and connect it to the control input of the circuit breaker.

4 Set the step time to `16.67e-3/4` so that the line will be energized at the positive peak source voltage.

5 Finally connect a To Workspace block from the Simulink Sinks library, at the output of the Gain block measuring `U2` and set the variable name as `U2`.

Open the **PI section Line** dialog box and make sure the number of sections is set to 1. Open the **Simulation/Parameters** menu. In the Solver section, select the `ode15s` integration algorithm. Keep the default parameters, except set the stop time to `0.02s`. Select the **Workspace I/O** tab and make sure that the time is returned in the variable `tout`. Open the scopes and start the simulation. Look at the waveforms of the sending and receiving end voltages on Scope `U1` and Scope `U2`.

Once the simulation is complete, copy the variables `tout` and `U2` into `t_1` and `U2_1` respectively.

```
t_1=tout;
U2_1=U2;
```

These two variables now contain the waveform obtained with a single pi section line model.

Open the **PI section Line** dialog box and change the number of sections from `1` to `10`. Start the simulation. Once the simulation is complete, copy the variables `tout` and `U2` into `t_10` and `U2_10` respectively.

```
t_10=tout;
U2_10=U2;
```

Finally, delete the pi section line model and replace it with a single phase Distributed Parameter Line block from the powerlib/Elements library. Set the number of phases to `1` and use the same R,L,C, and length parameters as for the pi section line (see Figure 1-1). Restart the simulation and save the time and `U2` voltage in `t_d` and `U2_d` variables.

You can now compare the three waveforms obtained with the three line models by plotting them on the same graph.

```
plot(t_1,U2_1,t_10,U2_10,t_d,U2_d);
```

These waveforms are shown in Figure 1-5. As expected from the frequency analysis performed during “Session 2: Analyzing a Simple Circuit”, the single pi section model does not respond to frequencies higher than 229 Hz. The 10 pi section model gives a better accuracy although high frequency oscillations are introduced by the discretization of the line. You can clearly see on the figure the propagation time delay of 1.03 ms associated with the distributed parameter line.

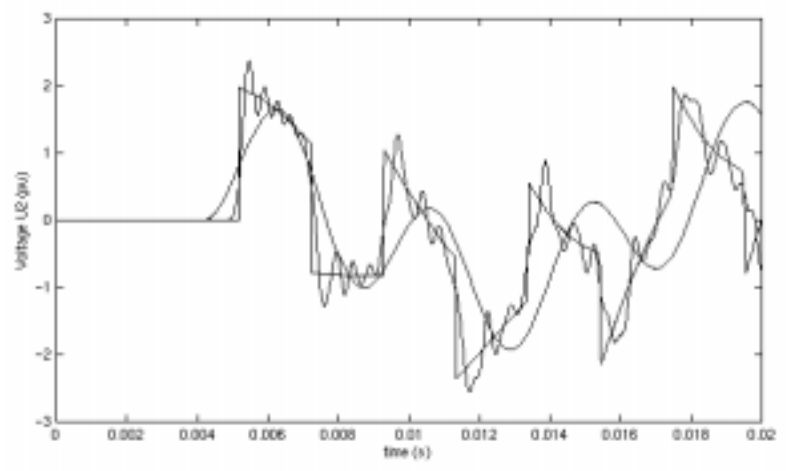


Figure 1-5 Receiving End Voltage Obtained With Three Different Line Models

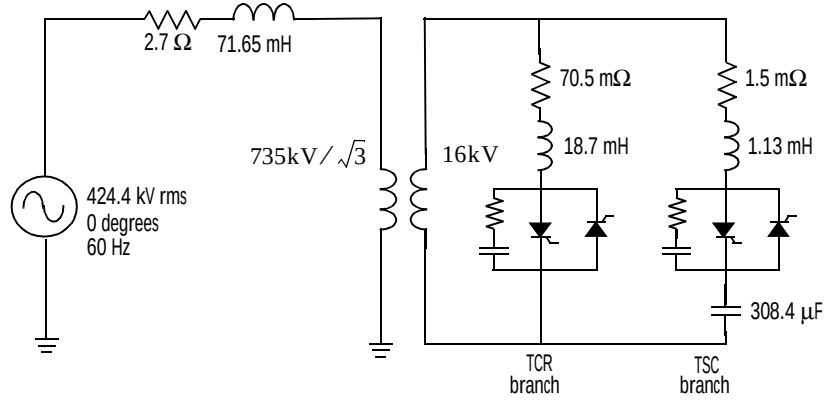
Session 4: Introducing Power Electronics

In this session you will:

- Learn how to use power electronic components
- Learn how to use transformers
- Change initial conditions of a circuit

The Power System Blockset has been designed to simulate power electronic devices. In this session, you will build a simple circuit using thyristors.

Consider the circuit given in Figure 1-6. It represents one phase of a static var compensator used on a 735kV transmission network. On the secondary of the 735 kV /16 kV transformer, two variable susceptance branches are connected in parallel: one thyristor controlled reactor (TCR) branch and one thyristor switched capacitor (TSC) branch.



Transformer parameters

Nominal power 110 MVA

Primary: Rated voltage 424.4 kV rms; leakage reactance=0.15 pu; resistance=0.002 pu

Secondary: Rated voltage 16 kV rms; leakage reactance=0 pu; resistance=0.002 pu

Magnetizing current at 1 pu voltage: Inductive: 0.2% Resistive: 0.2%

Thyristor parameters

Ron = 1mΩ Lon= 1µH; Vf=14*0.8 V (14 thyristors in series)

Snubber: Rs= 500Ω Cs= 0.15µF

Figure 1-6 One Phase of a TCR/TSC Static Var Compensator (SVC)

The TCR and TSC branches are both controlled by a valve consisting of two thyristor strings connected in antiparallel. An RC snubber circuit is connected across each valve. The TSC branch is switched on/off, thus providing discrete step variation of the SVC capacitive current. The TCR branch is phase controlled in order to obtain a continuous variation of the net SVC reactive current.

You will now build two circuits illustrating the operation of the TCR and the TSC branches.

Simulation of the TCR Branch

1. Open a new model and save it as `circuit3`.
2. Open the Power Electronics library and copy the Thyristor block into your `circuit3` model.
3. Open the **Thyristor** menu and set the parameters as follows:

   `(Ron=1e-3; Lon=1e-6; Vf=14*0.8; Rs=500; Cs=0.15e-6)`.

   Notice that the snubber circuit is integral to the **Thyristor** dialog box.
4. Rename this block `Th1` and duplicate it.
5. Connect this new thyristor `Th2` in antiparallel with `Th1` one as shown in Figure 1-7.

   As the snubber circuit has already been specified with `Th1`, the snubber of Th2 must be eliminated.
6. Open the `Th2` dialog box and set the snubber parameters to `Rs=Inf; Cs=0`.

   Notice that the snubber disappears on the `Th2` icon.

The linear transformer is located in the Elements library. Copy it, rename it to `TrA`, and open its dialog box. Set its nominal power, frequency, and winding parameters (`winding 1= primary; winding 2 = secondary`), as shown in Figure 1-6.

Note that the leakage reactance and resistance of each winding have to be specified directly in per unit quantities. As there is no tertiary winding, type `0`

in the field corresponding to winding 3. Notice that winding 3 disappears on the TrA block.

Finally, set the magnetizing branch parameters Rm and Xm at `[500, 500]`. These values correspond to 0.2% resistive and inductive currents, as specified in Figure 1-6.

Add a voltage source, series RL elements, and a ground block. Set the parameters as shown in Figure 1-6. Add a current measurement to measure the primary current. By using appropriate connectors, you should be able to interconnect the circuit as shown in Figure 1-7.

Notice that the thyristor blocks have an output identified by the letter "m." This output returns a Simulink vectorized signal containing the thyristor current `Iak` and voltage `Vak`. Connect a Demultiplexer with two outputs at the m output of `Th1`. Then connect the two multiplexer outputs to two oscilloscopes that you name `Scope_Ith1` and `Scope_Uth1`.

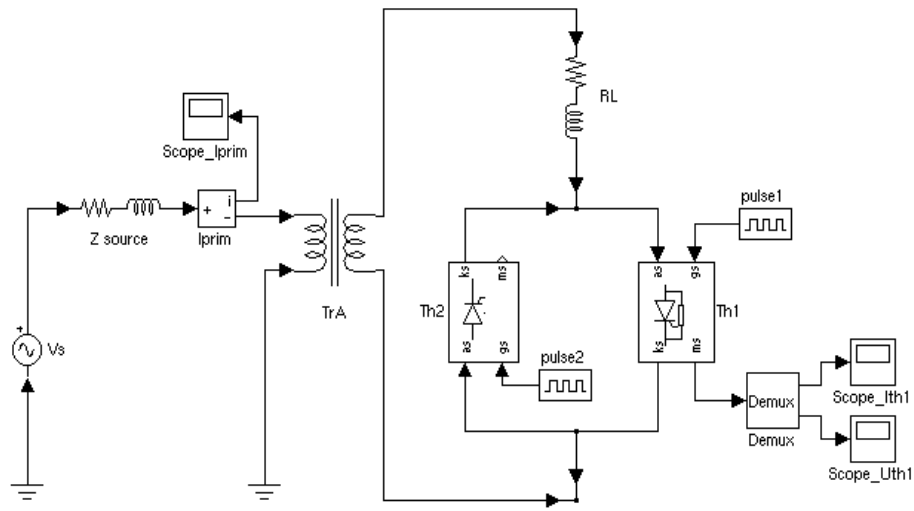


Figure 1-7 Simulation of the TCR branch

You will now model the synchronized pulse generators firing thyristors `Th1` and `Th2`. Copy two Simulink pulse generators into your system, name them `Pulse1` and `Pulse2`, and connect them to the gates of `Th1` and `Th2`.

Now you have to define the timing of the Th1 and Th2 pulses. At every cycle a pulse has to be sent to each thyristor α degrees after the zero crossing of the thyristor commutation voltage. Set the pulse1 and pulse2 parameters as follows:

```
Period : 1/60 s
Duty cycle: 1% (3.6 degrees pulses)
Amplitude :1
Start time : 1/60+T for Pulse1; 1/60+1/120+T for Pulse2
```

The pulses sent to Th1 are delayed by 180 degrees with respect to pulses sent to Th2. The delay T is used to specify the α firing angle. To get a 120 degrees firing angle, specify T in the workspace by typing

```
T=1/60/3;
```

Now open the **Simulation/Parameters** menu. Select the ode15s integration algorithm. Keep the default parameters, except set the stop time to 0.1. Open the three oscilloscopes and adjust the scales as follows: Time range 0.1; Iprim -100/+100A; Ith1: -500/+2000A; Uth1:-40000/+40000 V.

Start the simulation. The results are as shown in Figure 1-8.

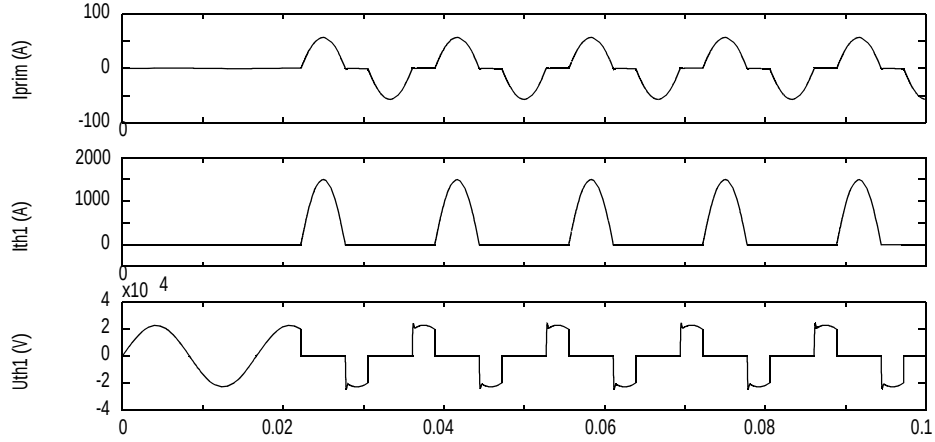


Figure 1-8 TCR Simulation Results

Simulation of the TSC Branch

You can now modify the TCR branch to create a TSC branch. To do this, save `circuit3` as a new system and name it `circuit4`.

Connect a capacitor in series with the RL inductor and `Th1`/`Th2` valve as shown in Figure 1-9. Change the R,L, and C parameters as shown in Figure 1-6. Connect a voltmeter and oscilloscope to monitor the voltage across the capacitor.

Contrary to the TCR branch, which was fired by a synchronous pulse generator, a continuous firing signal will now be applied to the two thyristors. Delete the two pulse generators. Copy a Step block from the Simulink/Sources library and connect its output at both gates of `Th1` and `Th2`. Set its step time at 1/60/4 (energizing at the first positive peak of the source voltage). Your circuit should now be similar to the one shown in Figure 1-9.

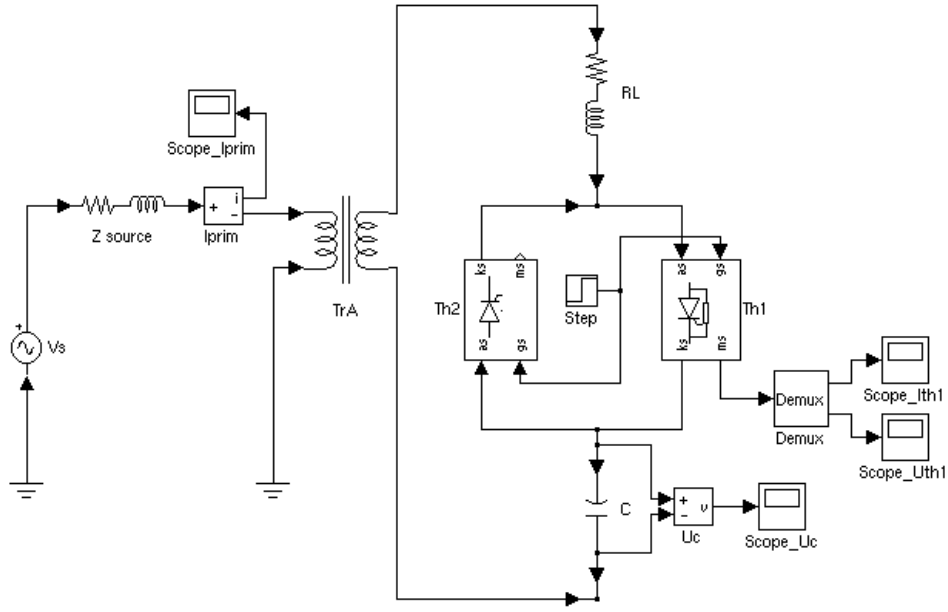


Figure 1-9 Simulation of the TSC Branch

Open the four oscilloscopes and adjust the scales as follows:

```
Time range 0.1 s; Iprim -500/+500A; Ith1: -1000/+15000/ A; Uth1:
-30000/+30000 V.; Uc: -50000/+50000.
```

Start the simulation.

As the capacitor is energized from zero, you can observe a low damping transient at 200Hz, superimposed with the 60 Hz component in the capacitor voltage and primary current. During ordinary TSC operation the capacitor will have an initial voltage left because of the last valve opening. To minimize the closing transient with a charged capacitor, the thyristors of the TSC branch must be fired with the correct polarity when the source voltage is at maximum value. The initial capacitor voltage corresponds to the steady-state voltage obtained when the thyristor switch is closed. The capacitor voltage is 17.67 kVrms when the valve is conducting. At the closing, the capacitor must be charged at the peak voltage.

$$Uc = 17670 \times \sqrt{2} = 24989\,Volts$$

You can now use the **Powergui** interface to change the capacitor's initial voltage. Open the **Powergui** interface. In the Set state variables section, a menu is available. A list of all the state variables with their default initial values appears. The value of the initial voltage across the capacitor C (variable Uc_C) should be -0.3141 V. This voltage is not exactly zero because the snubber allows circulation of a small current when both thyristors are blocked. Now select the Uc_C state variable and enter `24989` in the set to field. Then select the **Apply** button to make this change effective.

Start the simulation. As expected, the transient component of capacitor voltage and current have disappeared. The voltages obtained with and without initial voltage are compared in Figure 1-10.

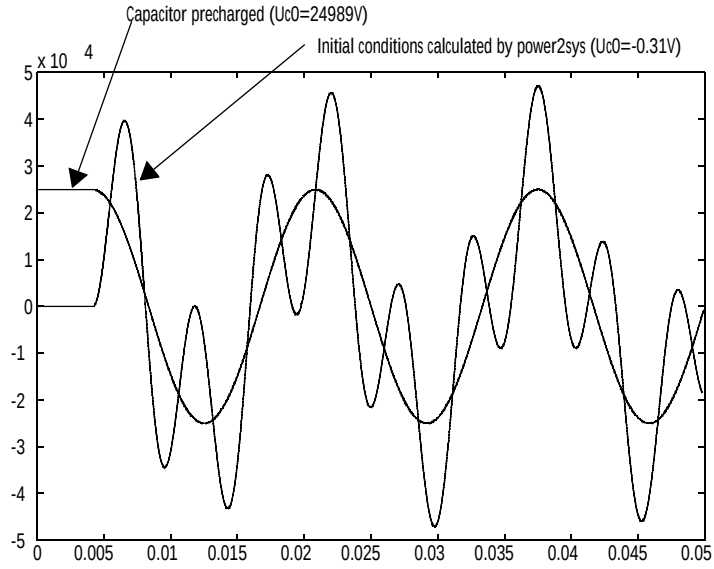


Figure 1-10 Transient Capacitor Voltage With and Without Initial Charge

Session 5: Simulating Three-Phase Systems and Using Electrical Machines

In this session you will:

- Learn how to use electrical machines
- Use the three-phase library
- Initialize machines to start simulation in steady state and use the **Machine Load Flow** option of the **Powergui**.

The Machines library of powerlib contains four of the most commonly used three-phase machines: simplified and complete synchronous machines, asynchronous machines and permanent magnet synchronous machines. Each machine can be used either in generator or motor mode. Combined with linear and nonlinear elements such as transformers, lines, loads, and breakers. they can be used to simulate electromechanical transients in an electrical network. They can also be combined with power electronic devices to simulate drives.

System Description

During this session you will simulate the three-machines system as shown in the single line diagram of Figure 1-11.

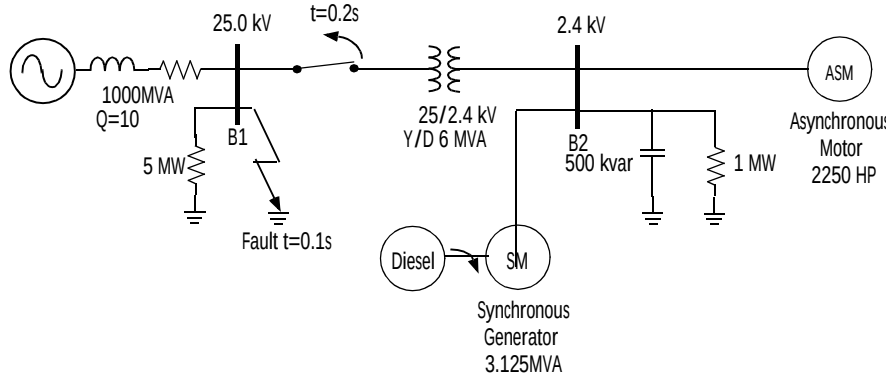


Figure 1-11 Diesel Generator and Asynchronous Motor on Distribution Network

This system consists of a plant (bus B2), simulated by a resistive and motor load (ASM) fed at 2.4 kV from a 25 kV distribution network through a 6 MVA 25/2.4 kV transformer, and from an emergency synchronous generator/diesel engine unit (SM). A 500 kvar capacitor bank is used for power factor correction at the 2.4 kV bus. The 25 kV network is modeled by a simple R-L equivalent source (short-circuit level 1000 MVA, quality factor X/R=10) and a 5 MW load. The asynchronous motor is rated 2250 HP, 2.4 kV, and the synchronous machine is rated 3.125 MVA, 2.4 kV.

Initially, the motor develops a mechanical power of 2000 HP and the diesel generator delivers 500 kW of active power. The synchronous machine controls the 2.4 kV B2 bus voltage at 1.0 pu and generates 500 kW of active power. At t=0.1s, a three-phase to ground fault occurs on the 25 kV system, causing the opening of the 25 kV circuit breaker at t=0.2s, and a sudden increase of the generator loading. During the transient period following the fault and islanding of the Motor/Generator system, the synchronous machine excitation system and the diesel speed governor will react to maintain the voltage and speed at a constant value.

This system has already been built with the Power System Blockset for load flow application. Open the Demos library of powerlib and double click on Machines and Load Flow (sim) demo.

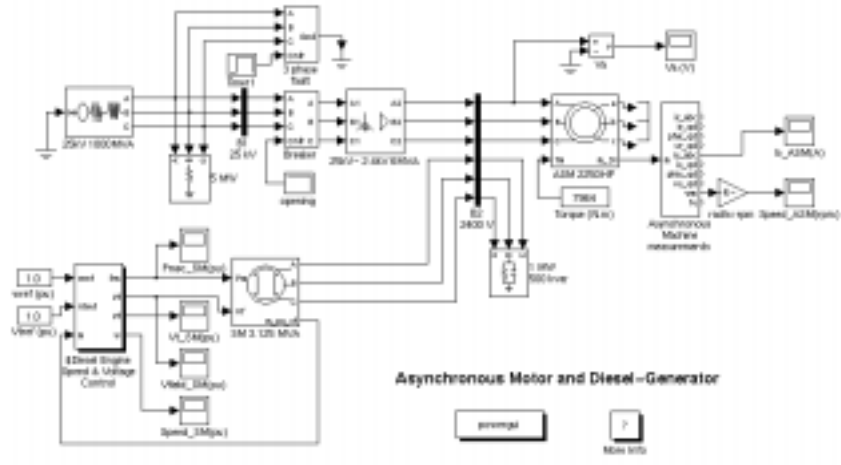


Figure 1-12 Power System of Figure 1-11 Built With the PSB

For the synchronous machine (SM), the block using standard parameters has been used, whereas the asynchronous motor (ASM) parameters are entered in S.I. units. The other three-phase elements (inductive voltage source, Y grounded/Delta transformer, and loads) are masked blocks built from the Three-Phase library provided in Extras. The fault and breaker systems are built with three single-phase circuit breakers. Special measurement blocks provided in the Machine library are used to demultiplex the SM and ASM machine outputs.

The SM voltage and speed outputs are used as feedback inputs to a Simulink control system that contains the diesel engine and governor block as well as an excitation block. The excitation system is the standard block provided in the Machines library. The SM parameters as well as the diesel engine and governor models are taken from reference [1].

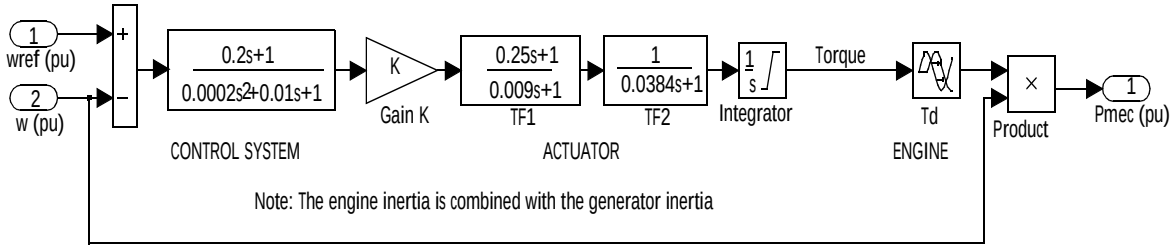


Figure 1-13: Diesel Engine and Governor System

If you simulate this system for the first time, you usually don't know what the initial conditions are for the SM and ASM to start in steady-state.

These initial conditions are:

SM: Initial values of speed deviation (usually 0%), rotor angle, magnitudes and phases of currents in stator windings, and initial field voltage required to obtain the desired terminal voltage under the specified load flow.

ASM: Initial values of slip, rotor angle, magnitudes, and phases of currents in stator windings.

Open the dialog box of the SM and ASM blocks. All initial conditions should be set at `0`, except for the initial SM field voltage $-V_f$, and ASM slip, which are set at `1 pu` Open the three oscilloscopes monitoring the SM and ASM speeds as well as the ASM stator currents. Start the simulation and observe the first 100 ms before fault is applied.

As the simulation starts, you will notice that the three ASM currents start from `0` and contain a slowly decaying DC component. The machine speeds will take a much longer time to stabilize because of the inertia of the motor/load and diesel/generator systems. In the example, the ASM even starts to rotate in the wrong direction because the motor starting torque is lower than the applied load torque. Stop the simulation.

To start the simulation in steady state with sinusoidal currents and constant speeds, all the machine states must be initialized properly. This is a difficult task to perform by hand, even for a simple system. In the next section you will learn how to use the Machine Load Flow option of the **Powergui** to perform a load flow and initialize the machines.

Select the **Machine Load Flow** button in the **Powergui**. A new window appears. Note that for the SM Bus type you have a menu allowing you to choose either PV Generator or Swing Generator.

For synchronous machines you ordinarily specify the desired terminal voltage and the active power that you want to generate (positive power for generator mode) or absorb (negative power for motor mode). This is possible as long as you have a *swing* or *slack bus* that will generate or absorb the excess power required to balance the active powers throughout the network.

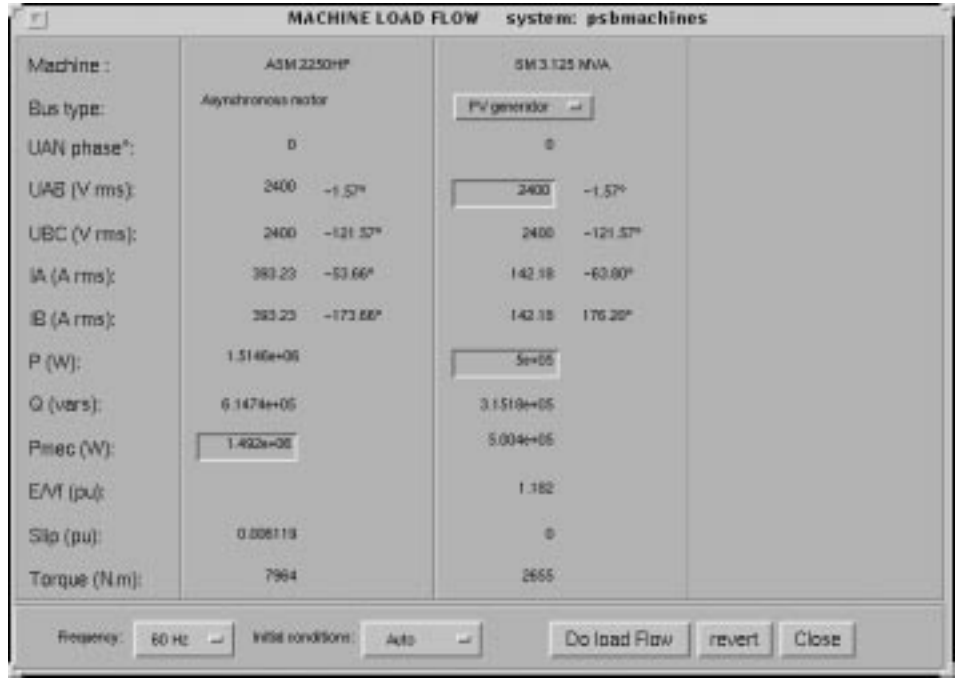
The swing bus can be either a voltage source or any synchronous machine. If you don't have any voltage source in your system, you must declare one of the machines as a swing machine. In the next section you will make a load flow with the 25 kV voltage source connected to bus B1 used as a swing bus.

Load Flow Without a Swing Machine

In the **Load Flow** window, your SM bus type should be already initialized as `PV generator` indicating that the load flow will be performed with the machine controlling its active power and terminal voltage. Specify the desired load flow by entering the following parameters:

```
SM:UAB (Vrms) = 2400; P(W)=500e3.
ASM:Pmec (W)= 2000*746.
```

Select the **Do Load Flow** button. Once the load flow is solved, the phasors of AB and BC machine voltages as well as currents flowing in phases A and B are updated as shown below.



The SM active and reactive powers, mechanical power, and field voltage are displayed.

```
SM:P=500 kW; Q =315 kvar;
Pmec=500.4 kW (or 500/3125=0.1601 pu)
Field voltage E/Vf=1.182 pu.
```

The ASM active and reactive powers absorbed by the motor, slip, and torque are also displayed.

```
ASM:P=1.515MW; Q =615 kvar; Pmec=1.492 MW (2000 HP)
Slip=0.006119pv; Torque=7964 N.m
```

This torque value should be already entered in the Constant block connected at the ASM torque input. If you now open the SM and ASM dialog boxes you

can see the updated initial conditions. If you select the **Steady-State** button in the **Powergui**, you will also see the updated values of the measurement outputs and voltages and currents of non-linear blocks. For example you should find that the magnitude of the Phase A voltage across the fault breaker (named `U_3 phase fault/Breaker1`) is 20.40 kV corresponding to a 24.985 kV rms phase-phase voltage.

To start the simulation in steady state, you should also initialize the states of the diesel engine /governor and SM excitation systems initialized according to the values calculated by the load flow. Open the Governor & Diesel Engine subsystem which is located inside the Diesel Engine Speed and Voltage Control subsystem**.** The initial mechanical power has been already set to `0.1601 pu` in the Governor.

Open the Excitation block. The initial terminal voltage and field voltage have been set, respectively, to `1.0` and `1.182 pu`. Note that the load flow automatically initializes the machine blocks but not the associated control blocks. Therefore, if you perform a new load flow you should change the initial values in the control blocks.

Open the four scopes displaying the terminal voltage, field voltage, mechanical power, and speed of the synchronous machine as well as the scope displaying the asynchronous motor speed. Start the simulation.

The simulation results are shown in Figure 1-14.

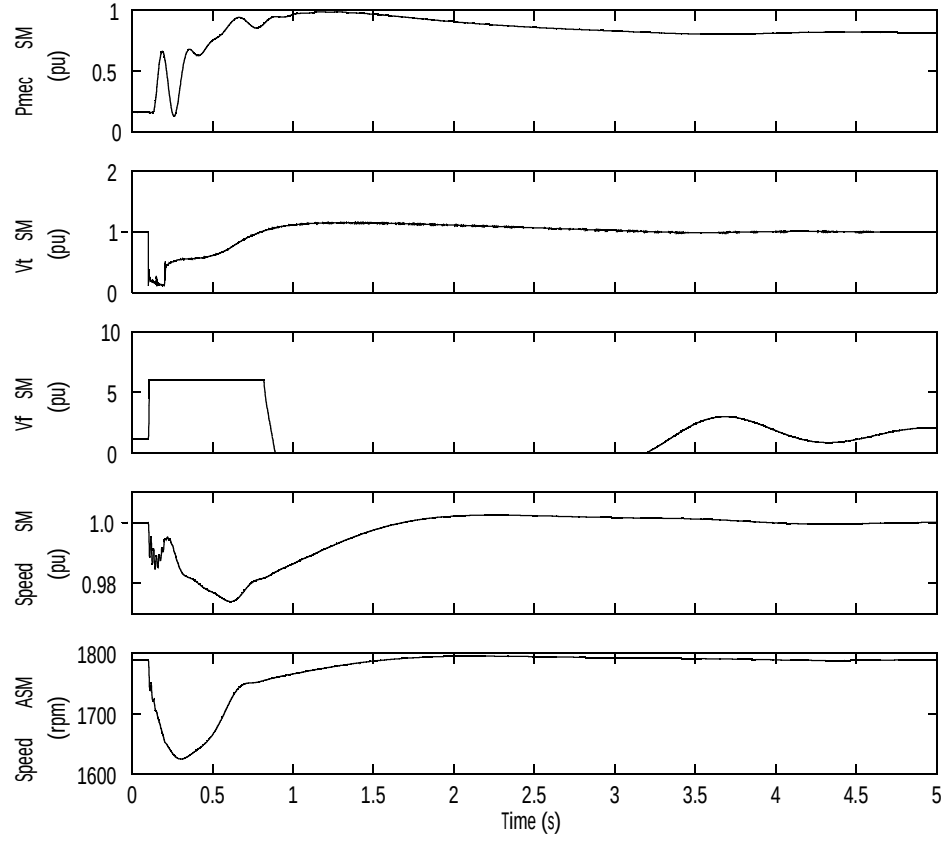


Figure 1-14 Simulation Results

Observe that, during the fault, the terminal voltage drops to about 0.2 pu and the excitation voltage hits the limit of 6 pu. After fault clearing and islanding, the SM mechanical power quickly increases from its initial value of 0.16 pu to 1 pu and stabilizes at the final value of 0.80 pu required by the resistive and motor load (1.0 MW resistive load + 1.51 MW motor load = 2.51 MW = 2.51/ 3.125= 0.80 pu). After three seconds, the terminal voltage stabilizes close to its reference value of 1.0 pu. The motor speed temporarily decreases from 1789 rpm down to 1625 rpm, then it recovers close to its normal value after two seconds.

If you increase the fault duration to 12 cycles by changing the breaker opening time to 0.3s, you will notice that the system collapses. The ASM speed slows to zero after two seconds.

Load Flow With a Swing Machine

In this section you will make a load flow with two machine types: a *PV generator* and a *Swing generator*. In your **psbmachines** model, delete the inductive source and replace it with the Simplified Synchronous Machine block (po units) that you will find in the Machines library. The two input ports and the last output port (m_pu) can be left unconnected. Rename it `SSM 1000MVA` and save this new system in your working directory as `psbmachine2`. Open the **SSM** dialog box and enter the following parameters.

1st line: `Pn(VA) Vn(Vrms) fn(Hz): [1000e6 25e3 60]`

2nd line: `H(sec) Kd() p () [inf 0 2]`

The inertia was specified to be infinite, so the speed and therefore the frequency of the machine will be kept constant.

3rd line: `R(pu) X(pu): [0.1 1.0]`

(Notice how easily you can specify an inductive short circuit level of 1000MVA and a quality factor 10 with the per unit system)

4th line: Leave all initial conditions at `0`.

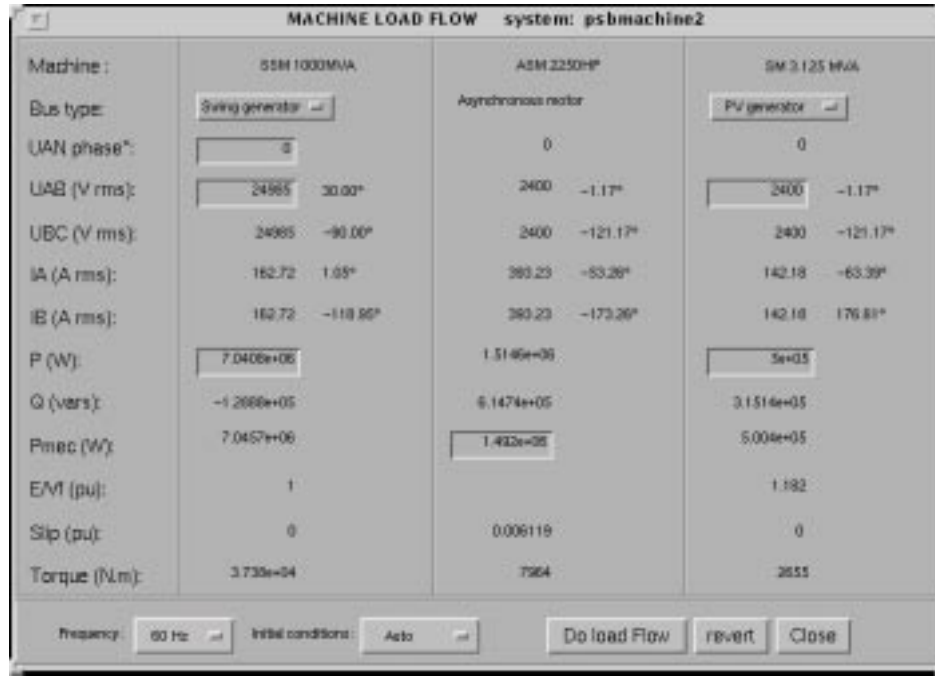
Select the **Machine Load Flow** button in the **Powergui**. Leave the SM bus type as `PV Generator` and change the SSM bus type to `Swing Generator`. Specify the load flow by entering the following parameters.

```
SM:UAB(Vrms) =2400; P(W)=500e3;
ASM:Pmec(W)= 2000*746.
```

For the SSM swing machine you only have to specify the requested terminal voltage (magnitude and phase). The active power is unknown. However you can specify an active power that will be used as an initial guess and help load flow convergence. Specify the following parameters:

`SSM:UAB(Vrms)=24985;` (Voltage obtained at bus B1 from the previous load flow)
`UAN Phase(degrees)=0;`
`Active power guess: P(W)=0;`

Select the **Do load Flow** button. Once the load flow is solved the following solution is displayed.



SM: Active and reactive electrical powers, mechanical power and field voltage:

```
P=500 kW; Q=315 kvar;
Pmec=500.4 KW (or 500/3125=0.1601 pu);
Field voltage E/Vf=1.182 pu
```

ASM: Active and reactive powers absorbed by the motor, slip and torque:

```
P=1.515MW; Q=615 kvar; Pmec=1.492 MW (2000 HP)
Slip=0.006119; Torque=7964 N.m
```

SSM: Active and reactive electrical powers, mechanical power and internal voltage:

```
P=7.041MW;Q=-129 kvar;
Pmec=7.046MW (or 7.046/1000=0.007046 pu); E/Vf=1.0 pu
```

As expected, the solution obtained is exactly the same as the one obtained with the inductive source. The active power delivered by the swing bus is 7.04 MW (6.0 MW resistive load + 1.51 MW load - 0.5 MW generated by SM = 7.01 MW, the difference (0.03 MW) corresponding to losses in the transformer).

Connect at inputs 1 and 2 of the SSM block two Constant blocks specifying respectively the required mechanical power (0.007046 pu) and internal voltage (1.0 pu). Restart the simulation. You should get the same waveforms as those of Figure 1-14.

References

[1] Yeager K.E, J.R. Willis “Modeling of Emergency Diesel Generators in an 800 Megawatt Nuclear Power Plant,” I*EEE Transactions on Energy Conversion*, Vol.8, No.3, September 1993.

2

Case Studies

The case studies in this section were built to provide examples of uses for the Power System Blockset. They are:

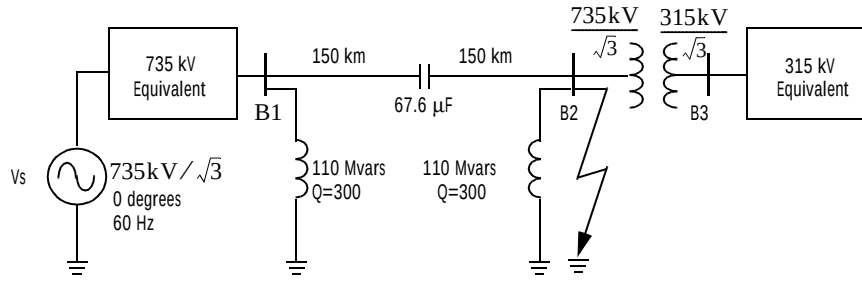
- Series Compensated Transmission Network
- Chopper Fed DC Motor Drive
- Synchronous Machine and Regulators
- Variable Frequency Induction Motor Drive
- HVDC Transmission System

Cases 1 and 5 are studies of AC and DC transmission on power systems. Cases 2 and 4 illustrate typical applications of the Power System Blockset to motor drives. Case 3 demonstrates the performance of a nonlinear voltage regulator on a synchronous alternator.

Series Compensated Transmission Network

The example in this section illustrates phenomena related to subsynchronous resonance in a series-compensated AC transmission network.

Description of the Transmission Network



Parameters: 735 kV equivalent network and line parameters (see Figure 1-1 of Tutorial)

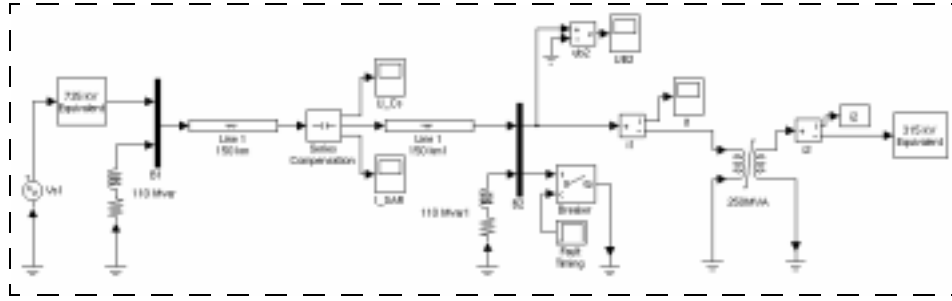
315 kV equivalent: See Subsystem2 on next page

Transformer: 250 MVA/phase; Primary/secondary leakage reactances: 0.15pu/0.0 pu

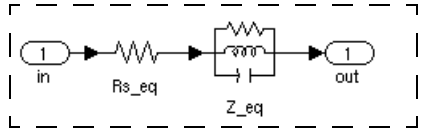
Saturation knee point: 1.20 pu; Xair core: 0.4 pu

Figure 2-1 Series Compensated Network

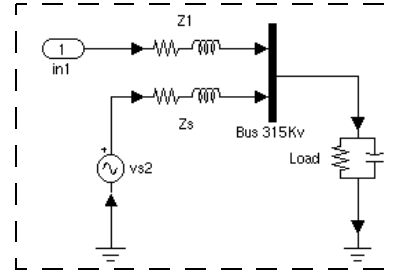
The system shown in Figure 2-1 consists of a 735 kV, 300 km transmission line fed by an equivalent system. The line is 40% series compensated at its center and it is shunt compensated at both ends. At its receiving end the line is connected to a 315 kV network through a 735/315 kV transformer. Only one phase of the system is represented. To study the transient behavior of this circuit when a phase-to-ground fault is applied at bus B2.



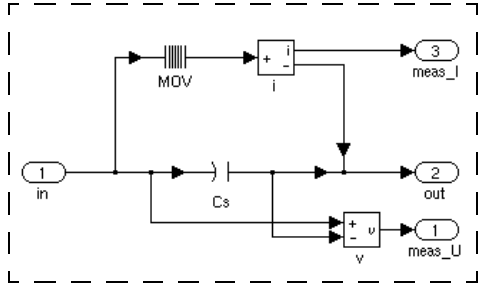
Series compensated network (psbcompensated)



Subsystem 1



Subsystem 2



Series Compensation

This network is available in the `psbcompensated.mdl` demonstration file. Load the `psbcompensated` system and save it in your working directory as `case1` to allow further modifications to the original system.

Compare the circuit modeled in the Power System Blockset with the schematic diagram of Figure 2-1. Note that the fault is simulated with a circuit breaker.

The circuit contains three nonlinear elements in the Series Compensation subsystem: the metal oxide varistor (MOV) protecting the series capacitor, the saturable transformer, and the circuit breaker.

Open the Series Compensation subsystem. The transmission line is 40% series compensated by a 67.6μF capacitor. The capacitor is protected by a metal oxide varistor (MOV block). If you open the dialog box of the MOV block, you will notice that it consists of 30 columns and that its protection level (specified at a reference current of 500 A/column) is set at 277 kV. This voltage corresponds to 2.5 times the nominal capacitor voltage obtained at a nominal current of 2 kA rms.

Open the **Transformer** dialog box and notice that the current-flux saturation characteristic has been set at

```
[0 0 ; 0.0012 1.2; 1 1.45] in pu
```

This data is the current and flux values at points 1, 2, and 3 of the piecewise linear approximation to the flux linkage curve shown in Figure 2-2.

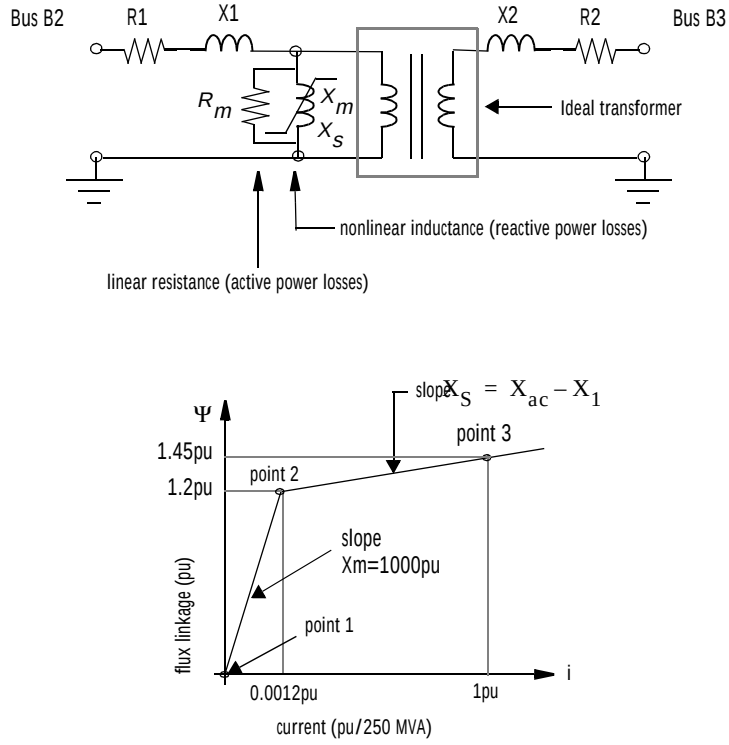


Figure 2-2 Saturable Transformer Model

The flux-current characteristic is approximated by two segments (see Figure 2-2). The saturation knee point is 1.2 pu. The first segment corresponds to the magnetizing characteristic in the linear region (for fluxes below 1.2 pu). At 1 pu voltage, the inductive magnetizing current is 0.0010/1.0= 0.001 pu, corresponding to 0.1% reactive power losses. The iron core losses (active power losses) are specified on the last line of the dialog box (resistance=1000 pu, corresponding to 0.1% losses).

The slope of the saturation characteristic in the saturated region is 0.25 pu. Therefore, taking into account the primary leakage reactance (X1=0.15 pu), the air core reactance of the transformer seen from the primary winding is 0.4pu/ 250 MVA).

Obtaining the Steady-State and State-Space Model

You can use the **Powergui** interface to find the steady-state measurements returned by the five voltage and current measurements.

You can also obtain the voltage and current across the three nonlinear elements. Notice that the current in the saturable branch of the transformer and in the MOV is zero because the linear circuit is initialized with these nonlinear elements disconnected.

If you look at the list of the state variables, you will find that the linear circuit consists of 18 states. Although the linear circuit comprises a total of 19 inductors and capacitors, it contains only 18 states. This is because the three capacitor voltages in the loop formed by the last capacitor of line 1, the series capacitor Cs, and the first capacitor of line 2 are not independent. Therefore one of the three capacitors UC_Cs is not an independent state variable.

The state-space model, state variable names, input names, and output names are contained respectively in variables `A`, `B`, `C`, `D`, `state_var`, `inputs` and `outputs` and are returned by `power2sys`.

```
[A,B,C,D,x0,state_var,inputs,outputs]=power2sys('psbcompensated');
```

In addition to the two voltage source inputs, the list of inputs contains the currents in the three nonlinear elements (last three lines of input matrix).

```
inputs =

U_Vs1
V_Subsystem2/vs2
I_Breaker
I_250MVA_core
I_Series Compensation/MOV
```

In addition to the outputs specified by the five voltage and current measurements, the list of outputs contains the voltages across the three nonlinear elements (first three lines of output matrix).

```
outputs =

U_Breaker
U_250MVA_core
U_Series Compensation/MOV
I_i1
I_i2
U_ub2
I_Series Compensation/I_MOV
U_Series Compensation/v
```

Frequency Analysis

Once the state-space model of the linear system is available, you can obtain the frequency response of the system. The I_Breaker input and the U_Breaker output are used to measure the impedance at bus `B2` where the circuit breaker is connected.

If you have the Control System Toolbox, you can compute the impedance of the network as function of frequency by using the `bode` function. In the Laplace domain, the impedance at bus `B2` is defined as the transfer function between the current injected at bus `B2` (input 3 of the system) and the voltage measured at the same bus (output 1 of the system)

$$Z(s) = \frac{U\_breaker(s)}{I\_Breaker(s)}$$

You can calculate and visualize the impedance at bus `B2` for the 0-500Hz range as follows:

```
freq=0:500;
w=2*pi*freq;
[mag,phase]=bode(A,B,C,D,3,w);
plot(freq,mag(:,1));
```

The impedance as a function of frequency is shown in Figure 2-3.

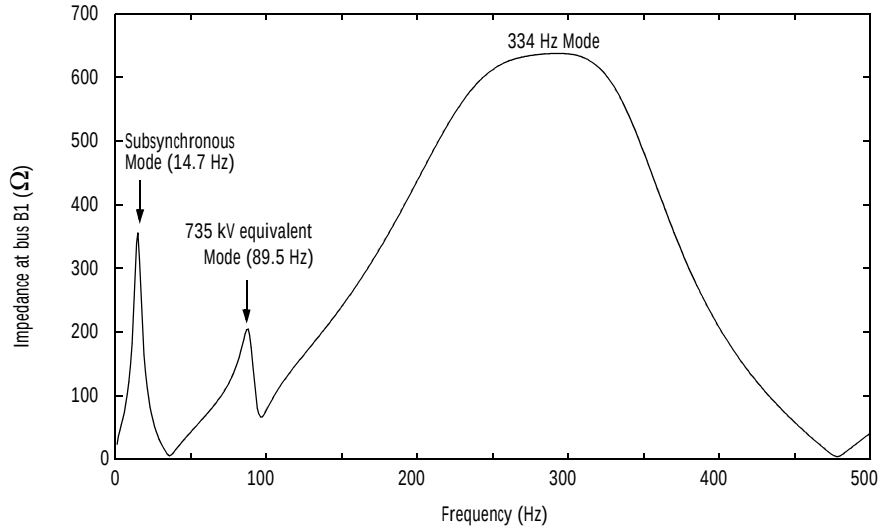


Figure 2-3 Impedance vs. Frequency Seen From Bus B2

The frequencies of the oscillatory modes can be found from the imaginary parts of the eigenvalues of matrix A

```
imag(eig(A))/2/pi

ans =

 1.0e+03 *

      0
      0
 1.2381
-1.2381
 1.0531
-1.0531
 0.7208
-0.7208
 0.3349
-0.3349
 0.2358
-0.2358
 0.0895
-0.0895
 0.0147
-0.0147
      0
      0
```

The 14.7, 89.5, and 334.9 Hz modes clearly appear on the impedance seen from bus `B1`. The 14.7 Hz mode is mainly due to a parallel resonance of the series capacitor with the shunt inductors. The 89.5 Hz mode is the first pole of the 735 kV equivalent network. These three modes are likely to be excited at fault clearing.

Transient Performance Under Fault Condition

Check that the Simulation parameters are set as follows:

```
Start time=0.0
Stop time=0.4
Solver: Variablestep, ode15s(stiff/NDF);
Max step size= auto
Initial step size= auto
Relative tolerance= 1e-3;
Absolute tolerance:0.01
Maximum order: 5
```

Open the four oscilloscopes and start the simulation.

The simulation results are plotted in the first four subplots (traces) in Figure 2-4. Observe the increase of transformer current I1 and capacitor voltage Ucs during the fault. The first peak of the IMOV current becomes symmetrical (+/- 4 kA). As the maximum current flowing in each column of the MOV(8.0/30 = 0.27 kA) is lower than the 500 A reference current used to specify the MOV protection voltage, the capacitor voltage is limited at a value slightly lower than the MOV protection level (277 kV).

The 14.7 Hz subsynchronous mode excited at fault clearing is clearly seen on Ub2 and Ucs. The 14.7 Hz voltage component appearing at bus `B2` drives the transformer into saturation. The transformer magnetizing current `Imag` plotted on the last trace of Figure 2-4 is calculated as the difference between the primary and secondary currents with the appropriate transformation ratio. It can be obtained by adding the appropriate blocks to the simulation to implement the equation

```
Imag= i1(:,2)-i2*315/735;
plot(Imag)
```

Alternatively, these variables are in the workspace and may be plotted from the MATLAB command line.

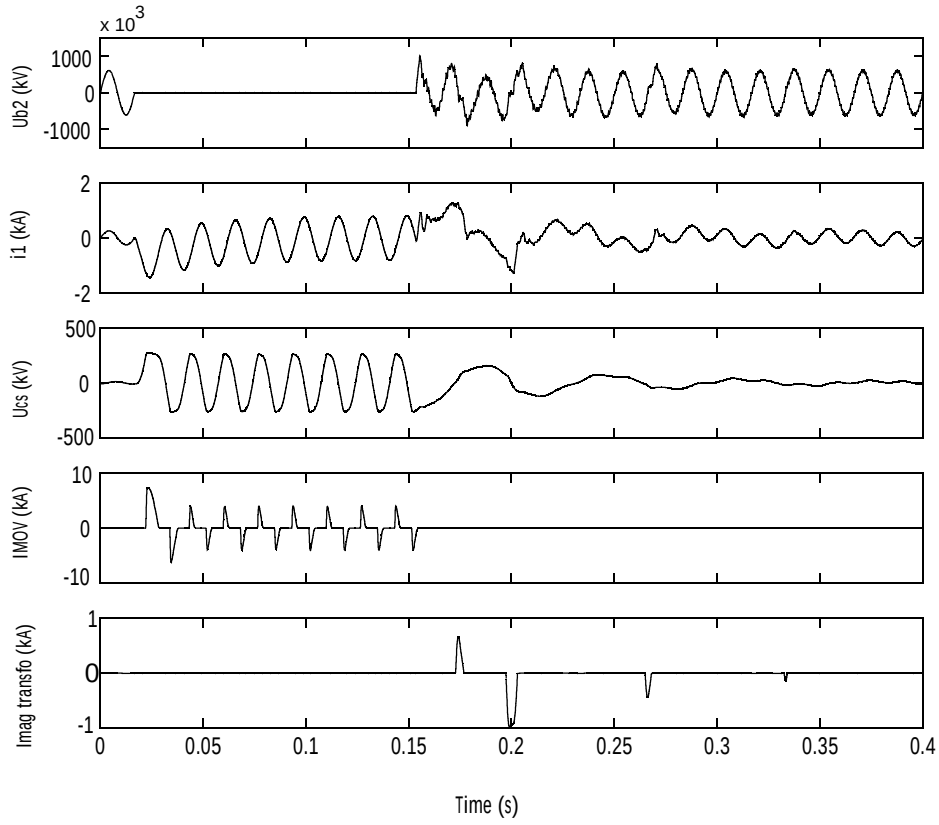


Figure 2-4 Voltage and Current Waveforms for the Series Compensation Case Study

Chopper-Fed DC Motor Drive

The example described in this section illustrates the application of the Power System Blockset to the operation of a DC motor drive in which the armature voltage is controlled by a GTO thyristor chopper.

The objective of this example is to demonstrate the use of electrical blocks, in combination with Simulink blocks, in the simulation of an electromechanical system with a control system. The electrical part of the DC motor drive including the DC source, the DC motor, and the chopper is built using blocks from the Elements and Power Electronics libraries. The mechanical part and the control system are built using Simulink blocks. The interface between the two is ensured by Measurement and Controlled Source blocks.

Description of the Drive System

A simplified diagram of the drive system is shown in Figure 2-5. The DC motor is fed by the DC source through a chopper that consists of the GTO thyristor, `Th1`, and the free-wheeling diode D1. The DC motor drives a mechanical load which is characterized by the inertia J, friction coefficient `B`, and load torque T_L.

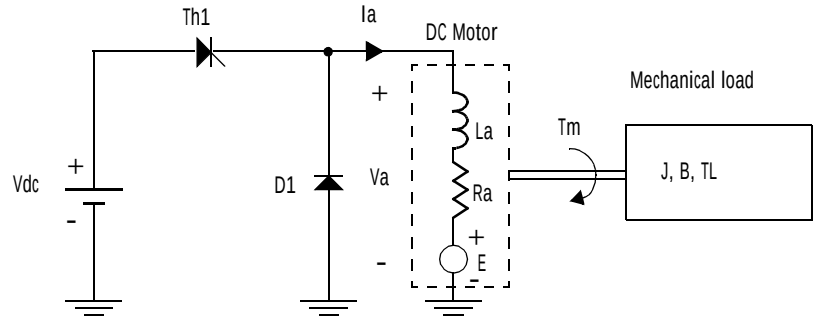


Figure 2-5 Chopper-fed DC Motor Drive

In this diagram, the DC motor is represented by its equivalent circuit consisting of inductor, L_a and resistor R_a, in series with the counter electromotive force (emf) E.

The back EMF is proportional to the motor speed:

$$E = K_E.\omega$$

where K_E is the motor voltage constant and ω is the motor speed.

With a constant excitation, the torque developed by the DC motor can be considered to be proportional to the armature current I_a:

$$T_m = K_T I_a$$

where K_T is the motor torque constant.

Thyristor Th1 is triggered by a pulse width modulated (PWM) signal to control the average motor voltage. Theoretical waveforms illustrating the chopper operation are shown in Figure 2-6.

