PSpice MATLAB Interface User Guide

Product Version 23.1
September 2023

Contents

1 Introduction

PSpice is a simulation program that simulates any mix of analog and digital devices. Used with design entry tools, OrCAD Capture or Design Entry HDL, you can think of PSpice as a software-based breadboard of your circuit that you can use to test and refine your design before manufacturing the physical circuit board (PCB) or integrated circuit (IC).

PSpice - MATLAB interface extends the PSpice visualization capabilities to the next level. In just one click, the interface allows you to export all or selected trace data to MATLAB, where you can analyze and visualize the data in an enhanced way. You can visualize simulation results on various different plots, such as Polar Plots and 3D plots. You can also customize the processing on waveform using customized MATLAB® scripts on the exported trace data.

PSpice Simulink Co-Simulation, which is co-simulation of PSpice and Mathworks® Simulink, combines the best-in-class software tools to provide unmatched design simulation environment for electrical and physical system together. PSpice Simulink Co-Simulation Interface allows you to substitute electronic blocks in PSpice, while the rest of the design is simulated using Simulink. As a result, you can now use a single prototype to co-simulate the electrical and mechanical systems. Co-simulation environment allows to simulate whole system with more realistic element models before trial manufacturing.

Understanding Various MATLAB-Related Flows

This guide is categorized in two parts: Part A and Part B. The following table explains the various flows that this guide covers:

Flow Covered	Description
PSpice - MATLAB Flow (Page 11)	PSpice - MATLAB interface is a PSpice-driven flow, where you can export the trace data to MATLAB using PSpice. MATLAB is only required for advanced plotting and visualization. For using measurement functions and MATLAB function in the behavioral expression, MATLAB works in the background and you will not have to launch the MATLAB user interface. For more details on how to start with PSpice - MATLAB interface for Advance Waveform Analysis, see the following chapters: ❑ Introducing the Advanced Waveform Analysis on page 13 ❑ Setting Up the PSpice - MATLAB Flow for Advanced Waveform Analysis on page 15 ❑ Working with Advanced Waveform Analysis Flows on page 19 ❑ Accessing Trace Data from the MATLAB Variable on page 47
MATLAB - PSpice flow (Page 55)	PSpice Simulink Co-Simulation Interface is a MATLAB-driven flow, where you can drive the PSpice block from MATLAB. For more details about how to start with the PSpice Simulink Co-Simulation interface, see the following chapters: ❑ Introducing PSpice Simulink Co-Simulation on page 57 ❑ Creating a Schematic on page 61 ❑ Interfacing PSpice Designs in MATLAB on page 89 ❑ Analyzing Simulink Models on page 97 ❑ Options and Data Collection in Simulation Settings on page 99

Part A: PSpice - MATLAB Interface for Advanced Waveform Analysis

The chapters in part A introduce you to PSpice - MATLAB interface and describes the various tools involved in working with PSpice -MATLAB interface for advance waveform analysis. The chapters also describe how to do advanced waveform analysis in PSpice project using MATLAB.

2

Introducing the Advanced Waveform Analysis

This chapter introduces the new PSpice - MATLAB interface and lists the requirements to run various flows for Advanced Waveform Analysis.

- Advanced Waveform Analysis on page 14
- Software Requirements for the Advanced Waveform Analysis on page 14

Advanced Waveform Analysis

PSpice has been used to simulate analog-only, mixed analog/digital, and digital-only circuits for a long time. Now, PSpice capabilities as a simulation software has been enhanced further with the integration of new PSpice - MATLAB Interface.

Various different visualization and plotting capabilities of MATLAB use plotting functions and measurement functions in the PSpice-MATLAB interface to generate new plots and graphs using the PSpice trace data.

Software Requirements for the Advanced Waveform Analysis

You need to have the following combination of The Mathworks and Cadence® OrCAD® and Allegro® products installed on your system:

1. The Mathworks products

   MATLAB R2019b or onwards

2. Cadence OrCAD products or Cadence Allegro products.

   Install one of the following design entry tools:

   ❑ Capture or Capture CIS

   ❑ Design Entry HDL

   Install the following simulator:

   ❑ PSpice A/D

3. To be able to use the PSpice - MATLAB Interface, ensure that you have license for one of these products in your license file.

   ❑ OrCAD PSpice Designer Plus

   ❑ Allegro PSpice System Designer

3

Setting Up the PSpice - MATLAB Flow for Advanced Waveform Analysis

This chapter explains how to set up the PSpice - MATLAB flow for Advance Waveform Analysis. It also covers details about running the MATLAB engine.

Setting Up the MATLAB Path

Before you use the PSpice - MATLAB flow, you need to:

1. Register MATLAB as an automation server
2. Set MATLAB installation path

Important

- ❑ The MATLAB path needs to be set only once in the Set MATLAB Path window. Once the path is set, you do not have to set it again.
- ❑ Update the MATLAB path if you have installed a newer supported version of MATLAB or want another MATLAB installation.

Register MATLAB as an automation server

To register MATLAB as an automation server:

1. Start MATLAB with the *Run as administrator option*.

   The MATLAB Command Window opens.
2. Run the command: `regmatlabserver`

OR

1. Open the Windows command prompt in the administrator mode.
2. Go to the installation directory containing the new version you want to use. For example: `<MATLAB>\<matlab_version>\bin`
3. Run the command: matlab -regserver

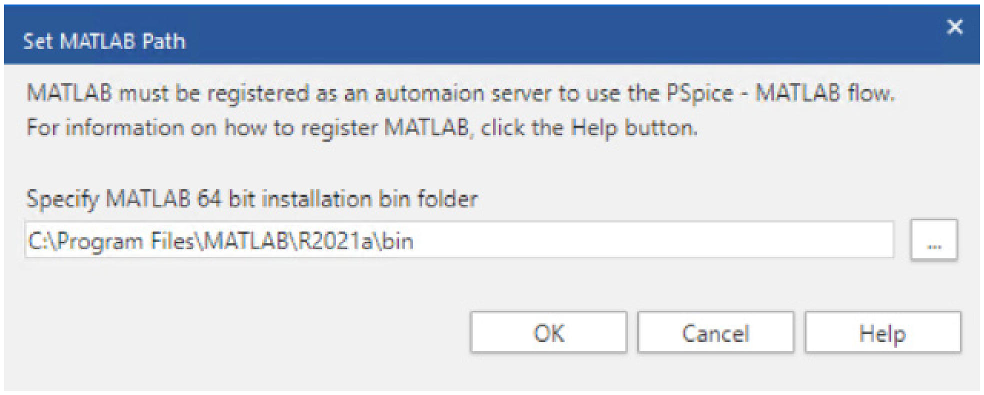
   A minimized MATLAB Command Window appears.
4. Open the MATLAB Command Window and then close it.
5. Restart PSpice.

Set MATLAB installation path

To set MATLAB installation path:

1. Open PSpice.
2. Choose *Tools – MATLAB – Set MATLAB Path*.

   The MATLAB path should end at the `bin` folder, such as `<MATLAB_installation>\bin`



Set MATLAB Path

MATLAB must be registered as an automaion server to use the PSpice - MATLAB flow.
For information on how to register MATLAB, click the Help button.

Specify MATLAB 64 bit installation bin folder

C:\Program Files\MATLAB\R2021a\bin

Running the MATLAB Engine

You need to start the MATLAB engine for a faster first launch of the MATLAB command window. If you have not started the MATLAB engine before exporting the trace data, the first launch of the MATLAB command window is delayed by few seconds. To run the MATLAB engine:

1. Launch PSpice.
2. Select *Tools – MATLAB – Start MATLAB Engine*.

Important

You need to specify the MATLAB path before you can access other commands in this menu. The *Start MATLAB Engine* option will be displayed only if you have specified the MATLAB path.

4

Working with Advanced Waveform Analysis Flows

The following flows are targeted for those users, who have electronics components as large part of a design. For all the following flows, simulation will happen in PSpice only and you will not be required to see MATLAB until you want to export PSpice trace data to MATLAB for specialized plotting.

The PSpice - MATLAB flows covered in this chapter are:

- Using MATLAB Visualization Capabilities in PSpice on page 20
- Calling the MATLAB function in PSpice on page 38
- Using the MATLAB-Based Measurement Function in PSpice on page 43

Using MATLAB Visualization Capabilities in PSpice

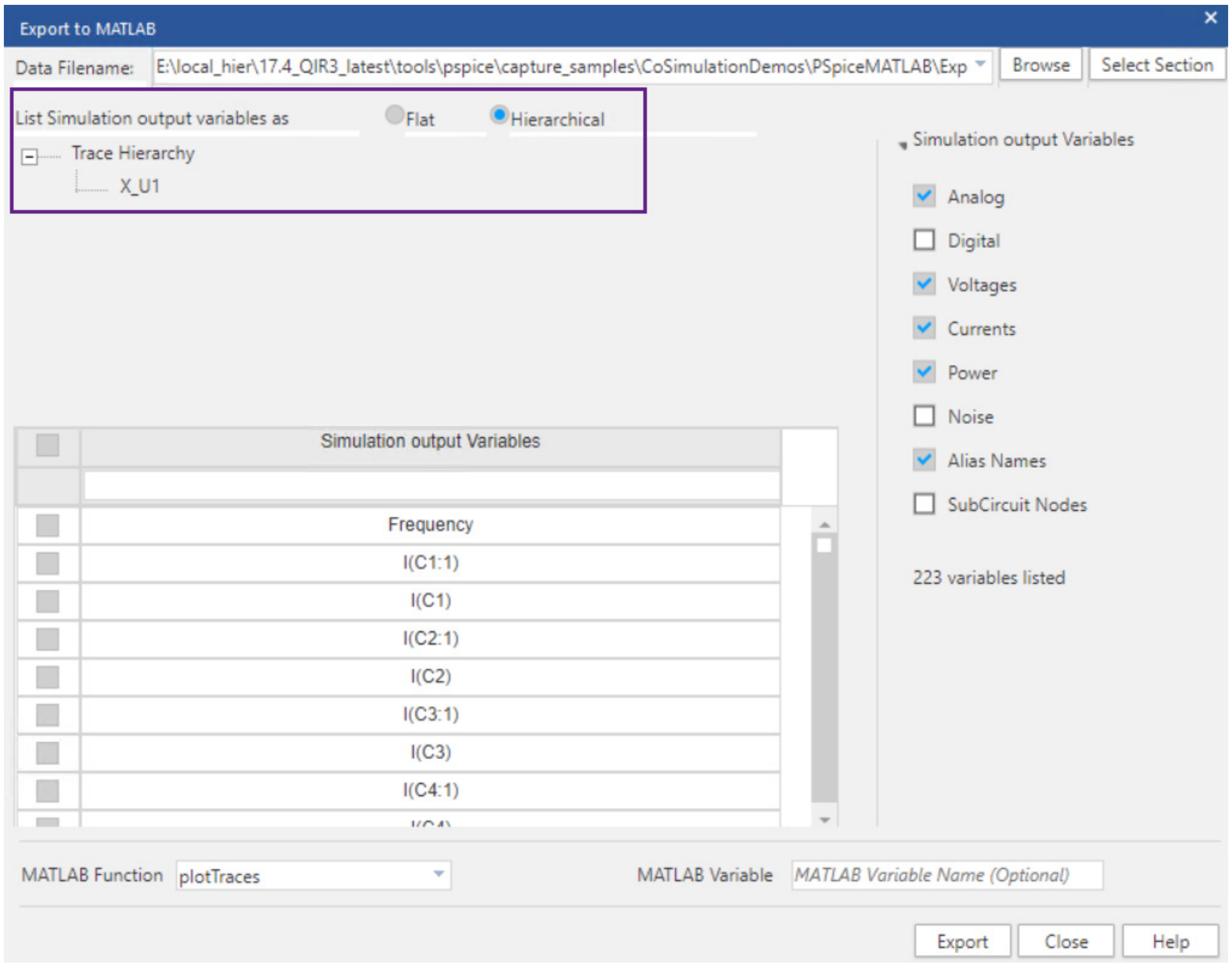
The new PSpice - MATLAB interface provides some of the best capabilities of MATLAB for specialized plotting. You can export the PSpice trace data to MATLAB and generate various plots, such as polar plots and 3D Plots.

Before you start using the MATLAB visualization capabilities in PSpice, ensure that you have set the MATLAB path and started the MATLAB engine. To know more about the setting the MATLAB path and starting the MATLAB engine, see the Setting Up the PSpice - MATLAB Flow for Advanced Waveform Analysis chapter.

In the Export to MATLAB window, you can do one or more of the following:

- Select one or more traces that you want to export to MATLAB for advanced plotting and visualization.

- Select the simulation output variable as `Flat` or `Hierarchical`. The following screen shot illustrates the hierarchical structure of the simulation output variables:



Export to MATLAB
Data Filename: E:\local_hier\17.4_QIR3_latest\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\Exp
Browse
Select Section
List Simulation output variables as
Flat
Hierarchical
Trace Hierarchy
X_U1
Simulation output Variables
Frequency
I(C1:1)
I(C1)
I(C2:1)
I(C2)
I(C3:1)
I(C3)
I(C4:1)
Export
Close
Help

List Simulation output variables as: Flat [✓ Hierarchical]

Simulation output Variables:
- [✓] Analog
- [ ] Digital
- [✓] Voltages
- [✓] Currents
- [✓] Power
- [ ] Noise
- [✓] Alias Names
- [ ] SubCircuit Nodes

223 variables listed

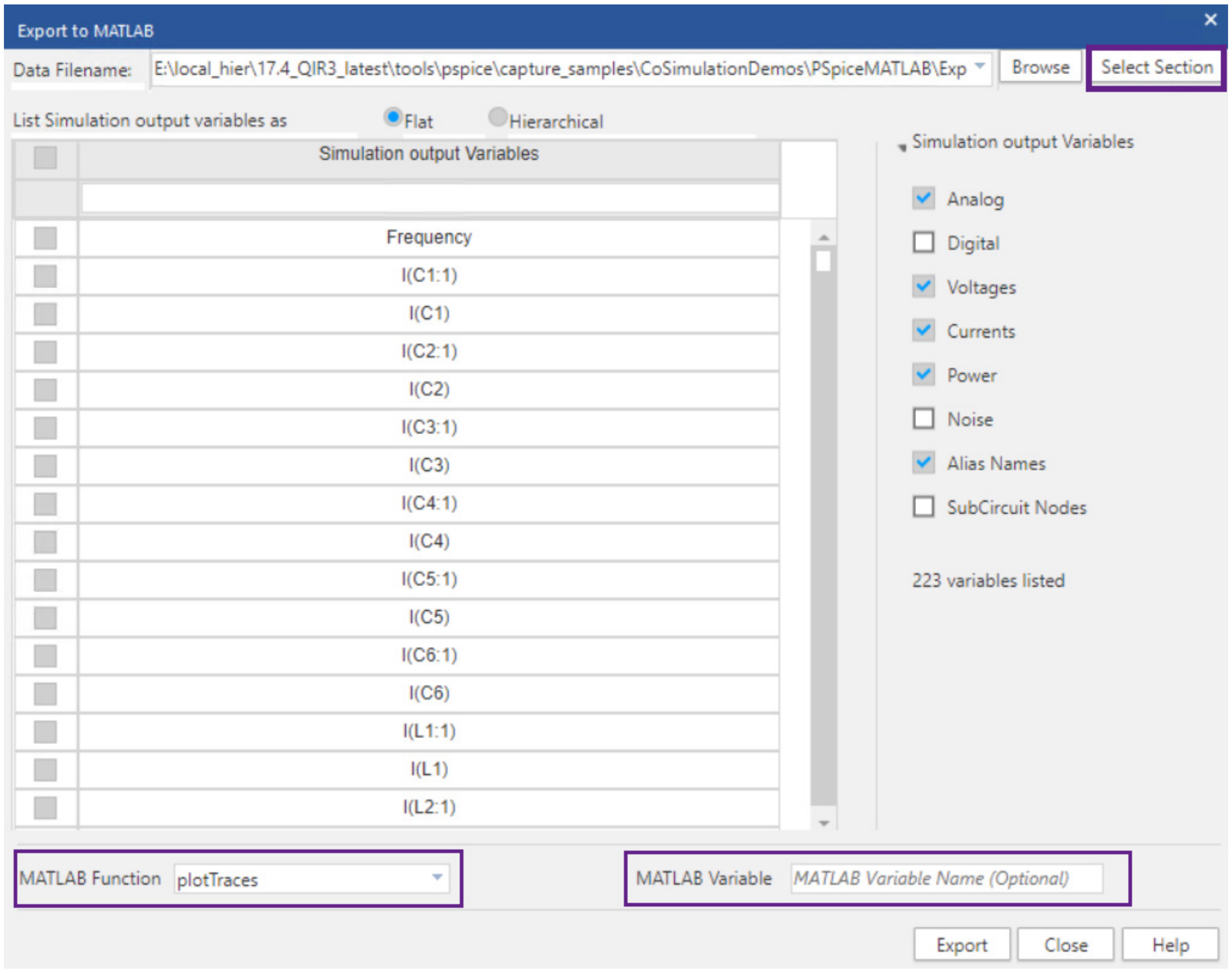
MATLAB Function: plotTraces

MATLAB Variable: *MATLAB Variable Name (Optional)*

- Use the default MATLAB-based function, `plotTraces`, to plot traces with respect to time in a figure panel.

- Provide a different name, such as TRAN, MyVAR, to the generate a MATLAB variable instead of the default variable created (`PSpiceData_<instance_number>`). To do that, specify the variable name in the *MATLAB Variable* field.

  Note: If you name the MATLAB variable using the *MATLAB Variable* field, ensure that the first character of this variable is an alphabet.



Data Filename: E:\local_hier\17.4_QIR3_latest\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\Exp

List Simulation output variables as [✓ Flat] Hierarchical

Simulation output Variables
Frequency
I(C1:1)
I(C1)
I(C2:1)
I(C2)
I(C3:1)
I(C3)
I(C4:1)
I(C4)
I(C5:1)
I(C5)
I(C6:1)
I(C6)
I(L1:1)
I(L1)
I(L2:1)

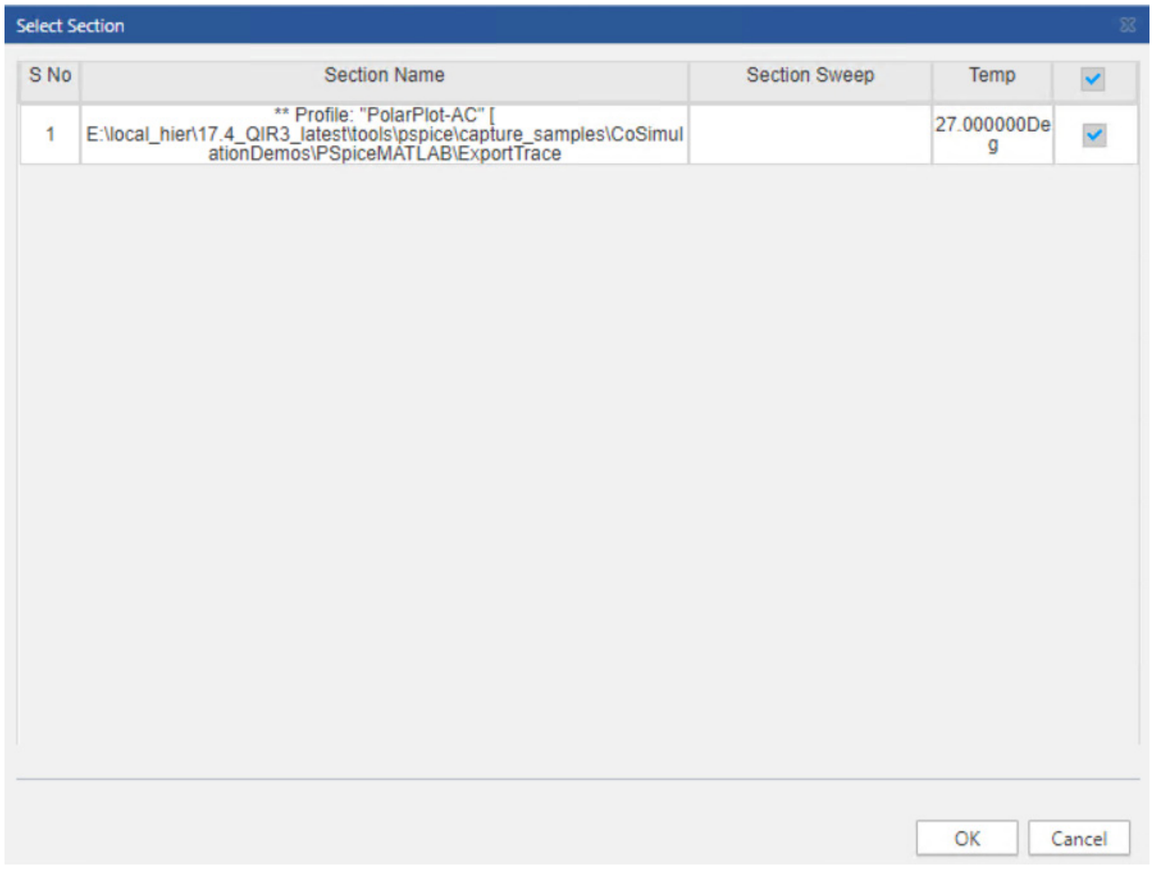
Simulation output Variables
- ✓ Analog
- ☐ Digital
- ✓ Voltages
- ✓ Currents
- ✓ Power
- ☐ Noise
- ✓ Alias Names
- ☐ SubCircuit Nodes

223 variables listed

MATLAB Function: plotTraces

MATLAB Variable: MATLAB Variable Name (Optional)

- Select one or more sections from the multi-section simulation data using the *Select Section* button.



S No	Section Name	Section Sweep	Temp	✓
1	** Profile: "PolarPlot-AC" [E:\local_hier\17.4_QIR3_latest\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace		27.000000Deg	✓

Note: It is recommended to select the *Export Selected Traces* option to select a set of traces, if the PSpice data file size is more than 800 MB.

Note: Measurement expressions are not exported to MATLAB because output of a measurement expressions is a single value.

Exporting of traces to MATLAB for Advanced Waveform Analysis

Use the PSpice - MATLAB interface to export a set of traces to MATLAB for Advanced Waveform Analysis. With this feature, you can:

- Generate special plots in MATLAB by directly exporting PSpice simulation results
- Generate special plots in MATLAB by exporting all traces
- Generate customized plot using a MATLAB script

Generate special plots in MATLAB by directly exporting PSpice simulation results

In this section, a design project, `pspicematlab_exporttrace.opj`, shipped with PSpice installation is used to explain how to generate special plots.

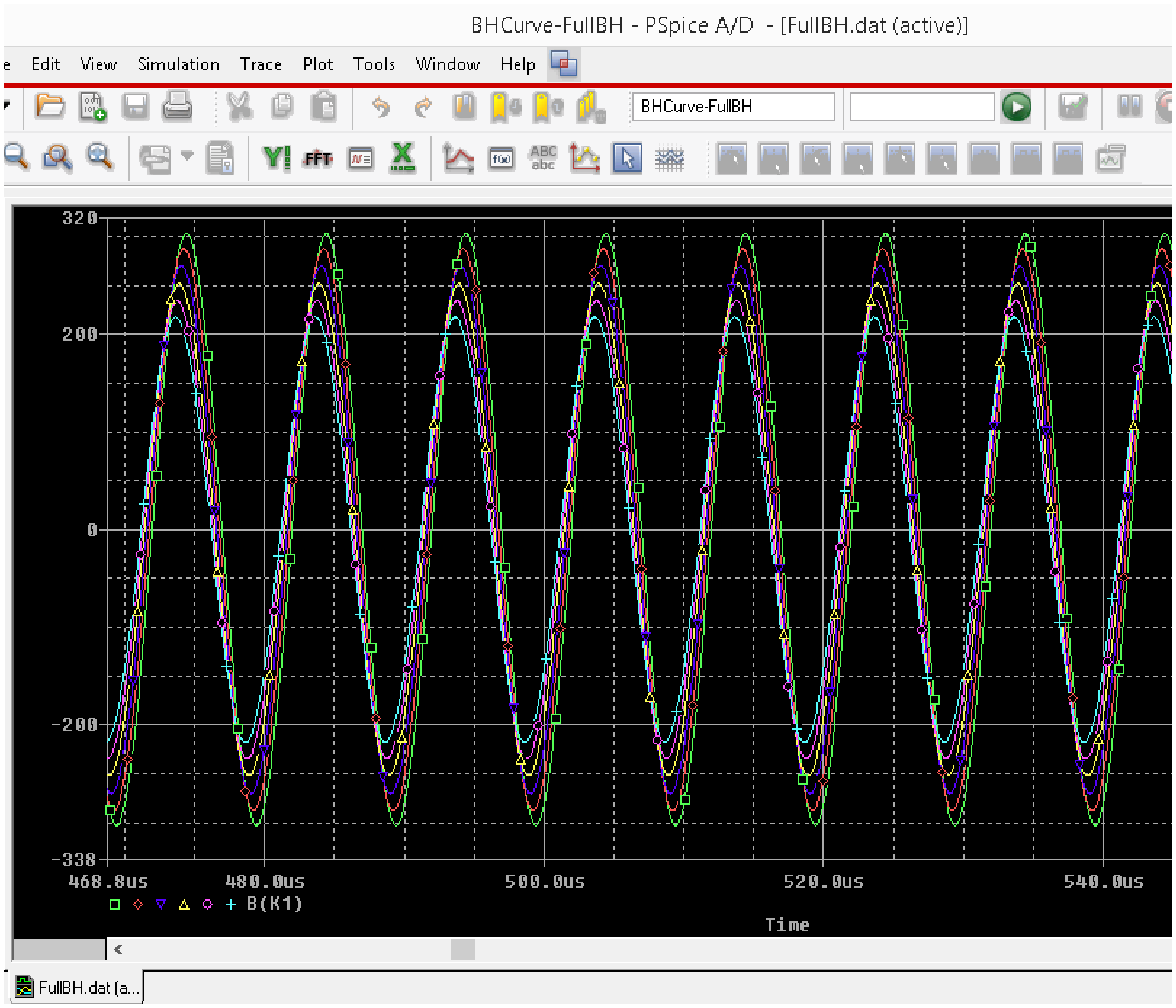
Procedure

1. Launch Capture.
2. Navigate to the location of the design project:
   `<installation_directory>\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace`
3. Select and project a project.

   In this example, select: `pspicematlab_exporttrace.opj`
4. Select `BHCurve` and simulate the design.

   The simulation results open in PSpice along with the Available Sections window.
5. Click *All* to select all the sections and then click *OK*.

The following figure shows the simulation results in PSpice.


BHCurve-FullBH - PSpice A/D - [FullBH.dat (active)]
Edit View Simulation Trace Plot Tools Window Help
BHCurve-FullBH
320
200
0
-200
-338
468.8us
480.0us
500.0us
520.0us
540.0us
B(K1)
Time
FullBH.dat (a...

6. To start the MATLAB engine, select *Tools – MATLAB – Start MATLAB Engine.*

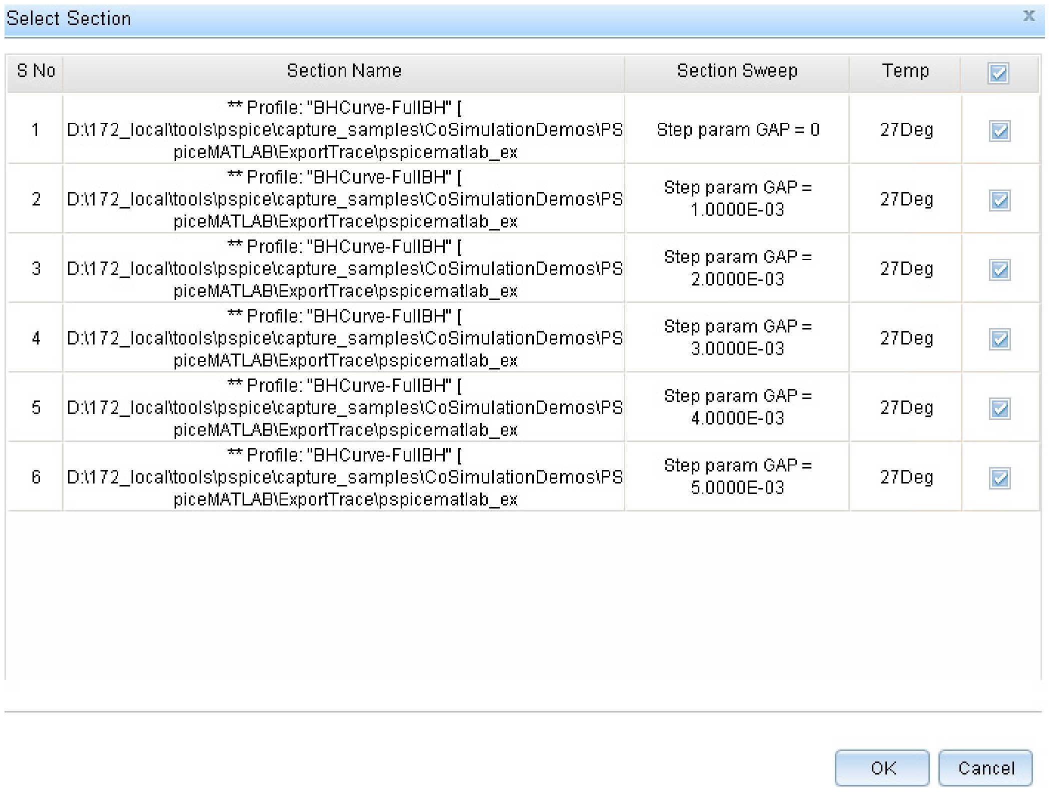
   The MATLAB Command Window appears.

7. Select *Tools – MATLAB – Export Selected Traces* to export selected traces to MATLAB after you have simulated the schematic design.

   The Export to MATLAB window appears.

8. Click the *Select Section* button to select the sections that you want to use to export the trace data. This step is optional.

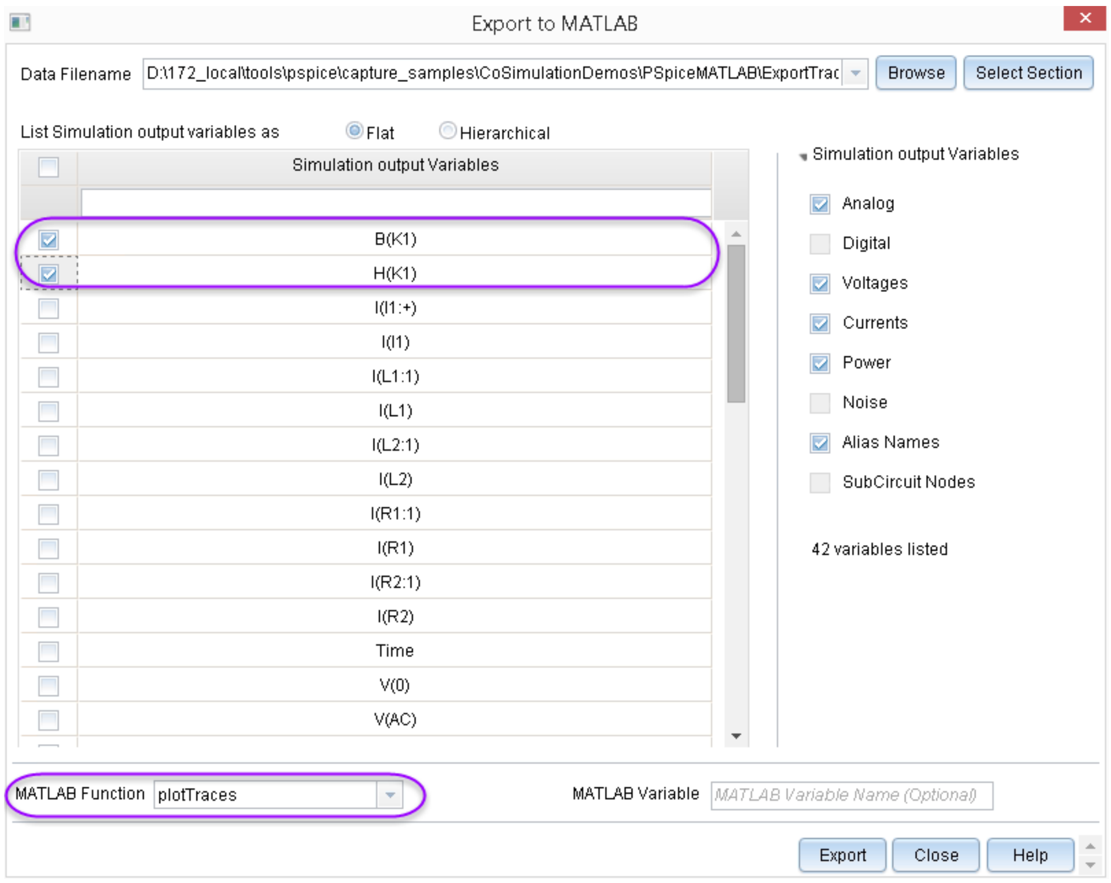
If you have multi-section simulation, you can select one or more sections from the Select Section window and click *OK*.


Select Section

S No	Section Name	Section Sweep	Temp	✓
1	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 0	27Deg	✓
2	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 1.0000E-03	27Deg	✓
3	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 2.0000E-03	27Deg	✓
4	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 3.0000E-03	27Deg	✓
5	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 4.0000E-03	27Deg	✓
6	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 5.0000E-03	27Deg	✓

9. Select the traces you want to export to MATLAB.

In this procedure, the default function, `plotTraces`, is used to plot the selected traces in the MATLAB plot window.



10. Specify the *MATLAB Variable* name that will have all the exported trace data. This step is optional.

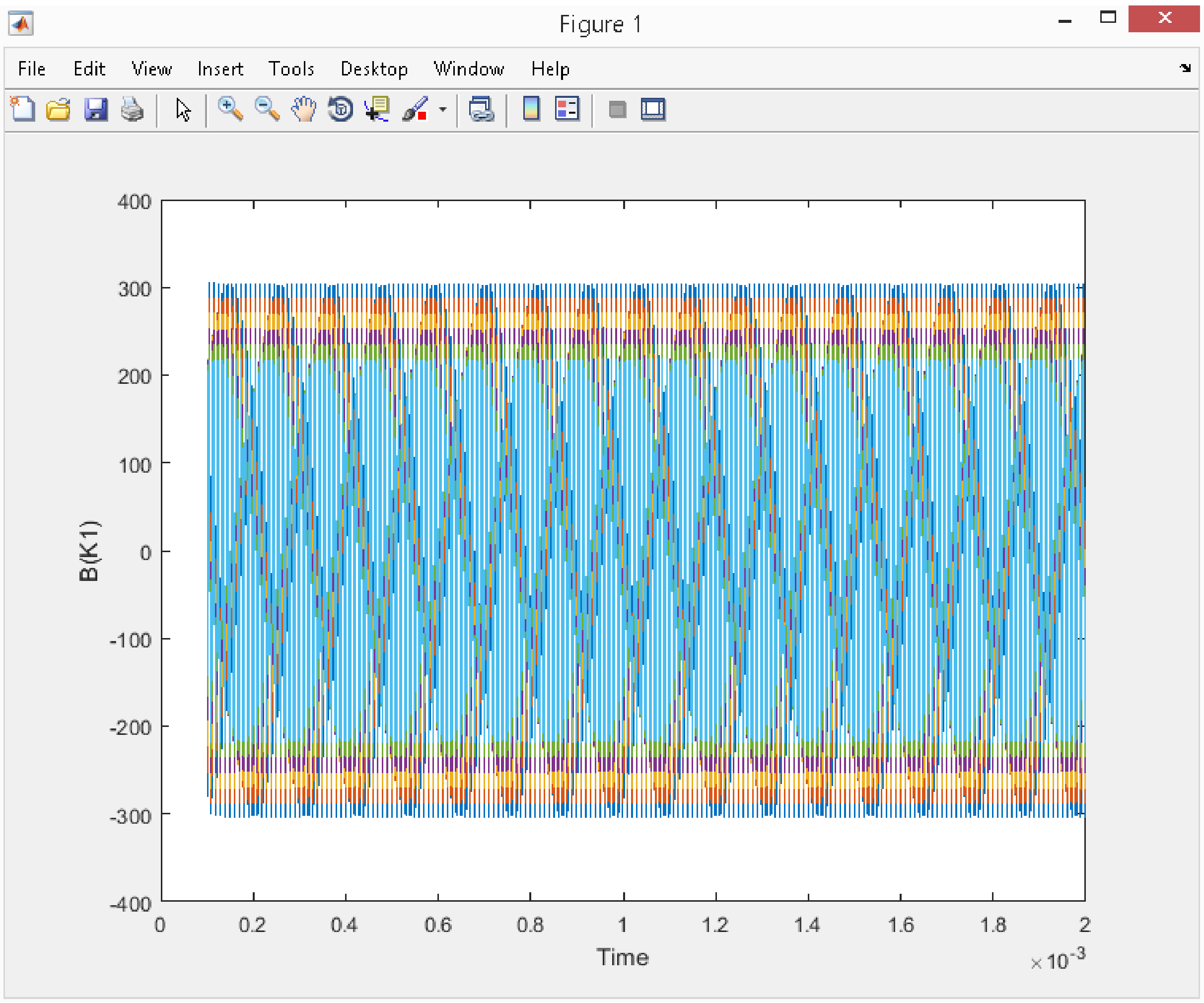
    Note: In case, you have not specified any variable name, a default MATLAB variable will be generated.

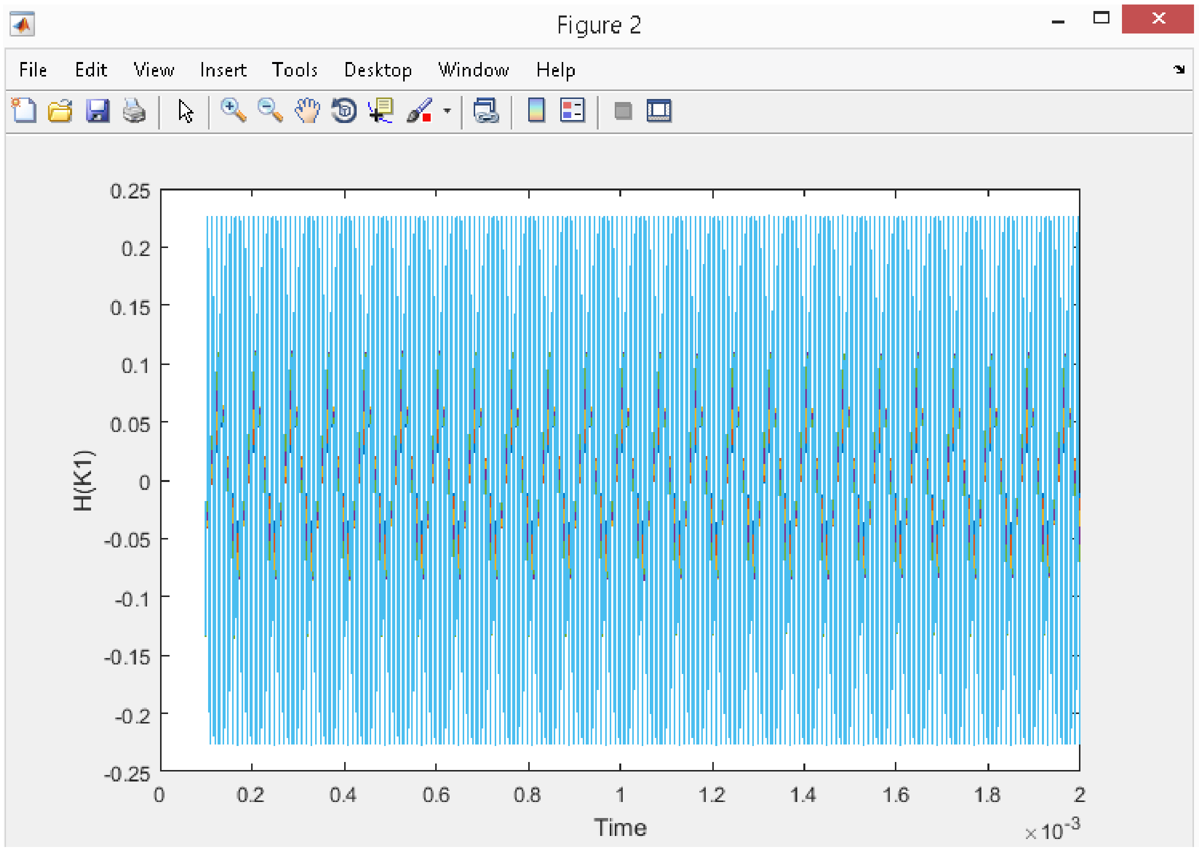
11. Once you have selected the traces, click the *Export* button.

