Introduction to Modeling of Switched-Mode Power Converters Using MATLAB and Simulink

Extensive introductory tutorials for MATLAB and Simulink, including Control Systems Toolbox and Simulink Control Design are available on the mathworks.com site. Assuming basic familiarity with these tools, the purpose of this document is to introduce how they can be applied to modeling and simulations of switched-mode power converters. Using a synchronous buck converter as an example, the document goes through the MATLAB/Simulink modeling steps based on the materials presented in ECEN5797 (Introduction to Power Electronics):

- Constructing switching models based on converter state-space description, and performing time-domain simulations of the switching converter model
- Masking and parameterizing Simulink models
- Constructing large-signal averaged models and performing time-domain simulations of the averaged models
- Small-signal linearization of averaged models and plotting converter frequency responses
- Construction of closed-loop models, plotting loop-gain frequency responses, and performing time-domain simulations based on averaged or switching models

The approaches presented in this document require MATLAB Control System Toolbox and Simulink Control Design. Specialized circuit-simulation Simulink toolboxes such as SimPowerSystems are not required. All examples are done in MATLAB/Simulink release R2010a. Student version with the Simulink Control Design add-on is sufficient.

Contents

A synchronous buck converter switching model

A synchronous buck converter is shown in Fig. 1. To model the converter in Simulink, it is useful to explicitly identify the converter input and output signals.

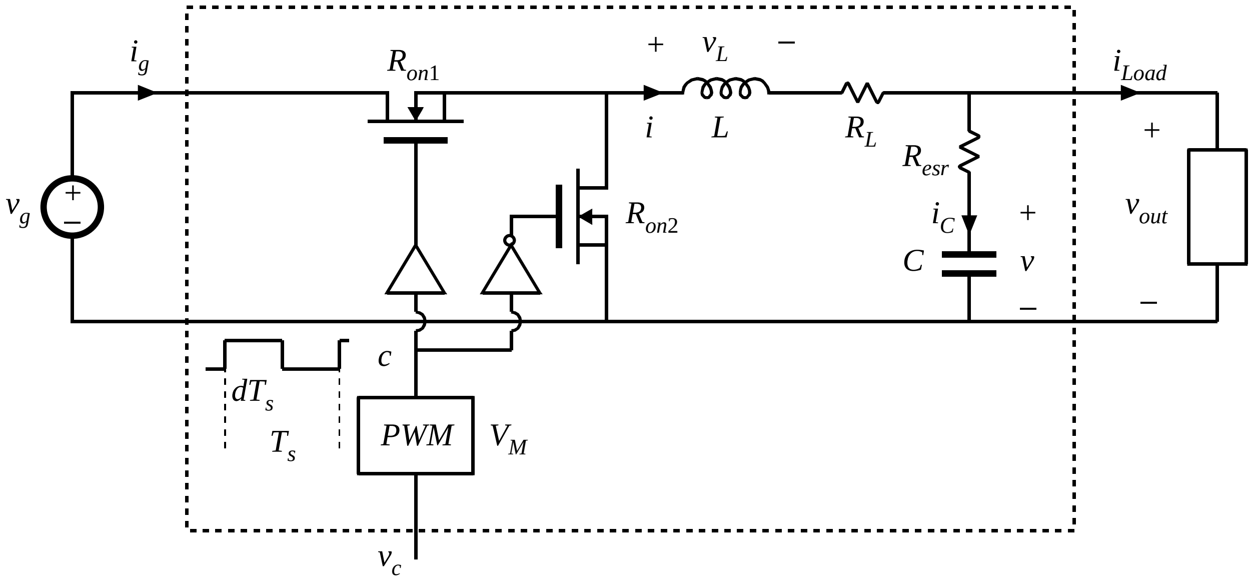
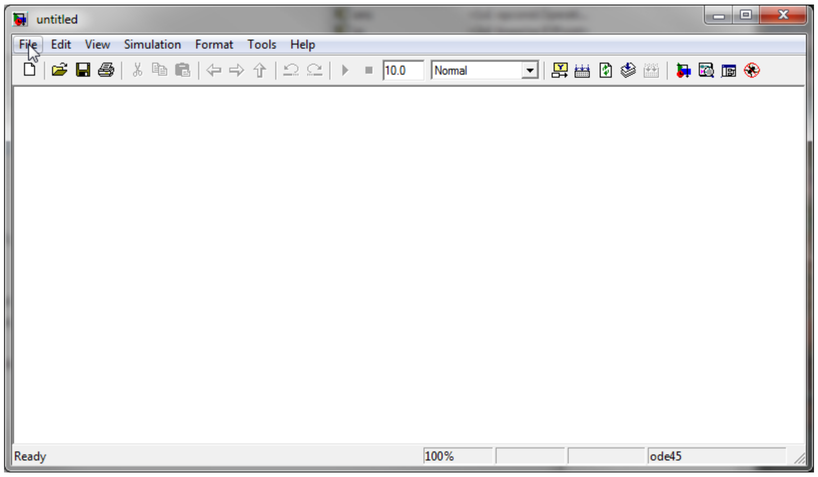


Figure 1: Synchronous buck converter. Numerical example: Vg = 5 V, L = 1 μH, RL = 10 mΩ, C = 200 μF, Resr = 0.8 mΩ, Ron1 = Ron2 = 20 mΩ, fs = 1 MHz, PWM ramp amplitude VM = 1 V.

In the converter of Fig. 1, the inputs are: input voltage vg, load current iLoad, and control voltage vc at the input of the pulse width modulator. The outputs are output voltage vout and input current ig. In addition, the converter states (inductor current i and capacitor voltage v) will be considered the outputs.

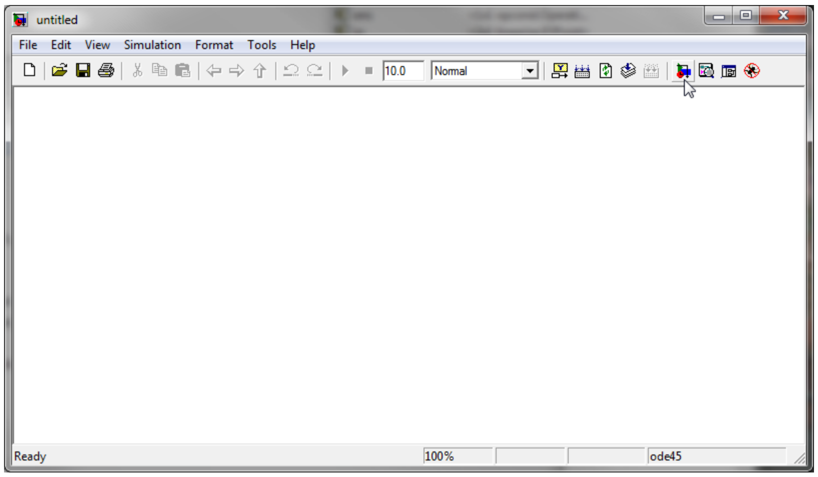
Top-level Simulink model: syncbuck_OL

Start MATLAB, and open a new model window (File, New, Model)