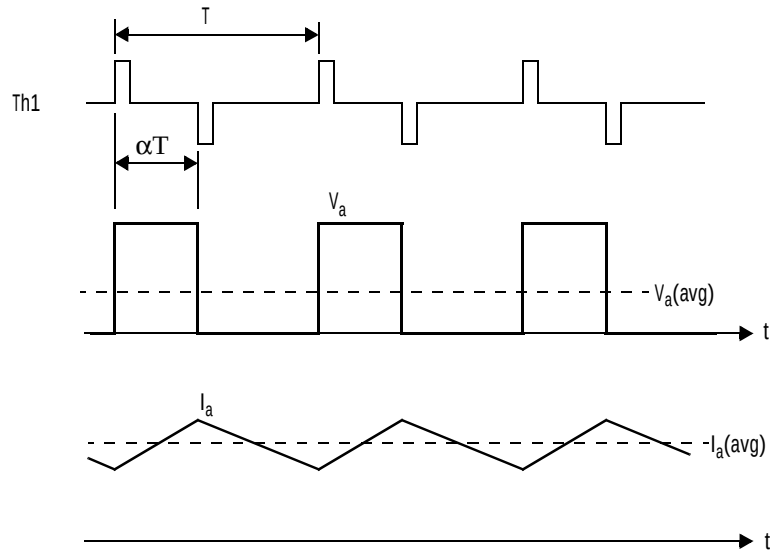


Figure 2-6 Waveforms Illustrating the Chopper Operation

The average armature voltage is a direct function of the chopper duty cycle α

$$V_a(avg) = \alpha.V_{dc}$$

Note that this relation is valid only when the current is continuous. In steady-state, the armature average current is equal to:

$$I_a(avg) = \frac{V_a(avg) - E}{R_a}$$

The peak-to-peak current ripple is

$$\Delta i = \frac{Vdc}{R_a}\frac{(1 - e^{-\alpha r} + e^{-r} - e^{-(1-\alpha)r})}{1 - e^{-r}}$$

where α is the duty cycle and r is the ratio between the chopper period and the electrical time constant at the variable-speed DC motor drive.

$$r = \frac{T}{(L_a / R_a)}$$

In this case study, we consider a variable-speed DC motor drive using a cascade control configuration. A block diagram of this drive is shown in Figure 2-7.

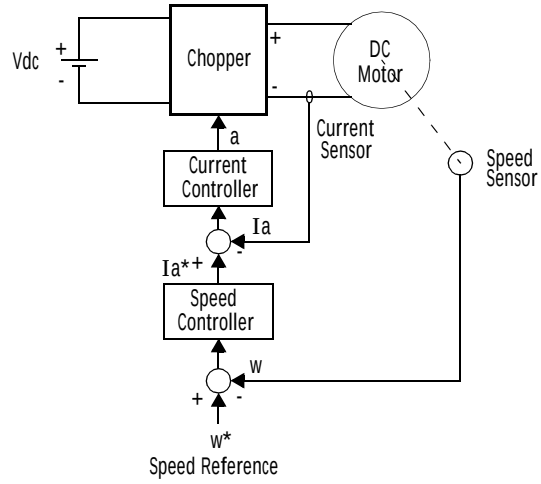


Figure 2-7 Variable-Speed DC Motor Drive

The motor torque is controlled by the armature current I_a, which is regulated by a current control loop. The motor speed is controlled by an external loop that provides the current reference I_a* for the current control loop.

Modeling the DC Drive

Open the `psbdcdrive.mdl` file of the powerlib library by typing `psbdcdrive` in the MATLAB command window. A circuit diagram titled `psbdcdrive` will appear. Before running the example, save this circuit as `case2.mdl` in your working directory so that you can make further modifications without altering the original file.

The drive system diagram is built using electrical blocks contained in the powerlib library combined with Simulink blocks. Voltage and current measurement blocks and controlled sources are used as interfaces between the two block types. The system diagram is shown in Figure 2-8.

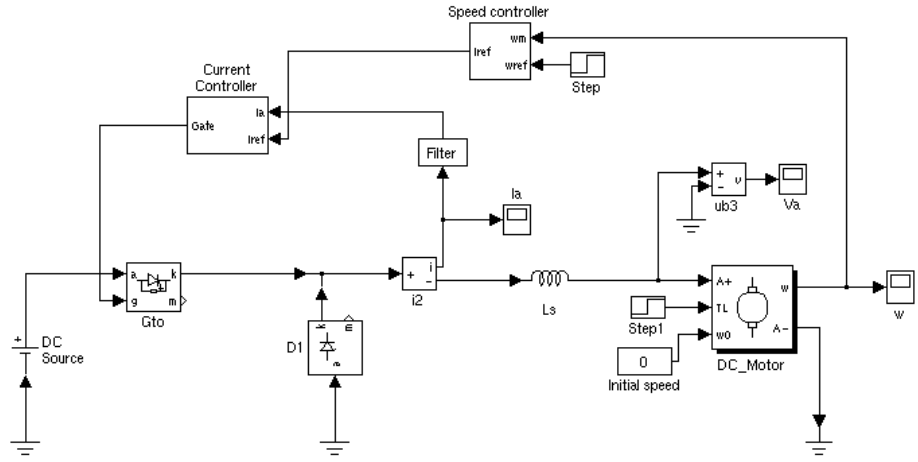
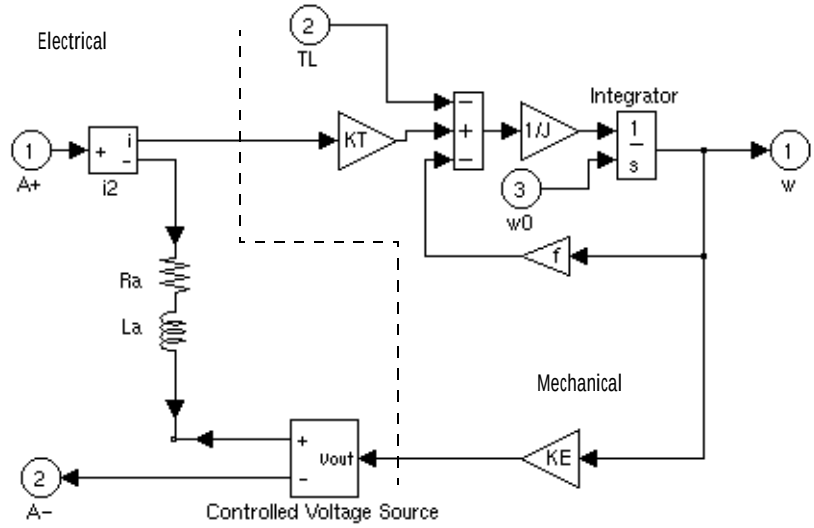


Figure 2-8 DC Motor Drive Using Power System Blockset (psbdcdrive.mdl)

The DC motor is modeled in two separate parts: electrical and mechanical. To view the motor Simulink model, select the DC motor block, then select **LookUnderMask** on the **Edit** menu.



The electrical part is represented by an RL circuit in series with a controlled voltage source, the value of which is $K_E\omega$. The mechanical part is represented by Simulink blocks

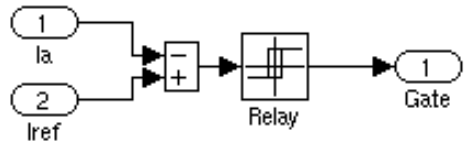
$$T_m = J\frac{d\omega}{dt} + B\omega + T_L$$

The initial motor speed can be set to any desired value by using the ω0 input of the DC Motor block.

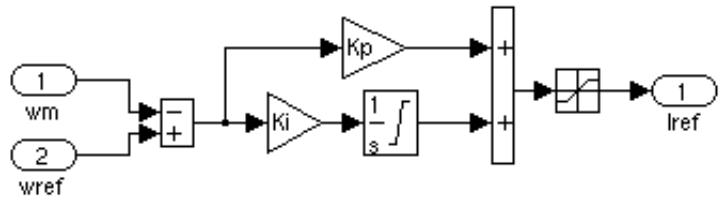
The motor used in this case study is a permanent-magnet DC motor having the following parameters: Ra = 0.5 Ω, La = 10 mH, KE =1.23 V/rad/s, KT = 1.23 N.m/A. A 10mH inductor (Ls) is connected in series with the DC motor to smooth the armature current.

The required trigger signal for the GTO thyristor is generated by a hysteresis current controller, which forces the motor current to follow the reference within +h/2 and -h/2 limits (h is the hysteresis band).

The current controller is also a masked block that contains the following.



The speed control loop uses a proportional-integral controller, which is implemented by Simulink blocks.



Simulation of the DC Drive

Set the simulation parameters in the **Simulation Parameters** menu as follows:

```
Simulation time: Start Time:0, Stop time: 1.5

Solver Type: Variable-step ode15s (stiff/NDF)
Max Step Size: auto
Initial Step Size: auto
Relative Tolerance: 1e-3
Absolute Tolerance: 1e-3
Maximum order: 5
```

Run the simulation by selecting **Start** from the **Simulation** menu in Simulink.

The motor voltage, current waveforms, and motor speed are displayed on three scopes connected to the variables V_a, I_a, and ω.

Once the simulation is completed, you can return to the MATLAB window to examine the results with more details by using the `plot` function.

Drive Starting

In this example, you simulate the starting transient of the DC drive. The inertia of the mechanical load is small in order to bring out the details of the chopper commutation details. The speed reference is stepped from 0 to 150 rad/s at t = 0.0 s and we observe the drive variables, speed, and current.

The transient responses for the starting of the DC motor drive are shown in Figure 2-9.

Note that the final system state vector can be saved by checking the Final State box in the **Workspace I/O/ Save to Workspace** in the **Simulation** parameters window.

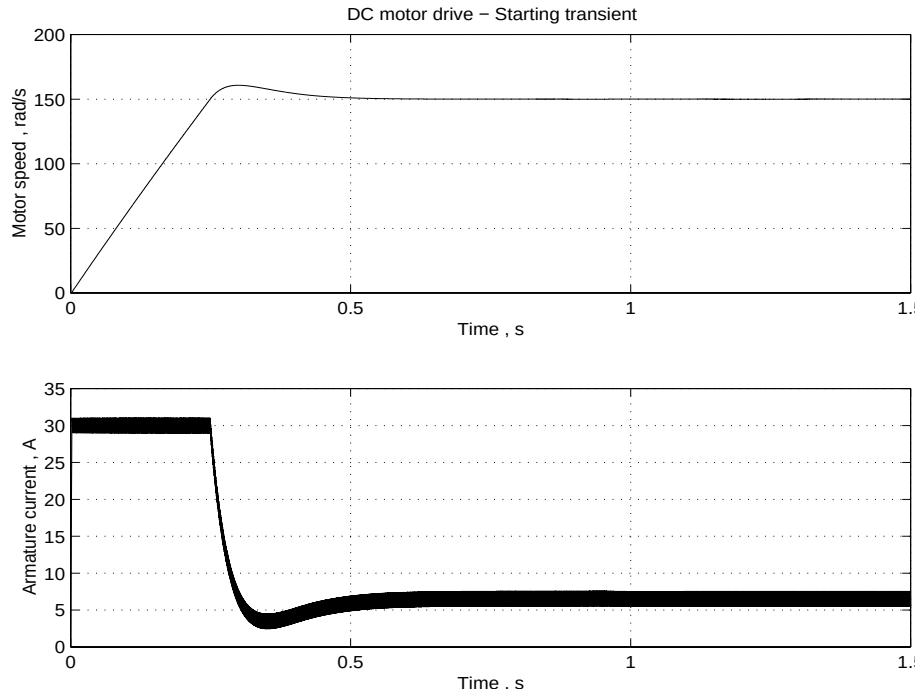


Figure 2-9 Starting of the DC Motor Drive

Steady-State Voltage and Current Waveforms

When the steady-state is attained, you can stop the simulation and plot the current and voltage waveforms using the variables V_a and I_a sent back to the MATLAB workspace by the scopes.

The DC motor current and voltage waveforms obtained at the end of the starting test are shown in Figure 2-10.

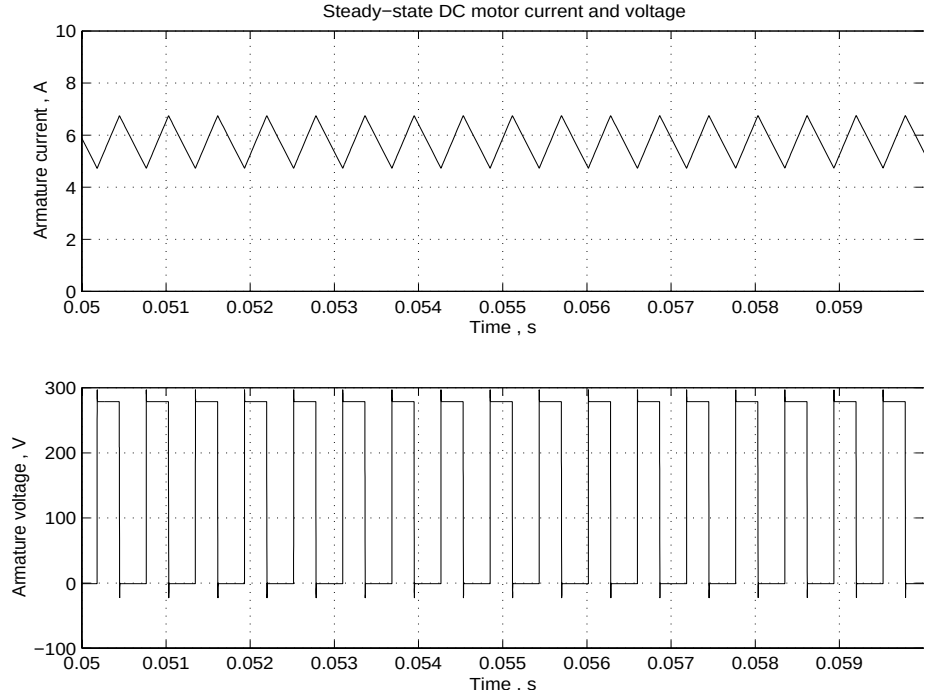


Figure 2-10 Steady-State Motor Current and Voltage Waveforms

Speed Regulation Dynamic Performance

Study the drive dynamic performance (speed regulation performance versus reference and load torque changes) by applying two successive changing operating conditions to the DC drive: a step change in speed reference and a step change in load torque.

Replace the constant wref and TL blocks in the diagram by two Simulink `step` functions with different starting times. The final state vector obtained with the previous simulation can be used as the initial condition so that the simulation will start from steady-state. Select **Workspace I/O/Load initial** in the **Simulation** parameters window and restart the simulation. The obtained response of the DC motor drive to successive changes in speed reference and load torque is shown in Figure 2-11.

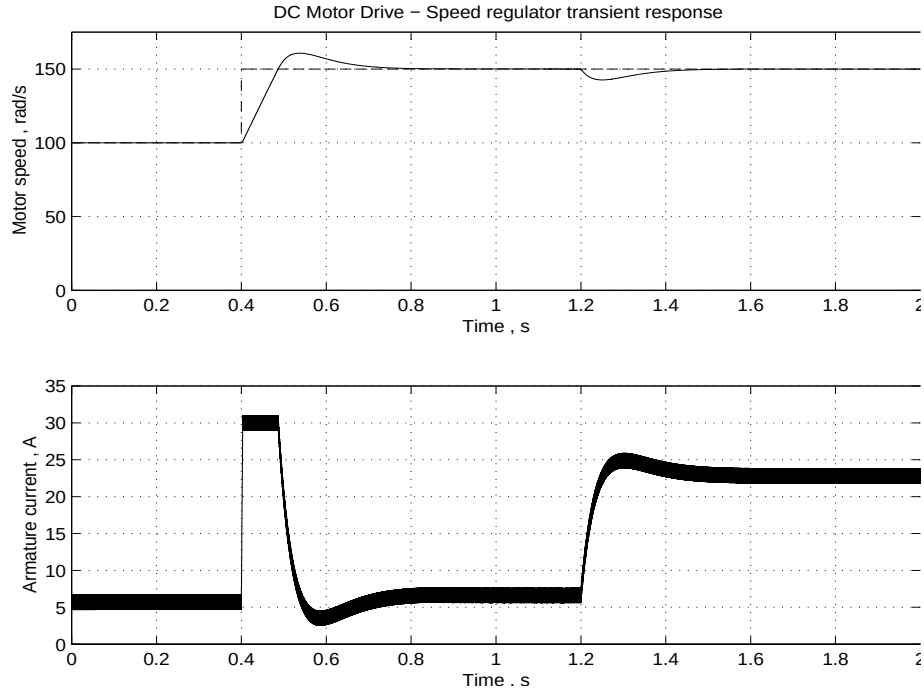


Figure 2-11 Dynamic Transient of the DC Motor Drive

References

[1] Leonhard,W. *Control of Electrical Drives*. Springer-Verlag, Berlin 1985.

Synchronous Machines and Regulators

This case study investigates the application of a multi-input, multi-output nonlinear controller to a system consisting of a hydraulic turbine and a synchronous generator connected to an infinite bus. The complete system is modeled using the Power System Blockset and Simulink blocks.

The objective of this case study is to demonstrate the use of the Synchronous Machine block connected to a complex control system implemented with Simulink blocks. The controller is based on a feedback linearization scheme. Its main goal is to control the rotor angle as well as the terminal voltage, to improve the stability properties, and to obtain good dynamic response. Simulation results will show that the nonlinear controller is able to replace the standard linear controllers and give better performance.

Introduction

Traditionally, stabilization of power systems was ensured by linear regulators such as the automatic voltage regulator (AVR), the speed governor, or the power system stabilizer (PSS). These compensators assume a linearized model of the power system around an operating point.

The demand for improved performance has created the need to operate power systems closer to the limits and therefore well outside the linear domain. Nonlinearities begin to have a significant effect then, especially after important disturbances that lead to a large variation of the operating point.

We propose to design a nonlinear controller that takes into account all the nonlinearities of the model. The objective of the controller is to regulate both the terminal voltage and the internal power angle. The control inputs are the field excitation voltage and the gate opening of the turbine.

Mathematical Model

The model considered is a single machine infinite bus (SMIB) system, as shown in Figure 2-12. The machine is a synchronous generator driven by a hydraulic turbine.

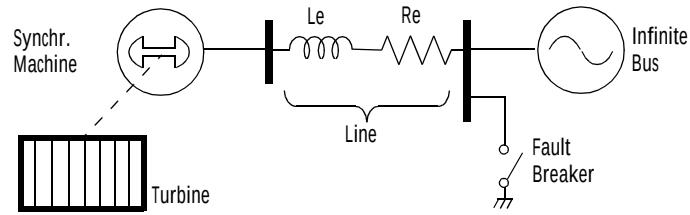


Figure 2-12 Diagram of Case Study

The dynamic equations of the machine that are used to derive the linear feedback controller are for the three-phase Synchronous Machine Hydraulic Turbine and Governor blocks (see Chapter 4, “Block Reference”). Because the synchronous machine is connected to an infinite bus, the dq terminal voltages v_d and v_q are constrained by the load equations. In the Park-transformed coordinates, we can write:

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = R_e \begin{bmatrix} i_d \\ i_q \end{bmatrix} + L_e \frac{d}{d\tau} \begin{bmatrix} i_d \\ i_q \end{bmatrix} - \omega L_e \begin{bmatrix} i_q \\ -i_d \end{bmatrix} + V^{\infty} \begin{bmatrix} \cos(\delta - a) \\ -\sin(\delta - a) \end{bmatrix}$$

This equation can be combined with the complete model of the SMIB system in the nonlinear state-space form $\dot{x} = F(x) + G(x)u$.

When *F(x)* and *G(x)* are given by

$$F(x) = \begin{bmatrix} A_{11}x_1 + A_{12}x_3 + A_{13}x_2x_7 + A_{14}x_4 + A_{15}x_5x_7 + A_{16}\cos(x_6 - a) \\ A_{21}x_1x_7 + A_{22}x_2 + A_{23}x_3x_7 + A_{24}x_4x_7 + A_{25}x_5 + A_{26}\sin(x_6 - a) \\ A_{31}x_1 + A_{32}x_3 + A_{33}x_2x_7 + A_{34}x_4 + A_{35}x_5x_7 + A_{36}\cos(x_6 - a) \\ A_{41}x_1 + A_{42}x_3 + A_{43}x_2x_7 + A_{44}x_4 + A_{45}x_5x_7 + A_{46}\cos(x_6 - a) \\ A_{51}x_1x_7 + A_{52}x_2 + A_{53}x_3x_7 + A_{54}x_4x_7 + A_{55}x_5 + A_{56}\sin(x_6 - a) \\ (x_7 - 1)\omega_r \\ A_{71}x_1x_2 + A_{72}x_2x_3 + A_{73}x_2x_4 + A_{74}x_1x_5 + A_{75}x_7 + A_{76}\dfrac{x_8^3}{x_7x_9^2} \\ A_{81} - A_{82}\dfrac{x_8^2}{x_9^2} \\ A_{91}x_9 \end{bmatrix}$$

$$G(x) = \begin{bmatrix} g_{11} & 0 & g_{31} & g_{41} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g_{92} \end{bmatrix}^T$$

You can find the explicit expressions of the coefficients A and g in Chapter 4, “Block Reference” and are omitted here for simplicity. The other terms of the state-space equation are defined as follows:

- $x = \begin{bmatrix} i_d & i_q & i_{fd} & i_{kd} & i_{kq} & \delta & \omega & q & G \end{bmatrix}^T$ is the vector of state variables
- $u = \begin{bmatrix} v_{fd} & u_G \end{bmatrix}^T$ is the vector of control inputs

The currents i_d, i_q and voltages v_d, v_q are the projection of the actual line currents and terminal voltages on the direct and quadrature axes (d-q frame). i_{fd} and v_{fd} represent the field current and voltage. i_{kq} and i_{kd}, the damper windings currents, and ω the angular speed of the machine. δ is the electrical

angle measured from a synchronously rotating frame. G and q are respectively the opening of the gate and the flow rate of the turbine. Finally, u_G is the voltage applied to the gate servo-motor.

Feedback Linearization Design

The input-output feedback linearization technique consists of the exact cancellation of the nonlinearities of the system in order to obtain a linear relationship between inputs and outputs in closed-loop. The nonlinear control law is deduced by successively differentiating each of the outputs until at least one input appears. Consider the first output as the terminal voltage V_t:

$$y_1 = V_t = \sqrt{V_d^2 + V_q^2}$$

The terminal voltage is a complicated function of the state variables in which the control input V_{fd} appears explicitly with a multiplying factor of a very small order of magnitude. Disregard this direct dependence between V_t and V_{fd} and we compute the time derivate of output y_1:

$$\frac{dy_1}{dt} = \alpha_1(x) + \beta_{11}(x)u_1 + \beta_{12}(x)u_2, \quad \text{where}$$

$$\alpha_1(x) = \frac{\partial V_t}{\partial x} \cdot F(x)$$

$$= \frac{1}{2V_t}\left(2V_d\frac{\partial V_d}{\partial x} + 2V_q\frac{\partial V_q}{\partial x}\right)F(x)$$

$$\beta_{11}(x) = \frac{\partial V_t}{\partial x} \cdot G_1(x)$$

$$= \frac{1}{2V_t}\left(2V_d\frac{\partial V_d}{\partial x} + 2V_q\frac{\partial V_q}{\partial x}\right)G_1(x)$$

$$\beta_{12}(x) = \frac{\partial V_t}{\partial x} \cdot G_2(x)$$

$$= 0$$

The second output, y_2, is the angle δ, which has to be differentiated three times before the inputs appear, which yields:

$$\begin{aligned}\frac{d^3 y_2}{dt^3} &= \omega_r \frac{\partial F}{\partial x} \cdot F(x) + \omega_r \frac{\partial F}{\partial x} G_1(x) u_1 + \omega_r \frac{\partial F}{\partial x} G_2(x) u_2 \\ &= \alpha_2(x) + \beta_{21}(x) u_1 + \beta_{22}(x) u_2\end{aligned}$$

If you can combine the equations of the outputs, you obtain the input-output nonlinear system:

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(3)} \end{bmatrix} = \begin{bmatrix} \alpha_1(x) \\ \alpha_2(x) \end{bmatrix} + \begin{bmatrix} \beta_{11}(x) & 0 \\ \beta_{21}(x) & \beta_{22}(x) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

and you can easily deduce the nonlinear control law

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \beta_{11}(x) & 0 \\ \beta_{21}(x) & \beta_{22}(x) \end{bmatrix}^{-1} \left(- \begin{bmatrix} \alpha_1(x) \\ \alpha_2(x) \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \right)$$

that will yield, in closed-loop, the exactly linearized input-output system.

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(3)} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Once the system has been linearized, you can apply any linear control design to regulate the outputs. Here, we have chosen the pole placement method and propose the following linear control law:

$$\begin{aligned} v_1 &= k_{11}(V_t - V_{tref}) \\ v_2 &= k_{21}(\delta - \delta_{ref}) + k_{22}(\dot{\delta} - \dot{\delta}_{ref}) + k_{23}(\ddot{\delta} - \ddot{\delta}_{ref}) + k_{24}(x_9 - x_{9ref}) \end{aligned}$$

The last term in the equation of v_2 is introduced in order to stabilize the internal dynamics of the system. These dynamics result because the original nonlinear system is a ninth order system while the linearized system is a fourth order system. We call this *partial linearization* and we must ensure that the remaining dynamics are asymptotically stable. You can find a complete treatment of this question in reference [1].

Simulation Results

You can test performance of the nonlinear controller on the nonlinear turbine-generator system. The controller and turbine are simulated using Simulink blocks while the generator is represented by the Synchronous Machine block from the powerlib library. A three-phase short-circuit has been simulated on the load busbar, and the fault has been cleared after 100 ms. The performance of the nonlinear controller is analyzed.

The case study is included in the `psbregulator.mdl` file. The system is illustrated in Figure 2-13. Before running the simulation, make sure that the simulation parameters are set as follows:

- Solver: `ode15s;` Maximum order: `5`
- Stop time: `1.0`
- Max step size: `auto`; Initial step size: `auto`
- Relative tolerance: `1e-3;` Absolute tolerance: `1e-6`
- Workspace I/O: Load initial states: `psbregulinit`

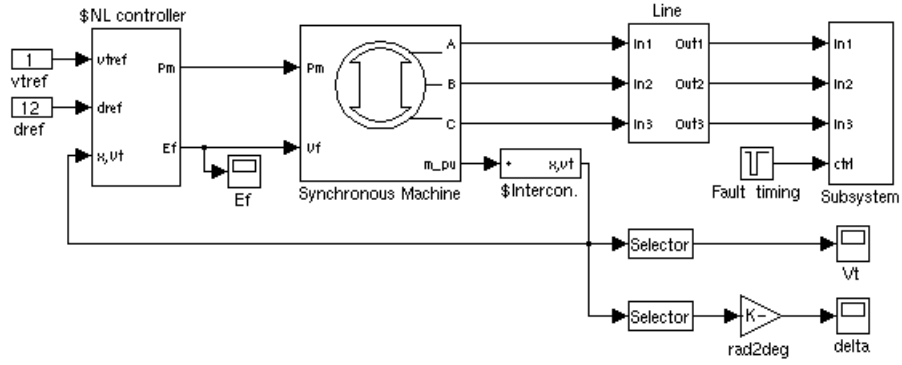


Figure 2-13 Simulink Diagram of Case Study (psbregulator)

Because of nonlinearities in this system, computation of initial conditions was not carried out. Instead, a long simulation (10 s) was executed and the final states saved in file `psbreguldata.mat`. These final states are used as the initial states in this case study. The simulation consequently starts in steady-state. At t=0.1s, the fault is suddenly applied and removed after 100 ms (6 cycles). The post-fault transient is then observed.

The nonlinear controller calls a MATLAB initialization functions to compute the gains before the simulation. Although this process has been automated to take into account the parameters in the dialog boxes of the various blocks, it is not recommended that any value in any block be changed.

If you decide to change some values, you must run a long simulation and the final states must be in a file called `psbregulinit.mat`. Figure 2-14 shows the response of the generator's terminal voltage, load angle, and the control effort of the regulator. You can observe how the stabilization of V_t is obtained in less than 0.25 seconds with this controller. The load angle takes longer to stabilize, because the time constant of the mechanical part of the system is much larger than the electrical time constants. If you want to compare results with classical regulators, replace the nonlinear controller with the same excitation system and Hydraulic Turbine and governor block used in the Synchronous Machine demo (`psbturbine.mdl`). You will notice that the system takes longer to stabilize than in this case study.

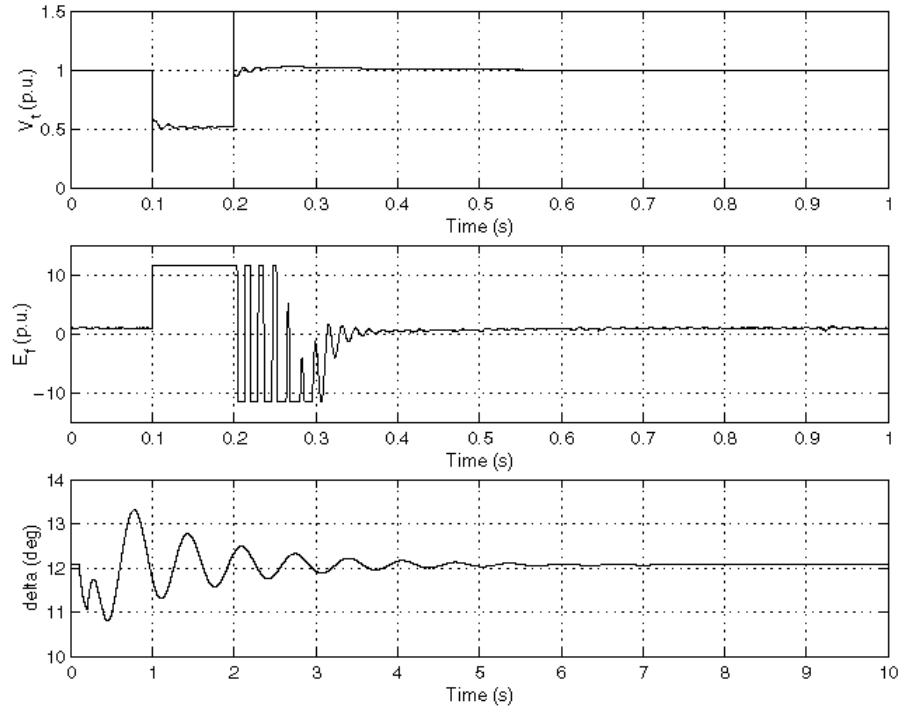


Figure 2-14 Simulation Results Obtained With Case Study

References

[1] Akhrif O., F.A. Okou, L.A. Dessaint, R. Champagne, “Multi-input Multi-output Feedback Linearization of a Synchronous Generator.” Canadian Conference on Electrical and Computer Engineering, 1996.

Variable-Frequency Induction Motor Drive

This case study presents a variable-frequency AC motor drive in which a pulse-width-modulated (PWM) inverter is used as a variable-voltage variable-frequency source to drive an induction motor in variable-speed operation.

The drive, including the motor, the power converter, and the speed control system, is modeled by using the Power System Blockset and Simulink blocks. The drive operation is studied for these different operating conditions: starting, steady-state, and transients.

The objective of this example is to demonstrate the use of Electrical Machine and Power Electronics blocks in combination with Simulink blocks in the simulation of a complex electromechanical system operating at high frequency. The electrical part of the AC motor drive, including the PWM inverter, is built using six Mosfet switches. The induction motor is represented by the Asynchronous Machine block, which models both electrical and mechanical dynamics. The control system, including current and speed regulators, is built using Simulink blocks. The interface between electrical and control systems is ensured by Measurement blocks.

Description of the Induction Motor Drive

The induction motor requires a variable-frequency three-phase source for variable-speed operation. This source can be realized by using a power converter system consisting of a rectifier connected to an inverter through a DC link.

Figure 2-15 shows a block diagram of the power circuit of a typical variable-frequency induction motor drive.

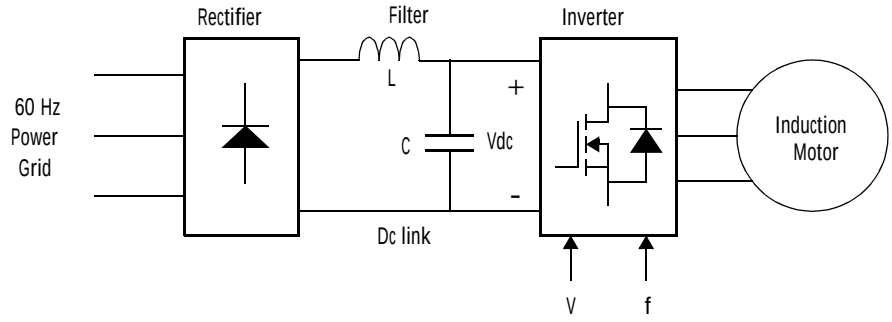


Figure 2-15 Variable-Frequency Induction Motor Drive

The power grid AC voltage is converted into a fixed DC voltage by the rectifier. The harmonics are filtered out by an LC filter to provide a smooth DC voltage which is then applied to the inverter input.

The inverter consists of six power switches that can be Mosfets, IGBTs, or GTOs, depending on the drive power capacity. Figure 2-16 shows a simplified diagram of a Mosfet inverter.

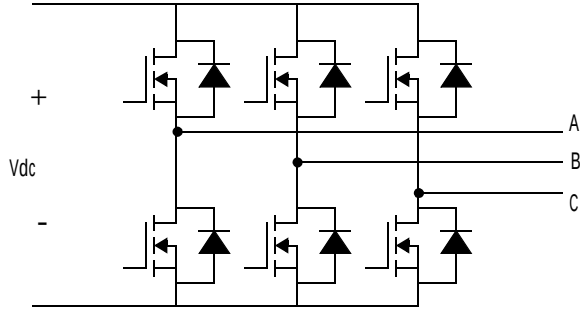


Figure 2-16 Mosfet Inverter

The inverter converts the DC link voltage into an adjustable three-phase AC voltage. Different control schemes can be used to control the inverter output voltage and frequency. One of the most utilized schemes is PWM in which three-phase variable sinusoidal voltage waveforms are obtained by modulating the on and off times of the power switches.

In industrial drive applications, the PWM inverter operates as a three-phase variable-frequency, variable-voltage source with fundamental frequency varying from zero to three times the motor nominal frequency.

In some control schemes, where a three-phase, variable-frequency current source is required, current control loops are added to force the motor currents to follow an input reference (usually sinusoidal).

The inverter-fed induction motor drive can be controlled by using various schemes depending on the application, desired performance, and controller design complexity. The most utilized schemes are:

- Stator V/Hz control
- Stator currents and open loop flux control
- Vector control (field-oriented control)

A Variable-Speed Induction Motor Drive

In this case study, we consider a variable-speed induction motor drive using impressed sinusoidal stator currents and open loop flux control. The motor torque is controlled by the *rotor frequency* (which is also called slip frequency). A block diagram of this drive is shown in Figure 2-17.

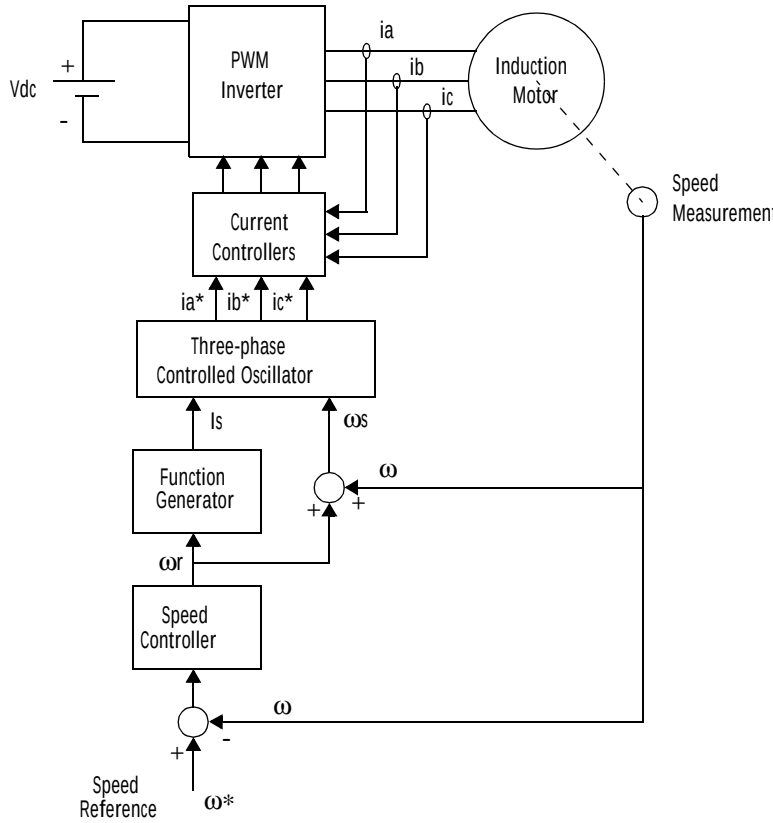


Figure 2-17 Variable-Speed Induction Motor Drive

The induction motor is fed by a current-controlled PWM inverter, which operates as a three-phase sinusoidal current source.

The sinusoidal current references ia*, ib*, and ic* for the current controllers; are generated by a three-phase oscillator. The oscillator amplitude and frequency are separately controlled by the signals I_s and ω_s representing respectively the stator current amplitude and frequency. The motor torque is controlled by the rotor frequency ω_r (slip frequency). The motor speed ω is compared to the reference ω^* and the error is processed by the speed controller to produce a torque command, which is the rotor frequency ω_r. The oscillator

frequency ω_s, which represents the stator current frequency, is simply the sum of the motor speed ω and the slip frequency ω_r.

The air gap flux is controlled in an open loop by a function generator that relates the current amplitude I_s to the rotor frequency ω_r according to the following function

$$I_s = I_m \sqrt{\frac{R_r^2 + (\omega_r L_{rr})^2}{R_r^2 + (\omega_r L_{lr})^2}}$$

where I_m is the nominal magnetizing current, R_r is the rotor resistance, L_{lr} is the rotor leakage inductance, and L_{rr} is the rotor total inductance.

The role of the speed controller is to keep the motor speed equal to the speed reference input in steady-state and to provide a good dynamic during transients. The controller can be a proportional-integral (PI) type.

Modeling the Induction Motor Drive

Open the `psbfrequency.mdl` file of the powerlib library by typing `psbfrequency` in the MATLAB command window. A circuit diagram titled `psbfrequency` will appear. Before running the example, save this circuit as `case4.mdl` in your working directory so that you can make further modifications without altering the original file.

Figure 2-18 shows the `psbfrequency` diagram in which blocks from the Power System Blockset and Simulink are used to model the induction drive.

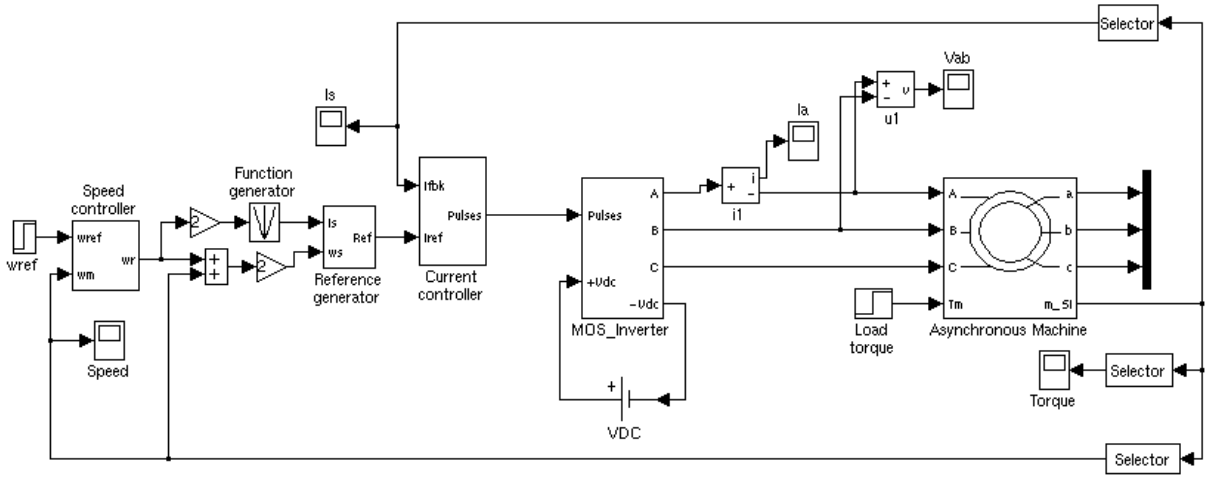


Figure 2-18 Variable Speed Induction Motor Drive (psbfrequency.mdl)

The induction motor is modeled by an asynchronous machine block. The motor used in this case study is a 3HP, 220 V, 4 pole, 60 Hz motor having the following parameters: Rs = 0.435 Ω, Lls = 2mH, Lm = 69.1 mH, Rr = 0.816 Ω, Llr = 2 mH.

The current-controlled PWM inverter block diagram is shown in Figure 2-19. The Mosfet inverter is built by using six Mosfet blocks. The DC link input voltage is represented by a 360 V DC voltage source.

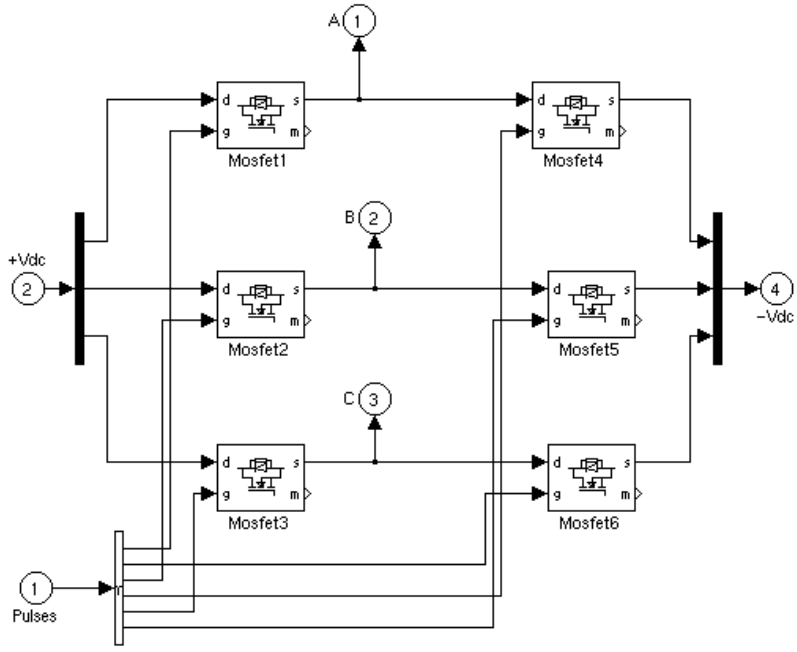


Figure 2-19 Mosfet Inverter

The current control section, (see Figure 2-20), located under the masked Current controller block, consists of three hysteresis controllers, is built with Simulink blocks. The motor currents are sensed by Current Measurement blocks.

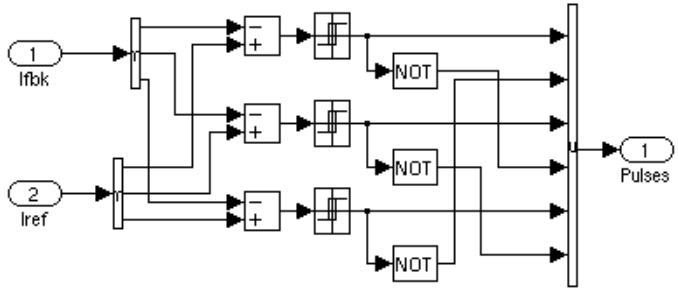


Figure 2-20 Current Control Section

A block diagram of the three-phase controlled oscillator, named Reference Generator, is shown in Figure 2-21. This oscillator produces the references ia*, ib*, and ic* for the current controllers.

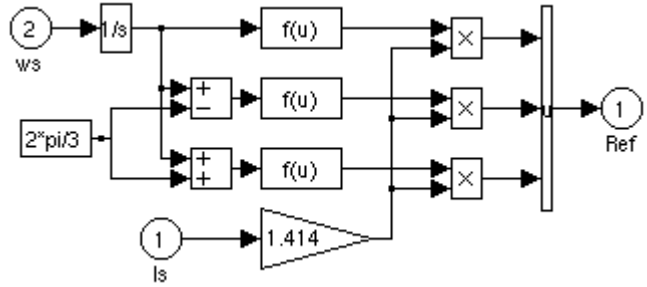


Figure 2-21 Three-Phase Controlled Oscillator

The speed controller (see Figure 2-22), located under the Speed controller block, is a proportional-integral (PI) type and is implemented using Simulink blocks.

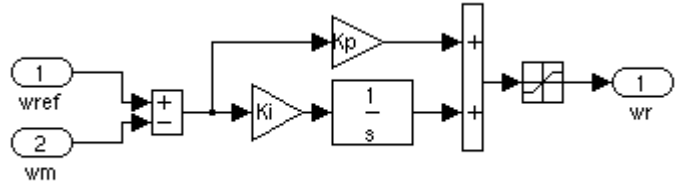


Figure 2-22 The Speed Controller

Simulating the Induction Motor Drive

The simulation parameters in the **Simulation Parameters** menu should be set as follows:

```
Simulation time: Start Time:0, Stop time: 1.5

Solver option: Type: Variable-step ode15s (stiff/NDF)
Max Step Size: auto
Initial Step Size: auto
Relative Tolerance: 1e-3
Absolute Tolerance: 1e-3
Maximum order: 5
```

Run the simulation by selecting **Start** from the **Simulation** menu in Simulink.

The motor voltage and current waveforms as well as the motor speed are displayed on three Scopes connected to the variables V_{ab}, I_s and ω.

Once the simulation is complete, return to the MATLAB window to examine the results with more details by using the `plot` command.

Drive Starting

You can start the drive by specifying `0` initial conditions for all state variables in **Powergui** interface. In this example, the speed reference is stepped from 0 to 100 rad/s at t = 0 and we observe the drive variables, speed, torque, and current.

The transient responses for the starting of the induction motor drive are shown in Figure 2-23.

Note that the final system state vector can be saved by selecting **Workspace I/O/Save final state** in the **Simulation** parameters window.

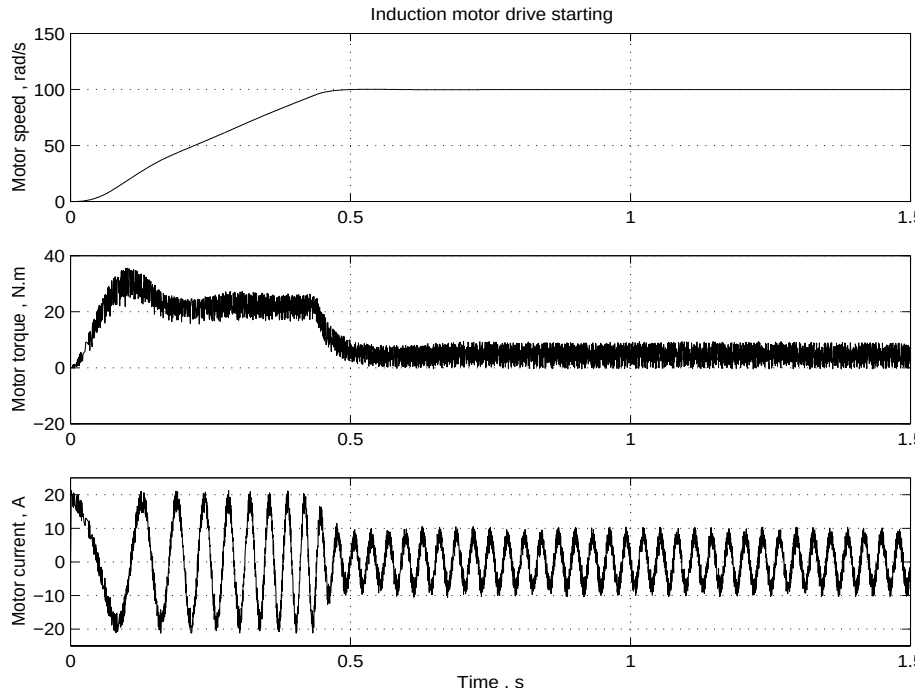


Figure 2-23 Starting of the Induction Motor Drive

Steady-State Voltage and Current Waveforms

When the steady-state is attained, you can stop the simulation and plot the voltage and current waveforms using the variables V_{ab} and I_a sent back in the MATLAB workspace by the scopes.

Figure 2-24 shows the motor current and voltage waveforms obtained at the end of the starting test.

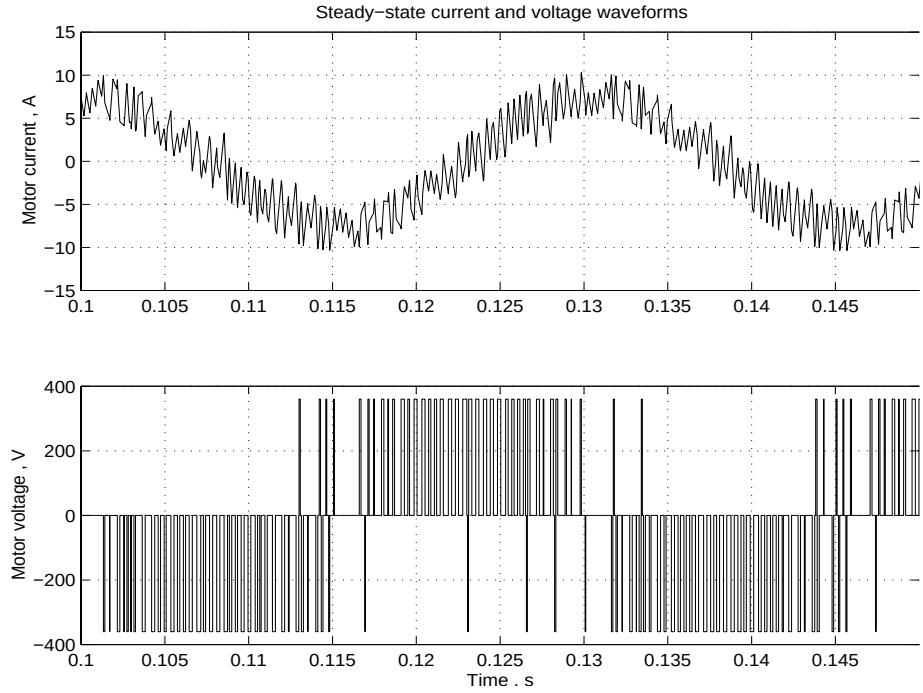


Figure 2-24 Steady-State Motor Current and Voltage Waveforms

Speed Regulation Dynamic Performance

We can study the drive dynamic performance (speed regulation performance versus reference and load torque changes) by applying two changing operating conditions to the drive: a step change in speed reference, and a step change in load torque.

Replace the constant w_{ref} and TL blocks in the diagram by two Simulink `step` functions with different starting times. The final state vector obtained with the previous simulation can be used as initial condition so that the simulation will start from steady-state. Select **Workspace I/O/Load initial states** in the **Simulation** parameters window and restart the simulation.

The obtained response of the induction motor drive to successive changes in speed reference and load torque is shown in Figure 2-25.

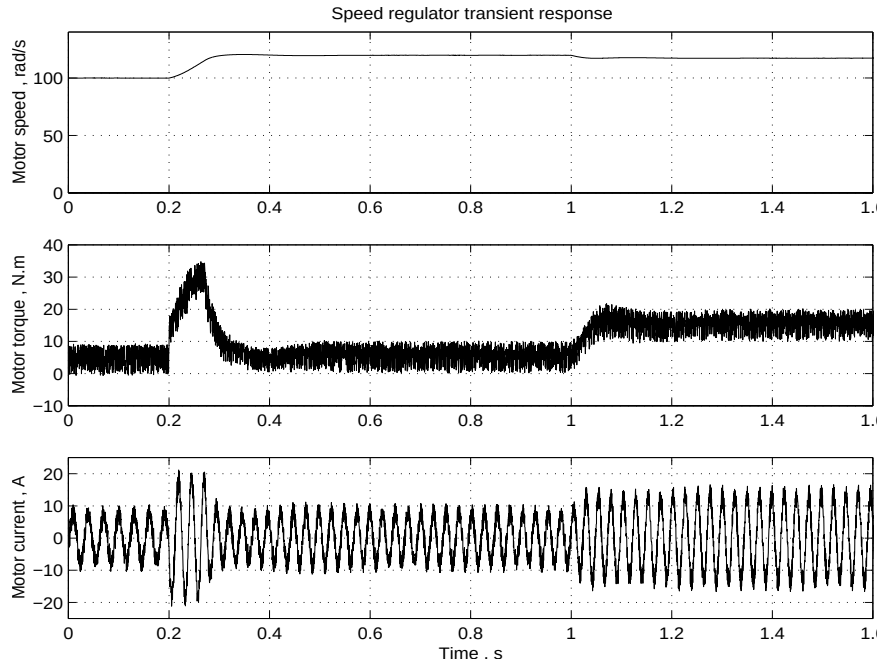


Figure 2-25 Dynamic Performance of the Induction Motor Drive

References

[1] Leonhard, W., *Control of Electrical Drives*. Springer-Verlag, Berlin 1985.

[2] Murphy, J. M. D., and F. G. Turnbull, *Power Electronic Control of AC Motors*. Pergamon Press, Oxford 1985.

[3] Bose, B. K., *Power Electronics and AC Drives*. Prentice-Hall, Englewood Cliffs 1986.

HVDC Transmission System

The final example in this section develops the concept of modeling a high voltage direct current (HVDC) transmission link [1]. Perturbations are applied in order to examine the system performance [2].

The objectives of this example are to demonstrate the use of the Six-Pulse Thyristor Bridge block and the Synchronized Six-Pulse Generator block in combination with Simulink blocks in the simulation of a simplified HVDC system. The electrical part representing the AC network is built using three-phase blocks. The control system, including a current regulator, is built using Simulink blocks. The interface between the electrical system and the control system is ensured by Measurement blocks.

Description of the Transmission System

A basic two-terminal monopolar HVDC system with earth return is shown in Figure 2-26.

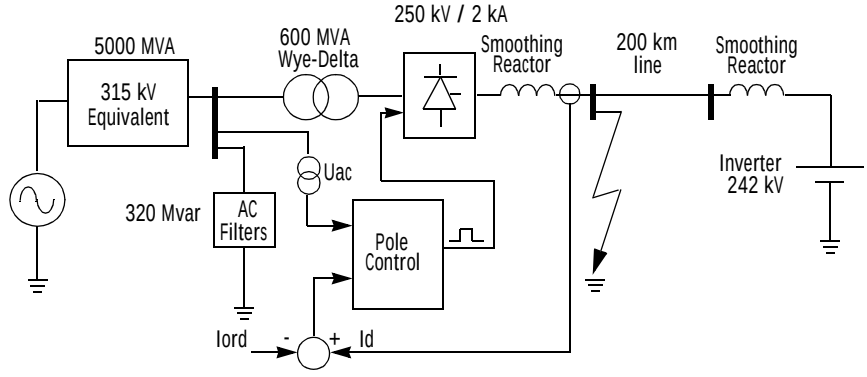


Figure 2-26 HVDC System

The 500 MW DC system transfers energy in the form of high-voltage direct current, 2.5 kA and 250 kV, to the inverter AC bus through a 200 km DC transmission line. The rectifier substation consists of a converter bridge with a 0.5H smoothing reactor, 320 Mvar AC filters, and a pole control system. The AC system has a short-circuit level of 5000 MVA, which is 10 times larger than the transmitted DC power, hence it is considered a strong AC system.

For the case study, we make the following simplifications:

- A basic six-pulse thyristor bridge is used at the rectifier.
- DC filters and a DC damping circuit are disregarded.
- The inverter terminal is replaced by a constant back EMF in series with a smoothing reactor.
- The rectifier controller is a current regulator and a pulse generator that uses phase control to determine the firing instants of the thyristors.

From the AC point of view, an HVDC converter acts as a source of harmonic current. From the DC point of view, it is a source of harmonic voltage. The order n of these characteristic harmonics is related to the pulse number p of the converter configuration: $n = kp \pm 1$ for the AC current, and $n = kp$ for the direct voltage, k being any integer. In the example, p = 6, so that injected harmonics on the AC side are 5, 7, 11, 13,. . . and on the DC side are 6, 12, 18,...

Open the Simulink window `psbhvdc.mdl` and save it in your directory as `case5` to allow further modifications to the original system. Compare the circuit modeled in the Power System Blockset with the schematic diagram of Figure 2-26.

Modeling the Basic HVDC System

The model shown in Figure 2-27, will be used to illustrate the response of the system to a step change command in current and to a DC line to ground fault followed by a load rejection.

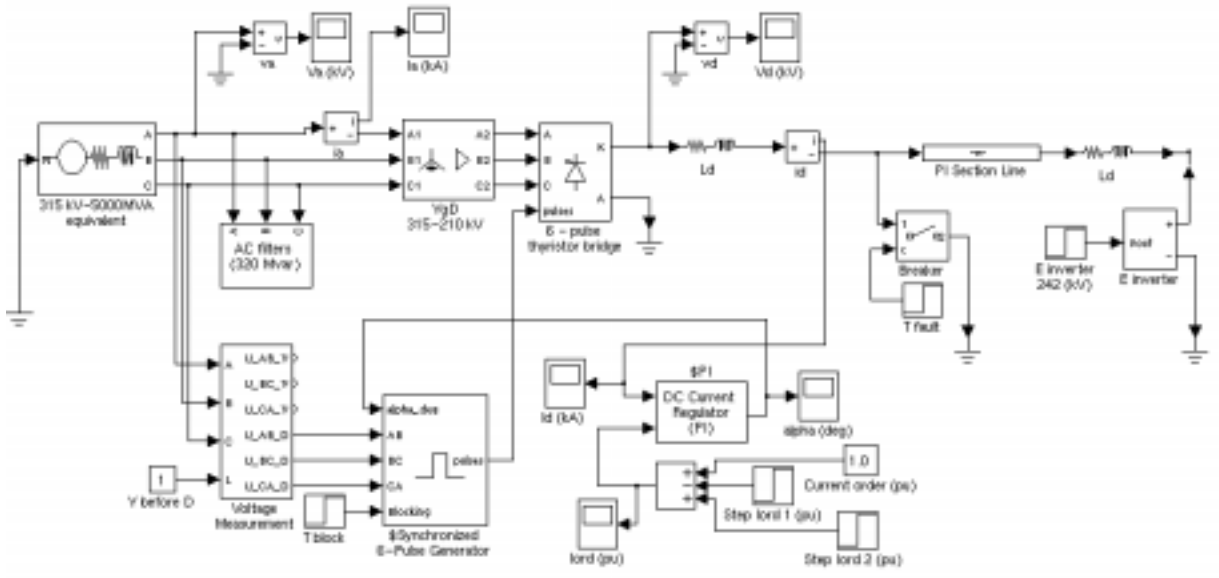


Figure 2-27 HVDC System Model

Observe that the six-pulse thyristor bridge as well as the converter transformer and the equivalent AC system are modeled using the three-phase models found in the powerlib_extras library.