    If you have not started the MATLAB Engine, you will see the MATLAB command window getting displayed. Also, a MATLAB figure panel is displayed with trace plot of the selected trace with respect to time.

    Note: Once you export the trace data to MATLAB, a MATLAB variable is generated, (`PSpiceData_`*`<instance_number>`*) if you have not specified any in the *MATLAB Variable* field. The instance number gets incremented every time you export a new trace data to MATLAB.

The following screen shots shows the plots generated using the default `plotTraces` function:





The MATLAB variable generated is a structured array, which can be used later in MATLAB using various customized functions.

Note: Some customized MATLAB scripts, which can be used to access the PSpice trace data from the MATLAB variable, are at the following location: `<`*`installation_directory`*`>\tools\pspice\tclscripts\orPSPMatlab\MatlabScripts`. For more information on how to access the trace data from the generated MATLAB variable, see the Accessing Trace Data from the MATLAB Variable chapter.

Caution

It is recommended to save your data in MATLAB before you close the MATLAB Command Window. Otherwise all the trace data, including variables generated, will be lost.

Generate special plots in MATLAB by exporting all traces

In this section, a design project, `pspicematlab_exporttrace.opj`, shipped with PSpice installation is used to explain how to export all traces.

Procedure

1. Launch Capture.
2. Navigate to the location of the design project:
   `<installation_directory>\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace`
3. Select and project a project.

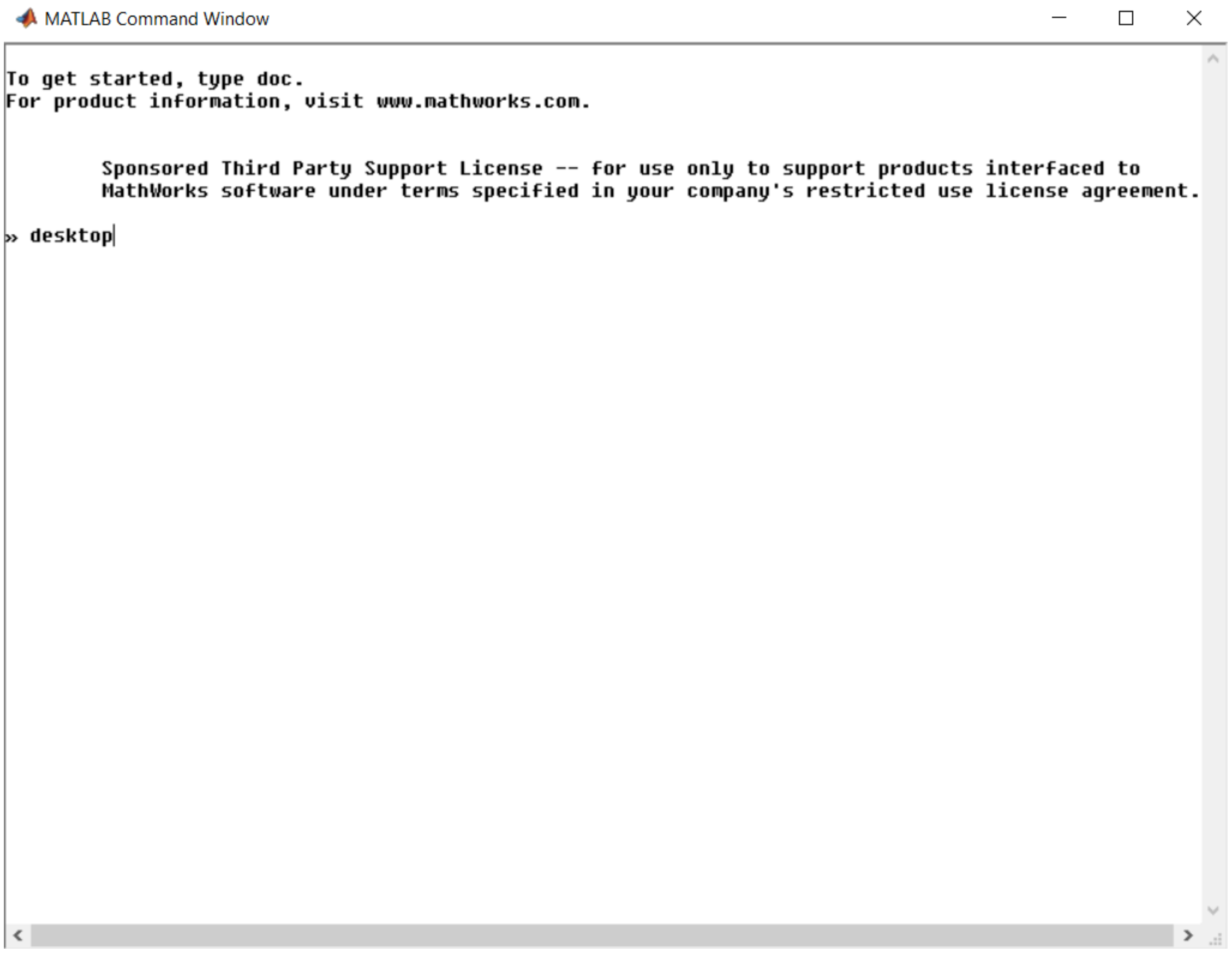
   In this example, select: `pspicematlab_exporttrace.opj`
4. Select `BHCurve` and simulate the design.

   The simulation results open in PSpice along with the Available Sections window.
5. Click *All* to select all the sections and then click *OK*.
6. To start the MATLAB engine, select *Tools – MATLAB – Start MATLAB Engine.*

   The MATLAB Command Window appears.
7. Select *Tools – MATLAB – Export All Traces* to export all the traces to MATLAB after you have simulated the schematic design.

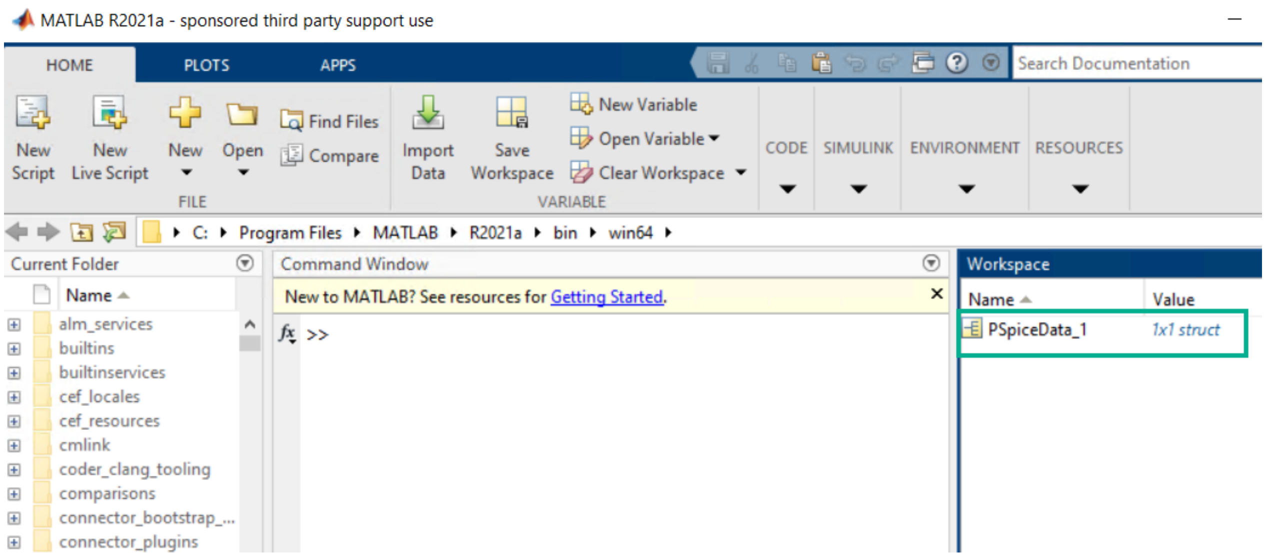
   The MATLAB Command Window appears.

8. Type the `desktop` command to open the desktop version of MATLAB.



```
To get started, type doc.
For product information, visit www.mathworks.com.


        Sponsored Third Party Support License -- for use only to support products interfaced to
        MathWorks software under terms specified in your company's restricted use license agreement.

>> desktop
```

In the desktop version of MATLAB, you can view that all the traces are exported under the MATLAB variable, `PSpiceData_1`.



Generate customized plot using a MATLAB script

PSpice allows you to add your customized MATLAB plotting functions and use them through the Export to MATLAB window. Using customized MATLAB functions, you can visualize different plots other than the default one (created using the `plotTraces` function).

For example, you can use customized function to create a polar plot or 3D plot that can help in understanding the data in more detail.

A customized MATLAB function must be saved as MATLAB script (`.m`) file.

In this section, a design project, `pspicematlab_exporttrace.opj`, shipped with PSpice installation is used to explain how to generate custom plots.

Procedure

1. Launch Capture.
2. Navigate to the location of the design project:

   `<installation_directory>\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace`
3. Select and open a project.

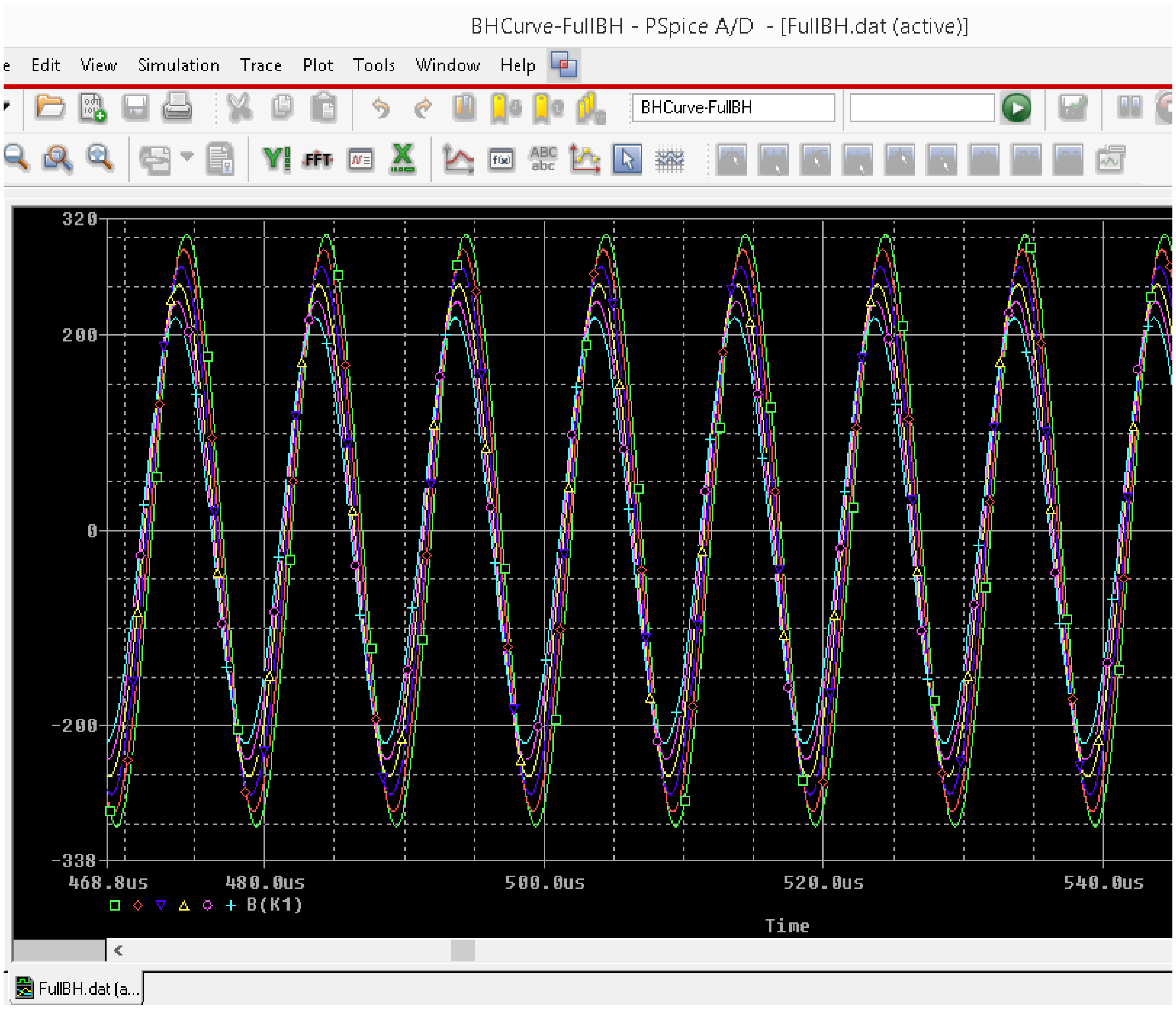
   In this example, select: `pspicematlab_exporttrace.opj`

4. Select `BHCurve` and simulate the design.

   The simulation results open in PSpice along with the Available Sections window.

5. Click *All* to select all the sections and then click *OK*.

   The following figure shows the simulation results in PSpice.


BHCurve-FullBH - PSpice A/D - [FullBH.dat (active)]
e Edit View Simulation Trace Plot Tools Window Help
BHCurve-FullBH
320
200
0
-200
-338
468.8us
480.0us
500.0us
520.0us
540.0us
□ ◇ ▽ △ ○ + B(K1)
Time
FullBH.dat (a...

6. To start the MATLAB engine, select *Tools – MATLAB – Start MATLAB Engine.*

   The MATLAB Command Window appears.

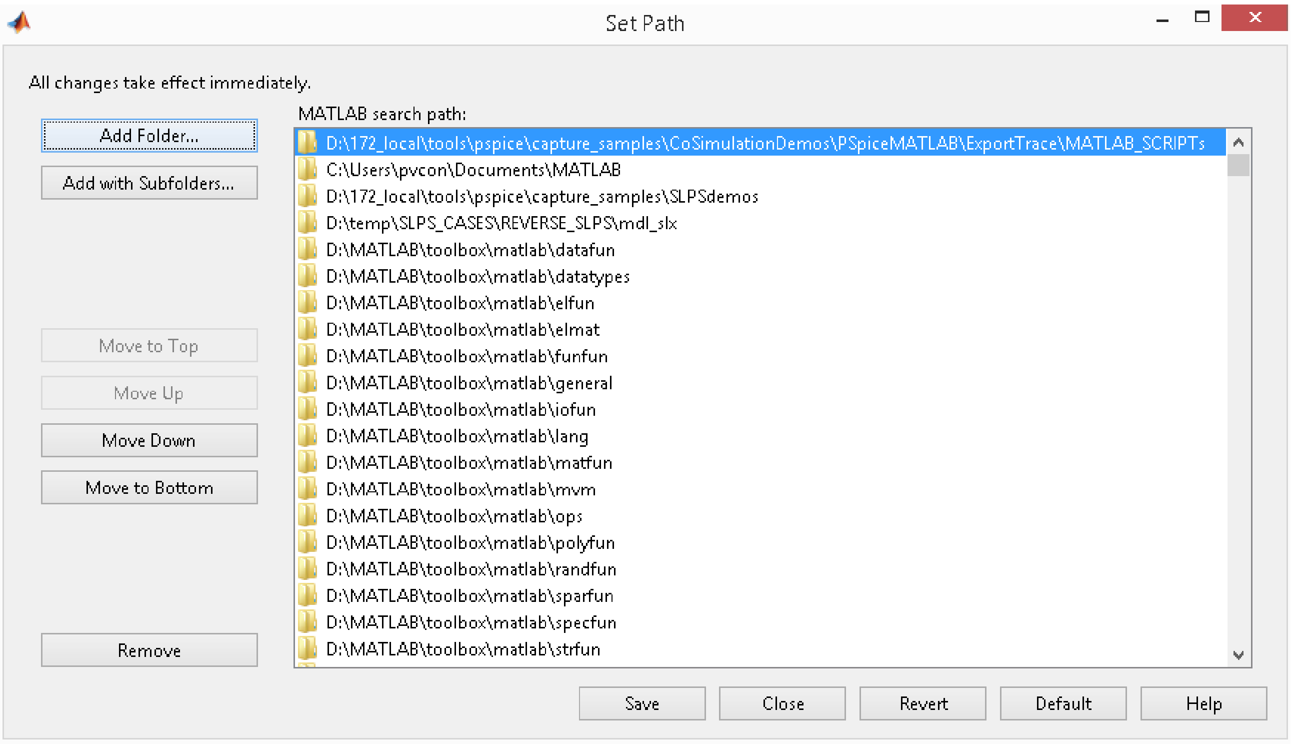
7. Type the `desktop` command to open the desktop version of MATLAB.

8. In the *HOME* tab, click *Set Path*.

The Set Path window appears.

9. Click the *Add Folder* button and browse to the location of the custom MATLAB script:

   `<installation_directory>\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\MATLAB_SCRIPTs`



All changes take effect immediately.

MATLAB search path:

```
D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\MATLAB_SCRIPTs
C:\Users\pvcon\Documents\MATLAB
D:\172_local\tools\pspice\capture_samples\SLPSdemos
D:\temp\SLPS_CASES\REVERSE_SLPS\mdl_slx
D:\MATLAB\toolbox\matlab\datafun
D:\MATLAB\toolbox\matlab\datatypes
D:\MATLAB\toolbox\matlab\elfun
D:\MATLAB\toolbox\matlab\elmat
D:\MATLAB\toolbox\matlab\funfun
D:\MATLAB\toolbox\matlab\general
D:\MATLAB\toolbox\matlab\iofun
D:\MATLAB\toolbox\matlab\lang
D:\MATLAB\toolbox\matlab\matfun
D:\MATLAB\toolbox\matlab\mvm
D:\MATLAB\toolbox\matlab\ops
D:\MATLAB\toolbox\matlab\polyfun
D:\MATLAB\toolbox\matlab\randfun
D:\MATLAB\toolbox\matlab\sparfun
D:\MATLAB\toolbox\matlab\specfun
D:\MATLAB\toolbox\matlab\strfun
```

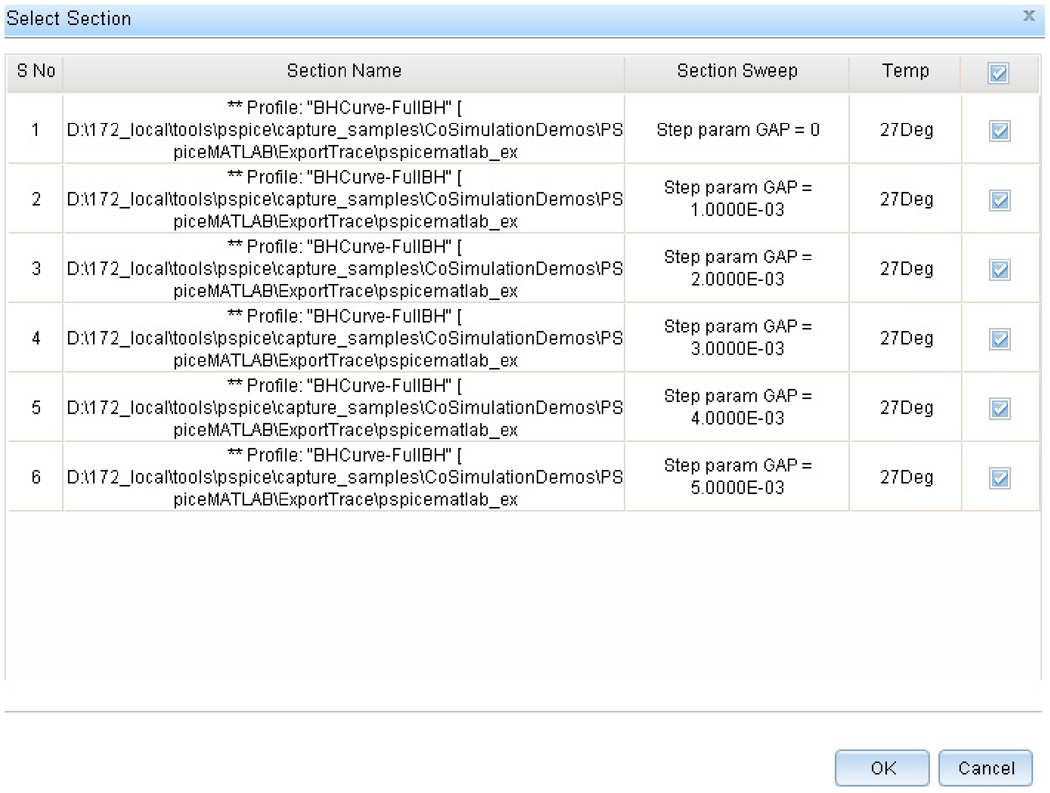
10. Click *Save*.

11. In PSpice, select *Tools – MATLAB – Export Selected Traces* to export selected traces to MATLAB.

    The Export to MATLAB window appears.

12. Click the *Select Section* button to select the sections that you want to use to export the trace data. This step is optional.

If you have multi-section simulation, you can select one or more sections from the Select Section window and click *OK*.


Select Section

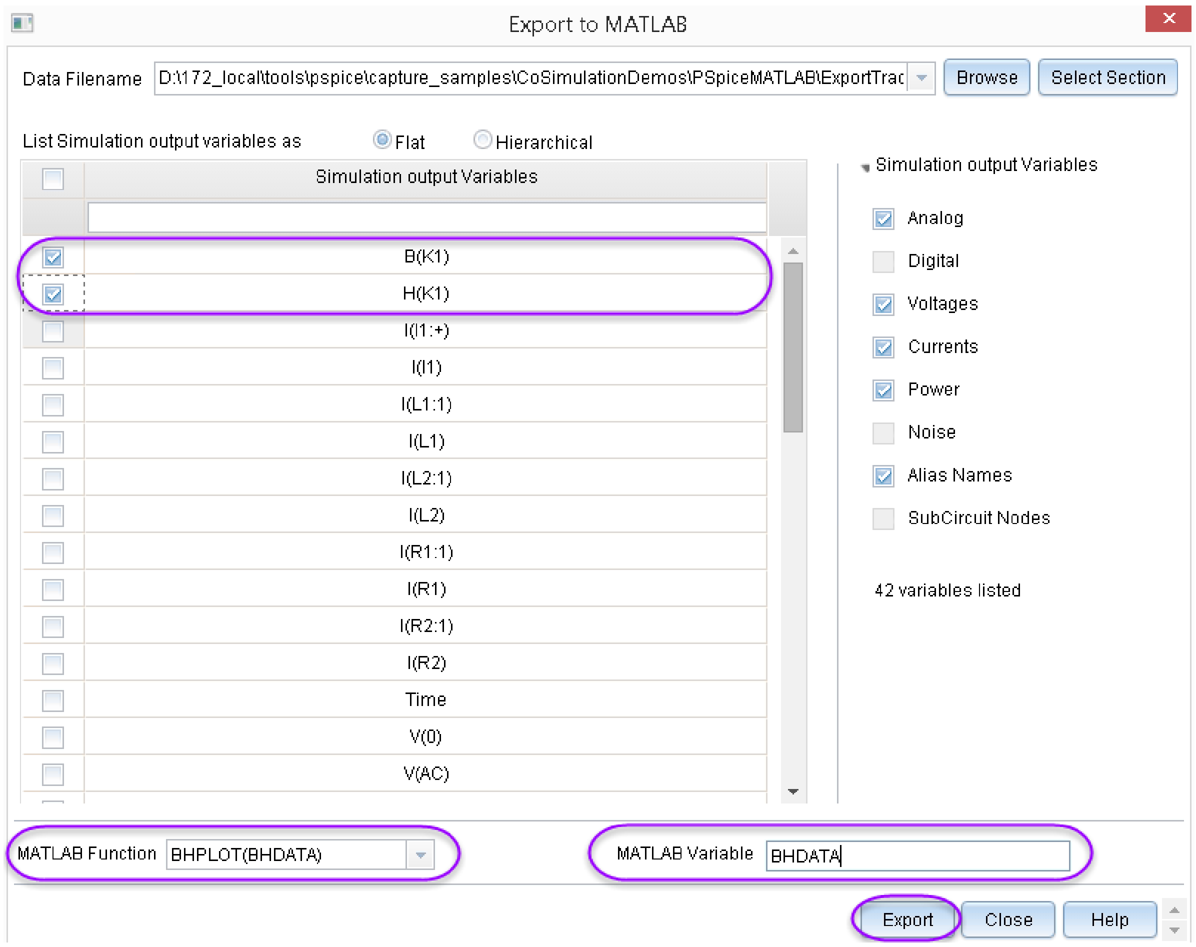
S No	Section Name	Section Sweep	Temp	✓
1	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 0	27Deg	✓
2	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 1.0000E-03	27Deg	✓
3	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 2.0000E-03	27Deg	✓
4	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 3.0000E-03	27Deg	✓
5	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 4.0000E-03	27Deg	✓
6	** Profile: "BHCurve-FullBH" [D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\pspicematlab_ex	Step param GAP = 5.0000E-03	27Deg	✓

13. Select the traces you want to export to MATLAB.

    In this example, select the traces, `B(K1)` and `H(K1)` under simulation output variable.

14. Specify a custom MATLAB function, `BHPLOT(BHDATA)`, to plot the selected traces in the MATLAB plot window.

15. Specify the MATLAB variable name, `BHDATA`, which will save all the exported trace data.



Data Filename D:\172_local\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrac

List Simulation output variables as [✓ Flat] Hierarchical

	Simulation output Variables
✓	B(K1)
✓	H(K1)
	I(I1:+)
	I(I1)
	I(L1:1)
	I(L1)
	I(L2:1)
	I(L2)
	I(R1:1)
	I(R1)
	I(R2:1)
	I(R2)
	Time
	V(0)
	V(AC)

Simulation output Variables

- ✓ Analog
- Digital
- ✓ Voltages
- ✓ Currents
- ✓ Power
- Noise
- ✓ Alias Names
- SubCircuit Nodes

42 variables listed

MATLAB Function BHPLOT(BHDATA)

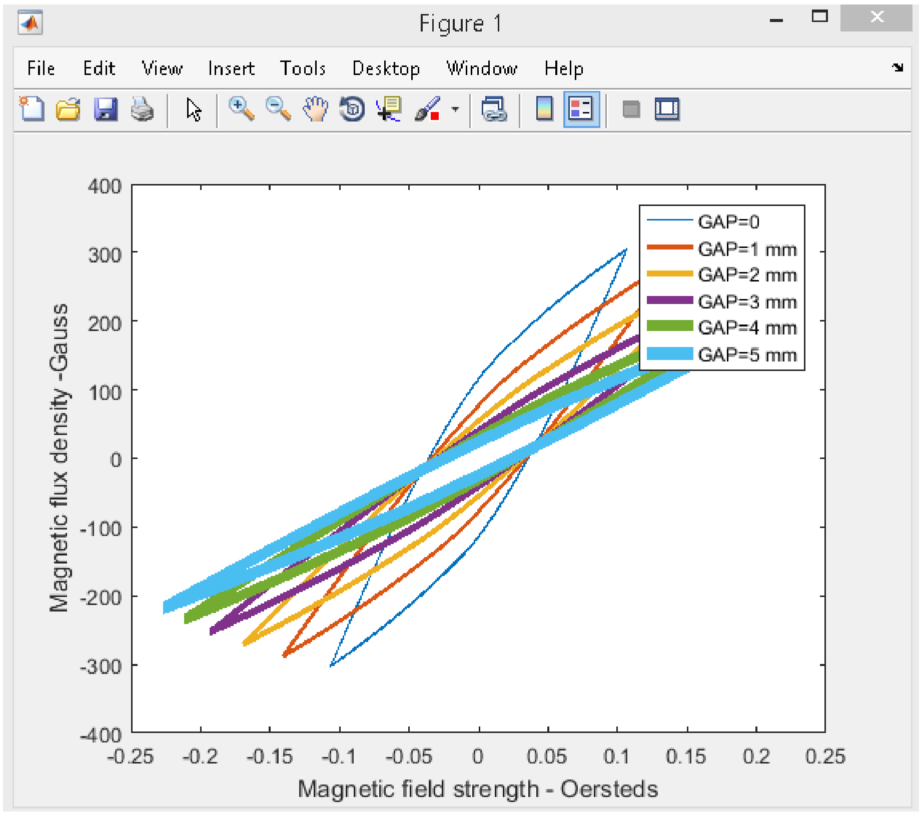
MATLAB Variable BHDATA

16. Click the *Export* button.

Note: If you have not started the MATLAB Engine, you will see the MATLAB command window getting displayed.

A MATLAB figure panel is displayed with trace plot of the selected trace with respect to time.

For this example, the following screen shot shows the BH plots that are generated with airgap values from 0 to 5 mm.



Calling the MATLAB function in PSpice

PSpice allows you to add various function for the behavioral controlled devices. These functions can be related to mathematical expressions, laplace transformations, logarithmic function and many more.

PSpice allows you to add a MATLAB function in the expression for the E and G behavioral controlled devices.

Where:

- E: Voltage Controlled Voltage Source
- G: Voltage Controlled Current Source

For more information, see *PSpice A/D Reference Guide*.

Various MATLAB functions can be added using the following syntax:

```
MATLABFUNCTION(<MATLAB_function>, <Parameter_1>, <Parameter_2>,
<number_of_parameters>)
```

Where:

- *`<MATLAB_function>`*: Is the name of the function in MATLAB
- *`<Parameter_1>`* and *`<Parameter_2>`*: Are arguments used in *`<MATLAB_function>`*
- *`<number_of_parameters>`*: number of arguments used in *`<MATLAB_function>`*

Important

Remember the following points before you use `MATLABFUNCTION` in PSpice:

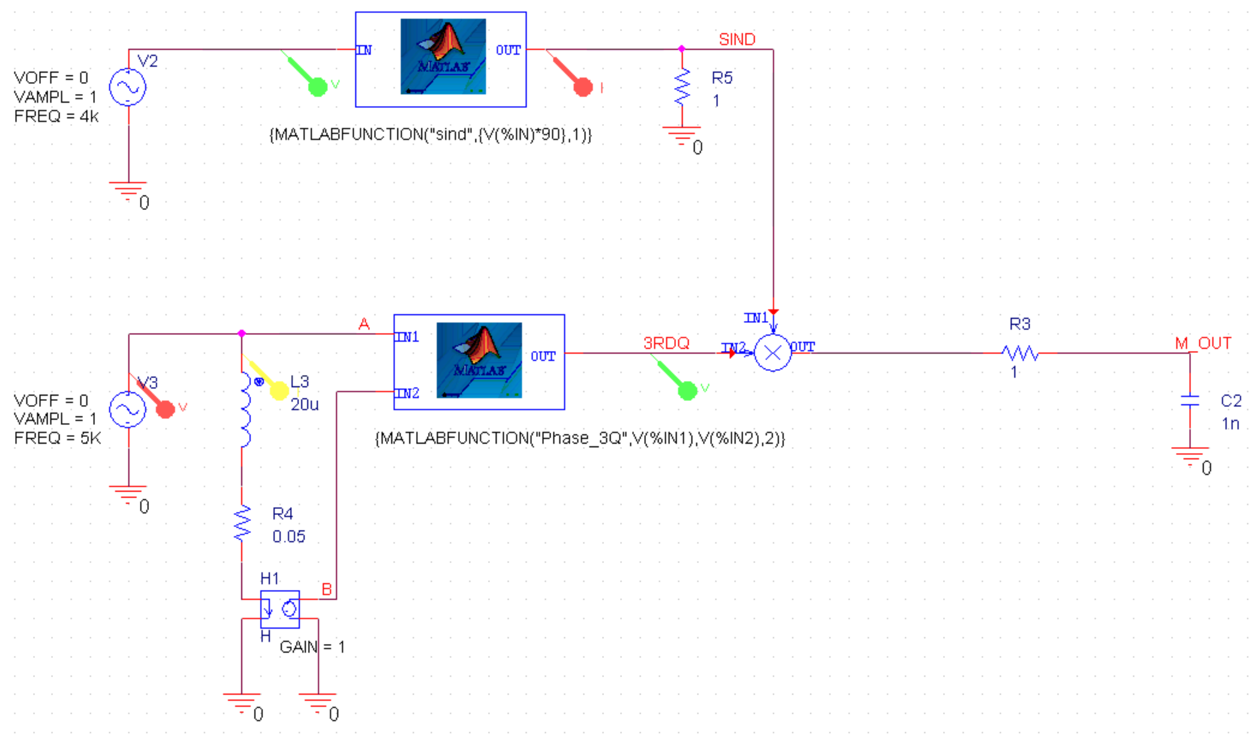
- You should correctly spell the *`<MATLAB_function>`* else simulation will abort or nothing will be displayed on the probe window. Similarly, correctly mention the number of arguments else you will get a simulation error.
- The *`<number_of_parameters>`* defines the number of arguments that the *`<MATLAB_function>`* can take in the `MATLABFUNCTION` function. If the *`<MATLAB_function>`* used in `MATLABFUNCTION` has five arguments, the *`<number_of_parameters>`* argument will be five.
- The parameter and return type of *`<MATLAB_function>`* used in `MATLABFUNCTION` is constant, not an array or a matrix. The constant values can be

one of the following types: integers, double, and float. If parameters other than constant are used, the `MATLABFUNCTION` function will not simulate.

Example

The following screen shot shows a design example that uses two MATLAB functions in PSpice. You can open this example from the following location:

`<installation_directory>\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\MATLABFunctions`



In the first function, `{MATLABFUNCTION("sind",{V(%IN)*90},1)}`:

- `sind` is the name of the MATLAB function. This function requires one argument and this is indicated by the last parameter of this expression, which is `1`.
- `V(%IN)*90` is the argument used in `sind`.
- `1` defines the number of parameters used in `sind`.

In the second function, `{MATLABFUNCTION("Phase_3Q",V(%IN1),V(%IN2),2)}`, which is a user-defined function created in MATLAB:

- `Phase_3Q` is the name of the user-defined MATLAB function. This function requires two arguments, which is indicated by the last parameter of this expression, `2`.
- `V(%IN1)` and `V(%IN2)` are the arguments used in `Phase_3Q`.
- `2` defines the number of parameters used in `Phase_3Q`.

To perform waveform analysis for this design, do the following:

1. **1.** If you are using any user-defined or customized MATLAB functions in the design, ensure that you have added the custom MATLAB script folder in the Set Path window of MATLAB. To do so:

   a. Launch *MATLAB R2019b* or onwards.

   b. In the *HOME* tab, click *Set Path*.

   The Set Path window appears.

   c. Click the *Add Folder* button and browse to the location of the custom MATLAB script:

   For this example, browse to the location:
   `<installation_directory>\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\ExportTrace\MATLAB_SCRIPTs`

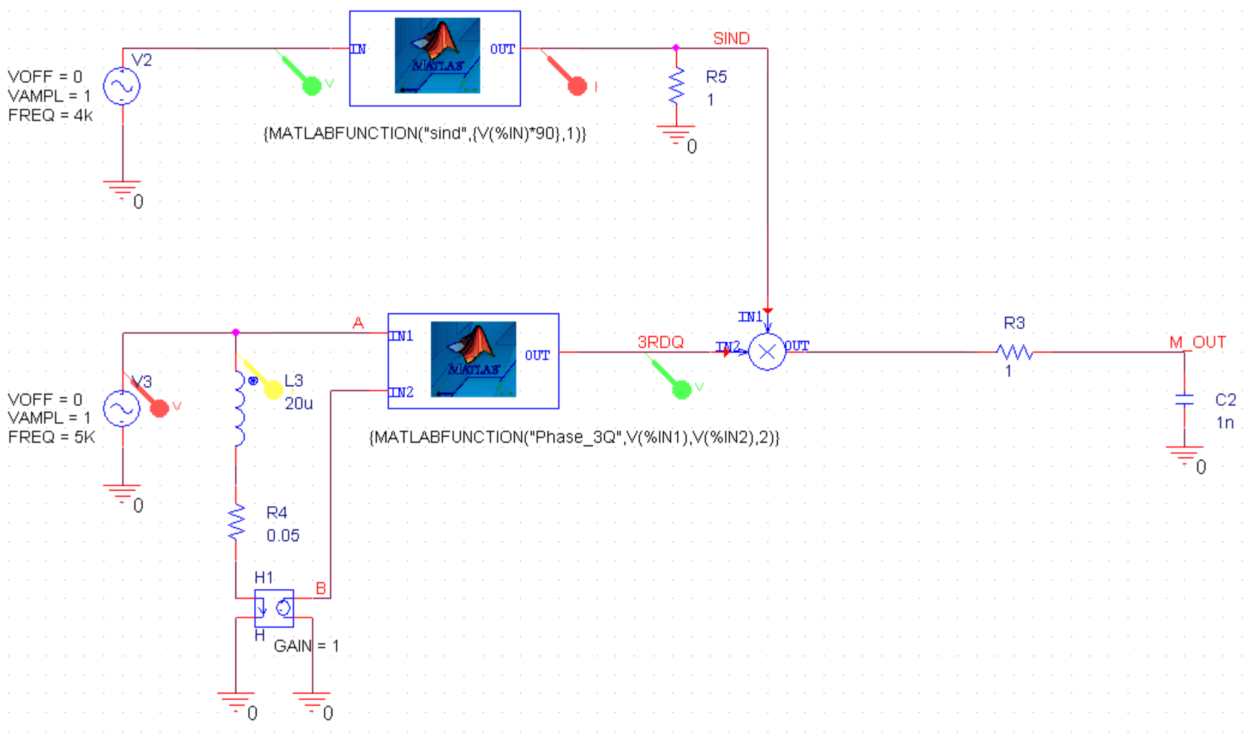
   d. Click *Save*.

2. **2.** Launch Capture.
3. **3.** Navigate to the location of the design project.

   For this example, navigate to
   `<installation_directory>\tools\pspice\capture_samples\CoSimulationDemos\PSpiceMATLAB\MATLABFunctions`

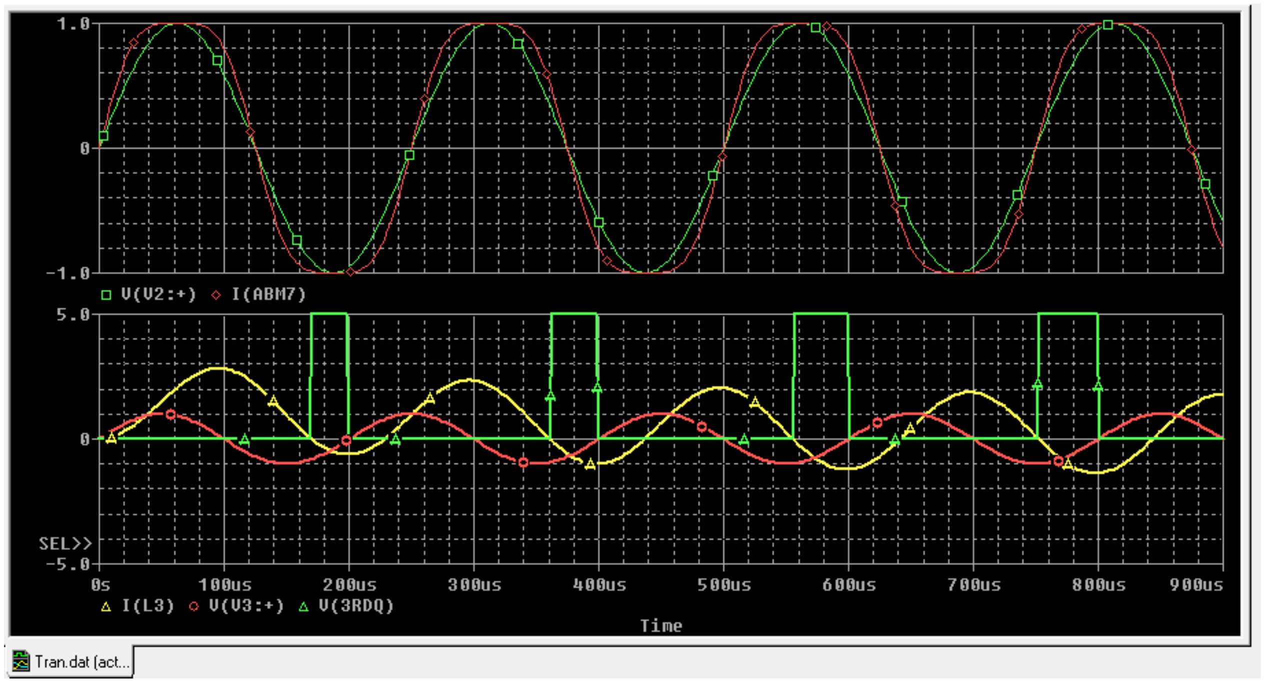
4. **4.** Select and open the project: `matlab_function.opj`

The design opens in Capture.