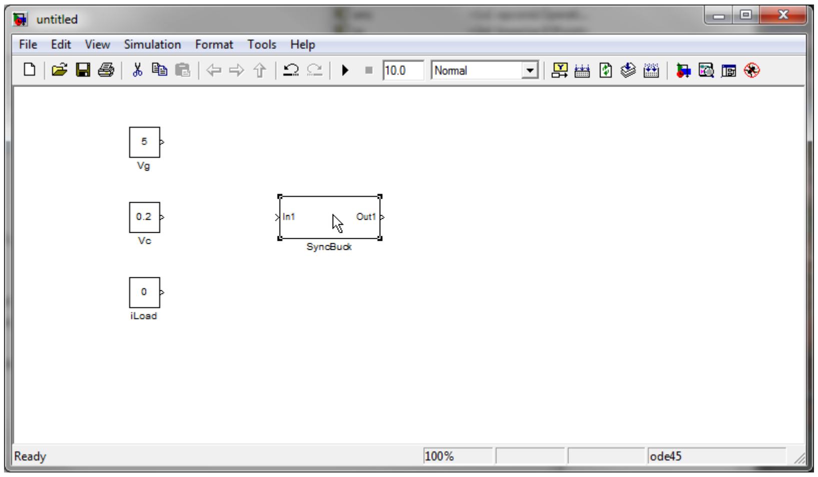
Name the model syncbuck_OL (File, Save As, syncbuck_OL)

Click to open the Simulink Library Browser



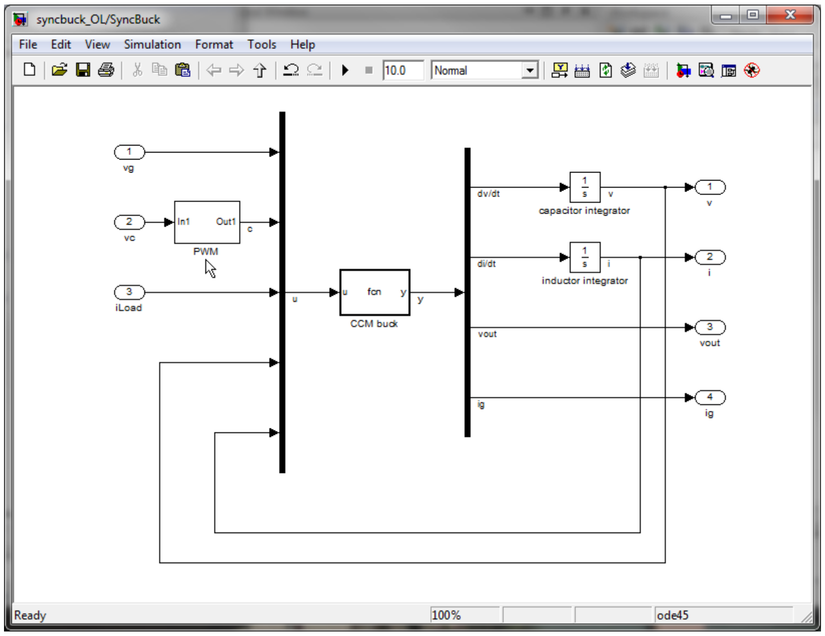
Add three Constant blocks to represent inputs, Vg, Vc, and iLoad

Add a Subsystem block, which will be used to model the buck converter, name it SyncBuck



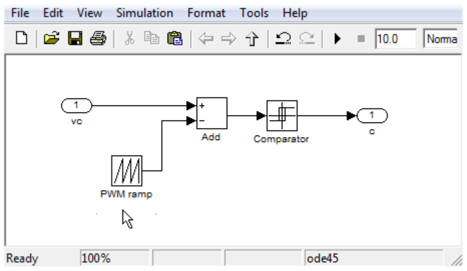
SyncBuck subsystem

Double click on the SyncBuck subsystem. In the subsystem window, add 3 input ports, 4 output ports, 2 integrators (1/s blocks), a Mux, a Demux, a subsystem to model the PWM, and an Embedded MATLAB function block to model the converter state equations.

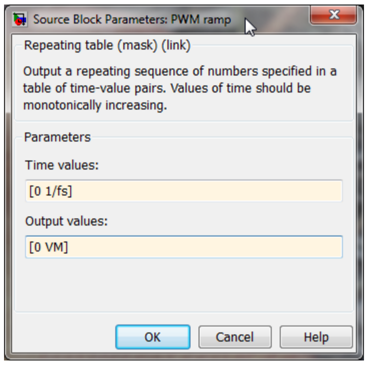


As you work through modeling steps, it is a good idea to save your work (as usual, ctrl-S is the key shortcut for save)

Double-click on the PWM subsystem, add a Relay block (from Discontinuities), a Repeating Sequence block (from Sources), and an Add block (from Math Operations). Change the Add block inputs to + and -. In the Relay block, zero is the default comparison threshold – that's fine. Optionally, you may rename the blocks to better describe the intended functions.



Double-click on the PWM ramp (Repeating Sequence) block. The Time values can be used to specify the switching period. For example, [0 1e-6] would be correspond to 1 MHz switching frequency. Better yet, we can enter an expression [0 1/fs] so that the same model can be used with switching frequency fs specified externally as a parameter. Similarly, output values [0 VM] will allow us to externally specify the PWM ramp amplitude VM as another parameter.


Source Block Parameters: PWM ramp

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:

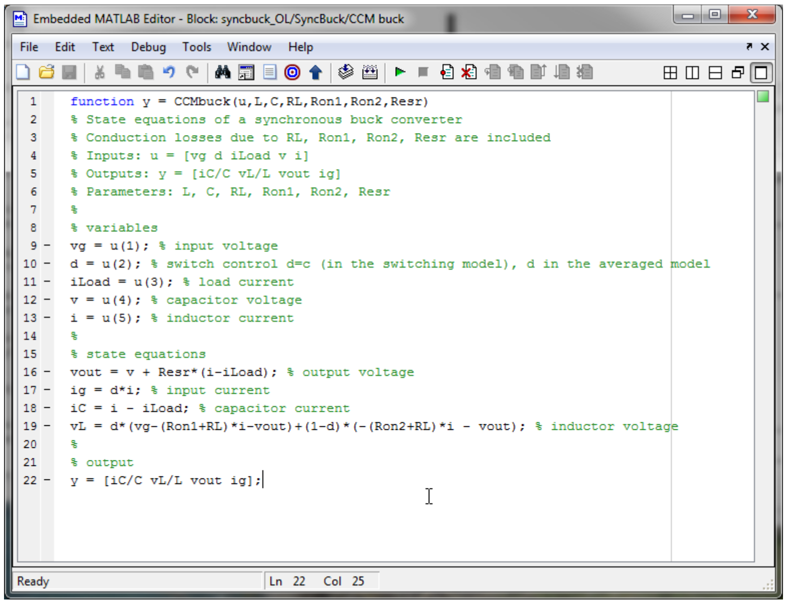
[0 1/fs]

Output values:

[0 VM]

Converter state equations

We now turn to writing the converter state equations in the Embedded MATLAB function block. Double-click on the block to open an Embedded MATLAB Editor window. Change the function name to better represent the intended function. In the function arguments add the parameters L, C, RL, Ron1, Ron2, Resr, and write the converter state equations. It is important to keep track of the order of variables in the input vector u and the output vector y.