The little damped 5000 MVA AC network is represented by a simple series RL circuit. The AC filters are shunt-connected RLC branches that present a low impedance path to ground for harmonics. They also appear as large capacitors at fundamental frequency, thus providing reactive power compensation for the rectifier consumption due to the firing delay α. The converter reactive power demand is approximately 60 percent of the power transmitted at full load. Look under the AC filters subsystem mask to see the high Q (100) tuned filters at the fifth and seventh harmonic and the low Q (2), or damped filter, used to eliminate the higher order harmonics, e.g. eleventh harmonic and higher. Extra reactive power is also provided by capacitor banks.

Figure 2-28 (a) demonstrates the magnitude, seen from the busbar where the filter is connected, of the combined filter and AC network impedance as a function of frequency. The low principal natural frequency, coinciding with the parallel resonance at 206 Hz, is a determining factor in the development of overvoltages and interaction with the DC system. Figure 2-28 (b) illustrates the magnitude of the impedance as a function of frequency of the DC system, as seen from the rectifier. It is composed by the DC line, represented by four pi sections and the two smoothing reactors.

Note the series resonance at 183 Hz, which corresponds to the main mode likely to be excited, on the DC side, under large disturbances.

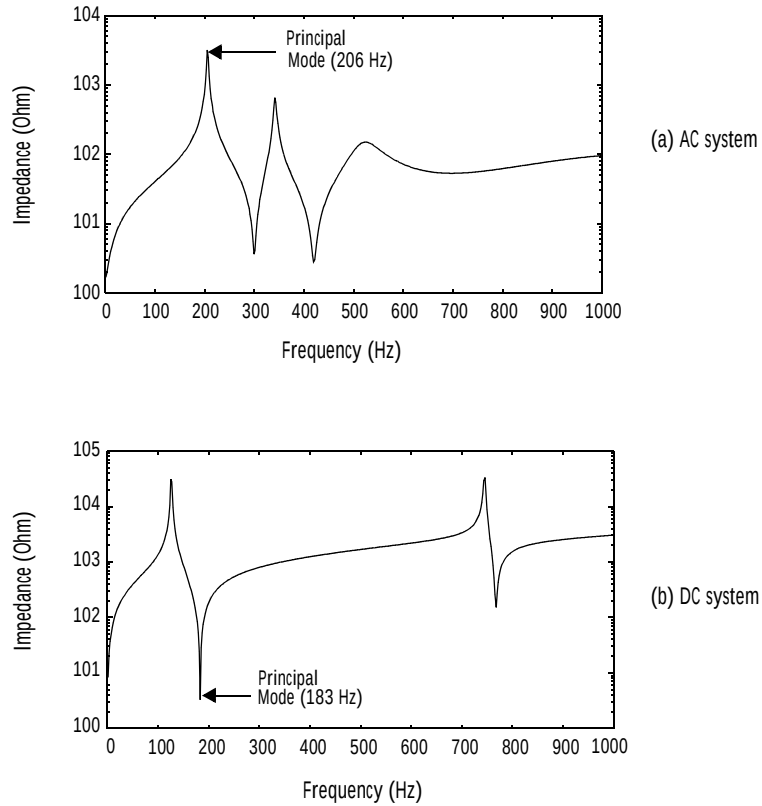


Figure 2-28 Impedance vs. Frequency

The 600 MVA three-phase converter transformer is represented by a linear model connected in Y/delta. The winding impedance of 0.12 pu is concentrated on the thyristor side.

The six-pulse thyristor bridge uses six of the simple thyristor models connected in a three-phase bridge configuration. Snubbers are connected in parallel as damping circuits for each valve. They limit the rate of rise and peak value of inverse voltage across the thyristor valves. Look under the mask to see the details.

To simulate the back EMF of the inverter terminal, use the Controlled Voltage Source block. In conjunction with a Step Source block, it is possible to vary the inverter voltage dynamically during the initialization stage of the simulation.

The pole control is composed of a current regulator, the voltage measurement subsystems, and the Synchronized Pulse Generator block in the powerlib_extras library. The details of the current regulator are shown in Figure 2-29.

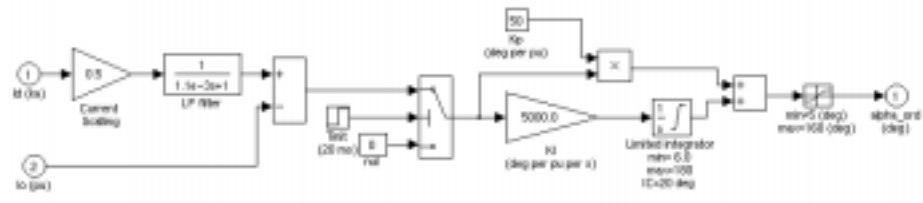


Figure 2-29 Current Regulator

The direct current measurement is filtered by a low-pass filter to remove the high frequency components. The current regulator has a proportional and integral characteristic (PI). It should have a high enough gain for low frequencies (≤10 Hz) to maintain the current response (Id) equal to the current order (Io) as long as α is within the minimum and maximum limits (5° and 160°). Also the current control system should be stable and fast.

Note that the current command is given in pu and corresponds to the rated current. The current regulator output is the delay angle command (`alpha_ord`). Also, the measured current in kA, is scaled to pu before it is used in the regulator. The regulator gains, Kp and KI, are adjusted during small perturbations in the current order (illustrated in Figure 2-30).

For the moment, disregard the initialization period of the simulation to see how the DC current and the firing angle respond to the current command changes of 0.2 pu, at t=0.4 s and t = 0.7 s. The regulator integrator gain was 5000°/pu/s, and the proportional gain was 50°/pu

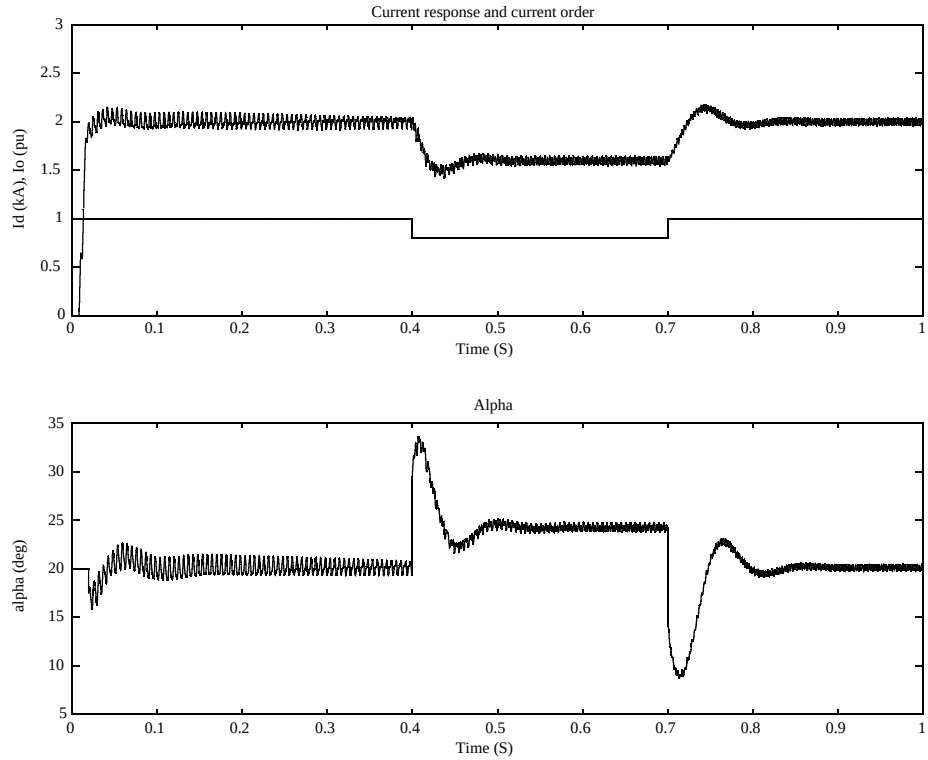


Figure 2-30 Current Order Step Response

The firing command pulse generator is synchronized to the fundamental frequency of the AC source. At the zero-crossings of the commutating voltages (AB, BC, CA), a ramp is reset. The synchronizing voltages are measured at the AC side of the converter transformer because the waveforms are less distorted.

A firing pulse is generated whenever the ramp value becomes equal to the desired delay angle provided by the current regulator. Pulse generation is enabled only after half a cycle. In steady-state, with perfect conditions the pulses occur at intervals of 60°. To improve the synchronizing waveforms, a low

Q bandpass filter tuned to 60 Hz is used to filter the harmonics introduced by the thyristor commutation (see the voltage measurement subsystem).

Note that it is important to specify through the Y_before_D input the type of delta connection used, i.e., leading or lagging. Synchronizing voltages for a Y-Y transformer connection are available if needed.

Steady-State and Model Start-Up

To obtain the desired steady-state values and to reach them promptly at the beginning of the simulation, some calculations and artifices are needed.

To derive the steady-state equations, make the following approximations:

- The AC system is perfectly balanced with sinusoidal voltages. The AC system and the transformer losses are neglected, and the thyristors are treated as ideal switches.
- The direct current is constant and ripple-free.

The following expression relates the mean direct voltage Vd to the direct current Id and other parameters

$$Vd = (Vdio \times \cos(\alpha)) - (Rc \times Id)$$

where Vdio is the ideal no-load direct voltage for a six pulse bridge:

$$Vdio = ((3\sqrt{2})/\pi) \times Vc$$

Vc is the phase-to-phase rms commutating voltage that is dependent on the AC system voltage and the transformer ratio.

Rc is the equivalent commutating resistance:

$$Rc = (3/\pi) \times Xc$$

Xc is the commutating reactance or transformer reactance referred to the valve side.

The following rectifier parameters were used in the simulation:

Vd = 250 kV; Id = 2 kA; $\alpha = 20 \times .1°$; Xc = 8.82 Ω; Vc = 211.1 kV. Note that Vc is slightly larger than the rated value of 210 kV.

By substituting in the first relation, we verify that the calculated rectifier means direct voltage.

$$Vd = (285.1\ kV \times \cos(20.1°)) - (8.82 \times 2kA) = 250.1\ kV$$

is practically the simulation value of 250 kV. The inverter direct voltage can be found from

$$Einv = Vd - (Rdc \times Id) = 242\ kV$$

where Rdc is the resistance of the line and smoothing reactances (4 Ω). Finally, another expression relates the mean direct voltage Vd to the commutation angle also called the overlap angle μ.

$$Vd = \frac{Vdio \times (\cos(\alpha) + \cos(\alpha + \mu))}{2}$$

Hence, μ can be expressed as

$$\mu = \text{acos}[(2 \times Vd / Vdio) - \cos(\alpha)] - \alpha$$

and by replacing the simulation values, we obtain

$$\mu = \text{acos}[(2 \times 250 / 285.1) - \cos(20.1°)] - 20.1° = 15.3°$$

The rectifier steady-state values are shown in Figure 2-31. Observe the commutation time interval corresponding to the angle μ of 14°. Notice the presence of the183 Hz oscillation in the α waveform related to the DC side mode, which is still not damped after one second of simulation.

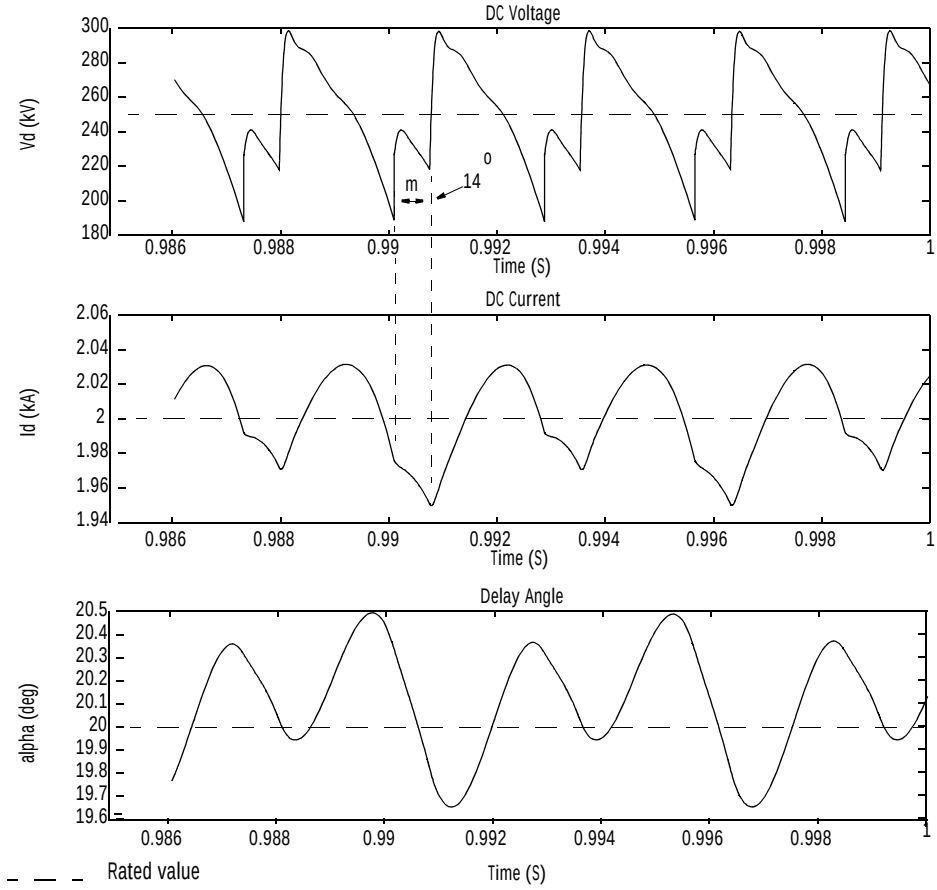


Figure 2-31 Steady-State

Before you can simulate the DC line fault, you must bring the DC system to steady-state operation. For the case study an artificial but fast initialization method completely different from the one used for the actual operation of a HVDC system. At the start, the required DC voltage is imposed by fixing α, but with no back EMF. This allows a fast increase of the DC current. Full inverter voltage is not applied until the DC current comes close to its rated value (2 kA).

Figure 2-29 shows how the current regulator output is forced to 20° during the first 20 ms of simulation.

Check that the simulation parameters are set as follows:

```
Solver: ode15s; Maximum order: 5

Stop Time= 0.250
Min Step Size= auto
Max Step Size= auto
Relative tolerance= 1e-3; Absolute tolerance:1e-3
Returned time variable: [t]
```

Then, start the simulation.

Observe in Figure 2-32 that the DC current starts to build up at 8.33 ms. This is the time at which the gate pulses are enabled in the pulse generator. The rate of increase of the DC current is a function of the total inductance in the DC circuit. At 17.5 ms, the back EMF is applied to hold the DC current at about 2 kA. At 20 ms, the current regulator resumes its normal operation and becomes to steady-state later on. Observe the 183 Hz mode in the DC current (due to the DC network resonance discussed previously).

DC Fault and Load Rejection

The DC fault to ground, at the beginning of the DC line, is applied by closing a breaker at 100 ms. The current shoots up to over 3.5 kA. Control action of the current regulator quickly pushes α further and within about 30 ms, the current level of 2 kA is reached again. The DC voltage is much smaller due to a larger firing angle α.

Observe the oscillations in the AC voltage (V_a), particularly the 206 Hz mode discussed earlier, induced by the fault and a big perturbation for the AC system.

To extinguish the fault the current is brought down to zero by blocking the converter. This complete DC load rejection imposed at 200 ms by blocking the firing pulses (C_p) causes an overvoltage on the AC side. Because a relatively strong AC system has been used, in comparison to the DC power, the overvoltage is small.

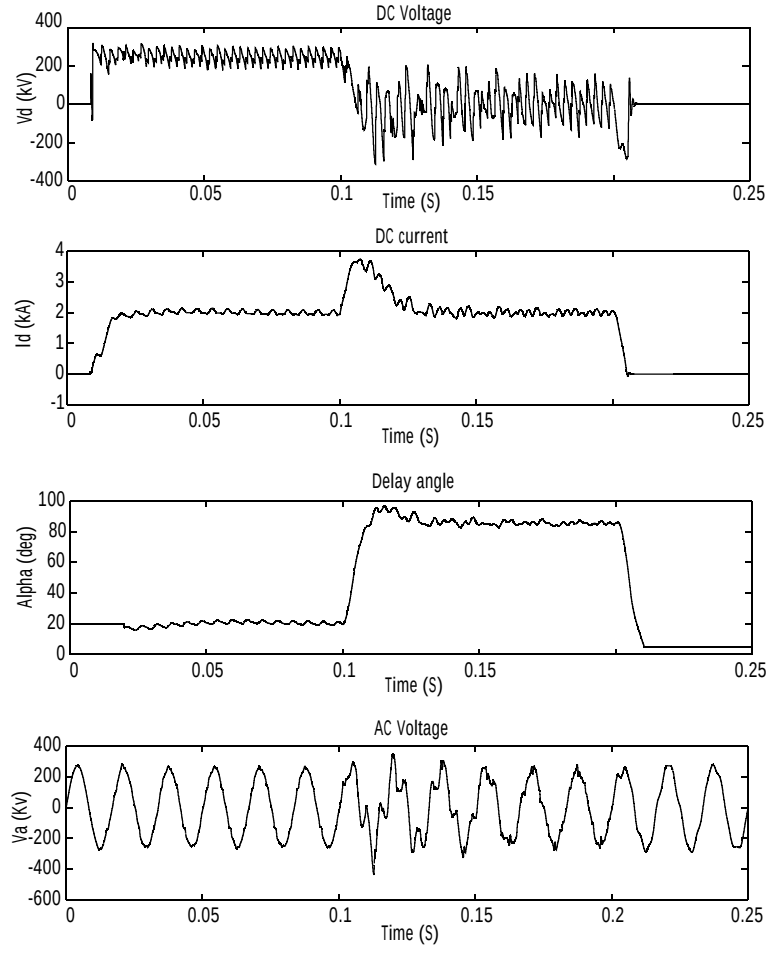


Figure 2-32 DC Fault and Load Rejection

References

[1] Arrilaga, J., *High Voltage Direct Current Transmission,* IEE Power Engineering Series 6, Peter Peregrinus Ltd., 1983.

[2] *Electromagnetic Transients Program (EMTP), Workbook IV (TACS),* EL-4651, Volume 4, Electric Power Research Institute, 1989.

3

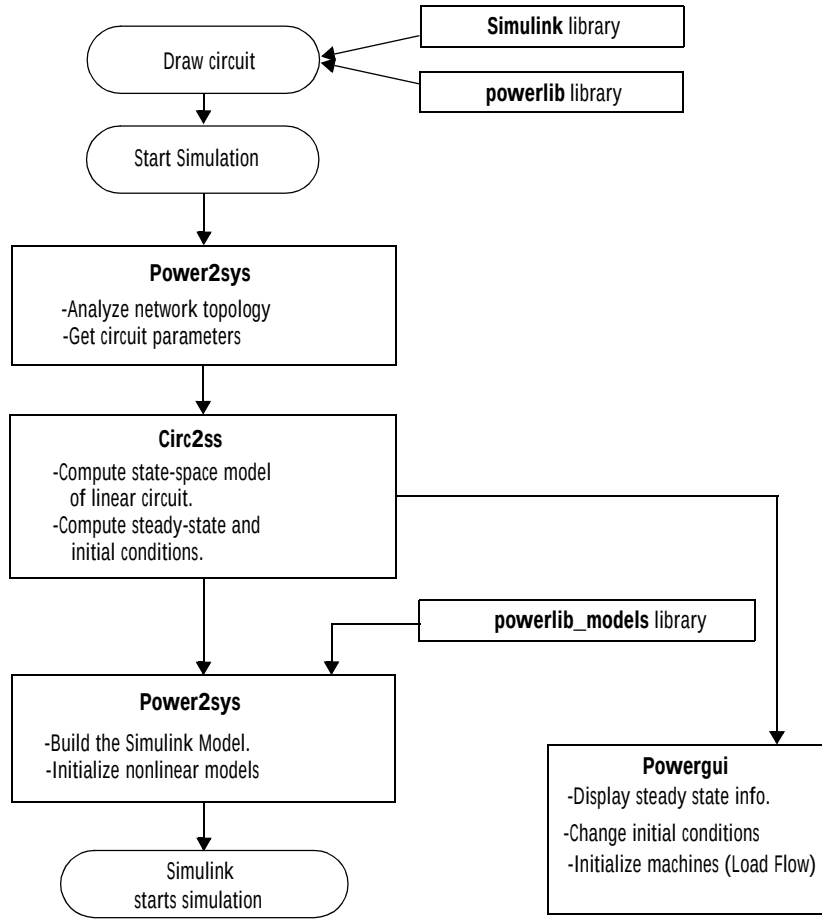
Advanced Topics

How the Power System Blockset Works

Once you have built your circuit with the electrical blocks of powerlib, you can start the simulation just as you start any other Simulink model. Each time you start the simulation, a special initialization mechanism is called. This initialization process computes the state-space model of your electric circuit and builds the equivalent system that can be simulated by Simulink.

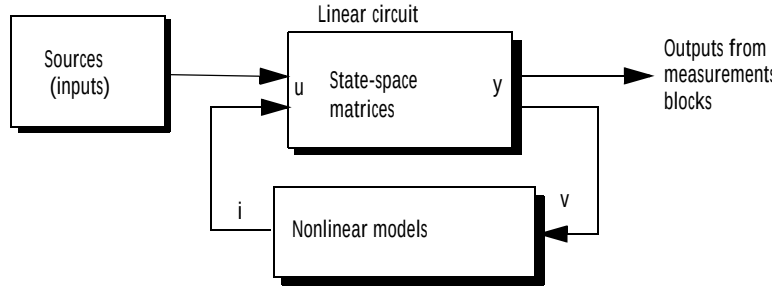
The `power2sys` function is part of the process. It gets the state-space model and builds the Simulink model of your circuit. `power2sys` can also be called from the command line to obtain the state-space model of the linear part of the circuit. When called by the initialization process, `power2sys` performs the following three steps:

- Sorts all the blocks contained in the system into two categories: the Simulink blocks and the Power System blocks. Then it gets the block parameters and evaluates the network topology. The Power System blocks are separated into linear and nonlinear blocks, and each electrical node is automatically given a node number.
- Once the network topology has been obtained, the state-space model of the linear part of the circuit is computed by the `Circ2ss` function. All steady-state calculations and initializations are performed at this stage.
- Builds the Simulink model of your circuit and stores it inside one of the measurement blocks. This means that you need at least one current or voltage measurement in your model. The connections between the equivalent circuit and measurements blocks are performed by invisible links using the Goto and From blocks.



The Simulink model uses a state-space block to model the linear part of the circuit. Pre-defined Simulink models are used to simulate nonlinear elements. These models can be found in the powerlib_models library available with the Power System Blockset. Simulink source blocks connected at the input of the state-space block are used to simulate the electrical sources blocks.

The following figure represents connections among the parts of the Simulink model. The nonlinear models are connected in feedback between voltage outputs and current inputs of the linear model.

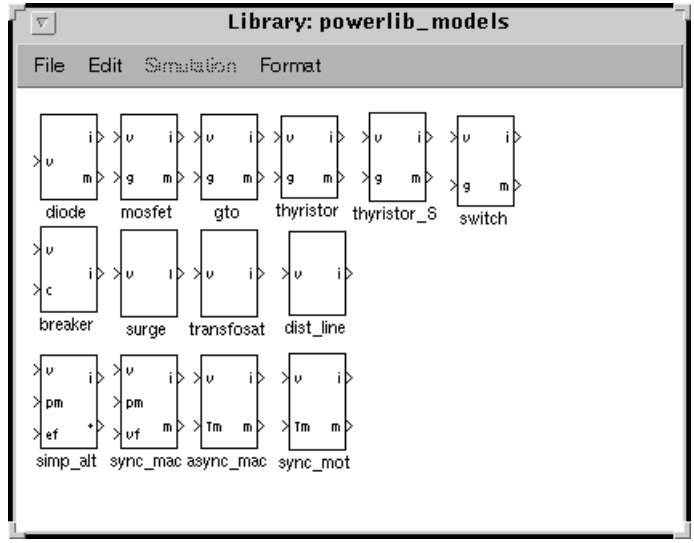


Once Power2sys has completed the initialization process, Simulink starts the simulation and you can observe waveforms on scopes connected at the outputs of your measurement blocks.

If you stop the simulation and drag a copy of the Powergui block into your circuit window, you will have access to the steady-state values of inputs, outputs, and state variables displayed as phasors. You can also use the **Powergui** interface to modify the initial conditions. Finally the **Powergui** allows you to perform a load flow with circuits involving three phase machinery and initialize the machine models so that the simulation starts in steady-state. This feature avoids long transients due to mechanical time constants of machines.

The Nonlinear Model Library

The building blocks used to assemble the Simulink model of the nonlinear circuit are stored in a library named powerlib_models. You normally don't need to work with the powerlib_models library. However, you may have to look inside the models or modify them for particular applications. You can access that library by typing `powerlib_models` in the MATLAB command window.



All the nonlinear blocks are simulated as current sources. They use a voltage input (output of the state-space model of the linear circuit) and their current output is fed into the state-space model. For complex models such as electrical machines requiring several inputs and outputs, vectorized signals are used. Useful internal signals are also returned by most of the models in a measurement output vector.

For example, the Asynchronous Machine model is stored in the block named `async_mac`. The model uses as inputs a vector of four voltages: two rotor voltages (VabR and VbcR) and two stator voltages (VabS and VbcS).

It returns a vector of four currents: two rotor currents (IaR and IbR) and two stator currents (IaS and IbS). The model also returns a measurement output vector of 20 signals that are accessible from an internal Goto block. When the

Asynchronous Machine block is used from powerlib this measurement output vector is accessible through the m output of the machine icon. You can get details on the model inputs and outputs from the documentation of the powerlib and powerlib_models block icons.

All the switch models (circuit breaker and power electronic devices) are vectorized. It means that a single model is used by Power2sys to simulate all the devices having the same type. The voltage inputs and current outputs are connected to the State-Space block through vector lines and selector blocks.

Limitations With the Nonlinear Models

Because nonlinear models are simulated as current sources, they cannot be connected in series with inductors and their terminals cannot be left open.

If, for example, you feed a machine through an inductive source, Power2sys will prompt you with an error message. This can be avoided by connecting large resistances in parallel with the source inductances or across the machine terminals.

For power electronic switches, a series RC snubber circuit is provided with each device. You won't have any problem if you keep these snubber circuits in service. For circuit breakers you may have to connect an external resistance or snubber circuit across the breaker terminals.

Modifying the Nonlinear Models

This version of the Power System Blockset does not allow you to add a block icon in the powerlib library and its corresponding model in the powerlib_models library. However, you can temporarily modify the nonlinear models provided with the Power System Blockset. To use your own powerlib_models library you must first copy the `powerlib_models.mdl` file in your working directory or in any other directory. If you are using a directory different from the current directory, you must specify this new directory in the MATLAB search path in front of the standard blockset directory.

You can then customize this new powerlib_models library as long as you don't change the names of the blocks, the number of inputs and outputs, and the number of parameters in their dialog box. The next time that you run the simulation, the modifications will take place in your circuit.

Which Integration Algorithm Must Be Used

Simulink provides a variety of solvers. Most of the variable step solvers will work well with linear circuits. However circuits containing nonlinear models, especially circuits with circuit breakers and power electronics, require stiff solvers.

Fastest simulation speed is usually achieved with `ode23tb` used with default parameters. You can also use `ode15s` with default parameters. The choice of the absolute tolerance will also have an impact on the simulation speed. Selecting too small a tolerance can slow the simulation down considerably. The choice of the absolute tolerance depends on the maximum expected magnitudes of the state variables (inductor currents and capacitor voltages). For example, if you work with high power converters where expected voltage and currents are thousands of Volts and Amperes, an absolute tolerance of 1e-1 or even 1.0 should be sufficient. If you are working with low power circuits involving maximum values of 100 V and 10 A you should use a smaller absolute tolerance like 1e-3 or 1e-2.

How to Increase Simulation Speed

Once the proper solver type and parameters have been selected, there are still some ways of optimizing the simulation speed:

- Very fast modes may be created by switching circuits. Such is the case when a circuit breaker or an electronic switch is connected across a capacitor. The switch internal R-L impedance together with the capacitance C of the linear circuit may produce fast poorly damped frequencies that will slow down the simulation. You will find in the reference documentation of the circuit breaker a selection criterion to be used for the switch Ron, Lon parameters, thus allowing damping of undesired high frequency modes.
- Simulating large systems or complex power electronic converters may be time consuming. If you have to repeat several simulations from a particular operating point you can save time by specifying a vector of initial states in the **Simulation/Parameters/Workspace IO** menu. You have to save this vector of initial conditions from a previous simulation run.
- Reducing the number of open scopes and the number of returned points will also help in reducing the simulation time.
- For circuits containing large control systems, the processing time required by Power2sys can also be reduced. Because Power2sys systematically explores all the blocks and their sub-blocks in order to build the network topology, this processing may become time consuming when your circuit is associated to a large number of Simulink blocks. You can prevent Power2sys from exploring these Simulink blocks by preceding their names with a dollar sign. For example: `$control_block`. To use this option, you must make sure that no powerlib blocks are contained in `$control_block`.

Changing Your Circuit Parameters

Each time you change a parameter of the powerlib blocks, you have to restart the simulation in order to reevaluate the state-space model and update the parameters of the nonlinear models. However, you may change any source parameter (Magnitude, Frequency or Phase) during the simulation. The modification will take place as soon as you apply the modification or close the source block menu.

As for the Simulink blocks, all the powerlib block parameters that you specify in the dialog box can contain MATLAB expressions using symbolic variable names. Before running the simulation, you must assign a value to each of these variables in your MATLAB workspace. This allows you to perform parametric studies by changing the parameter values in a MATLAB script.

Customizing Your Own Blocks

The Power System blockset provides a variety of basic building blocks to build more complex electric blocks. Using the masking feature of Simulink you can assemble several elementary blocks of powerlib into a subsystem, build your own parameter dialog box and create the desired block icon.

The Extra library provided with the blockset gives examples of masked blocks that have been used to create a three-phase library. Open this library and note how the masked blocks have been created.

You can even make your block icons change dynamically according to the parameter values. For example, open the dialog box of the **3-phase RLC parallel load block** provided in the Extra/Three-Phase library and notice how its icon changes when the active and reactive powers are successively set to `0`.

4

Block Reference

What Each Block Reference Page Contains

Blocks appear in alphabetical order and contain some or all of the following information:

- The block name and icon
- The purpose of the block
- A description of the block's use
- The block dialog box and parameters
- Additional information, as it applies to the block:
  - Inputs and outputs - A description of the inputs and outputs of the block
  - Assumptions and limitations to the block's use
- An Example using the block
- A *See Also* of related blocks

The Power System Block Libraries

The Power System Blockset main library organizes its blocks into libraries according to their behavior. The **powerlib** window displays the block library icons and names:

- The Electrical Sources library contains blocks that generate electric signals
- The Elements library contains linear and nonlinear network elements
- The Power Electronics library contains power electronics devices
- The Machines library contains machinery models
- The Connectors library contains blocks that can be used to interconnect blocks in various situations
- The Measurements library contains blocks for the current and voltage measurements
- The powerlib Extras library contains the powerlib Extra block library of tri-phase blocks and specialized measurements and control blocks. These blocks are not documented in the block reference section.
- The Demos library contains useful demos and case studies. These demos are not documented in the block reference section.

Table 4-1 Electrical Sources Library

Block Name	Purpose
AC Current Source	Implement a sinusoidal current source
AC Voltage Source	Implement a sinusoidal voltage source
Controlled Current Source	Implement a controlled current source
Controlled Voltage Source	Implement a controlled voltage source
DC Voltage Source	Implement a DC voltage source

Table 4-2 Elements Library

Block Name	Purpose
Breaker	Implement a circuit breaker opening at current zero crossing
Distributed Parameter Line	Implement an N-phases distributed parameter line model with lumped losses
Linear Transformer	Implement a two- or three-windings linear transformer
Mutual Inductance	Implement a magnetic coupling between two or three windings
Parallel RLC Branch	Implement a parallel RLC branch
Parallel RLC Load	Implement a linear parallel RLC load
PI Section Line	Implement a single phase transmission line with lumped parameters
Saturable Transformer	Implement a two- or three-windings saturable transformer
Series RLC Branch	Implement a series RLC branch
Series RLC Load	Implement a linear series RLC load
Surge Arrester	Implement a metal-oxide surge arrester

Table 4-3 Power Electronics Library

Block Name	Purpose
Diode	Implement a diode model
GTO	Implement a GTO-thyristor model
Ideal Switch	Implement an ideal switch model

Table 4-3 Power Electronics Library

Block Name	Purpose
Mosfet	Implement a mosfet model
Thyristor	Implement a thyristor model

Table 4-4 Machines Library

Block Name	Purpose
Asynchronous Machine	Model the dynamics of a three-phase asynchronous machine
Excitation System	Provide an excitation system for the synchronous machine and regulate its terminal voltage in generating mode
Hydraulic Turbine and Governor	Model a hydraulic turbine and a PID governor system
Permanent Magnet Synchronous Machine	Model the dynamics of a three-phase permanent magnet synchronous machine with sinusoidal flux distribution
Simplified Synchronous Machine	Model the dynamics of a simplified three-phase synchronous machine
Synchronous Machine	Model the dynamics of a three-phase salient-pole synchronous machine

Table 4-5 Connectors Library

Block Name	Purpose
Bus Bar	Implement a labeled network node
Ground	Provide a connection to the ground
Neutral	Implement a local common node in the circuit

Table 4-6 Measurements Library

Block Name	Purpose
Current Measurement	Measure a current in a circuit
Voltage Measurement	Measure a voltage in a circuit

Purpose

Implement a sinusoidal current source

Library

Electrical Sources Library

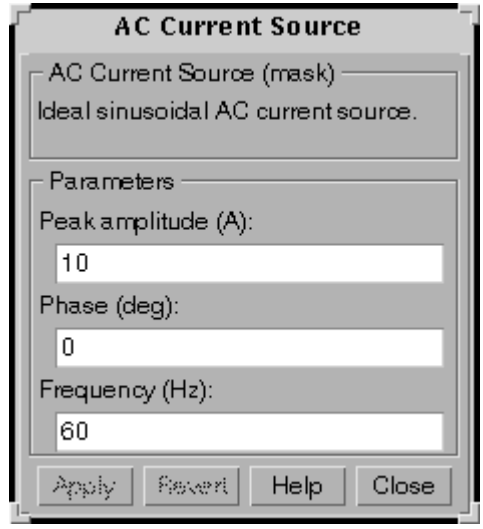
Description

The AC Current Source block implements an ideal AC current source. The positive current direction is indicated by the arrow in the block icon. The generated current is described by the following relationship:

$$I = Amplitude \times \sin(2pi \times Frequency \times t + Phase \times pi/180)$$

Negative values are allowed for amplitude and phase. Negative frequency is not allowed, otherwise Simulink signals an error, and the block will display a question mark in the block icon. You can modify the three source parameters at any time during the simulation.

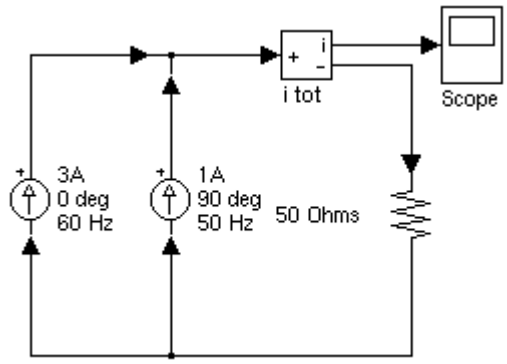
Dialog Box



AC Current Source

Example

Parallel connection of two Current Source blocks is used to sum two sinusoidal currents in a resistor.



This circuit is available in the `psbaccurrent.mdl` file.

See Also

Controlled Current Source

Purpose

Implement a sinusoidal voltage source

Library

Electrical Sources Library

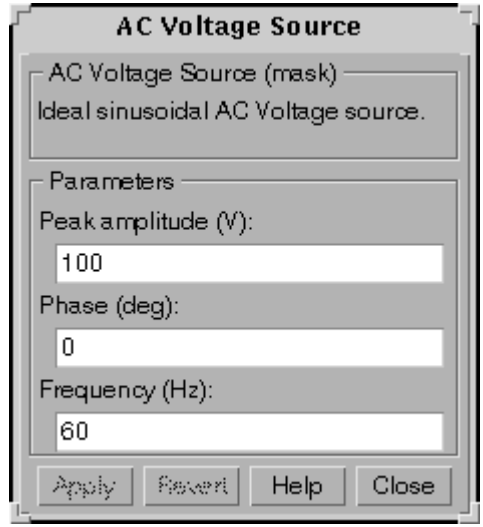
Description

The AC Voltage Source block implements an ideal AC voltage source. The output and input of the block correspond respectively to the positive and negative terminals of the source. The generated voltage *U* is described by the following relationship:

$$U = Amplitude \times \sin(2pi \times Frequency \times t + Phase \times pi/180)$$

Negative values are allowed for amplitude and phase. Negative frequency is not allowed, otherwise Simulink signals an error, and the block will display a question mark in the block icon. You can modify the three source parameters at any time during the simulation.

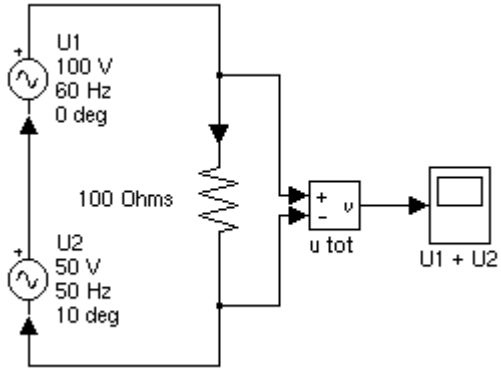
Dialog Box



AC Voltage Source

Example

Two Voltage Source blocks at different frequencies are connected in series across a resistor. The sum of the two voltages is read by a Voltage Measurement block.



This circuit is available in the `psbacvoltage.mdl` file.

See Also

Controlled Voltage Source, **DC Voltage Source**

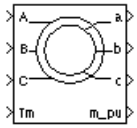
Purpose

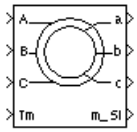
Model the dynamics of a three-phase asynchronous machine

Library

Machines Library

Description

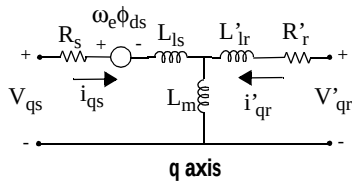


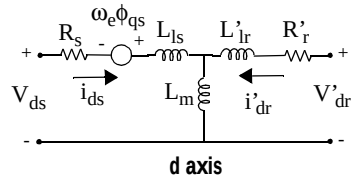


The Asynchronous Machine block operates in either generating or motoring mode. The mode of operation is dictated by the sign of the mechanical torque (positive for motoring, negative for generating). The electrical part of the machine is represented by a fourth-order state-space model and the mechanical part by a second-order system. All electrical variables and parameters are viewed from the stator. This is indicated by the prime signs (') in the machine equations given below. All stator and rotor quantities are in the rotor reference frame (qd frame). The subscripts used are defined as follows:

- *d*: d axis quantity
- *q*: q axis quantity
- *r*: rotor quantity
- *s*: stator quantity
- *l*: leakage inductance
- *m*: mutual inductance

Electrical System





$$V_{qs} = R_s i_{qs} + \frac{d}{dt}\varphi_{qs} + \omega_e \varphi_{ds}$$

$$V_{ds} = R_s i_{ds} + \frac{d}{dt}\varphi_{ds} - \omega_e \varphi_{qs}$$

$$V'_{qr} = R'_r i'_{qr} + \frac{d}{dt}\varphi'_{qr}$$

$$V'_{dr} = R'_r i'_{dr} + \frac{d}{dt}\varphi'_{dr}$$

$$T_e = 1.5p(\varphi_{ds} i_{qs} - \varphi_{qs} i_{ds})$$

where

$$\varphi_{qs} = L_s i_{qs} + L_m i'_{qr}$$

$$\varphi_{ds} = L_s i_{ds} + L_m i'_{dr}$$

$$\varphi'_{qr} = L'_r i'_{qr} + L_m i_{qs}$$

$$\varphi'_{dr} = L'_r i'_{dr} + L_m i_{ds}$$

$$L_s = L_{ls} + L_m$$

$$L'_r = L'_{lr} + L_m$$

Mechanical System

$$\frac{d}{dt}\omega_r = \frac{1}{2H}(T_e - F\omega_r - T_m)$$

$$\frac{d\theta}{dt} = \omega_r$$

The Asynchronous Machine parameters are defined as follows (all quantities in the rotor reference frame):

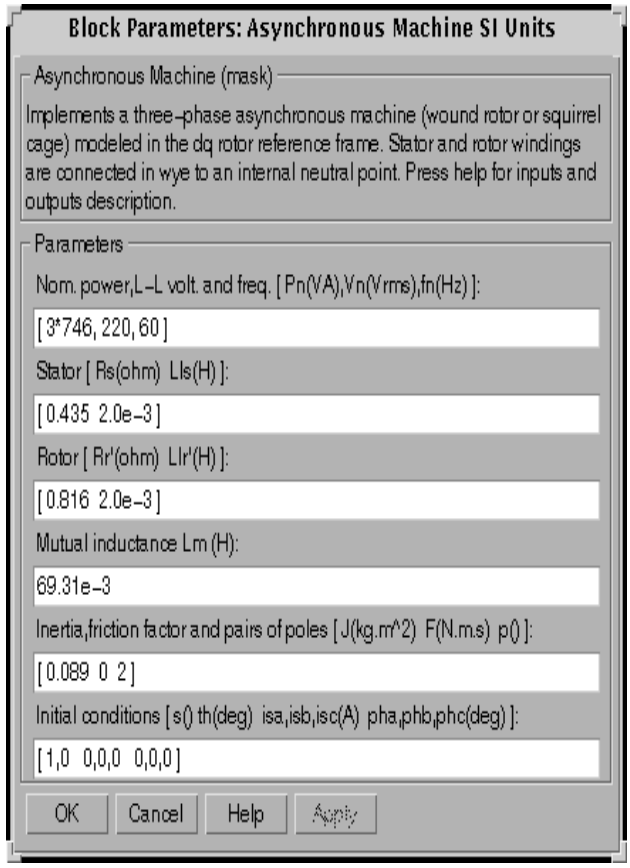
- R_s, L_{ls}: stator resistance and leakage inductance
- R'_r, L'_{lr}: rotor resistance and leakage inductance
- L_m: mutual inductance
- L_s, L'_r: total stator and rotor inductances
- V_{qs}, i_{qs}: q axis stator voltage and current
- V'_{qr}, i'_{qr}: q axis rotor voltage and current
- V_{ds}, i_{ds}: d axis stator voltage and current

- V'_{dr}, i'_{dr}: d axis rotor voltage and current
- φ_{qs}, φ_{ds}: stator q and d axis fluxes
- φ'_{qr}, φ'_{dr}: rotor q and d axis fluxes
- ω_r : angular velocity of the rotor
- p: number of pole pairs
- ω_e : electrical angular velocity ($\omega_r \times p$)
- T_e: electromagnetic torque
- T_m: shaft mechanical torque
- θ : rotor angular position
- J: combined rotor and load inertia (set to infinite to simulate locked rotor)
- H: combined rotor and load inertia constant (set to infinite to simulate locked rotor)
- F: combined rotor and load viscous friction

Parameters and Dialog Boxes

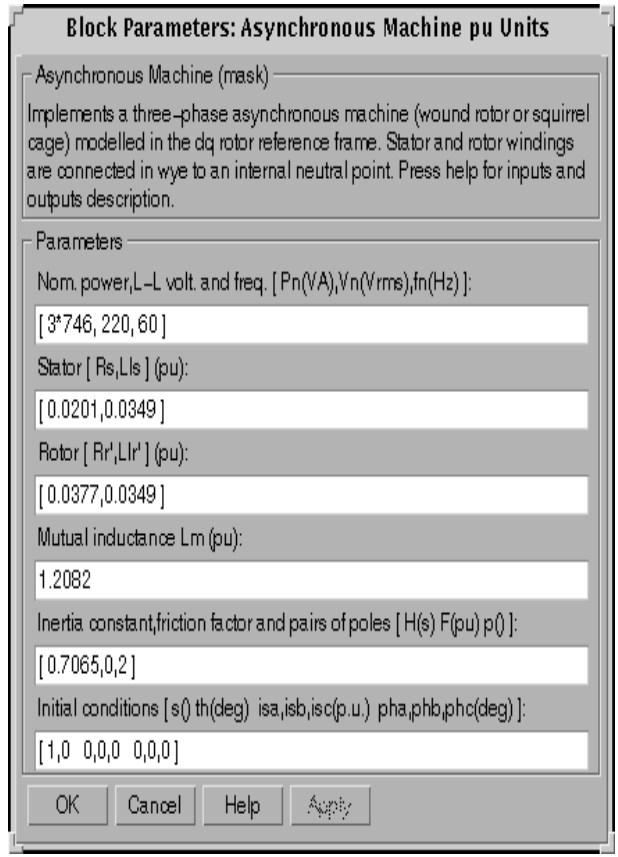
In the powerlib library you can choose between two Asynchronous Machine blocks to specify the electrical and mechanical parameters of the model.

S.I. Units Dialog Box



If you choose to enter the parameters in S.I. units, you must first enter the nominal power (VA), line-to-line rms voltage (V) and frequency (Hz). The second entry is the stator resistance (ohm) and leakage inductance (H). The third entry is the rotor resistance (ohm) and leakage inductance (H), both viewed from the stator. The fourth entry is the mutual inductance (H). The fifth entry is the inertia ($kg.m^2$), the viscous friction coefficient (N.m.s) and the number of pole pairs. Finally, the sixth entry is where you specify initial slip, electrical angle (degrees) and stator currents (magnitude in A, phase angle in degrees).

Per Unit (pu) Dialog Box



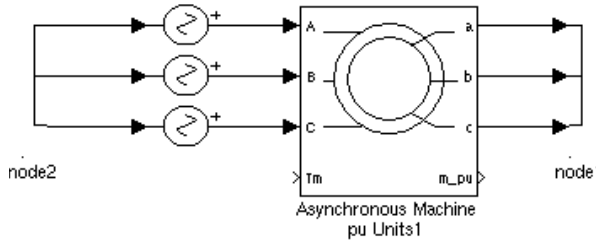
If you choose to enter the parameters in per unit, you must enter at the first line the nominal power (VA), line-to-line rms voltage (V) and frequency (Hz). You then enter the electrical and mechanical parameters expressed in pu, with the exception of H, which is expressed in s. Initial conditions are entered the same way as in the S.I. units case, except that the current magnitudes are in pu.

Note: These two blocks simulate the same asynchronous machine model. The only difference is that the parameters are expressed in different units.

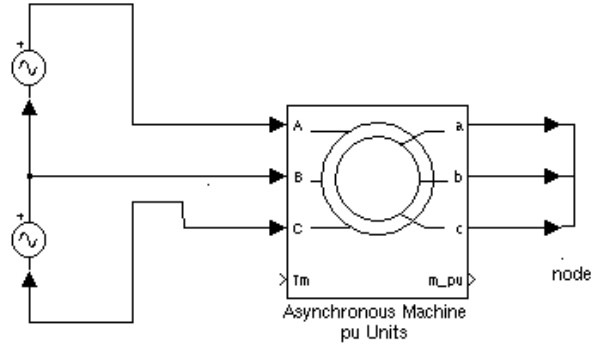
Inputs and Outputs

The electrical inputs of the block are the three electrical connections of the stator, and the electrical outputs are the three electrical connections of the rotor. Note that the neutral connections of the stator and rotor windings are not available; three-wire Y connections are assumed. The rotor's connections should ordinarily be short-circuited or connected to an external circuit, for example external resistors or a power converter.

You must be careful when you connect ideal sources to the machine's stator. If you choose to supply the stator via a three-phase Y-connected infinite voltage source, you must use three sources connected in Y.



However, if you choose to simulate a source delta connection, you must only use two sources connected in series:

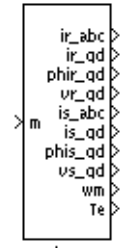


The Simulink input of the block is the mechanical torque at the machine's shaft. This input must be positive in the motoring mode and negative in the generating mode.

The Simulink output of the block is a vector containing 20 variables. They are, in order (refer to "Description" section, all currents flowing into machine):

- 1-3:rotor currents i'_{ra}, i'_{rb} and i'_{rc}
- 4-9:i'_{qr}, i'_{dr}, φ'_{qr}, φ'_{dr}, v'_{qr} and v'_{dr}
- 10-12:stator currents i_{sa}, i_{sb} and i_{sc}
- 13-18:i_{qs}, i_{ds}, φ_{qs}, φ_{ds}, v_{qs} and v_{ds};
- 19-20:ω_r and T_e

These variables can be demultiplexed by using the special Asynchronous Machine Demux block provided in the Machine library.

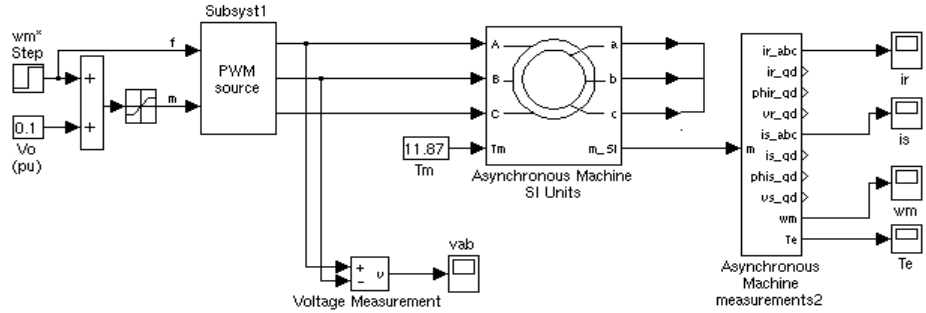


Limitations

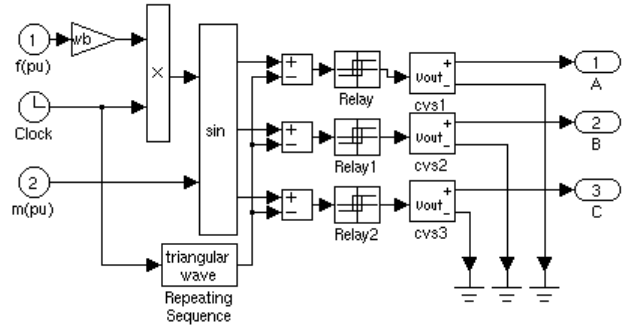
The Asynchronous Machine block does not include a representation of the effects of stator and rotor iron saturation.

Example

This example illustrates the use of the Asynchronous Machine block in motoring mode. It consists of an asynchronous machine in an open-loop speed control system. The machine's rotor is short-circuited and the stator is fed by a PWM inverter, which is built with Simulink blocks and interfaced to the Asynchronous Machine block through the Controlled Voltage Source blocks. The inverter uses sinusoidal pulse-width modulation, which is described in [1]. The base frequency of the sinusoidal reference wave is set at 60 Hz and the triangular carrier wave's frequency is set at 360 Hz. The 3 HP machine is connected to a constant load of nominal value (11.87 N.m). It is started and reaches the setpoint speed of 0.5 pu at t=1.0 second, at which time the setpoint is stepped to 0.75 pu. The parameters of the machine are those found in the **SI** dialog box.



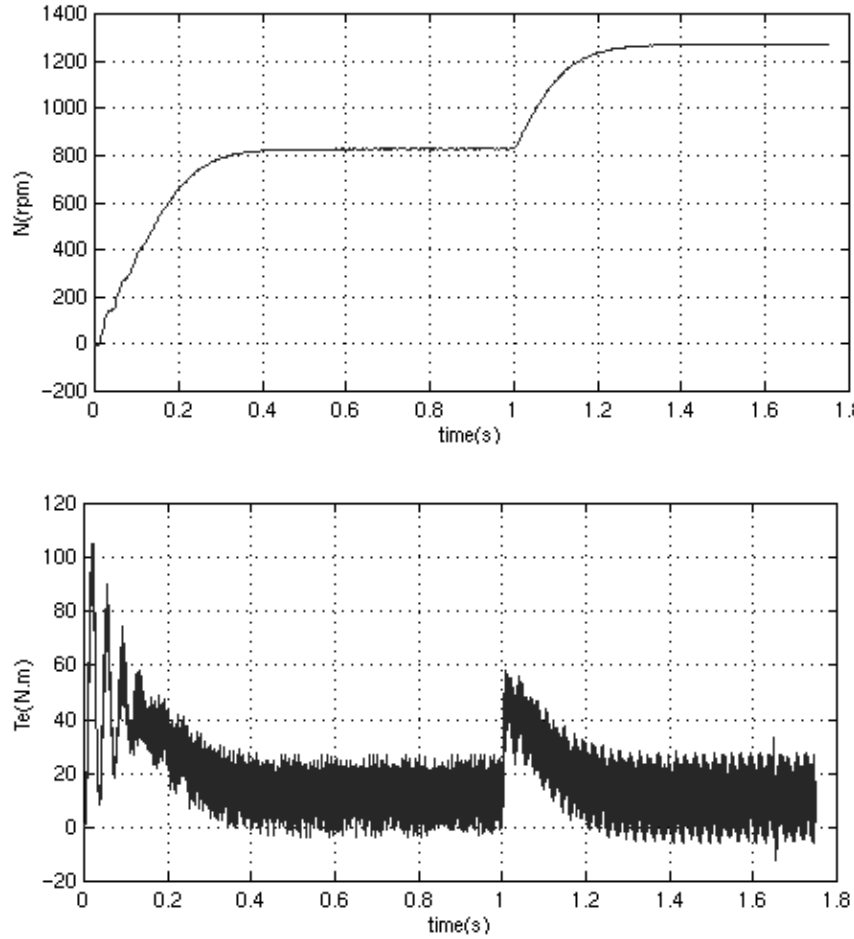
Subsyst1 (PWM source)



Open the Simulink diagram by typing `psbpwm`, or by double clicking on Asynchronous Machine (sim) in the demos library of powerlib. Set the simulation parameters as follows:

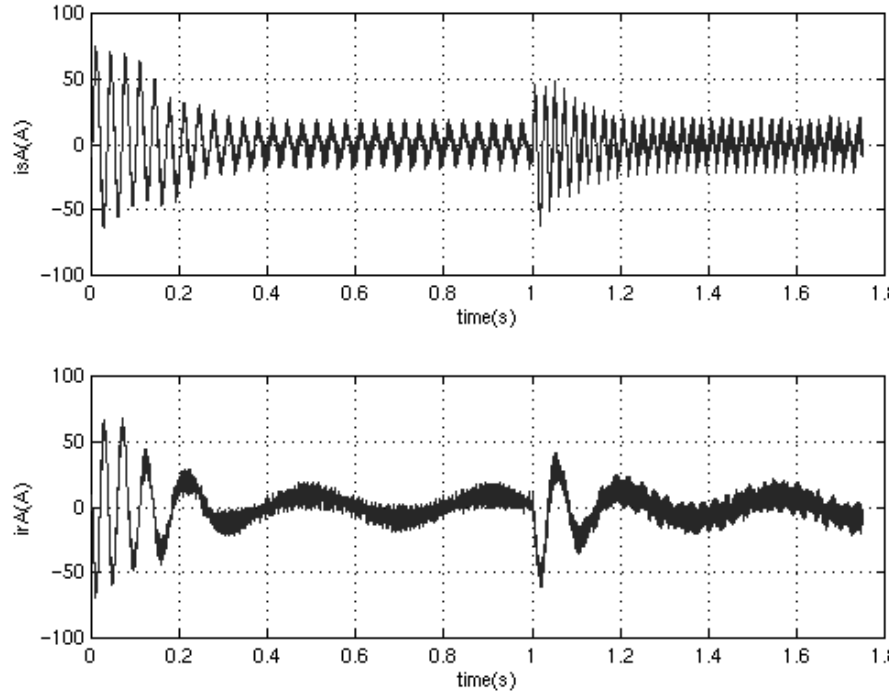
- Integrator type: `Stiff, ode15s`
- Stop time: `1.75 s`
- Integration options: Use default options, except for Relative and Absolute tolerance, which must be set to `1e-4` and `1e-8`, respectively.

Run the simulation by choosing **Start** from the **Simulation** menu. Once the simulation is completed, observe the machine's speed and torque.