5. Run the simulation.

A MATLAB figure panel is displayed with trace plot of the selected trace with respect to time.



Using the MATLAB-Based Measurement Function in PSpice

PSpice allows you to use various measurement functions to evaluate the characteristics of the waveform generated after simulation is completed. For more information on various measurement functions, see the *Measurement Expressions* chapter in the *PSpice User Guide*.

You can use the MATLAB-based measurement functions to evaluate the waveforms generated in PSpice. When these functions are used in PSpice, MATLAB is used in the background to run the measurement functions, which evaluate the selected trace(s).

By default, the following two general purpose MATLAB-based functions have been added in PSpice:

- `MATLABFunction1(`*`<trace_name>`*, *`<MATLAB_function>`*`)`

  Where:

  - *`<trace_name>`*: Is the name of the trace in PSpice
  - *`<MATLAB_function>`*: Is the name of the MATLAB function to be evaluated on *`<trace_name>`*

  Use this function to call any MATLAB function that uses one argument and returns a constant expression.

- `MATLABFunction2(`*`<trace_name1>, <trace_name2>, <MATLAB_function>`*`)`

  Where:

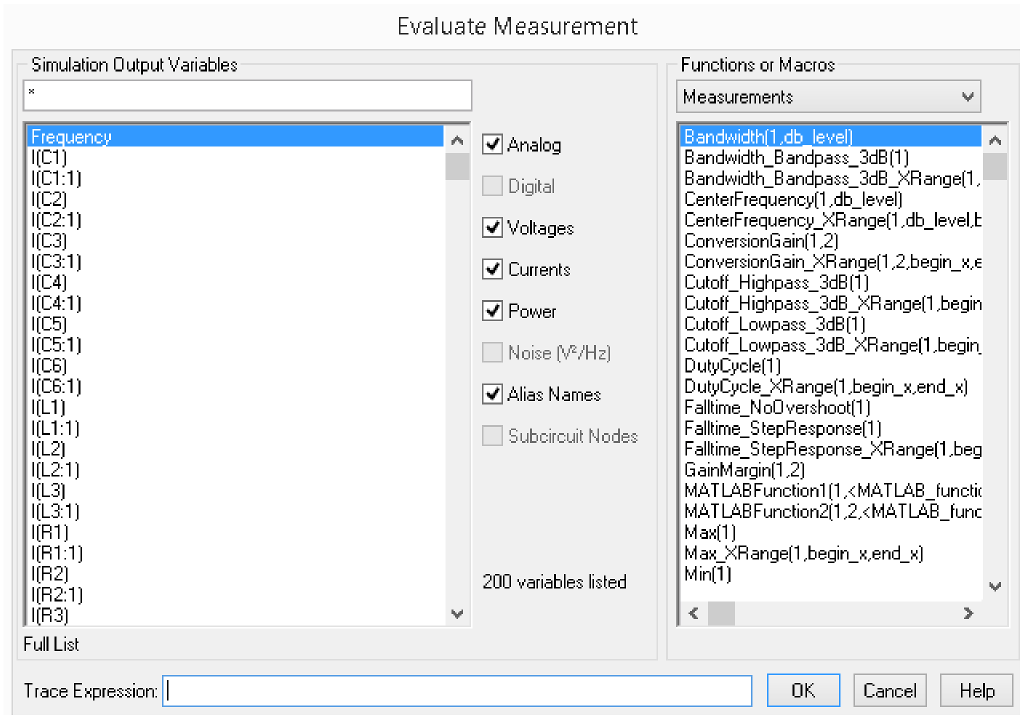
  - *`<trace_name1>, <trace_name2>`*: Are names of the traces in PSpice
  - *`<MATLAB_function>`*: Is the name of the MATLAB function to be evaluated on *`<trace_name1>`* and *`<trace_name2>`*

  Use this function to call any MATLAB function that uses two arguments and returns a constant expression.

Perform the following steps to evaluate the characteristics of a waveform using the MATLAB-based measurement functions:

1. In PSpice, choose *Trace – Evaluate Measurement*.

The Evaluate Measurement window appears.



2. From the *Functions or Macros* section, select any one of the two MATLAB- based measurement function.

   Once you select the function, the *Trace Expression* field gets updated in with function's definition.

3. Specify the arguments for the selected MATLAB-based measurement function:

   a. Trace Name: From the *Simulation Output Variables* section, select the trace. It gets updated in the *Trace Expression* field.

   b. Name of the MATLAB function: In the *Trace Expression* field, specify the name of the MATLAB function.

   For `MATLABFunction1`:

   - ❑ For example: `MATLABFunction1(V(out), risetime)`
     - ❍ `V(out) is the trace name.`
     - ❍ `risetime is the` name of the MATLAB function.

   For `MATLABFunction2`:

   - ❑ For example: `MATLABFunction2(V(out), I(out), risetime)`

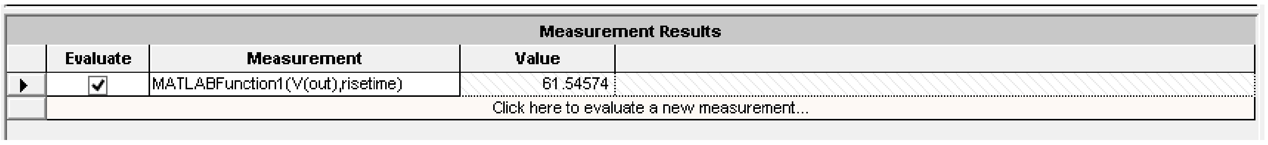
- `V(out)` and `I(out)` are trace names.
- `risetime is the` third argument, which is the MATLAB function.

4. Click *OK*.

   The measurement function is displayed in the Measurement Results window of the PSpice probe.

5. Select the measurement function from the Measurement Results window.

   The value evaluated is displayed.



Measurement Results

	Evaluate	Measurement	Value
▶	✓	MATLABFunction1(V(out),risetime)	61.54574
	Click here to evaluate a new measurement...		

Note: Ensure the following before you evaluate a waveform using a MATLAB-based measurement function in PSpice:

- The MATLAB function name is correct.
- The number of arguments mentioned are correct.

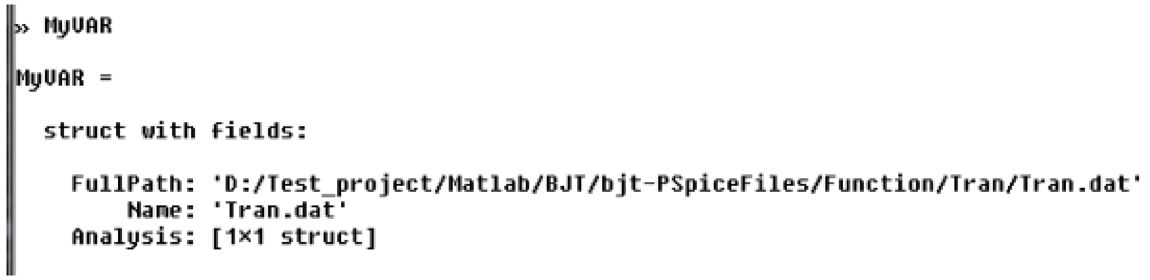
For incorrect expressions, PSpice will always return value=0.

5 Accessing Trace Data from the MATLAB Variable

This chapter explains about the generated variable, how to extract data from the generated variable, and how to use the data in MATLAB to generate various plots. Following topics are covered in this chapter:

What is the generated MATLAB variable?

Whenever you export the traces to MATLAB, either a default MATLAB variable is generated in the MATLAB Command Window, such as `PSpiceData_N`, where `N` is a digit; or a customized MATLAB variable is generated in the MATLAB Command Window, such as `MyVAR`.

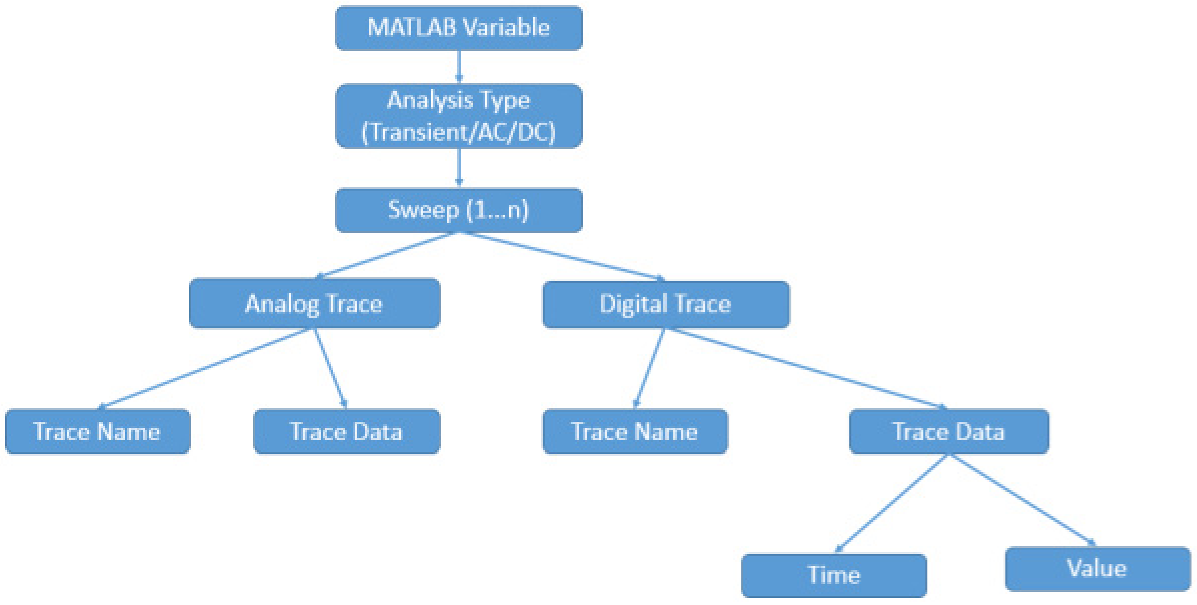


```
>> MyVAR

MyVAR =

  struct with fields:

    FullPath: 'D:/Test_project/Matlab/BJT/bjt-PSpiceFiles/Function/Tran/Tran.dat'
        Name: 'Tran.dat'
    Analysis: [1×1 struct]
```

The MATLAB variable is a structured array that contains the following:

- full path of the trace data file (`.dat`)
- name of the trace data file
- trace data in the `Analysis` struct variable

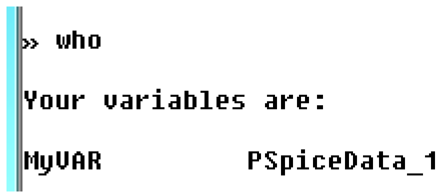
The following diagram illustrates the tree structure of the generated MATLAB variable:



How to check if the MATLAB variable is generated or not?

To check if the MATLAB variable is generated in the MATLAB command window, type *who* on the command prompt of the MATLAB Command Window.

Once done, you will see the name of the variable getting displayed if it is generated. The following screen shot shows that two variables, `MyVAR` and `PSpiceData_1`, are present in the current MATLAB Command Window session:



```
>> who

Your variables are:

MyVAR        PSpiceData_1
```

How to launch the desktop version of MATLAB from the MATLAB command window?

Type `desktop` in the MATLAB Command Window's prompt to launch MATLAB's desktop version.

How to access the trace data from the MATLAB variable using customized MATLAB scripts?

As the trace data is saved as a structured array in the MATLAB variable, you can access the trace data using the customized scripts shipped with the Cadence hierarchy or using various MATLAB commands.

How to access the previously generated MATLAB variable in the current session of MATLAB workspace?

You can not access previously generated MATLAB variables in the current session of the MATLAB workspace unless you have saved the previous MATLAB workspace session. As MATLAB workspace sessions are independent of each other, therefore all the trace data is lost after you close them.

If you want to access the currently generated MATLAB variable again, you need to save the workspace as `.mat` file. To save a MATLAB current workspace as `.mat` file, click *Save*

Workspace in the *Home* tab. Once you save the workspace, you can re- use the generated MATLAB variable in the new MATLAB workspace session by opening the saved the `.mat` file.

Where are various customized scripts stored in the Cadence hierarchy and how to use them?

To access the customized MATLAB scripts from the Cadence hierarchy, browse the following location in the MATLAB's current folder window: `<`*`installation_directory`*`>\tools\pspice\tclscripts\orPSPMatlab\MatlabScripts`

Following MATLAB scripts can be used to access the trace data stored in the generated MATLAB variable:

- getPspiceData
- getPrimaryParamData
- getPrimaryParamName
- getSecondaryParamData
- getSecondaryParamName
- listPSpiceDataStructAccess
- listPSpiceDataStructAccessAll
- plotTraces

getPspiceData

Use this customized MATLAB script to get the trace data of the MATLAB variable.

Syntax

```
getPspiceData(<varName in MATLAB workspace>, '<analysisName>', '<traceName>')
```

Parameter Description

`varName` – Name of variable created in MATLAB

`traceName` – Trace name to be used with single quotation marks `analysisName` – Analysis name to be used with single quotation marks

getPrimaryParamData

Use this customized MATLAB script to get the trace data of the MATLAB variable's primary parameter.

Syntax

```
getPrimaryParamData(<varName in MATLAB workspace>, '<analysisName>')
```

Parameter Description

`varName` – Name of variable created in current session of the MATLAB Command Window

`analysisName` – Analysis name to be used with single quotation marks

getPrimaryParamName

Use this customized MATLAB script to get the name of the MATLAB variable's primary parameter.

Syntax

```
getPrimaryParamName(<varName in MATLAB workspace>, '<analysisName>')
```

Parameter Description

`varName` – Name of variable created in current session of the MATLAB Command Window

`analysisName` – Analysis name to be used with single quotation marks

getSecondaryParamData

Use this customized MATLAB script to get the trace data of the MATLAB variable's secondary parameter.

Syntax

```
getSecondaryParamData(<varName in MATLAB workspace>, '<analysisName>')
```

Parameter Description

`varName` – Name of variable created in current session of the MATLAB Command Window

`analysisName` – Analysis name to be used with single quotation marks

getSecondaryParamName

Use this customized MATLAB script to get the name of the MATLAB variable’s secondary parameter.

Syntax

```
getSecondaryParamName(<varName in MATLAB workspace>, '<analysisName>')
```

Parameter Description

`varName` – Name of variable created in current session of the MATLAB Command Window

`analysisName` – Analysis name to be used with single quotation marks

listPSpiceDataStructAccess

Use this customized MATLAB script to list a specific trace information in a log file. The generated log file will be saved at the same location as the `.dat` file.

Syntax

```
listPSpiceDataStructAccess(<varName>, '<matlabVarName>', '<traceName>')
```

Parameter Description

`varName` – Name of variable created in current session of the MATLAB Command Window `matlanVarName` – Same as `varName` and to be used with single quotation marks

`traceName` – Trace name to be used with single quotation marks

listPSpiceDataStructAccessAll

Use this customized MATLAB script to list information of all traces in a log file (`.log`). The generated log file will be at the same location as the `.dat` file.

Syntax

```
listPSpiceDataStructAccessAll(<varName>, '<matlabVarName>')
```

Parameter Description

`varName` – Name of variable created in current session of the MATLAB Command Window

`matlanVarName` – Same as `varName` and to be used with single quotation marks

plotTraces

Use this customized MATLAB script to plot selected traces.

Syntax

```
plotTraces(<varName in MATLAB workspace>, '<analysisName>', '<traceList>')
```

Parameter Description

`varName` – Name of variable created in current session of the MATLAB Command Window

`analysisName` – Same as `varName` and to be used with single quotation marks

`traceList` – Space separated list of traces to be used with single quotation marks

Part B: PSpice Simulink Co-Simulation Interface

The chapters in part B introduce you to PSpice Simulink Co-Simulation interface and describes the various steps involved in working with PSpice Simulink Co-Simulation.

The chapters also describe how to prepare a PSpice project for simulation and work with CIR files.

6

Introducing PSpice Simulink Co-Simulation

This chapter introduces the PSpice Simulink Co-Simulation interface and lists the requirements to run it:

- Introducing the PSpice Simulink Co-Simulation Interface on page 58
- Requirements for PSpice Simulink Co-Simulation Interface on page 58

Introducing the PSpice Simulink Co-Simulation Interface

PSpice Simulink Co-Simulation interface is an interface tool that links PSpice to the MATLAB® Simulink® system simulator, provided by The Mathworks.

PSpice is a SPICE-based simulator used for simulating electrical and electronic circuits, and The Mathworks tools are used for system designing. PSpice Simulink Co-Simulation interface integrates these two simulators to provide a simulation flow that can be used to design any kind of system with electronic subsystems.

Requirements for PSpice Simulink Co-Simulation Interface

You should have the following combination of The Mathworks and Cadence OrCAD and Allegro products installed on your system:

1. The Mathworks products
   - *MATLAB R2019b* or onwards
   - *Simulink*
2. Cadence OrCAD products or Cadence Allegro products

   Install one of the following design entry tools:
   - Capture or Capture CIS
   - Design Entry HDL

   Install the following simulator:
   - PSpice A/D
3. To be able to use the PSpice - MATLAB Interface, ensure that you have license for one of these products in your license file.
   - OrCAD PSpice Designer Plus
   - Allegro PSpice Systems Simulator

Setting Up the PSpice Simulink Co-simulation Solution

You can use one of the following two ways to setup the PSpice Simulink Co-simulation solution:

- Automatic Setup
- Manual Setup

Automatic Setup

To ensure that the MATLAB path settings are done automatically, do the following in PSpice:

1. Ensure that you have set the MATLAB path in PSpice. For details see, Setting Up the MATLAB Path.
2. Select *Tools – MATLAB – Co-simulation* to open MATLAB.

Manual Setup

When you want to launch MATLAB directly from the Start menu, instead of launching it from PSpice (*Tools – MATLAB – Co-simulation*), you need to set the PSpice Simulink Co-simulation solution path manually. To do so:

1. Set the PATH variable of your system as:
   `<installation_directory>\tools\bin`
2. Set the PSpice Simulink Co-simulation path in the PSpice installation in MATLAB. To do so:
   a. Launch MATLAB.
   b. Under Home tab, select Set Path.
   c. Click the *Add Folder* button and browse to the following location:
   `<installation_directory>\tools\pspice\pspCosim`

7

Creating a Schematic

This chapter explains how to create and edit a schematic using either OrCAD Capture or Allegro Design Entry HDL, simulate the schematic using PSpice, and then create and setup a block diagram using MATLAB.

Creating a Schematic

You can use either OrCAD Capture or Allegro Design Entry HDL to create and edit schematics.

OrCAD Capture is a schematic design tool set for the Windows environment. With Capture, you can draft schematics and produce connectivity and simulation information for printed circuit boards and programmable logic designs.

Design Entry HDL helps you capture the design of a printed circuit board (PCB) in the schematic form. Design Entry HDL organizes schematic information into pages. It captures and displays only one page of schematic information at a time. Design Entry HDL is a by-reference editor because it references all parts in the schematic from various libraries that reside at the reference or local area. To know more about Design Entry HDL, refer to the *Allegro Design Entry HDL User Guide*.

Note: Refer to the chapter Designing for other EDA applications of the online OrCAD Capture User Guide on how to incorporate a SPICE netlist into a circuit. Refer to the Transient analysis chapter of the online PSpice User Guide on how to perform transient analysis using PSpice.

The following section explains how to create schematics.

Using OrCAD Capture

Perform the following steps to create a schematic using Capture:

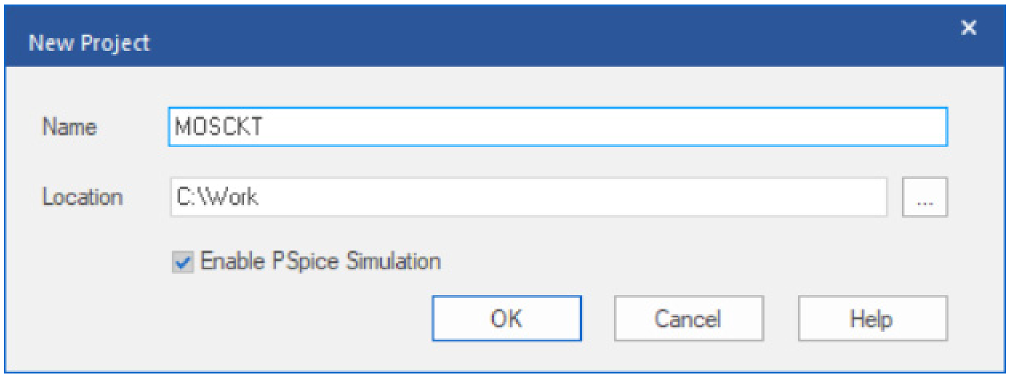
1. Create a project in Capture.
2. Create the schematic.

To create a new project, perform the following tasks:

1. Launch Capture.
2. Select *File – New – Project*.

   The New Project dialog box appears.
3. Specify the project name. In this example, enter `MOSCKT` in the *Name* field.
4. Specify a work directory in the *Location* field. For this example, create a directory, `Work` in the `C` drive.

5. Select *Enable PSpice Simulation*.


New Project

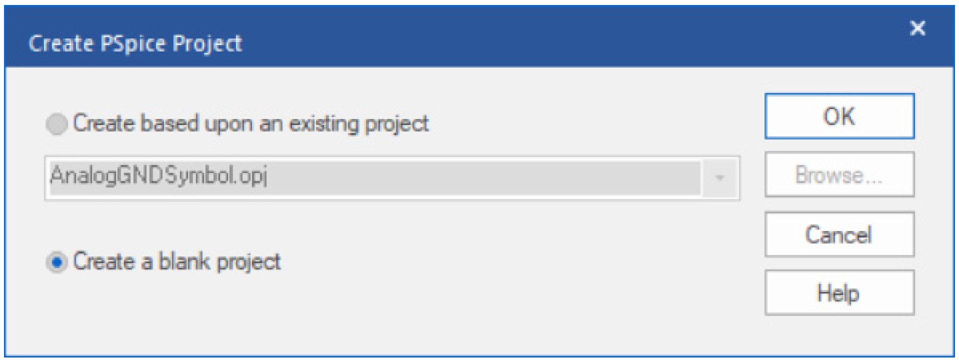
Name: MOSCKT

Location: C:\Work

[✓] Enable PSpice Simulation

6. Click *OK*.

   The Create PSpice Project dialog box appears because you had selected *Enable PSpice Simulation* check box.


Create PSpice Project

○ Create based upon an existing project

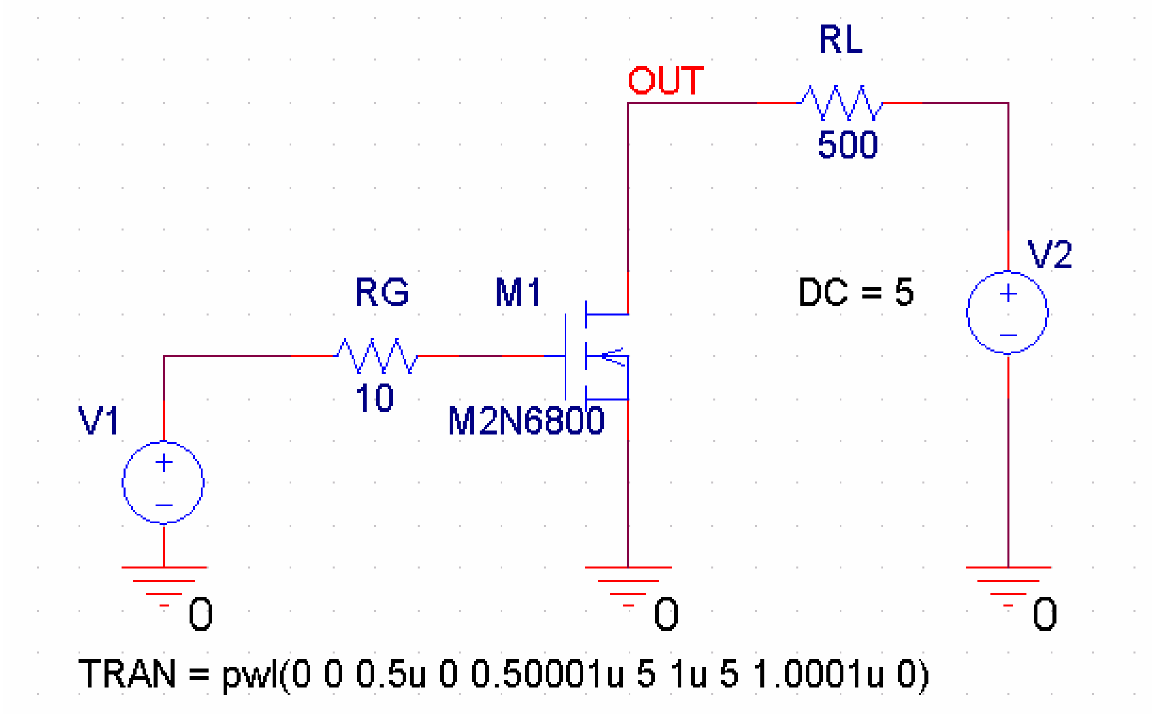
AnalogGNDSymbol.opj

● Create a blank project

7. Select *Create a blank project* and click *OK*.

   The project window and schematic page is displayed.

You have created the project, now you can create a schematic. Create the circuit as shown in the following figure.



This circuit is comprised of a MOSFET, resistors, and voltage sources. The input signal of the circuit for verifying operation is a 0.5μ wide 1-shot pulse. Attach the node name `OUT` to the drain node. You will use this name to specify output voltage from the PSpice block later. Use the following libraries to create the schematic design:

Library	Component
`pwrmos.olb`	`M2N6800`
`analog.olb`	`R`
`source.olb`	`VSRC`

Select *File – Save* to save the schematic. You can now use PSpice to simulate the schematic.

Using Design Entry HDL

To create a schematic using Design Entry HDL, perform the following tasks:

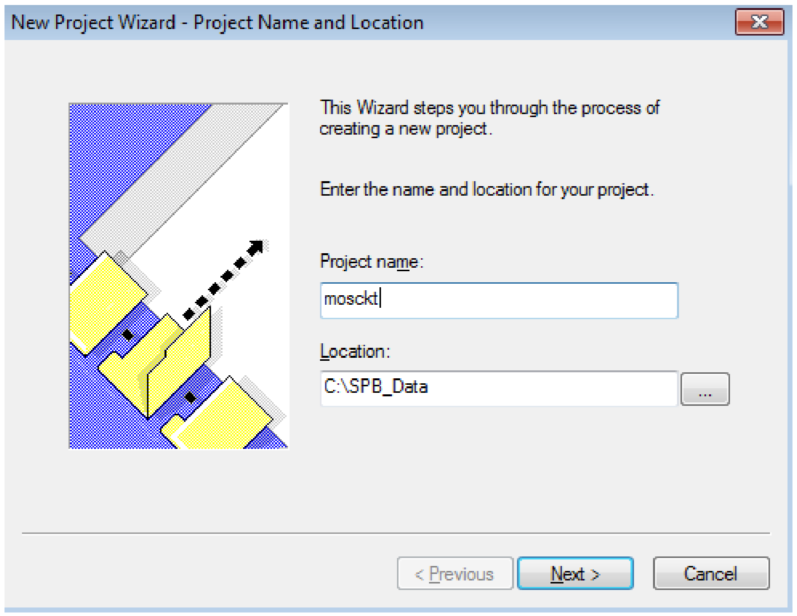
1. Create a project in Project Manager.
2. Create the schematic.

To create a new project:

1. Launch Project Manager.
2. Select *File – New – New Design*.

   The *New Project Wizard* appears. You can also click the *Create Design Project* icon to open the New Project Wizard.
3. In the *Name* field, specify the name of the project, `mosckt`.

   Note: In Design Entry HDL, project can contain only lower-case characters, numbers, and the underscore symbol '_'.


New Project Wizard - Project Name and Location

This Wizard steps you through the process of creating a new project.

Enter the name and location for your project.

Project name:
mosckt

Location:
C:\SPB_Data

4. In the *Location* field, specify the path to the directory where you want to create the project.
5. Click *Next*.

   The *Project Libraries* dialog box appears.
6. Select the required libraries from the list of *Available Libraries*, and click *Add*.

   The libraries are added to the *Project Libraries* list.
7. Click *Next*.

The *Design Name* dialog box appears.

8. In the *Design Name* field, specify the name of the top-level design.
9. Click *Next*.

   The Summary dialog box displays the project details.
10. Click *Finish* to create the project.

    A message appears to indicate that project creation was successful.
11. Click *OK*.

In Project Manager, click the *Design Entry* icon to open Design Entry HDL.

Note: Refer to Allegro Design Entry HDL User Guide or Allegro Design Entry HDL Tutorial to learn more about creating schematics using Design Entry HDL.

Using PSpice to Simulate Schematic

PSpice A/D simulates analog-only, mixed analog/digital, and digital-only circuits. PSpice A/D's analog and digital algorithms are built into the same program so that mixed analog/digital circuits can be simulated with tightly coupled feedback loops between the analog and digital sections without any performance degradation.

Before using PSpice A/D to simulate your design, you need to set up analyses.

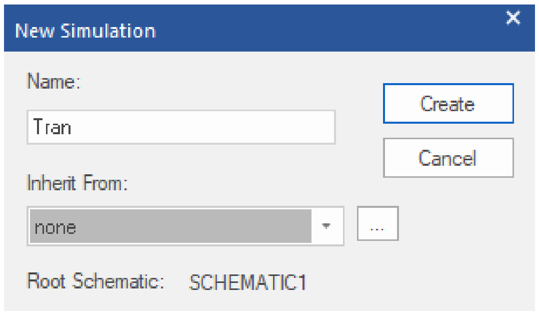
Important

PSpice is referred to as PSpice Simulator in Design Entry HDL. As a result, in Design Entry HDL, the PSpice menu is called PSpice Simulator.

To set up analyses, do the following:

1. Select *PSpice – New Simulation Profile* in Capture or *PSpice Simulator – New Simulation Profile* in Design Entry HDL to open the New Simulation dialog box.
2. Specify the name of the simulation profile in the *Name* field. In this example, specify the name as *Tran*.

3. Select *none* from the *Inherit From* list.



New Simulation

Name: Tran

Inherit From: none

Root Schematic: SCHEMATIC1

4. Click *Create*.
   The Simulation Settings dialog box appears.
   For this example, in the Simulation Settings dialog box:
   a. Select `Time Domain (Transient)` as the *Analysis Type*.
   b. Specify the *Run To Time* as `3u` (3u seconds).
   c. Click *OK*.

Note: To learn more about the Simulation Settings dialog box options, see Chapter 8, Setting Up Analyses and Starting Simulations, of the PSpice User Guide.

You have set up the analyses.

Next, select *PSpice – Run* to run the simulation. PSpice A/D opens and displays an empty plot window.

Place a marker to display a waveform on the schematic. Select *PSpice – Markers – Voltage Level* on the MOSFET drain node, as shown in Figure 7-1, to display a drain voltage waveform.

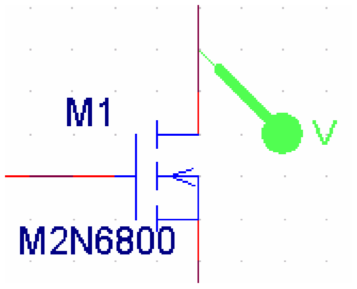
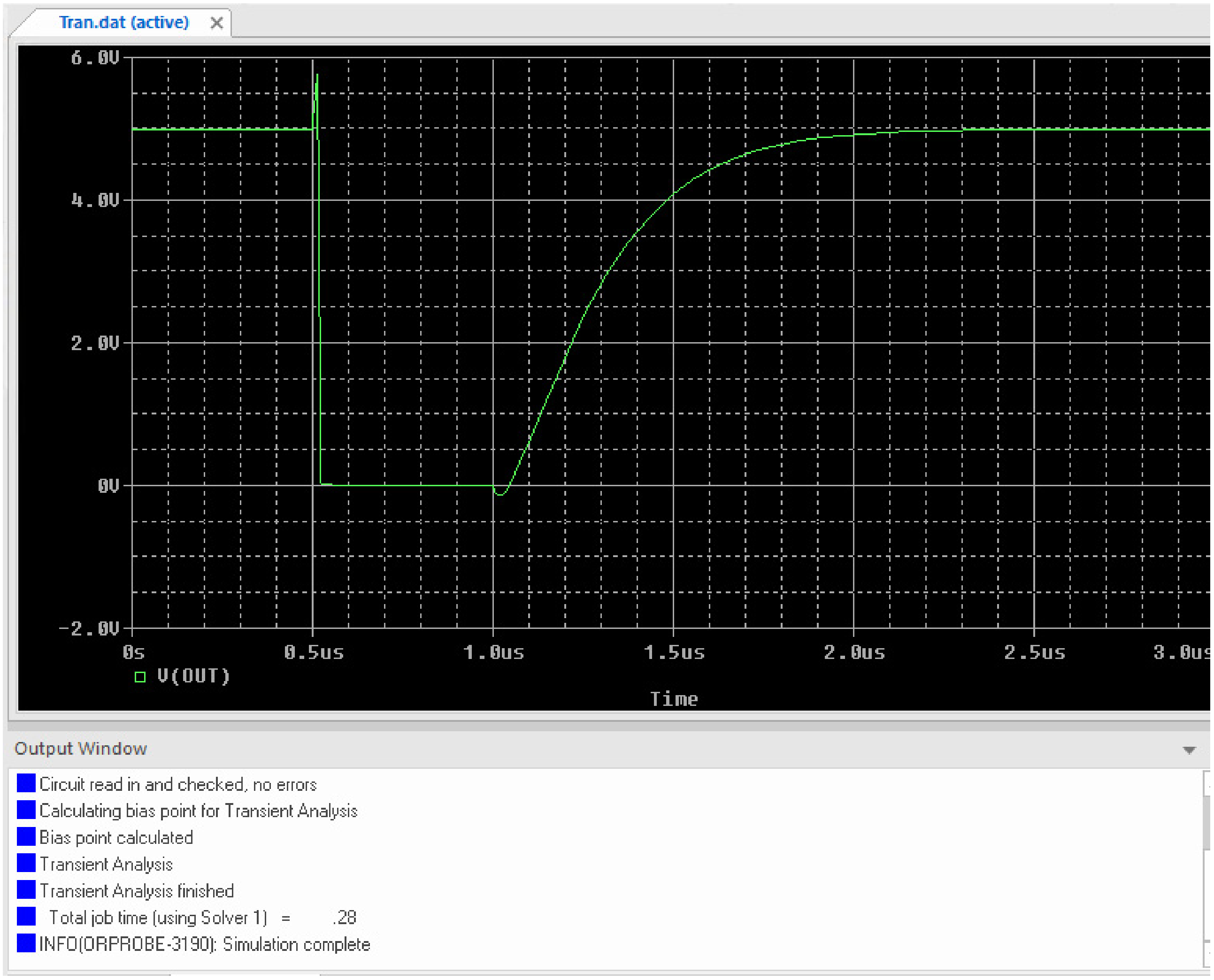


Figure 7-1 Voltage Level on MOSFET

The drain voltage waveform is displayed in the PSpice A/D window, as shown in Figure 7-2.



```
Circuit read in and checked, no errors
Calculating bias point for Transient Analysis
Bias point calculated
Transient Analysis
Transient Analysis finished
 Total job time (using Solver 1)  =       .28
INFO(ORPROBE-3190): Simulation complete
```

Figure 7-2 Drain Voltage waveform

Note: If a schematic diagram or settings are changed in Capture, you must execute analysis once using PSpice A/D in order for the changes to be reflected in PSpice Simulink Co-simulation.

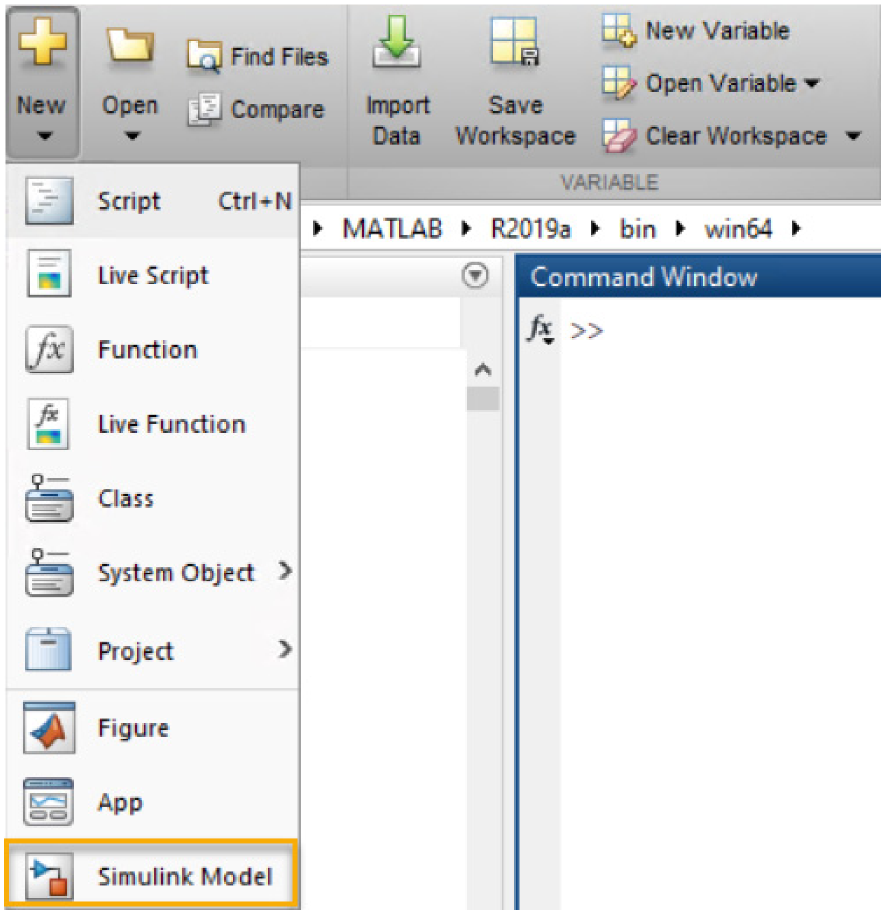
Creating and Setting up Block Diagram Using MATLAB

In this section, you will learn how to use MATLAB to create a block, set PSpice Simulink Co-simulation block parameters, set up Simulink analyses, and simulate using both PSpice and Simulink.

Creating a Block

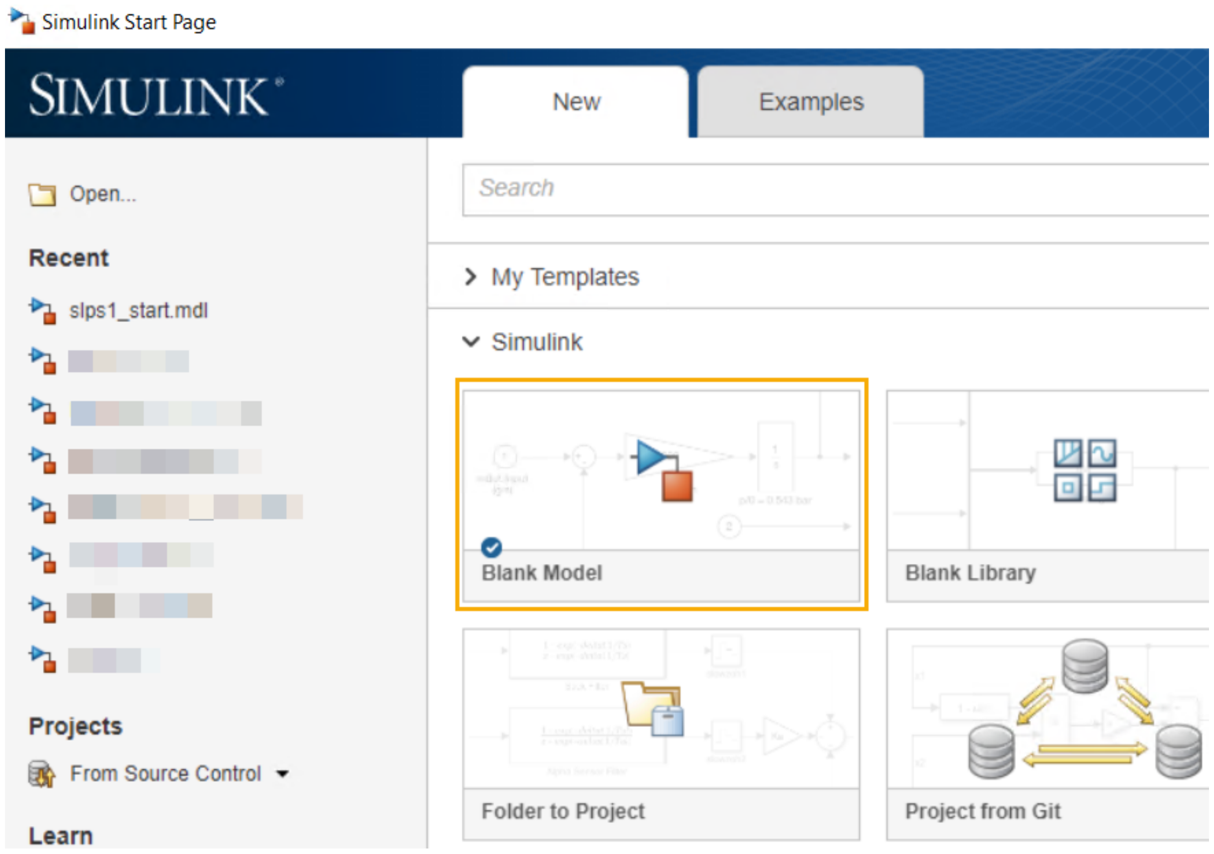
To create a block, perform the following tasks:

1. Launch MATLAB.
2. Using the MATLAB Command Window, set the current MATLAB directory to be the work directory where PSpice files are saved.
3. Under the *Home* Tab, in the *File* section, click the *New* menu and select *Simulink Model* to create a new model.