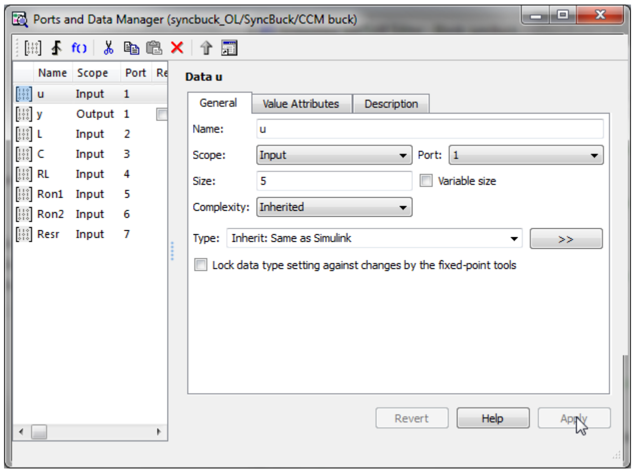

Embedded MATLAB Editor - Block: syncbuck_OL/SyncBuck/CCM buck

```
function y = CCMbuck(u,L,C,RL,Ron1,Ron2,Resr)
% State equations of a synchronous buck converter
% Conduction losses due to RL, Ron1, Ron2, Resr are included
% Inputs: u = [vg d iLoad v i]
% Outputs: y = [iC/C vL/L vout ig]
% Parameters: L, C, RL, Ron1, Ron2, Resr
%
% variables
vg = u(1); % input voltage
d = u(2); % switch control d=c (in the switching model), d in the averaged model
iLoad = u(3); % load current
v = u(4); % capacitor voltage
i = u(5); % inductor current
%
% state equations
vout = v + Resr*(i-iLoad); % output voltage
ig = d*i; % input current
iC = i - iLoad; % capacitor current
vL = d*(vg-(Ron1+RL)*i-vout)+(1-d)*(-(Ron2+RL)*i - vout); % inductor voltage
%
% output
y = [iC/C vL/L vout ig];
```

Next, it is necessary to let the function know that the input u is a vector of 5 input variables, and that L, C, RL, Ron1, Ron2, and Resr are the parameters.

Open the Ports and Data Manager window (Click on Tools, Edit Data/Ports)

Click on u, enter 5 in Size, and click Apply



Ports and Data Manager (syncbuck_OL/SyncBuck/CCM buck)

Name	Scope	Port
u	Input	1
y	Output	1
L	Input	2
C	Input	3
RL	Input	4
Ron1	Input	5
Ron2	Input	6
Resr	Input	7

Data u

General | Value Attributes | Description

Name: u

Scope: Input Port: 1

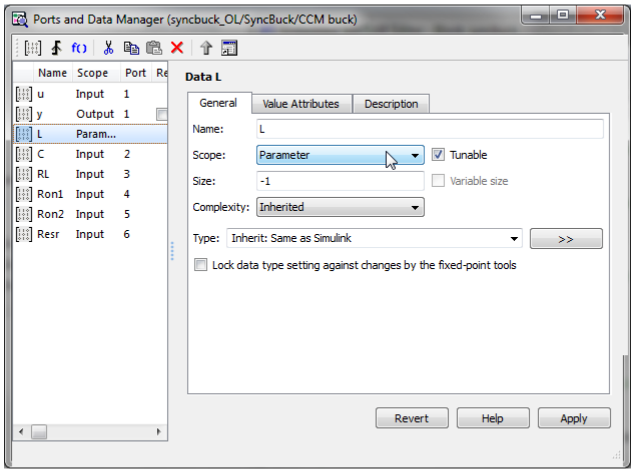
Size: 5 Variable size

Complexity: Inherited

Type: Inherit: Same as Simulink

Lock data type setting against changes by the fixed-point tools

Click on L, change Scope to Parameter, and click Apply



Ports and Data Manager (syncbuck_OL/SyncBuck/CCM buck)

Name	Scope	Port
u	Input	1
y	Output	1
L	Param...	
C	Input	2
RL	Input	3
Ron1	Input	4
Ron2	Input	5
Resr	Input	6