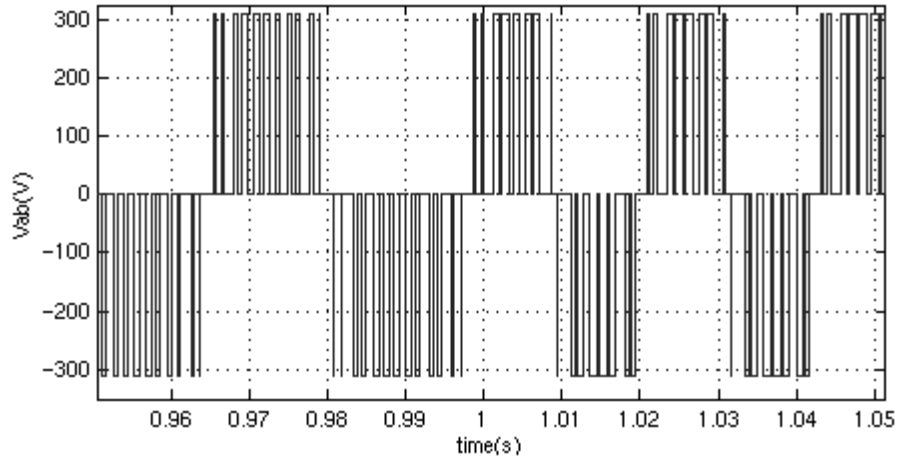


The top graph shows the machine's speed going from 0 to 810 rpm (0.5 pu) and then to 1252 rpm (0.75 pu). The bottom graph shows the electromagnetic torque developed by the machine. Since the stator is fed by a PWM inverter, a noisy torque is observed.

This noise is not visible in the speed however because it is filtered out by the machine's inertia, but it can also be seen in the stator and rotor currents, which are observed next.



Finally, look at the output of the PWM inverter. Because nothing of interest can be seen at the simulation time scale, the graph concentrates on the moments preceding and following the speed step (around 1 second).



References

[1] Dubey G.K., *Power Semiconductor Controlled Drives*, section 8.1.4, Prentice-Hall, Inc., 1989.

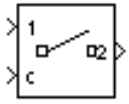
Purpose

Implement a circuit breaker opening at current zero crossing

Library

Elements Library

Description

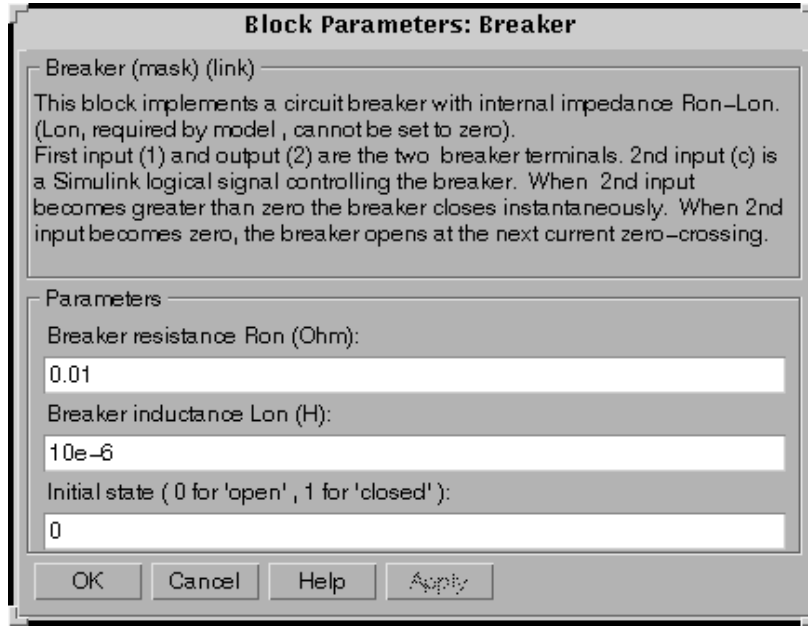


The Breaker block implements a circuit breaker that is controlled by a Simulink signal applied on its second input. The control signal must be either 0 or 1, 0 for open and 1 for closed. The arc extinction process is simulated by opening the breaker when the current passes through zero (first current zero-crossing following the transition of the Simulink control input from 1 to 0).

When the breaker is closed, it behaves as a series RL circuit. The R and L values can be set as small as necessary in order to be negligible compared with external components (typical values Ron=10mΩ, Lon=10μH). When the breaker is open, it has an infinite impedance.

If the breaker's initial state is set to 1 (closed), `power2sys` automatically initializes all the states of the linear circuit and breaker initial current so that the simulation starts in steady-state.

Dialog Box



Limitations

The circuit breaker is modeled as a current source driven by the voltage appearing across its terminals. Therefore, it cannot be connected in series with an inductor or another current source. You can avoid this by connecting a large resistor across its terminals. The internal breaker inductance cannot be set to zero. However, a null resistance is allowed.

It can happen that a breaker is connected across a capacitor. Depending on the capacitor value, breaker resistance and inductance, a high-frequency poorly damped oscillatory voltage, and current oscillation can be produced when the breaker closes. If the breaker parameters (Ron, Lon) are not properly selected, this will result in a slow simulation speed. This situation can arise, when a circuit breaker is used to simulate a phase-to-ground fault at one end of the PI line section.

To damp this high frequency mode, set the inductance of the circuit breaker sufficiently small. A good practice is to achieve a damping factor z<0.5 for the RLC circuit formed by the circuit breaker and the capacitor. This condition is obtained for

$$L_{on} < R_{on}^{2}\ C$$

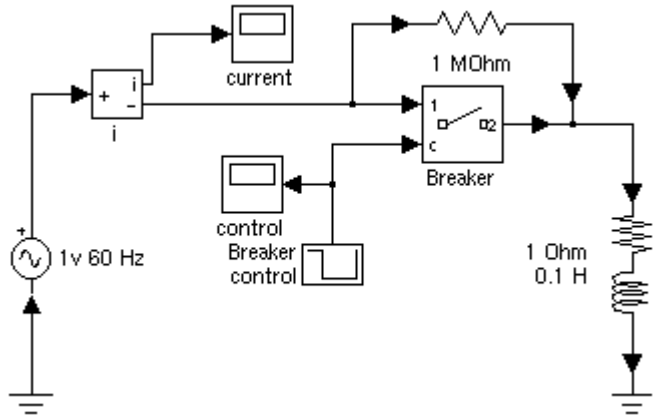
For example, for a capacitor C=1μF and a breaker resistance Ron=0.01Ω, the breaker inductance Lon should be selected so that:

$$L_{on} \cong 10^{-10} H$$

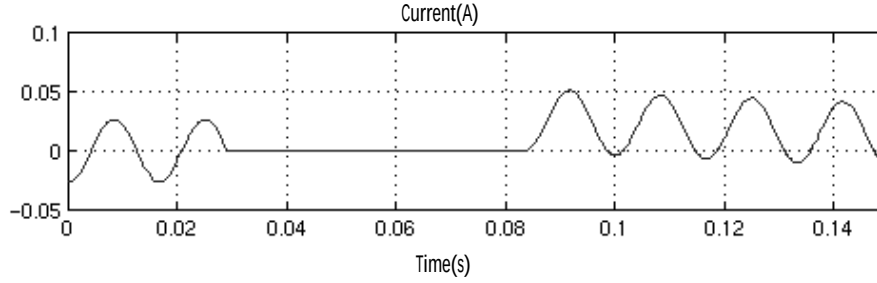
You must use a stiff integration algorithm to simulate circuits with a circuit breaker. `Ode15s` usually gives the best simulation speed.

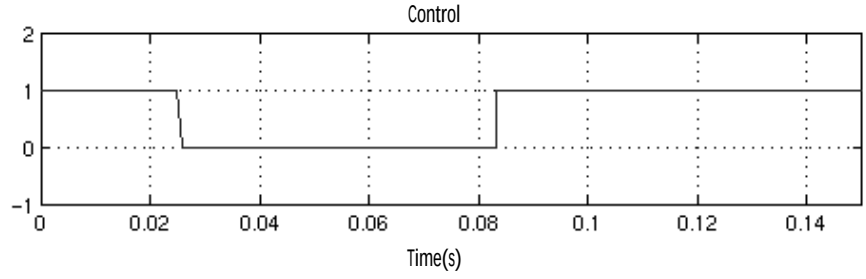
Example

A circuit breaker is connected in series with a series RL circuit on a 60Hz voltage source. The breaker is initially closed and an opening order is given at t = 1.5 cycles, when current reaches a maximum. The current stops at the next zero crossing; then the breaker is reclosed at a zero crossing of voltage at t = 5 cycles.



This circuit is available in the `psbbreaker.mdl` file. Simulation produces the following results.



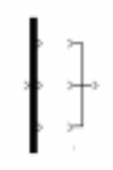


Bus Bar

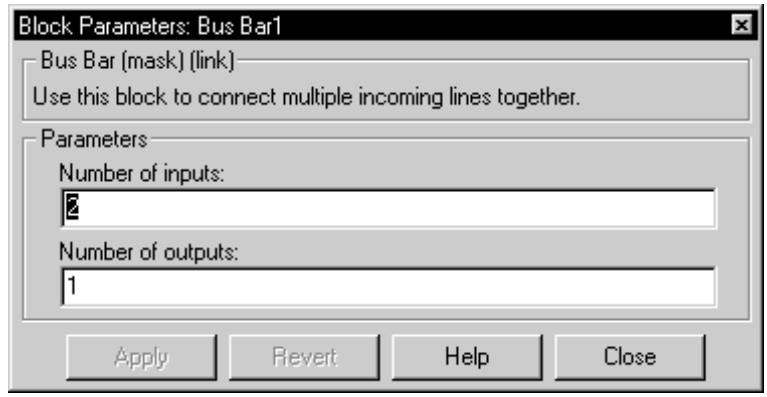
Purpose Implement a labeled network node

Library Connectors Library

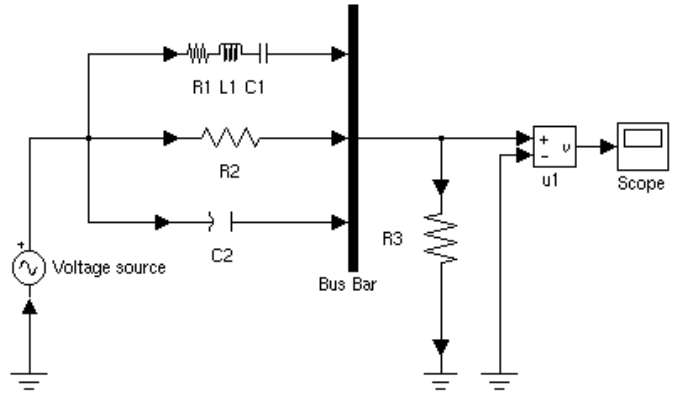
Description The Bus Bar block is used to interconnect components. It allows multiple electrical block outputs and inputs to be connected together.



Dialog Box



Example A three input, one output bus bar is used to connect three elements in parallel.



This circuit is available in the `psbbusbar.mdl` file.

See Also

Ground, **Neutral**

Circ2ss

Purpose

Compute the state-space model of a linear electrical circuit

Synopsis

You must call `circ2ss` with a minimum of seven input arguments.

```
[A,B,C,D,states,x0,x0sw,rlsw,u,x,y,freq,Asw,Bsw,Csw,Dsw,Hlin]=
circ2ss(rlc,switches,source,line_dist,yout,y_type,unit)
```

You can also specify additional arguments. To use options, the number of input arguments must be 12, 13, 14 or 16.

```
[A,B,C,D,states,x0,x0sw,rlsw,u,x,y,freq,Asw,Bsw,Csw,Dsw,Hlin]=
circ2ss(rlc,switches,source,line_dist,yout,y_type,unit,net_arg1,
net_arg2,net_arg3,...
netsim_flag,fid_outfile,freq_sys,options,vary_name,vary_val)
```

Description

Computes the state-space model of a linear electrical circuit expressed as

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

where x is the vector of state-space variables (inductor currents and capacitor voltages), u is the vector of voltage and current inputs, and y is the vector of voltage and current outputs.

When you build a circuit from the Power System Block set icons of the powerlib library, `circ2ss` is automatically called by `power2sys`. `circ2ss` has also been made available as a stand-alone function for expert users. This allows you to generate state-space models without using the Power System Blockset graphical interface and to access options that are not available through powerlib. For example, you can specify transformers and mutual inductances with more than three windings.

The linear circuit can contain any combination of voltage and current sources, RLC branches, multi-winding transformers, mutually coupled inductances, and switches. The state variables are inductor currents and capacitor voltages.

The State-Space block containing A,B,C,D and x0 matrices computed by `circ2ss` can then be used in a Simulink system to perform simulation of the electrical circuit. Nonlinear elements (such as mechanical or electronic switches, transformer saturation, machines, and distributed parameter lines) can be connected to the linear circuit. These Simulink models are interfaced

with the linear circuit through voltage outputs and current inputs of the state-space model. You can find the models of the nonlinear elements provided with the Power System Blockset in the powerlib_models library.

Input Arguments

The number of input arguments must be 7, 12, 13, 14, or 16. Arguments 8 to 16 are optional. The first seven arguments that must be specified are (see format following):

- `rlc`: Branch matrix specifying the network topology as well as the resistance R, inductance L, and capacitance C values.
- `switches`: Switch matrix. Specify an empty variable if no switches are used.
- `source`: Source matrix specifying the parameters of the electrical voltage and current sources. Specify an empty variable if no sources are used.
- `line_dist`: Distributed parameter line matrix. Specify an empty variable if no distributed lines are used.
- `yout`: String matrix of output expressions.
- `y_type`: Integer vector indicating output types (zero for voltage output, one for current output).
- `unit`: String specifying the units to be used for R L and C values in the rlc matrix. If `unit= 'OHM'`, R L C values are specified in ohms at the fundamental frequency specified by `freq_sys` (default value is 60Hz). If `unit= 'OMU'`, R L C values are specified in ohms, millihenries (mH) and microfarads (µF).

The following arguments are optional. Some of them are used to pass arguments from the `power2sys` function. Hereafter, we describe only the arguments to be specified when `circ2ss` is used as a stand-alone function.

- `net_arg1, net_arg2 net_arg3`: Used to pass arguments from `power2sys`. Specify empty variables [] for each of these variables.
- `netsim_flag`: Integer controlling the messages displayed during the execution of `circ2ss`. Default value is zero, no output file is produced.

  If `netsim_flag` =0, the version number, number of states, inputs, outputs and modes are displayed. Output values are displayed in polar form for each source frequency.

  If `netsim_flag`=1, only version number, number of states, inputs and outputs are displayed.

  If `netsim_flag`=2, no message is displayed during execution.

- `fid_outfile`: File identifier of the `circ2ss` output file containing parameter values, node numbers, steady-state outputs, and special messages. Default value is zero.
- `freq_sys` : Fundamental frequency (Hz) considered for specification of XL and XC reactances if `unit` is set to 'OHM'. Default value is 60 Hz.
- `options`: Integer vector specifying five options. Default values are [0 0 0 0 0].

  `option(1)` : Reference node number used for ground of pi transmission lines. If -1 is specified, the user will be prompted to specify a node number.

  `option(2)`: Specify normalization of A B C D matrices in order to use the per-unit system. This option is not available from `power2sys`. There is no normalization if `option(2)=0`. If `option(2)=1 cir2ss` will use per-unit system. The base voltage and base power are specified by `option(3)` and `option(4)`. If `option(2)=-1` the user will be prompted to specify a base voltage and a base power.

  `option(3)`: Base voltage (kVrms Phase-Phase)

  `option(4)`: Base power (MVA 3 phase)

  `option(5)`: Enter `0`
- `vary_name`: String matrix containing the symbolic variable names used in output expressions. These variables must be defined in your MATLAB workspace.
- `vary_val`: Vector containing the values of the variable names specified in `vary_name`.

Output Arguments

- `A,B,C,D`: State-space matrices of the linear circuit with all switches open. A (`nstates`, `nstates`), B (`nstates`, `ninput`), C (`noutput`, `nstates`), D (`noutput`, `ninput`) where `nstates` is the number of state variables, `ninput` is the number of inputs, and `noutput` is the number of outputs.
- `states`: String matrix containing the names of the state variables. Each string has the following format:
  -Inductor currents: `Il_bxx_nzz1_zz2`
  -Capacitor voltages: `Uc_bxx_nzz1_zz2`
  where:
  `-xx` = branch number
  `-zz1`= 1st node number of the branch
  `-zz2` = 2nd node number of the branch

The last lines of the states matrix, which are followed by an asterisk, indicate inductor currents and capacitor voltages which are not considered as state variables. This situation arises when inductor currents or capacitor voltages are not independent (inductors in series or capacitors forming a loop). The currents and voltages followed by an asterisk can be expressed as a linear combination of the other state variables:

- x0: Column vector of initial values of state variables considering the open or closed status of switches
- x0sw: Vector of initial values of switch currents
- rlsw: Matrix (nswitch,2) containing the R and L values of series switch impedances in Ohm and Henry. nswitch is the number of switches in the circuit.
- u, x, y: Matrices u(ninput, nfreq), x (nstates, nfreq) and y (noutput, nfreq) containing the steady-state complex values of inputs, states, and outputs. nfreq is the length of the freq vector. Each column corresponds to a different source frequency, as specified by the next argument freq.
- freq: Column vector containing the source frequencies ordered by increasing frequencies
- Asw, Bsw, Csw, Dsw: State-space matrices of the circuit including the closed switches. Each closed switch adds one extra state to the circuit.
- Hlin: Three dimension array (nfreq, noutput, ninput) of the nfreq complex transfer impedance matrices of the linear system corresponding to each frequency of the freq vector

Format of the RLC Input Matrix

Two formats are allowed:

- Six columns: No branch numbering. Branch numbers correspond to the RLC line numbers.
- Seven columns: Branch numbering; Nobr variable is imposed by the user.

Each line of the RLC matrix must be specified according to the following format:

```
[node1, node2, type, R, L, C, Nobr] for RLC or line branch
[node1, node2, type, R, L, C, Nobr] for transformer magnetizing branch
[node1, node2, type, R, L, U, Nobr] for transformer winding
[node1, node2, type, R, L, U, Nobr] for mutual inductances
```

- node1: First node number of the branch. The node number must be positive or zero. Decimal node numbers are allowed.
- node2: Second node number of the branch. The node number must be positive or zero. Decimal node numbers are allowed.

  type: Integer indicating the type of connection of RLC elements, or the transmission line length (negative value).

  type=0: Series RLC element,

  type=1: Parallel RLC element

  type=2: Transformer winding

  type=3: Coupled (mutual) winding

  If type is negative value: transmission line modeled by a PI section. See details below.

For a mutual inductor or a transformer having N windings, N+1 consecutive lines must be specified in rlc matrix:

1. N lines with type=2 or type=3; (one line per winding). Each line specifies R/L/U or R/Xl/Xc, where [R/L, R/Xl] = winding resistance and leakage reactance for a transformer or winding resistance and self reactance for mutually coupled windings. U is the nominal voltage of transformer winding (specify 0 if type =3).
2. One extra line with type=1 for the magnetizing branch of a transformer (parallel Rm/Lm or Rm/Xm) or one line with type=0 for a mutual impedance (series Rm/Lm or Rm/Xm).

For a transformer magnetizing branch or a mutual impedance, the first node number is an internal node located behind the leakage reactance of the first winding. The second node number must be same as the second node number of the first winding.

To model a saturable transformer you must use a nonlinear inductance instead of the linear inductance simulating the reactive losses. Set the Lm/Xm value to zero (no linear inductance) and use the transfosat block with proper flux-current characteristic. This block can be found in the powerlib_models library. This block must be connected to the State-Space block between a voltage output (voltage across the magnetizing branch and a current input (current source injected into the transformer internal node). See the example given at the end of the Circ2ss documentation.

If type is a negative value: length (km) of a transmission line simulated by a PI section.
For a transmission line, the R,L,C or R/Xl/Xc values must be specified in ohm/km or ohm, mH, µF /km.

- R: Branch resistance (ohms or pu)
- Xl: Branch inductive reactance (ohms or pu at freq_sys) or transformer winding leakage reactance (ohms)
- L: Branch inductance (mH)
- Xc: Branch capacitive reactance (ohms or pu at freq_sys) (The negative sign of Xc is optional)
- C: Capacitance (µF)
- U: Nominal voltage of transformer winding. Same units Volts or kV must be used for each winding. For a mutual inductance (type=3), this value must be set to zero.

  Zero values for R, L or Xl, C or Xc in a series or parallel branch indicate that the corresponding element does not exist.

The following restrictions apply for transformer and mutually coupled winding R-L values: Null values are not allowed for winding resistances. Specify a very low value (e.g., 1e-6 pu, based on rated voltage and power) to simulate a quasi ideal transformer. The leakage reactances can be set to zero. The resistive part of the magnetizing branch Rm of a transformer must have a finite value. Specify a very high value (low losses, e.g, Rm=1e4 pu or 0.01% current based on rating) to simulate a quasi ideal transformer. The inductive part of the magnetizing branch can be set to infinite (No reactive losses; specify Xm=0). A null value is not allowed for the mutual inductance of coupled windings. The resistive part of series mutual branch can be set to zero.

Format of the SOURCE Input Matrix

Three formats are allowed:

- Five columns: All sources are generating the same frequency specified by freq_sys
- Six columns: The frequency of each source is specified in column 6
- Seven columns: The 7th column is used to specify the type of nonlinear element modeled by the current source.

Each line of the Source matrix must be specified according to the following format:

```
[node1, node2, type, amp, phase, freq, model]
```

- `node1, node2`: Node numbers corresponding to the source terminals Polarity conventions: Voltage source: `node1` is the positive terminal. Current source: Positive current flowing from `node1` to `node2` inside the source.
- `type`: Integer indicating the type of source: `0` for voltage source; `1` for current source
- `amp`: Amplitude of the AC or DC voltage or current (V, A or pu)
- `phase`: Phase of the AC voltage or current (degree)
- `freq`: Frequency (Hz) of the generated voltage or current. Default value is 60 Hz. For a DC voltage or current source specify `phase=0` and `freq=0`. `amp` can be set to a negative value. The generated signals are: `amp*sin(2*pi*freq*t+phase)` for AC, `amp` for DC.
- `model`: Integer specifying the type of nonlinear element modeled by the current source (saturable inductance, thyristor, switch. . .). Used by `power2sys` only.

Format of the SWITCHES Input Matrix

Switches are nonlinear elements simulating mechanical or electronic devices such as circuit breakers diodes or thyristors. As for other nonlinear elements, they are simulated by current sources driven by the voltage appearing across their terminals. Therefore, they cannot have a perfectly zero impedance. They are simulated as ideal switches in series with a series R-L circuit. Various models of switches (circuit breaker, ideal switch, and power electronic devices) are available in the powerlib_models library. They must be interconnected to the State-Space block through appropriate voltage outputs and current inputs.

The switch parameters must be specified in a line of the Switches matrix in seven different columns according to the following format:

```
[node1, node2, status, R, L/Xl, no_I, no_U]
```

- `node1, node2`: Node numbers corresponding to the switch terminals,
- `status`: Code indicating the initial status of the switch at `t=0`
  `0`= open
  `1`= closed
- `R`: Resistance of the switch when closed (ohms or pu)

- `L/Xl`: Inductance of the switch when closed (mH) or inductive reactance (ohms or pu at `freq_sys`).

Note: For these last two fields, the same units as specified for the `rlc` matrix must be used. Null inductance values are not allowed. However, the resistance can be set to zero.

The next two fields specify the current input number and the voltage output number to be used for interconnecting the switch model to the state-space block.

- `no_I`: Current input number coming from the output of the switch model.
- `no_U`: Voltage output number driving the input of the switch model.

Format of the LINE_DIST Matrix

The distributed parameter line model contains two parts:

1. A linear part containing current sources and resistances that are connected at the line sending and receiving buses together with the linear circuit.
2. A nonlinear part available in the dist_line block of the powerlib_models library. This block performs the phase to mode transformations of voltage and currents and simulates the transmission delays for each mode. The dist_line block must be connected to appropriate voltage outputs and current inputs of the State-Space block. The line parameters have to be specified in the line_dist matrix and also in the dist_line block.

Each row of the line_dist matrix is used to specify a distributed parameter transmission line. The number of columns of line_dist depends on the number of phases of the transmission line.

For an `nphase` line, the first `(4+3*nphase+nphase^2)` columns are used. For example, for a three-phase line, 22 columns are used.

```
[nphase, no_I, no_U, long, L/Xl,  Zc,  Rm,  speed,  Ti]
```

- nphase: Number of phases of the transmission line.
- no_I: Input number in the source matrix corresponding to the first current source Is_1 of the line model. Each line model uses 2*nphase current sources specified in the source matrix as follows:

  Is_1 Is_2...Is_nphase for the sending end followed by

  Ir_1 Ir_2...Ir_nphase for the receiving end.
- no_U: Output number of the state-space corresponding to the first voltage output Vs_1 feeding the line model. Each line model uses 2*nphase voltage outputs in the source matrix as follows:

  Vs_1 Vs_2...Vs_nphase for the sending end followed by

  Vr_1 Vr_2...Vr_nphase for the receiving end.
- long: Length of the line (km)
- Zc: Vector of the nphase modal characteristic impedances (ohms)
- Rm: Vector of the nphase modal series linear resistances (ohms/km)
- speed: Vector of the nphase modal propagation speeds (km/s)
- Ti: Transformation matrix from mode to phase currents such that Iphase=Ti.Imod. The nphase*nphase matrix must be given in vector format [col_1, col_2,... col_nphase].

Format of the YOUT Matrix

The desired outputs are specified by a string matrix yout. Each line of the yout matrix must be an algebraic expression containing a linear combination of states and state derivatives specified according the following format:

- Uc_bn: Capacitor voltage of branch #n
- Il_bn: Inductor current of branch #n
- dUc_bn,dIl_bn: Derivative of Uc_bn or Il_bn
- Un, In: Source voltage or current specified by line #n of the source matrix
- U_nx1_x2: Voltage between node x1 and node x2
- I_bn: Current in branch #n. For a parallel RLC branch, I_bn corresponds to the total current IR+IL+IC
- I_bn_nx: Current flowing into node x of a pi transmission line specified by line #n of the rlc matrix. (This current includes the series inductive branch current and the capacitive shunt current).

To form a valid MATLAB expression, each output expression is built from voltage and current variable names defined above, their derivatives, constants, other variable names, parenthesis and operators (+-*/^). For example:

```
yout =
str2mat(['R1*I_b1+Uc_b3-L2*dIl_b2','U_n10_20','I2+3*I_b5']);
```

If variable names are used (as R1 and L2 in the above example), their names and values must be specified by the two input arguments vary_name and vary_val.

Sign Conventions for Voltages And Currents

I_bn Il_bn, In: Branch current, inductor current of branch #n or current of source #n is oriented from node1 to node2.

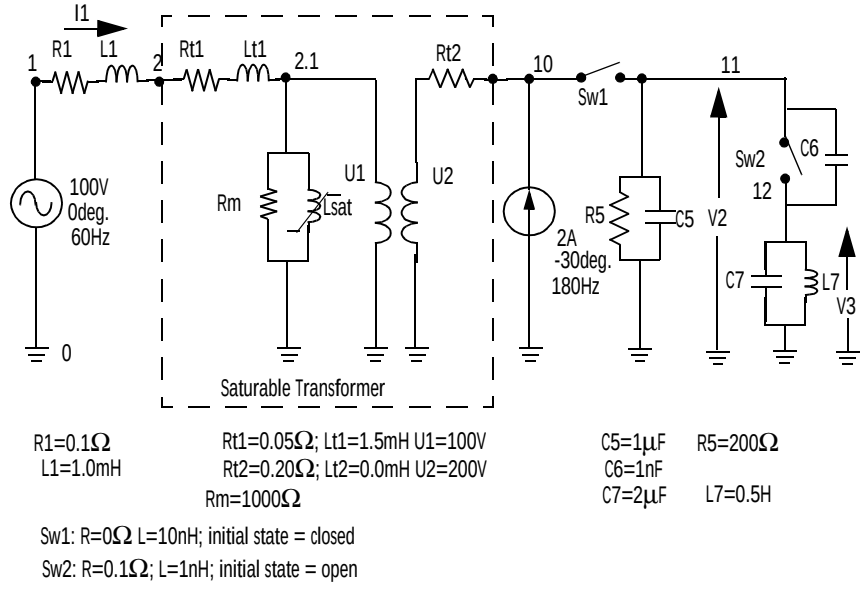
I_bn_nx: Current at one end (node x) of a pi transmission line. If x = node1, the current is entering the line. If x = node2, the current is going out of the line.

Uc_bn, Un: Voltage across capacitor or source voltage (Unode1 - Unode2)

U_nx1_x2: Voltage between nodes x1 and x2 = Ux1 - Ux2. Voltage of node x1 with respect to node x2.

Example

The following circuit consists of two sources (one voltage source and one current source), two series RLC branches (R1-L1 and C6), two parallel RLC branches (R5-C5 and L7-C7), one saturable transformer and two switches (Sw1 and Sw2). Sw1 is initially closed whereas Sw2 is initially open. Three measurement outputs are specified (I1,V2 and V3). This circuit has seven nodes numbered 0, 1, 2, 2.1, 10, 11 and 12. Node 0 is used for the ground. Node 2.1 is the internal node of the transformer where the magnetization branch is connected.



You can use the `circ2ss` function to find the state-space model of the linear part of the circuit. The nonlinear elements `Sw1`,`Sw2` and `Lsat` must be modeled separately by means of current sources driven by the voltage appearing across their terminals. Therefore you must foresee three additional currents sources and three additional voltage outputs for interfacing the nonlinear elements to the linear circuit.

You will obtain the state-space model of the circuit by entering the following commands in a .m executable file, available in the psbcirc2ss.m file.

```
unit='OMU' % specifies units : Ohms mH uF
rlc=[
%node1 node2   typeR   L   C(uF)/U(V)
1  2   0       0.1     1       0%R1 L1
2  0   2       0.05    1.5     100%transfo Wind.#1
10 0   2       0.20    0       200%transfo Wind.#2
2.10   1       1000    0       0%transfo mag.branch
11 0   1       200     0       1%R5 C5
11 12  0       0       0       1e-3%C6
12 0   1       0       500     2%L7 C7
];
source=[
%node1 node2   typeU/I     phasefreq
1  0   0       100         0   60 % Voltage source
0  10  1       2           -30 180% Current source
2.10   1       0           0   0% Saturation
10 11  1       0           0   0% Sw1
11 12  1       0           0   0% Sw2
];
switches=[
%node1 node2   status  R(ohm)  L(mH)I#U#
10     11      1       0       10e-6 42% Sw1
11     12      0       0.1     1e-6 53% Sw2
];
% outputs
%=========
y_u1='U_n2.1_0';   %U_sat= Saturable reactor voltage
y_u2='U_n10_11';   %U_Sw1= Voltage across Sw1
y_u3='U_n11_12';   %U_Sw2= Voltage across Sw2
y_i4='I_b1';       %I1 measurement
y_u5='U_n11_0';    %V2 measurement
y_u6='U_n12_0';    %V3 measurement
yout=str2mat(y_u1,y_u2,y_u3,y_i4,y_u5,y_u6); %outputs
y_type=[0,0,0,1,0,0]; %output type 0=voltage 1=current
[A,B,C,D,states,x0,x0sw,rlsw,u,x,y,freq,...
Asw,Bsw,Csw,Dsw,Hlin]=...
circ2ss(rlc,switches,source,[],yout,y_type,unit);
```

While circ2ss is executing, the following messages are displayed:

```
Computing state-space representation of linear electrical
circuit...
(4 states ; 5 inputs ; 6 outputs)
Oscillatory modes and damping factors:
F=159.115Hz zeta=4.80381e-008
Steady state outputs @ F=0 Hz :
y_u1= 0Volts
y_u2= 0Volts
y_u3= 0Volts
y_i4= 0Amperes
y_u5= 0Volts
y_u6= 0Volts

Steady state outputs @ F=60 Hz :
y_u1= 99.81Volts < -1.144 deg.
y_u2= 3.77e-006Volts < 93.17 deg.
y_u3= 199.4Volts < -1.148 deg.
y_i4= 2.099Amperes < 2.963 deg.
y_u5= 199.4Volts < -1.148 deg.
y_u6= 0.01652Volts < 178.9 deg.

Steady state outputs @ F=180 Hz :
y_u1= 11.4Volts < 53.48 deg.
y_u2= 1.324e-006Volts < 155.2 deg.
y_u3= 22.78Volts < 52.47 deg.
y_i4= 4.027Amperes < 146.5 deg.
y_u5= 22.83Volts < 52.47 deg.
y_u6= 0.0522Volts < 52.47 deg.
```

The names of the state variables are returned in the states string matrix:

```
states =

Il_b2_n2_2.1
Uc_b5_n11_0
Uc_b6_n11_12
Il_b7_n12_0
Il_b1_n1_2*
Uc_b7_n12_0*
```

Although this circuit contains six inductors and capacitors, there are only four state variables. The names of the state variables are given by the first four lines of the states matrix. The last two lines are followed by an asterisk indicating that these two variables are a linear combination of the state variables:

```
Uc_b7_n12_0 =  + Uc_b5_n11_0 - Uc_b6_n11_12
Il_b1_n1_2 =  + Il_b2_n2_2.1
```

The A,B,C,D matrices contain the state-space model of the circuit without nonlinear elements (sw1 switch open). The x0 vector contains the initial state values considering the switch sw1 closed. The Asw, Bsw, Csw, Dsw matrices contain the state-space model of the circuit considering the closed switch sw1. In our example, the system contains an extra state corresponding to the current flowing into the Sw1 inductor. The xosw vector contains the initial current in the closed switch.

```
A =
-4.0006e+005    0           0     0
-1.8998e-008   -4995        0    -499.25
 1.8998e-005   -4992.5      0     4.9925e+005
 0              2          -2     0
Asw =

-4.0006e+005    0           0     08e+005
-1.8998e-008   -4995        0    -499.259.99e+005
 1.8998e-005   -4992.5      0     4.9925e+0059.985e+005
 0              2          -2     0
 2e+011        -1e+008      0    -4.0002e+011
```

The system source frequencies are returned in the freq vector:

```
freq =

     0    60   180
```

The corresponding steady-state complex outputs are returned in the (6x3) y matrix where each column corresponds to a different source frequency.

For example, you can obtain the magnitude of the six voltage and current outputs at 60Hz as follows:

```
abs(y(:,2))

ans =

 99.808
  3.7696e-006
199.43
  2.0994
199.42
  0.01652
```

The initial values of the four state variables are returned in the x0 vector. You must use this vector in the State-Space block to start the simulation in steady-state.

```
x0 =

 2.3303
14.113
14.071
 3.1393e-005
```

The initial values of switch currents are returned in x0sw. To start the simulation in steady-state, you must use these values as initial currents for the nonlinear model simulating the switches.

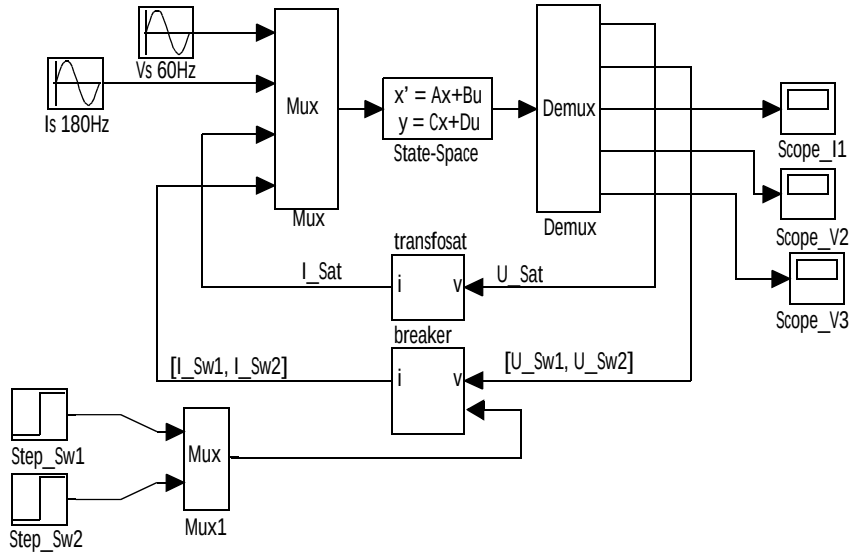
```
x0sw =

0.16155
0
```

The Simulink model of the circuit is shown in the following figure. If you had used the powerlib library to build your circuit, the same Simulink system would have been generated automatically by the `power2sys` function. The corresponding version of this circuit built with the Power System Blockset is available in the `psbcirc2ss_slk.mdl` file.

The linear part of the circuit is simulated by the State-Space block. Appropriate inputs and outputs are used to connect the switch and saturable reactance models to the linear system. You can find the breaker and transfosat

blocks in the powerlib_models library containing all the nonlinear models used by the Blockset. As the breaker model is vectorized, a single block is used to simulate the two switches `Sw1` and `Sw2`.



See Also

power2sys

Controlled Current Source

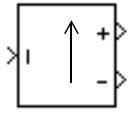
Purpose

Implement a controlled current source

Library

Electrical Sources Library

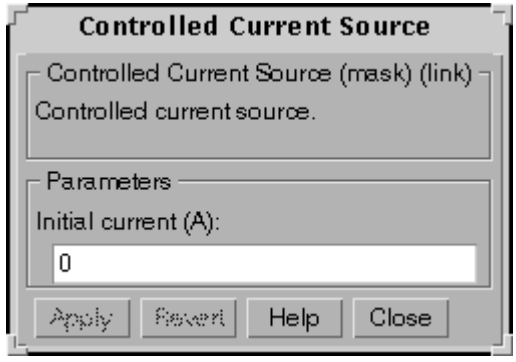
Description



The Controlled Current Source block provides a current source controlled by a Simulink signal. The positive current direction is as shown by the arrow in the block icon. The initial current level is set in the dialog box as a parameter.

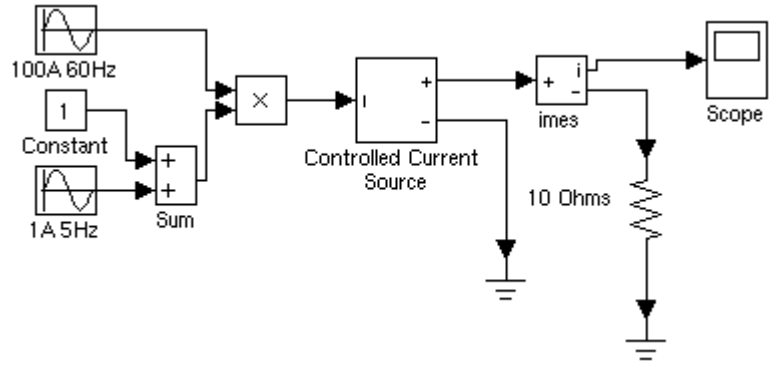
If the initial current is set to a nonzero value, this current will be considered as a DC current, and the states of the linear circuit will be initialized accordingly. To start the simulation in steady-state, the initial value of the Simulink signal must correspond to the initial current entered in the dialog box.

Dialog Box

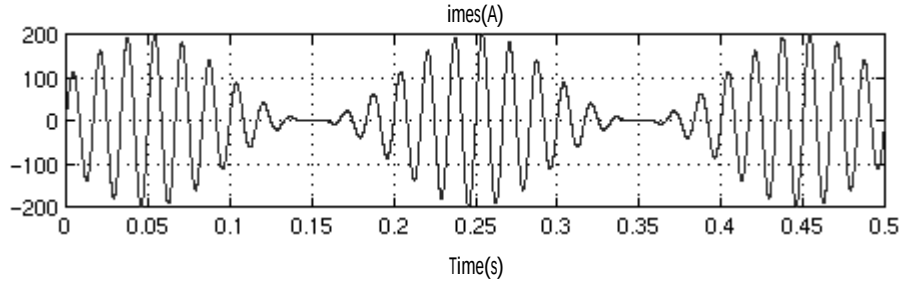


Example

Generate a 60Hz current modulated at 5Hz with the Controlled Current Source.



This circuit is available in the `psbcontrolcurr.mdl` file. Simulation produces the following waveforms:



See Also

Controlled Voltage Source

Controlled Voltage Source

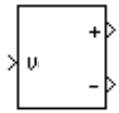
Purpose

Implement a controlled voltage source

Library

Electrical Sources Library

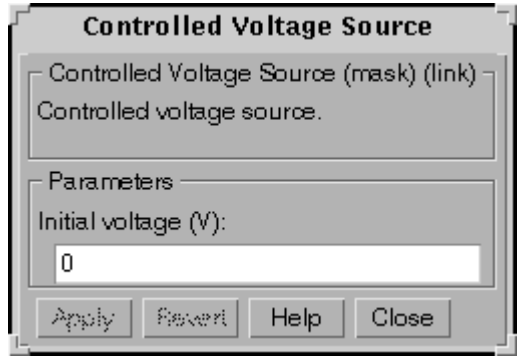
Description



The Controlled Voltage Source block provides a voltage source controlled by a Simulink signal. The first and second outputs of the block are respectively the positive and the negative terminals of the voltage source. The initial voltage level is set in the dialog box.

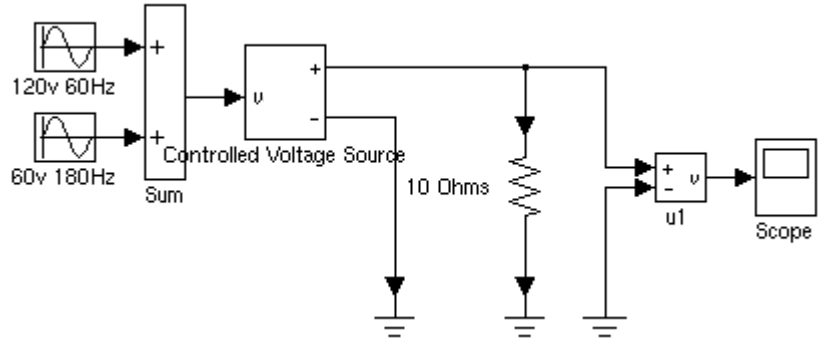
If the initial voltage is set to a nonzero value, this voltage will be considered as a DC voltage and the states of the linear circuit will be initialized accordingly. To start the simulation in steady-state, the initial value of the Simulink signal must correspond to the initial voltage entered in the dialog box.

Dialog Box

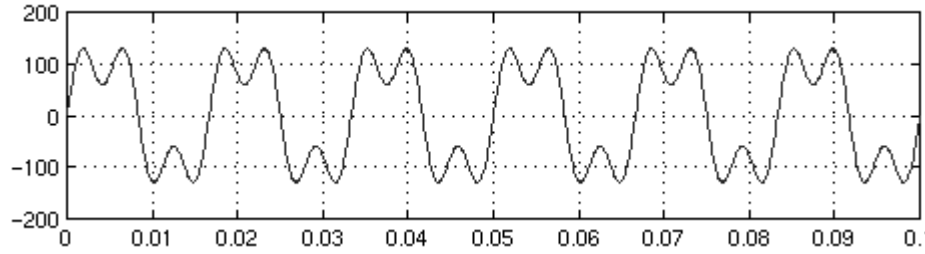


Example

Generate a 60 Hz sinusoidal voltage containing a third harmonic.



This circuit is stored in the file `psbcontrolvolt.mdl`. Simulation produces the following waveform



See Also

Controlled Current Source

Current Measurement

Purpose

Measure a current in a circuit

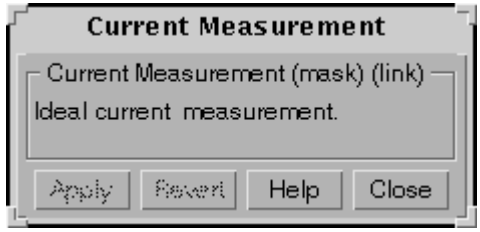
Library

Measurements Library

Description

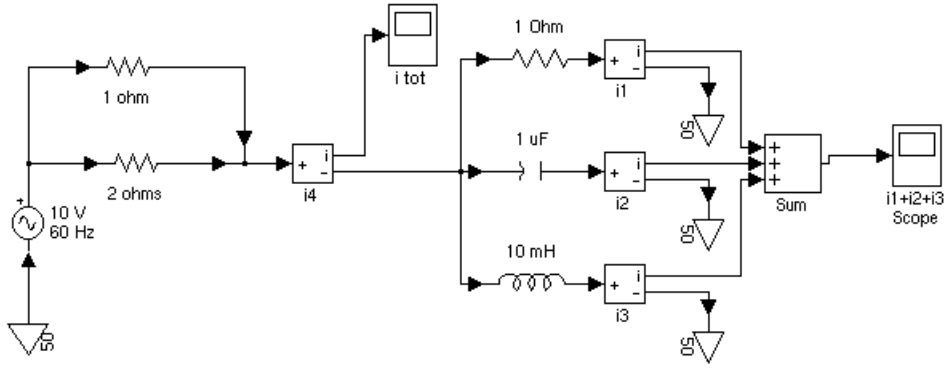
The Current Measurement block is used to measure the instantaneous current flowing in any electrical block or connection line. The first output provides a Simulink signal that can be used by other Simulink blocks.

Dialog Box



Example

Four Current Measurement blocks are used to read currents in different branches of a circuit. The two scopes display the same current. This circuit is available in the `psbcurrmeasure.mdl` file.



See Also

Voltage Measurement

Purpose

Implement a DC voltage source

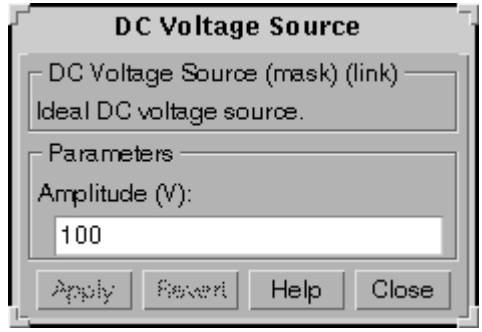
Library

Electrical Sources Library

Description

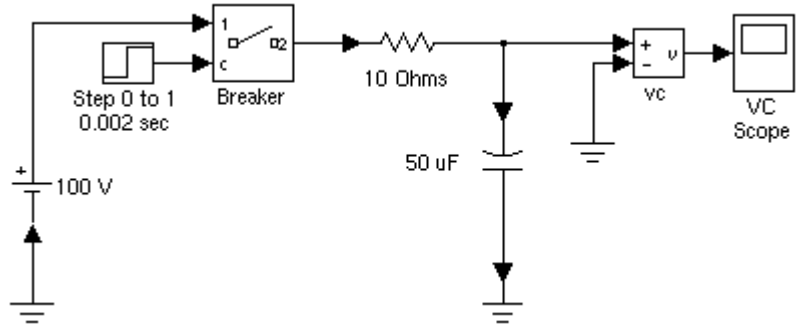
The DC Voltage Source block implements an ideal DC voltage source. The output and input are respectively the positive and negative source terminals. The voltage level is set in the dialog box. You can modify the voltage at any time during the simulation.

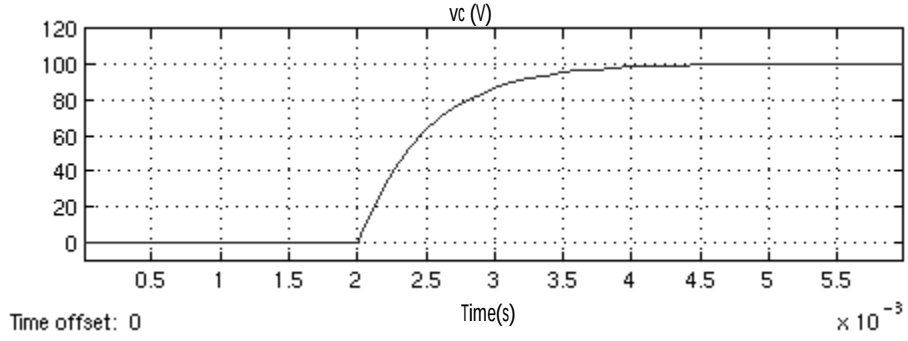
Dialog Box



Example

Simulation of the transient response of a first order RC circuit. This circuit is available in the `psbdcvoltage.mdl` file.





See Also **AC Voltage Source, Controlled Voltage Source**

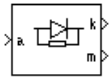
Purpose

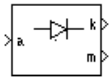
Implement a diode model

Library

Power Electronics Library

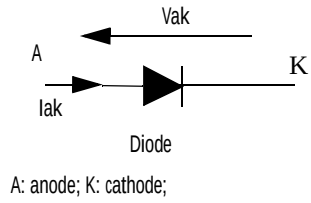
Description

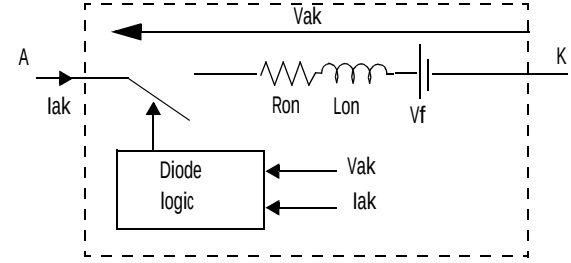




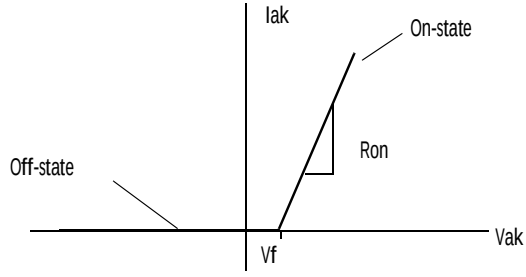
The Diode is a semiconductor device that is controlled by its own voltage and current. When the diode is forward biased (Vak>0), it starts to conduct with a small forward voltage across it. It turns off when the current flow into the device becomes zero. When the diode is reverse biased (Vak<0), it stays in the off-state.

The diode is modeled as resistor, inductor, and DC voltage source (Vf), connected in series with a switch. The switch is controlled by the voltage Vak and current Iak.

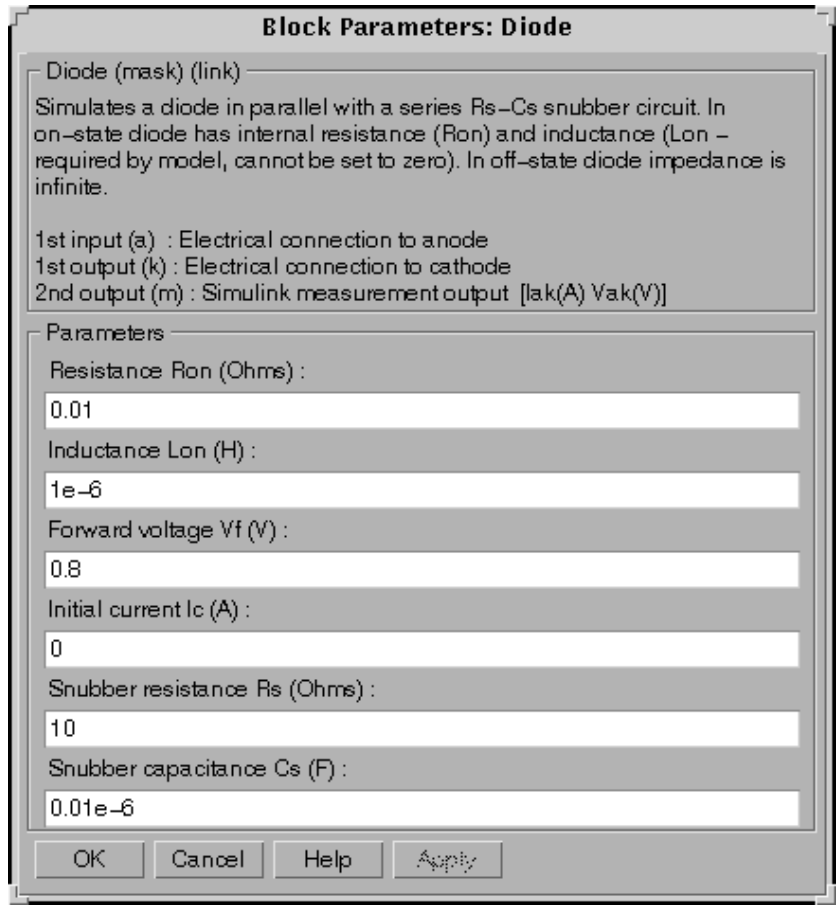




The Diode block also contains a series Rs-Cs snubber circuit, which is usually connected in parallel with the diode.The static VI characteristic of this model is shown in the figure below.



Dialog Box



Because of modeling constraints explained in the Assumptions and Limitations section, the inductance Lon cannot be set to zero. You can specify a snubber which is purely resistive (`Cs = Inf`) or purely capacitive (R_S=0). If you specify either `Rs=Inf` or `Cs=0`, the snubber is eliminated and it disappears on the diode icon.

The initial current, Ic, flowing in the diode, is usually set to zero so that the simulation is started with the diode blocked.

You may specify an initial current Ic value corresponding to a particular state of the circuit. In such a case, all states of the linear circuit must be set accordingly. Initializing all states of power-electronic converter is a complex task. Therefore, this option is useful only with simple circuits.

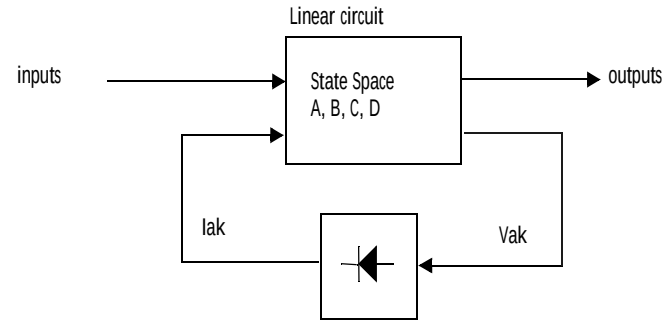
Inputs and Outputs

The first input and output are the diode terminals connected respectively to the anode (a) and the cathode (k). The second output (m) is a Simulink measurement output vector [Iak, Vak] returning the diode current and voltage.

Assumptions and Limitations

The Diode block implements a macro-model of the diode. It does not take into account either the geometry of the device or the complex physical processes underlying the state change [1]. The leakage current in the blocking state and the reverse-recovery (negative) current are not considered. In most circuits, the reverse current does not affect converter or other device characteristics.

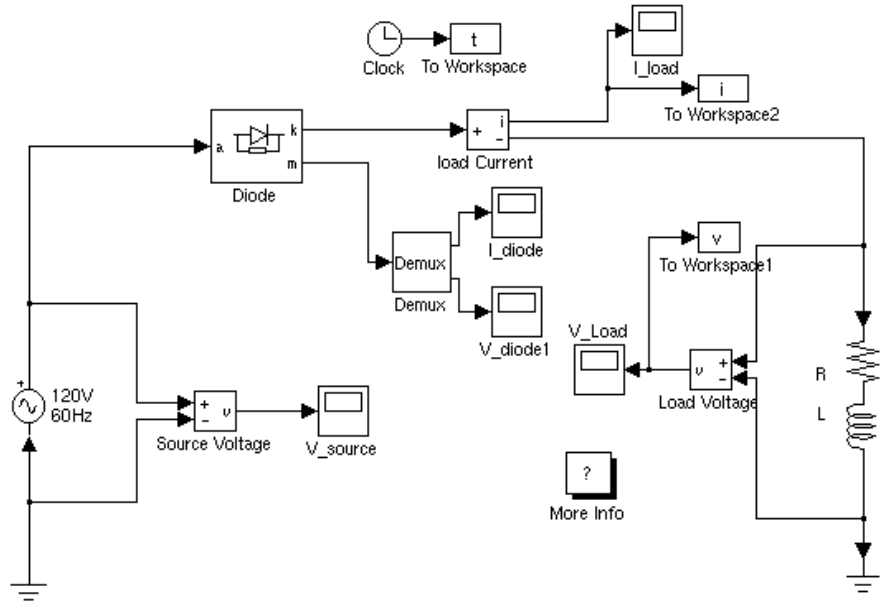
The diode is modeled as a nonlinear element interfaced with the linear circuit, as shown below.



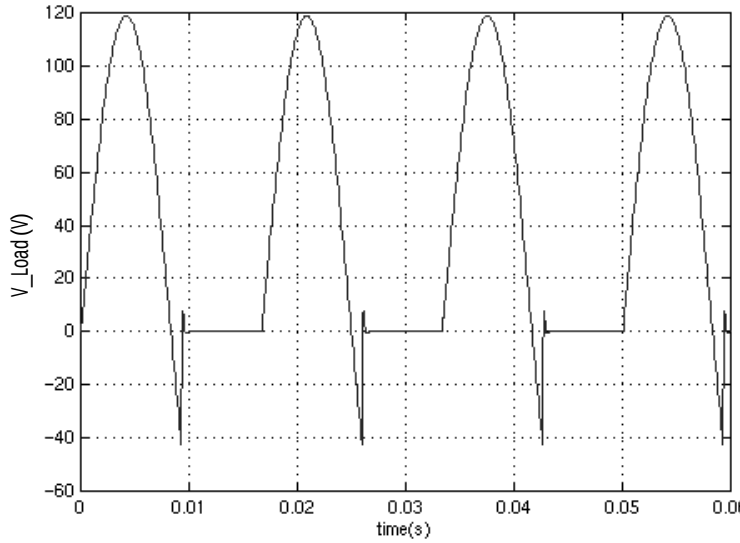
To avoid an algebraic loop in the Simulink representation, the diode inductance Lon cannot be set to zero. Each diode adds an extra state to the electrical circuit model. As the diode is modeled as a current source, it cannot be connected in series with an inductor, a current source, or an open circuit, unless a snubber circuit is used.

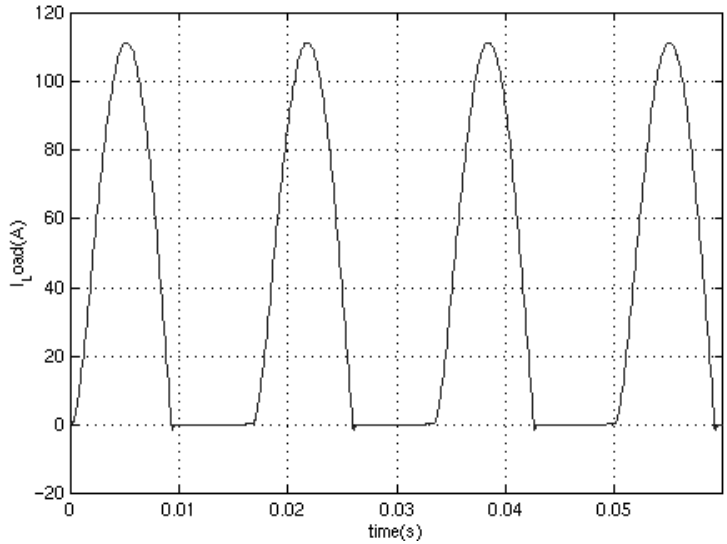
You must use a stiff integrator algorithm to simulate circuits containing diodes. `ode23tb` and `ode15s` usually give best simulation speed.