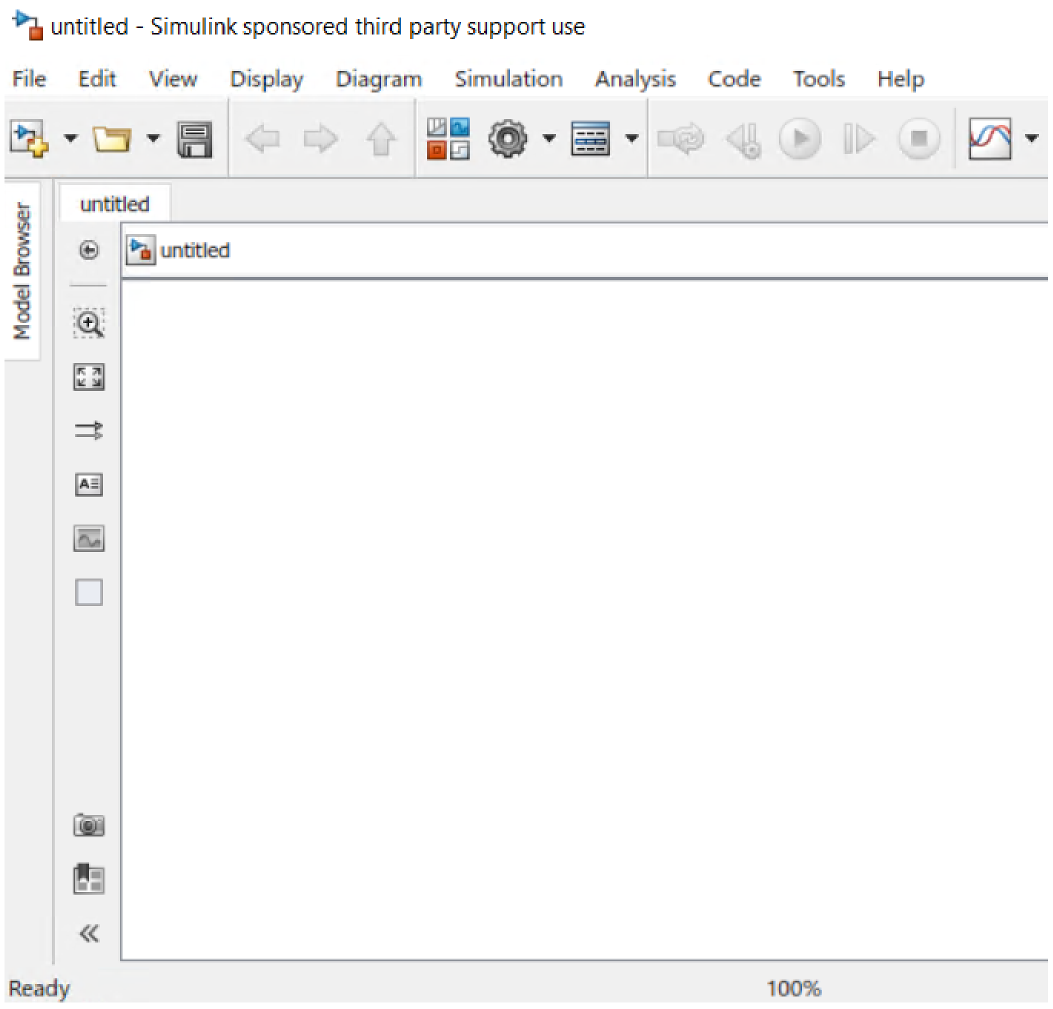


The Simulink Start Page appears.

4. Under the Simulink section, click *Blank Model*.

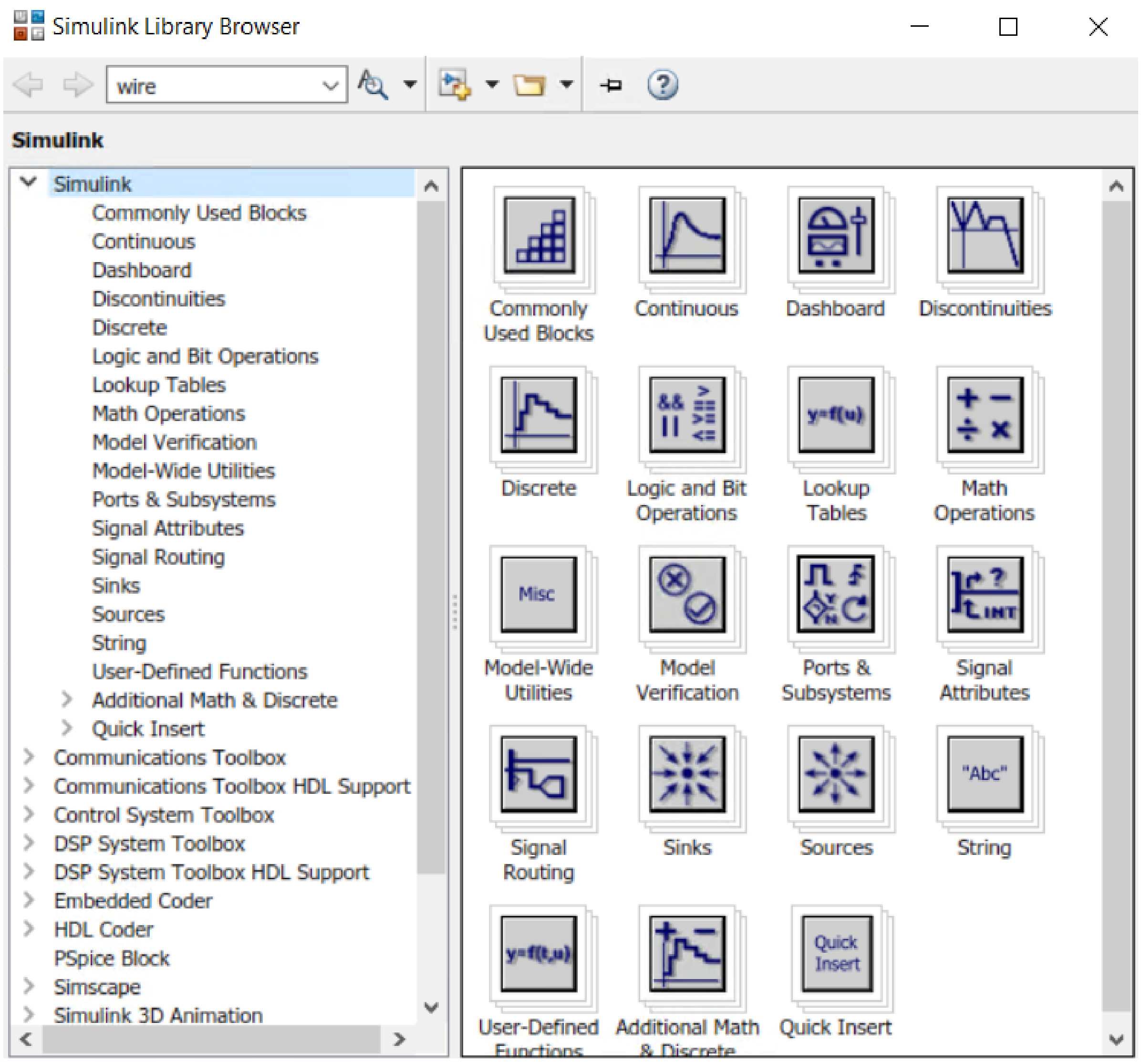


The Simulink model window opens.



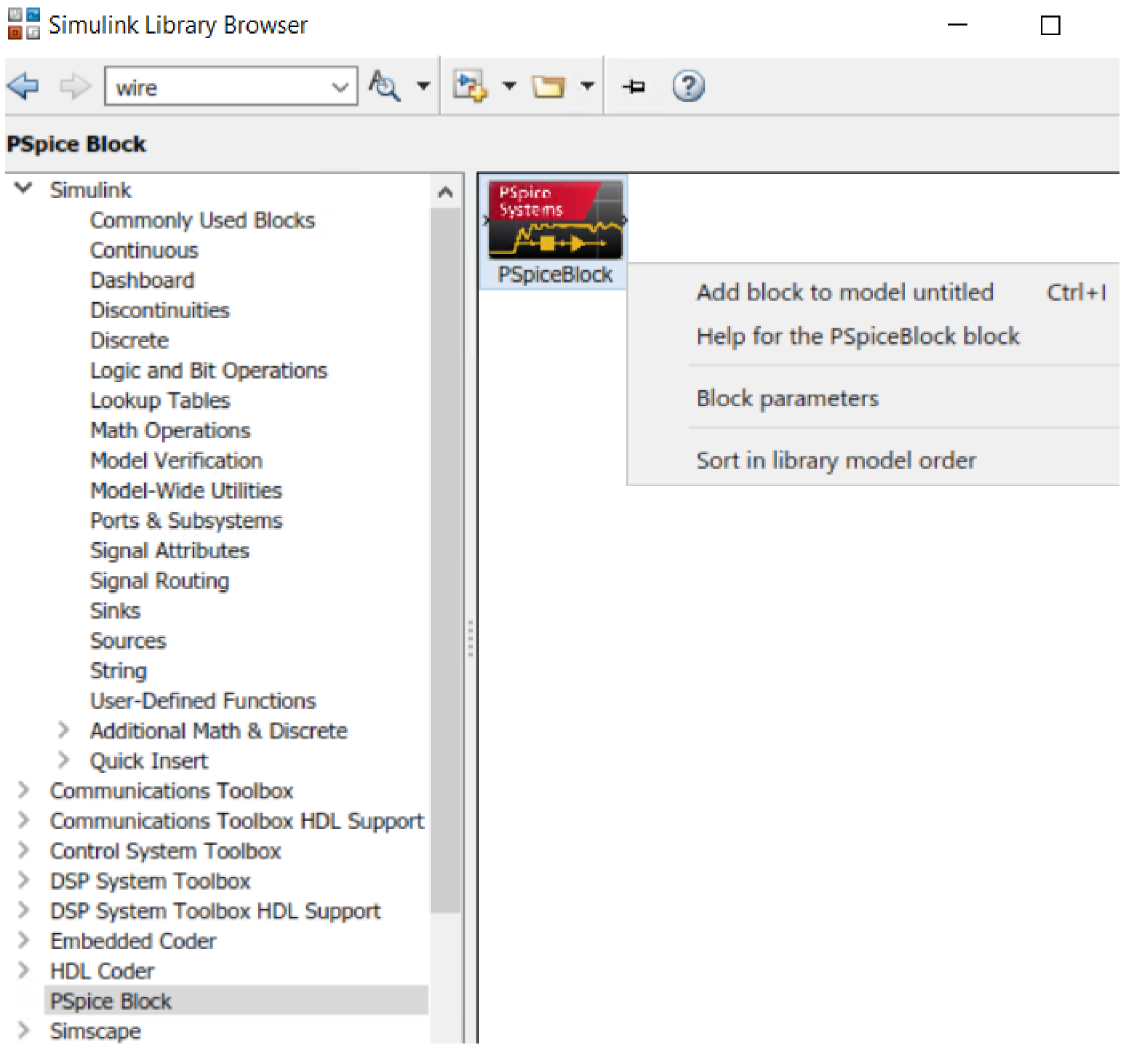
5. Click the *Library Browser* icon or select *View – Library Browser*.

The Simulink Library Browser window appears.



6. Select *PSpice Block* from the left pane.

7. Right-click *PSpice Block* in the right pane and select *Add block to model untitled*, or drag this icon into the Simulink project window.



8. Create a block diagram. In this example, create a block diagram as shown in Figure 7-3, using the blocks, Repeating Sequence/Sources and Scope/Sinks.

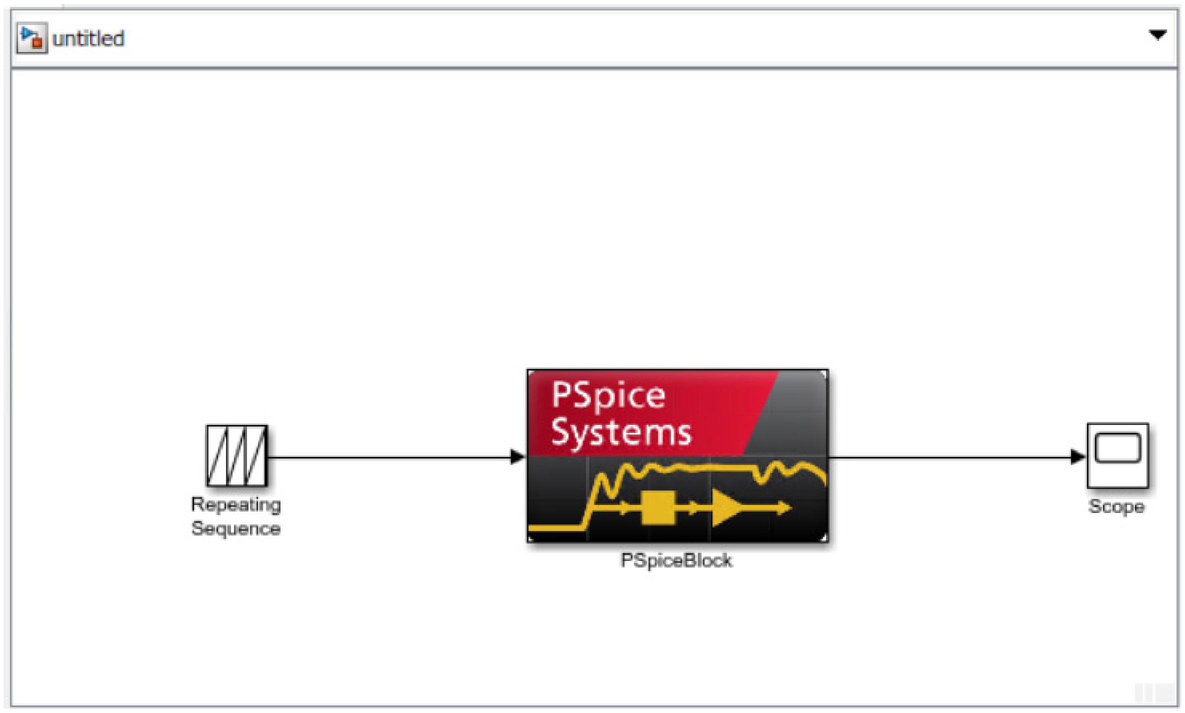
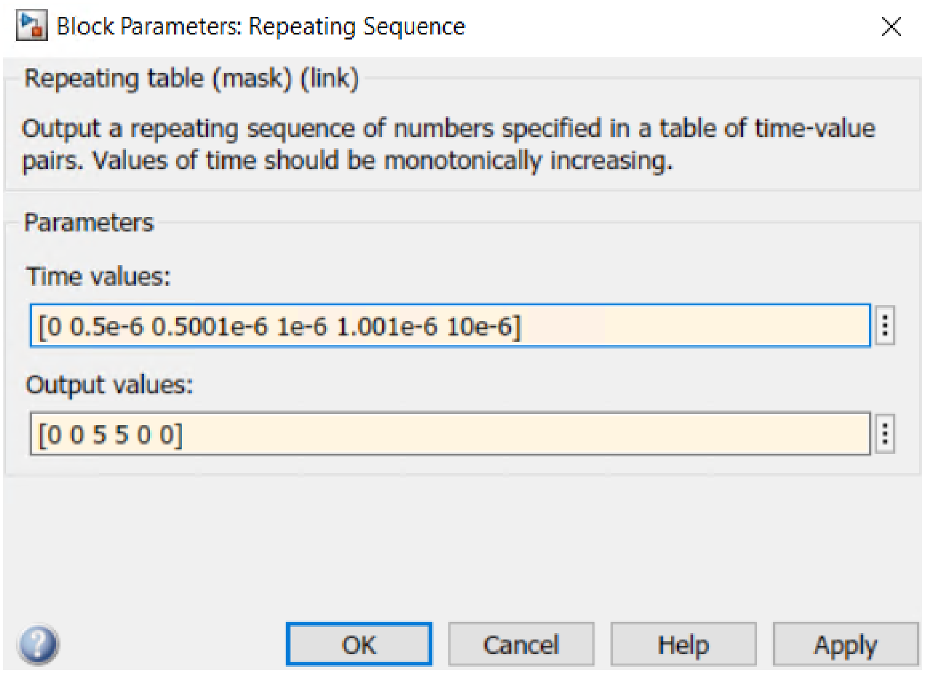


Figure 7-3 Block diagram using Repeating Sequence/Source and Scope/Sinks

9. Double-click the *Repeating Sequence* block to specify the repeating sequence parameters as shown in Figure 7-4.


Block Parameters: Repeating Sequence

Repeating table (mask) (link)

Output a repeating sequence of numbers specified in a table of time-value pairs. Values of time should be monotonically increasing.

Parameters

Time values:

[0 0.5e-6 0.5001e-6 1e-6 1.001e-6 10e-6]

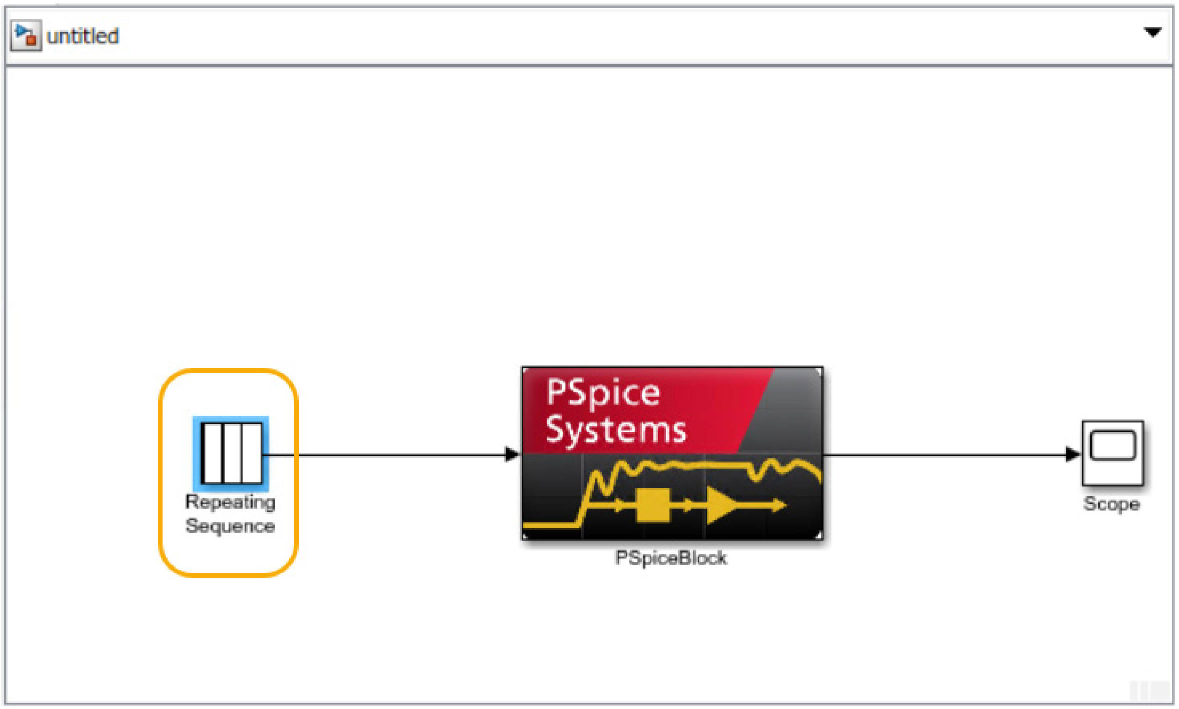
Output values:

[0 0 5 5 0 0]

Figure 7-4 Repeating Sequence parameters

10. Click *Apply* and *OK*.

The graphic of *Repeating Sequence* changes.



11. Select *File – Save* to save this Simulink model in the current directory. In this example, use the file name as `mosckt.slx`.

Important

You can place only one PSpice Block in a single Simulink model. If you want to incorporate multiple circuits, use Capture to create multiple circuit diagram pages within a project, and connect all the data lines in the Simulink model, which need to be connected to the circuit, to a single PSpice block.

Now that you have created a block and saved it, you will specify the block parameters.

Note: If you have existing PSpice SLPS designs, you need to manually migrate them to the new PSpice Co-simulation format before you can use those designs in new the PSpice Co-simulation environment. For details, see, Migrating Existing PSpice SLPS Designs to New PSpice Simulink Co-simulation Design.

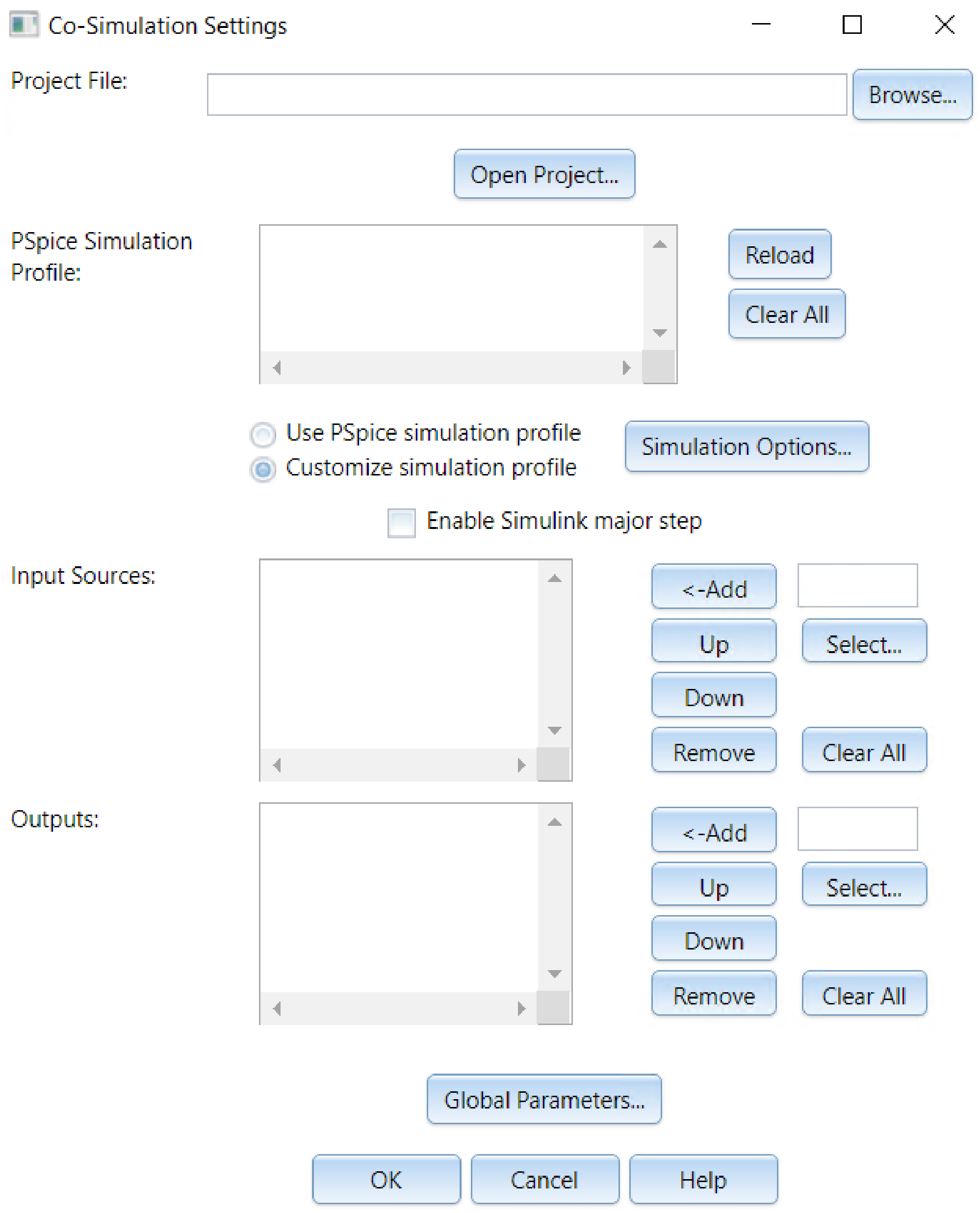
Setting Up PSpice Block Parameters

You need to specify the following main parameters for the PSpice block:

- Capture project file location

- Input sources and outputs for the PSpice block
- PSpice simulation and data collection options

Double-click the PSpice block to open the Co-Simulation Settings window.



Use PSpice simulation profile [✓ Customize simulation profile]

☐ Enable Simulink major step

Specify the following parameters:

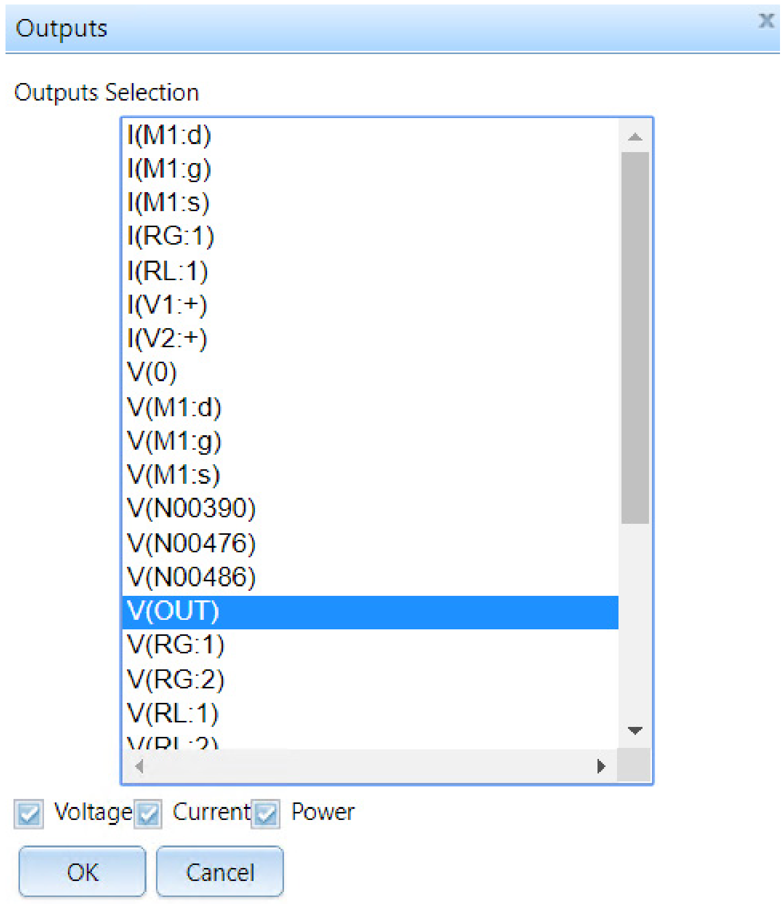
- Specify the Capture project file path for `MOSCKT.opj` in *Project File* field. You can also use the *Browse* button to specify the project file path.

- Click the *Select* button under *Input Sources*.

  The Inputs dialog box opens. Select the power source that you want to designate as the source for the PSpice block. In this example, select the source `V1`.

  Note: At least one input must be designated for PSpice Simulink Co-simulation, but you may want to assign a circuit with only output to PSpice Simulink Co-simulation (i.e. an oscillator). In this case, create a dummy circuit (for example, with a power source and resistance) within the circuit diagram page, do not connect it with the original circuit, and allocate some data to this power source.

- Click the *Select* button under *Outputs*. Specify an output for the PSpice block. In this example, select `V(OUT)` from the list displayed in the Outputs dialog box.



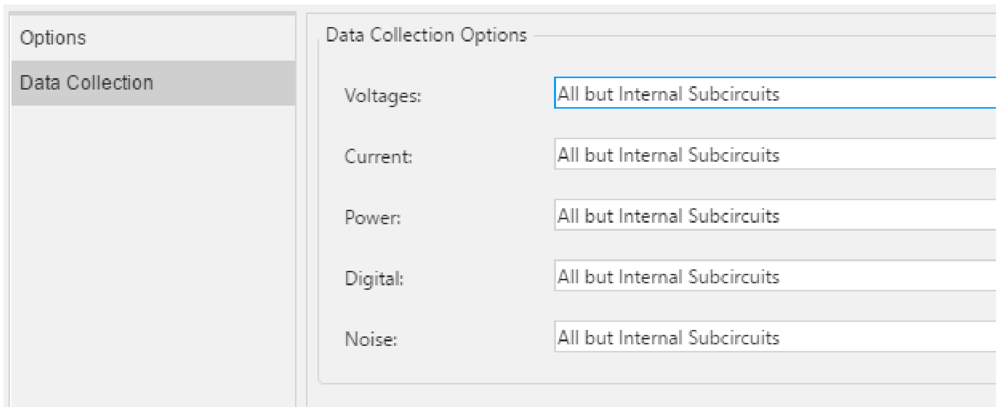
I(M1:d)
I(M1:g)
I(M1:s)
I(RG:1)
I(RL:1)
I(V1:+)
I(V2:+)
V(0)
V(M1:d)
V(M1:g)
V(M1:s)
V(N00390)
V(N00476)
V(N00486)
V(OUT)
V(RG:1)
V(RG:2)
V(RL:1)

- After specifying the input sources and output, you can also specify the following:
  - Simulation options
  - Data collection options

In the Co-simulation Settings window, click the *Simulation Options button.*

The Simulation Settings window appears.

Click the *Data Collection* tab and select the required option from the drop-down menu. If you select *All*, more data will be generated and the analysis speed will be slower. Therefore, it is not recommended to change the default value for any of these options unless you need it.



Voltages:	All but Internal Subcircuits
Current:	All but Internal Subcircuits
Power:	All but Internal Subcircuits
Digital:	All but Internal Subcircuits
Noise:	All but Internal Subcircuits

In the Co-simulation Settings window, click the *Global Parameters* button. The Global Parameters window appears. Specify any already existing global parameters in the Capture design in this dialog box.

After completing the required settings, click *OK* to close the Co-Simulation Settings window. You can now set up Simulink analyses.

Setting Up Simulink Analyses

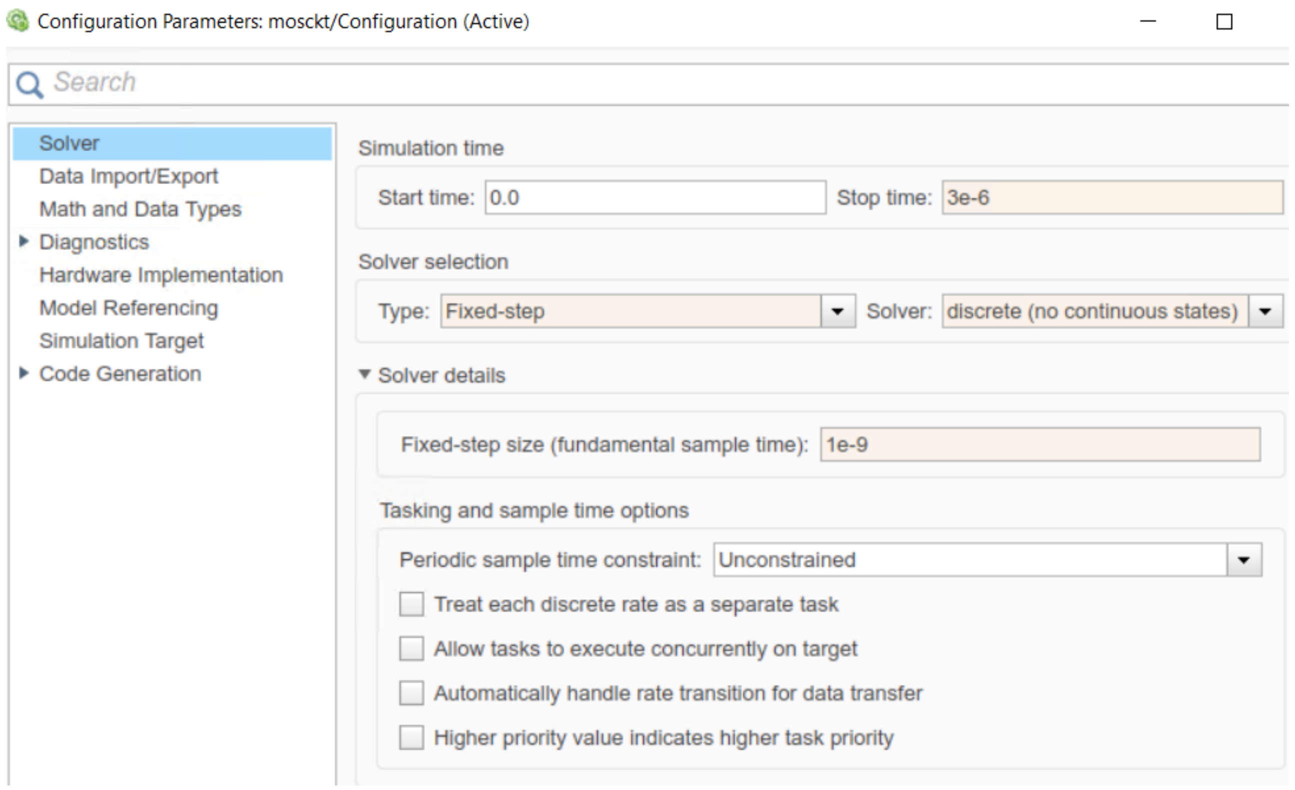
You can use the Configuration Parameters dialog box to set up a Simulink analyses.

In the Simulink project window, select *Simulation – Model Configuration Parameters*. The Configuration Parameters dialog box appears.

For this example, specify the following parameters in the Configuration Parameters dialog box:

1. Under *Simulation time*, specify `3e-6` (3usec) in the *Stop time* field.
2. Under *Solver options*:
   - ❑ In the *Type* list box, select `Fixed-step`.
   - ❑ In the *Solver* list box, select `discrete (no continuous states)`.

3. Under *Solver details*, in the *Fixed-step size (fundamental sample time)* list box, specify `1e-9`.



Configuration Parameters: mosckt/Configuration (Active)

Simulation time

Start time: 0.0
Stop time: 3e-6

Solver selection

Type: Fixed-step
Solver: discrete (no continuous states)

Solver details

Fixed-step size (fundamental sample time): 1e-9

Tasking and sample time options

Periodic sample time constraint: Unconstrained

- [ ] Treat each discrete rate as a separate task
- [ ] Allow tasks to execute concurrently on target
- [ ] Automatically handle rate transition for data transfer
- [ ] Higher priority value indicates higher task priority

Figure 7-5 Configuration Parameters dialog box

4. Click *OK*.

5. In the Simulink project window, select *File – Save* to save the parameter settings.

Simulating and Verifying Results

You can now simulate the analyses using Simulink and then verify the PSpice block results from both Simulink and PSpice.

To simulate using Simulink project window, select *Simulation – Run*. When analysis is finished, double-click *Scope* to display the waveform.

The waveform that appears in Simulink, shown in Figure 7-6, is similar to the one displayed PSpice.

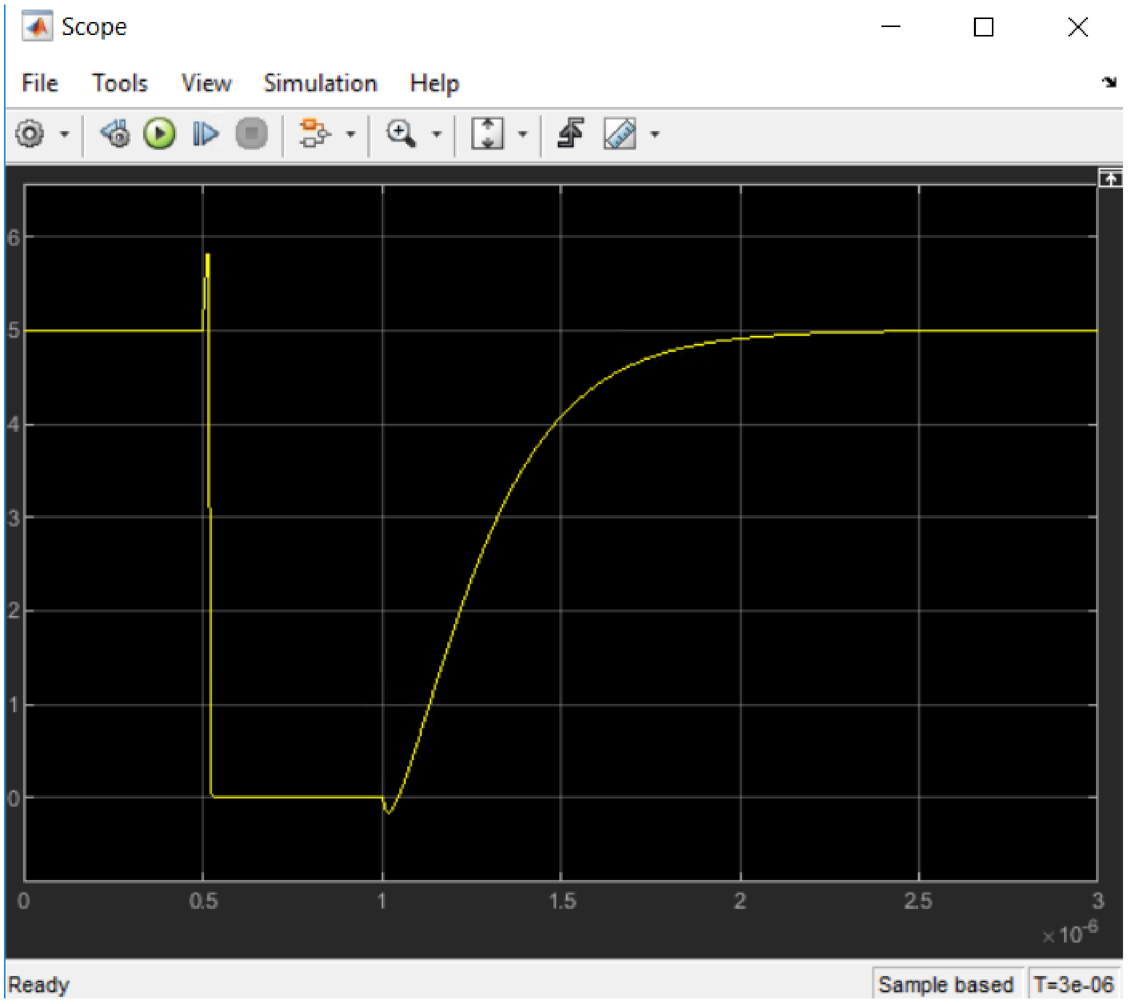


Figure 7-6 The waveform in the Scope window

The simulation results are also available in PSpice.

For this example, to see the PSpice Simulink Co-simulation simulation results in PSpice, perform the following tasks:

1. From PSpice, select *File – Open*.
2. Navigate to the location:

   `<current_project_directory>\MOSCKT-PSpiceFiles\SCHEMATIC1\Tran_slps_\`
3. Select `Tran_slps__SLPS.dat` and click *Open*.
4. Select *Trace – Add Trace*.

   The Add Traces dialog box opens.
5. Select `V(X_PSpiceBlock.Out).`
6. Click *OK*.