Data L

General | Value Attributes | Description

Name: L

Scope: Parameter [✓] Tunable

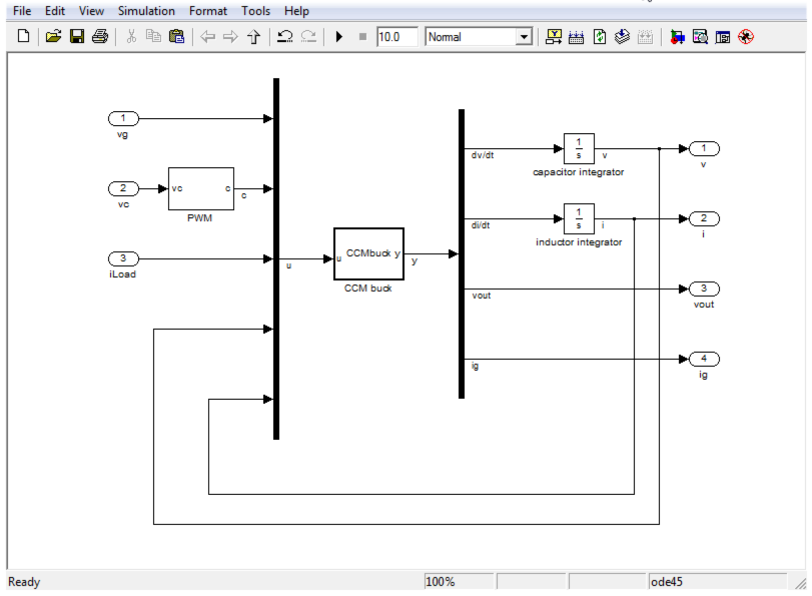
Size: -1 Variable size

Complexity: Inherited

Type: Inherit: Same as Simulink

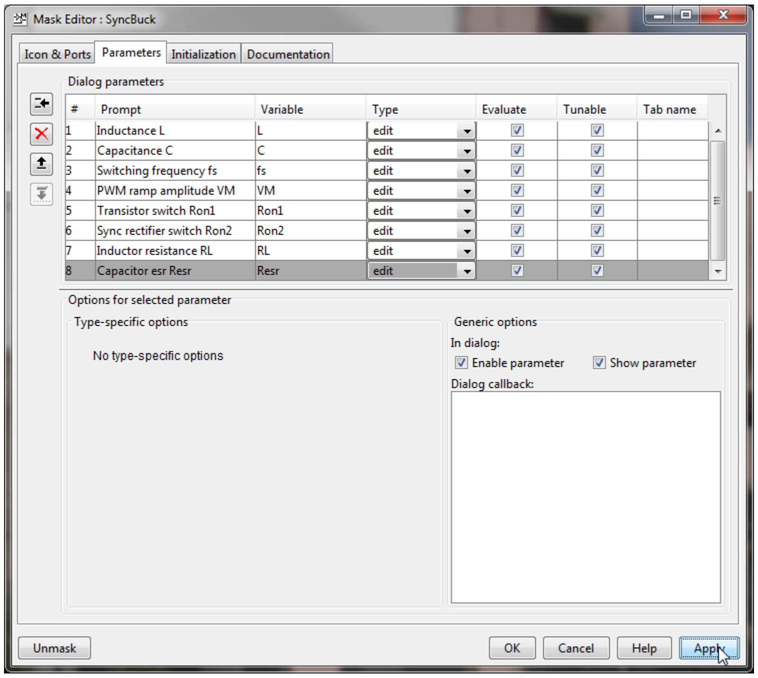
Lock data type setting against changes by the fixed-point tools

Repeat this step with other parameters: C, RL, Ron1, Ron2, Resr. Close the Ports and Data Manager window, and the Embedded MATLAB Editor. The SyncBuck subsystem should now look like this:



Masking and parameterizing the subsystem

It remains to provide a way to set the parameters of the converter model, i.e. the SyncBuck subsystem. A convenient way to do so in Simulink is to “mask” the subsystem. In the main syncbuck_OL window, right-click on the SyncBuck block and select Mask Subsystem…. A Mask Editor window shows up. Click on the Parameters, Click on the Add button, and enter the Prompt and the names of the Variables representing the parameters. The order is not important, but the Variable names must exactly match the symbols used for the parameters. Click Apply and OK to close to the Mask Editor.



Dialog parameters

#	Prompt	Variable	Type	Evaluate	Tunable	Tab name
1	Inductance L	L	edit	✓	✓	
2	Capacitance C	C	edit	✓	✓	
3	Switching frequency fs	fs	edit	✓	✓	
4	PWM ramp amplitude VM	VM	edit	✓	✓	
5	Transistor switch Ron1	Ron1	edit	✓	✓	
6	Sync rectifier switch Ron2	Ron2	edit	✓	✓	
7	Inductor resistance RL	RL	edit	✓	✓	
8	Capacitor esr Resr	Resr	edit	✓	✓	

Options for selected parameter

Type-specific options

No type-specific options

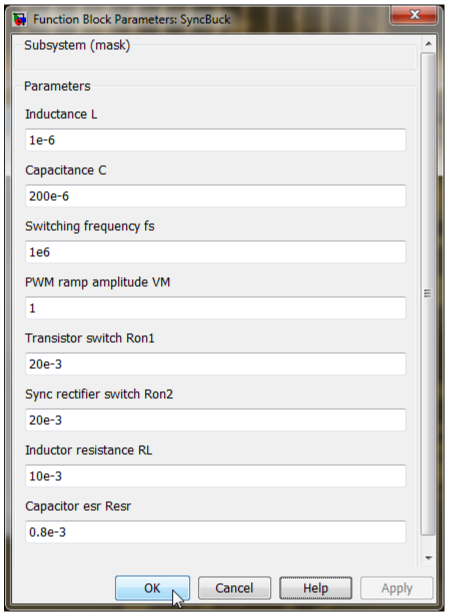
Generic options

In dialog:

✓ Enable parameter ✓ Show parameter

Dialog callback:

Back in the syncbuck_OL model window, double click on the SyncBuck subsystem. Instead of the subsystem internals, a dialog window now pops up, allowing you to enter the converter parameters. Following the numerical example in Fig. 1, enter the following parameters, click Apply and OK to close the dialog window for the SyncBuck subsystem.

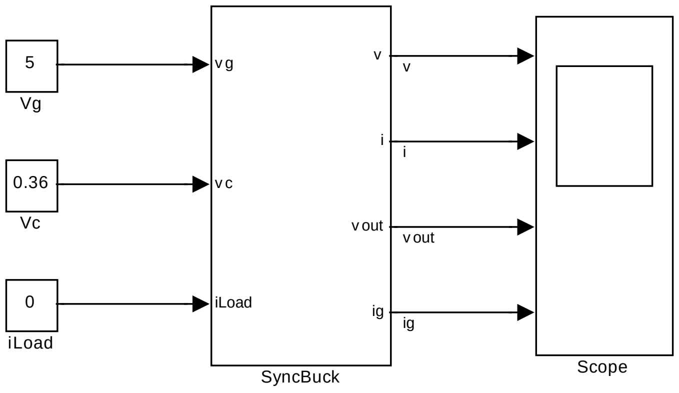


Inductance L
1e-6
Capacitance C
200e-6
Switching frequency fs
1e6
PWM ramp amplitude VM
1
Transistor switch Ron1
20e-3
Sync rectifier switch Ron2
20e-3
Inductor resistance RL
10e-3
Capacitor esr Resr
0.8e-3

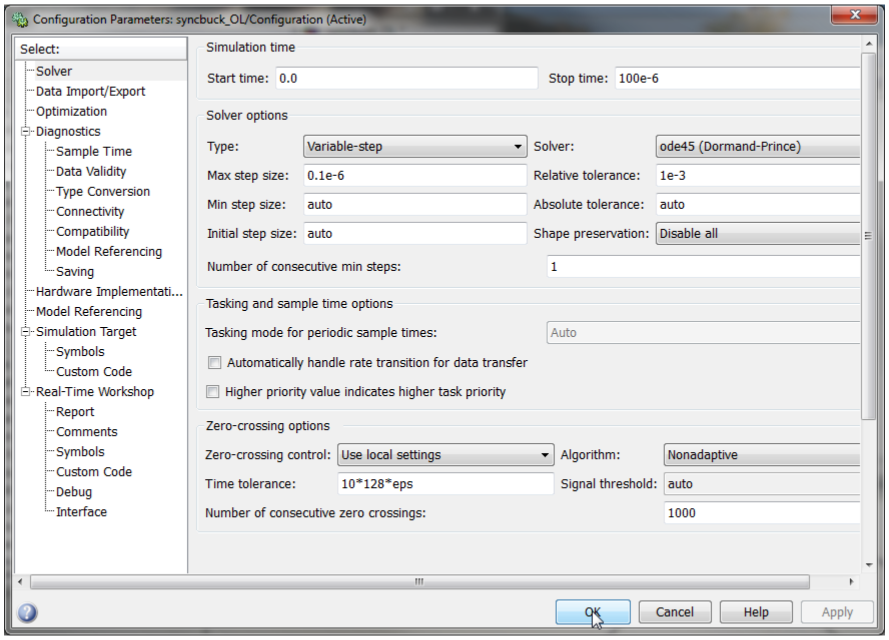
Further modifications in the masked subsystem can always be made by right-clicking on the subsystem and selecting Look Under Mask (to view or modify the model) or Edit Mask (to view or modify the parameters, description, help, or appearance of the block).