Example Single pulse rectifier consisting of a diode, RL load, and AC source.



This circuit is available in the `psbdiode.mdl` file. Simulation produces the following results.





References

[1] Rajagopalan V., *Computer-Aided Analysis of Power Electronic Systems*, Marcel Dekker, Inc., New York, 1987.

[2] Mohan N., *Power Electronic, Converters, Applications and Design*, John Wiley & Sons, Inc., New York, 1995.

See Also

Thyristor, **Mosfet**, **GTO**, **Ideal Switch**

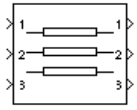
Purpose

Implement an N-phase distributed parameter transmission line model with lumped losses

Library

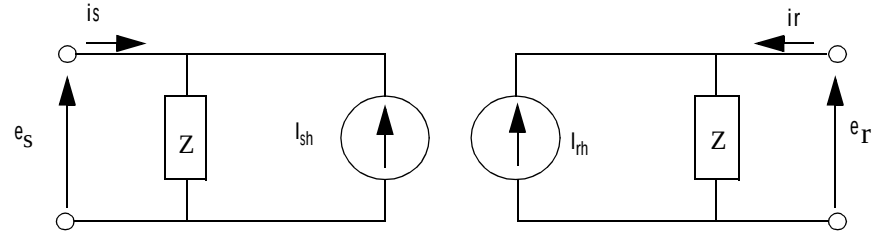
Elements Library

Description



The Distributed Parameter Line block implements an N-phase distributed parameter line model with lumped losses. The model is based on the Bergeron's traveling wave method used by the Electromagnetic Transient Program (EMTP)[1]. In this model, the lossless distributed LC line is characterized by two values (for a single phase line): the surge impedance $Z_c = \sqrt{L/C}$ and the phase velocity $v = 1/\sqrt{LC}$.

The model uses the fact that the quantity e+Zi, where e is line voltage and i is line current, entering one end of the line must arrive unchanged at the other end after a transport delay of $\tau = d/v$, where d is the line length. By lumping R/4 at both ends of the line and R/2 in the middle and using the current injection method of the Power System Blockset, the following two port models are derived:



$$I_{sh}(t) = \left(\frac{1+h}{2}\right)\left[\frac{1}{Z}e_r(t-\tau) + hi_r(t-\tau)\right] + \left(\frac{1-h}{2}\right)\left[\frac{1}{Z}e_s(t-\tau) + hi_s(t-\tau)\right]$$

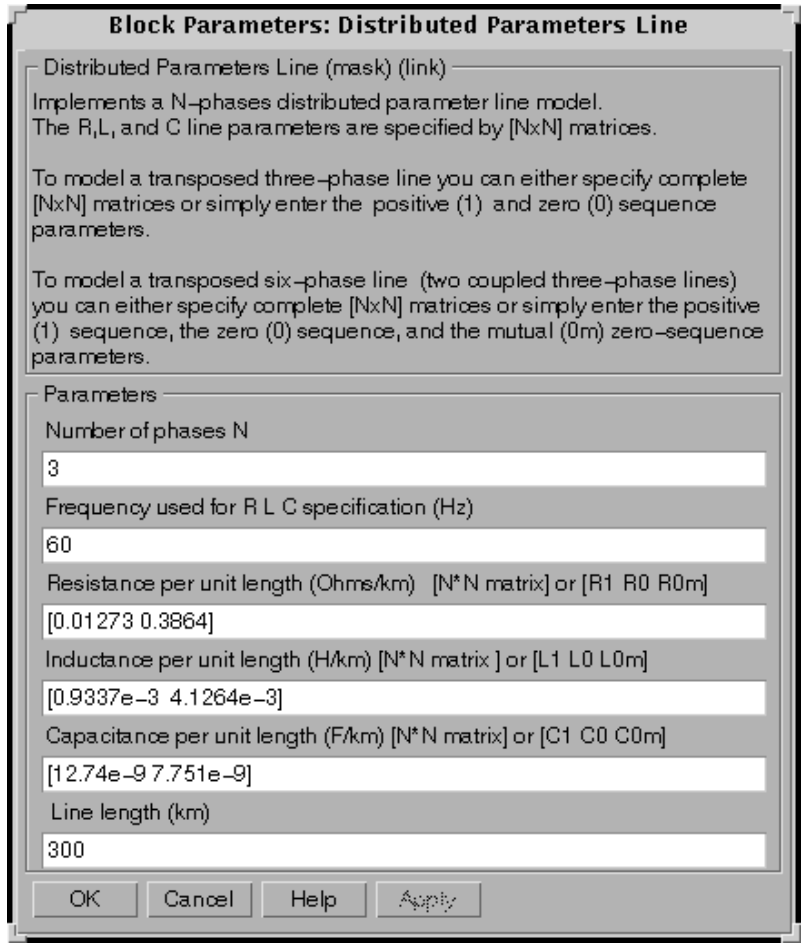
$$I_{rh}(t) = \left(\frac{1+h}{2}\right)\left[\frac{1}{Z}e_s(t-\tau) + hi_s(t-\tau)\right] + \left(\frac{1-h}{2}\right)\left[\frac{1}{Z}e_r(t-\tau) + hi_r(t-\tau)\right]$$

where $Z = Z_C + \frac{R}{4}$ $h = \frac{Z_C - \frac{R}{4}}{Z_C + \frac{R}{4}}$ $Z_C = \sqrt{\frac{L}{C}}$ $\tau = d\sqrt{LC}$

For multi-phase line models, modal transformation is used to convert line quantities from phase values (line currents and voltages) into modal values independent of each other. The previous calculations are made in the modal domain before being converted back to phase values.

In comparison to the pi sections line model, the distributed line represents wave propagation phenomena and line end reflections with much less error. See the comparison between the two models in the example section.

Dialog Box



The first entry in the dialog box specifies the number of phases of the model. The block icon dynamically changes according to the number of phases that you specify. When you apply the parameters or close the dialog box, the number of inputs and outputs is updated. The icon also displays the individual conductors. If you specify more than three phases, only one conductor is displayed.

The second entry specifies the frequency used to compute the line parameters. This entry is needed for the computation of the modal impedance and admittance matrices.

The third, fourth, and fifth entries are the R(ohms/km) L(H/km) and C(F/km) matrices. For unsymmetrical lines, you must specify the complete R L C matrices of the line.

If you want to model your line as a symmetrical line (continuously transposed), you can also specify the sequence parameters. This is permitted for the following types of lines: two-phases, three-phases transposed and six-phases transposed (double-circuit three-phase line with zero sequence coupling only between the two circuits). Therefore the two- and three-phase lines require two dimension vector entries [R1 R0] [L1 L0] [C1 C0] while the six-phase line requires three-dimension vector entries [R1 R0 R0m] [L1 L0 L0m] [C1 C0 C0m], where the subscripts 1,0, and 0m hold respectively for positive-sequence, zero-sequence, and mutual-zero-sequence.

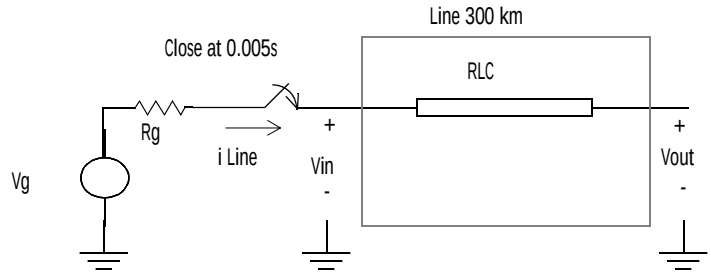
Finally, the last entry specifies the line length in Km.

Limitations

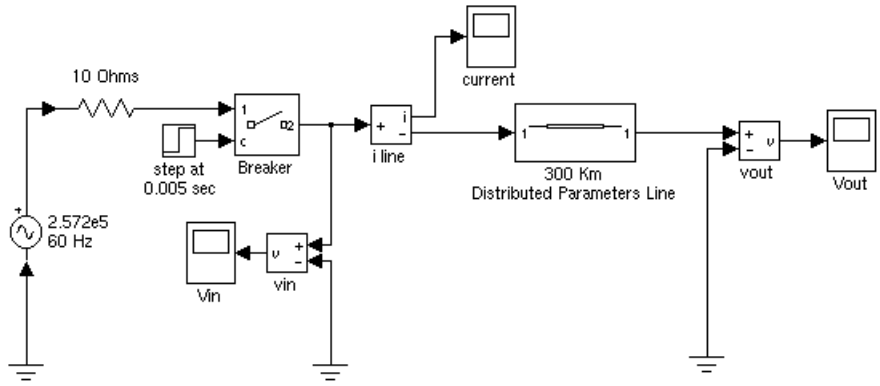
One limitation of this line model is its failure to represent accurately the frequency dependence of R L C parameters of real power lines. Indeed, because of skin effects in the conductors and ground, the R and L matrices exhibit strong frequency dependence, causing an attenuation of the high frequencies.

Example

Obtain the line energization voltages and current in the following circuit.

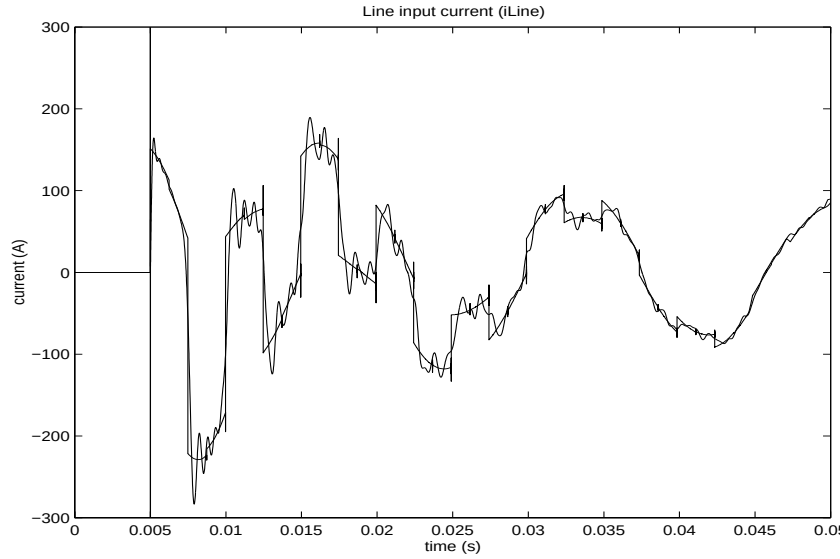


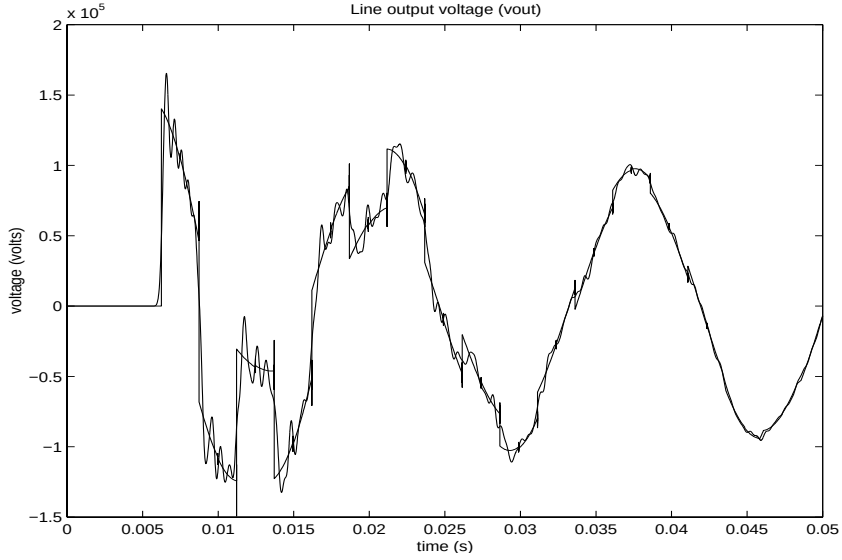
This circuit is available in the `psbdistline.mdl` file.



The sending end current and receiving end voltage obtained with the distributed parameter line are compared with a 10-pi-section line. On both graphs, the PI line model shows high frequency oscillatory modes superimposed on the 200Hz characteristic frequency of travelling waves. These oscillations due to PI sections are not found with the distributed parameter line with lumped losses model.

Note: Notice that the high current peak obtained with the PI line model at the breaker closing (due to the first section capacitor charging) does not exist with the distributed parameter line model.





References

[1] H. Dommel, “Digital Computer Solution of Electromagnetic Transients in Single and Multiple Networks,” *IEEE Transactions on Power Apparatus and Systems*, Vol PAS-88, No. 4, April 1969

See Also

PI Section Line

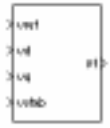
Purpose

Provide an excitation system for the synchronous machine and regulate its terminal voltage in generating mode

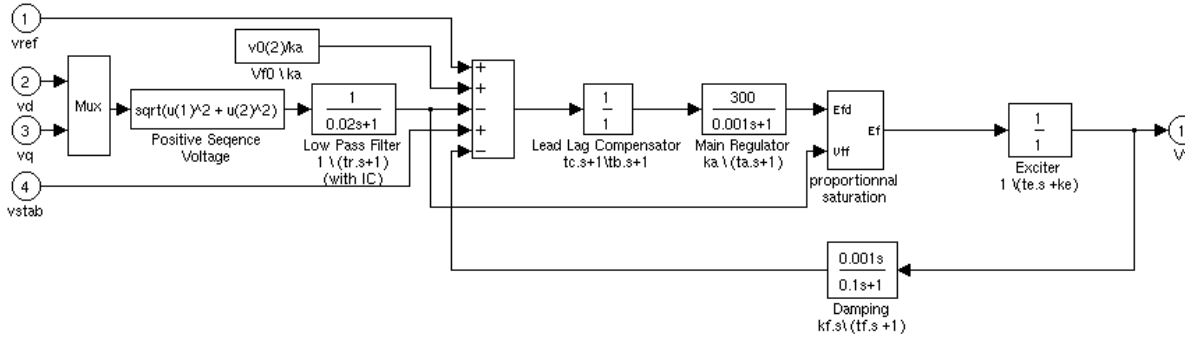
Library

Machines Library

Description



The Excitation System block is a Simulink system-implementing the DC exciter described in [1]. The basic elements that form the Excitation System block are the voltage regulator and the exciter. The voltage regulator consists of a main regulator with gain Ka and time constant Ta and a lead-lag compensator with time constants Tb and Tc. A derivate feedback is also provided with gain Kf and time constant Tf. The limits Efmin and Efmax are imposed to the output of the voltage regulator. The upper limit can be constant and equal to Efmax or variable and equal to the rectified stator terminal voltage Vtf times a proportional gain Kp. If Kp is set to zero, the former will apply. If Kp is set to a positive value, the latter will apply. The stator terminal voltage transducer is represented by a first-order low-pass filter with time constant Tr.



The exciter is represented by the following transfer function between the exciter voltage Vfd and the regulator's output ef:

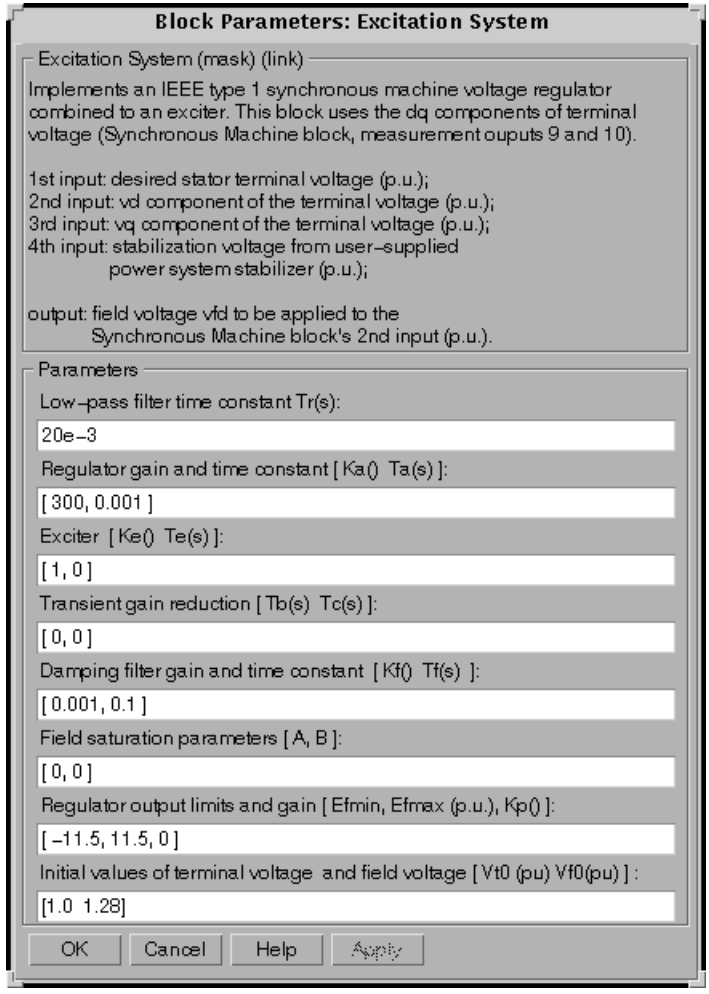
$$\frac{V_{fd}}{ef} = \frac{1}{Ke + sTe + S(Vfd)}$$

where $S(V_{fd})$ is a nonlinear function that represents the magnetic saturation of the exciter. This saturation function is given by:

$$S(Vfd) = Ae^{BVfd}$$

The last entry of the dialog box is used to specify the initial values of the terminal voltage and field voltage. The values used to initialize all states of the model allow you to start the simulation in steady-state.

Dialog Box



Inputs and Outputs

The first input of the block is the desired value of the stator terminal voltage. The following two inputs are the v_q and v_d components of the terminal voltage. The fourth input can be used to provide additional stabilization of power system oscillations. All inputs are in pu. The output of the block is the field voltage Vf for the Synchronous Machine block (p.u).

References

[1] IEEE "Recommended Practice for Excitation System Models for Power System Stability Studies". *IEEE Std*. 421.5-1992, August 1992

See Also

Hydraulic Turbine and Governor, **Synchronous Machine**

Ground

Purpose

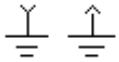
Provide a connection to the ground

Library

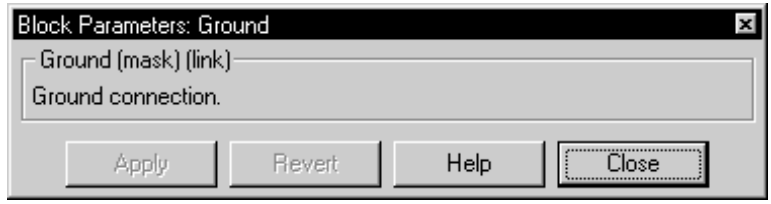
Connectors Library

Description

The Ground block implements a connection to the ground. For drawing facility, two types of Ground blocks are provided: one block with an input and one block with an output.

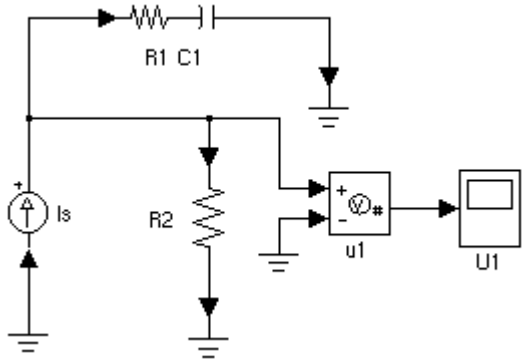


Dialog Box



Example

The following circuit shows an application of both types of Ground blocks. This circuit is available in the `psbground.mdl` file.



See Also

Neutral, **Bus Bar**

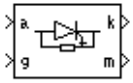
Purpose

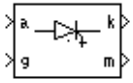
Implement a GTO-thyristor model

Library

Power Electronics Library

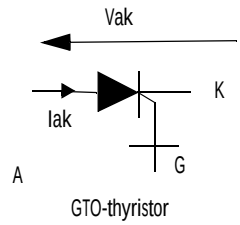
Description

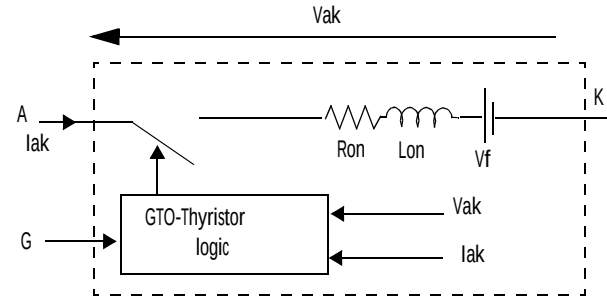




The Gate Turn-Off (GTO) thyristor block is a semiconductor device that can be turned on and off via a gate signal. Like a thyristor, the GTO-thyristor can be turned on by a positive gate signal (G>0). However, unlike the thyristor, which can be turned off only at a zero crossing of current, the GTO can be turned off by applying a gate signal equal to zero.

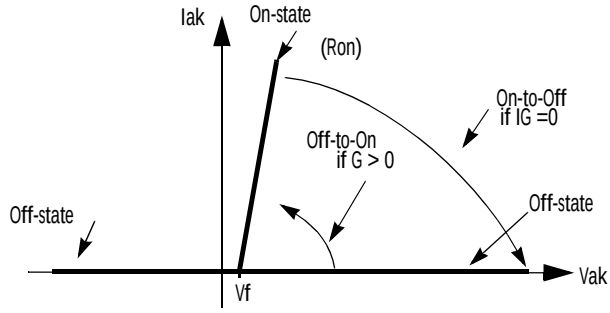
The model is simulated as a resistor Ro), inductor Lon, and DC voltage source (Vf) connected in series with a switch. The switch is controlled by a logical signal depending on the voltage Vak, current Iak, and the gate signal G.



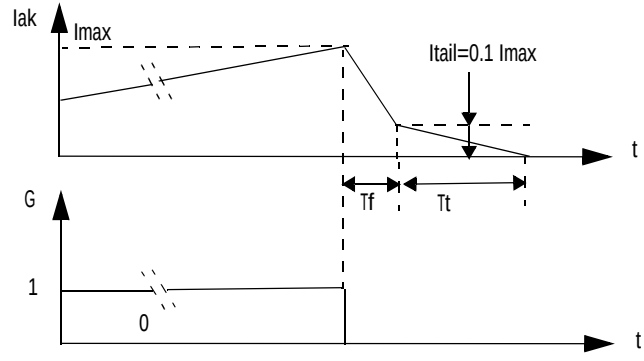


A: anode; K: cathode; G: gate

The GTO-thyristor turns on when the anode-cathode voltage is greater than Vf and a positive pulse signal is present at the gate input (G>0). When the gate signal is set to zero, the GTO starts to block, but its current does not stop instantaneously.



Since the current extinction process of a GTO-thyristor contributes significantly to the turn-off losses, the turn-off characteristic is built into the model. The current decrease is approximated by two segments. When the gate signal becomes zero, the current Iak first decreases from the value Imax (value of Iak when the GTO starts to open) to Imax/10, during the fall time (Tf), and then from Imax/10 to zero during the tail time (Tt). The GTO-thyristor turns off when the current Iak becomes zero. The latching and holding currents are not considered.



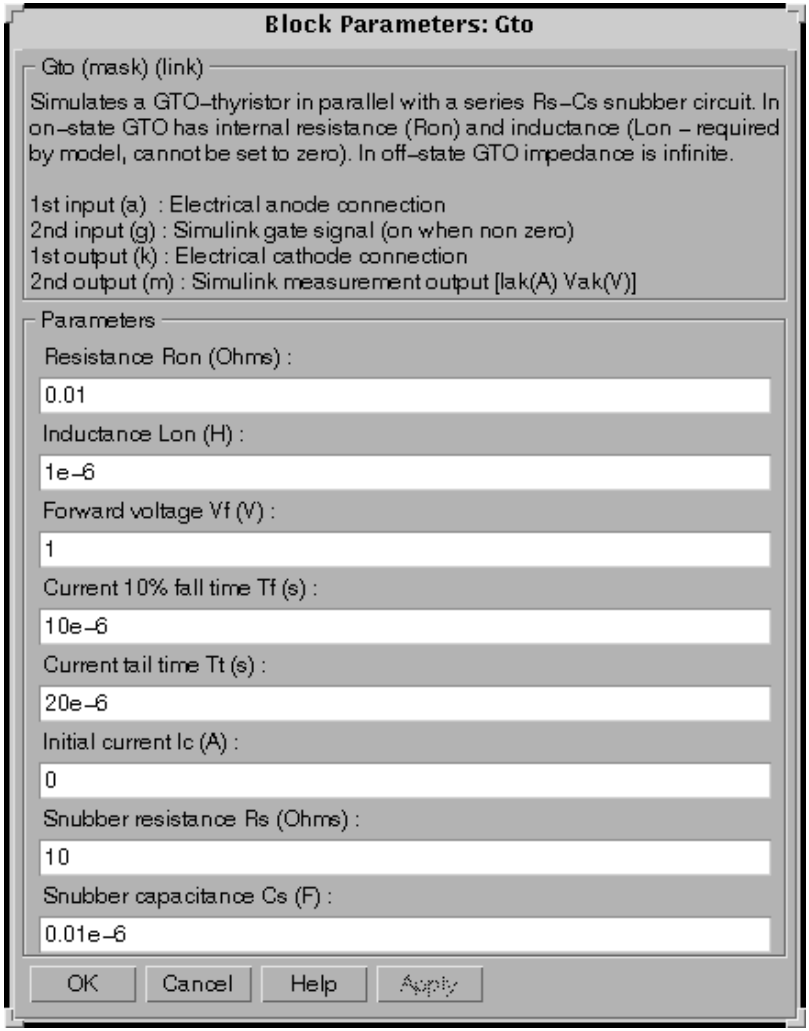
Vf, Ron and Lon are the forward voltage drop while in conduction, the forward conducting resistance, and the inductance of the device, respectively.

The GTO-thyristor block also contains a series Rs-Cs snubber circuit which is usually connected in parallel with the device.

Due to modeling constraints, the inductance Lon cannot be set to zero. You can specify a snubber which is purely resistive (`Cs = Inf`) or purely capacitive (`Rs=0`). If you specify either `Rs=Inf` or `Cs=0`, the snubber is eliminated and it disappears on the GTO-thyristor icon.

The initial current Ic is usually set to zero, so that the simulation is started with the GTO-thyristor blocked. However, you may specify an Ic value corresponding to a particular state of the circuit. In such a case, all states of the linear circuit must be set accordingly. Initializing all states of a power-electronic converter is a complex task. Therefore, this option is useful only with simple circuits.

Dialog Box



Inputs and Outputs

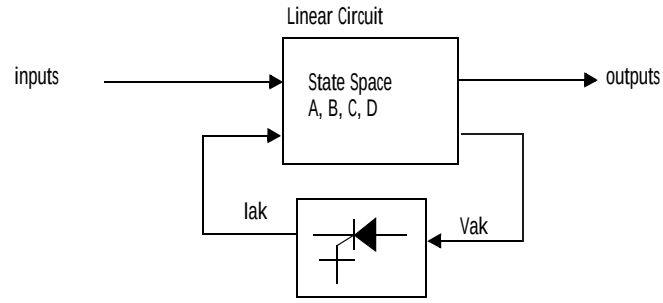
The first input and output are the GTO-thyristor terminals connected respectively to anode (a) and cathode (k). The second input (g) is a Simulink signal applied to the gate. The second output (m) is a Simulink measurement output vector [Iak, Vak] returning the GTO-thyristor current and voltage.

Assumptions and Limitations

The GTO-thyristor block implements a macro-model of a real GTO-thyristor, and it does not take into account either the geometry of the device or the underlying physical processes of the device [1].

The GTO-thyristor requires a continuous application of the gate signal (G>0) in order to be in the on-state (with Iak>0). The latching current and the holding current are not considered. The critical value of the derivative of the re-applied anode-cathode voltage is not considered.

The GTO-thyristor is modeled as a nonlinear element interfaced with the linear circuit, as shown below.



To avoid an algebraic loop, the GTO-thyristor inductance Lon cannot be set to zero. Each GTO-thyristor adds an extra state to the electrical circuit model. The GTO-thyristor is modeled as a current source. It cannot be connected in series with an inductor, a current source, or an open circuit, unless a snubber circuit is used.

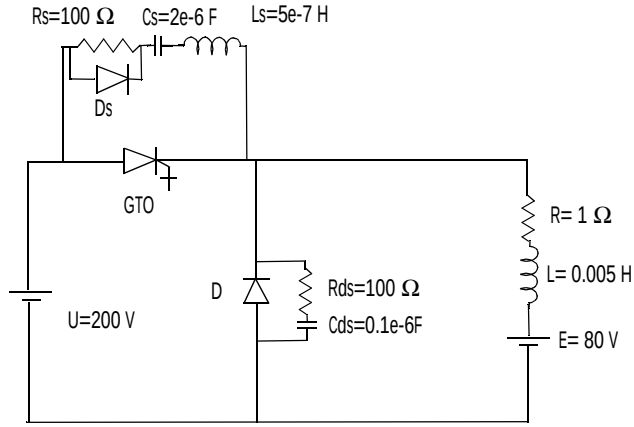
You must use a stiff integrator algorithm to simulate circuits containing GTO-thyristors. `ode23tb` and `ode15s` usually give best simulation speed.

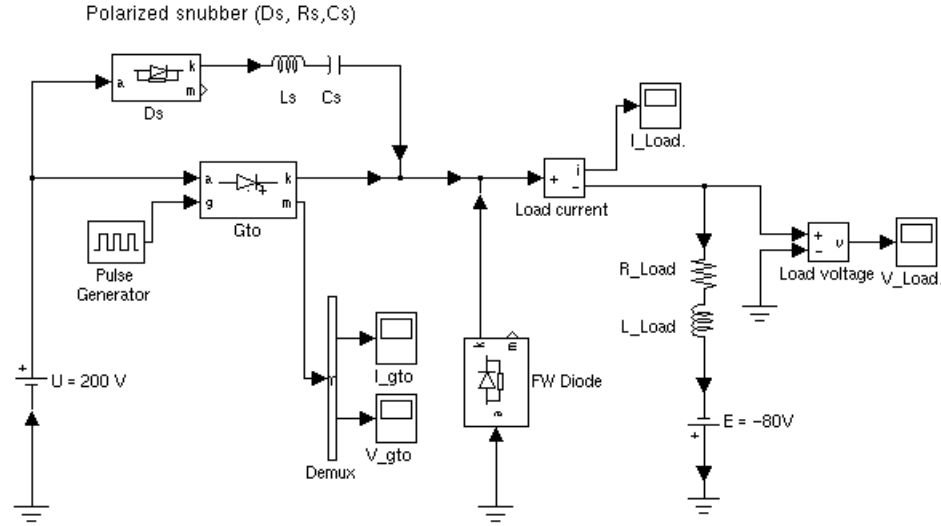
Example

The following example illustrates the use of the GTO-thyristor block in a buck converter topology. The basic polarized snubber circuit is connected across the GTO. The snubber circuit consists of a capacitor Cs, a resistor Rs, and a diode Ds. The parasitic inductance Ls of the snubber circuit is also taken into consideration.

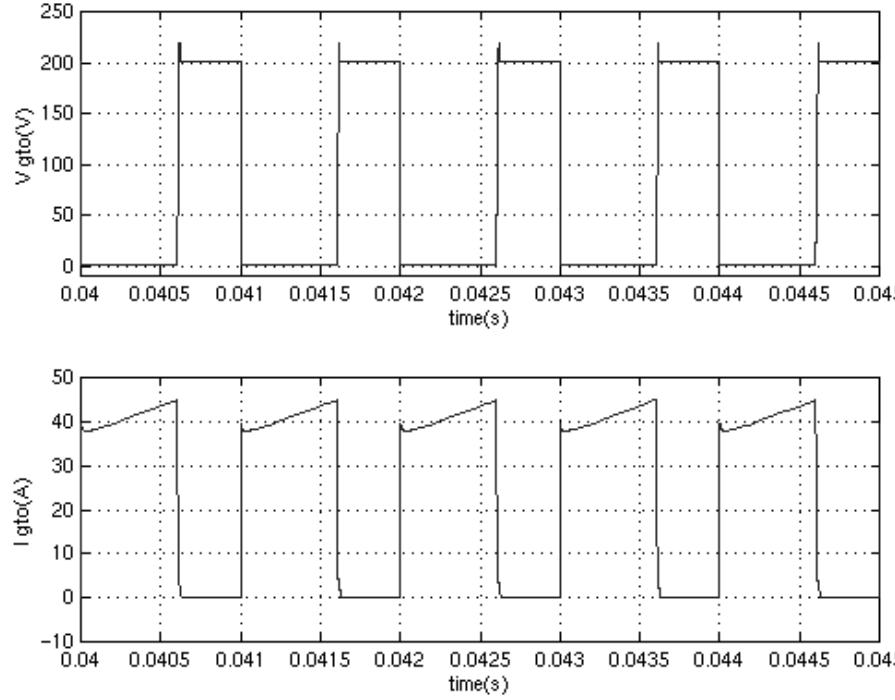
The parameters of the GTO are those found in the Dialog Box section, except for the internal snubber, which is out of service (`Rs=Inf Cs=0`). The switching

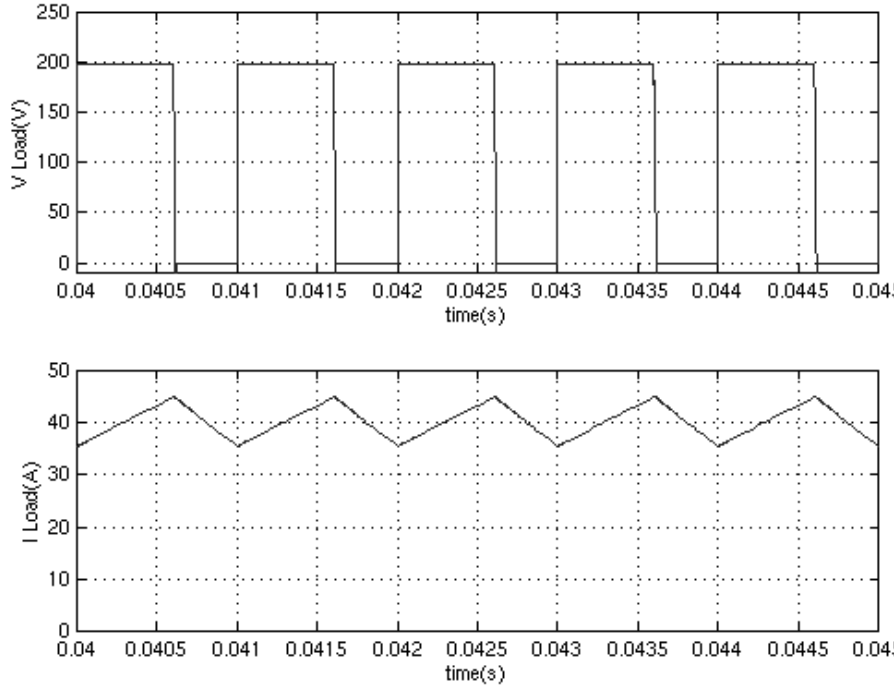
frequency is 1000 Hz and the pulse width is 216 degrees (duty cycle 60%). This example is available in the `psbbuckconv.mdl` file.





Run the simulation and observe the GTO voltage and current as well as the load voltage and current.





References

[1] Mohan N., *Power Electronic, Converters, Applications and Design*, John Wiley & Sons, Inc., New York, 1995

See Also

Diode, **Thyristor**, **Mosfet**, **Ideal Switch**

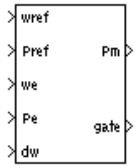
Purpose

Model a hydraulic turbine and a PID governor system

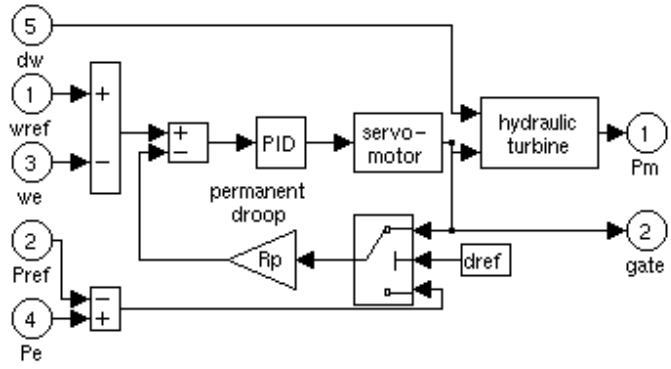
Library

Machines Library

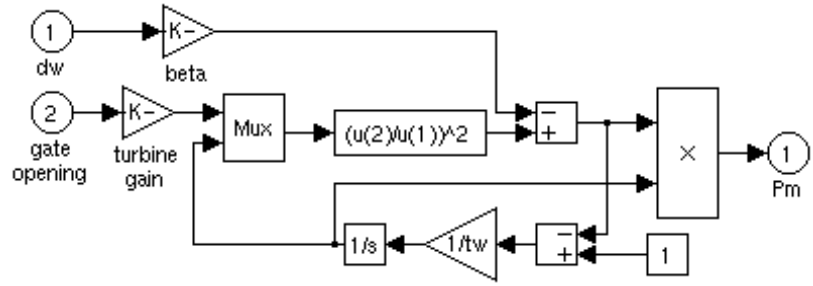
Description



The Hydraulic Turbine and Governor implements a hydraulic turbine model, a PID governor system, and a servo-motor. The static gain of the governor is equal to the inverse of the permanent droop Rp in the feedback loop. The input to this feedback loop can be selected to be the gate position or the electrical power deviation by setting the droop reference parameter in the dialog box to one or zero, respectively.



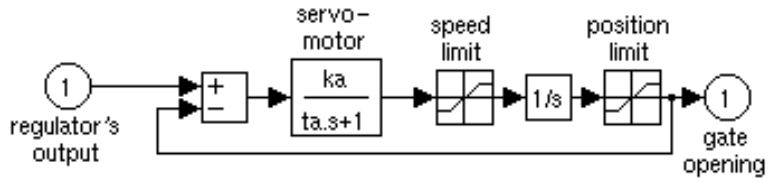
The hydraulic turbine is modeled by a nonlinear system with a water starting time Tw.



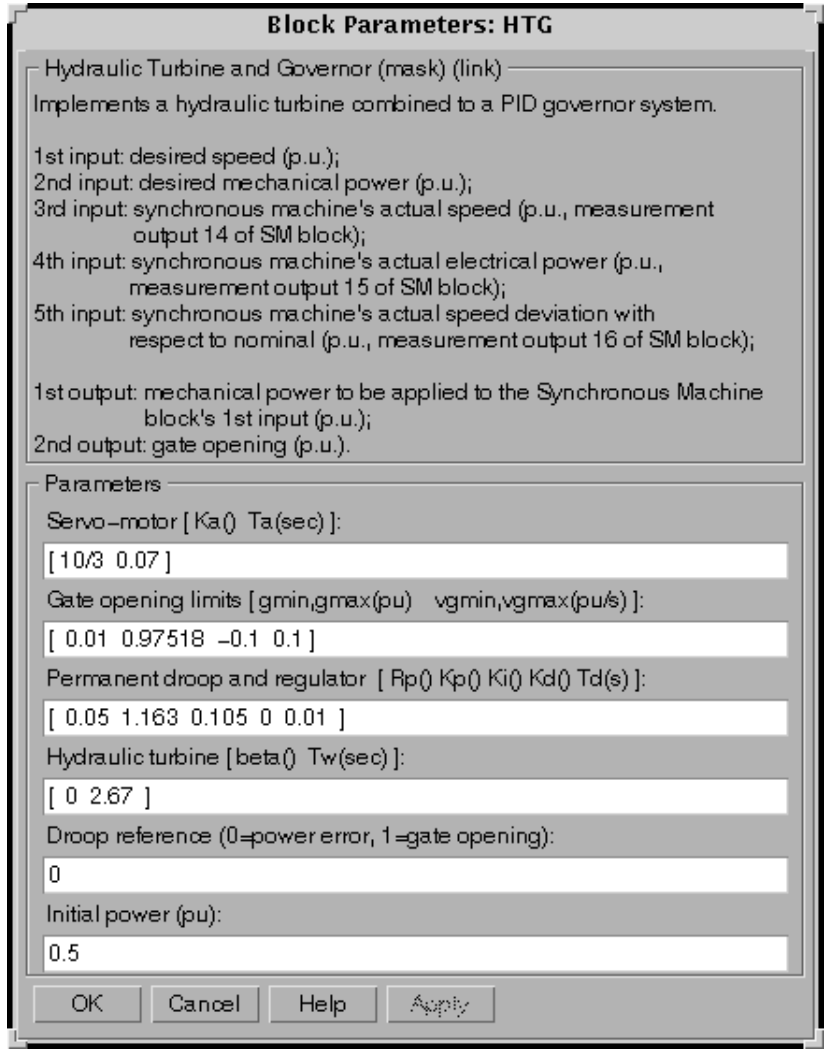
The PID regulator has a proportional gain Kp, an integral gain Ki and a derivative gain Kd. The high frequency gain of the PID is limited by a first-order low-pass filter with time constant Td.

The gate servo-motor is modeled by a second-order system with gain Ka and time constant Ta. The gate's opening is limited between gmin and gmax and its speed is limited between vgmin and vgmax.

The last entry of the dialog box is used to specify the initial output power. This value which is used to initialize all the states of the model allows you to start the simulation in steady-state.



Dialog Box


Block Parameters: HTG
Hydraulic Turbine and Governor (mask) (link)
Implements a hydraulic turbine combined to a PID governor system.
1st input: desired speed (p.u.);
2nd input: desired mechanical power (p.u.);
3rd input: synchronous machine's actual speed (p.u., measurement output 14 of SM block);
4th input: synchronous machine's actual electrical power (p.u., measurement output 15 of SM block);
5th input: synchronous machine's actual speed deviation with respect to nominal (p.u., measurement output 16 of SM block);
1st output: mechanical power to be applied to the Synchronous Machine block's 1st input (p.u.);
2nd output: gate opening (p.u.).
Parameters
Servo–motor [Ka() Ta(sec)]:
[10/3 0.07]
Gate opening limits [gmin,gmax(pu) vgmin,vgmax(pu/s)]:
[0.01 0.97518 –0.1 0.1]
Permanent droop and regulator [Rp() Kp() Ki() Kd() Td(s)]:
[0.05 1.163 0.105 0 0.01]
Hydraulic turbine [beta() Tw(sec)]:
[0 2.67]
Droop reference (0=power error, 1=gate opening):
0
Initial power (pu):
0.5
OK
Cancel
Help
Apply

Inputs and Outputs

The first two inputs are the desired speed and mechanical power. The third and fourth inputs are the machine's actual speed and electrical power. The fifth input is the speed deviation. Inputs 2 and 4 can be left unconnected if you want to use the gate position as input to the feedback loop instead of the power deviation. All inputs are in pu. The outputs of the block are mechanical power Pm for the Synchronous Machine block and gate opening (both in pu).

Example

See the Synchronous Machine block.

See Also

Excitation System, **Synchronous Machine**

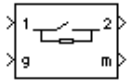
Purpose

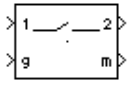
Implement an Ideal Switch model

Library

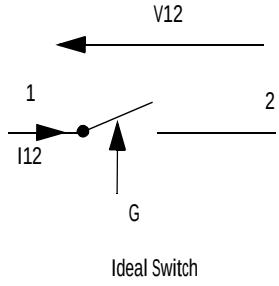
Power Electronics Library

Description

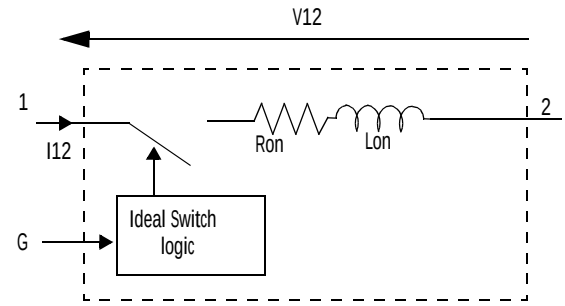




The Ideal Switch does not correspond to a particular physical device. When used with appropriate switching logic, it can be used to model a simplified semiconductor device such as a GTO or a Mosfet, or even a power circuit breaker with current chopping. The switch is simulated as a resistor (Ron) and inductor (Lon) in series with a switch controlled by a logical signal G.



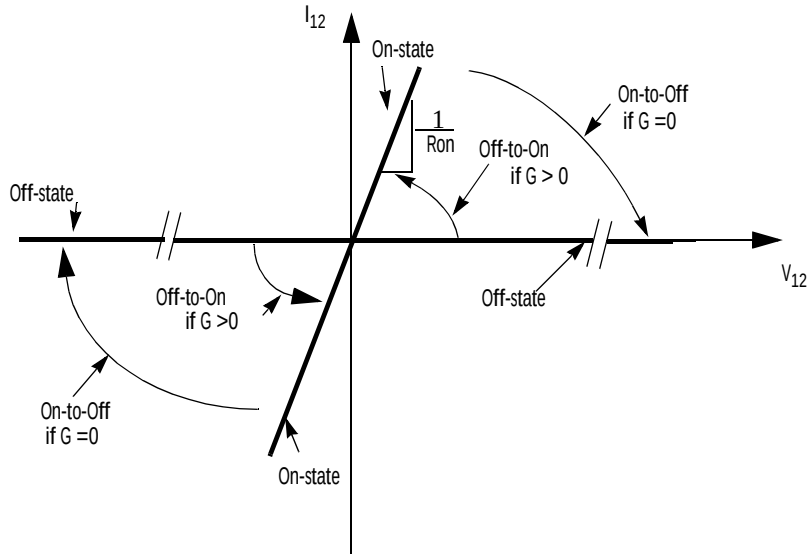
1: input; 2: output; G: gate



The Ideal Switch is fully controlled by the gate signal (G>0 or G=0). It has the following characteristics:

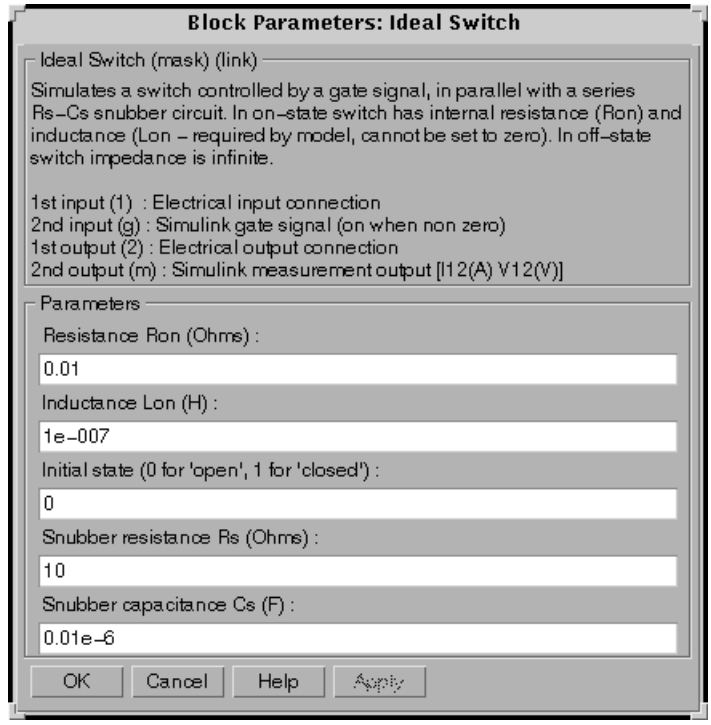
- Blocks any forward or reverse applied voltage with zero current flow when G=0
- Conducts any bidirectional current with quasi zero voltage drop when G>0
- Switches instantaneously between on and off state when triggered

The Ideal Switch turns on when a positive signal is present at the gate input (G>0). It turns off when the gate signal equals zero (G=0).



The Ideal Switch block also contains a series Rs-Cs snubber circuit, which is usually connected in parallel with the Ideal Switch. You can specify a snubber that is purely resistive (Cs = Inf) or purely capacitive (Rs=0). If you specify either Rs=Inf or Cs=0, the snubber is eliminated and it disappears on the Ideal Switch icon.

Dialog Box



Because of modeling constraints explained in the Assumptions and Limitations section, the inductance Lon cannot be set to zero. If the switch initial state is set to 1 (closed), the states of the linear circuit are automatically initialized so that the simulation starts in steady-state.

Inputs and Outputs

The first input (1) and output (2) are the Ideal Switch electrical connections. The second input (g) is a Simulink signal applied to the gate. The second output (m) is a Simulink measurement output vector [I_{12},V_{12}] returning the Ideal Switch current and voltage.

Assumptions and Limitations

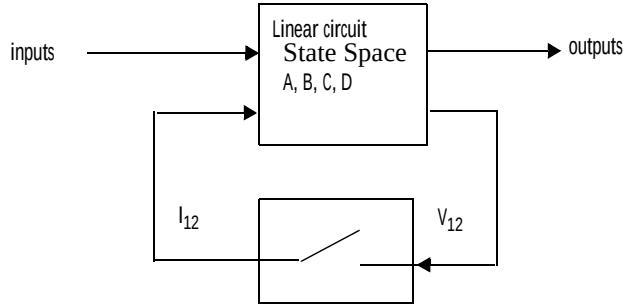
It can happen that the switch is connected across a capacitor. Depending on the values of the capacitor, switch resistance and inductance, high frequency poorly damped current and voltage oscillations can be produced when the switch closes if the Ron-Lon parameters are not properly selected. This will

result in a slow simulation speed. A good practice is to achieve a damping factor of z>0.5 for the Ron-Lon-C circuit. This condition is obtained when

$$Lon \le Ron^2 C$$

For example, with Ron =0.01 Ω and C= 1e-6 F, you should select Lon≅1e-10 H.

The Ideal Switch is modeled as a nonlinear element interfaced with the linear circuit, as shown below.



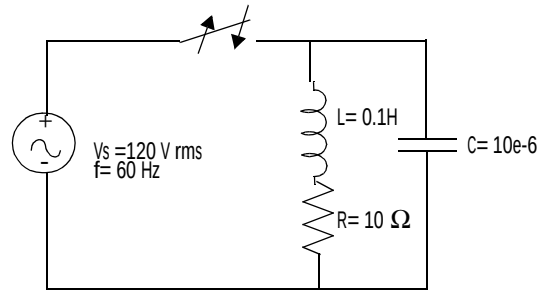
To avoid an algebraic loop, the Ideal Switch inductance Lon cannot be set to zero. Each Ideal Switch adds an extra state to the electrical circuit model. The Ideal Switch is modeled as a current source. It cannot be connected in series with an inductor, a current source or an open circuit, unless a snubber circuit is used.

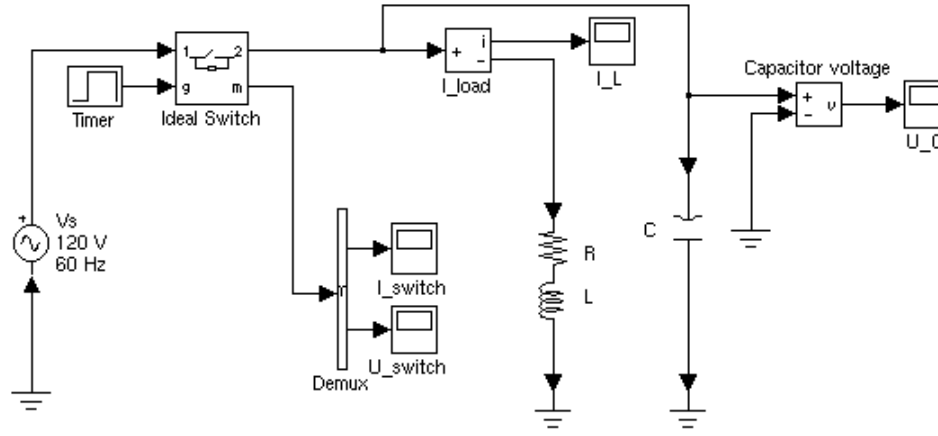
You must use a stiff integrator algorithm to simulate circuits containing Ideal Switches. `ode23tb` and `ode15s` usually give best simulation speed.

Example

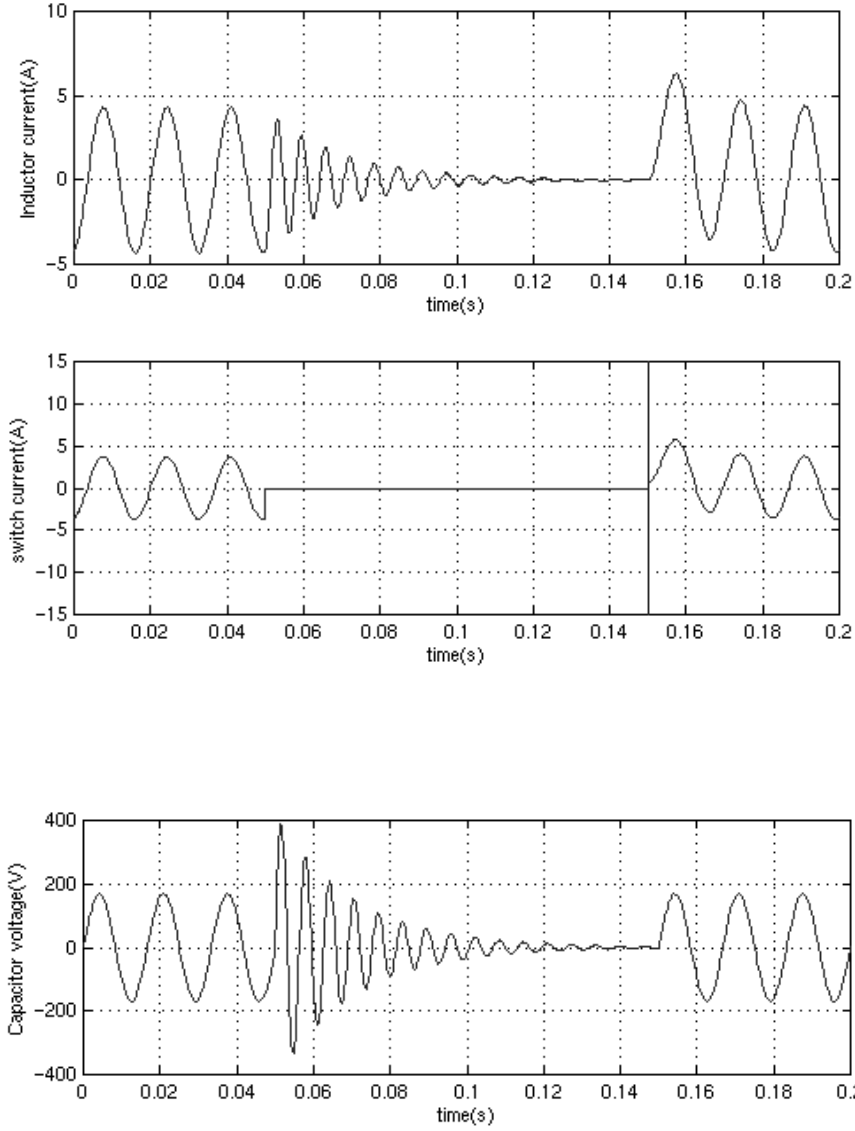
An Ideal Switch is used to switch an RLC circuit on an AC source (60 Hz). The switch, which is initially closed, is first open at t=36 ms and then closed at t=100 ms.

As the switch is used to switch a capacitor, its inductance Lon has been set to a very small value to satisfy the $Lon \le Ron^2 C$ condition. This example is available in the `psbswitch.mdl` file.





Run the simulation and observe the inductor current, the switch current and the capacitor voltage. Notice the high frequency overvoltage produced by inductive current chopping.



References

[1] N. Mohan, *Power Electronic, Converters, Applications and Design*, John Wiley & Sons, Inc., New York, 1995.

Purpose

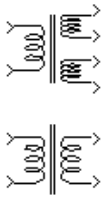
Implement a two- or three-winding linear transformer

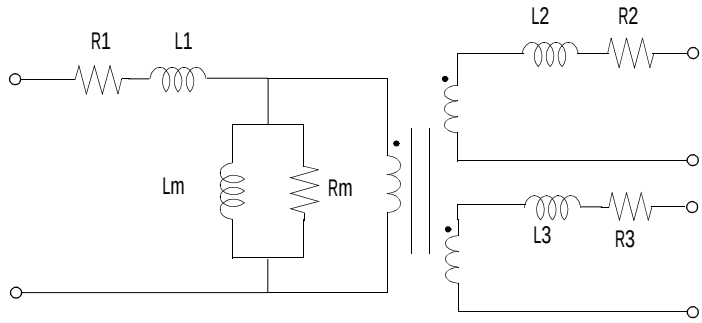
Library

Elements Library

Description

The Linear Transformer block model shown below consists of three coupled windings wound on the same core.





The model takes into account the winding resistances (R1 R2 R3), the leakage inductances (L1 L2 L3), as well as the magnetizing characteristics of the core which is modeled by a linear (Rm Lm) branch.