The PSpice Probe window displays the PSpice Simulink Co-simulation simulation result, as shown in Figure 7-7:

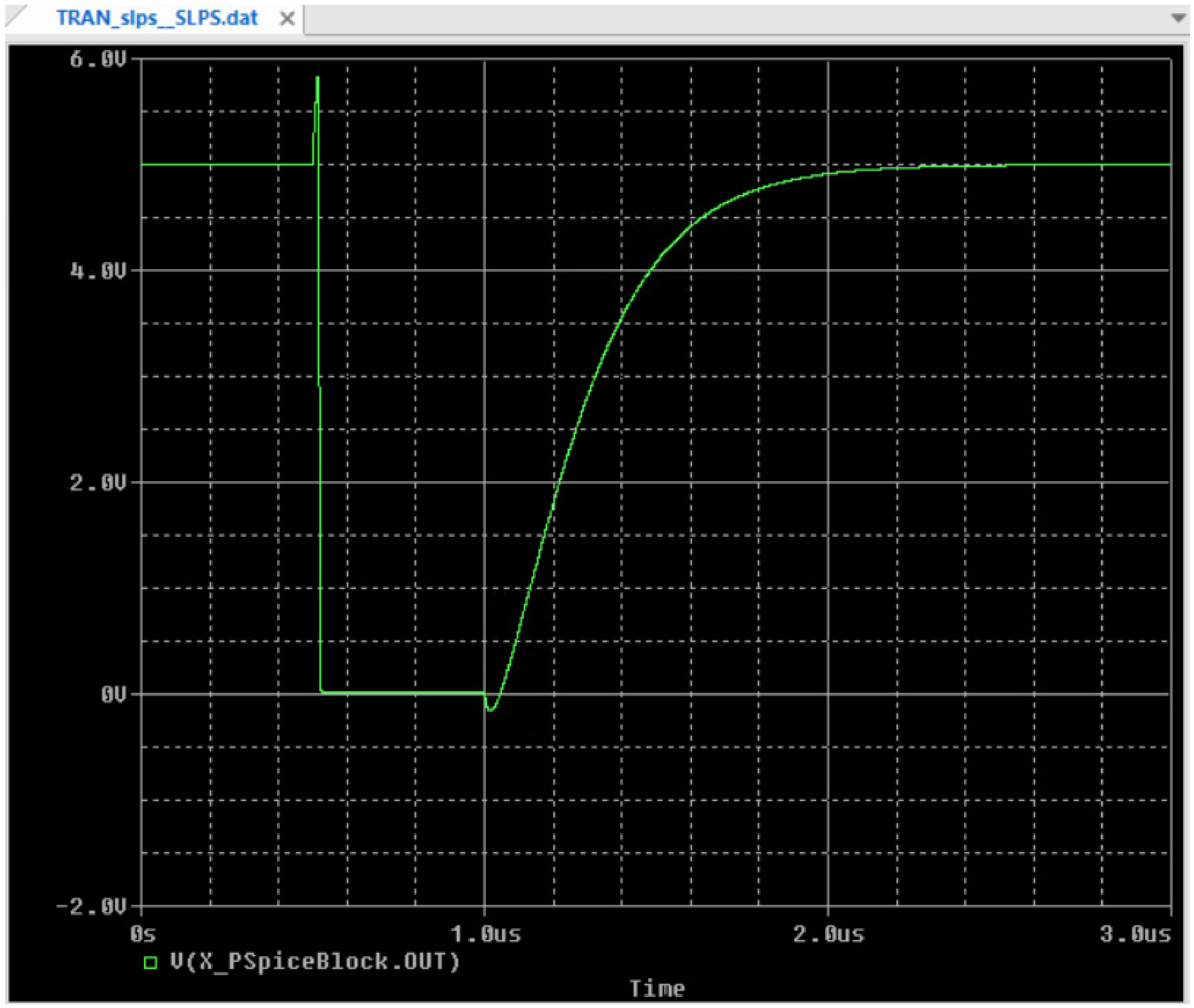


Figure 7-7 PSpice Simulink Co-simulation simulation result in PSpice

Important

Before co-simulation, if select `None` in the *Data Collection* tab of the Simulation Settings window, simulation results are not saved in PSpice.

Migrating Existing PSpice SLPS Designs to New PSpice Simulink Co-simulation Design

PSpice Simulink Co-simulation allows you to easily migrate the existing PSpice SLPS designs to new PSpice Simulink Co-simulation designs.

To use the existing designs in the new solution, you need to only replace the existing SLPS block with the new PSpice block. Once done, the existing designs will run in the new solution.

Migrating Single SLPS Block

To replace the old SLPS block in existing designs, do the following:

1. Perform the following pre-migration tasks:
   a. Open the old Simulink design in MATLAB.
   b. Double-click the old SLPS block.
   c. Note the old interface settings. These will be used to specify the settings in the new PSpice block.
2. Migrate the old Simulink designs to work in the new PSpice Simulink Co-simulation solution. To do so:
   a. Ensure that:
      - You have set up the PSpice Simulink Co-simulation solution. For details, see Setting Up the PSpice Simulink Co-simulation Solution.
      - You remove the existing PSpice-SLPS solution path (`<installation_directory>\tools\pspice\slps`) from the MATLAB's path settings.

   b. Open MATLAB from PSpice using *Tools – MATLAB – Co-Simulation*.
   c. In the MATLAB Command Window, browse to the directory of the old design.
   d. In the left panel, double-click the old `.mdl/.slx` file.

      The Simulink project window opens.
   e. Click the *Library Browser* icon or select *View – Library Browser*.

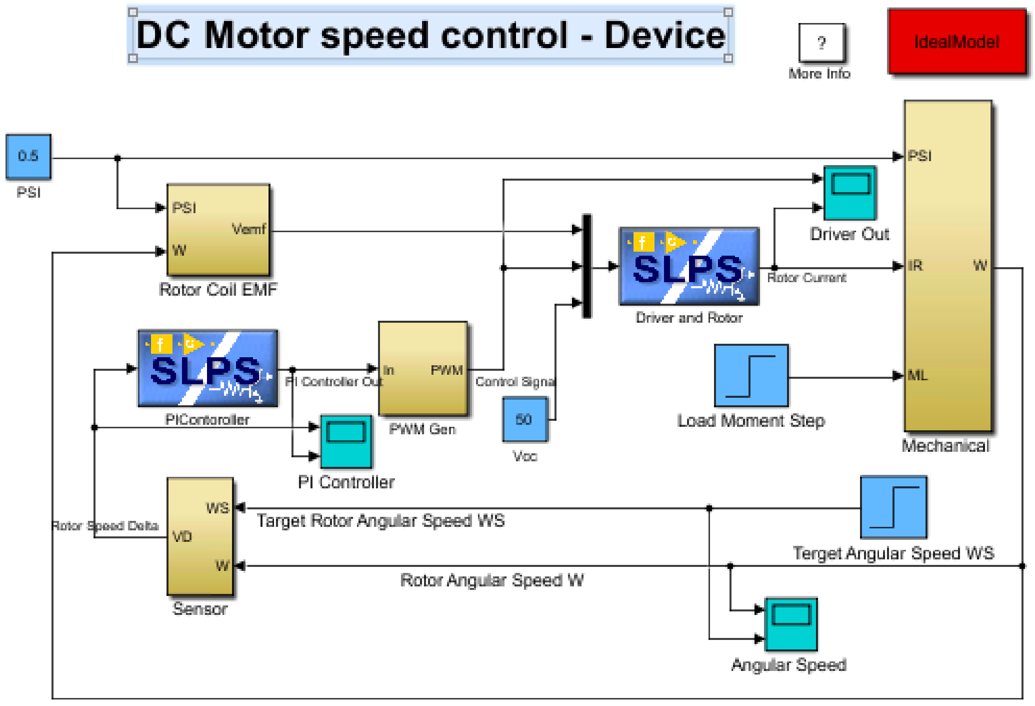
      The Simulink Library Browser window appears.
   f. Select *PSpice Block* from Simulink Library Browser.
   g. Drag and drop the PSpice block to the Simulink project window.
   h. Replace the old SLPS block with the new PSpice block, placed in the previous step.
   i. Double click this new block and specify the input and output port connections that you had noted for the old block.
   j. Save the Simulink design.

Migrating Multiple SLPS Blocks

If you have more than one SLPS blocks in your design, perform the following migration tasks:

1. Open the design containing multiple SLPS blocks in MATLAB.
2. Double click each SLPS block to:
   - ❑ See the Capture projects they are mapped to
   - ❑ Note their interface settings. These will be used to specify the new settings in the single new PSpice block.

   For example, in the following design there are two SLPS blocks mapping to two different Capture projects.

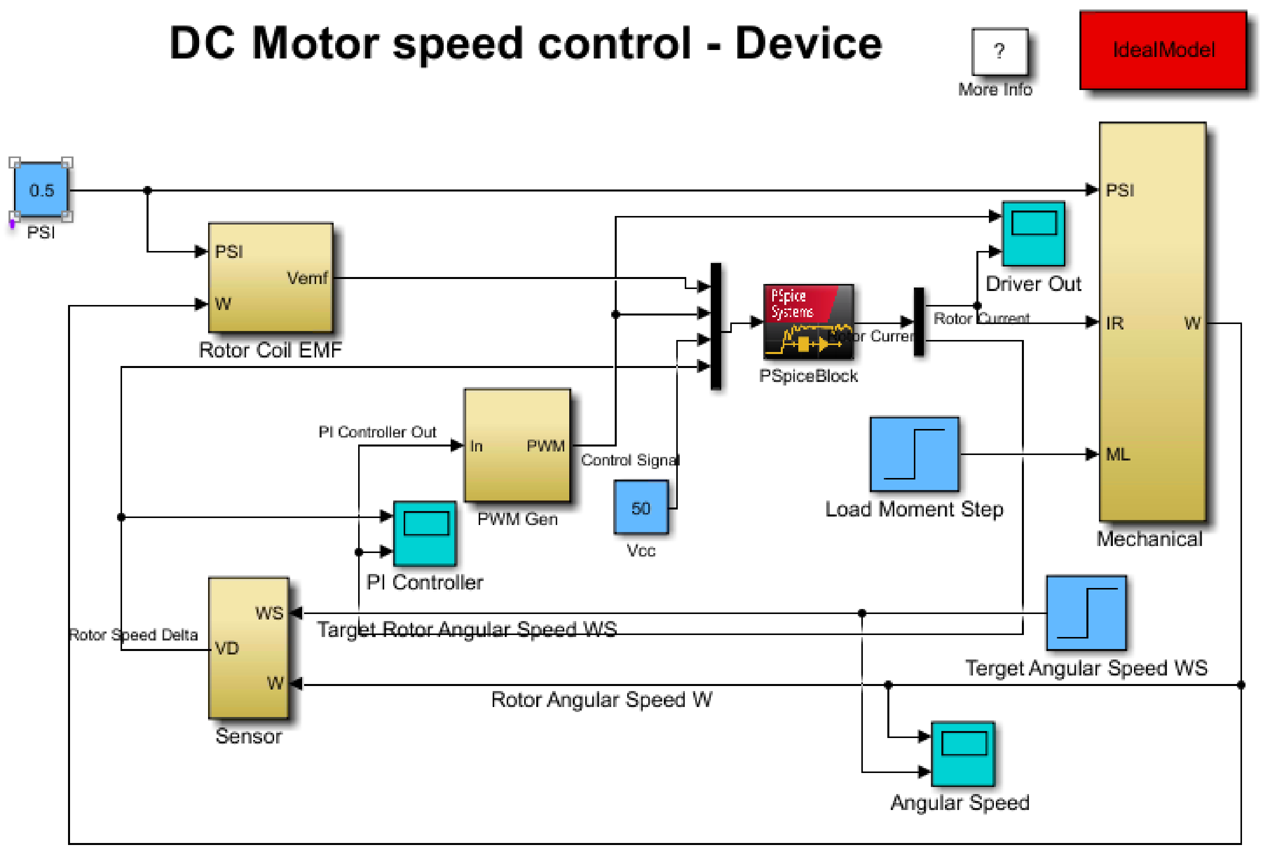


3. For this example, open both the projects in Capture.
4. Create a new Capture project and add these two designs to this project.
5. Save the project and run simulation.
6. Open MATLAB from PSpice using *Tools – MATLAB – Co-Simulation*.
7. In the MATLAB Command Window, browse to the directory that contains the old design with multiple SLPS blocks.

8. From the left panel, open this design and modify it to:

a. Replace the two old blocks with the new PSpice block.

b. Connect the Simulink interfaces mapping of the old blocks to the new block.



c. Double click the new PSpice block and specify the new interface settings in the Co-Simulation Settings window.

d. Save the design.

e. In the Simulink project window, select *Simulation – Run*.

8

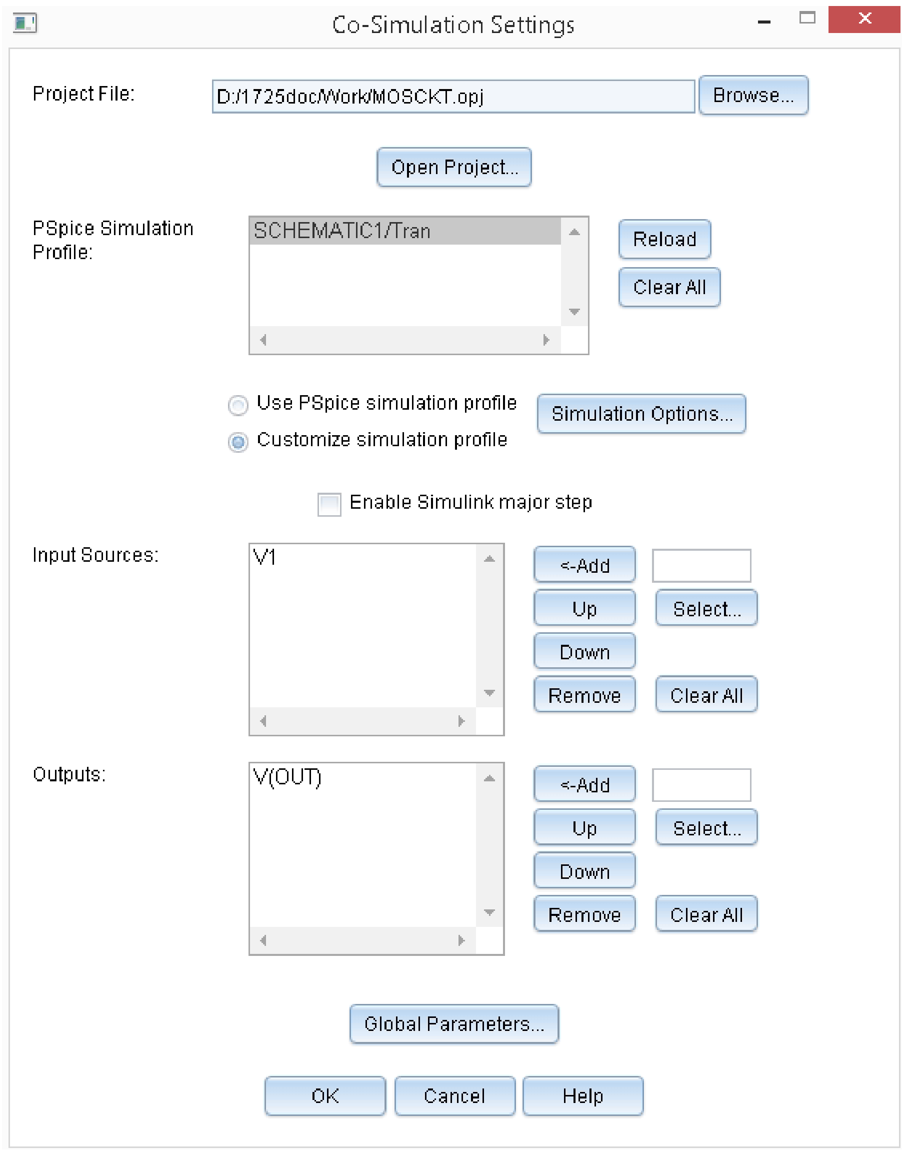
Interfacing PSpice Designs in MATLAB

This chapter explains how to prepare PSpice circuits by creating CIR files and describes the various options of the Co-Simulation Settings window. It also explains the data exchange between Simulink and PSpice:

- Using the Co-Simulation Settings Window on page 90
- Setting up Simulink Simulations on page 95

Using the Co-Simulation Settings Window

You can open the Co-Simulation Settings window by double-clicking the PSpice Block in the Simulink project window, as shown in Figure 8-1.



Project File: D:/1725doc/Work/MOSCKT.opj

PSpice Simulation Profile: SCHEMATIC1/Tran

( ) Use PSpice simulation profile
(●) Customize simulation profile

[ ] Enable Simulink major step

Input Sources: V1

Outputs: V(OUT)

Figure 8-1 Co-simulation Settings window

The Co-simulation Settings window allows you to specify the following parameters for the PSpice block:

- *Project File*: Designates a Capture project file (`*.opj`) containing a PSpice schematic to be assigned.
- *Open Project*: Opens the designated project in Capture. If a project has not been designated, Capture will open without a project. You cannot use this button to open a designated project if Capture has already started. Either close Capture before clicking this button, or manually open the designated project from Capture.

  Note: If Capture is already open when you start Capture by pressing click Open Project from Co-Simulation Settings, you cannot open a project.
- *Reload*: Updates information about changes done in the schematic.
- *Clear All*: Clears the items in the Co-Simulation Settings window.
- *PSpice Simulation Profile*: Lists all simulation profiles in the project specified in the Capture Project File text box.
- *Use PSpice simulation profile*: Uses the default simulation profile settings.
- *Customize simulation profile*: Allows you to use the customized simulation settings.
- *Simulation Options*: Allows you to customize the co-simulation settings using the Simulation Settings dialog box. For more information on simulation settings dialog box parameters, see the Options and Data Collection in Simulation Settings chapter.
- *Enable Simulink major step*: Select this check box to enable Data Exchange between Simulink and PSpice only at Simulink major steps. Selecting this option may increase the transient timing error between Simulink and PSpice. However, calculation time may be shorter because of a decrease in the number of calculation steps, that is, data exchanges between Simulink and PSpice.
- *Input Sources*: Designates the voltage source (V*) and current source (I*) for supplying input data (from Simulink to the PSpice block) into the circuit. If a voltage source is selected, the input data will be supplied to the circuit as a voltage value, and if a current source is selected, it will be supplied as a current value. The sequence listed here is the sequence of input signals to the PSpice block. At least one input source must be specified.
- Inputs: Click *Select* in the *Input Sources* section to open this dialog box. It lists all power sources contained in the circuit referenced by the CIR file specified in the *PSpice Simulation Profile* list box.
- Add/Up/Down/Remove buttons for Input Sources: Adds, changes sequence, and deletes items, respectively, from the *Input Sources* list box.
- *Outputs*: Designates data in the circuit to be output from the PSpice block to Simulink. The sequence listed here is the sequence of output signals to Simulink.

You can set node voltage, current passing through a device, and dissipation of a device as values for the PSpice Simulink Co-simulation output. The specification format for output values corresponds to that of PSpice.

- Node voltage:

  Specified by the syntax: `V([`*`NODENAME`*`])`

  Where *`NODENAME`* is the name of the node in the circuit.

- Current passing through a device or pin inflow current:

  Specified by the following syntax: `I ([`*`DEVICENAME`*`]:[`*`PINNAME`*`])`

  Where *`DEVICENAME`* is the reference name of the device and *`PINNAME`* is name of pin whose current is referred.

  Example 1:

  `I(Q1:B)`

  Where:

  - `Q1` is the device name
  - `B` is the pin name

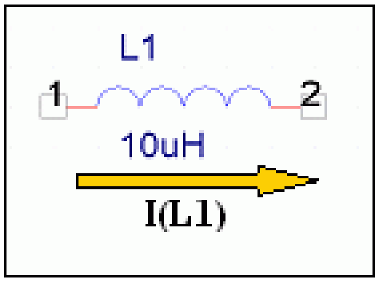
  Example 2:

  `I(VDC:+)`

  Where:

  - `VDC` is the device name
  - `+` is the pin name

  The sign of current passing through a device conforms to PSpice. The direction from Pin `1` to Pin `2` of the device is defined as positive, as shown in the following figure.



- Power dissipation of device:

Specified by the syntax: `W([`*`DEVICENAME`*`])`

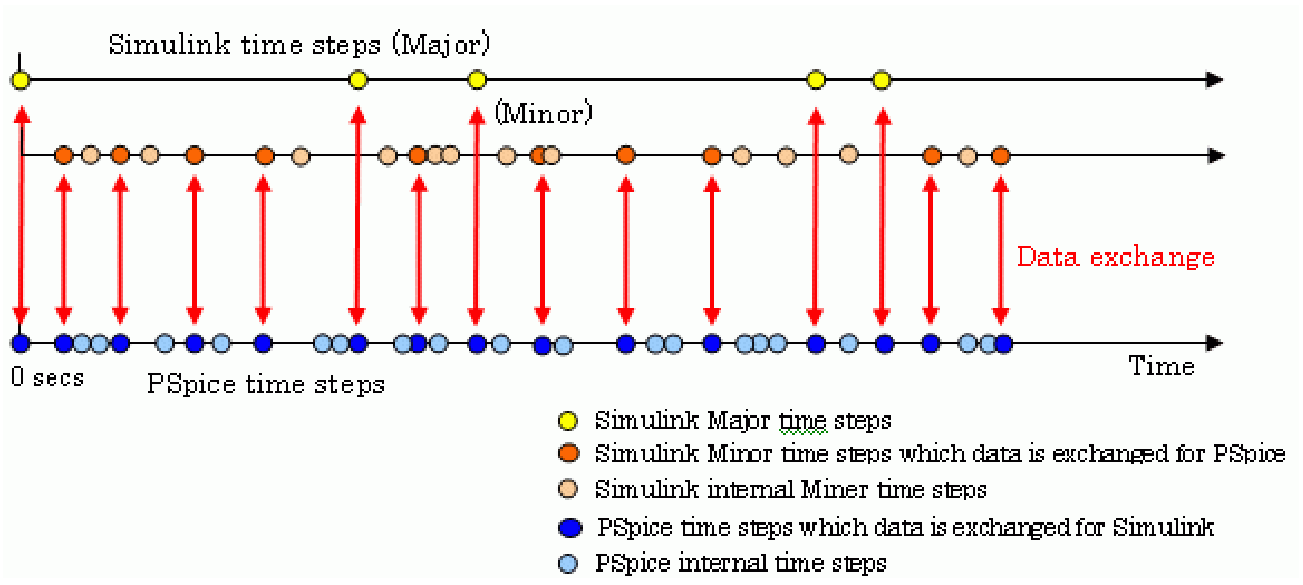
Power dissipation is calculated as I*V. However, since the absolute value of current is taken, the current directions do not matter.

- *Outputs*: Click *Select* in the *Outputs* section to open the *Outputs* dialog box. It lists all output variables in the simulation profile specified in the *PSpice Simulation Profile* list box.
- Add/Up/Down/Remove buttons for Outputs: Adds, changes sequence, and deletes items, respectively, from the *Outputs* list box.
- Global Parameters: Click this button to specify if any global parameters are defined in the design.

Data Exchange between Simulink and PSpice

When a PSpice block is placed in a Simulink model, two different analysis engines are used for simulation, where transient analysis will be done at their respective time step.

Data exchange between Simulink and PSpice A/D via PSpice Simulink Co-simulation is shown in the following figure:


Simulink time steps (Major)
(Minor)
Data exchange
0 secs
PSpice time steps
Time
Simulink Major time steps
Simulink Minor time steps which data is exchanged for PSpice
Simulink internal Miner time steps
PSpice time steps which data is exchanged for Simulink
PSpice internal time steps

In Simulink, you cannot check an internal property of PSpice that has a smaller time step than that of Simulink. In addition, there are also some Simulink minor steps for which data is not exchanged with PSpice, as you can see from the figure above.

To check the internal results of PSpice, designate one of the items other than `None` under *Data Collection* in the Simulation Settings window. Then use PSpice A/D to open the PSpice data file `Tran_slps__SLPS.dat` after completing an analysis.

Setting up Simulink Simulations

You can use the Configuration Parameters dialog box to specify the settings for the Simulink simulations. To open this dialog box, in the Simulink model window, select *Simulation – Model Configuration Parameters*.

PSpice Simulink Co-simulation can use any one of the items from the *Type* list under *Solver* options:

- `Fixed-step`
- `Variable-step`

Because PSpice Simulink Co-simulation data exchange only operates with each Simulink step, in order to avoid overlooking phenomena from the PSpice circuit, Simulink's maximum step size (Simulink's fixed step size when fixed step is selected) must be a sufficiently small value. The value, however, should not be smaller than is needed, or Simulink's overall analysis may become slow.

As you can see from Figure 8-2, you cannot obtain the correct waveform if the maximum step size is set to a large value.

:

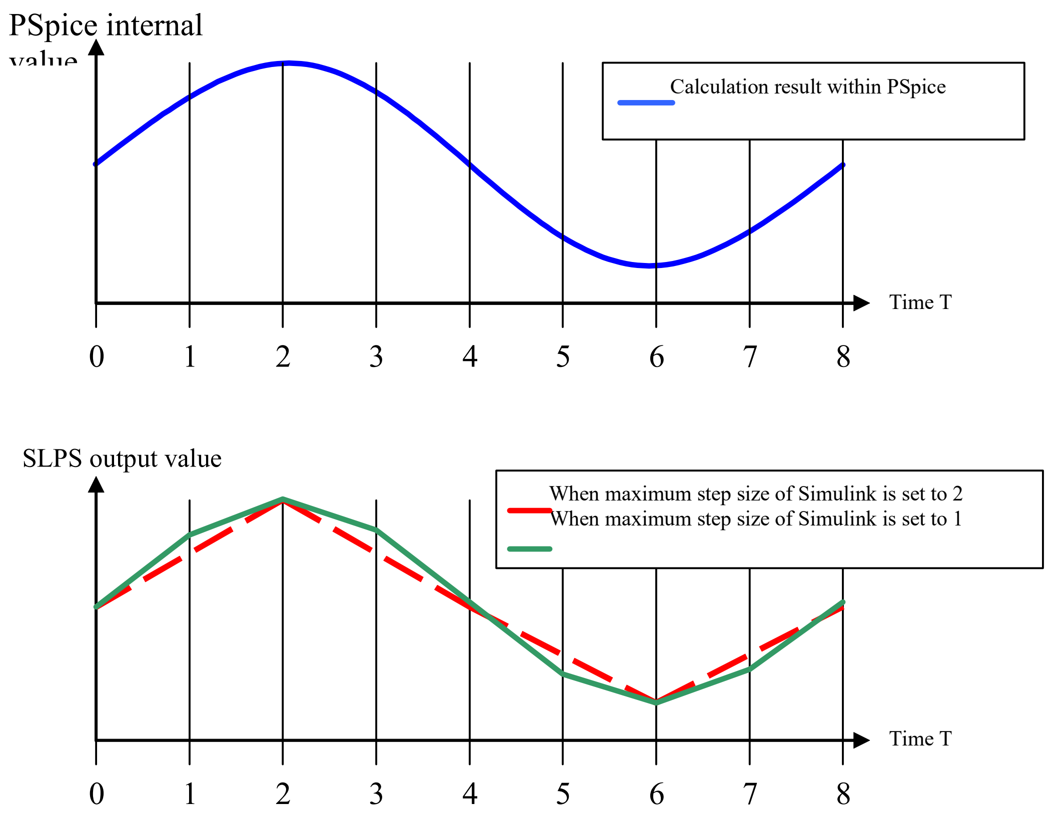


Figure 8-2 Waveform with large maximum step size value

If you want to see a waveform within PSpice in order to determine the step size, check it by starting PSpice A/D and opening the PSpice data file, though it will not be displayed in Simulink.

Note: When the PSpice block is inserted in the feedback loop, calculation in the PSpice block is performed by PSpice, which operates in a different memory space than that of Simulink. In principle, a delay of one Simulink cycle step will occur between the PSpice block's input and its output. To minimize this effect, Simulink’s time steps must be sufficiently small.

9

Analyzing Simulink Models

This chapter explains how to analyze a Simulink model, change parameters to avoid convergence errors, and verify results using PSpice:

Analyzing Co-Simulation Results

You can start a simulation to analyze a Simulink model containing an PSpice block. To do so, choose *Simulation – Run* from the Simulink model window.

If a convergence error occurs within PSpice during analysis with Simulink, increase the option parameter ITL4 of PSpice up to the value specified in ITL4 Max in Simulation Settings, and retry analysis. However, if PSpice cannot get a result, a convergence error message is displayed.

You can recalculate convergence by changing the analysis options using the Simulation Settings dialog box.

After you change these parameters, click OK. PSpice will continue with the calculation using the changed analysis options.

Normally, if a convergence error occurs during analysis, set the value of ITL4 to a large value, such as 1000. As a result, you can improve convergence without changing analysis accuracy.

Viewing Analysis Results

You can view the Simulink model analyses results using the following two methods:

- Check the output of PSpice Simulink Co-simulation using the *Scope* interface node in Simulink project window.
- View the results obtained in co-simulation by opening the PSpice data file in PSpice. This will display the waveform in the PSpice A/D window.

When you call PSpice from PSpice Simulink Co-simulation to perform an analysis, PSpice creates a data file (`*.dat`) and an output file (`*.out`). A PSpice data file stores calculation data within PSpice Simulink Co-simulation, either for the entire circuit or for the part selected as PSpice Simulink Co-simulation output. If you select `None` as the PSpice Data Collection option in the Simulation Settings window, PSpice will create an empty data file. The output file contains information, such as the analysis log, in text format and is used to verify internal errors that might occur during analysis. The data file and the output files are created in the directory designated for the CIR file.

10

Options and Data Collection in Simulation Settings

Use the *Options* tab of the Simulation Settings dialog box to fine-tune how PSpice performs calculations for PSpice Simulink Co-Simulation, as well as what information to save to the simulation output file (`*.OUT`):

Analog Simulation Options

Use the Analog Simulation settings to fine-tune analog simulation accuracy, set iteration limits, set operating temperature, and specify MOSFET parameters.

In the Simulation Settings dialog box, the option names shown before the text box correspond to the option names used in the PSpice `.OPTIONS` command. For more information about this command, refer to the *PSpice Reference Guide*.

Click this option…	To do this…
General	Enter values for speed level, tolerances, and minimum conductance.
Auto Converge	Suggest relaxed limits for various options that PSpice can modify during a simulation to achieve convergence.
MOSFET Option	Enter values for the default drain area, default source area, default length, and default width.

The following tables defines all the options for the Analog Simulation category:

Flag options in Analog Simulation	**Meaning**
`ADVCONV`	Enables all convergence algorithms, such as Pseudo Tran, STEPGMIN, and step sources. ON by default.

Flag options in Analog Simulation	Meaning
`AutoConverge`	Suggest relaxed limits for various options that PSpice can modify during a simulation to achieve convergence.
`Restart`	Restart the convergence calculation
`PREORDER`	Presorts the matrix diagonal by Markowitz counts.

Options	Description	Units	Default
General			
`SPEED_LEVEL`	increases simulation performance by optimizing switching behavior of models. If increase in simulation performance is not needed, set SPEED_LEVEL=0.	-	3
`RELTOL`	relative accuracy of V and I		0.001
`VNTOL`	best accuracy of voltages	volt	1.0 uV
`ABSTOL`	best accuracy of currents	amp	1.0 pA
`CHGTOL`	best accuracy of charges	coulomb	0.01 pC
`GMIN`	minimum conductance used for any branch	ohm^{-1}	1.0E-12
`ITL1`	Maximum number of iterations of convergence calculation during bias point analysis		150.0
`ITL2`	Maximum number of iterations of convergence calculation during DC analysis		20
`ITL4`	Maximum number of iterations of convergence calculation at each time step during transient analysis		10
`TNOM`	Nominal Temperature value	Celsius	27 degrees Celsius
`THREADS`	maximum number of threads. Set to 0 for engine default. `THREADS=1` implies single-thread.		Number of cores/2

Options	Description	Units	Default
Auto Converge			
`ITL1, ITL2, ITL4, RELTOL, ABSTOL, VNTOL`	Same as *General* section		
`PIVTOL`	absolute magnitude required for pivot in matrix solution		1.0E-13
MOSFET Option			
`DEFAD`	MOSFET default drain area (AD).	$meter^2$	0.0
`DEFAS`	MOSFET default source area (AS).	$meter^2$	0.0
`DEFL`	MOSFET default length (L).	meter	100.0 u
`DEFW`	MOSFET default width (W).	meter	100.0 u

Analog Advanced Options

Use the Analog Advanced settings to enter values for the total transient iteration limit, relative magnitude for matrix pivot, and absolute magnitude for matrix pivot.

In the Simulation Settings dialog box, the option names shown before each text box correspond to the option names used in the PSpice `.OPTIONS` command. For more information about this command, refer to the *PSpice Reference Guide*.

Click this...	To do this…
General	Enter values for speed level, tolerances, and minimum conductance.
Bias Point	Suggest relaxed limits for various options that PSpice can modify during a simulation to achieve convergence.
Transient	Enter values for the default drain area, default source area, default length, and default width.

The following tables defines all the options in the tab for the Analog Advanced category:

Flag option for Analog Advanced	Meaning
`NOGMINI`	Specifies not to add GMIN across current sources.
`BRKDEPSRC`	Sets automatic break-points for behavioral sources.
`CONVAID`	Generates .1OP file for debugging purpose when convergence fails.
`STEPGMIN`	Enables GMIN stepping. This causes a GMIN stepping algorithm to be applied to circuits that fail to converge. GMIN stepping is applied first, and if that fails, the simulator falls back to supply stepping.
`NOSTEPSRC`	Do not run source stepping algorithm for bias point convergence.
`NOSTEPDEP`	Do not step dependent sources during source stepping algorithm for bias point convergence.
`GMINSRC`	Enables step GMIN inside source-stepping
`PSEUDOTRAN`	Uses Pseudo-Transient Method
`TRANCONV`	Enables alternate path search if transient simulation fails.

Options	Description	Units	Default
General			
`ITL5`	total repeating limit for all points for transient analysis (ITL5=0 means ITL5=infinity)	-	0.0
`PIVREL`	relative magnitude required for pivot in matrix solution	-	1.0E-3
`PIVTOL`	absolute magnitude required for pivot in matrix solution	-	1.0E-13
`SOLVER`	performance package solution algorithm (Solver = 0 selects the original solution algorithm; Solver = 1 selects the advanced solution algorithm)	-	1

`DMFACTOR`	Sets the relative factor for minimum delta. The value specifies the relative value by which the minimum time step size is changed. The value should be less than or equal to 1 and a factor of `10`, such as `.1`, `.001`, or `.0001`.		
`WCDEVIATION`	worst case deviation. It can have double values between 0 and 1.	-	Same as `RELTOL`
`LIMIT`	the absolute voltage limit. The default, 0, specifies that there is no limit on data values. You can modify it to a large value, such as 1e12, to eliminate overflow errors, especially when using exponential sources.	-	0
`DIODECJO`	Minimum value for Diode junction capacitance	ohm	0
`DIODERS`	Minimum value for Diode ohmic resistance	ohm	0
`BJTCJ`	minimum value for BJT Base-collector zero-bias depletion capacitance (Cjc), Base-emitter zero-bias depletion capacitance (Cje), and zero-bias collector substrate capacitance (Cjs)	farad	0
Bias Point			
`GMINSTEPS`	the GMIN stepping size in integer (any positive value). Set to 0 for engine default.	-	Same as `ITL1`
`ITL6`	the number of steps of the source stepping algorithm. Can have any positive integer value. Set to 0 for engine default.	-	Same as `ITL1`
`PTRANSTEP`	number of steps for a pseudo transient analysis to find the operating point. Can be any positive integer value. Set to 0 for engine default.	-	Same as `ITL1`
Transient			
`METHOD`	integration method (values can be either TRAPEZOIDAL or GEAR)	-	-
`TRTOL`	tolerance for integration error calculated using transient analysis. It is a relative tolerance where a higher TRTOL value results in bigger time steps and reduced accuracy. The TRTOL value should NOT be greater than 1/RELTOL.	-	7

CSHUNT	shunt capacitance added from all nodes of the design to GND. Recommended value is 1pF.	farad	0

Gate Level Simulation Options

Use the Gate Level Simulation settings to set timing, I/O levels for interfaces, drive strength, and error message limits.

Click this…	To do this…
General	Enter values to set delay or initial state in flip-flops or latches.
Advanced	Enter values for the minimum output drive resistance, maximum output drive resistance, overdrive ratio, default delay calculation, and error message limits.

The following tables defines all the options for the Gate Level Simulation category:

Flag option for Gate Level Simulation	**Meaning**
NOPRBMSG	Suppresses simulation error messages in Probe data file.

Options	**Description**	**Units**	**Default**
General			
DIGMNTYMX	default delay selector: 1=min, 2-typical, 3=max, 4=min/max	-	2.0
DIGINITSTATE	sets initial state of all flip-flops and latches in circuit: 0=clear, 1=set, 2=X	-	2.0
DIGIOLVL	default digital I/O level: 1-4;	-	1.0
Advanced			
DIGDRVF	minimum drive resistance (Input/Output UIO type model, DRVH (high) and DRVL (low) values)	ohm	2.0

`DIGDRVZ`	maximum drive resistance (UIO type model, DRVH and DRVL values)	ohm	20K
`DIGOVRDRV`	ratio of drive resistances required to allow one output to override another driving the same node	-	3.0
`DIGMNTYSCALE`	scale factor used to derive minimum delays from typical delays	-	0.4
`DIGTYMXSCALE`	scale factor used to derive maximum delays from typical delays	-	1.6
`DIGERRDEFAULT`	default error limit per digital constraint device	-	20.0
`DIGERRLIMIT`	maximum digital error message limit	-	0

Output File Options

Use the Output File settings to select the types of information PSpice saves to the simulation output file.

The following table defines all the options for the Output File category:

Flag option for Output File	Meaning
`ACCT`	Summary and accounting information is printed at the end of all the analyses (refer to your *PSpice User's Guide* for further information on ACCT).
`EXPAND`	Lists devices created by subcircuit expansion and lists contents of the bias point file.
`LIBRARY`	Lists lines used from library files.
`LIST`	Lists a summary of the circuit elements (devices).
`NOBIAS`	Suppresses the printing of the bias point node voltages.
`NODE`	Lists a summary of the connections (node table).
`NOECHO`	Suppresses a listing of the input file(s).
`NOMODE`	Suppresses listing of model parameters and temperature updated values.
`NOOUTMSG`	Suppresses simulation error messages in output file.
`NOPAGE`	Suppresses paging and the banner for each major section of output.
`OPTS`	Lists values for all options.

Options	Description	Units	Default
`NUMDGT`	Number of digits in printed values.		This is 4 by default.
`WIDTH`	same as the .WIDTH OUT= statement (can be set to either 80 or 132)		Default is 80.

Data collection options for simulation profiles

Use the *Data Collection* tab of the Simulation Settings dialog box to restrict the captured simulation data. This is especially useful for large circuit designs that produce more data than you need for waveform analysis.

Data Collection Options

Option	Description
`All`	All data will be collected and stored.
`All but Internal Subcircuits`	All data will be collected and stored except for internal subcircuits of hierarchical designs (top level data only).
`At Markers Only`	Data will only be collected and stored where markers are placed.
`None`	No data will be collected.

A

Sample Circuits

The PSpice Simulink Co-Simulation demonstrations and example files for:

- Capture are available at:

  `<installation_directory>\tools\pspice\capture_samples\CoSimulationDemos\PSpiceCoSim`

- Design Entry HDL are available at:

  `<installation_directory>\tools\pspice\concept_samples\CoSimulationDemos\PSpiceCoSim`

This section explains some circuits, which explain the use of the PSpice Simulink Co-Simulation interface.

- RLC Circuit
- RC Circuit
- Nonlinear Load (NLLOAD)
- PLL Model
- Switched Reluctance Motor Control
- Switching Power Supply
- DC Motor Control System

RLC Circuit

Use a simple RLC circuit (RLCCir), as shown in the following figure to create the schematic diagram.