Switching model simulation

Back in the syncbuck_OL model window, connect the inputs and add a Scope block (from Sinks). In the Scope block set the Number of Axes to 4 and uncheck the “Limit data points to last” box in Data History. The top-level Simulink model should now look as shown below. Note that Vc is set to 0.36. Since VM = 1 V, this Vc results in duty-cycle D = 0.36. Given Vg = 5 V, the output dc voltage should ideally be Vout = D*Vg = 1.8 V.



Finally, to setup the simulation parameters, click on Simulation, Configuration Parameters, enter desired stop time (e.g. 100 microseconds), and change the Max step size from auto to one tenth of a switching period. Click Apply and OK to close the Configuration Parameters window.


Configuration Parameters: syncbuck_OL/Configuration (Active)

Simulation time

Start time: 0.0 Stop time: 100e-6

Solver options

Type: Variable-step Solver: ode45 (Dormand-Prince)

Max step size: 0.1e-6 Relative tolerance: 1e-3

Min step size: auto Absolute tolerance: auto

Initial step size: auto Shape preservation: Disable all

Number of consecutive min steps: 1

Tasking and sample time options

Tasking mode for periodic sample times: Auto

☐ Automatically handle rate transition for data transfer

☐ Higher priority value indicates higher task priority

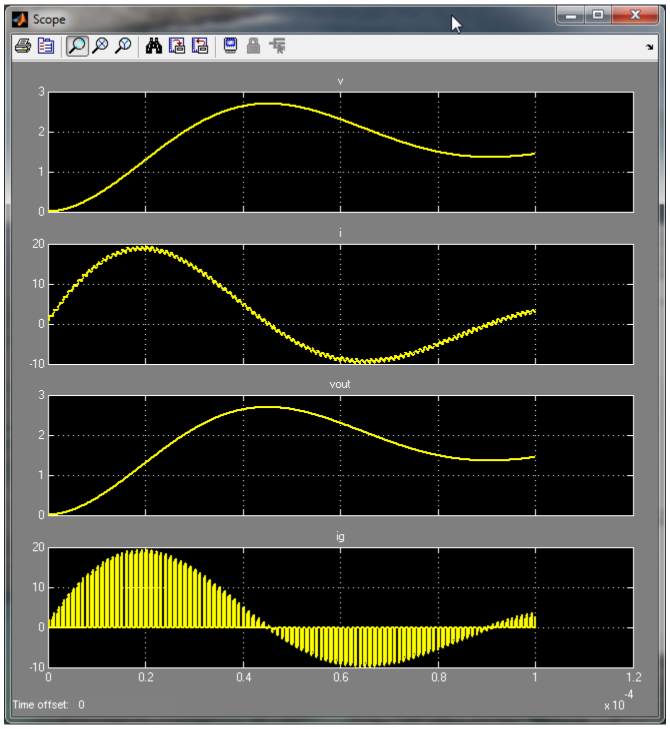
Zero-crossing options

Zero-crossing control: Use local settings Algorithm: Nonadaptive

Time tolerance: 10*128*eps Signal threshold: auto

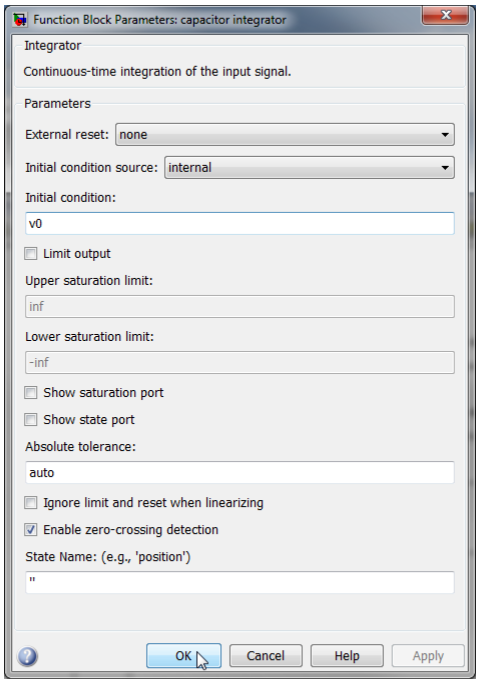
Number of consecutive zero crossings: 1000

In the syncbuck_OL model window Click on the Start simulation button. Double click on the Scope block to view the results. The waveforms show a start-up transient from zero initial conditions, and include switching ripples.



Adding parameters to a masked subsystem

It would be convenient to add initial values for the capacitor voltage and the inductor current as additional parameters for the SyncBuck subsystem. To do so, Look Under Mask, double-click on the capacitor integrator, and enter v0 in the Initial condition field.


Function Block Parameters: capacitor integrator

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

v0

☐ Limit output

Upper saturation limit:

inf

Lower saturation limit:

-inf

☐ Show saturation port

☐ Show state port

Absolute tolerance:

auto

☐ Ignore limit and reset when linearizing

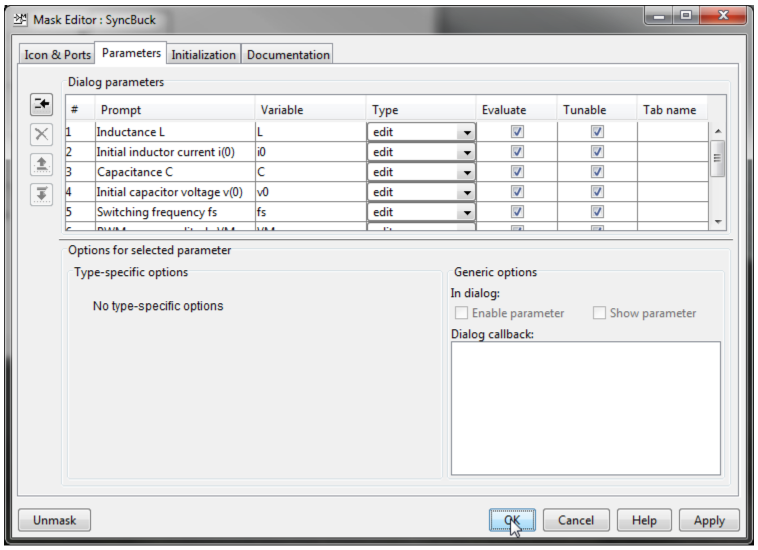
☑ Enable zero-crossing detection

State Name: (e.g., 'position')

''

Similarly, enter i0 as the initial value of the inductor current in the Inductor integrator block.