To comply with industry practice, you must specify the resistance and inductances per unit (pu) based on the transformer rated power (Pn in VA) and nominal voltage of the winding (V_n in V_{rms}). The base resistance and inductance are defined as follows:

$$Rbase = 1\ pu = \frac{(Vn)^2}{Pn}$$

$$Lbase = 1\ pu = \frac{Rbase}{2\pi f_n}$$

For example, for the default parameters of winding one specified in the dialog box section:

$$Lbase = \frac{720.3}{2\pi 60} = 1.91H$$

$$Rbase = \frac{(424.35 \times 10^3)^2}{250 \times 10^6} = 720.3\Omega$$

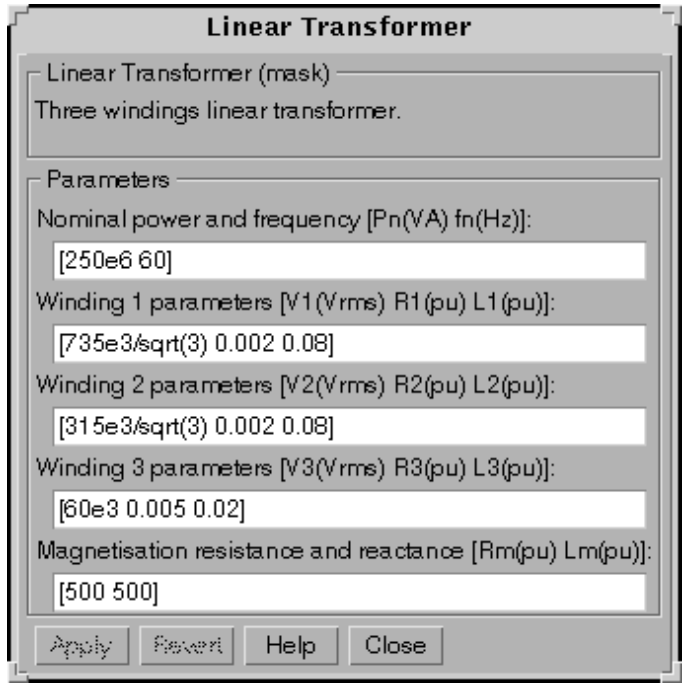
$$R_1 = 0.002pu \times 720.3\Omega = 1.44\Omega$$

$$L_1 = 0.08pu \times 1.91H = 0.1528H$$

$$R_m = 500pu \times 720.3\Omega = 3.6 \times 10^5\Omega$$

$$L_m = 500pu \times 1.91H = 955H$$

Dialog Box

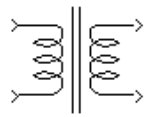


Specify in the first entry the nominal power rating and frequency of the transformer. Then specify in the following three entries the parameters of each

winding (nominal voltage in volts rms, resistance and leakage inductance in pu). Specify in the last entry the resistance and inductance simulating the core active and reactive losses both in pu.

Inputs and Outputs

Input one, output one, and output three (if it exists) are at the same instantaneous polarity. If you set the entry for the third winding to zero, the block will become a transformer with two windings and a new icon will be displayed:



Limitations

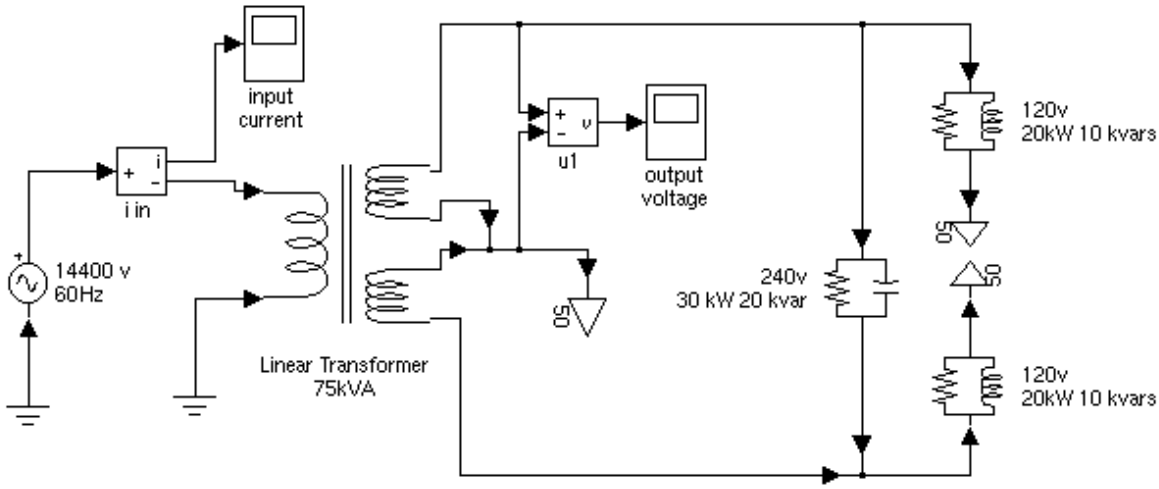
Because of modeling constraints the following restrictions apply: The winding resistances cannot be set to zero, however leakage inductances can be set to zero. Use values as small as necessary to simulate quasi-zero resistances. Similarly, the magnetizing resistance Rm must have a finite value but the reactive magnetizing losses can be set to zero by specifying Lm=Inf.

Windings can be left floating (i.e, not connected by an impedance to the rest of the circuit). The floating winding will be connected internally to the main circuit through a resistor. This invisible connection does not affect voltage and current measurements.

Example

Typical residential distribution transformer network feeding line to neutral and line to line loads. This circuit is available in the `psbtransformer.mdl` file.

Linear Transformer



See Also **Saturable Transformer, Mutual Inductance**

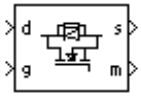
Purpose

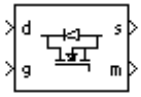
Implement a Mosfet model

Library

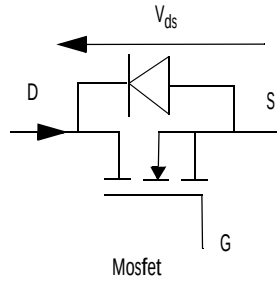
Power Electronics Library

Description

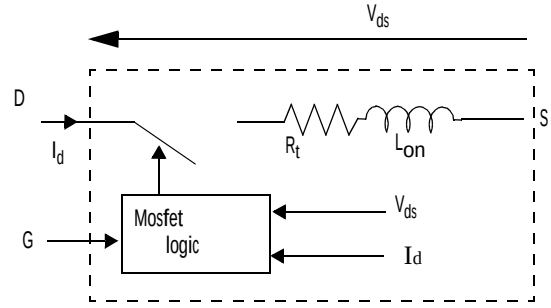




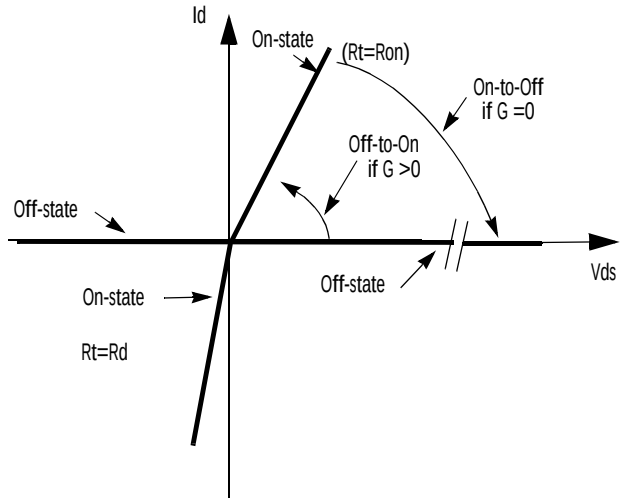
The Metal-Oxide-Semiconductor-Field-Effect-Transistor (Mosfet) block is a semiconductor device controllable by the gate signal (G > 0) if its current Id is positive (Id>0). The Mosfet block is connected in parallel with an internal diode which turns on when the Mosfet block is reverse biased (Vds < 0). The model is simulated as a series combination of a variable resistor (R_t) and inductor (L_{on}) in series with a switch controlled by a logical signal (G>0 or G=0).



D: drain; S: source; G: gate



The Mosfet block turns on when the drain-source voltage is positive and a positive signal is applied at the gate input (G >0).



With a positive current flowing through the device, the Mosfet block turns off when the gate input becomes zero. If the current Id is negative (Id flowing in the internal diode) and the gate signal is zero (G = 0), the Mosfet block turns off when the current Id becomes zero (Id = 0).

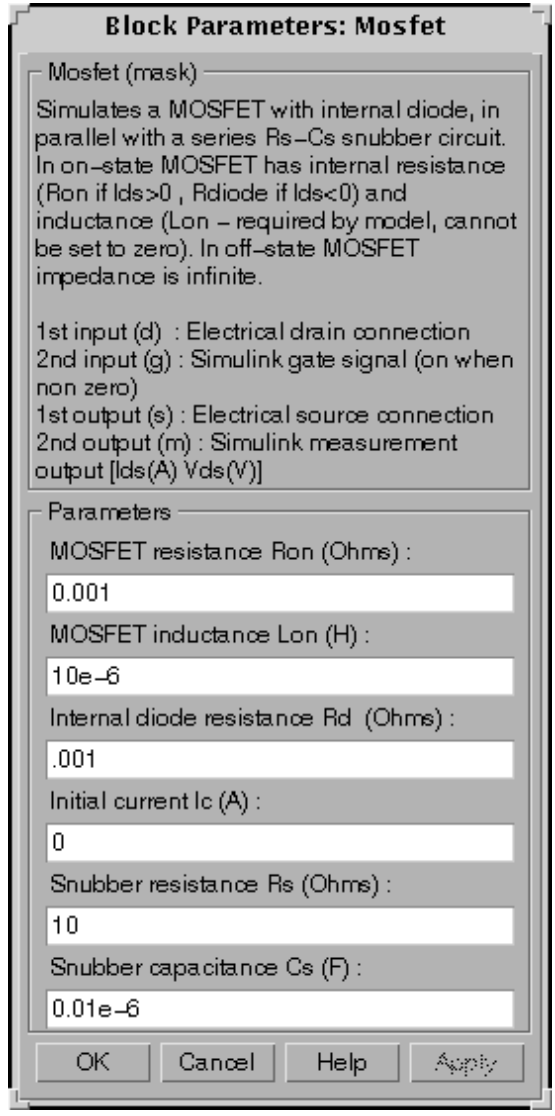
Note that the on-state resistance Rt depends on the drain current direction:

- Rt= Ron if Id > 0, where Ron represents the typical value of the forward conducting resistance of the Mosfet block
- Rt= Rd if Id < 0, where Rd represents the internal diode resistance

The Mosfet block also contains a series Rs-Cs snubber circuit, which is usually connected in parallel with the Mosfet block. You can specify a snubber that is purely resistive (Cs = Inf) or purely capacitive (Rs=0). If you specify either Rs=Inf or Cs=0, the snubber is eliminated and it disappears on the Mosfet icon

The initial current Ic flowing in the Mosfet block is usually set to zero, so that the simulation is started with Mosfet blocked. However, you may specify an Ic value corresponding to a particular state of the circuit. In such a case, all states of the linear circuit must be set accordingly. Initializing all states of a power-electronic converter is a complex task. Therefore, this option is useful only with simple circuits.

Parameters and Dialog Box



Inputs and Outputs

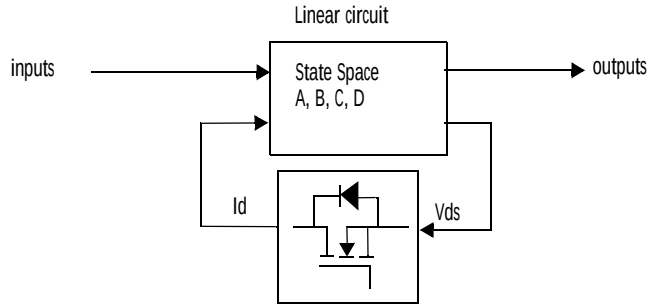
The first input and output are the Mosfet connection to drain (d) and source (s). The second input (g) is a logical Simulink signal applied to the gate. The second

output is a Simulink measurement vector [Id, Vds] returning the Mosfet current and voltage.

Assumptions and Limitations

The Mosfet block implements a macro-model of the real Mosfet device. It does not take into account either the geometry of the device or the complex physical processes [1].

In the Simulink representation, the Mosfet is modeled as a nonlinear element interfaced with the linear circuit as shown below.

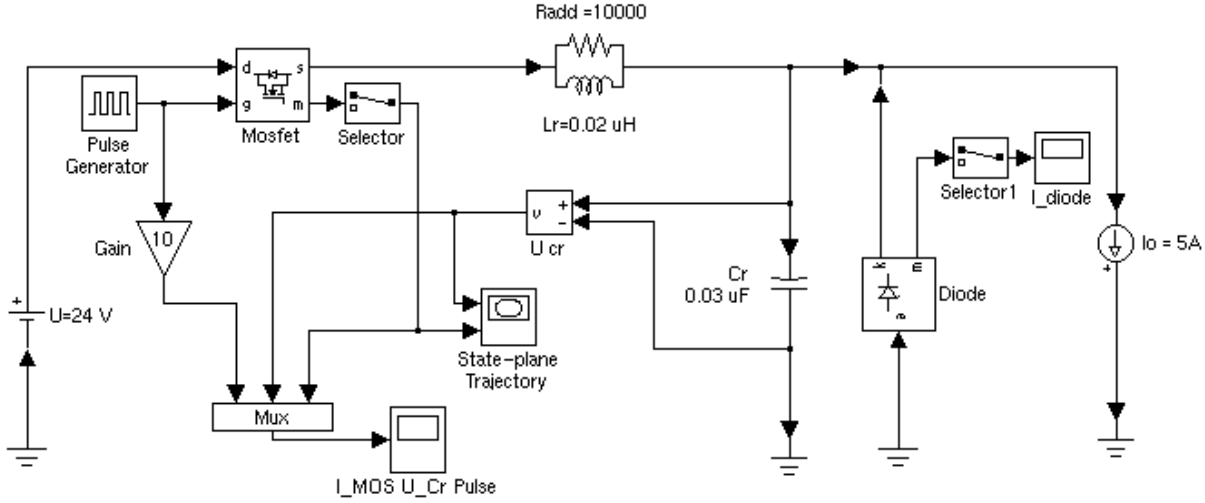


To avoid an algebraic loop, the Mosfet inductance Lon cannot be set to zero. Each Mosfet adds an extra state to the electrical circuit model. Since the Mosfet is modeled as a current source, it cannot be connected in series with an inductor, a current source, or an open circuit, unless a snubber circuit is used.

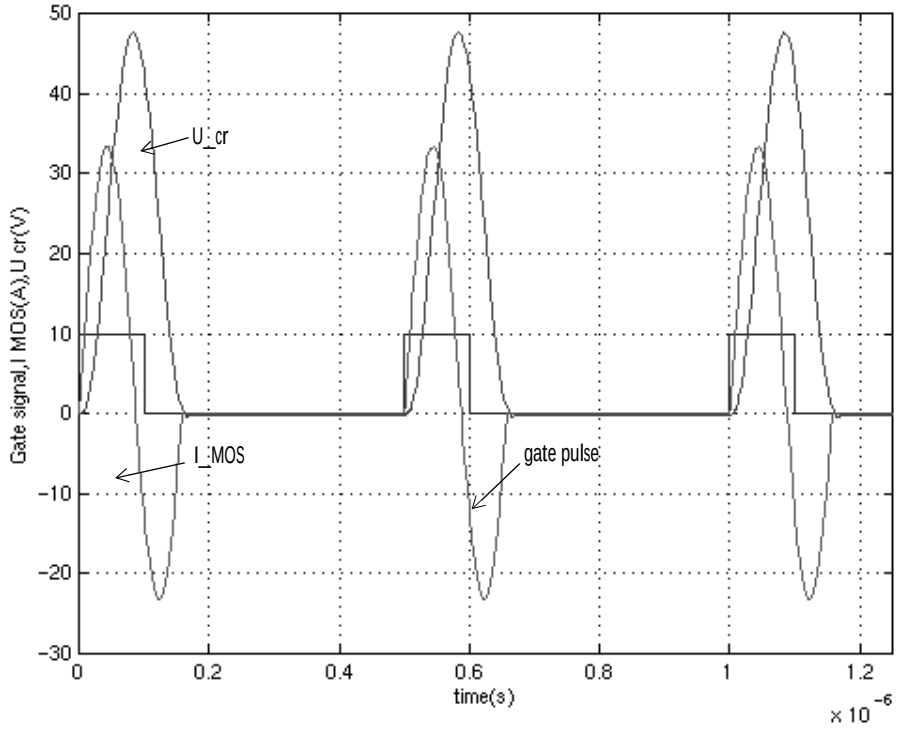
You must use a stiff integrator algorithm to simulate circuits containing mosfets. `ode23tb` and `ode15s` usually gives best simulation speed.

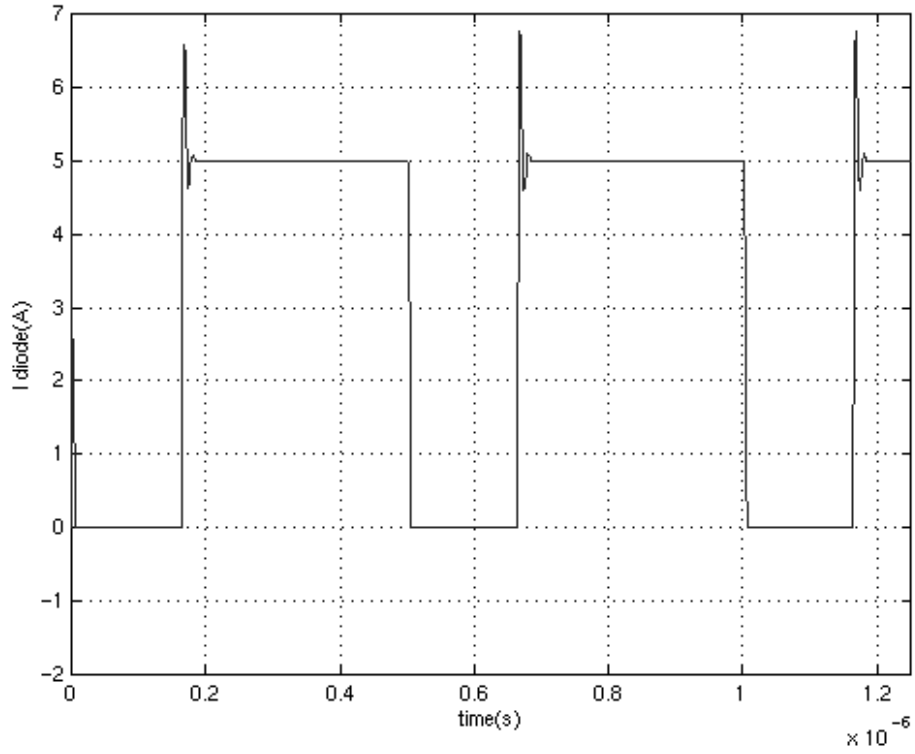
Example

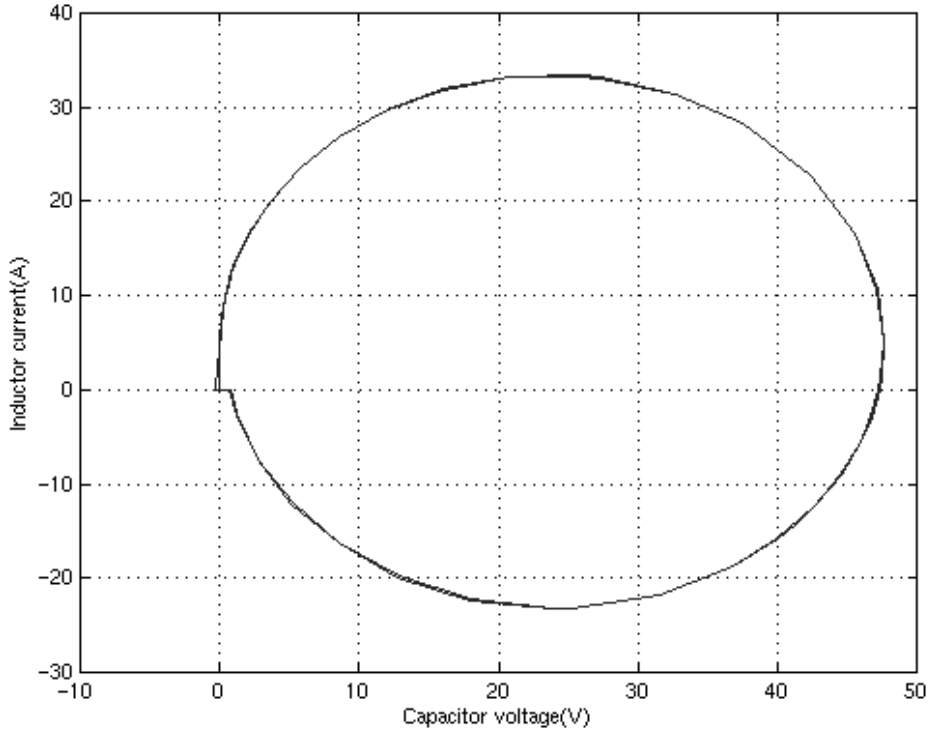
The following example illustrates the use of the Mosfet block in a Zero-Current-Quasi-Resonant Switch converter. In such a converter, the current produced by the Lr-Cr resonant circuit flows through the device, thus causing it to turn on and off at zero current [1]. The switching frequency is 2 MHz and the pulse width is 72 degrees (duty cycle: 20%). This example is available in the `psbmosconv.mdl` file



Run the simulation and observe the capacitor voltage, the mosfet current, the gate pulse signal, the diode current, and the state-plane trajectory (inductor current versus capacitor voltage).


Gate signal,I MOS(A),U cr(V)
time(s)
x 10^-6
U_cr
I_MOS
gate pulse
50
40
30
20
10
0
-10
-20
-30
0
0.2
0.4
0.6
0.8
1
1.2


I diode(A)
time(s)
x 10^-6
7
6
5
4
3
2
1
0
-1
-2
0
0.2
0.4
0.6
0.8
1
1.2



References

[1] Mohan N., *Power Electronic, Converters, Applications and Design*, John Wiley & Sons, Inc., New York, 1995.

See Also

Diode, **GTO**, **Ideal Switch**, **Thyristor**

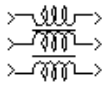
Purpose

Implement a magnetic coupling between two or three windings

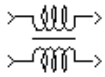
Library

Elements Library

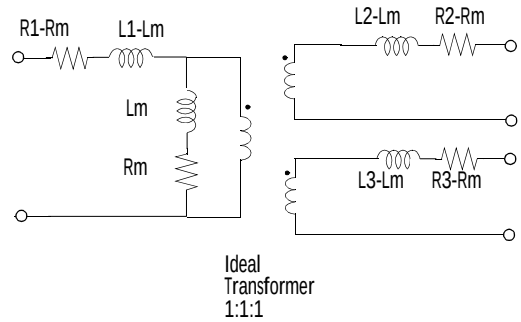
Description



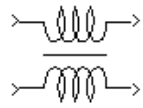
The Mutual Inductance block implements a magnetic coupling between three separate windings. Specify the self resistance and inductance of each winding on the first three entries of the dialog box and the mutual resistance and inductance in the last entry.



The electrical model for this block is given below



If the entry for the third winding is set to zero, the block represents the magnetic coupling between two inductances and a new icon is displayed

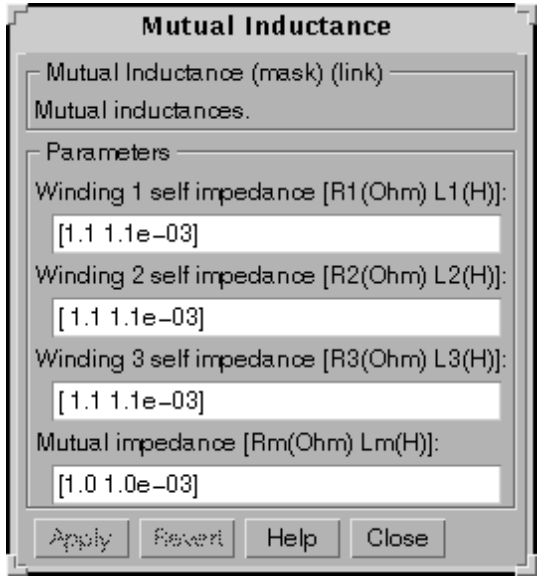


Inputs and Outputs

The inputs of the Mutual Impedance block are at the same instantaneous polarity.

Mutual Inductance

Dialog Box



Limitations

Due to modeling constraints, the following restrictions apply:

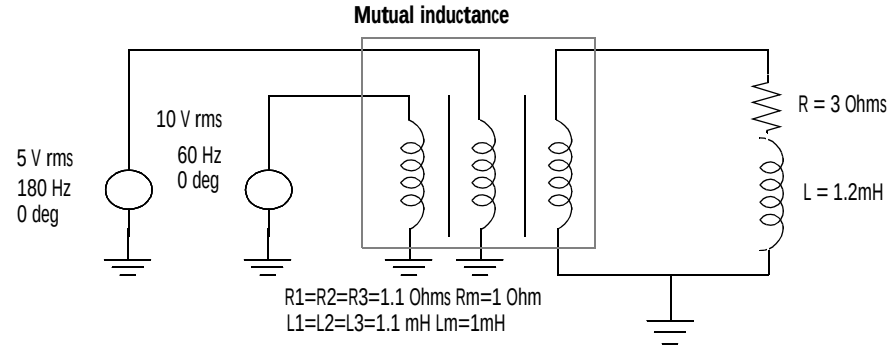
$$Rs > 0,\ Rs > Rm$$
$$Lm \neq 0,\ Ls \neq Lm$$

The winding resistances (self resistances) must be positive and greater than the mutual resistance. Negative values are allowed for the self and mutual inductances as long as the self inductances are different from the mutual inductance. The mutual inductance must be different from zero but a zero mutual resistance is allowed.

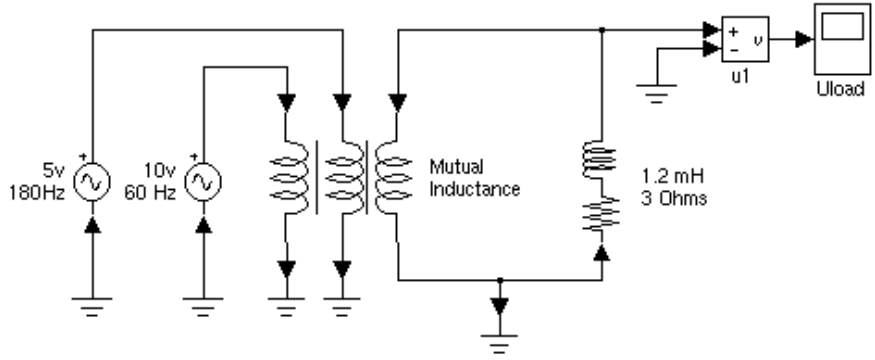
Windings can be left floating (i.e, not connected by an impedance to the rest of the circuit). The floating winding will be internally connected to the main circuit through a resistor. This invisible connection does not affect voltage and current measurements.

Example

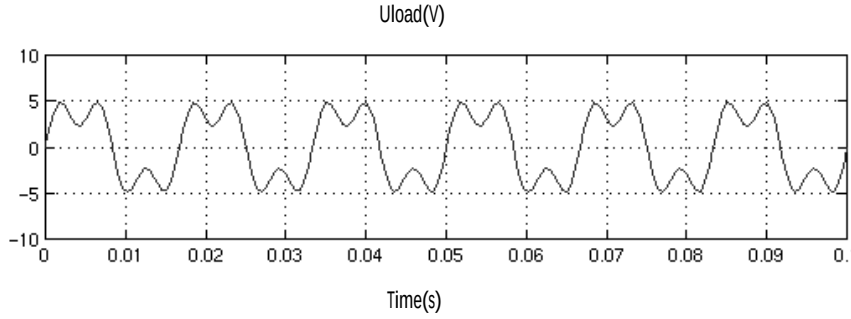
Three coupled windings are used to inject a third harmonic voltage into a circuit fed at 60 Hz.



This example is available in the `psbmutual.mdl` file.



Simulation produces the following load voltage waveform:



See Also **Linear Transformer, Saturable Transformer**

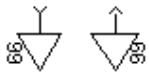
Purpose

Implement a local common node in the circuit

Library

Connectors Library

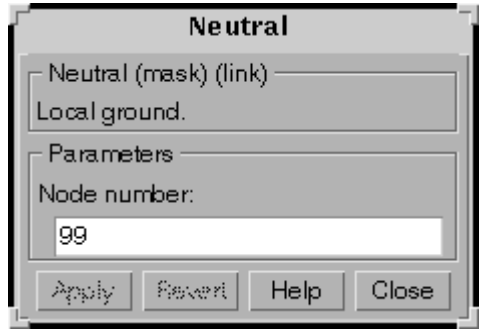
Description



The Neutral block implements a local common node with a specific node number. If the node number is set to zero, the Neutral block automatically makes a connection to ground. The node number is displayed on the icon. You can use this block to create a floating neutral or to interconnect two points without drawing a connection line.

For the drawing facility, two types of neutral blocks are available in the library: one block with an input and one block with an output.

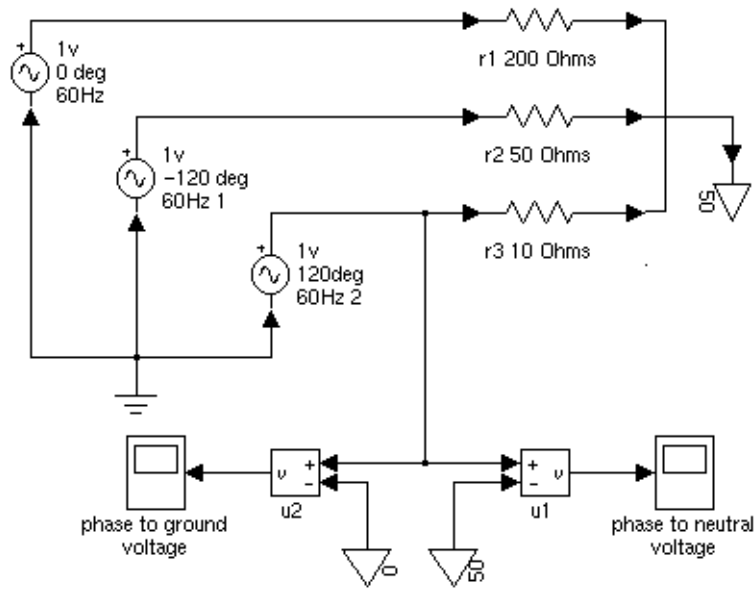
Dialog Box



Example

The following circuit uses three Neutral blocks. Three resistors are connected to a floating neutral (node 50). The phase-to-ground voltage is measured with the u2 block and the phase-to-neutral voltage is measured with the u1 block. This example is available in the `psbneutral.mdl` file.

Neutral



See Also **Ground**

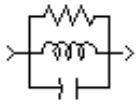
Purpose

Implement a parallel RLC branch

Library

Elements Library

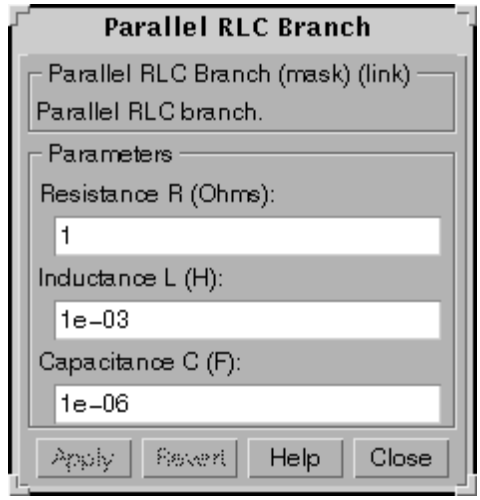
Description



The Parallel RLC Branch block implements a single resistor, inductor and capacitor or a parallel combination of these. To eliminate either the resistance, inductance, or capacitance of the branch, the R, L, and C values must be set respectively to infinity, infinity, and zero. Only existing elements will be displayed in the block icon.

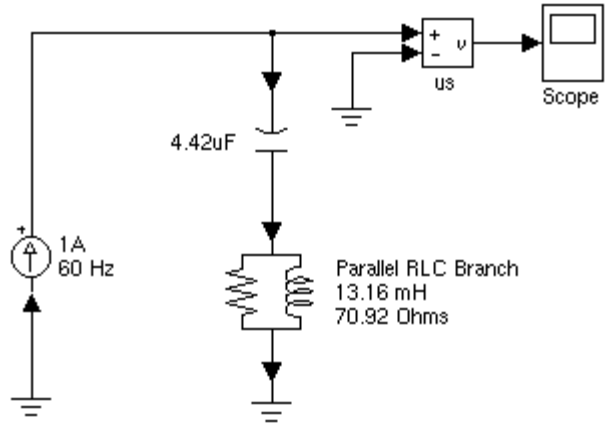
Negative values are allowed for resistance, inductance, and capacitance.

Dialog Box



Example

Obtain the frequency response of an eleventh-harmonic filter tuned at 660 Hz connected on a 60Hz power system is shown in the following figure. This example is available in the `psbparalbranch.mdl` file.



The network impedance in Laplace domain is:

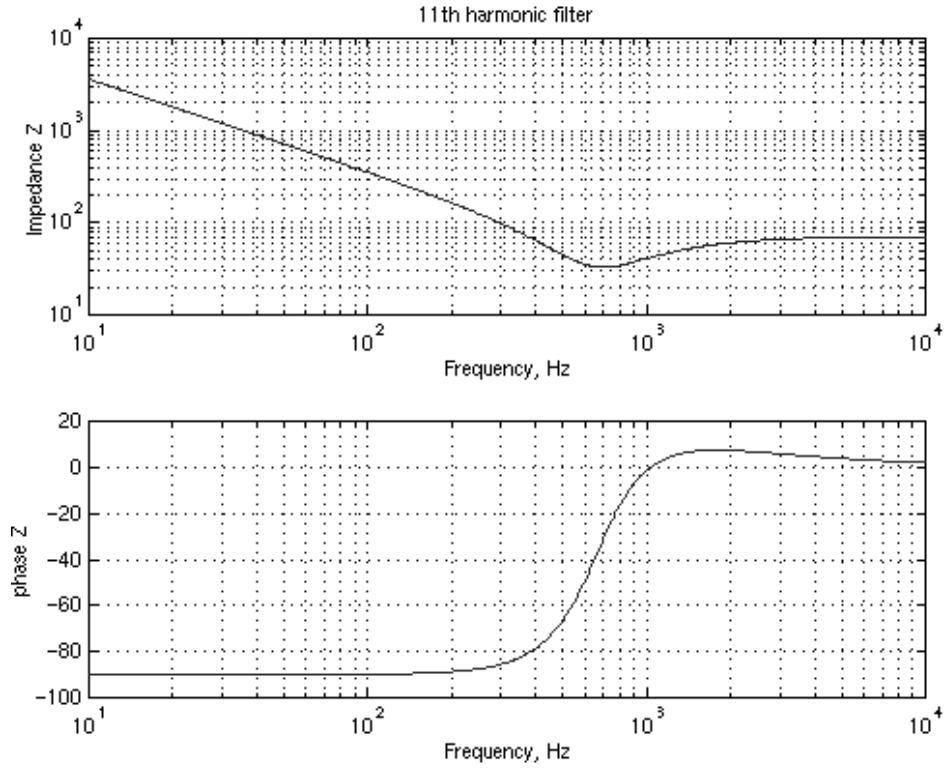
$$Z(s) = \frac{V(s)}{I(s)} = \frac{RLCs^2 + Ls + R}{LCs^2 + RCs}$$

To obtain the frequency response of the impedance you have to get the state-space model (A B C D matrices) of the system.

This system is a one input (Is) and one output (Vs) system. If you own the Control System Toolbox, you can get the transfer function Z(s) from the state-space matrices.

```
[A,B,C,D] = power2sys('psbparalbranch');
freq = logspace(1,4,500);
w = 2*pi*freq;
[Z,phaseZ] = bode(A,B,C,D,1,w);
subplot(2,1,1)
loglog(freq,Z)
grid
title('11th harmonic filter')
xlabel('Frequency, Hz')
ylabel('Impedance Z')
subplot(2,1,2)
semilogx(freq,phaseZ)
xlabel('Frequency, Hz')
ylabel('phase Z')
grid
```

Frequency response of the filter:



See Also **Parallel RLC Load**, **Series RLC Branch**, **Series RLC Load**

Purpose

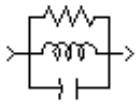
Implement a linear parallel RLC load

Library

Elements Library

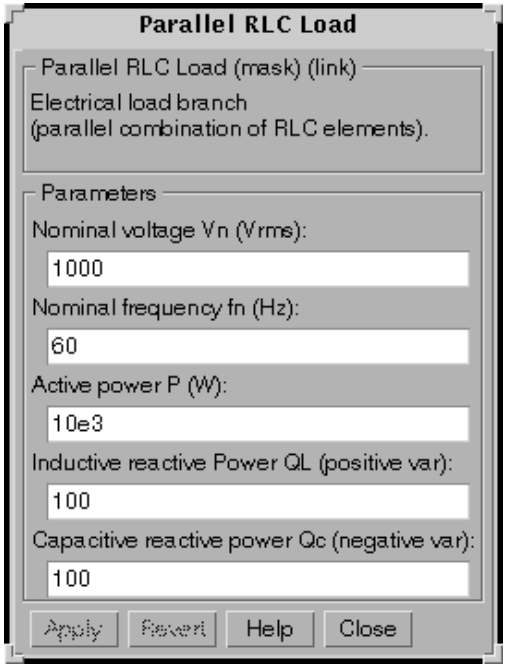
Description

The Parallel RLC Load block implements a linear load as a parallel combination of RLC elements.



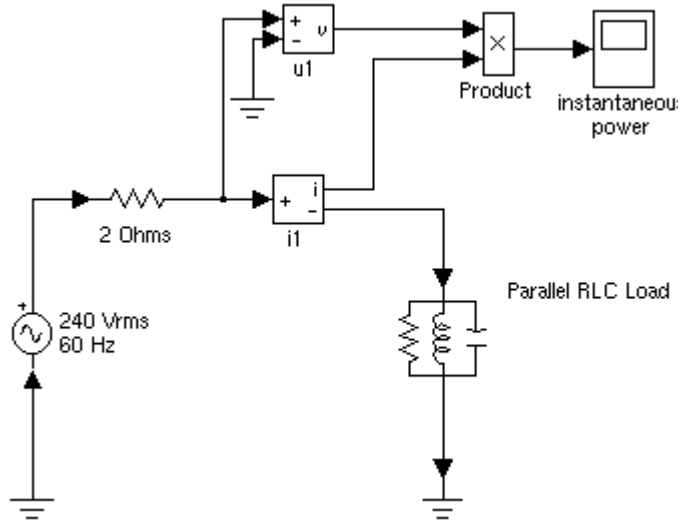
Dialog Box

Enter the value of the nominal voltage and nominal frequency in the first two entries. Enter the desired active power, the inductive reactive power, and the capacitive reactive power in the last three entries. The inductive and capacitive reactive powers should be positive values. At the specified frequency, the load will exhibit a constant impedance and its power will be proportional to the square of the applied voltage. Only elements associated with nonzero powers will be displayed in the block icon.

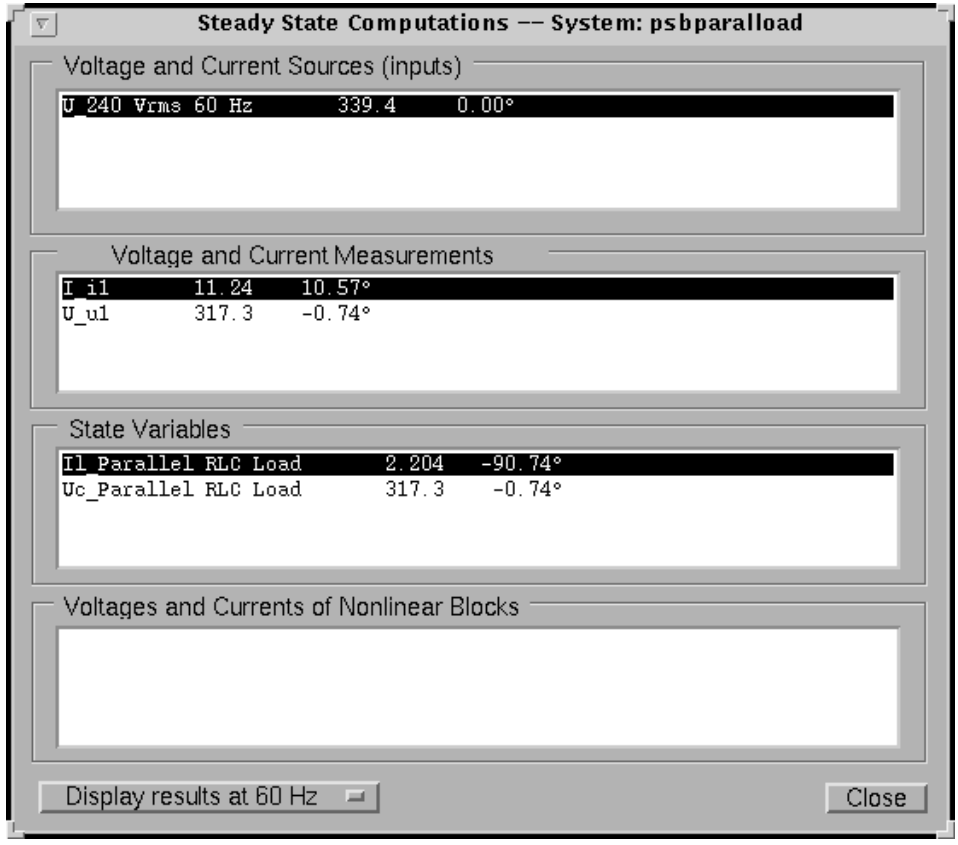


Example

Find the steady-state values of load voltage and current in the following circuit. This example is available in the `psbparalload.mdl` file.



Drag the **Powergui** block from the powerlib library into your circuit. Double click on the **Powergui** icon and output the voltage and current phasors by clicking on the **Steady state** button.



See Also **Parallel RLC Branch**, **Series RLC Load**, **Series RLC Branch**

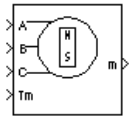
Purpose

Model the dynamics of a three-phase permanent magnet synchronous machine with sinusoidal flux distribution

Library

Machines Library

Description



The Permanent Magnet Synchronous Machine (PMSM) block operates in either motoring or generating mode. The mode of operation is dictated by the sign of the mechanical torque (positive for motoring, negative for generating). The electrical and mechanical parts of the machine are each represented by a second-order state-space model. The model assumes that the flux established by the permanent magnets in the stator is sinusoidal, which implies that the electromotive forces are sinusoidal.

The block implements the following equations expressed in the rotor reference frame (qd frame):

Electrical System

$$\frac{d}{dt}i_d = \frac{1}{L_d}v_d - \frac{R}{L_d}i_d + \frac{L_q}{L_d}p\omega_r i_q$$

$$\frac{d}{dt}i_q = \frac{1}{L_q}v_q - \frac{R}{L_q}i_q - \frac{L_d}{L_q}p\omega_r i_d - \frac{\lambda p\omega_r}{L_q}$$

$$T_e = 1.5p[\lambda i_q + (L_d - L_q)i_d i_q]$$

where (all quantities in the rotor reference frame):

- L_q, L_d: q and d axis inductances
- R: resistance of the stator windings
- i_q, i_d: q and d axis currents
- v_q, v_d: q and d axis voltages
- ω_r : angular velocity of the rotor
- λ : amplitude of the flux induced by the permanent magnets of the rotor in the stator phases
- p: number of pole pairs
- T_e: electromagnetic torque

Mechanical System

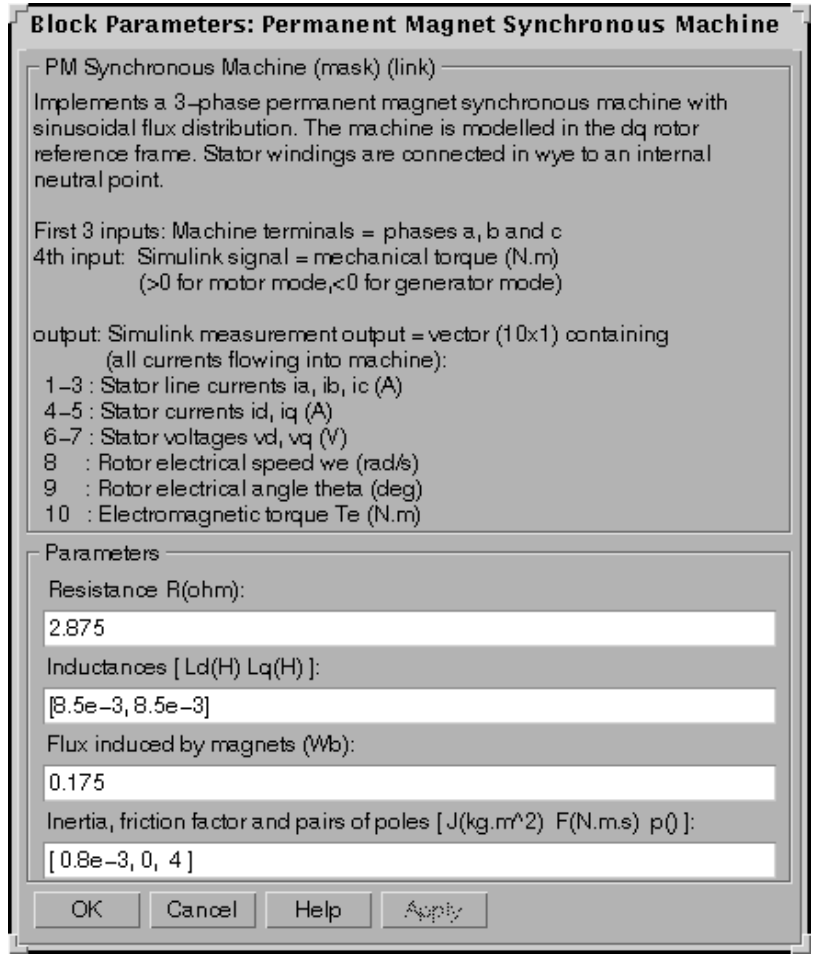
$$\frac{d}{dt}\omega_r = \frac{1}{J}(T_e - F\omega_r - T_m)$$

$$\frac{d\theta}{dt} = \omega_r$$

where:

- J: Combined inertia of rotor and load
- F: Combined viscous friction of rotor and load
- θ : Rotor angular position
- T_m: Shaft mechanical torque

Dialog Box



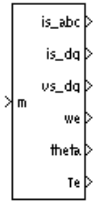
Inputs and Outputs

The first three inputs are the electrical connections of the machine's stator. The fourth input is the mechanical torque at the machine's shaft (Simulink signal). This input should ordinarily be positive because the PMSM block is usually used in a motor mode. Nevertheless, you can apply a negative torque input if you choose to use the PMSM block in generating mode.

The block outputs a vector containing the following ten variables (all currents flowing into machine):

- 1-3: Line currents i_a, i_b and i_c, in A;
- 4-5: q and d axis currents i_q and i_d, in A;
- 6-7: q and d axis voltages v_q and v_d, in V;
- 8: Rotor electrical speed ω_e, in rad/sec;
- 9: Rotor electrical angle θ_e, in degrees;
- 10: Electromagnetic torque T_e, in N.m.

These variables can be demultiplexed by using the special Permanent Magnet Synchronous Machine Demux block provided in the powerlib/Machine library.

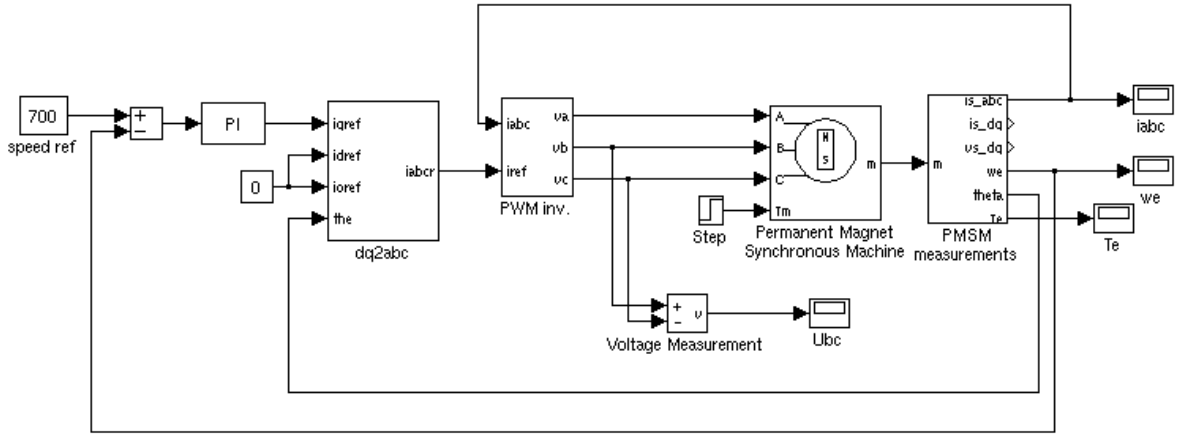


Assumption

The block assumes a linear magnetic circuit with no saturation of the stator and rotor iron. This assumption can be made because of the large air gap usually found in permanent magnet synchronous machines.

Example

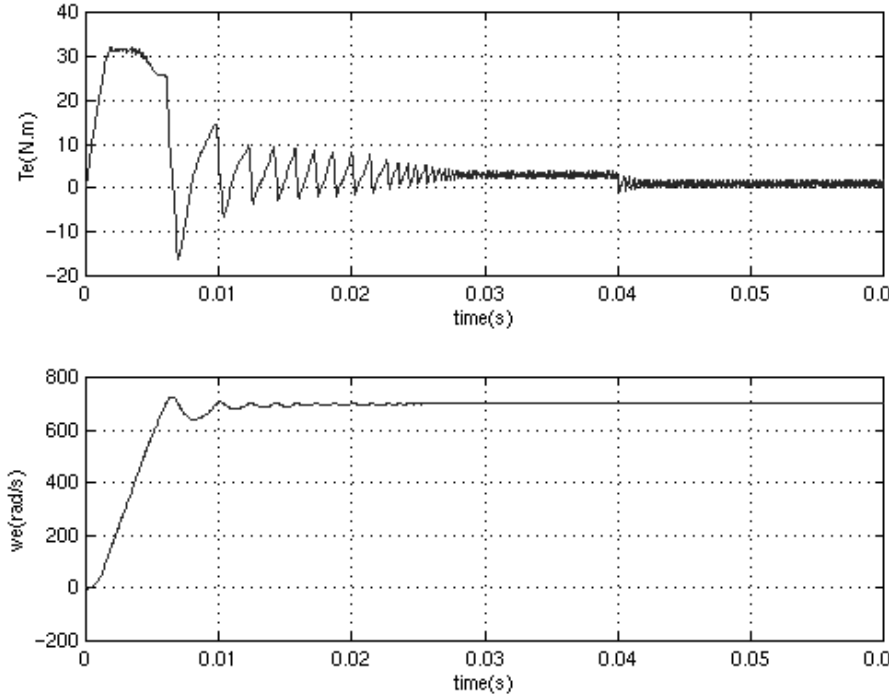
This example illustrates the use of the PMSM block in the motoring mode with a closed-loop control system built entirely in Simulink. The interfacing is done using Controlled Voltage Source blocks from the powerlib/Electrical Sources library. The complete system consists of a PWM inverter built with ideal switches (Simulink Relay blocks). Two control loops are used; the inner loop is used to regulate the motor line currents and the outer loop regulates the motor's speed. The mechanical torque applied at the motor's shaft is originally 3 N.m (nominal) and steps to 1 N.m at t=0.025 sec. The parameters of the machine are those found in the Dialog Box section.



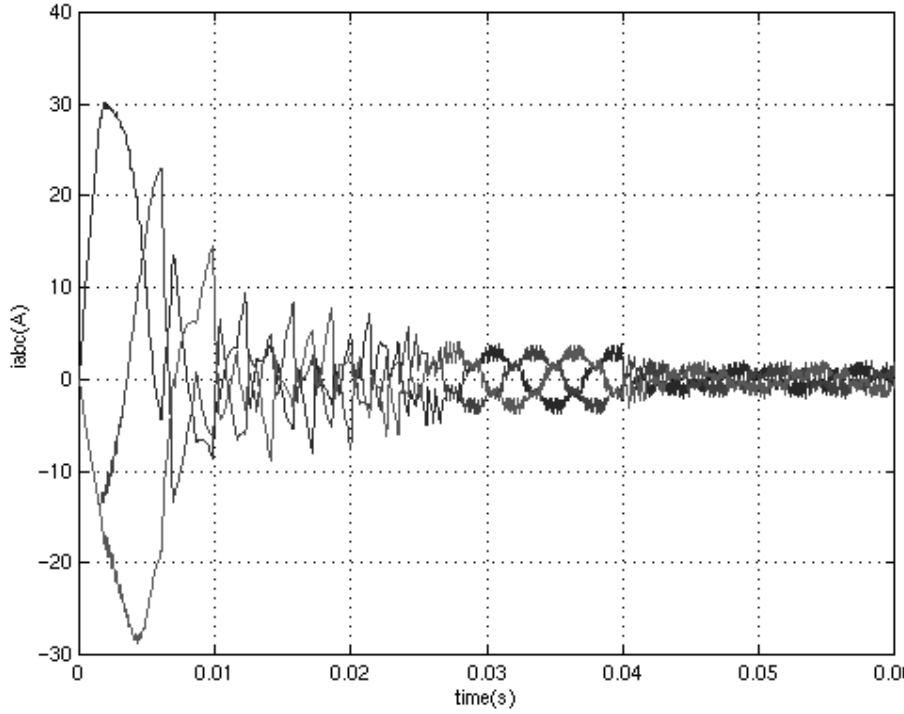
Open the Simulink diagram by typing `psbpmmotor` or by choosing Permanent Magnet Sync Mach (sim) from the demos group in the powerlib library. Set the simulation parameters as follows:

- Integrator type: Stiff, `ode15s`
- Stop time: 0.06
- Integration options: Use default options, except for Absolute tolerance which you can set to 1e-3.

Run the simulation. Once the simulation is completed, observe the motor's torque and speed.



The torque climbs to nearly 32 N.m when the motor starts but stabilizes rapidly to its nominal value (3 N.m), until the step is applied, at which point the torque oscillates slightly before stabilizing to its new value (1 N.m). As for the speed, you can see that it stabilizes quite fast at start-up and is not affected by the load step. Now, observe the line currents.



As for the torque, the currents are high when the machine starts, but stabilize quickly to their nominal value until the step is applied. Then they oscillate before stabilizing to a lower value, corresponding to the load torque decrease.

PI Section Line

Purpose

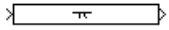
Implement a single phase transmission line with lumped parameters

Library

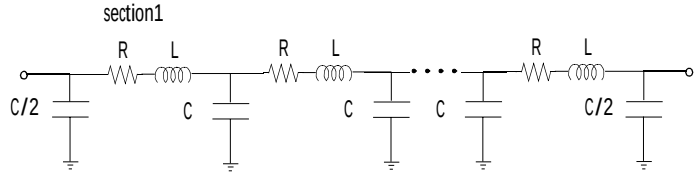
Elements Library

Description

The PI Section Line block implements a single phase transmission line with parameters lumped in pi sections.



For a transmission line, the resistance, inductance and capacitance are uniformly distributed along the line. An approximate model of the distributed parameter line is obtained by cascading several identical pi sections as shown in the figure below.



Unlike the distributed parameter line which has an infinite number of states, the pi section linear model has a finite number of states that permit a state-space model to be used to derive its frequency response. The number of sections to be used depends on the frequency range to be represented.

A good approximation of the maximum frequency range represented by the pi line model is given by the following equation:

$$f_{max} = \frac{Nv}{8l}$$

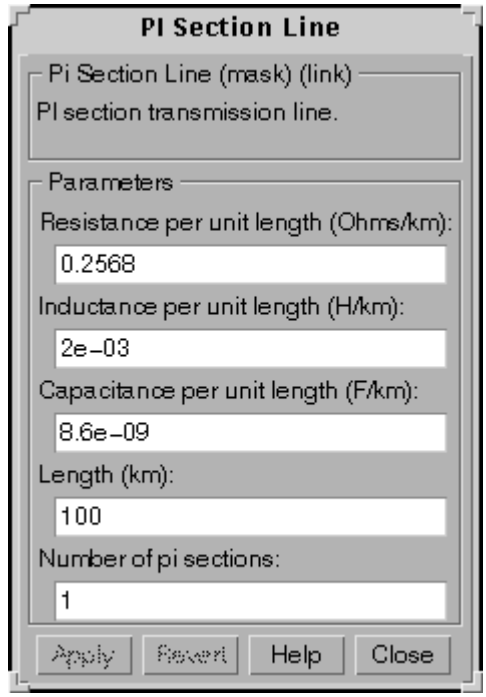
where:

- N = Number of pi sections
- v = Propagation speed in km/s = $1/\sqrt{LC}$ L(H/km) C(F/km)
- l = Line length in km

For example, for a 100 km aerial line having a propagation speed of 300 000 km/s, the maximum frequency range represented with a single pi section is approximately 375 Hz. For studying interactions between a power system and

a control system, this simple model would be sufficient. However for switching surge studies involving high frequency transients in the kHz range, much shorter pi sections should be used. In fact, accurate results would probably only be obtained by using a distributed parameters line model.