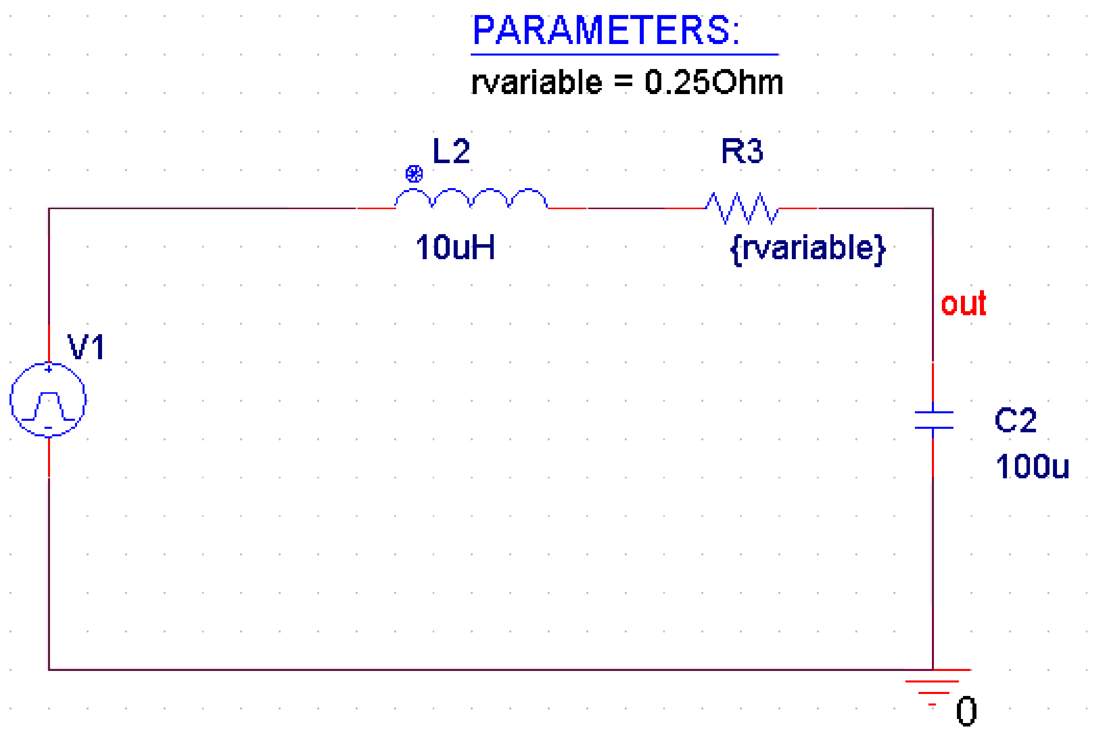


Figure A-1 RLC Circuit

Simulink Block Diagram

Figure A-2 shows the Simulink block diagram (`RLC.slx`).

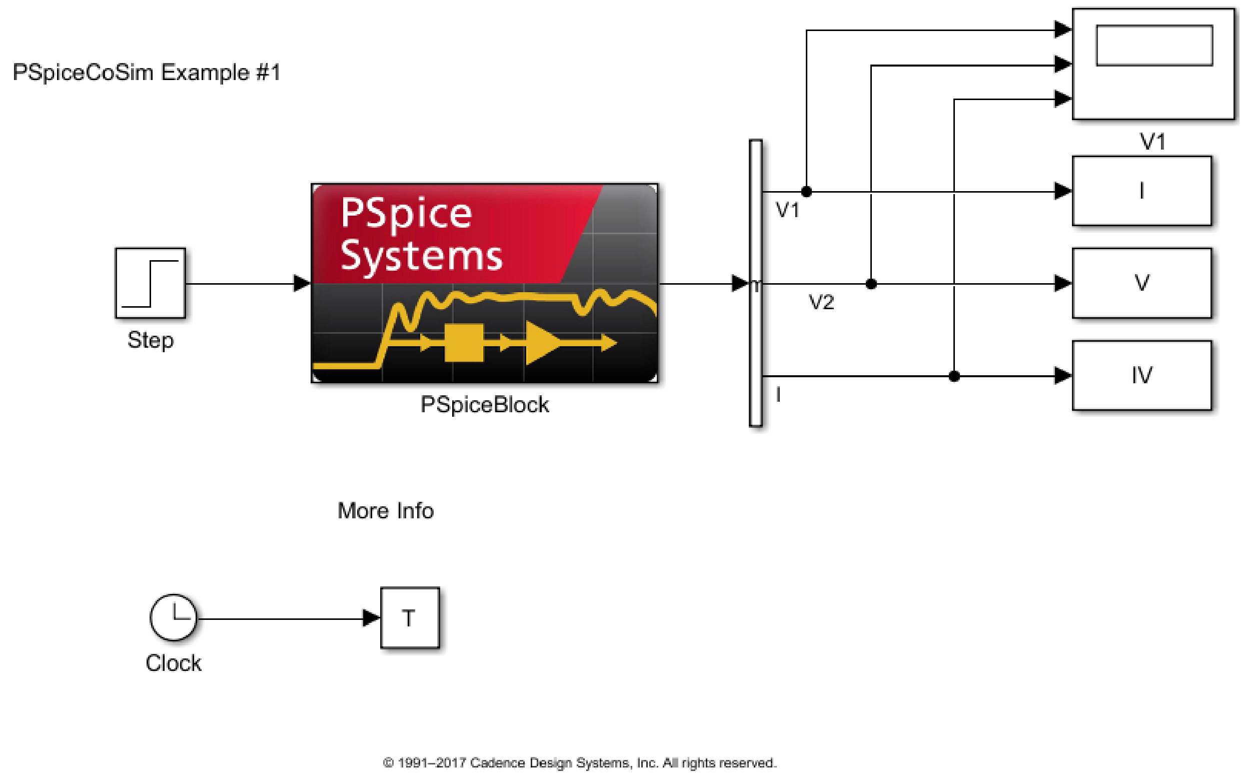


Figure A-2 Simulink block diagram

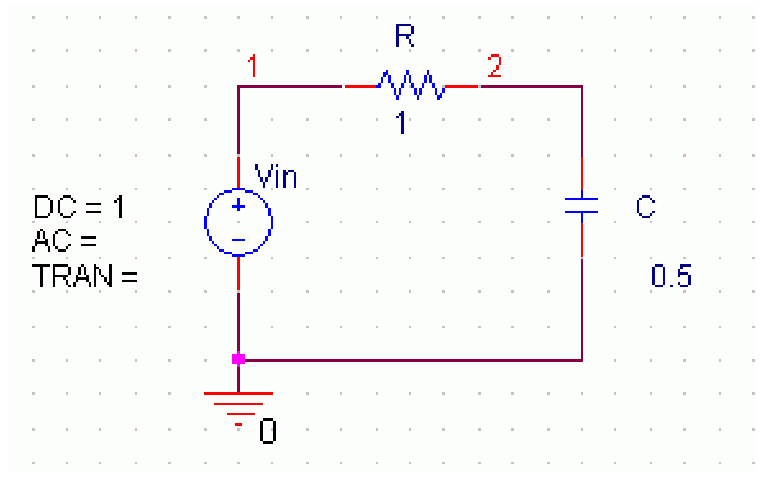
The following table list the various PSpice block parameters.

Parameter	Description
Capture project file	RLC.opj
PSpice circuit file	tran.cir
Input Source	V1 Input data from Simulink as a voltage value, using the voltage source V1 in the circuit.

Parameter	Description
Outputs	I(C2:1)
	V(OUT)
	W(R3)

RC Circuit

Use a simple RC circuit (RCCIR), as shown in the following figure to create the schematic diagram.



Simulink Block Diagram

The following figure shows the Simulink block diagram, `RCMDL_SLPS.slx`.

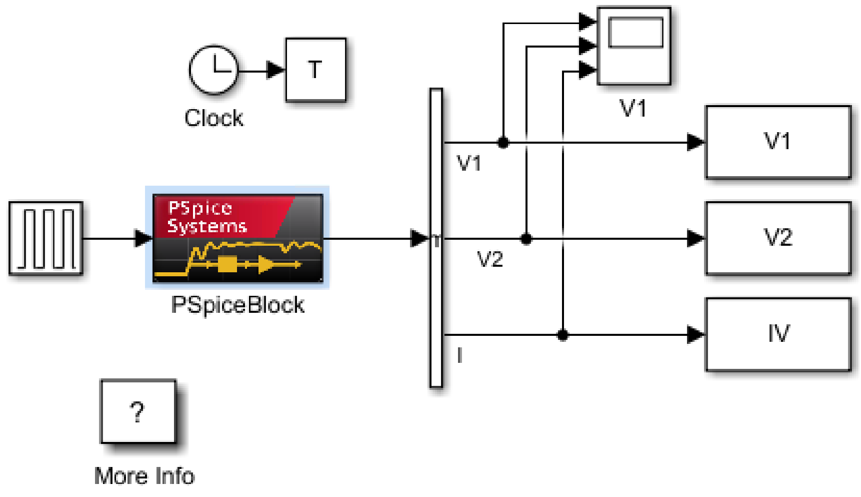


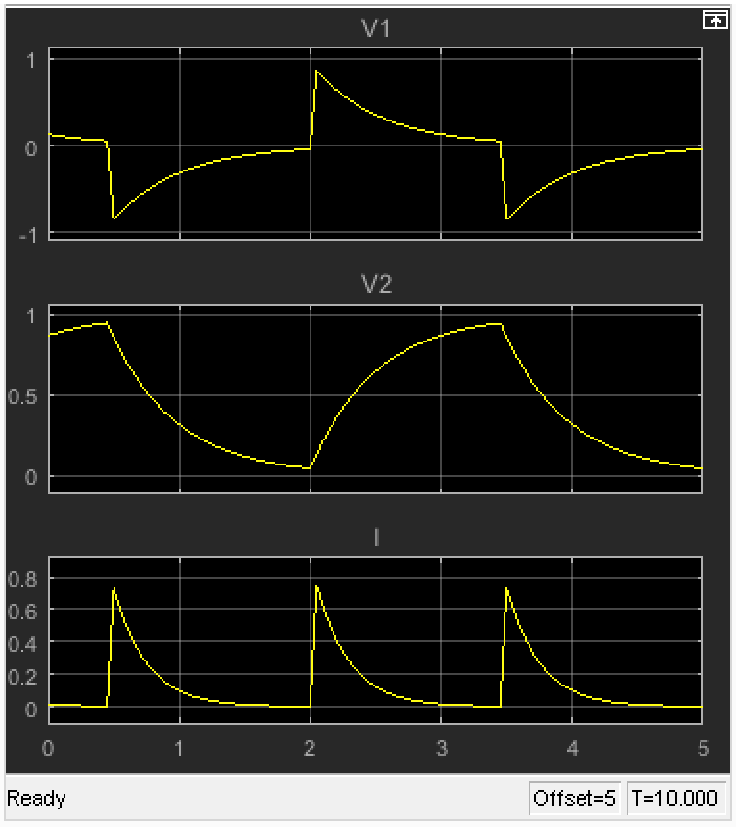
Figure A-3 Simulink block diagram

The following table list the various PSpice block parameters.

`Parameter`	Description
`Capture project file`	RCCIR.opj
`PSpice circuit file`	tran.cir
`Input source`	Vin
	Input data from Simulink as a voltage value, using the voltage source Vin in the circuit.
`Outputs`	I(R:1)
	V(C:2)
	W(R)
	For Simulink, the current value passing through the resistor R, the output voltage at node 2, and the power dissipation through resistor R.
`Data saving option`	None
	In order not to save analyzed data in SLPS and to speed up analysis, select None.
`ITL4 Max`	10
	Convergence errors rarely occur because this is a linear circuit, so set it to the default value of 10 times.

Analysis Settings and Results

In the Simulink analysis settings, analysis is set up with a step size of 0.1 seconds, from 0 to 10 seconds. Following results are seen when you run the simulation from the Simulink window.



Nonlinear Load (NLLOAD)

In this example, you will simulate the state where the values of devices in the circuit are controlled by Simulink and the load connected to the circuit is nonlinear.

Schematic

The electric circuit is roughly divided into two parts, an amplifier and a load connected to the output of the amplifier, as shown in the following figure. The load resistance is determined by Simulink.

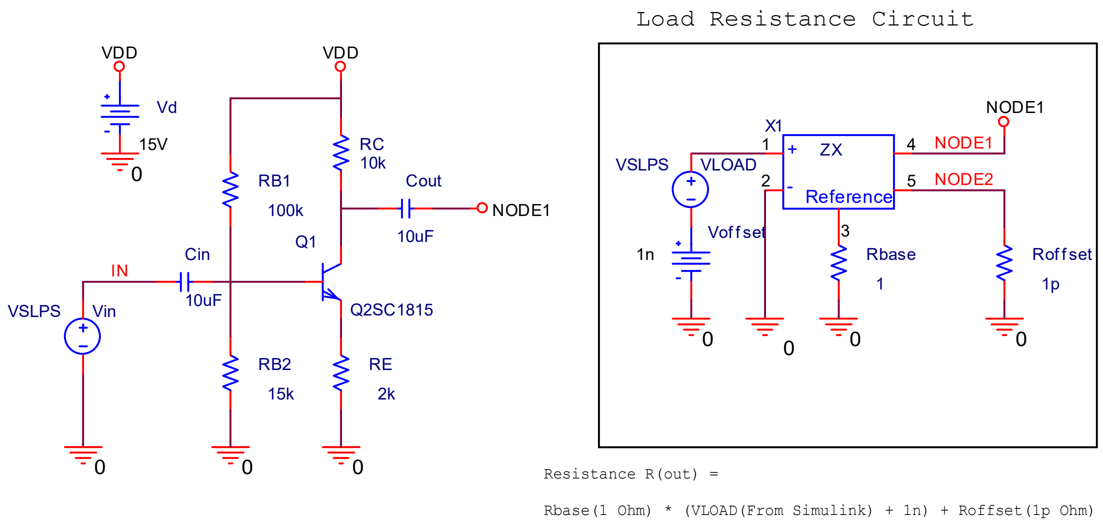


Figure A-4 Nonlinear load circuit

The voltage control impedance ZX/anl_misc.olb is used as the load resistance of the circuit. The output impedance of this device is determined by the following formula:

$$Z_{out} = Z_{ref} \times V$$

where:

- Z_{out}:Impedance between output pins
- Z_{ref}:Reference impedance (connected to Reference pin)
- V:Control voltage

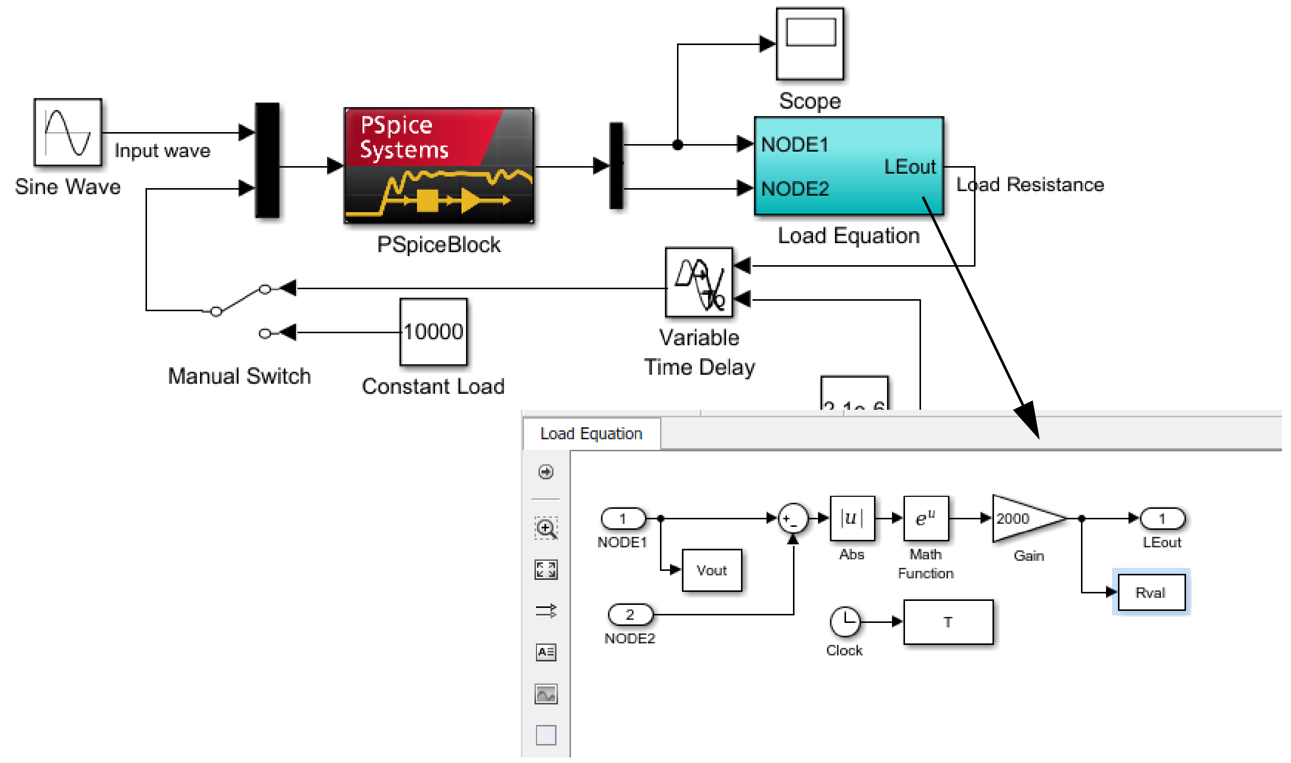
In the circuit, a VSLPS voltage source, which adds a 1nV offset, is connected between the ZX control voltage pins, and a 1Ω reference resistor is inserted between the Reference pin and ground. A 1 $\rho\Omega$ load impedance offset resistor is connected between NODE2 and ground, and NODE1 is connected to the output of the amplifier circuit.

Therefore, the load impedance in respect to the amplifier circuit is given by the following formula:

$$Z_{load} = 1\Omega \times (VSLPS + 1nV) + 1p\Omega$$

Simulink Model

The following figure shows the Simulink block diagram, `NLLOAD_SLPS.slx`.



The voltage values of NODE1 and NODE2 are taken as output from SLPS. The load resistance characteristic is determined by the hierarchical block "Load Equation". The resistance value is returned to SLPS and is determined as follows:

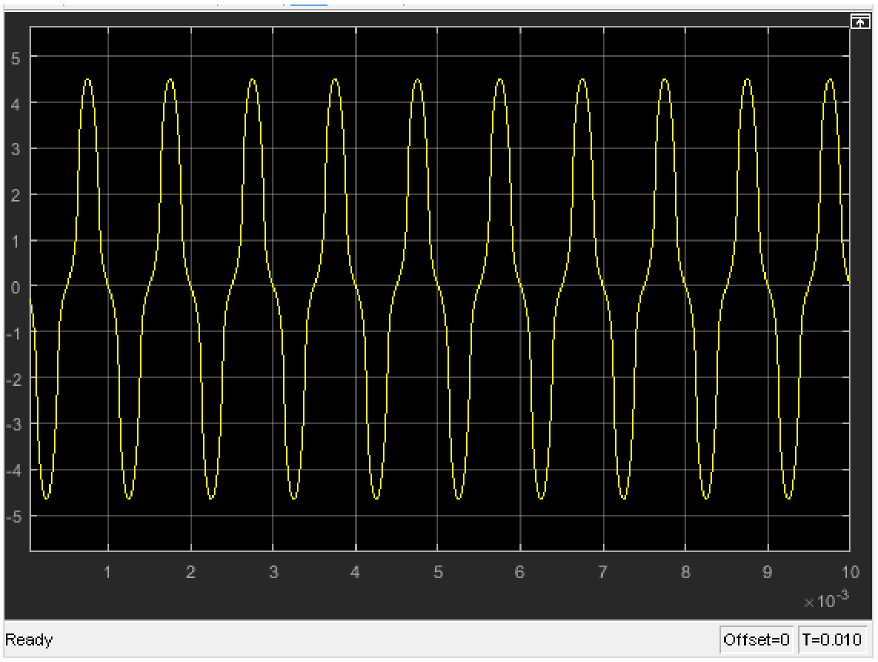
$$LE_{out} = 2000 \times e^{abs(V(NODE1) - V(NODE2))}$$

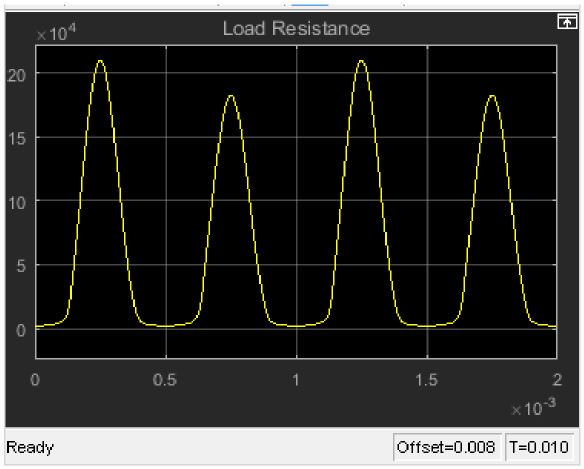
The resistance value LE_{out} is an exponential function of the voltage differential between NODE1 and NODE2.

Note: In this model, when the SLPS block is inserted in the feedback loop, a delay of one cycle step of Simulink occurs between the feedback loop's input (SLPS block's input) and output (SLPS block's output). Therefore, to obtain high-precision results, you must force Simulink to take small time steps.

Results

The following figure shows the output voltage (V (NODE1)) waveform in the upper part and the waveform of the resistance value in the lower part. To plot the resistance value, you need to add another SCOPE at the `Load Resistance` node in the block diagram.





The output voltage vs. resistance characteristic is shown in the following figure and can be plotted by running the `nlloadplt.m` file.

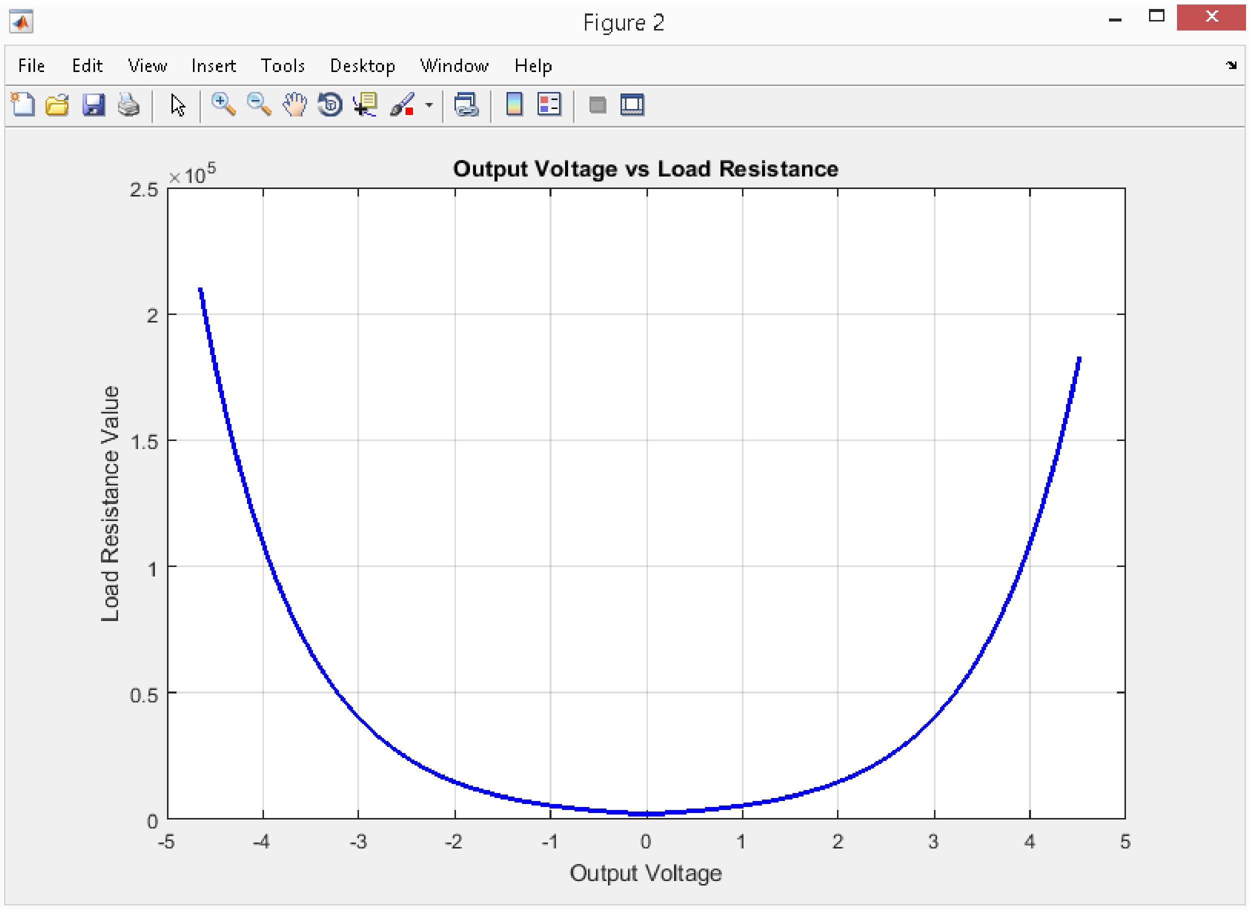


Figure A-5 Output voltage vs. resistance characteristics

You can confirm that the load value changes with time, and, as a result, the output voltage waveform is distorted.

This example shows a nonlinear load, but it can be applied to load fluctuation of a motor or counter EMF.

PLL Model

A Phase Locked Loop (PLL) circuit can halve transmitter output that is synchronized with the input signal and, as a result, expands usability to digital circuit blocks, mobile communications, and so on.

PLL Composition

PLL is comprised of the following three modules:

- Phase comparator
- Loop filter
- VCO

The phase comparator detects the phase difference between two signals, and outputs that difference. As shown in the Figure, A-6 it detects the phase difference between the input signal to the PLL, and the feedback PLL output signal.

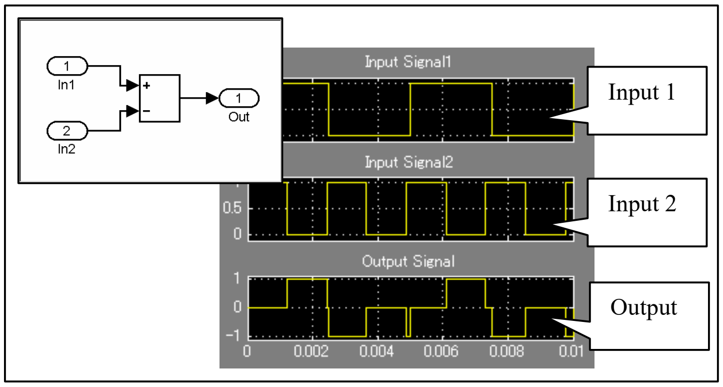
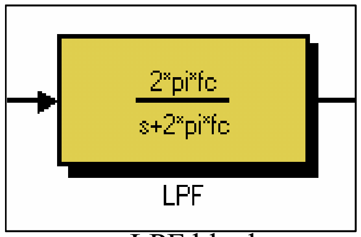


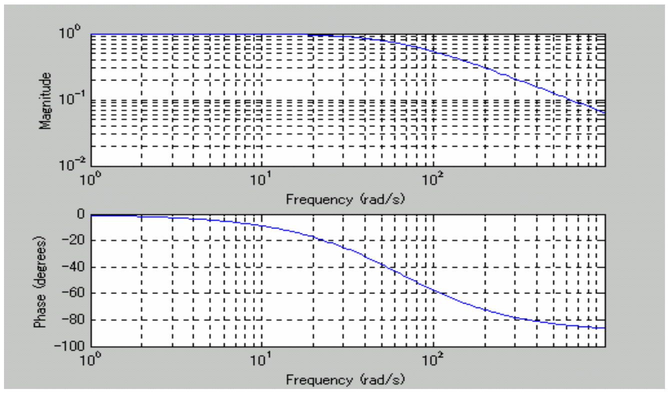
Figure A-6 Phase comparator

The PLL input/output waveform to be designed here is assumed to be a pulse waveform, so it is given by a simple adder, as shown in the figure. In reality, the output value offset is adjusted by the characteristics of the loop filter and VCO being used.

The loop filter eliminates AC components from the phase comparator output and creates a VCO control signal for the next stage. A loop filter is an Low Pass Filter (LPF), and a transmission function block is used in Simulink. In Figure A-7, the frequency characteristic, when fc=10Hz, can be checked using the freqs function of MATLAB.



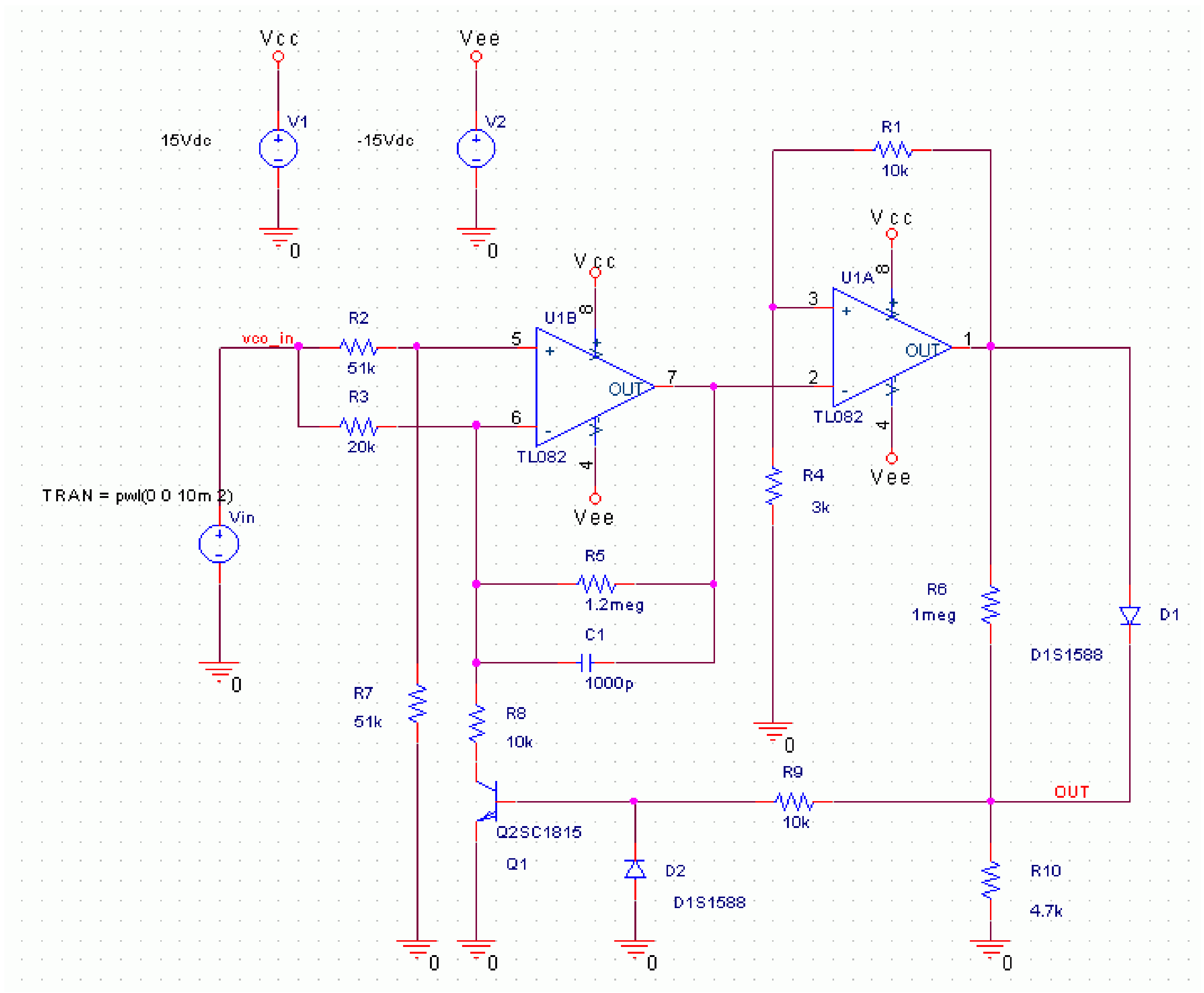
LPF block



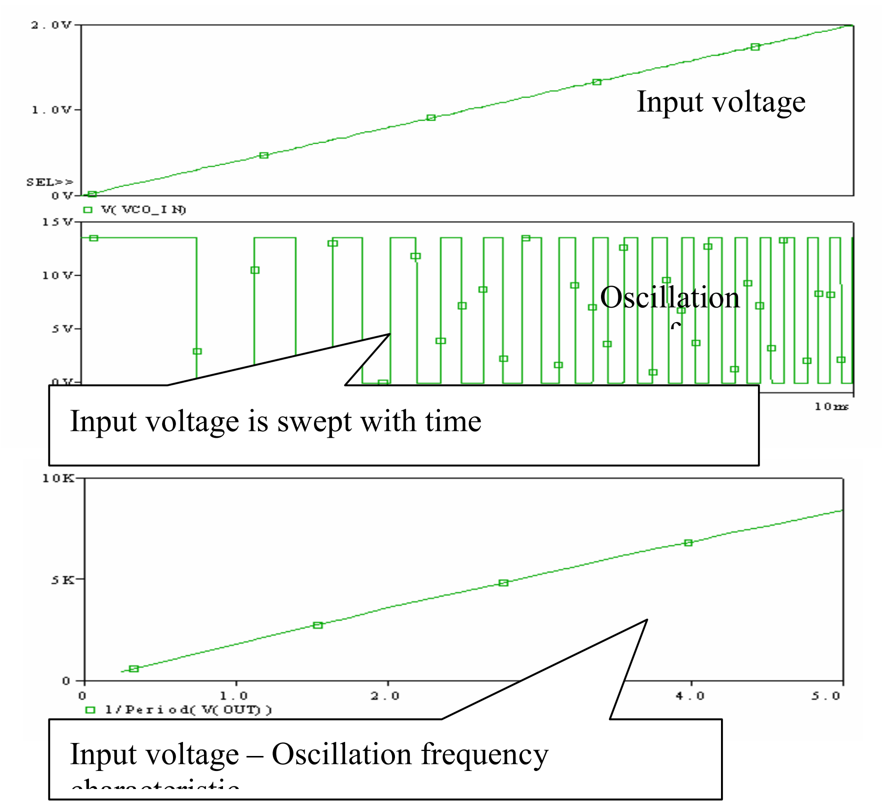
Results for freqs (2*pi*10, [1,

Figure A-7 Frequency characteristic at fc=10Hz

A Voltage Controlled Oscillator (VCO) is an oscillating circuit where oscillation frequency varies with control voltage. In the end, the output of this VCO becomes the output of the PLL. Therefore, the VCO control voltage is controlled so that the oscillation frequency is synchronized with the PLL input signal. The VCO electrical circuit described in the Capture project file VCO.opj is shown in the following figure.



The simulation results using PSpice are shown in Figure A-8:



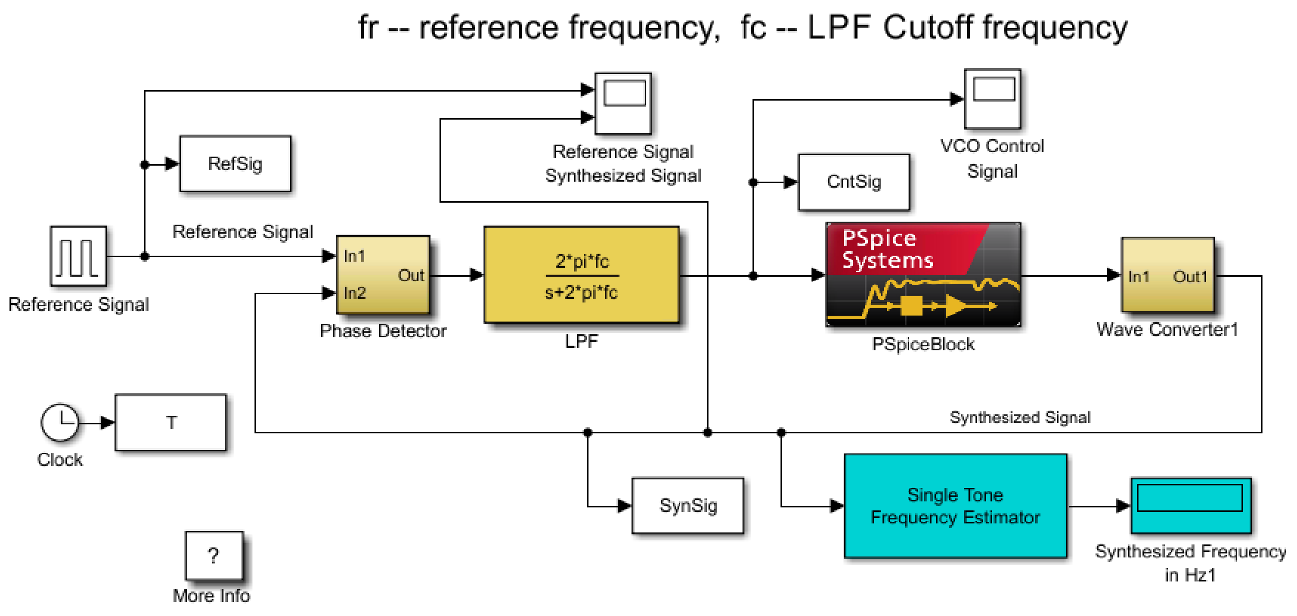
Results of VCO analysis using PSpice

Figure A-8 Simulation result for VCO analysis using PSpice

You can see that the oscillation frequency is about 1.8 KHz at an input voltage of 1V, and it increases almost linearly along with an increase in the input voltage.

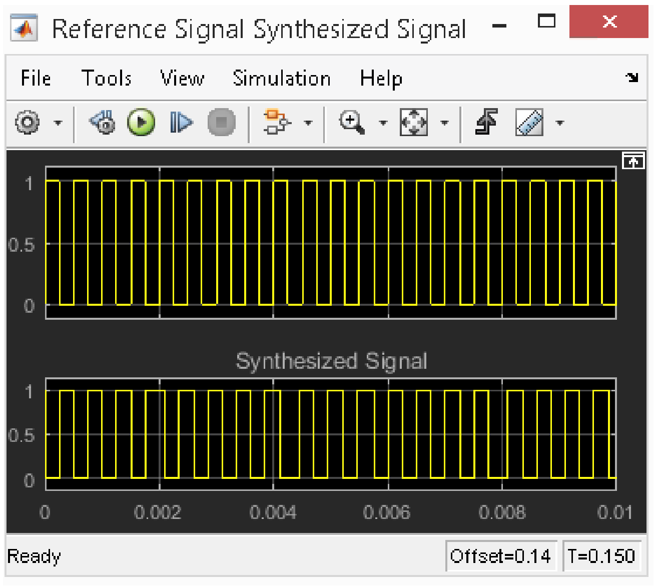
PLL model incorporating VCO electric circuit

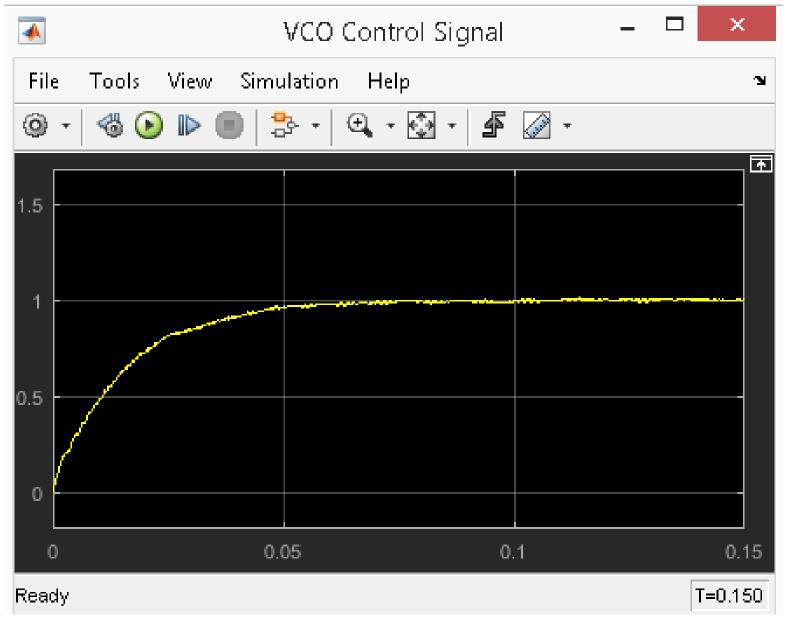
The Simulink PLL model is shown in the following figure.



In this section, an electrical circuit is incorporated by SLPS in VCO only, and the entire PLL operation is checked. If an electrical circuit is incorporated in a loop filter to phase comparator, delays of elements used in the circuit and the effects of the circuit's time constant can also be confirmed.

The simulation results for VCO are shown in the following figures.





From the VCO control signal, it can be confirmed that the oscillation frequency is locked after 70ms.

Switched Reluctance Motor Control

When designing a motor control system, you need to understand the characteristics of the entire system, such as torque determined by motor dimension, the electric circuit including motor inductance, and switching device, and position control associated with rotor rotation.

Analyzing magnetic field using finite element method

With ANSYS, a general-purpose finite element analysis program, you analyze electromagnetic field of the motor form and obtain the inductance value to model a motor.