Next Edit Mask and enter the new parameters


Mask Editor : SyncBuck

Icon & Ports | Parameters | Initialization | Documentation

Dialog parameters

#	Prompt	Variable	Type	Evaluate	Tunable	Tab name
1	Inductance L	L	edit	✓	✓	
2	Initial inductor current i(0)	i0	edit	✓	✓	
3	Capacitance C	C	edit	✓	✓	
4	Initial capacitor voltage v(0)	v0	edit	✓	✓	
5	Switching frequency fs	fs	edit	✓	✓	

Options for selected parameter

Type-specific options

No type-specific options

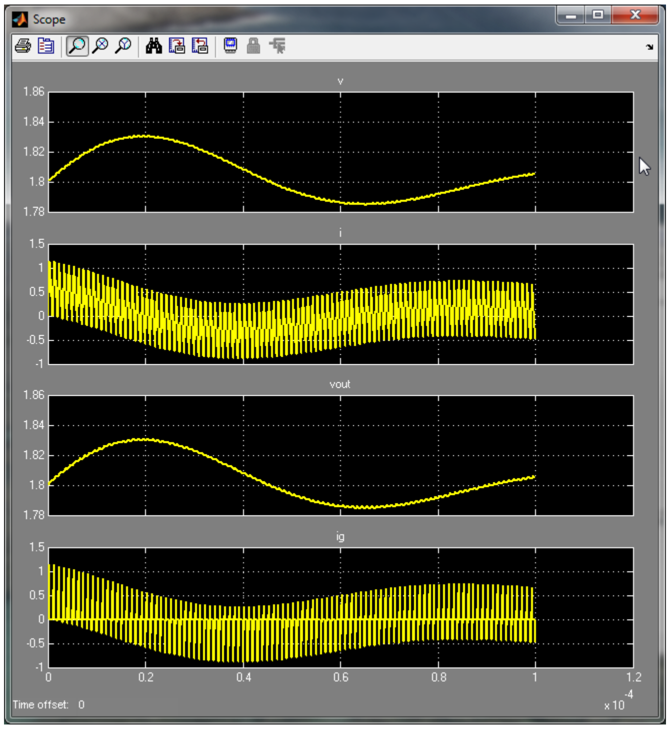
Generic options

In dialog:

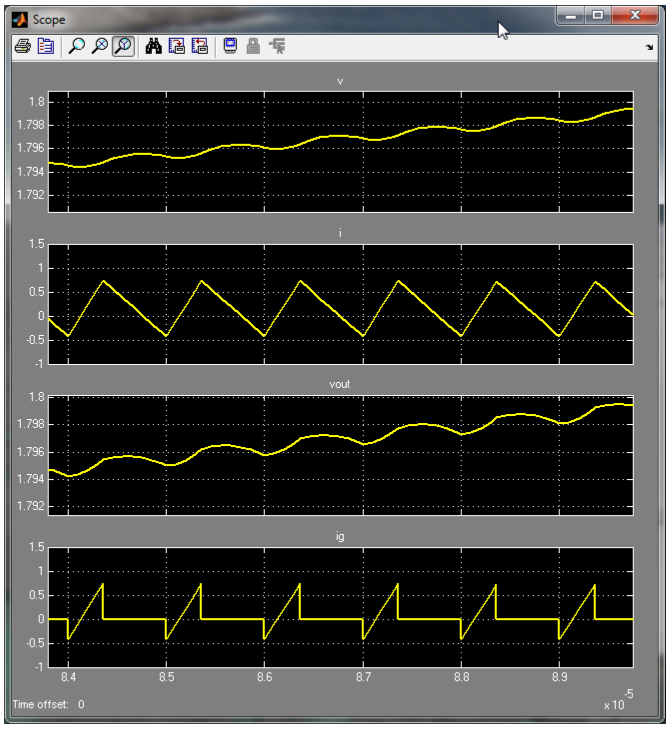
☐ Enable parameter ☐ Show parameter

Dialog callback:

Back in the syncbuck_OL model window, double-click on the SyncBuck subsystem to enter iL(0) = 0 and vc(0) = 1.8, and run simulation again. Since the converter starts from the initial conditions much closer to the steady-state values, the start-up transient waveforms are now close to steady-state.



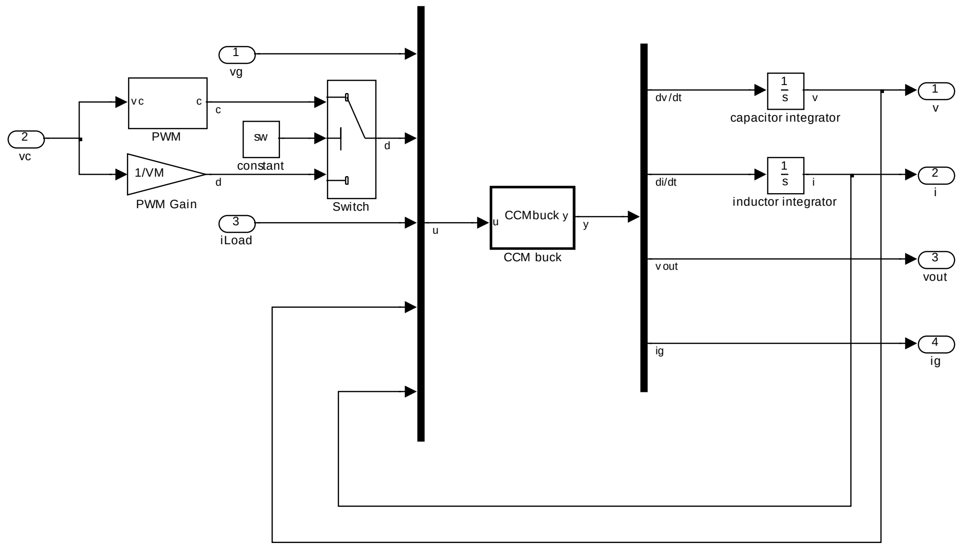
You may zoom-in to look at the switching-ripple details