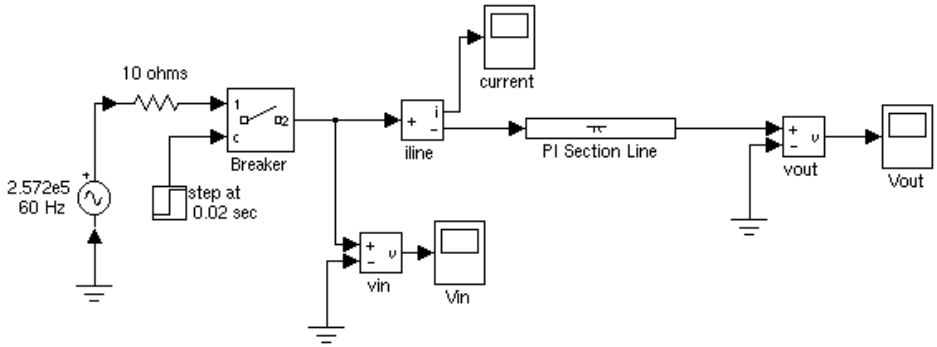
Dialog Box



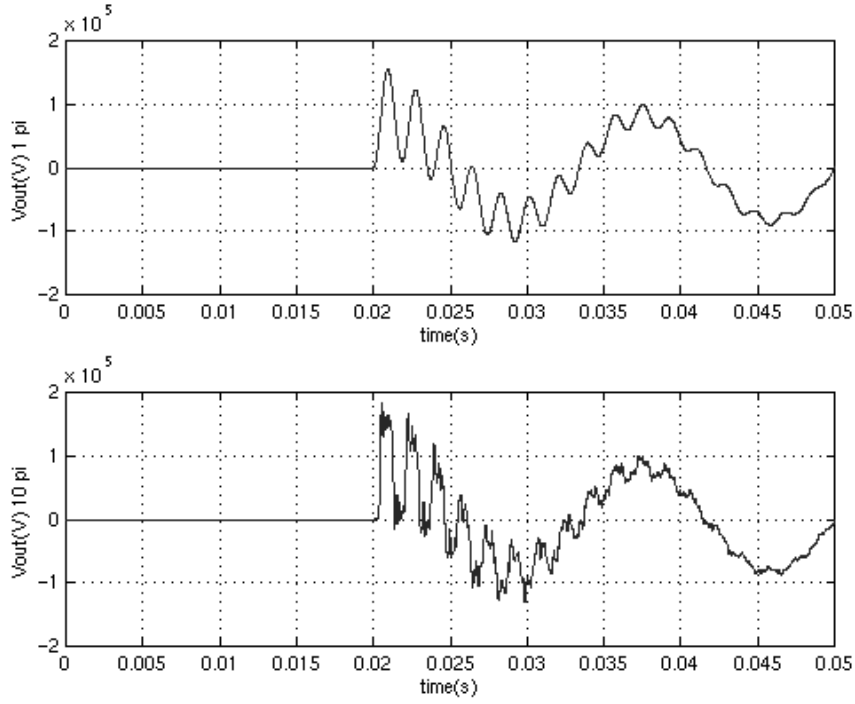
Enter the resistance, inductance, and capacitance per unit length in the first three entries. Enter the line length and the desired number of pi sections in the last two entries.

Example

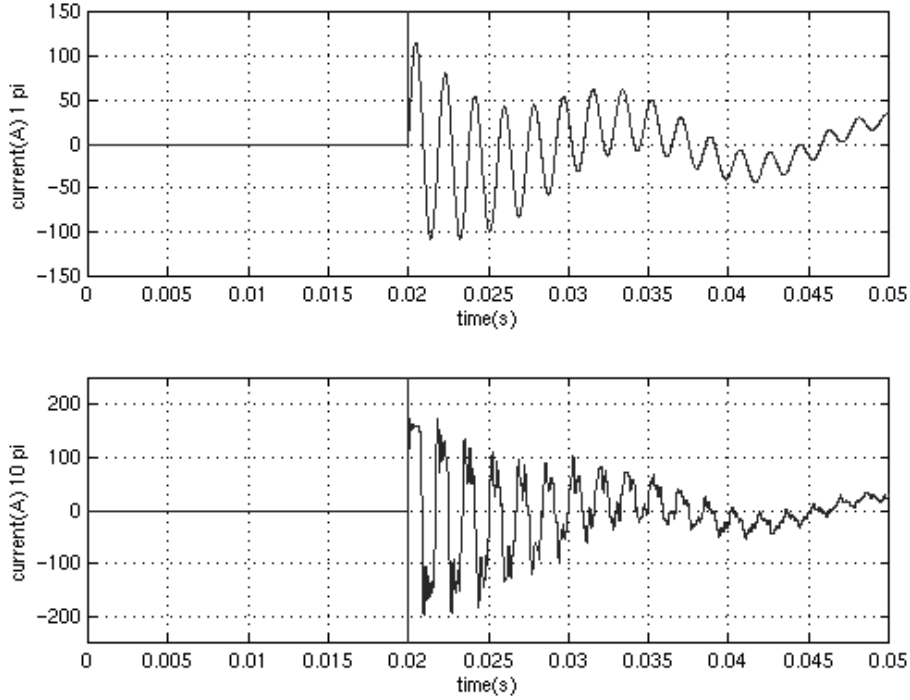
Obtain the line energization voltages and current in the following circuit. This circuit is available in the `psbpiline.mdl` file.



The results obtained with the line modeled by one pi section of 100km and 10 pi sections of 10km are shown below.


× 10^5
2
1
0
−1
−2
Vout(V) 1 pi
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
time(s)
× 10^5
2
1
0
−1
−2
Vout(V) 10 pi
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
time(s)

PI Section Line

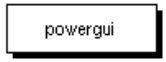


See Also **Distributed Parameter Line**

Purpose

Graphical user interface for the analysis of Power System circuits

Description

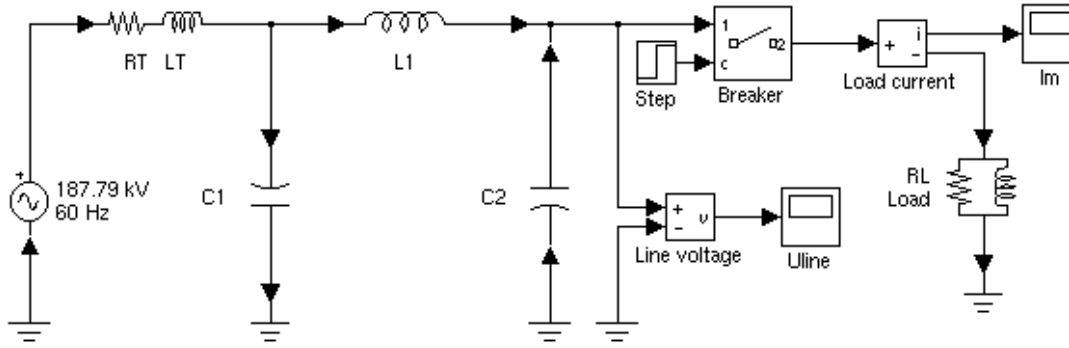


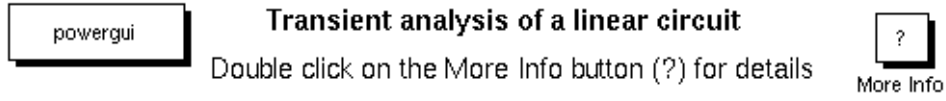
The **Powergui** block opens a Graphical user interface that displays steady-state values of measured current and voltages as well as all state variables (inductor currents and capacitor voltages). The **Powergui** interface allows you to modify the initial states in order to start the simulation from any initial conditions. It also allows Load Flow computation and initialization of three phase networks containing machines.

To use the **Powergui** interface, copy the **Powergui** block into your model and double-click on the block to open it.

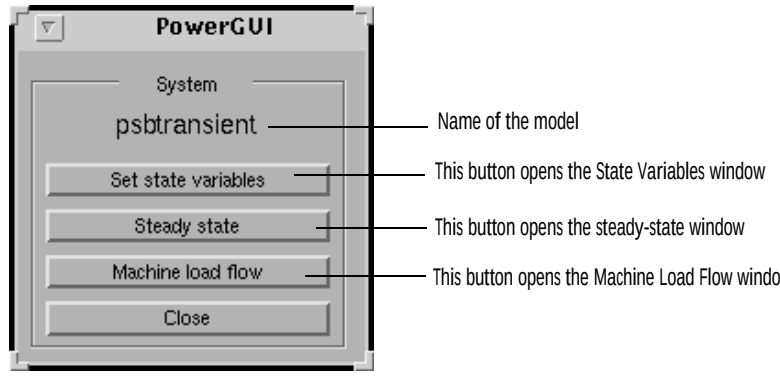
Example

The graphical interface is presented here for the `psbtransient.mdl` circuit demo entitled Transient Analysis.

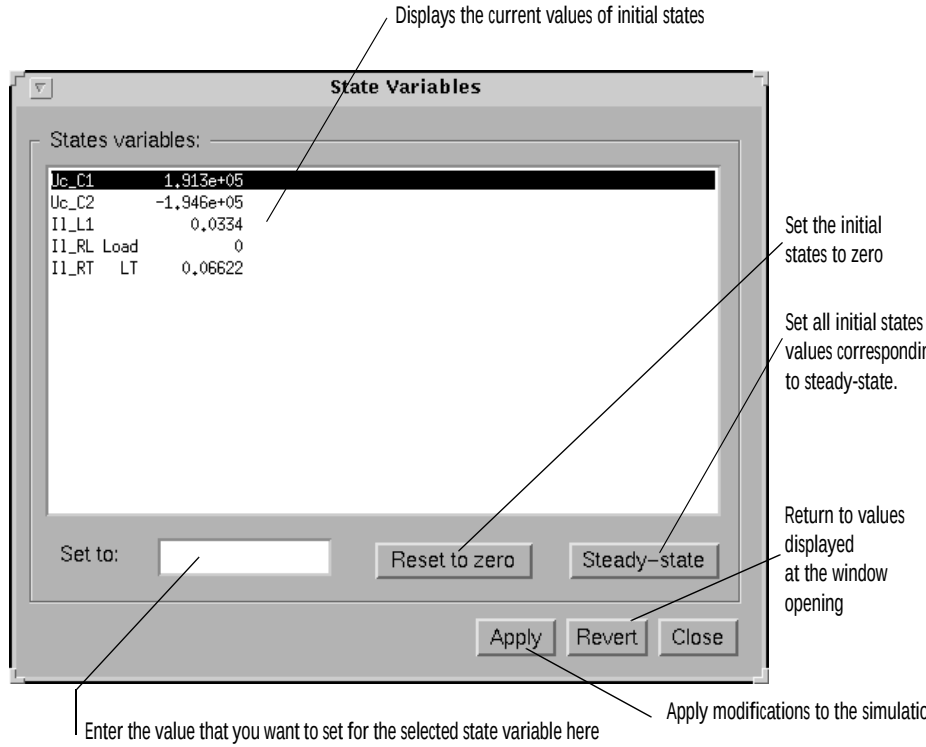




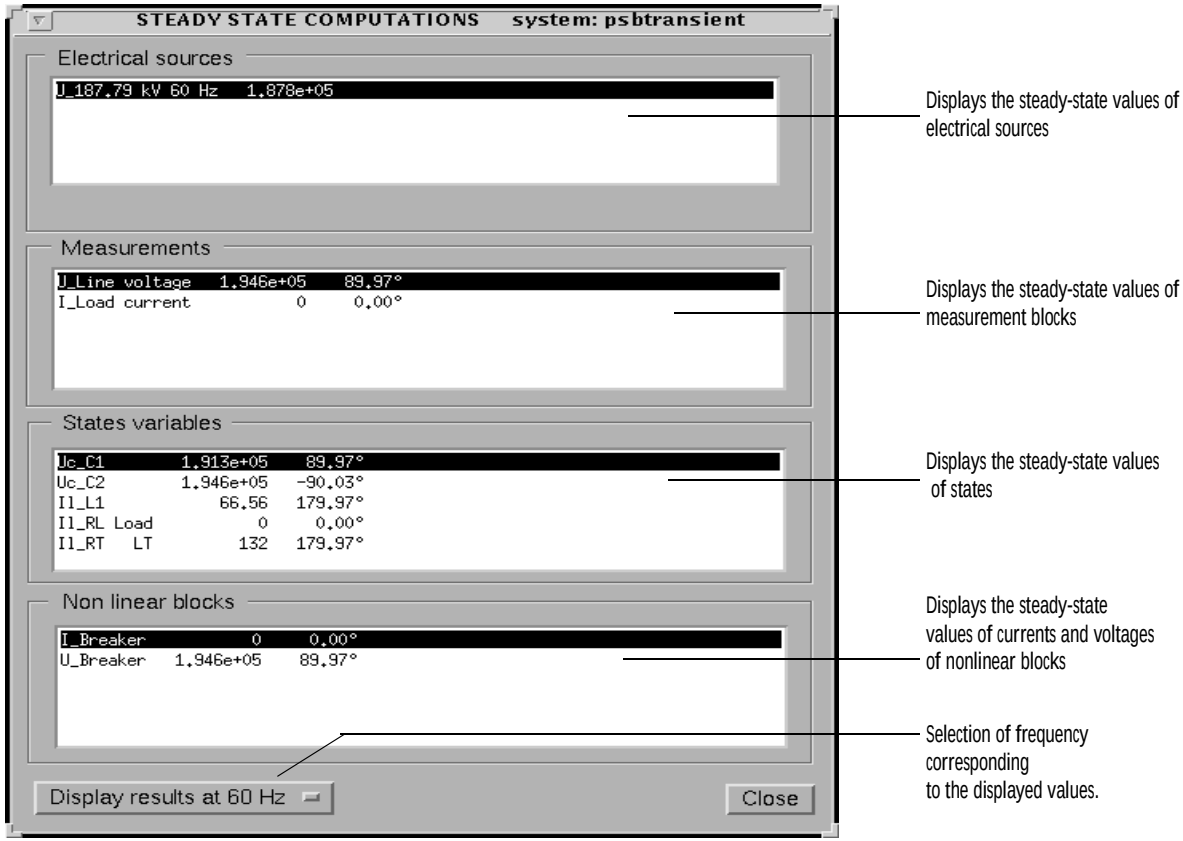
If the **Powergui** block is not in the model, copy it from the powerlib library. Start the simulation to establish the state space model (this occurs automatically through Power2sys). Double-click on the block to open the following window:



The Set State Variables window is used to set initial states before the simulation.

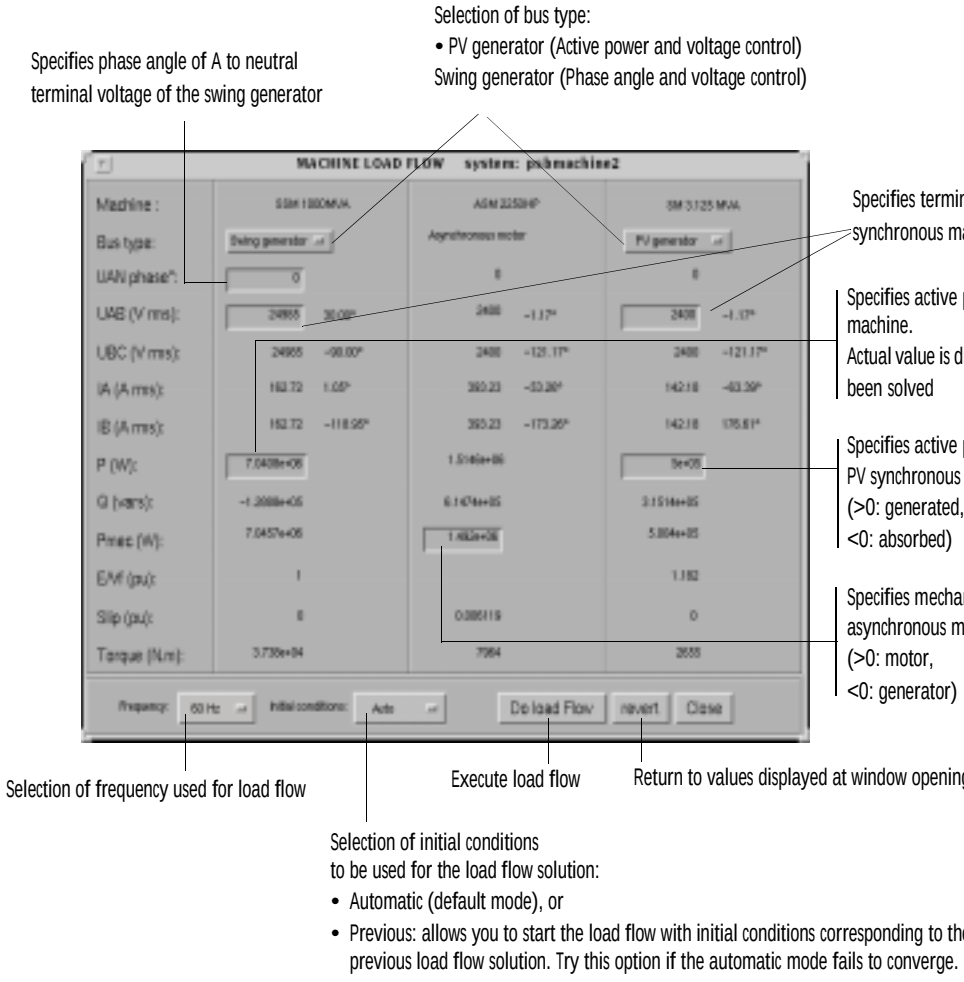


The Steady-State window displays the phasor values (magnitude and phase) of sources, measured voltages and currents, state variables as well as voltages and currents of nonlinear blocks.



The **Machine Load Flow** window allows you to perform load flows and initialize three-phase circuits containing the following three types of machines: Simplified Synchronous Machine, Synchronous Machine, and Asynchronous Machine. Once the load flow has been solved, initial conditions are automatically set in the dialog boxes of the machines and you can start the simulation in steady-state.

The use of the machine load flow is illustrated in “Session 5: Simulating Three-Phase Systems and Using Electrical Machines” in Chapter 1. We reproduce here the window displayed for the three machine case



See Also

power2sys

Purpose

Analyze an electric circuit built with the Power System Blockset

Syntax

```
POWER2SYS('simwin');
[A,B,C,D,x0,state_var,inputs,outputs,uss,xss,yss,freqyss,Hlin]=
   POWER2SYS('simwin');
[A,B,C,D,x0,state_var,inputs,outputs,uss,xss,yss,freqyss,Hlin]=
   POWER2SYS('simwin','n');
```

Description

The Power2sys block is used to computes the equivalent state-space model of an electrical network built with the Power System Blockset. `power2sys` extracts the linear part of the `simwin` system and builds the corresponding state-space model. State variables are the inductor currents and capacitor voltages. Nonlinear elements are simulated by current sources (state-space model inputs) driven by the voltages (state-space model outputs) across the nonlinear elements.

```
[A,B,C,D,x0,state_var,inputs,outputs,uss,xss,yss,freqyss,
 Hlin]=POWER2SYS('simwin')
```

computes the equivalent state-space model of the Simulink window named `simwin`. A,B,C,D are the standard matrices of the state-space system described by the equations

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

x0 is a vector containing the initial conditions of the state variables listed in `state_var`.

state_var is a string matrix containing names of the state variables (x vector). Each line of `state_var` begins with a prefix `Uc_` for capacitor voltages or `Il_` for inductor currents, followed by the name of the block in which the element (C or L) is found. Inductor current direction and capacitor voltages polarities are defined by the input and output of the block. The following conventions are used:

- Current flowing in the arrow direction is positive.
- voltage = Vinput-Voutput.

A suffix is added to the line for blocks containing more than two inductances or capacitors: for example, the three winding Linear Transformer block will produce three `state_var` lines, one for each leakage inductance, with the suffix `coil:x` where x is the winding number of the transformer.

`inputs` is a string matrix containing names of the inputs of the system (vector u). Each line of `inputs` begins with a prefix `U_` for voltage sources or `I_` for current sources, followed by the name of the source block.

The following conventions are used for inputs:

- Source current flowing in the arrow direction is positive.
- Positive source voltage is indicated by a + sign on the icon.

A suffix may be added to the input for blocks containing more than one source. For example, the Simplified Alternator block produces two current inputs with suffixes AB and BC.

`outputs` is a string matrix containing names of the outputs of the state-space system (vector y). Each line of `outputs` begins with a prefix `U_` for voltage outputs or `I_` for current outputs, followed by the name of the block which produces the output. Sign conventions are indicated by the polarities of the voltage measurement and current measurement blocks.

`uss`, `xss`, and `yss` are complex matrices containing the steady-state values (phasors) of inputs, states and outputs. If voltage and current sources all generate the same frequency, these are column vectors. If sources with different frequencies are used, each column of the matrices corresponds to a frequency contained in the `freqyss` vector.

`freqyss` is a column vector containing the `n_freq` input source frequencies ordered by increasing values.

`Hlin` is the complex transfer impedance three-dimension array (`n_output` by `n_input` by `n_freq`) of the linear system corresponding to the frequencies contained in the `freqyss` vector. For a particular frequency, `Hlin` is defined by

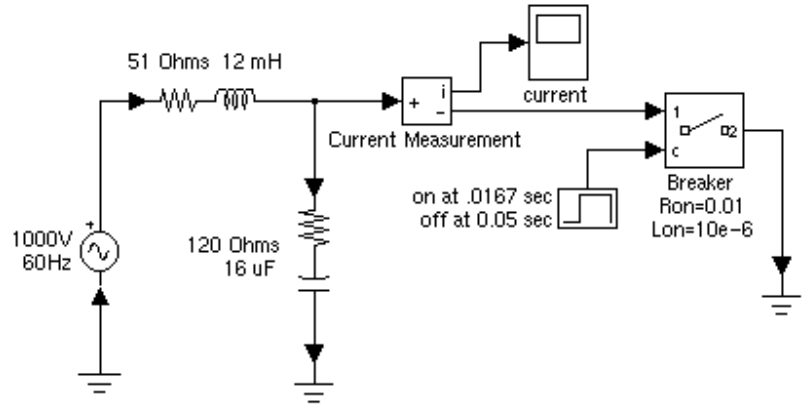
$$yss(:,ifreq) = Hlin(:,:,ifreq) \times uss(:,ifreq)$$

Netlist. When called with a second argument `'n'`, `powers2sys` generates a netlist stored in a file, `simwin.net`. This file contains the node numbers

automatically generated by power2sys, as well as parameter values of all linear elements. See formats in the circ2ss reference section.

Example

Obtain the state-space matrices and steady-state voltages and currents for the following circuit.



First open the `psbnetsim2.mdl` example then execute `power2sys`:

```
psbnetsim2
[A,B,C,D,x0,states,inputs,outputs,uss,xss,yss,freqyss,
Hlin]=power2sys('psbnetsim2')

A =
    1.0e+04 *
          0      6.2500
   -0.0083      -1.4250

B =
    1.0e+04 *
          0     -6.2500
    0.0083       1.000

C =
     1    120
     0      0

D =
```

```
      0   -120
      0    1
```

Initial values of states:

```
x0 =
 -513.1443
    2.9190

states =
Uc_120 Ohms 16 uF
Il_51 Ohms  12 mH

inputs =
U_1000V 60Hz
I_Breaker  Ron=0.01 Lon=10e-6

outputs =
U_Breaker  Ron=0.01 Lon=10e-6
I_Current Measurement
```

Steady-state phasor values of input sources, states, and outputs

```
uss =
        1000
              0

xss =
   1.0e+02 *
   4.8392 - 5.1314i
   0.0310 + 0.0292i

yss =
   1.0e+02 *
   8.5535 - 1.6287i
            0
```

The system contains only one source frequency:

```
freqyss =
    60
```

Transfer function matrix:

```
Hlin =
   0.8553 - 0.1629i -44.3596 + 4.4368i
        0             1.0000
```

There are two state variables in this circuit. Note that the breaker block is a nonlinear element, which is represented by a current source (second input) driven by the voltage across the breaker (first output).

See Also

Circ2ss

Saturable Transformer

Purpose

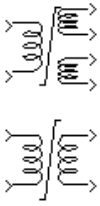
Implement a two- or three-winding saturable transformer

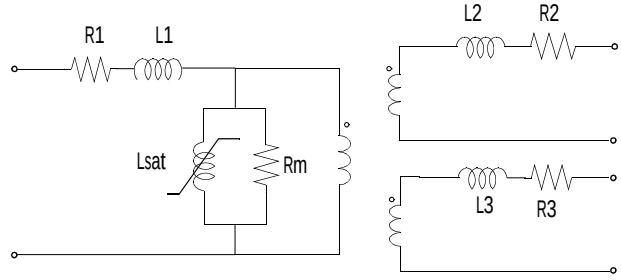
Library

Elements Library

Description

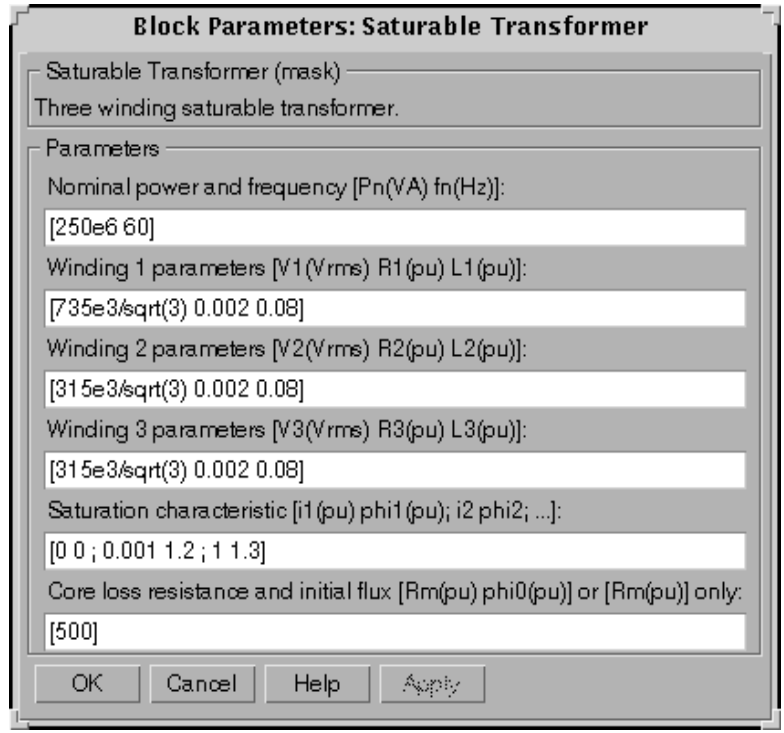
The Saturable Transformer block model shown below consists of three coupled windings wound on the same core.





The model takes into account the winding resistances (R1 R2 R3), the leakage inductances (L1 L2 L3) as well as the magnetizing characteristics of the core, which is modeled by a resistance Rm simulating the core active losses and a saturable inductance Lsat. The saturation characteristic is specified as a piece-wise linear characteristic.

Dialog Box and Parameters



Specify in the first entry the nominal power rating and frequency of the transformer.

Then specify in the following three entries the parameters of each winding (nominal voltage in volt rms, resistance, and leakage inductance in p.u).

The fifth entry is used to specify the saturation characteristic. Specify a series of current (pu)/flux (pu) pairs starting with (0,0).

Specify in the last entry the active power dissipated in the core by entering the equivalent resistance Rm in pu. In the last entry you can also specify the initial flux phi0(p.u). This initial flux becomes particularly important when the transformer is energized. If phi0 is not specified, the initial flux will be automatically adjusted so that the simulation starts in steady-state.

To comply with industry practice, you must enter the resistance and inductances in per unit (pu) based on the transformer rated power (Pn in VA) and nominal voltage of the winding (Vn in Vrms). The base resistance and inductance are defined as follows:

$$Rbase = 1\ pu = \frac{(Vn)^2}{Pn}$$

$$Lbase = 1\ pu = \frac{Rbase}{2\pi f_n}$$

For example, for the default parameters specified in the dialog box.

$$Rbase = \frac{(424.35 \times 10^3)^2}{250 \times 10^6} = 720.3\Omega$$

$$Lbase = \frac{720.3}{2\pi 60} = 1.91H$$

$$R_1 = 0.002pu \times 720.3\Omega = 1.44\Omega$$

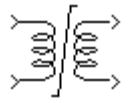
$$L_1 = 0.08pu \times 1.91H = 0.1528H$$

$$R_m = 500pu \times 720.3\Omega = 3.6 \times 10^5\Omega$$

Inputs and Outputs

Input one, output one and output three (if it exists) are at the same instantaneous polarity.

If you set the entry for the third winding to zero, the blockset will consider a transformer with two windings and a new icon will be displayed:



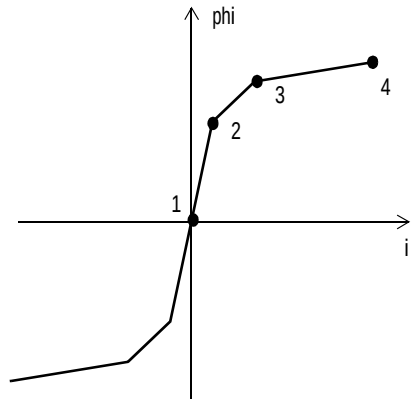
Restrictions

Because of modeling constraints the following restrictions apply: The winding resistances cannot be set to zero; however, leakage inductances can be set to zero. Use values as small as possible to simulate quasi-zero resistances. Similarly, the magnetizing resistance Rm must have a finite value.

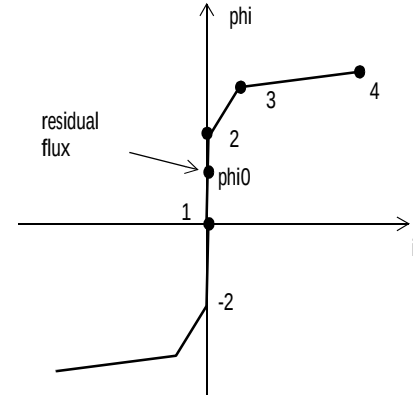
Windings can be left floating (i.e, not connected by an impedance to the rest of the circuit). However, the floating winding will be connected internally to the main circuit through a resistor. This invisible connection does not affect voltage and current measurements.

Figure (a) shows the piece-wise linear saturation characteristic. The points 2, 3, and 4 are entered as (i, phi) pair values in the dialog box Saturation characteristic section.

The saturation model does not include hysteresis. Therefore, if you want to specify a residual flux phi0, the second point of the saturation characteristic should correspond to a zero current as shown on the figure (b) below.



(a) No residual flux can be specified.

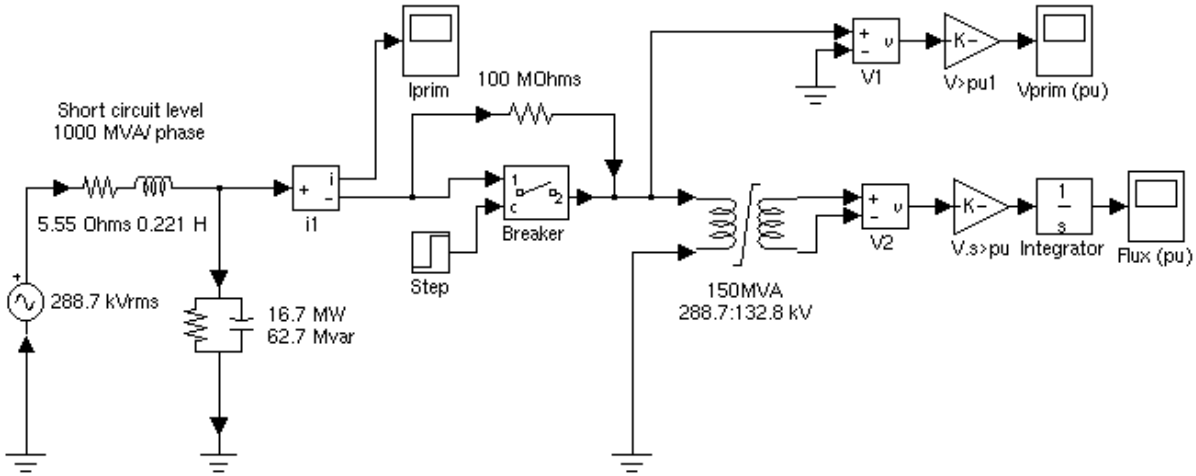


(b) A residual flux can be specified between points 2 and -2.

Example

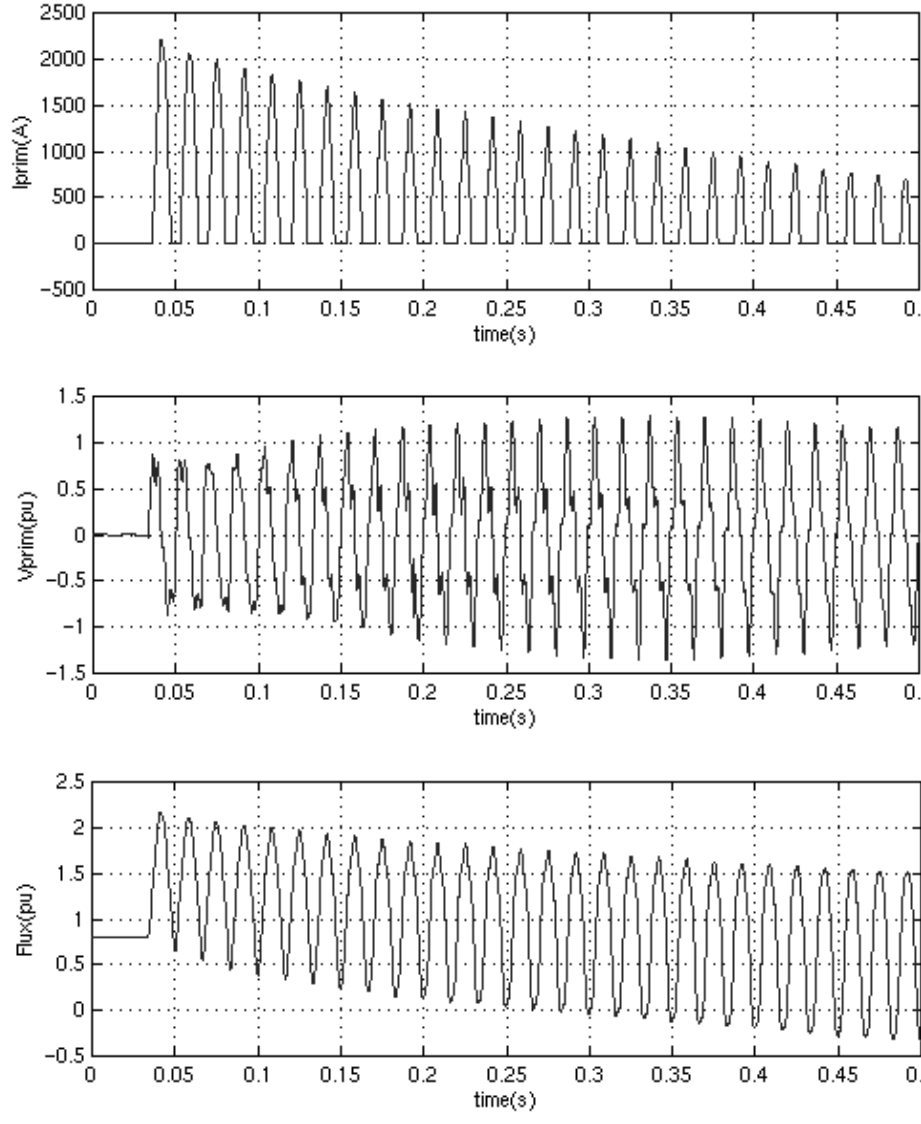
Energization of one phase of a three-phase 450 MVA, 500/230 kV transformer on a 3000 MVA source. The transformer parameters are:

Nominal power: 150MVA, 60Hz, Winding 1 parameters (primary): 500KVrms/ sqrt(3), R=0.002pu X=0.08pu, winding 2 parameters (secondary): 230KVrms/ sqrt(3), R=0.002pu X=0.08pu, Core loss resistance: 500pu, Saturation characteristic: [0 0; 0 1.2; 1.0 1.52], residual flux=0.8 pu.



This circuit is available in `psbxfosaturable.mdl`. It illustrates the transformer saturation effect on system current and voltage. Plotted waveforms show the transformer inrush current, the primary voltage and the flux.

As the source is resonant at the 4th harmonic, you can observe a high 4th harmonic content in the primary voltage. The flux is obtained by integrating the secondary voltage.



See Also **Linear Transformer, Mutual Inductance**

Series RLC Branch

Purpose

Implement a series RLC branch

Library

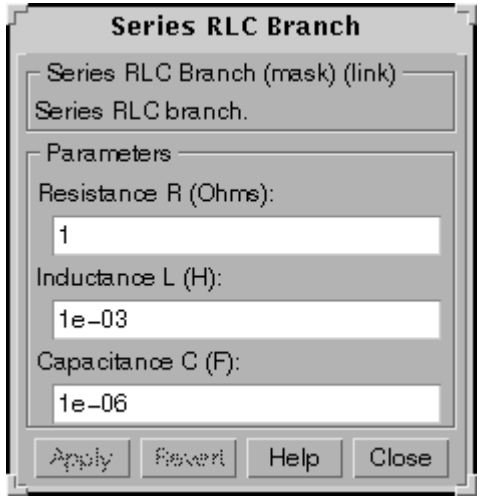
Elements Library

Description

The Series RLC Branch block implements a single resistor, inductor, or capacitor, or a series combination of these. To eliminate either the resistance, inductance, or capacitance of the branch, the R, L and C values must be set respectively to zero, zero, and infinity. Only existing elements will be displayed in the block icon.

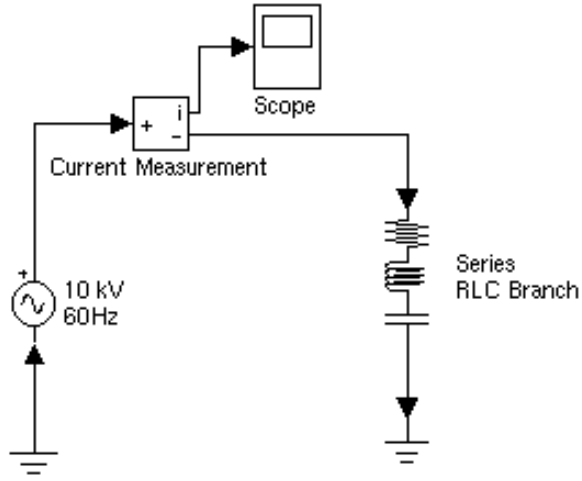
Negative values are allowed for resistance inductance and capacitance.

Dialog Box



Example

Obtain the frequency response of a fifth-harmonic filter (tuned frequency = 300 Hz) connected on a 60Hz power system. This example is available in `psbseriesbranch.mdl`.



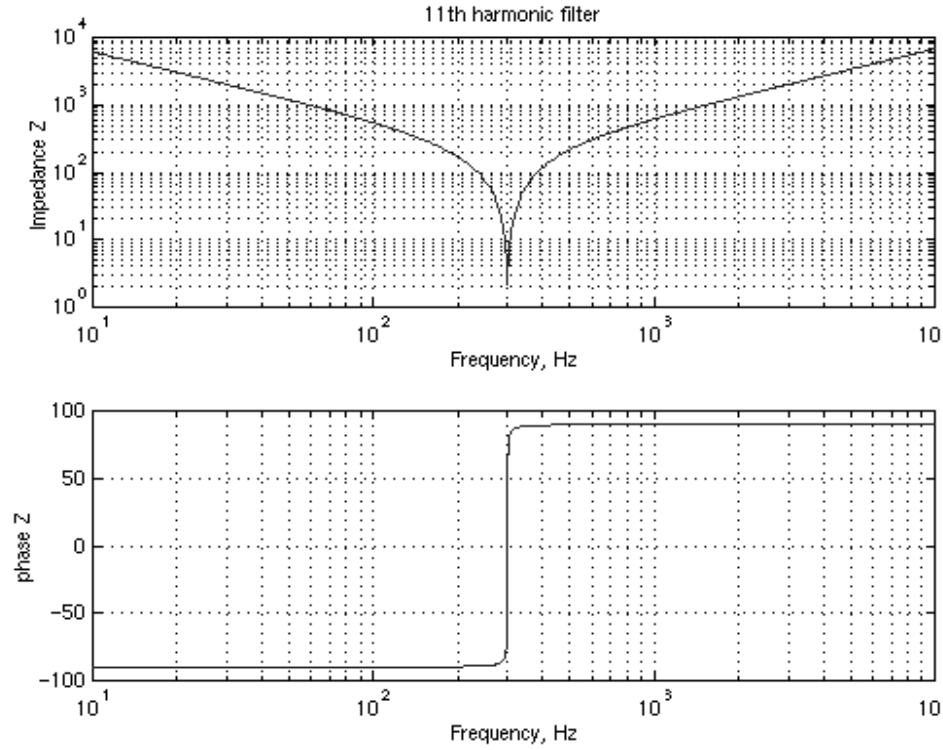
The network impedance in Laplace domain is:

$$Z(s) = \frac{V(s)}{I(s)} = \frac{LCs^2 + RCs + 1}{Cs}$$

To obtain the frequency response of the impedance you have to get the state-space model (A B C D matrices) of the system.

This system is a one input (Vsource) and one output (Current Measurement) system. If you own the Control System Toolbox, you can get the transfer function Z(s) from the state-space matrices as follows:

```
[A,B,C,D] = power2sys('psbseriesbranch');
freq = logspace(1,4,500);
w = 2*pi*freq;
[Y,phaseY] = bode(A,B,C,D,1,w);
% invert Y(s) to get Z(s)
Z = 1./Y;
phaseZ = -phaseY;
subplot(2,1,1)
loglog(freq,Z)
grid
title('5th harmonic filter')
xlabel('Frequency, Hz')
ylabel('Impedance Z')
subplot(2,1,2)
semilogx(freq,phaseZ)
xlabel('Frequency, Hz')
ylabel('phase Z')
grid
```



See Also **Series RLC Load**, **Parallel RLC Branch**, **Parallel RLC Load**

Series RLC Load

Purpose

Implement a linear series RLC load

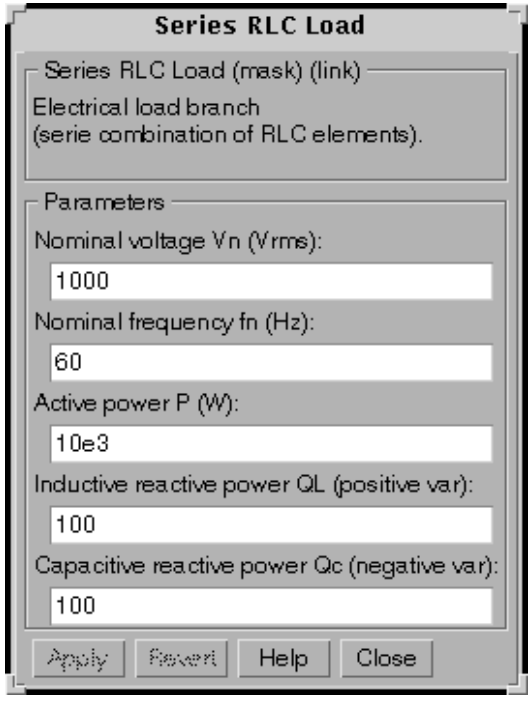
Library

Elements Library

Description

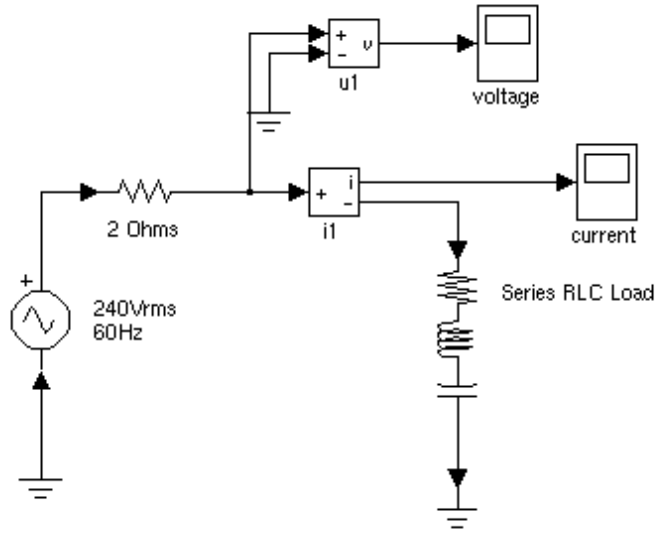
The Series RLC Load block implements a linear load as a series combination of RLC elements. Enter the value of the nominal voltage and nominal frequency in the first two entries. Enter the desired active power, the inductive reactive power and the capacitive reactive power in the last three entries. Only elements associated with nonzero powers will be displayed in the block icon. The inductive and capacitive reactive powers should be entered as positive values. At the specified frequency, the load will exhibit constant impedance and its power will be proportional to the square of the applied voltage.

Dialog Box

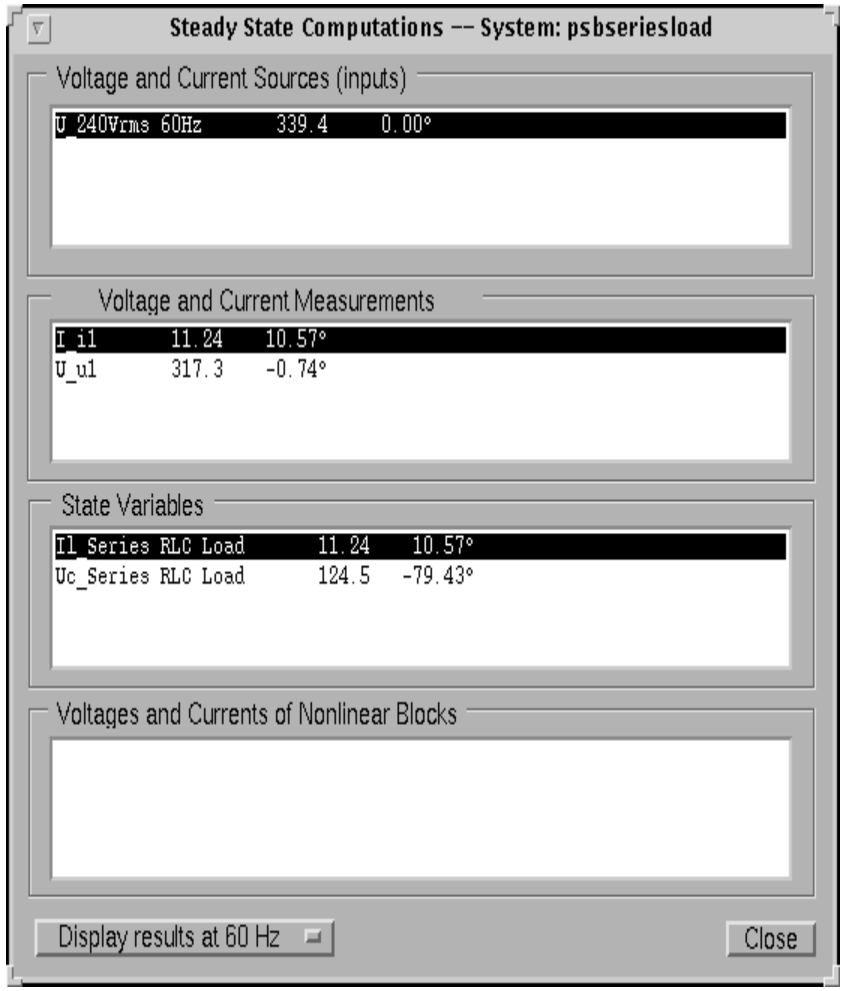


Example

Find the steady-state values of load voltage and current in the following circuit. This example is available in `psbseriesload.mdl`.



Use the **Powergui** interface to output the voltage and current phasors by clicking on the **Steady state** button.



See Also **Series RLC Branch**, **Parallel RLC Branch**, **Parallel RLC Load**

Purpose

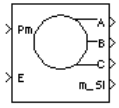
Model the dynamics of a simplified three-phase synchronous machine

Library

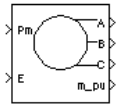
Machines Library

Description

The Simplified Synchronous Machine block models both the electrical and mechanical characteristics of a simple synchronous machine.



The electrical system for each phase consists of a voltage source in series with an RL impedance, which implements the internal impedance of the machine. The value of R can be zero but the value of L must be positive.



The Simplified Synchronous Machine block implements the mechanical system described by

$$\Delta\omega(t) = \frac{1}{2H}\int_0^t (Tm - Te)\,dt - Kd\Delta\omega(t)$$

$$\omega(t) = \Delta\omega(t) + \omega_0$$

where

$\Delta\omega$ = Speed variation with respect to speed of operation

H = Constant of inertia

Tm = Mechanical torque

Te = Electromagnetic torque

Kd = Damping factor

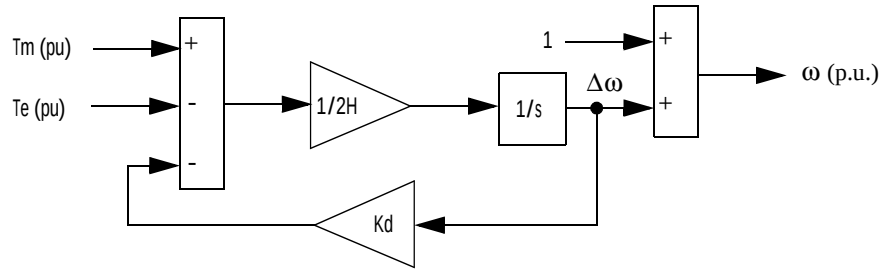
$\omega(t)$ = Mechanical speed of rotor

ω_0 = Speed of operation (1pu)

Although the parameters can be entered in either SI units or per unit in the dialog box, the internal calculations are done in per unit (pu).

The following block diagram illustrates how the mechanical part of the model is implemented. Notice that the model computes a deviation with respect to the speed of operation, and not to the absolute speed itself.

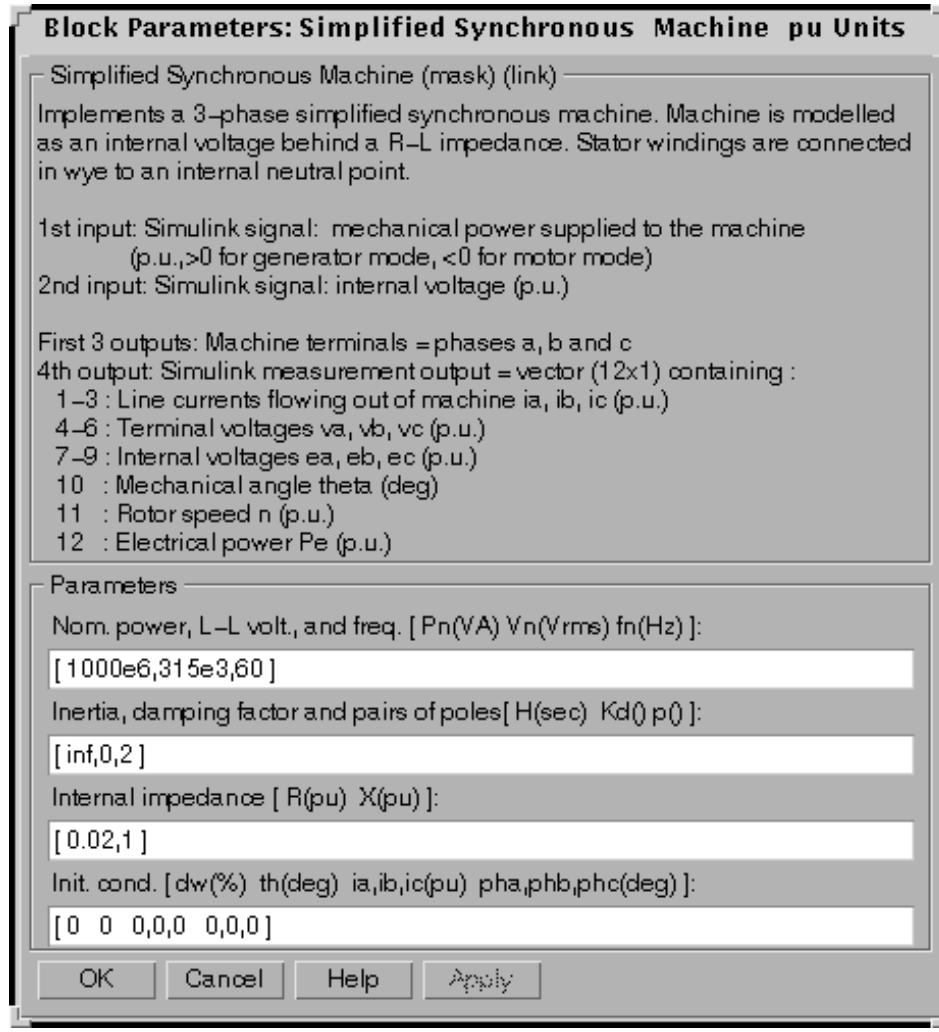
Simplified Synchronous Machine



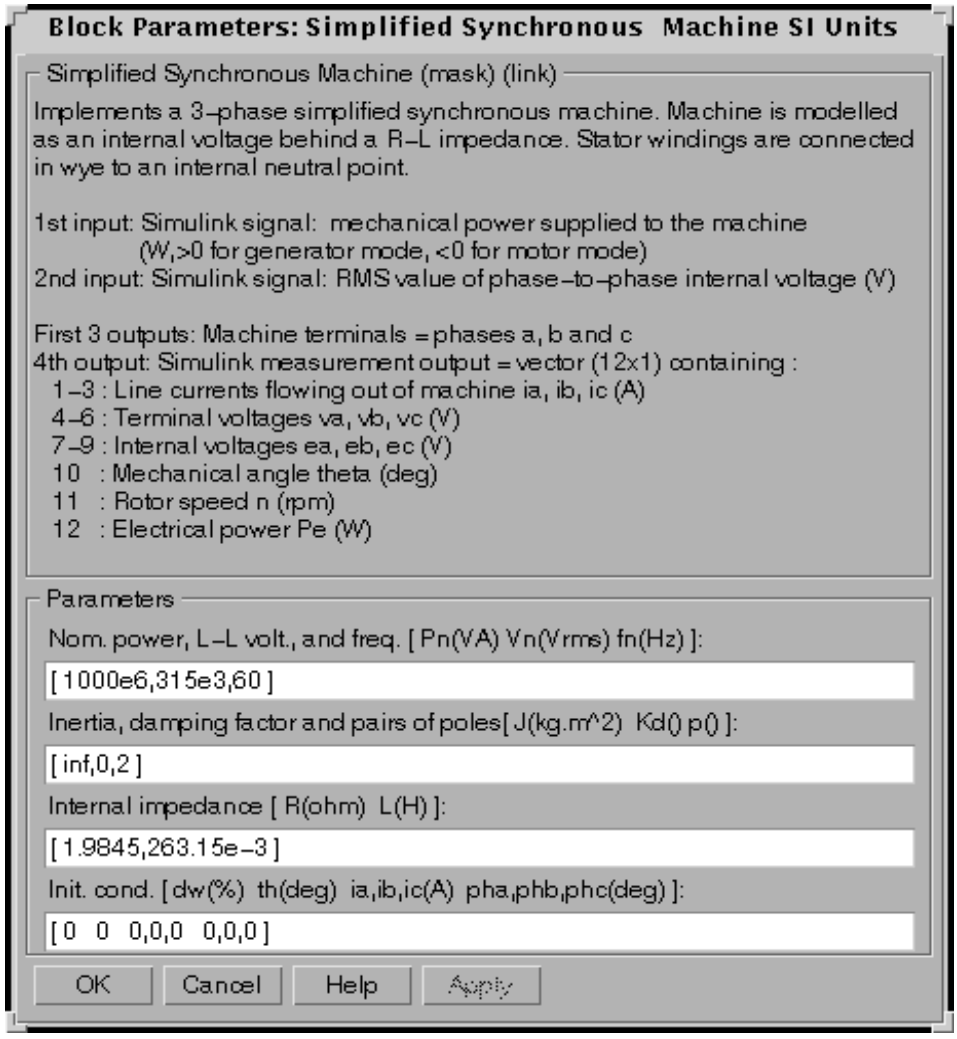
Parameters and Dialog Box

In the powerlib library you can choose between two Simplified Synchronous Machine blocks to specify the electrical and mechanical parameters of the model.

Per Unit (pu) Dialog Box



SI Units Dialog Box



The nominal power, frequency and voltage are used to compute nominal torque and convert SI units to pu.

The moment of inertia and damping factor are used to model the mechanical behavior of the machine. The damping factor has been scaled to act like the

damping factor of a second order system. This means that for no overshoot and minimum settling time, a damping factor of 0.9 would be used.

The internal impedance specifies the value of resistance and reactance for each phase. The initial speed deviation and angle specify the initial conditions of the two integrators in the mechanical part of the model. Finally, initial values for the three line currents can also be specified, which allows the simulation to be started in steady-state.

Note: These two blocks simulate exactly the same Simplified Synchronous machine model; the only difference is the way of entering the parameter units.

Inputs and Outputs

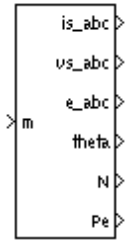
The first input of the Simplified Synchronous Machine block is the mechanical power supplied to the machine. This input can be a constant or the output of the Hydraulic Turbine and Governor block. The frequency of the voltage sources depends on the mechanical speed of the machine. The amplitude of these voltages is given by the second input of the block, which can be a constant or the output of a voltage regulator. If you use SI units, these two inputs should be in watts and volts phase-to-phase rms. If you use pu both inputs should be in pu

The first three outputs are the electrical terminals of the stator. The last output of the block is a vector containing the following 12 variables:

- 1-3: Line currents (flowing out of the machine) ia, ib, ic
- 4-6: Terminal voltages va, vb, vc
- 7-9: Internal voltages ea, eb, ec
- 10: Electrical angle theta
- 11: Rotor speed n
- 12: Electrical power Pe

These variables can be demultiplexed by using the special Simplified Synchronous Machine Demux block provided in the Machine library.