The switched reluctance motor (SRM) used here as an analysis example employs a 3-phase excitation motor, with 6 stator poles and 4 rotor poles, as shown in Figure A-9.

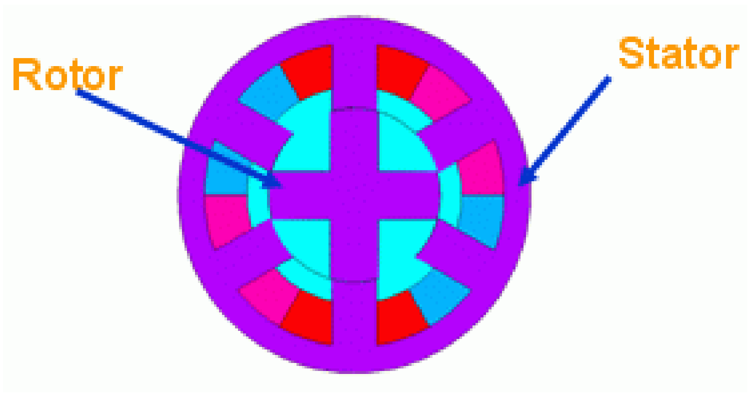


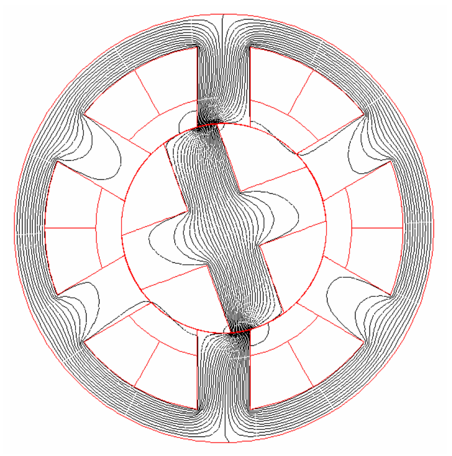
Figure A-9 SRM Model

The SRM can obtain torque using only the magnetic force between rotors and stators. Therefore, if L be the excitation coil inductance, i the current, and W the magnetic energy, the torque T can be expressed using the following formula:

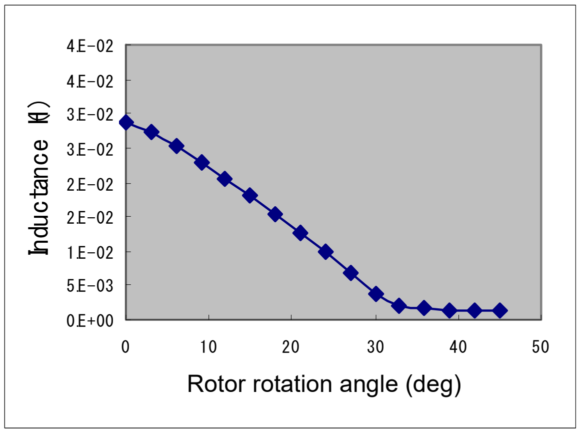
$$T = \frac{dW}{d\theta} = \frac{1}{2} \cdot i^2 \cdot \frac{dL(\theta)}{d\theta}$$

Therefore, torque is derived from changes in inductance. The inductance of the excitation coil to drive the motor varies with the rotation angle of the rotor. You can obtain the inductance value for each angle by performing magnetic field analysis of the motor.

The analysis result and the inductance related to rotor angle is shown by Figure A-10:



Analysis results (flux lines: θ=25)



Inductance related to rotor angle

Figure A-10 Analysis result and rotor angle

Modeling of electric circuit using PSpice

Figure A-11 shows the electrical circuit model.

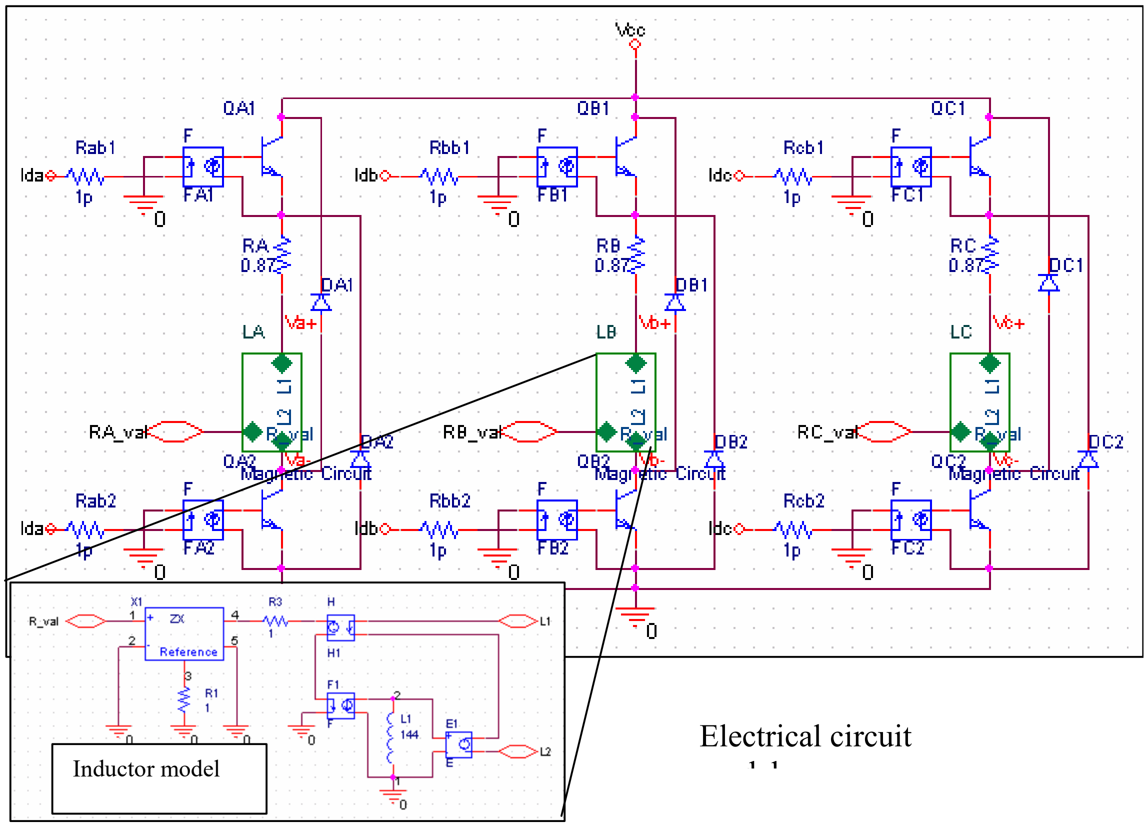
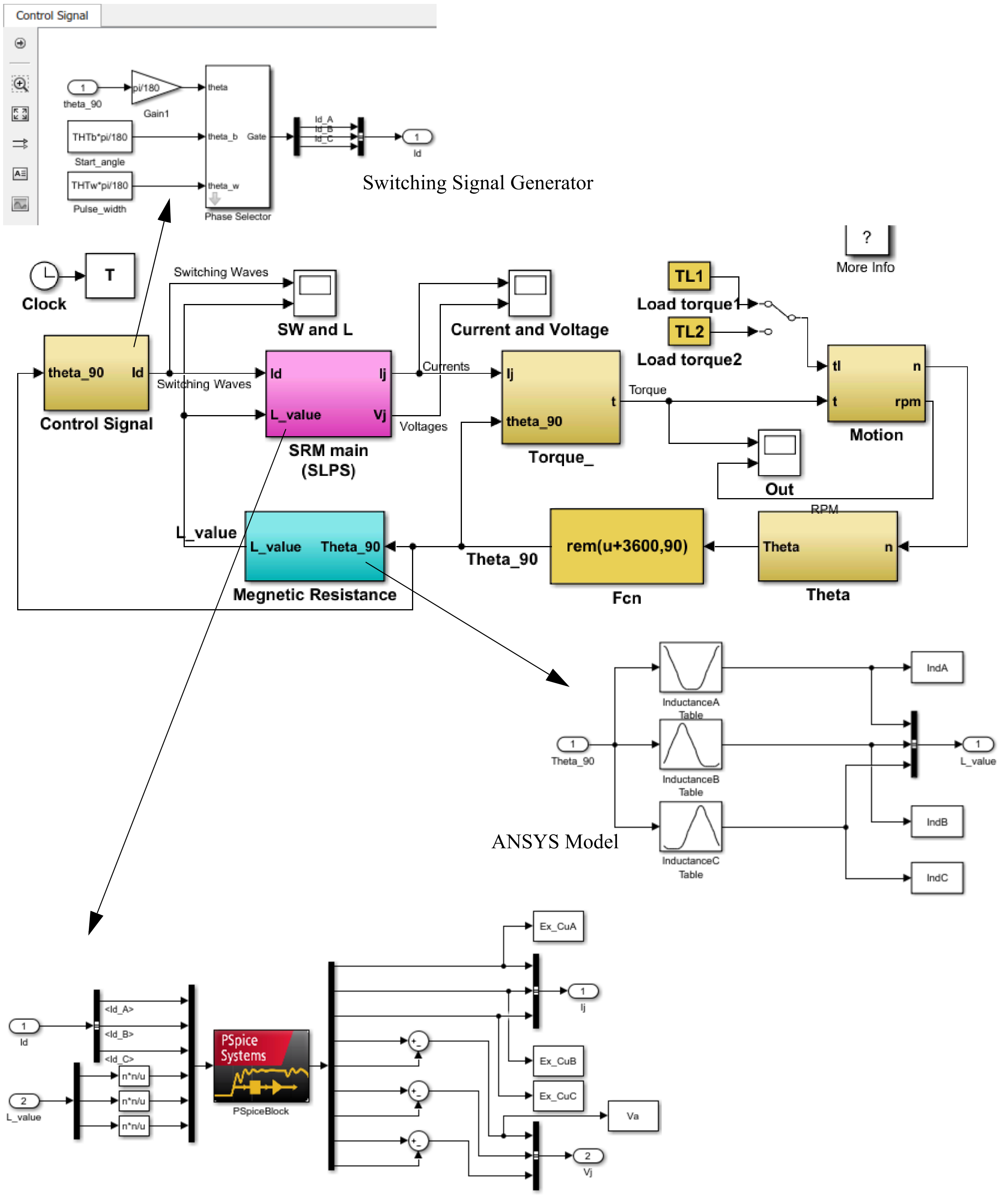


Figure A-11 Electrical circuit model

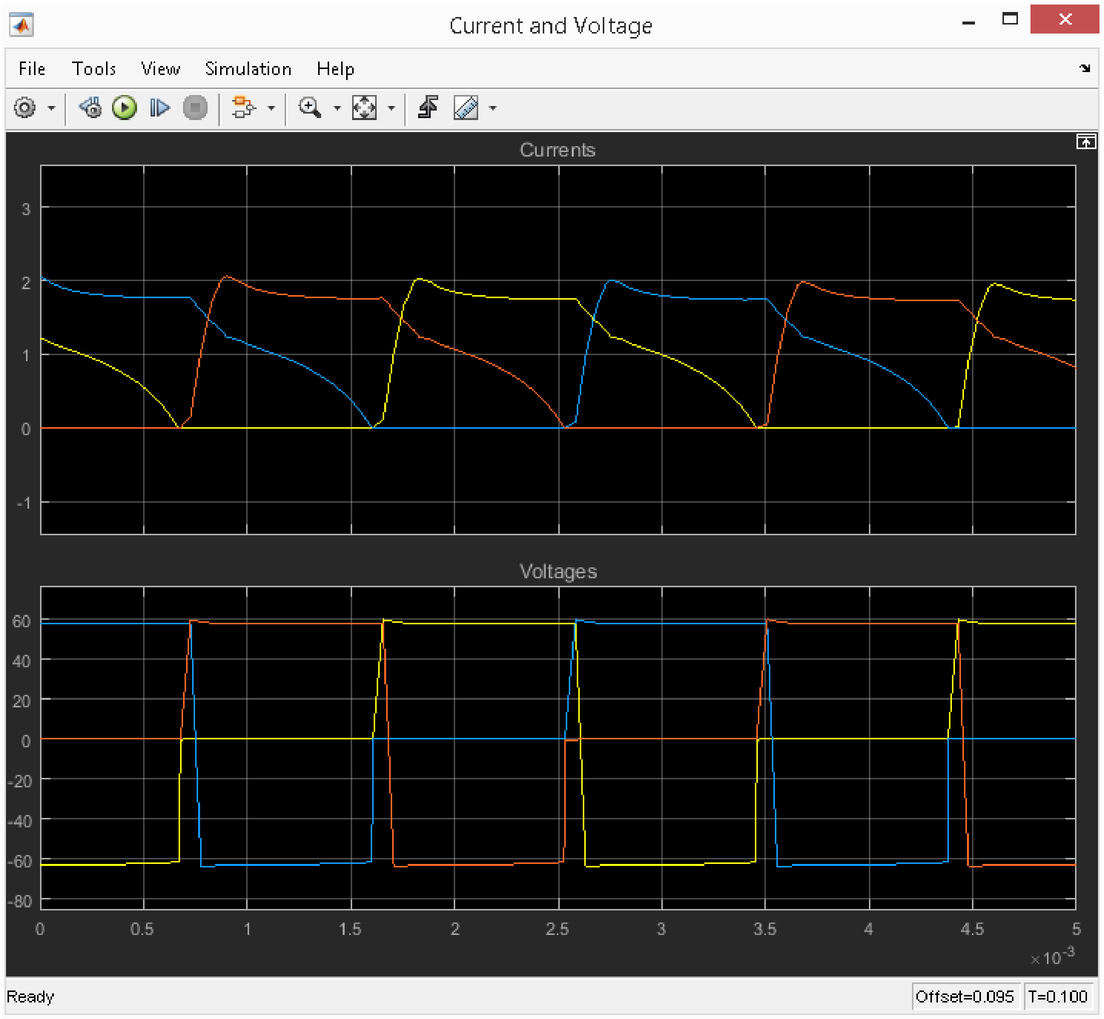
The inductor model placed in the circuit indicates motor inductance components. The inductor value can be determined externally using modeling techniques that separate electrical characteristics and magnetic circuits. During analysis, analysis results from ANSYS are referenced by Simulink, and the inductance value determined by the rotor position is passed as a parameter. The gate signal of the switching transistor is also provided from Simulink, the current value passing through the inductance is monitored, and its value is returned to the outside.

Simulink model

The entire motor control model created with Simulink is shown in the following figure.

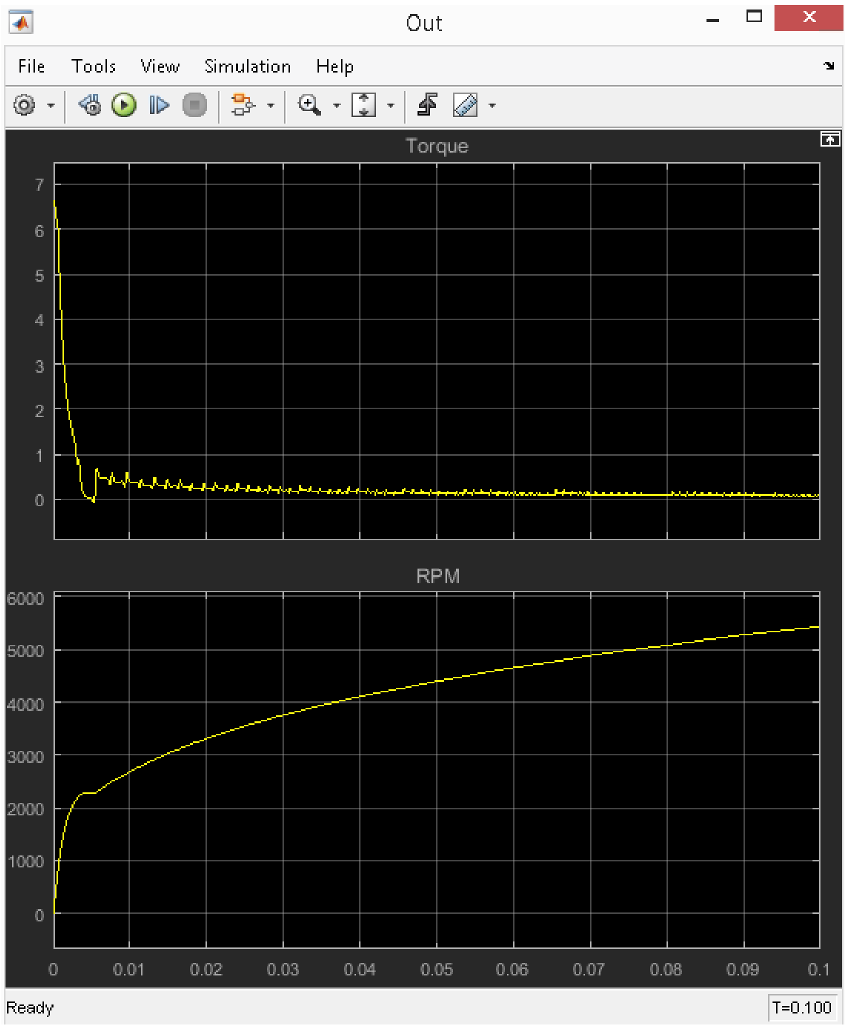

Control Signal
theta_90
pi/180
Gain1
THTb*pi/180
Start_angle
THTw*pi/180
Pulse_width
theta
theta_b
Gate
theta_w
Phase Selector
Id_A
Id_B
Id_C
Id
Switching Signal Generator
?
More Info
Clock
T
Switching Waves
SW and L
Current and Voltage
TL1
Load torque1
TL2
Load torque2
theta_90
Id
Control Signal
Switching Waves
Id
Ij
L_value
Vj
SRM main
(SLPS)
Currents
Voltages
Ij
theta_90
t
Torque_
Torque
tl
n
t
rpm
Motion
Out
RPM
L_value
L_value
Theta_90
Megnetic Resistance
Theta_90
rem(u+3600,90)
Fcn
Theta
n
Theta
InductanceA
Table
InductanceB
Table
InductanceC
Table
IndA
IndB
IndC
Theta_90
L_value
ANSYS Model
Id
L_value
<Id_A>
<Id_B>
<Id_C>
n*n/u
n*n/u
n*n/u
PSpice
Systems
PSpiceBlock
Ex_CuA
Ij
Ex_CuB
Ex_CuC
Va
Vj

The results for current and voltage values for each phase is shown in the following figure.



I

The torque and rotation speed results are shown in the following figure.



The number of rotations varies with load torque, motor dimensions, switching device and switching waveform. These effects can also be simulated.

Switching Power Supply

Power electronics circuits, such as switching power supplies and motor control, might have both electric circuits that switch a large current and high voltage in a short time using a power device, and control circuits with ICs. If you attempt to model an entire circuit of this type using an electric circuit simulator such as SPICE, you must express all of the control parts with the electric device level. In this case, the entire circuit becomes complex, so you may have to spend a lot of time not for verification but for other tasks at the initial stage of design. As the number of devices increases, convergence errors are likely to occur. As a result, it may take longer to analyze data or, in the worst-case scenario, analysis might stop before it is complete.

On the other hand, if the entire circuit is simulated with the system simulator MATLAB/ Simulink, it is easier to express the control algorithm in numerical formulas, but it is difficult to accurately express the rise or fall characteristics of the switching device, and the electrical characteristics of inductors and transformers because it does not have a library of devices.

Electric circuit design using PSpice

Figure A-12 shows the circuit diagram with the forward converter inserted:

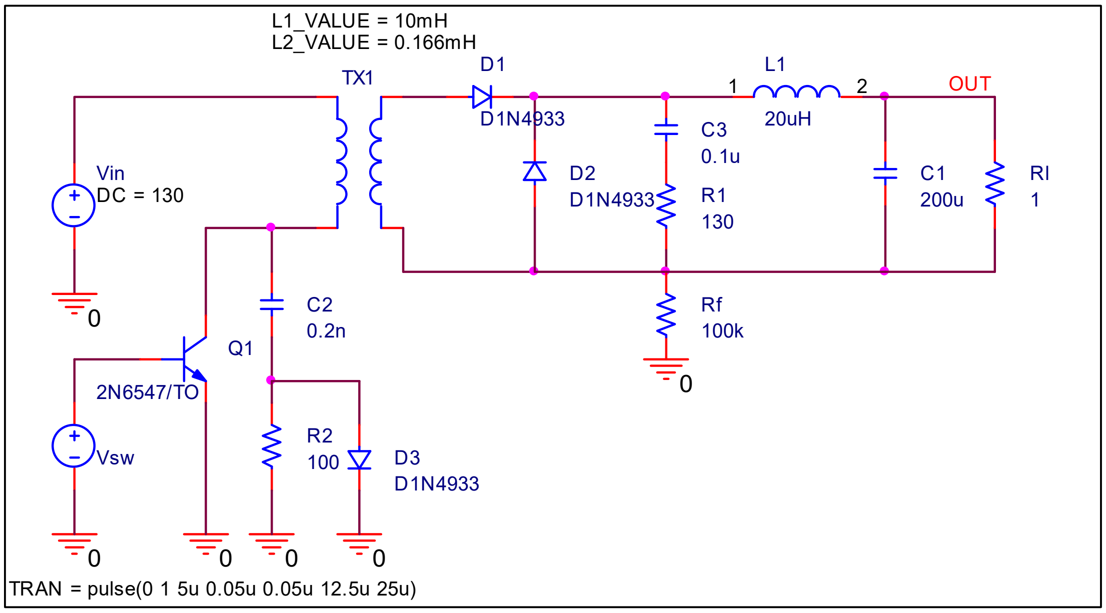


Figure A-12 Circuit with forward converter

The conditions for this circuit are as follows:

- Rated input voltage:130V
- Output voltage:10V
- Output current: 10A

Here, find the optimal voltage of the snubber circuit's capacitor connected to Q1 in order to minimize power loss due to switching.

Figure A-13 shows the VCE waveform of Q1 when the value of C2 is varied from 0.1n to 1.0n using the PSpice function of parametric analysis.

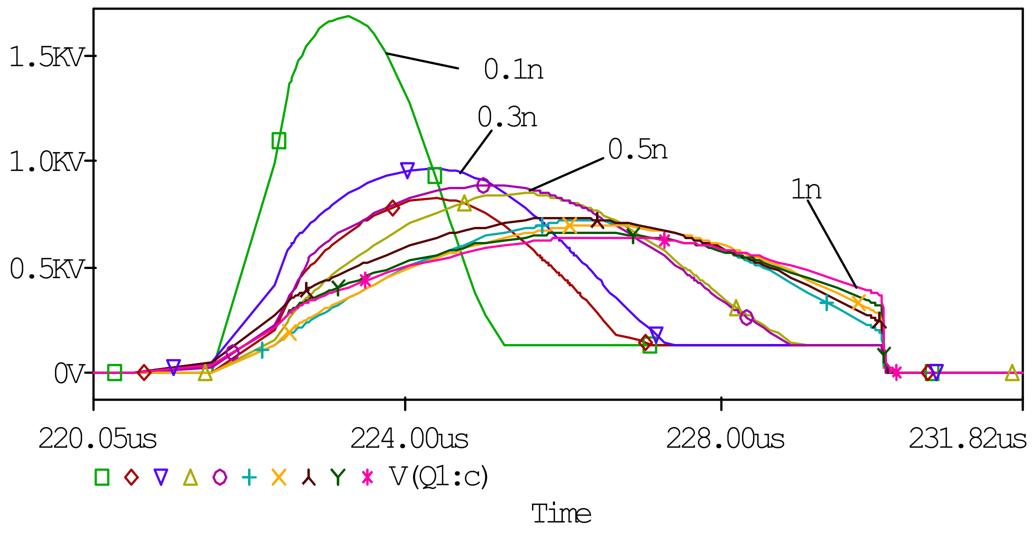


Figure A-13 VCE waveform of Q1

From these results, you can see that the optimal value is approximately 0.5nF where power loss is limited and VCE takes the smallest value during switching.

When 0.5nF is applied to C2, the output current, voltage and VCE waveforms, as shown in the Figure A-14, appear:

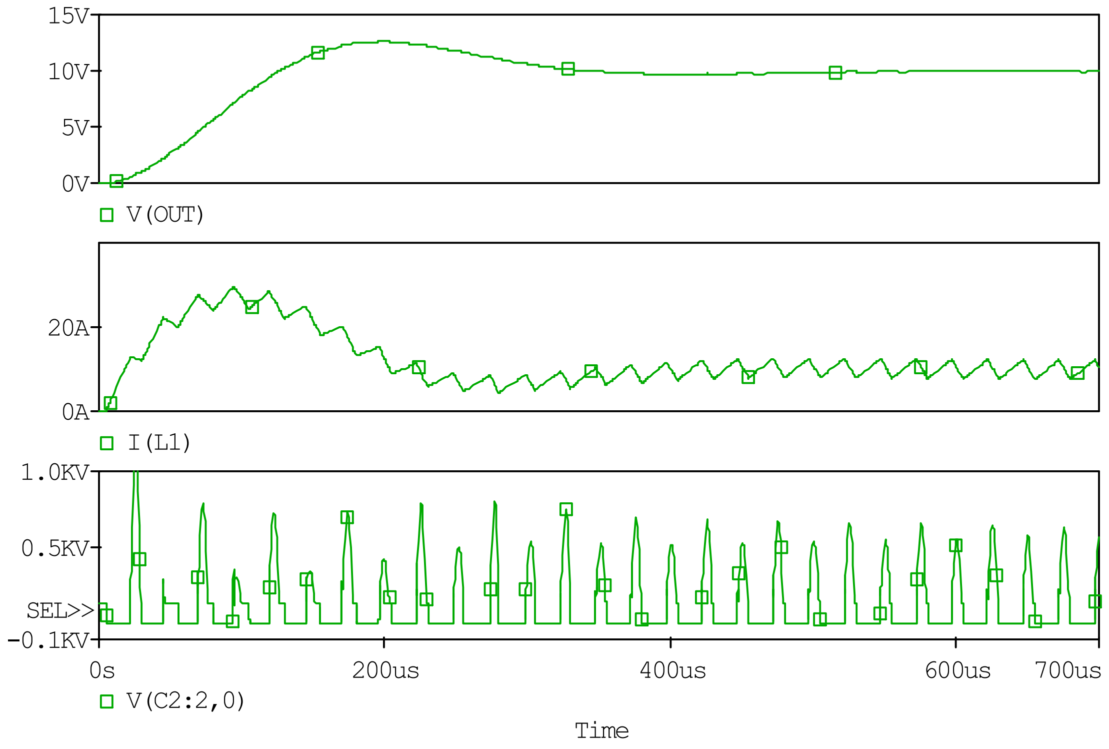


Figure A-14 Voltage and VCE waveform

To specify load from Simulink, use a variable impedance ZX instead of the load resistor R1, thereby making it possible to determine the resistance value using Vload. The circuit is shown in Figure A-15.

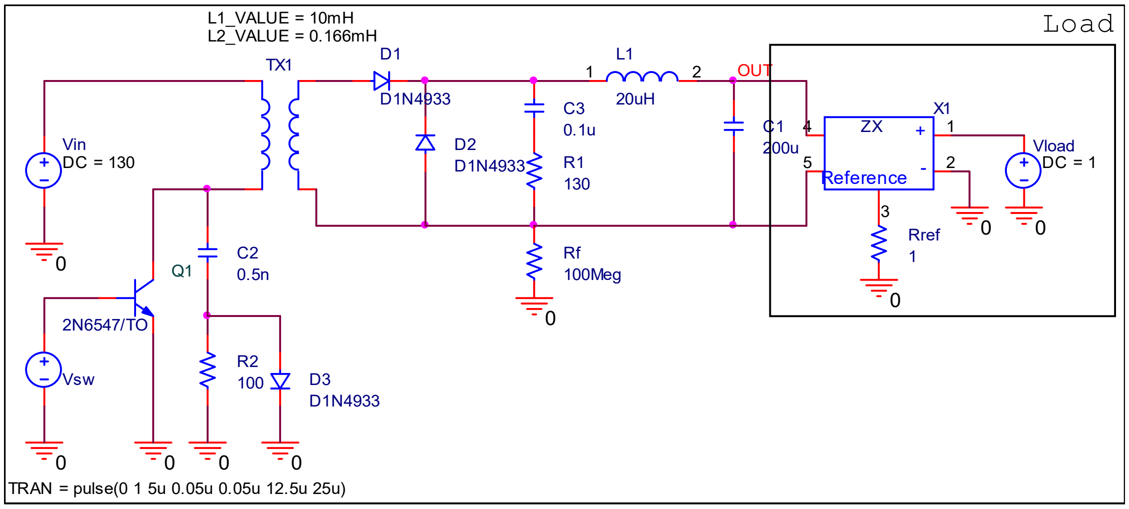


Figure A-15 Circuit using variable impedance

Control system design using Simulink

Using MATLAB/Simulink, design a PWM controller by placing an SLPS block as shown in Figure A-16.

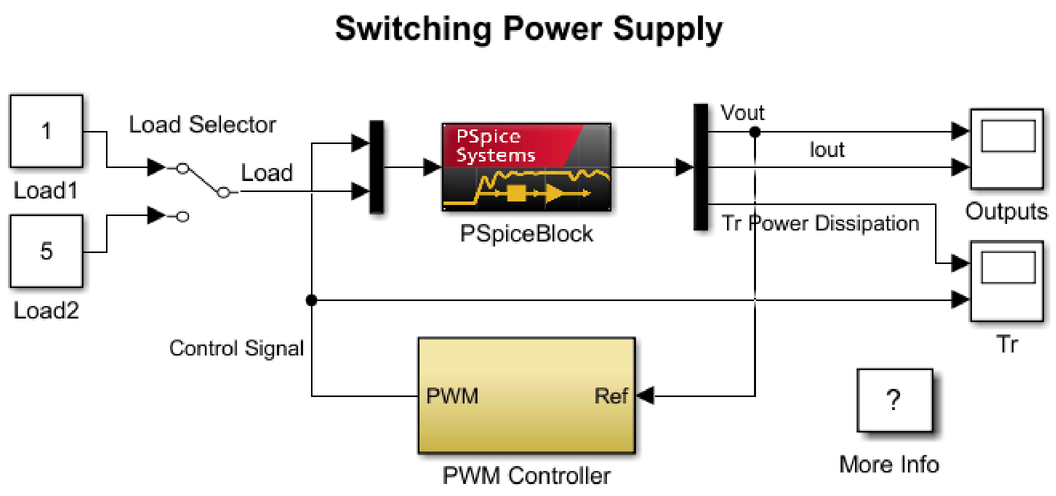


Figure A-16 PWM Controller design

Assign the previous circuit diagram to the SLPS block, designate the output voltage V_{out} and output current I_{out}, and the power dissipation of the switched transistor T_r Power Dissipation. In the PWL control model, the PI controlled output voltage is compared with the reference value Reference, and then with a triangle wave, to create a PWM signal.

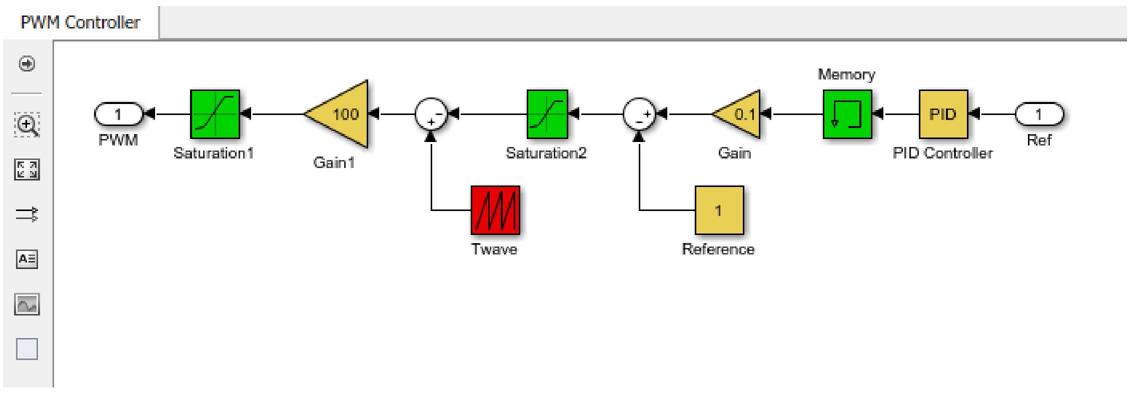


Figure A-17 PWL Controller

In the PWL control model, the PI controlled output voltage is compared with the reference value, Reference, and then with a triangle wave to create a PWM signal.

Results

Set up analyses in Simulink is as follows:

- Stop time: 1e-3 sec
- Solver type: Fixed step
- Solver: ode5
- Step size: 1e-6 sec

Figure A-18, Figure A-19, and Figure A-20 show simulation results when the load resistance is set to 1Ω.

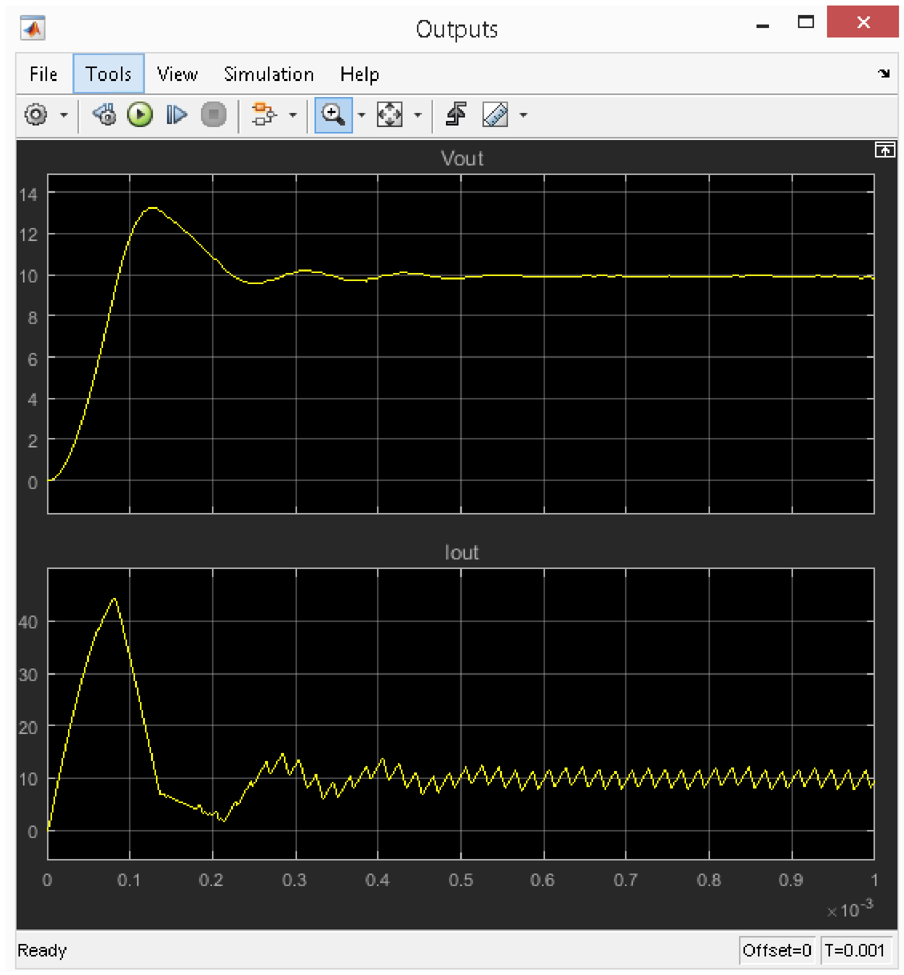


Figure A-18 Output voltage and current

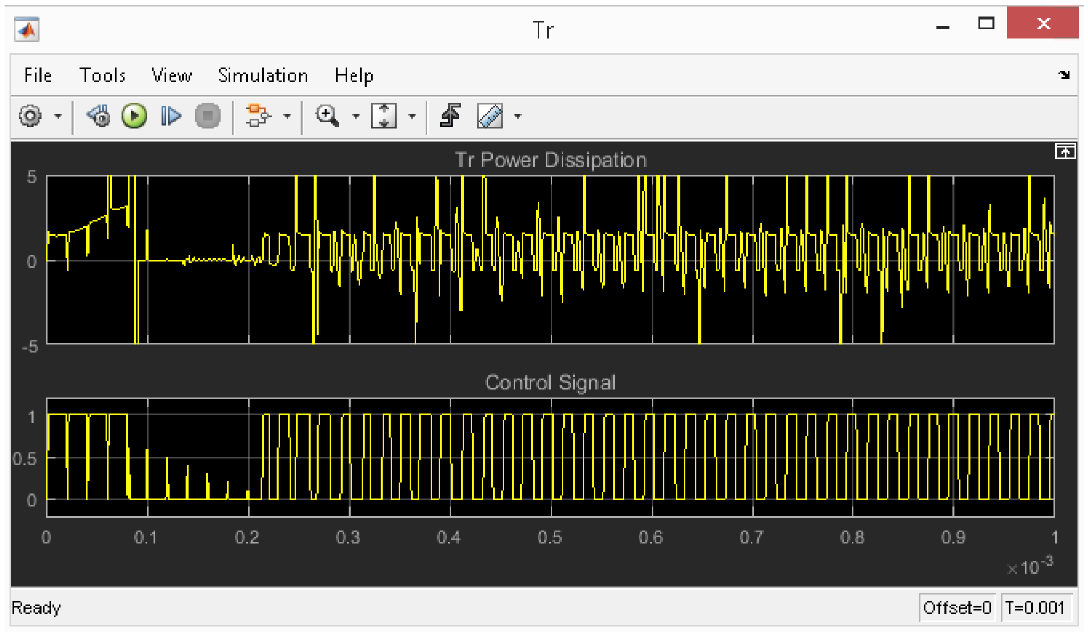


Figure A-19 Transistor power dissipation and switching signal waveform

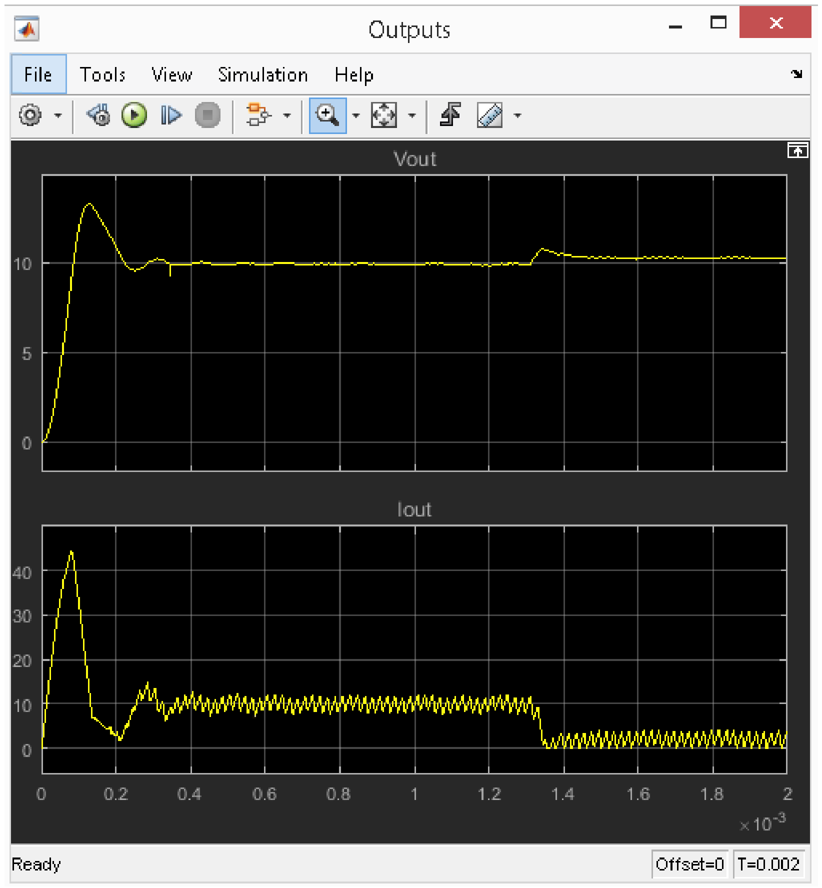


Figure A-20 Changing load resistance from 1 to 5Ω near 1.3mS

From the results, you can see that output stabilizes at about 0.4mS after starting the simulation, and it changes along with the fluctuations in load.

In this design example, PSpice has been used for electrical circuits, and the Simulink design environment for control. Electrical circuits have been used for the parts where electrical characteristics (like element delay) have an effect and those which are difficult to express in formulas. To design the basic circuit, PSpice has been used. To design the control system which drives the circuit, a Simulink model which is flexible in expressing numerical formulas has been used. Finally, both models have been combined using SLPS, with which the entire system has been simulated while taking electrical characteristics into consideration.