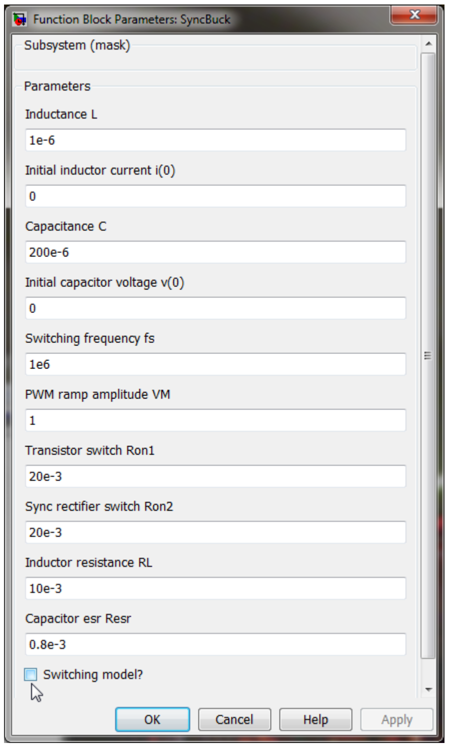
Averaged model of the synchronous buck converter

While transient simulations may be of interest to verify converter operation or performance, such simulations are of limited value in the design process. As emphasized in ECEN5797 and ECEN5807, averaging provides a way to remove switching-ripple and modulation sideband complexities, and focus on low-frequency converter dynamics most relevant in the controller design process. To construct an averaged model of the synchronous buck converter example, we follow the state-space averaging approach and take advantage of the fact that the converter state equations have already been written in the form that allows replacing the switch control signal c(t) with the averaged duty cycle waveform d(t). As a result, simple modification described below allows the same Simulink model to be used for switching or for averaged model simulations.

Look Under Mask of the SyncBuck subsystem again, and modify the path between the control voltage vc and the switch control c using a Switch block (from Signal Routing), a Gain block (from Math Operations) and a Constant block. Change the Switch threshold to 0.5, and enter parameter sw as the constant. This parameter can be set to 0 (to get an averaged model in which the PWM is modeled as a 1/VM gain to generate the duty-cycle d), or 1 (to get a switching model in which the actual PWM is used to generate the pulsating switch control c(t)).



Next Edit Mask, and add another parameter as a checkbox. The parameter dialog window now includes a check box that can be used to activate the switching model. Leave it unchecked to work with the averaged model.



Function Block Parameters: SyncBuck

Subsystem (mask)

Parameters

Inductance L

1e-6

Initial inductor current i(0)

0

Capacitance C

200e-6

Initial capacitor voltage v(0)

0

Switching frequency fs

1e6

PWM ramp amplitude VM

1

Transistor switch Ron1

20e-3

Sync rectifier switch Ron2

20e-3

Inductor resistance RL

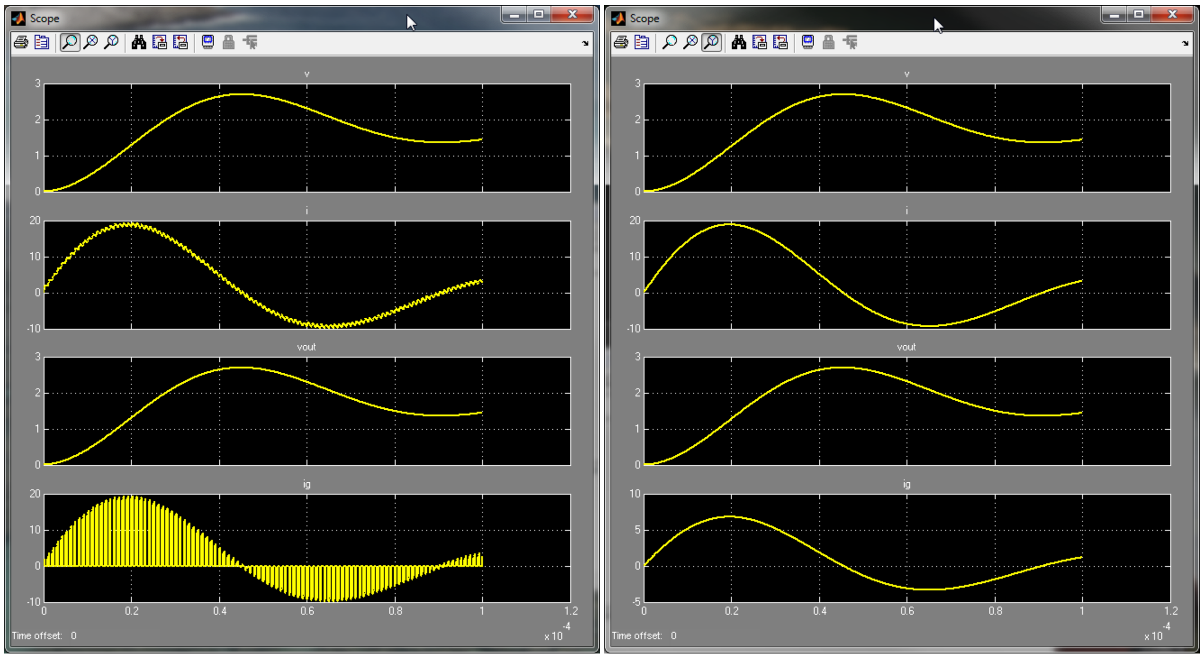
10e-3

Capacitor esr Resr

0.8e-3

☐ Switching model?

Leave the box unchecked and rerun the start-up simulation from zero initial conditions. You may compare the waveforms to the switching simulations: switching ripples are no longer shown, but the large-signal averaged model retains the converter dynamics essential for the controller design.



Small-signal linearization and frequency responses

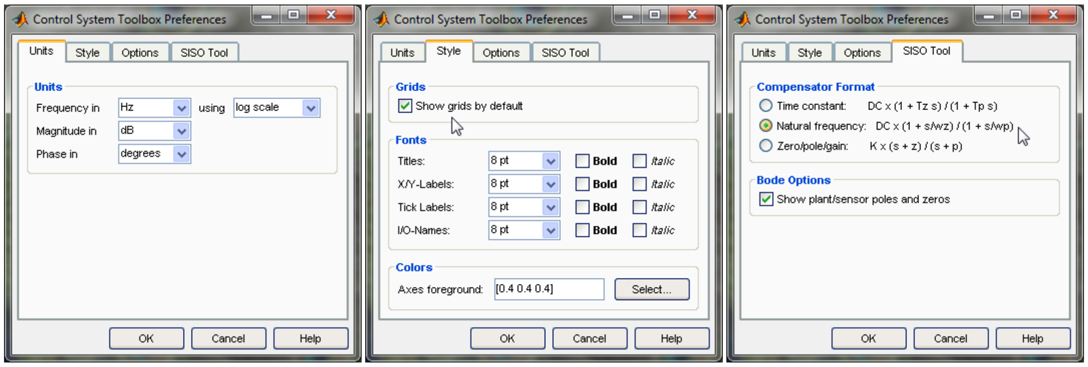
Similar to Spice .ac simulation, MATLAB/Simulink can linearize the large-signal averaged model. The linearized model can then be used to examine converter open-loop or closed-loop frequency responses, perform or verify controller design, etc. In this area, MATLAB Simulink capabilities (with the Control Systems Toolbox and the Simulink Control Design components) far exceed capabilities of traditional Spice simulation tools.