Simplified Synchronous Machine

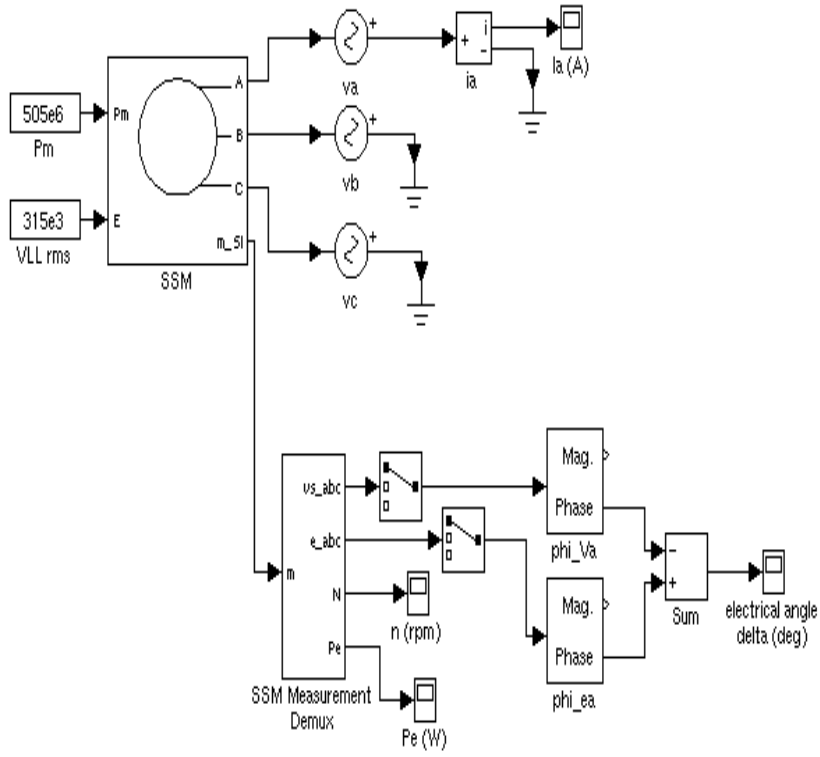


Assumptions

The electrical system of the Simplified Synchronous Machine consists solely of a voltage source behind a synchronous reactance and resistance. All the other self and mutual inductances of the armature, field and damping windings are neglected. The three voltage sources and RL impedance branches are Y-connected (3 wires). The load may or may not be balanced.

Example

The following example uses the Simplified Synchronous Machine block. In this example, the Simplified Synchronous Machine representing a 1000 MVA 315 kV equivalent source is connected to an infinite bus (three AC Voltage Source blocks) and is used as a synchronous generator. The R term is necessary to prevent series connection of the two current sources modeling the machine and the external inductance. The internal resistance of the machine is set to 0.02 pu, or 1.9845 ohms. Its inductance is set in such a way that the total impedance is 1 pu (L=263.15 mH). The inertia of the machine is 56290 kg.m^2. This example is included in the `psbsimplealt.mdl` file.



In this example, the machine has an initial speed deviation of 0.5%. The initial mechanical angle and the initial currents ia, ib, ic are set to zero. The power transfer between the machine and the bus is given by the following relation:

$$P_T = \frac{V_1 \times V_2}{X} \times \sin\delta$$

P_T = power transfer (500 MW)

V_1 = machine voltage (315 kV)

V_2 = bus voltage (315 kV)

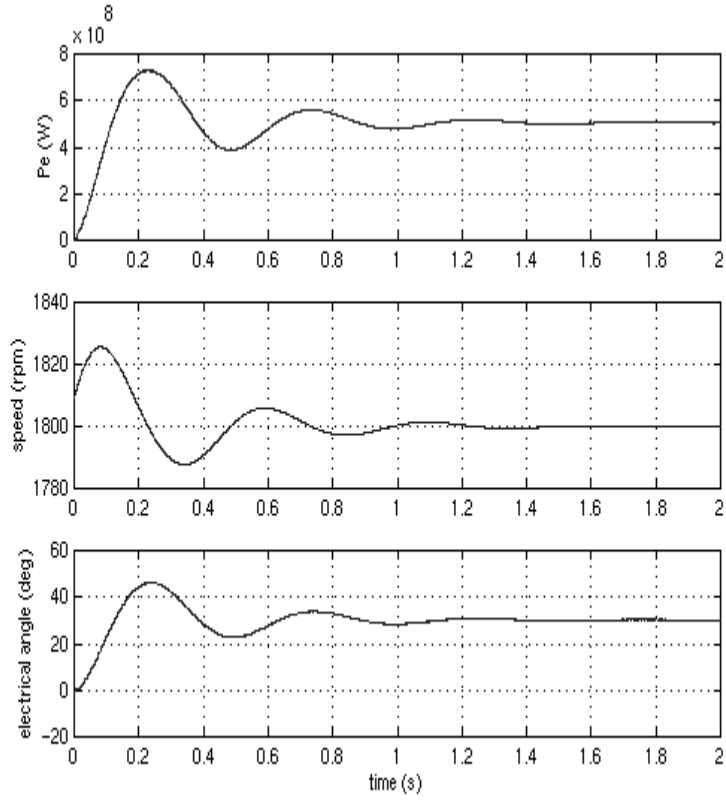
X = total reactance (263.15 mH) 120 x π

δ = electrical angle difference between machine internal voltage and terminal voltage

With the above parameters, the steady-state internal voltage is 30 degrees ahead of the terminal voltage (δ=+30 deg).

The machine is supplied with 505 MW of mechanical power in order to compensate for the machine resistive losses. The electrical angle δ is displayed as the phase difference between the internal and terminal voltage of phase A. With Simulation Parameters set as follows, the following results are obtained.

- Solver type: `ode15s`(stiff/NPF)
- Stop time: 2.0



The speed vs. time graph clearly shows that the machine is initially running at a speed of 1.005 pu (1809 rpm) and that speed stabilizes itself at its nominal value of 1800 rpm. As expected the electrical power supplied by the machine stabilizes at 500 MW and the electrical angle δ starts from zero and stabilizes at 30 degrees. The mechanical system is clearly under-damped, the damping factor being set to 0.3.

See Also

Synchronous Machine, Hydraulic Turbine and Governor

Surge Arrester

Purpose

Implement a Metal-Oxide surge arrester

Library

Elements Library

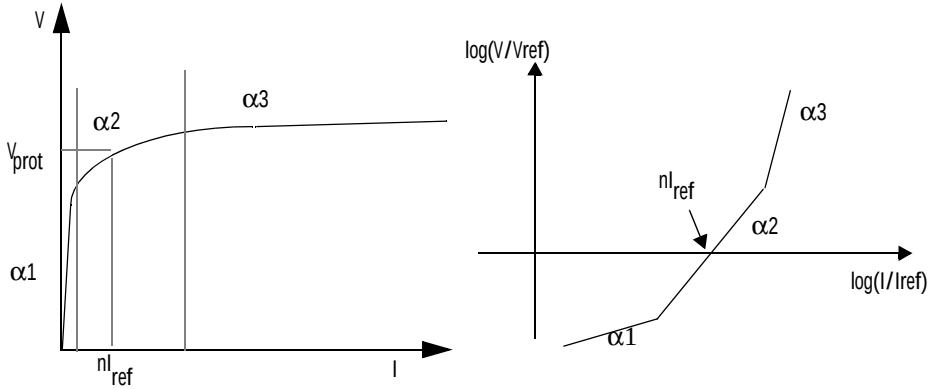
Description

The Surge Arrester block implements a highly nonlinear resistor used to protect power equipment against overvoltages. For applications requiring high power dissipation, several columns of metal-oxide discs are connected in parallel inside the same porcelain housing. The nonlinear V-I characteristic of each column of the surge arrester is modeled by a combination of three exponential functions of the form:

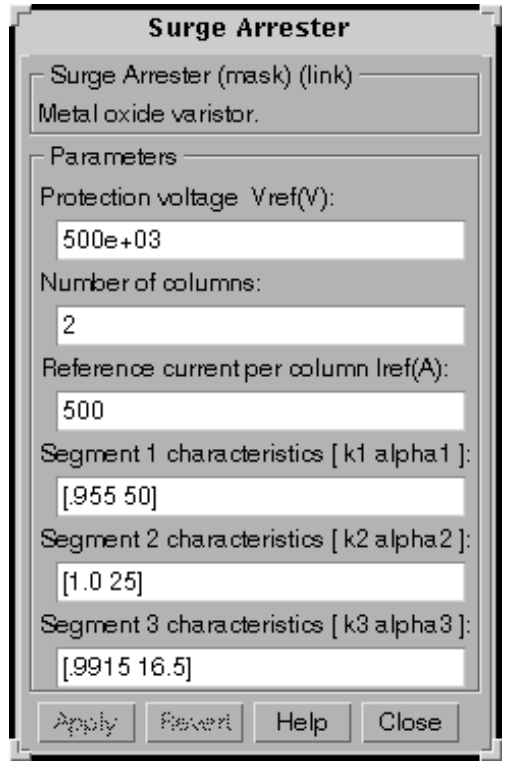
$$\frac{V}{V_{ref}} = K_i\left(\frac{I}{I_{ref}}\right)^{1/\alpha_i}$$

The protection voltage obtained with a single column is specified at a reference current (usually 500A or 1kA). Default parameters k and α given in the dialog box fit the average V-I characteristic provided by the main metal oxide arrester manufacturers. They do not change with the protection voltage. The required protection voltage is obtained by adding discs of zinc oxide in series in each column.

This V-I characteristic is graphically represented as follows (on a linear scale and on a logarithmic scale).



Dialog Box



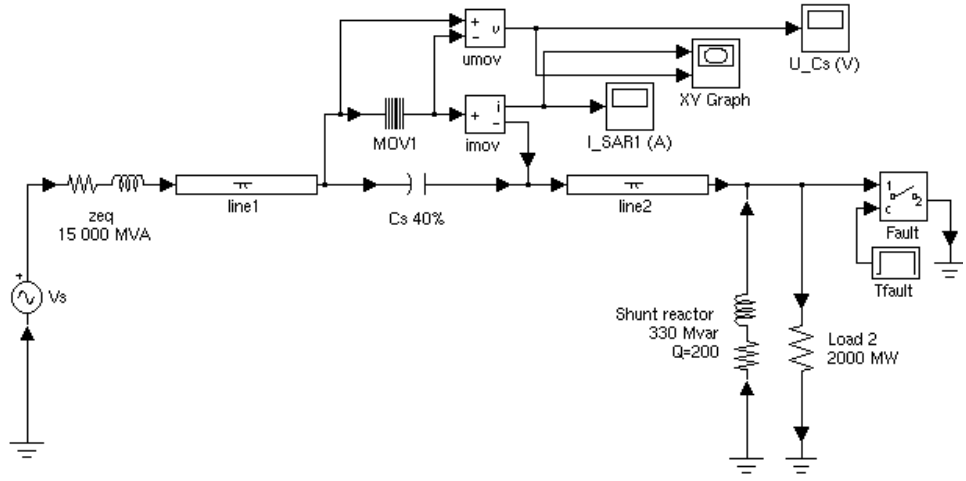
Restrictions

The Surge Arrester block is modeled as a current source driven by the voltage appearing across its terminals. Therefore, it cannot be connected in series with an inductor or another current source. As the Surge Arrester is highly nonlinear a stiff integrator algorithm must be used to simulate the circuit. `ode15s` and `ode23tb` solvers with default parameters usually give the best simulation speed. As this element contains no states, it will produce an algebraic loop because its current (input to state space model of the circuit) varies simultaneously with the voltage (output of state-space model). Simulink will signal an algebraic loop, but it usually solves the circuit without difficulty.

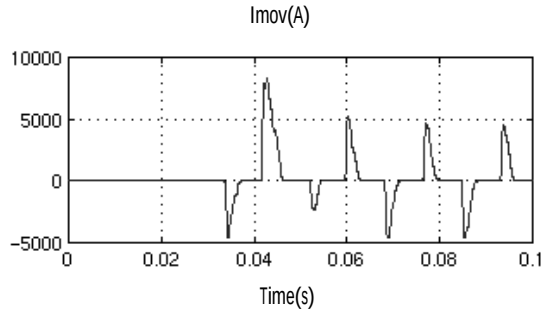
Example

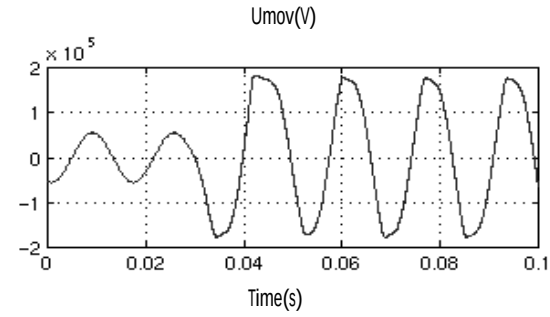
The example provided in the `psbarrester.mdl` demonstration file illustrates the use of metal oxide varistors (MOV) on a 735 kV series compensated network. Only one phase of the network is represented. The capacitor

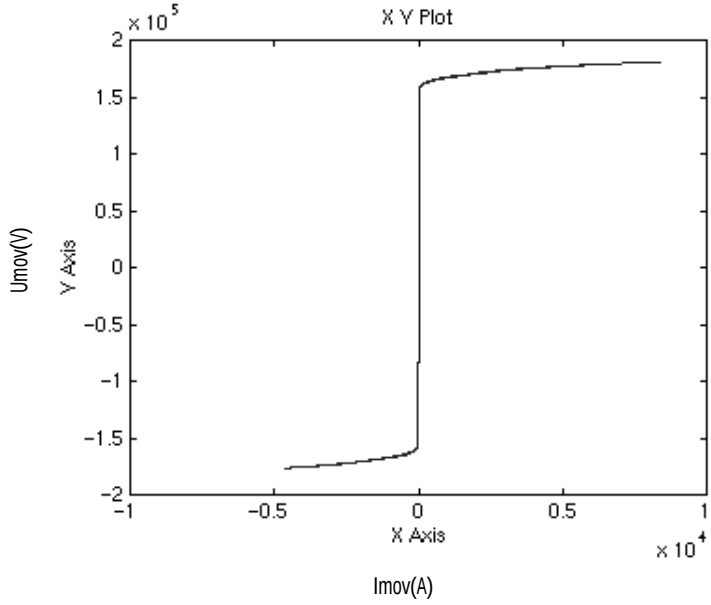
connected in series with the line is protected by a 30 column arrester. At t=0.03 seconds, a fault is applied at the load terminals. The current increases in the series capacitor and produces an overvoltage which is limited by the MOV. Then the fault is cleared at t=0.3 seconds.



At fault application, the resulting overvoltage makes the MOV to conduct. The waveforms displayed by Umov and Imov measurements as well as the V-I characteristic plotted by the X-Y scope are shown below.


Imov(A)
10000
5000
0
−5000
0
0.02
0.04
0.06
0.08
0.1
Time(s)


Umov(V)
× 10^5
2
1
0
−1
−2
0
0.02
0.04
0.06
0.08
0.1
Time(s)


X Y Plot
x 10^5
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
Umov(V)
Y Axis
−1
−0.5
0
0.5
1
X Axis
x 10^4
Imov(A)

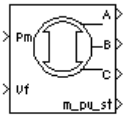
Purpose

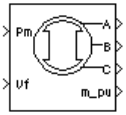
Model the dynamics of a three-phase salient-pole synchronous machine

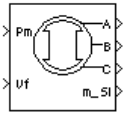
Library

Machines Library

Description





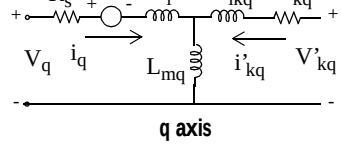


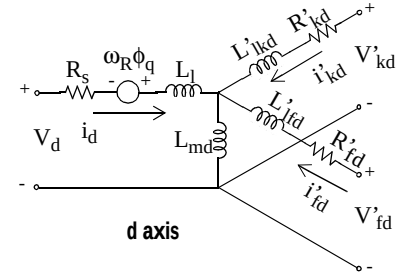
The Synchronous Machine block operates in generating or motoring modes. The operating mode is dictated by the sign of the mechanical power (positive for generating, negative for motoring). The electrical part of the machine is represented by a fifth-order state-space model and the mechanical part is the same as in the Simplified Alternator block.

The model takes into account the dynamics of the stator, field, and damper windings. The equivalent circuit of the model is represented in the rotor reference frame (qd frame). All rotor parameters and electrical quantities are viewed from the stator. They are identified by primed variables. The subscripts used are defined as follows:

- *d,q:* d and q axis quantity
- *R,s:* Rotor and stator quantity
- *l,m:* Leakage and mutual inductance
- *f,k:* Field and damper winding quantity

The electrical model of the machine is:





With the following equations:

$$V_d = R_s i_d + \frac{d}{dt}\varphi_d - \omega_R \varphi_q$$

$$V_q = R_s i_q + \frac{d}{dt}\varphi_q + \omega_R \varphi_d$$

$$V'_{fd} = R'_{fd} i'_{fd} + \frac{d}{dt}\varphi'_{fd}$$

$$V'_{kd} = R'_{kd} i'_{kd} + \frac{d}{dt}\varphi'_{kd}$$

$$V'_{kq} = R'_{kq} i'_{kq} + \frac{d}{dt}\varphi'_{kq}$$

$$\varphi_d = L_d i_d + L_{md}(i'_{fd} + i'_{kd})$$

$$\varphi_q = L_q i_q + L_{mq} i'_{kq}$$

$$\varphi'_{fd} = L'_{fd} i'_{fd} + L_{md}(i_d + i'_{kd})$$

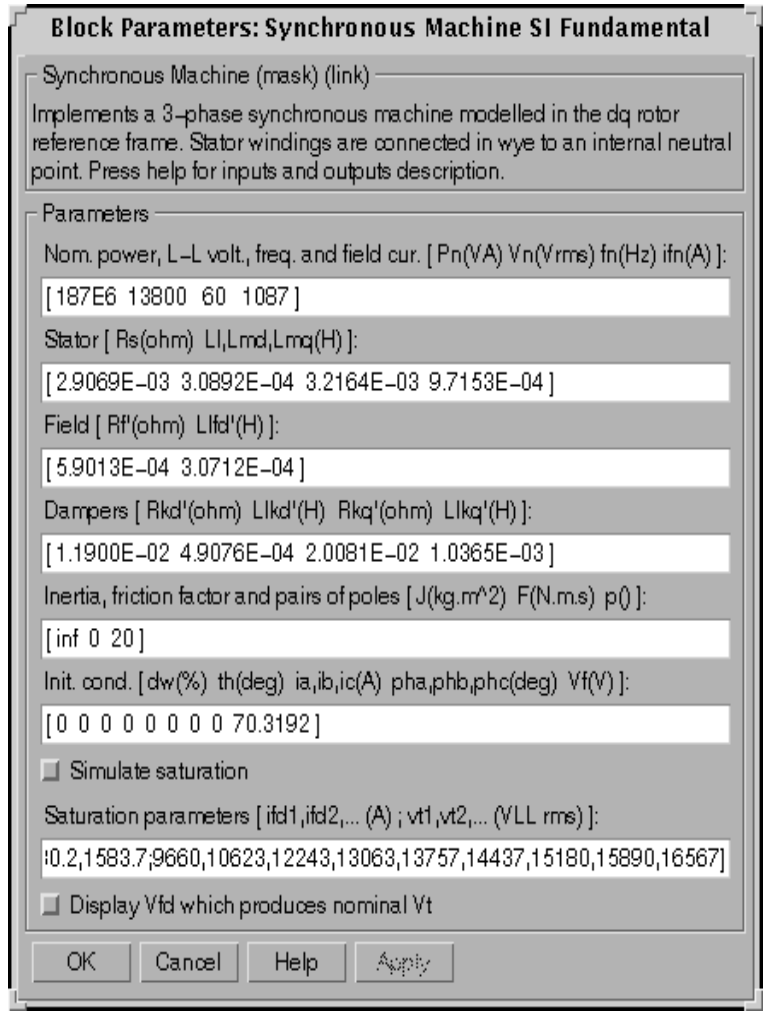
$$\varphi'_{kd} = L'_{kd} i'_{kd} + L_{md}(i_d + i'_{fd})$$

$$\varphi'_{kq} = L'_{kq} i'_{kq} + L_{mq} i_q$$

Parameters and Dialog Boxes

In the powerlib library, you can choose between three Synchronous Machine blocks to specify the parameters of the model.

Fundamental Parameters in SI Units



The first line of this dialog box is where you specify the nominal parameters:

- Total three-phase apparent power Pn, in VA
- RMS line-to-line voltage Vn, in Vrms

- Electrical frequency fn, in Hz
- Field current ifn, in A

The nominal field current is the current that produces nominal terminal voltage under no-load conditions. This model was developed with all quantities viewed from the stator. The nominal field current makes it possible to compute the transformation ratio of the machine, which allows you to apply the field voltage viewed from the rotor, as in reality. This also allows the field current, which is a variable in the output vector of the model, to be viewed from the rotor. If the value of the nominal field current is not known, you must enter zero. Since the transformation ratio cannot be determined in this case, you will have to apply the field voltage as viewed from the stator and the field current in the output vector will also be viewed from the stator.

You specify the following stator parameters on the second line of the dialog box:

- Resistance Rs, in Ω
- Leakage inductance LI, in H
- d axis mutual inductance Lmd, in H
- q axis mutual inductance Lmq, in H

You specify the field parameters next, on the third line:

- Resistance Rf′, in Ω
- Leakage inductance LIfd′, in H

The fourth line consists of the damper parameters:

- d axis resistance Rkd′, in Ω
- d axis leakage inductance LIkd′, in H
- q axis resistance Rkq′, in Ω
- Leakage inductance LIkq′, in H

The fifth line contains the mechanical parameters:

- Inertia J, in $kg.m^2$
- Viscous friction coefficient F, in N.m.s.
- Number of pairs of poles p

You enter the initial conditions (9) for the model on the sixth line:

- Speed deviation dw, in percent of nominal speed
- Electrical angle of the rotor th, in degrees
- Peak line currents ia, ib, ic, in A
- Phase angles pha, phb, phc, in degrees
- Initial field voltage Vf, in V

You can specify the initial field voltage in one of two ways. If you know the nominal field current (first line, last parameter) enter in the dialog box the initial field voltage in Volts DC viewed from the rotor. Otherwise, enter a zero as nominal field current, and specify the initial field voltage in Volts AC rms phase-to-phase viewed from the stator. The nominal field voltage viewed from the stator can be easily determined by checking the "Display Vfd which produces a nominal Vt" checkbox at the bottom of the dialog box. You can also determine the nominal field voltage viewed from the stator with a no-load test as follows:

- Enter an arbitrary value as the initial field voltage
- Connect a constant block with the same value to the field voltage input of the Synchronous Machine block
- Run a simulation
- Measure the terminal voltage obtained
- Correct the initial field voltage and start over until a satisfactory result is obtained.

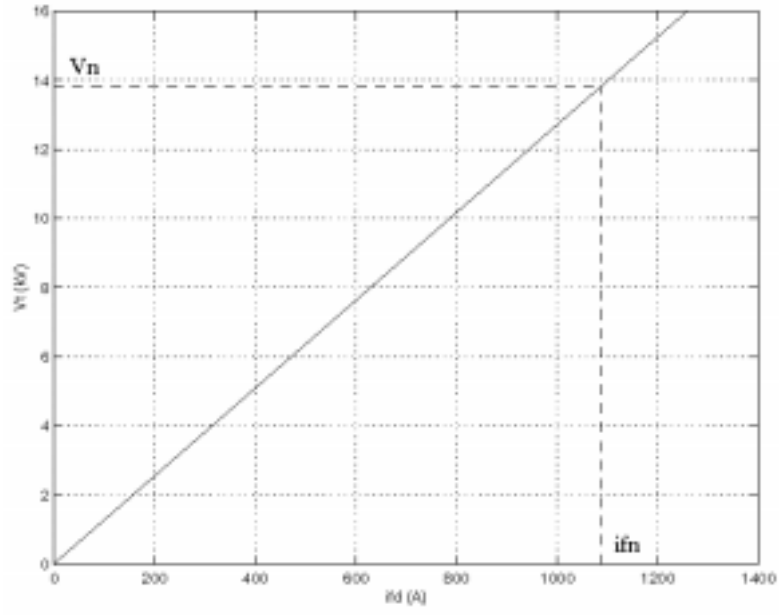
Finally, you can optionally specify saturation parameters for the Synchronous Machine to model magnetic saturation of the rotor and stator iron. Saturation is modeled by a nonlinear function, in this case a polynomial, using points on the no-load saturation curve. To simulate saturation, you must enter a 2 by n matrix, where n is the number of points taken from the saturation curve. The first row of this matrix contains the values of field currents while the second row contains values of corresponding terminal voltages. The first point (first column of the matrix) must correspond to the point where the effect of saturation begins. You must also check the Simulate saturation checkbox to simulate saturation. This checkbox allows you to enter the matrix of parameters for simulating the saturation. In simulations where you don't

simulate saturation, simply uncheck the box. You can later simulate saturation by checking the box without having to re-enter the entire matrix.

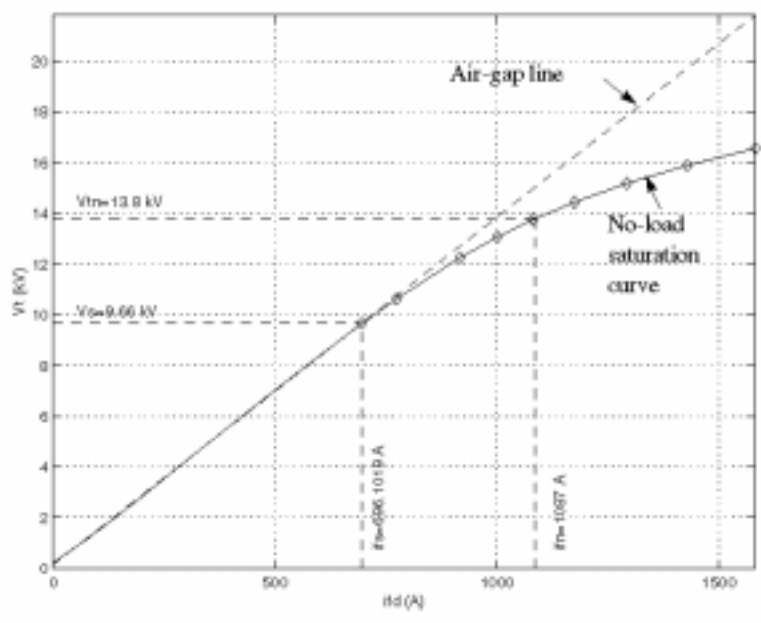
In this case the relationship between ifd and Vt obtained is linear (no saturation).

As an example, without saturation, a typical curve might be as shown.

Here ifn is 1087A and Vn is 13800 V line-to-line, which is also 11268 V peak line-to-neutral.



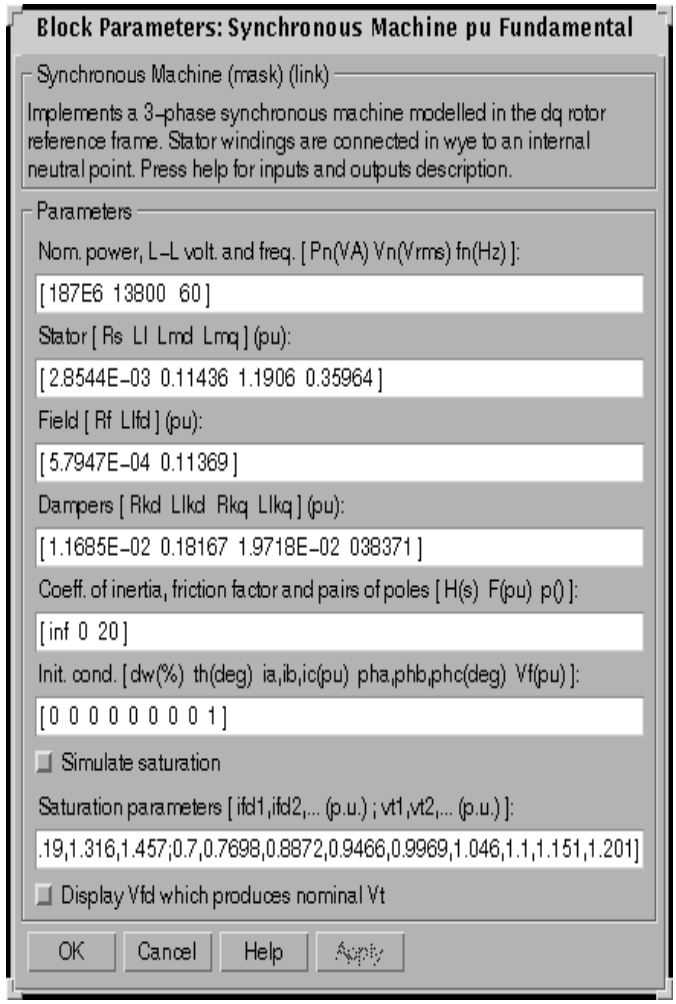
When saturation is modeled, a polynomial is fitted to the curve corresponding to the matrix of points you enter. The more points you enter, the better the fit to the original curve. The next figure illustrates this graphically (the diamonds are the actual points entered in the dialog box).



In this particular case, the following values were used:

- ifn = 1087 A
- ifd = [695.64, 774.7, 917.5, 1001.6, 1082.2, 1175.9, 1293.6, 1430.2, 1583.7] A
- Vt = [9660, 10623, 12243, 13063, 13757, 14437, 15180, 15890, 16567] V

Fundamental Parameters in pu



The first line of this dialog box is where you specify the nominal parameters:

- Total three-phase apparent power Pn, in VA
- RMS line-to-line voltage Vn, in Vrms
- Electrical frequency fn, in Hz

This line is identical to the first line of the fundamental parameters in SI dialog box, except that you don't specify a nominal field current. This value is not required here because we don't need the transformation ratio. Since rotor quantities are viewed from the stator, they are converted to pu using the stator base quantities derived from the three nominal parameters above.

The second, third and fourth line contain exactly the same parameters as in the previous dialog box (stator, field and dampers), but they are expressed here in pu instead of SI units.

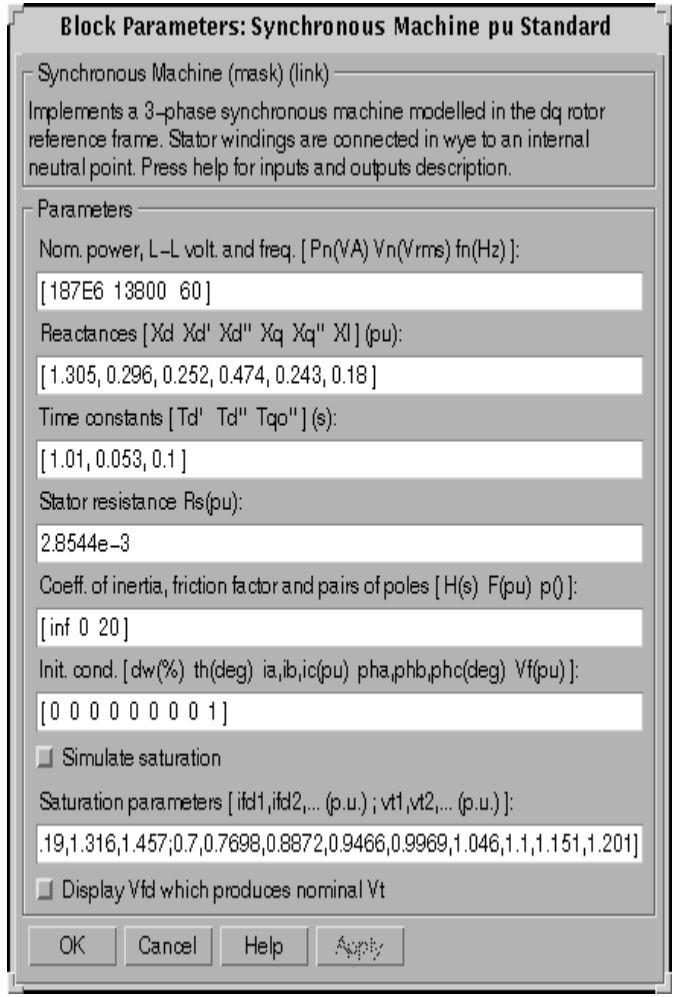
The fifth line contains the mechanical parameters, but expressed in pu:

- Inertia constant H, in seconds, where H is the ratio of energy stored in the rotor at nominal speed over the nominal power of the machine
- Viscous friction coefficient F, in pu.
- Number of pairs of poles p

The sixth line contains the initial conditions, as before, but the initial line currents and field voltage are expressed in pu instead of SI units.

The last line is where you specify the Saturation parameters, as before. However, the parameters must now be entered in per unit using the nominal field current, multiplied by the d axis mutual inductance, and nominal rms line-to-line voltage as base values for the field current, and terminal voltage, respectively.

Standard Parameters in pu



The first line of this dialog box is identical to the first line of the fundamental parameters in pu dialog box and contains nominal parameters.

You specify the machine's reactances on the second line (all in pu):

- d axis synchronous reactance Xd
- d axis transient reactance Xd'
- d axis subtransient reactance Xd"
- q axis synchronous reactance Xq
- q axis subtransient reactance Xq"
- Leakage reactance Xl

The third line contains the machine's time constants (all in s):

- d axis transient short-circuit time constant Td'
- d axis subtransient short-circuit time constant Td"
- q axis subtransient open-circuit time constant Tqo"

The fourth line is where you enter the stator resistance Rs, in pu

The last three lines are identical to the previous fundamental parameters in pu dialog box and consist of mechanical parameters, initial conditions and saturation parameters.

Note: These three blocks simulate exactly the same Synchronous machine model, the only difference is the way of entering the parameter units.

Inputs and Outputs

The units of inputs and outputs will vary according to which dialog box was used to enter the block parameters. For the non-electrical connections, there are two possibilities. If the first dialog box (fundamental parameters in SI units) is used, the inputs and outputs are in SI units (except for dw in the vector of internal variables, which is always in pu, and angle θ_e, which is always in degrees). If the second or third dialog boxes are used, the inputs and outputs are in pu.

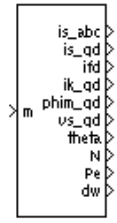
The first input is the mechanical power at the machine's shaft. In the generating mode, this input can be a positive constant or function or the output of a prime mover block (see the Hydraulic Turbine and Governor block). In the motoring mode, this input is usually a negative constant or function.

The second input of the block is the field voltage which can be supplied by a voltage regulator (see the Excitation System block) in the generating mode and is usually a constant in the motoring mode.

The first three outputs are the electrical terminals of the stator. The last output of the block is a vector containing 16 variables. They are, in order:

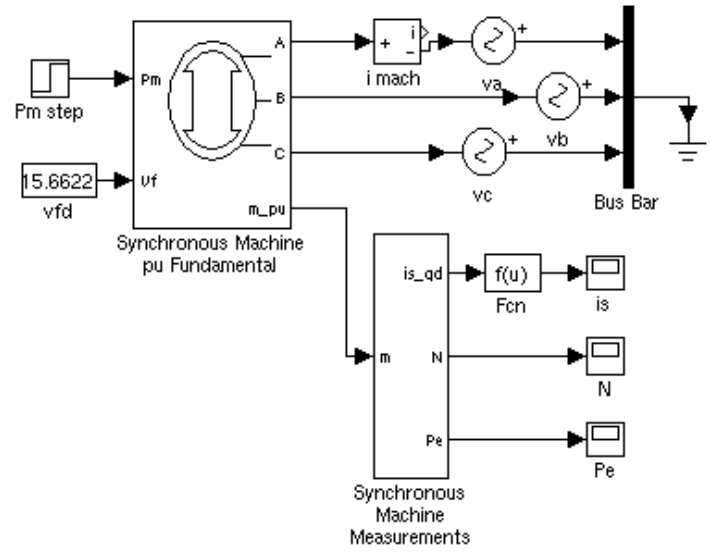
- 1-3: Stator currents (flowing out of machine) i_{sa}, i_{sb} and i_{sc}
- 4-5: q and daxis stator currents (flowing out of machine) i_q, i_d
- 6-8: Field and damper winding currents (flowing into machine) i_{fd}, i_{kq} and i_{kd}
- 9-10: q and d axis mutual fluxes φ_{mq},φ_{md}
- 11-12: q and d axis stator voltages v_q,v_d
- 13: Rotor electrical angle θ_e
- 14: Rotor speed ω_r
- 15: Electrical power Pe
- 16: Rotor speed deviation dw

These variables can be demultiplexed by using the special Synchronous Machine Demux block provided in the Machine library.



Example

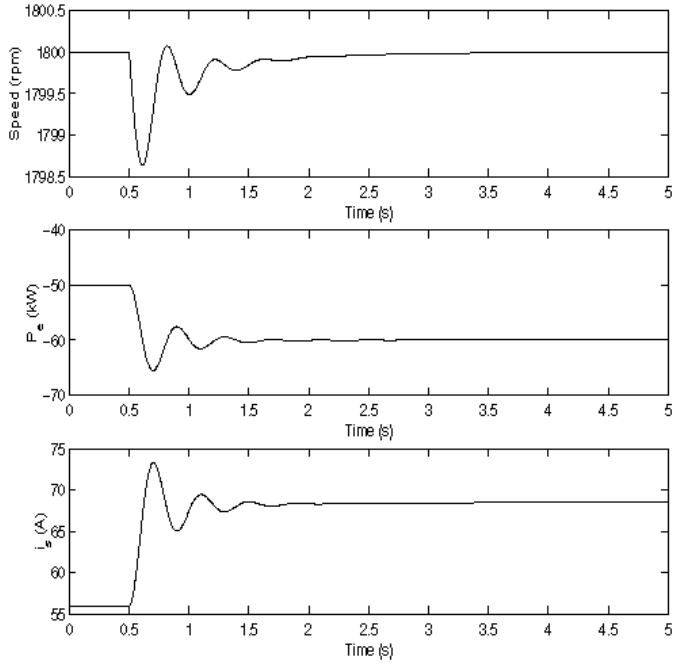
This example, available in the `psbsyncmachine.mdl` file, illustrates the use of the Synchronous Machine in motoring mode. The simulated system consists of an industrial grade synchronous motor (150 HP, 440V) connected to an infinite bus. After the machine reaches a stable speed, the load (mechanical power) is changed from 50 kW to 60 kW. The initial conditions are set in such a way that the simulation starts in steady-state. Open the simulink diagram by typing `psbsyncmachine`.



Set the Simulation parameters as follows:

- Stop time: 5
- Solver options: Use default settings, except for Max. step size, which you must set to 0.005

Run the simulation and observe the speed, power and current of the motor.



Because this is a four pole machine, the nominal speed is 1800 rpm. The initial speed is 1800 rpm as prescribed (top graph). The load passes from 50 kW to 60 kW at t=0.5 s. The machine then oscillates before stabilizing to 1800 rpm.

Now, look at the electrical power (middle graph). Since we are in motoring mode, the machine absorbs power and Pe is negative. As expected, the power starts at -50 kW until the load is changed at t=0.5 seconds, at which point the power oscillates before settling at -60 kW.

Finally, look at the stator current i_s. As expected, the current starts with the value corresponding to a three-phase power of 50 kW (56 A), before oscillating and settling to the value corresponding to a 60 kW load (68.5 A).

References

[1] Krause, P.C., *Analysis of Electric Machinery*, section 12.5, McGraw-Hill, 1986.

[2] Kamwa, I., et al., "Experience with Computer-Aided Graphical Analysis of Sudden-Short-Circuit Oscillograms of Large Synchronous Machines", Vol.10, *IEEE Transactions on Energy Conversion*, No.3, September 1995.

See Also

Simplified Synchronous Machine, **Excitation System**, **Hydraulic Turbine and Governor**

Thyristor

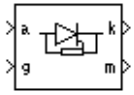
Purpose

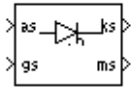
Implement a thyristor model

Library

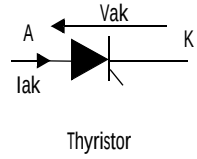
Power Electronics Library

Description

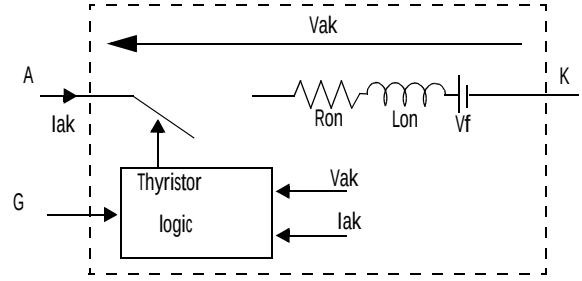




The Thyristor block is a semiconductor device that can be turned on via a gate signal. The thyristor is modeled as a resistor (Ron), inductor (Lon), and DC voltage source (Vf), connected in series with a switch. The switch is controlled by a logical signal depending on the voltage Vak, the current Iak and the gate signal (G).

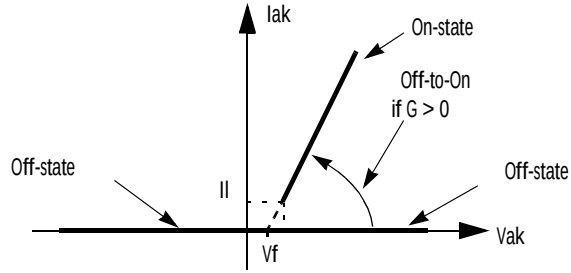


A: anode; K: cathode; G: gate.



The Thyristor block also contains a series Rs-Cs snubber circuit, which is usually connected in parallel with the thyristor.You can specify a snubber which is purely resistive (Cs = Inf) or purely capacitive (Rs=0). If you specify either Rs=Inf or Cs=0, the snubber is eliminated and it disappears on the thyristor icon.

The static VI characteristic of this model is shown in the figure that follows.



The thyristor turns on when the anode-cathode voltage is greater than Vf and a positive pulse signal is present at the gate input (G>0). The pulse height must be greater than zero and last long enough to allow the thyristor anode current to become larger than the latching current Il.

The thyristor turns off when the current flowing in the device becomes zero (Iak=0) and a negative voltage appears across the anode and cathode for at least a period of time equal to the turn-off time Tq. If the voltage across the device becomes positive within a period of time less than Tq, the device will turn on automatically even if the gate signal is low (G = 0) and the anode current is less than the latching current. Furthermore, if during turn on, the device current amplitude stays below the latching current level specified in the dialog box, the device turns off after the gate signal level becomes low (G = 0).

The turn-off time Tq represents the carrier recovery time: it is the time interval between the instant the anode current has decreased to zero and the instant when the thyristor is capable of withstanding positive voltage Vak without turning on again.

Simplified Model and Detailed Model

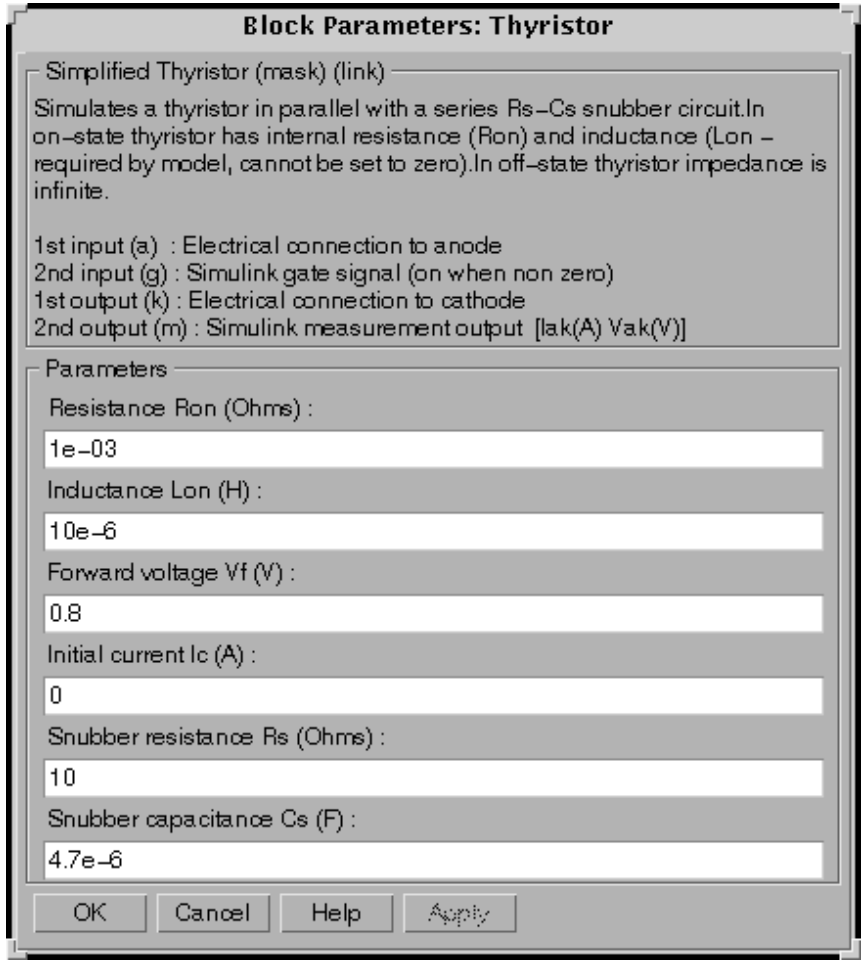
To optimize simulation speed, two models of thyristors are available: the simplified model and the detailed model. For the simplified thyristor model, the latching current Il and recovery time Tq are assumed to be zero.

Enter the thyristor parameters Ron, Lon, and Vf and the Rs-Cs snubber parameters in the dialog box. Due to modeling constraints explained below, the inductance Lon cannot be set to zero.

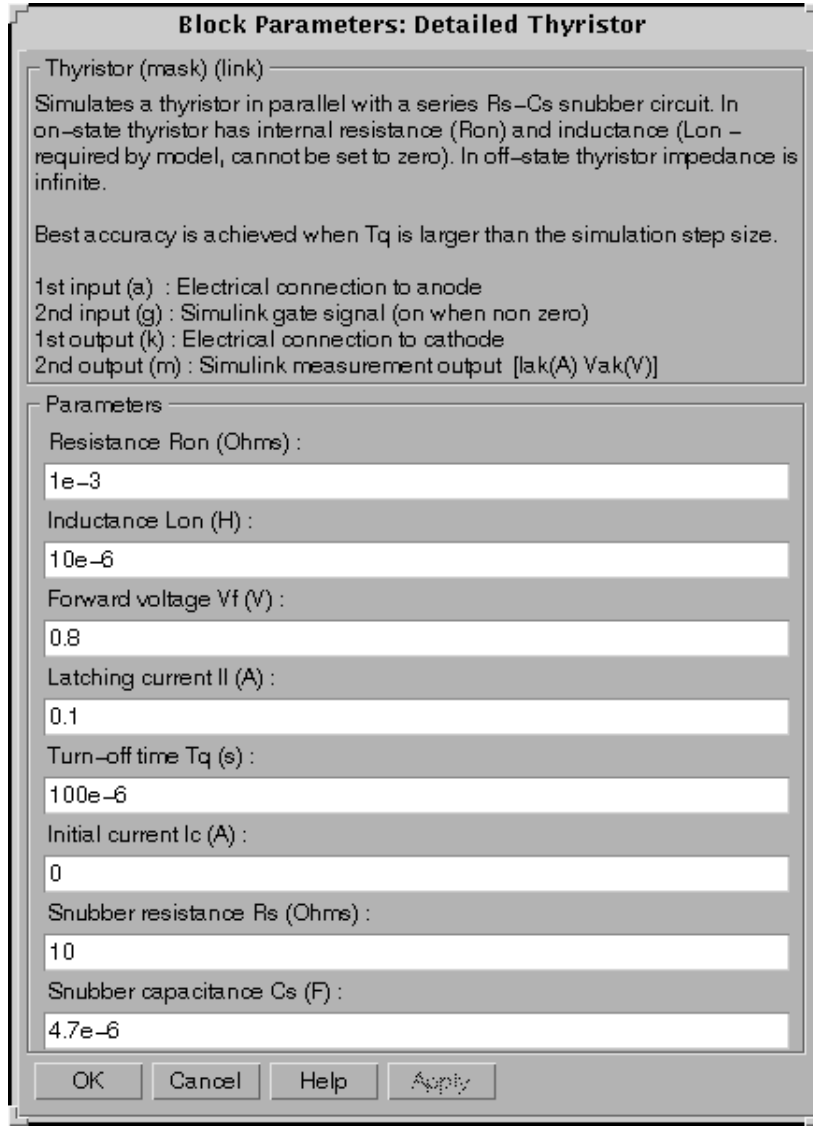
The initial current Ic flowing in the thyristor is usually set to zero so that the simulation is started with the thyristor blocked. However, you may specify an Ic value corresponding to a particular state of the circuit. In such a case all states of the linear circuit must be set accordingly. Initializing all-states of a power-electronic converter is a complex task. Therefore, this option is useful only with simple circuits.

For applications where the thyristor is used in general purpose rectifier circuits fed by a voltage source at a frequency of 50 or 60 Hz, the typical parameters values specified in the dialog box can be used.

Dialog Boxes Simplified Thyristor Dialog Box



Detailed Thyristor Dialog Box



Thyristor

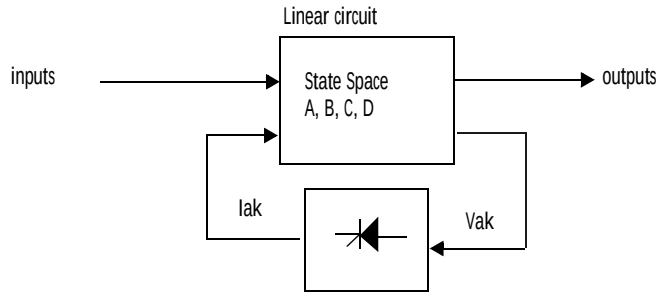
Inputs and Outputs

The thyristor icon consists of two inputs and two outputs. The first input and output are the thyristor terminals connected respectively to anode (a) and cathode (k). The second input (g) is a Simulink logical signal applied to the gate (G). The second output (m) is a Simulink measurement output vector [Iak, Vak] returning the thyristor current and voltage.

Assumptions and Limitations

The Thyristor block implements a macro-model of the real thyristor. It does not take into account either the geometry of the device or complex physical processes that model the behavior of the device [1-2]. The forward breakover voltage and the critical value of the derivative of the reapplied anode-cathode voltage are not considered by the model.

In the Simulink representation, the thyristor is modeled as a nonlinear element interfaced with the linear circuit as shown below.



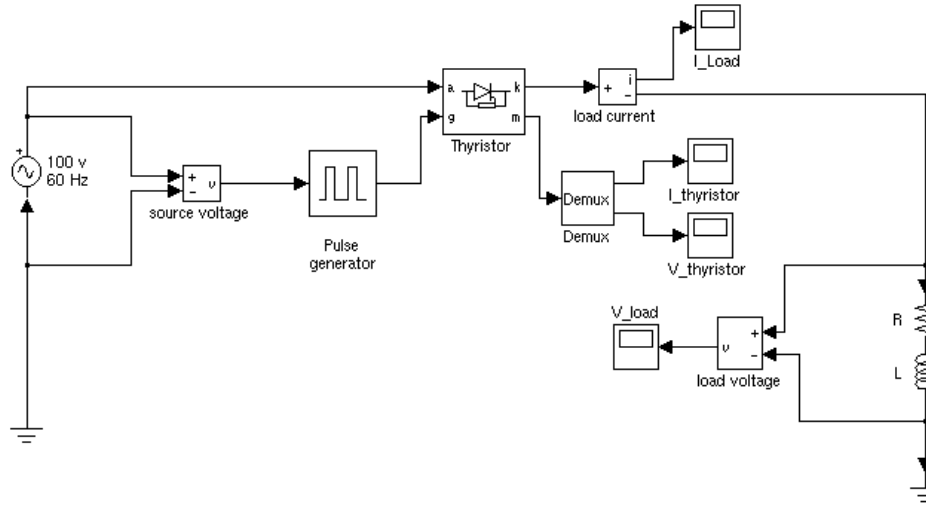
Therefore, in order to avoid an algebraic loop, the thyristor inductance Lon cannot be set to zero. Each thyristor adds an extra state to the electrical circuit model. As the thyristor is modeled as a current source, it cannot be connected in series with an inductor, a current source or an open circuit unless a snubber circuit is used.

You must use a stiff integrator algorithm to simulate circuits containing thyristors. `ode23tb` and `ode15s` usually give best simulation speed.

Example

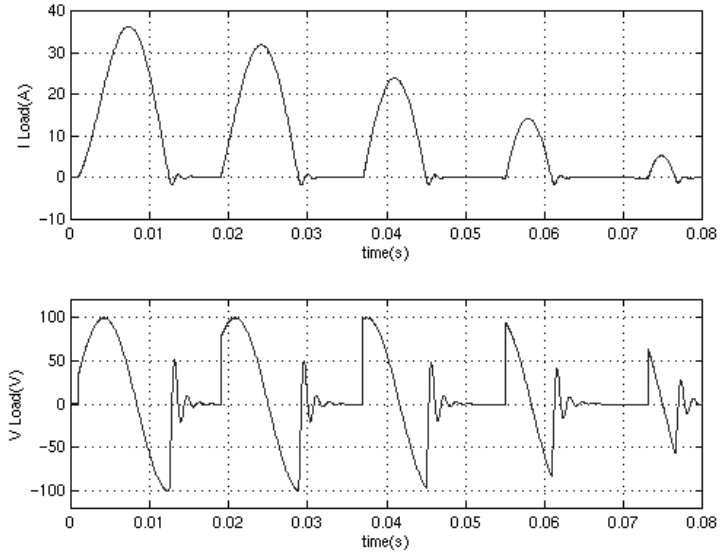
Single pulse thyristor rectifier feeding a RL load. The gate pulses are obtained from a pulse generator synchronized on the source voltage. The circuit is available in the `psbthyristor.mdl` file. The following parameters are used:

R=1Ω; L=10mH; Thyristor block: Ron=0.001 Ω, Lon=1e-5 H, Vf=0.8 V, IC=0 A, Rs=20 Ω, Cs=4e-6 F. Simulation parameters: `ode15s`; Relative tolerance: 1e-3; Absolute tolerance: 1e-3.

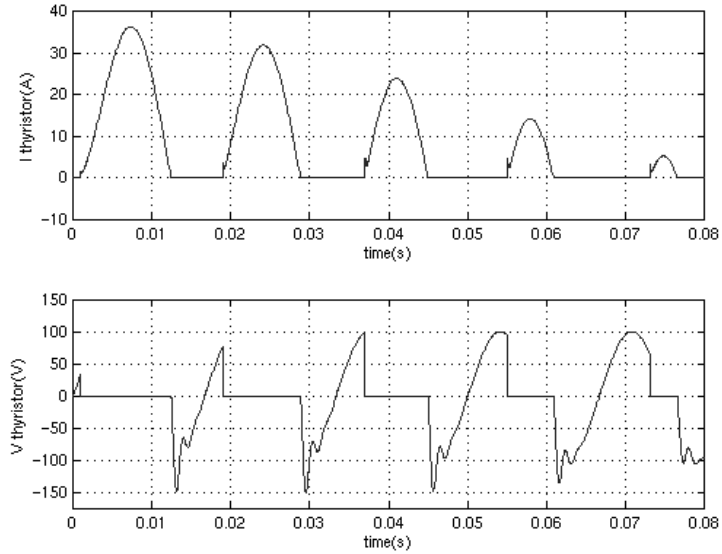


The firing angle is varied by the Pulse Generator block synchronized on the voltage source. Run the simulation and observe the load current and voltage, and the thyristor current and voltage.

Load current and voltage:



Thyristor current and voltage



References

[1] Rajagopalan, *V., Computer-Aided analysis of Power Electronic Systems*, Marcel Dekker, Inc., New York,1987.

[2] Mohan, N., *Power Electronic, Converters, Applications and Design*, John Wiley & Sons, Inc., New York, 1995.

See Also

Diode, **Mosfet**, **GTO**, **Ideal Switch**

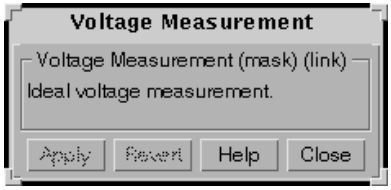
Voltage Measurement

Purpose Measure a voltage in a circuit

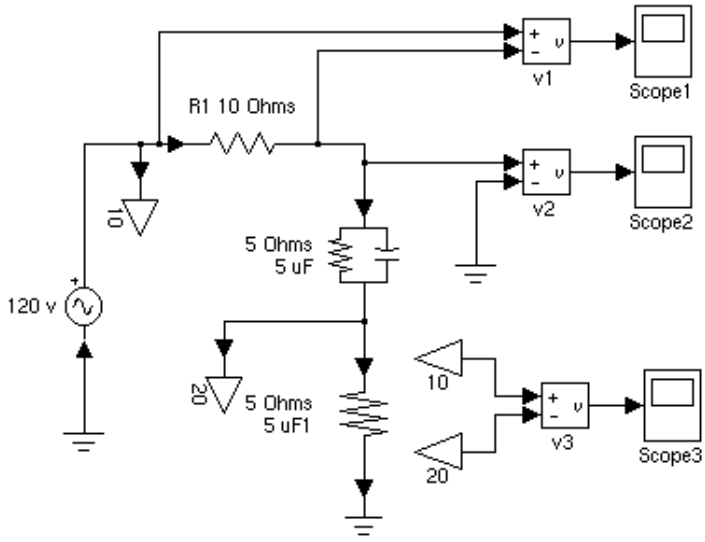
Library Measurements Library

Description The Voltage Measurement block is used to measure the instantaneous voltage between two electric nodes. The output is a Simulink signal that can be used by other Simulink blocks.

Dialog Box



Example The following example uses three Voltage Measurement blocks to read voltages. This example is available in the `psbvoltmeasure.mdl` file.



See Also **Current Measurement**

Index

S

T

V