DC Motor Control System

Due to the trade-off between simulation speed and result accuracy, it is not an efficient method to use highly accurate models at the first stage of designing, as the simulation speed will be very slow. It is recommended to use proper models for each stage of designing process. You may find that proper modeling makes optimizing the system parameters easier.

Outline of DC motor control system

Figure A-21 shows the diagram of a DC motor control system.

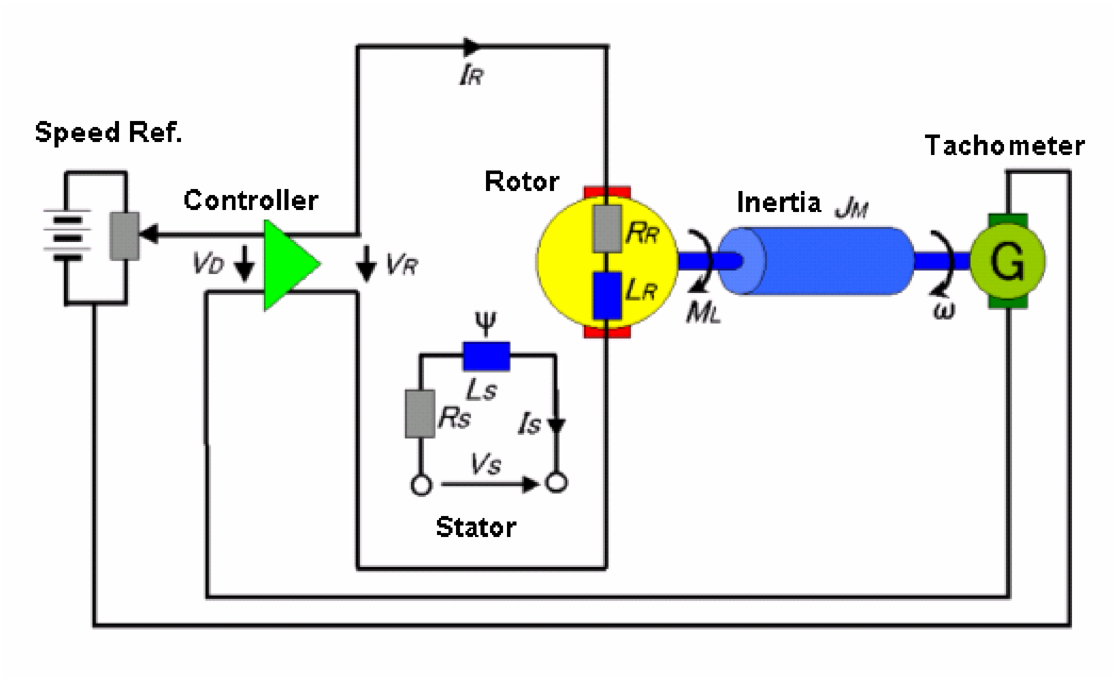


Figure A-21 DC motor control system

This diagram consists of a stator or field winding which produces a constant magnetic flux for excitation. This winding is modeled by inductance *Ls* and resistance *Rs*. When voltage *Vs* is applied to the stator winding, current Is flows which produces flux. The rotor winding is also modeled by its inductance *LR* and resistance *RR*. When voltage VR is applied to the rotor winding, current IR flows. VR is the output of a controller which is driven by a differential input voltage VD. The signal flow is from VD to VR but not in the reverse direction. As a result, the armature reaction between rotor and stator is neglected.

First, write the mesh equation for the stator winding:

$$V_s = R_s \cdot I_s + \frac{d\psi}{dt}$$

In general, the magnetic flux Ψ is a nonlinear function of the stator current I_s. For simplicity, assume ψ to be a linear function of I_s:

$$\psi = L_s \cdot I_s$$

As a result,

$$V_s = R_s \cdot I_s + L_s \cdot \frac{dI_s}{dt}$$

or

$$\frac{dI_s}{dt} = \frac{V_s - (R_s \cdot I_s)}{L_s}$$

For the mesh equation of the rotor winding:

$$V_R = R_R \cdot I_R + L_R \cdot \frac{dR_R}{dt} + V_I$$

or

$$\frac{dI_R}{dt} = \frac{V_R - (R_R \cdot I_R) - V_I}{L_R}$$

For the moments on the rotor shaft, you can write the balance equation:

$$J_M \cdot \frac{d\omega}{dt} = M_R - M_L - M_F$$

where,

$$M_R = C_2 \cdot \psi \cdot I_R$$

and

$$M_F = C_3 \cdot \omega$$

M_R is the rotor driving torque, which is a linear function of excitation flux ψ and rotor current I_R multiplied by constant C_2. M_L is the given load moment and M_F is the internal friction moment of the motor which is assumed to be a linear function of angular velocity ω multiplied by constant C_3. J_M is the moment of inertia of all rotating masses mechanically connected to the rotor.

The controller is driven by the voltage difference V_D between the speed potentiometer voltage, which sets the nominal angular velocity ω_S and the tachometer voltage, which is proportional to angular velocity ω. You can write V_D as the difference between the angular velocities multiplied by constant C_4:

$$V_D = C_4 \cdot (\omega_S - \omega)$$

For the controller, use a PI characteristic which is described by:

$$V_R = C_5 \cdot \int V_D dt + C_6 \cdot V_D$$

Simulink Model

First, model the whole system in Simulink, as shown in figure A-22.

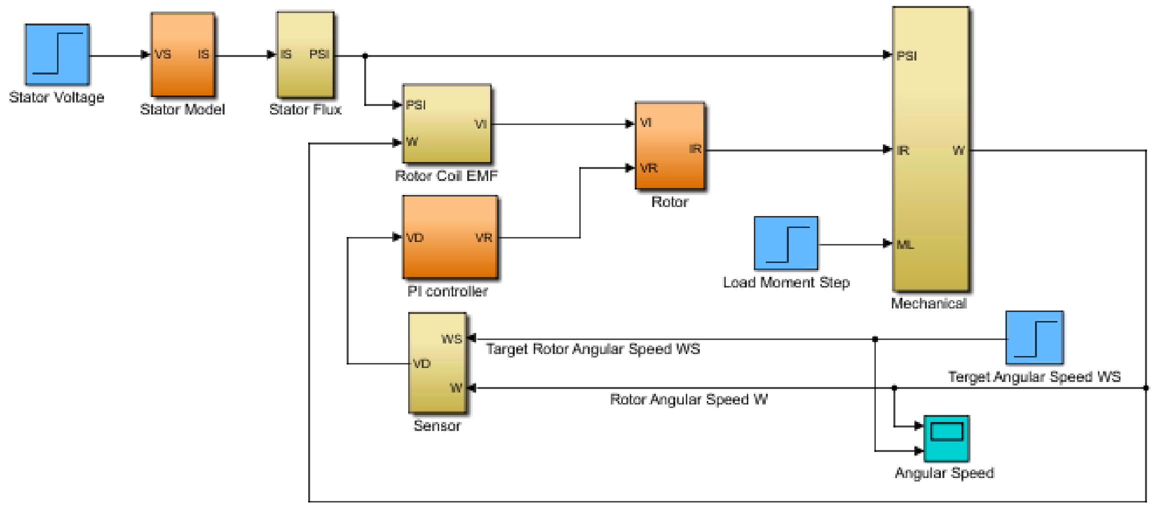


Figure A-22 System model

Specify the parameters:

$C_1 = 0.50$

$C_2 = 0.50$

$C_3 = 0.10$; % N*m*s

$C_4 = 0.01$; % V*s

$C_5 = 606.0$; % 1/s

C_6 = 100.0

L_R = 0.10; % H = V*s/A

L_S = 10.0; % H = V*s/A

R_R = 0.50; % Ohm = V/A

R_S = 500.0; % Ohm = V/A

J_M = 1.0; % $kg \cdot m^2$

Figure A-23 shows the plots for the angular velocities $\omega \cdot S(t)$) and $\omega(t)$.

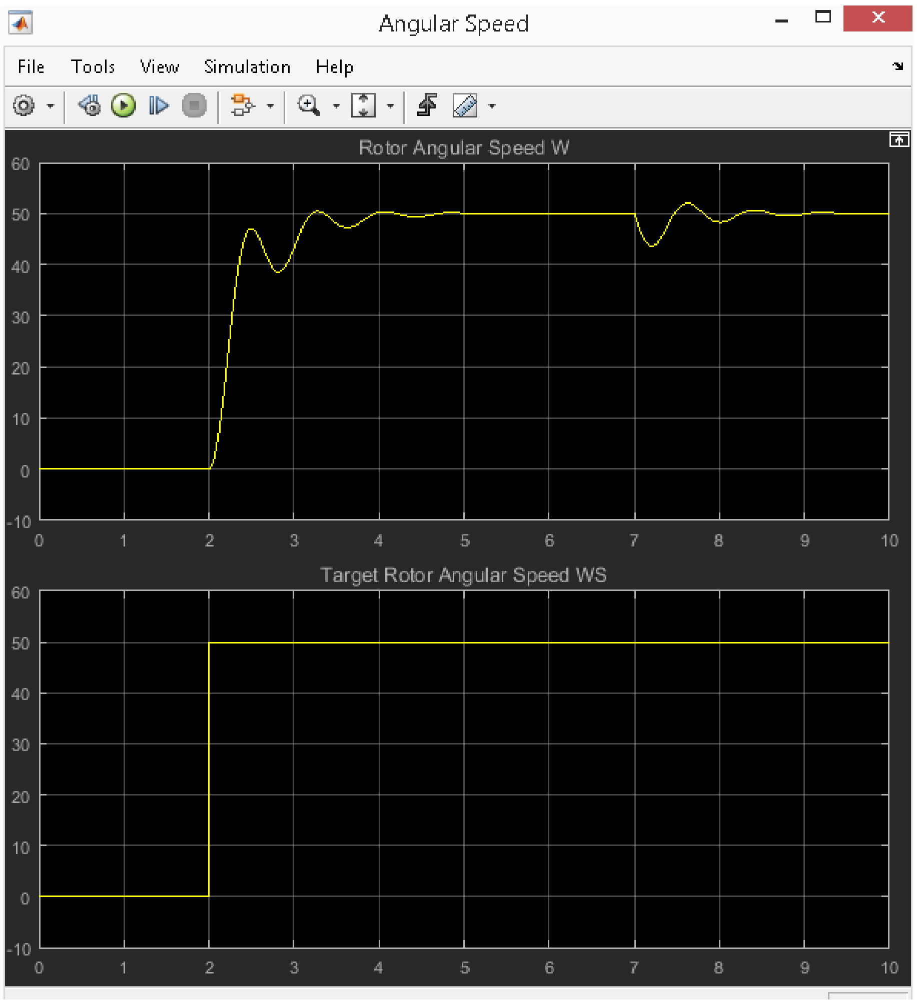


Figure A-23 Plots for the angular velocities $\omega \cdot S(t)$) and $\omega(t)$

The stator voltage V_S is switched on at time t=1s. ωS is set to 50 rad/s at 2s and the rotor starts to turn. At about t=6s, ω reaches the steady state of 50 rad/s.

At t=7s, a load moment of 50 Nm is applied and the rotor speed reduces to 43.6 rad/s. Therefore, the controller increases VR and the new steady state of ωS=50 rad/s is reached again at about t=10s.

Simulink-PSpice Model (Ideal Opamp)

With the results of the Simulink model available for reference, you can now replace some of the Simulink blocks to PSpice circuits. Figure A-24 shows the system block diagram.

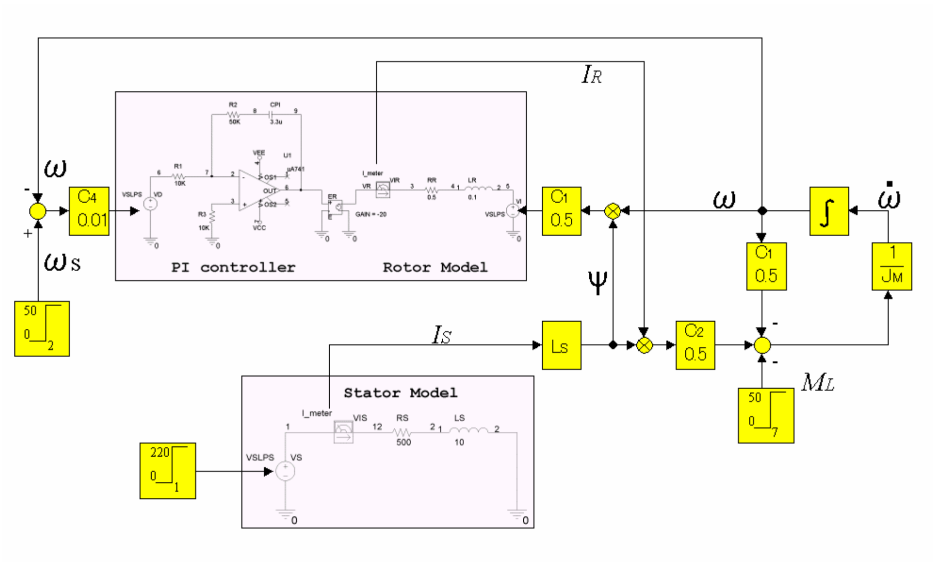


Figure A-24 Ideal Opamp system block diagram

Figure A-25 shows the Simulink model, which includes PSpice circuits using SLPS.

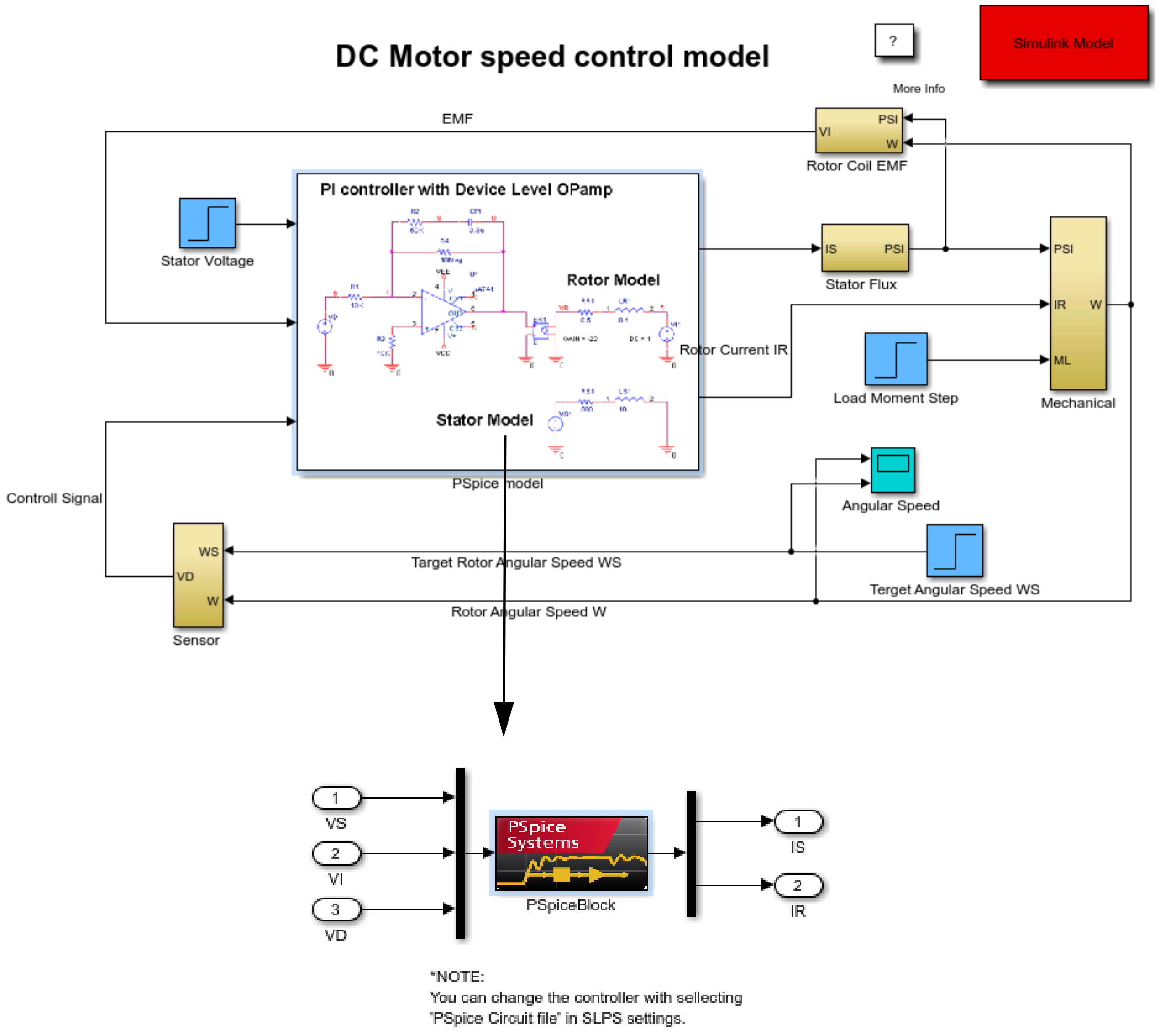


Figure A-25 Simulink model including PSpice circuits

Figure A-26 shows the stator circuit.
f

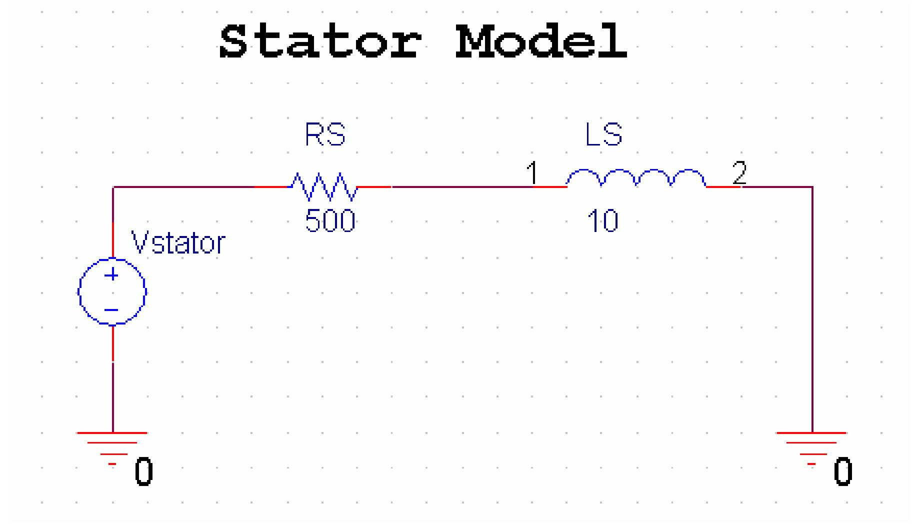


Figure A-26 Stator model

Figure A-27 shows the rotor circuit:

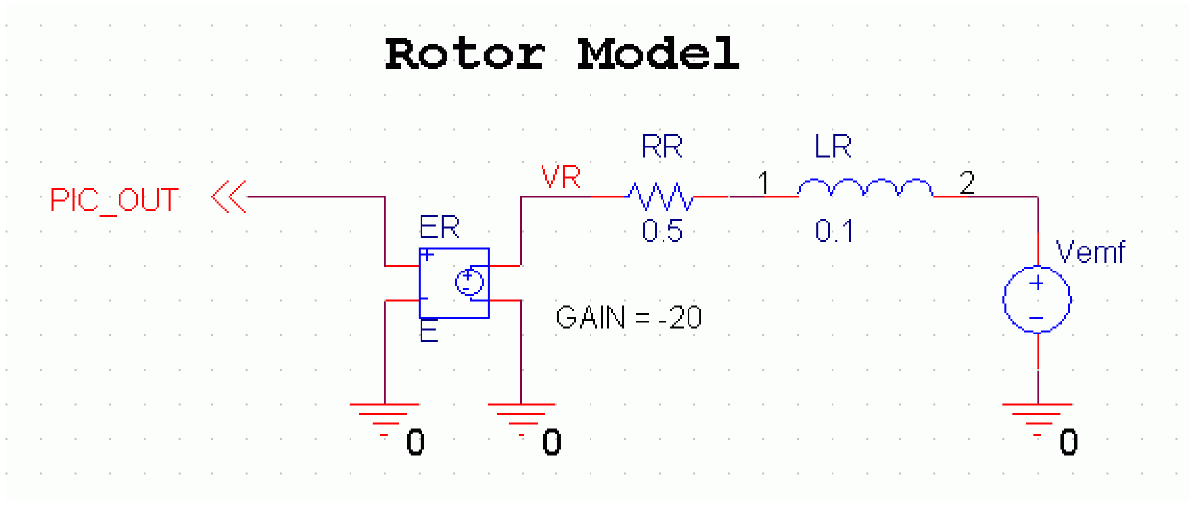


Figure A-27 Rotor model

Figure A-28 shows the PI controller circuit:

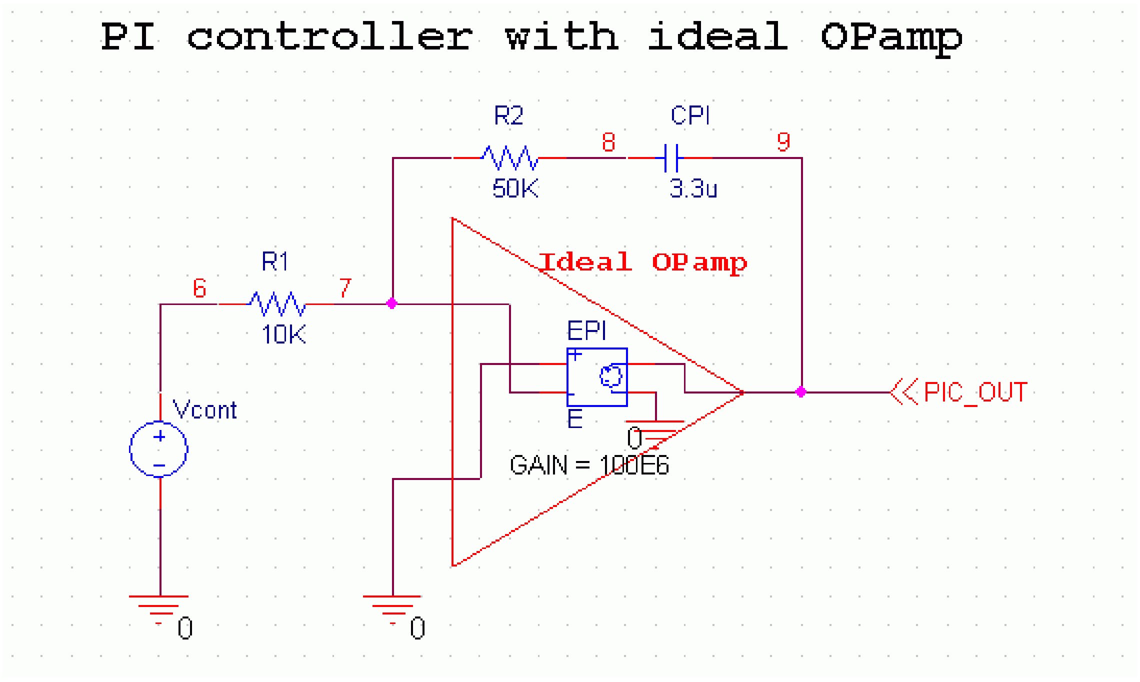


Figure A-28 PI controller circuit

Circuit elements V_S, R_S, L_S and E_R, R_R, L_R implement the equation for $\frac{dI_S}{dt}$ and $\frac{dI_R}{dt}$. For the operational amplifier(Opamp) in the PI controller, first use an ideal model with very high open-loop gain, infinite input resistance, zero output resistance, zero input offset voltage, zero input offset and bias current, and no output voltage saturation. This behavior is easily modeled in SPICE by a linear voltage-controlled voltage source. The PI controller output voltage V(9)(PIC_OUT) is given by the equation.

$$V(9) = \left(\frac{1}{R_1 C}\int V_D dt + \frac{R_2}{R_1}V_D\right)$$

with the factors,

$$\frac{1}{R_1 C} = \frac{1}{10k\Omega \cdot 3 \cdot 3.3\mu F} = 30.3$$

and

$$\frac{R_2}{R_1} = \frac{50k\Omega}{10k\Omega} = 5$$

The corner frequency f_0 where the I and P characteristics meet is:

$$f_o = \frac{1}{2\pi \cdot 50k\Omega \cdot 3.3\mu F} = 0.965Hz$$

Voltage $V(9)$ controls the voltage-controlled voltage source E_R, which has a gain value of -20. Therefore, you obtain:

$$V(3) = -20 \cdot V(9) = 606\int V_D dt + 100V_D$$

Comparing this equation to the one for V_R in the pure Simulink model, you will observe that both are identical because of constant values C_5=606 and C_6=100. The PSpice controller circuit, therefore, implements the same equation V_R for as in subsystem PI-CONTROLLER in Simulink model.

Simulink-PSpice Model Result (Ideal Opamp)

Figure A-29 shows the simulation results, $\omega(t)$ and $\omega S(t)$) for PI controller with ideal Opamp:

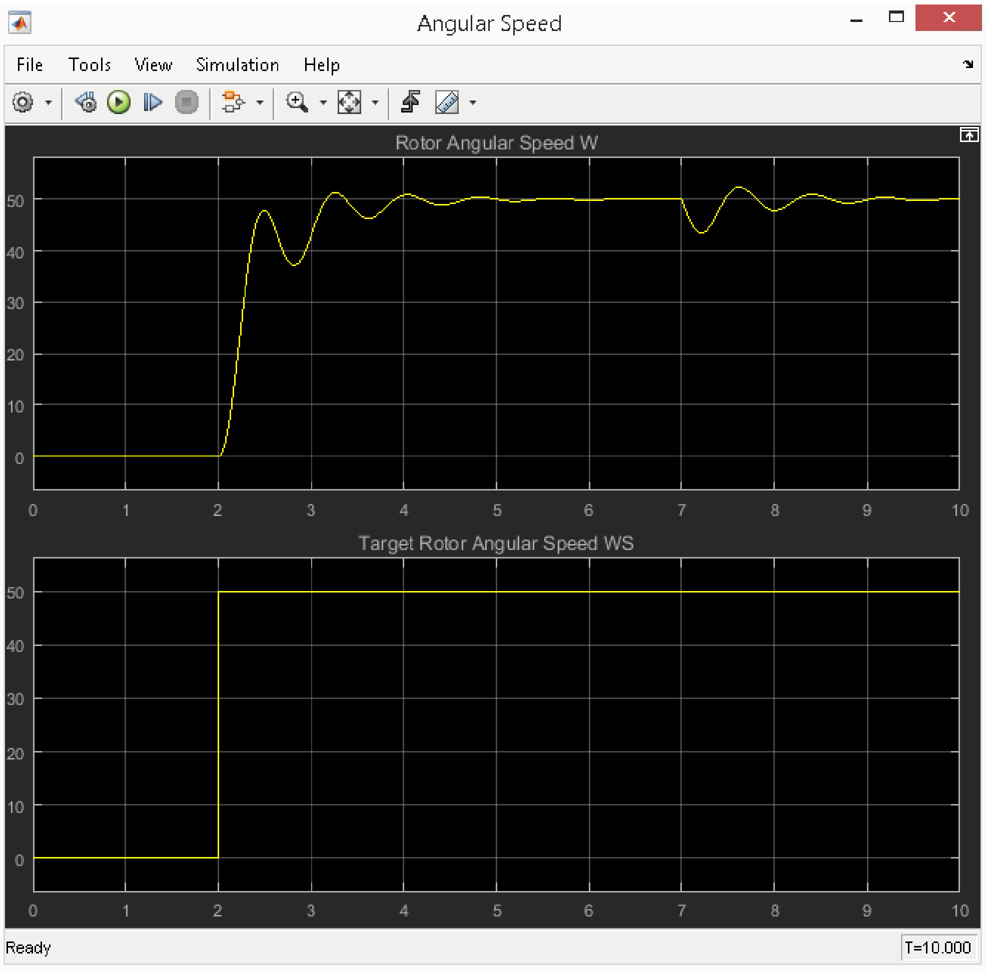


Figure A-29 Simulation results for PI controller with ideal Opamp

Simulink-PSpice Model (Device Opamp)

You can now replace the ideal Opamp model in the PI controller with a device-level model of a standard µA741 Opamp, as shown in Figure A-30:

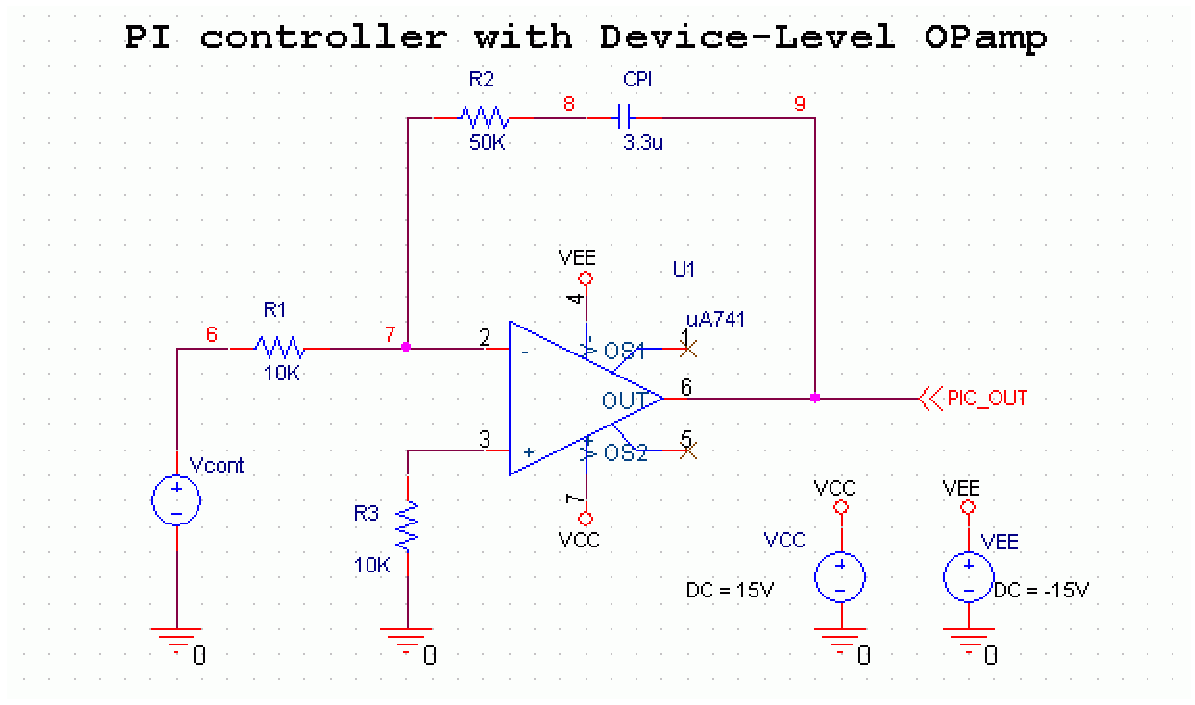


Figure A-30 PI controller with uA741

Figure A-31 shows the simulation results, $\omega S(t)$ for PI controller with µA741:

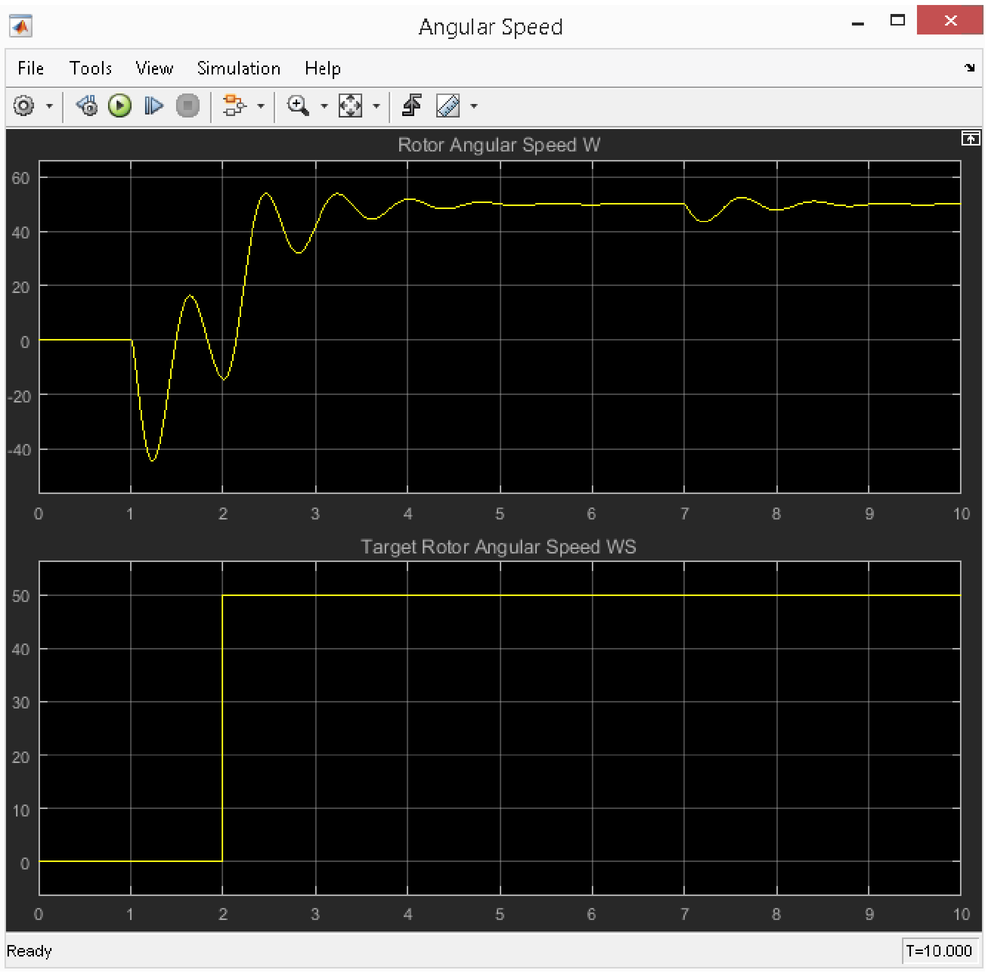


Figure A-31 Simulations results for PI controller with uA741

After the excitation is turned on at 1s, the motor starts to turn in the opposite direction., The device-level model behaves differently from the ideal Opamp model used in the controller. This is because the Opamp output goes into saturation as a result of the input offset voltage and the very high open-loop gain at DC operating point calculation at t=0.

To avoid the offset problem, $10M\Omega$ resister R_4 was put in the feedback loop to limit the DC gain to 1000, as shown in Figure A-32.

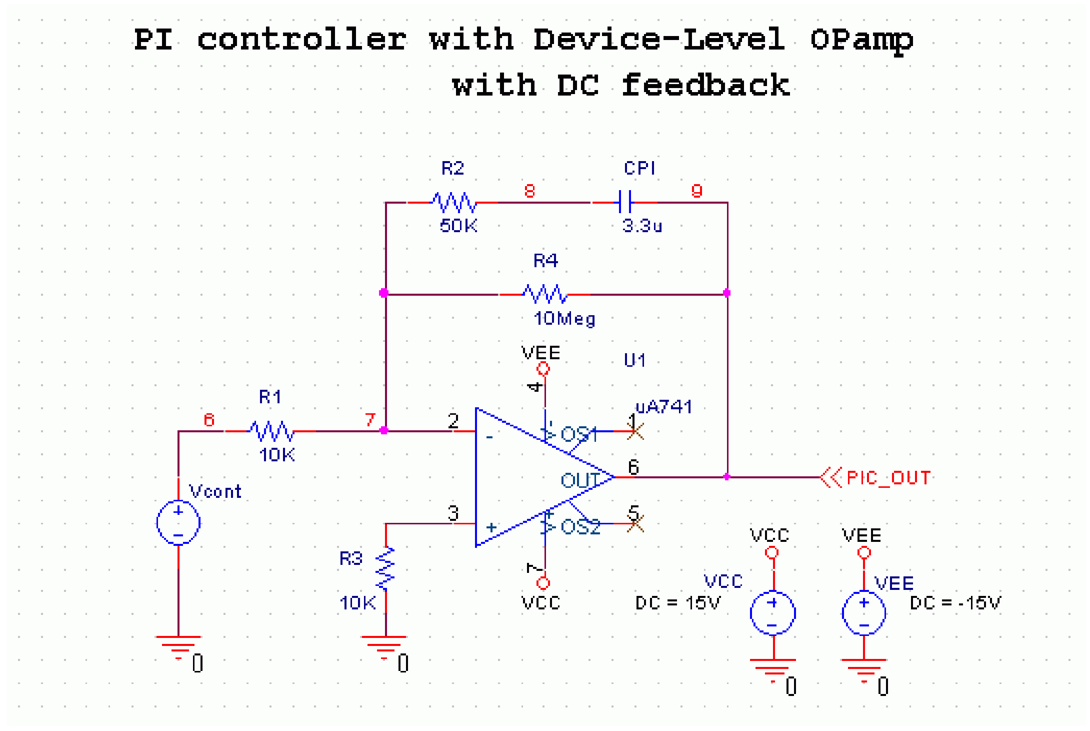


Figure A-32 PI controller circuit with DC feedback

Figure A-33 shows the result of the simulation for $\omega S(t)$ for PI controller with $R_4 = 10M\Omega$:

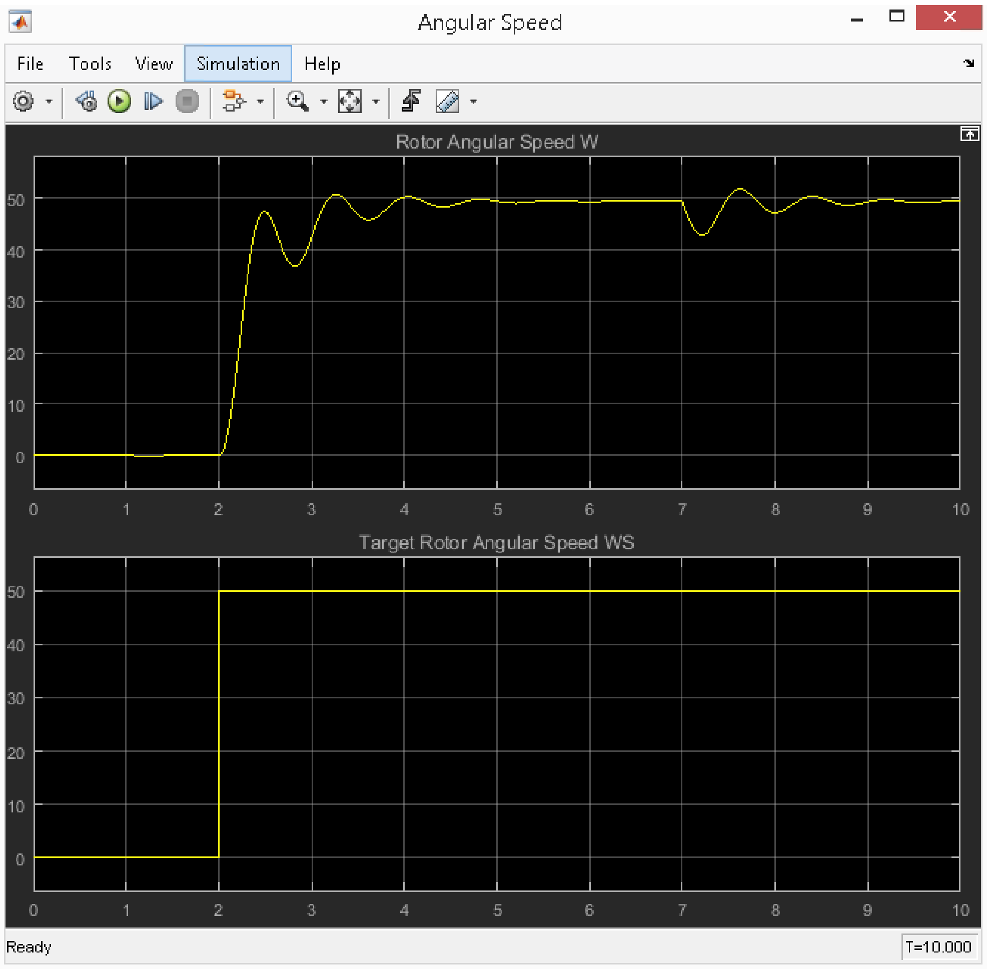


Figure A-33 Simulation results for PI controller with DC feedback

Because the Opamp output offset voltage is reduced and the initial transient in the opposite direction is also reduced. A negative effect of the reduced DC gain is that the steady-state response of $\omega(t)$ differs slightly from 50 rad/s.