Control System Toolbox preferences

Before starting work on linearized models, let's change a few Control System Toolbox preferences. In the main MATLAB window, at the command prompt enter

```
>> ctrlpref
```

and adjust the preferences as shown below



Units
Frequency in: Hz using log scale
Magnitude in: dB
Phase in: degrees

Grids
☑ Show grids by default

Fonts

Titles:	8 pt	☐ **Bold**	☐ *Italic*
X/Y-Labels:	8 pt	☐ **Bold**	☐ *Italic*
Tick Labels:	8 pt	☐ **Bold**	☐ *Italic*
I/O-Names:	8 pt	☐ **Bold**	☐ *Italic*

Colors
Axes foreground: [0.4 0.4 0.4]

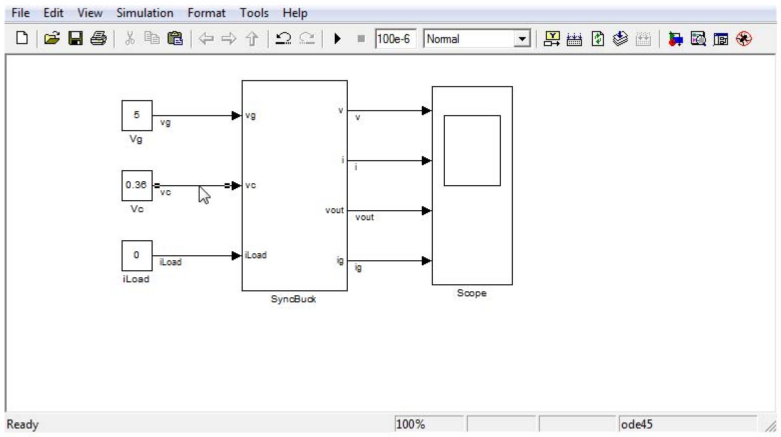
Compensator Format
○ Time constant: DC x (1 + Tz s) / (1 + Tp s)
● Natural frequency: DC x (1 + s/wz) / (1 + s/wp)
○ Zero/pole/gain: K x (s + z) / (s + p)

Bode Options
☑ Show plant/sensor poles and zeros

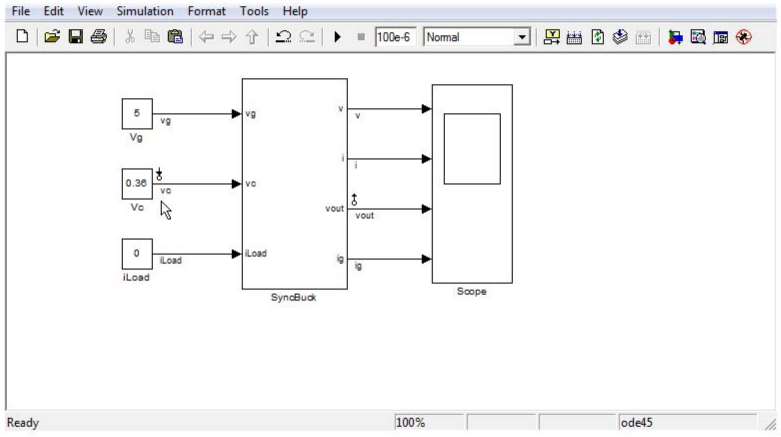
Linearization inputs and outputs

In the syncbuck_OL model, first make sure that the Switching model? checkbox is unchecked, i.e. that you are working with the averaged converter model. Otherwise, linearization and frequency-domain simulation steps described below cannot be applied – these steps do not make sense with the switching model.

First input and output points for the linearized model must be added to the syncbuck_OL model. Suppose that we are interested in finding the open-loop control-to-output transfer function Gvc = (1/VM)Gvd, i.e. the small-signal transfer function from vc to vout. Right-click on the vc signal, i.e. the connection between the Vc constant block and the vc input port of the SyncBuck subsystem



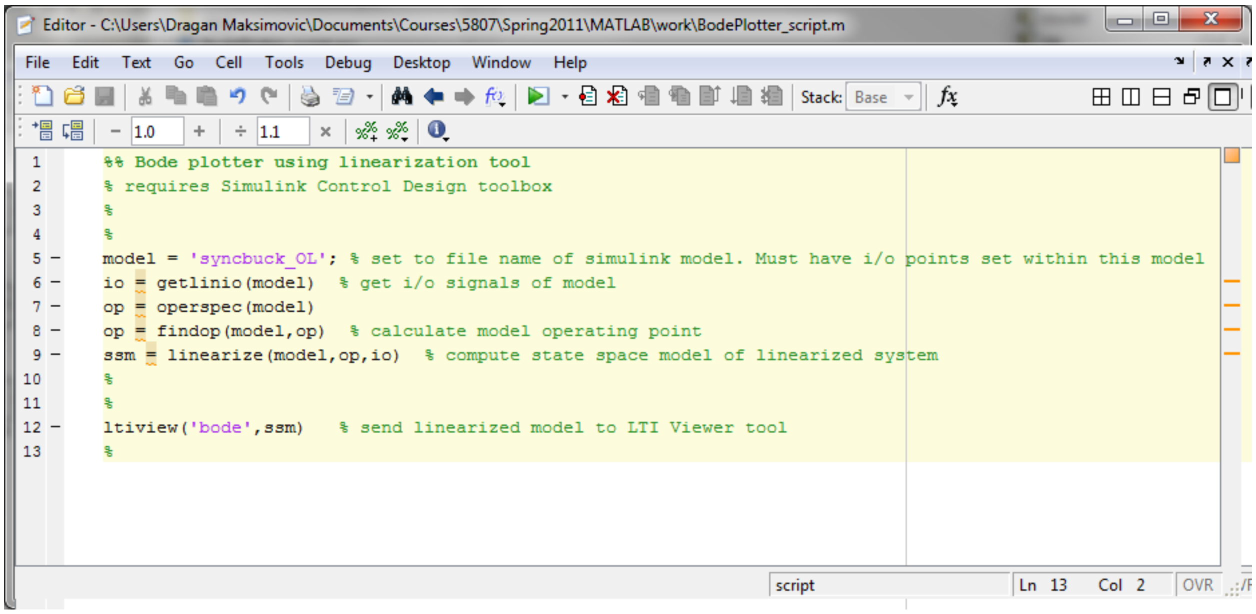
Select Linearization Points, Input Point. Similarly, right-click on the vout signal and select Linearization Points, Output Point. Note the small arrows indicating the input and the output point.



Simulink model linearization and plotting of frequency responses

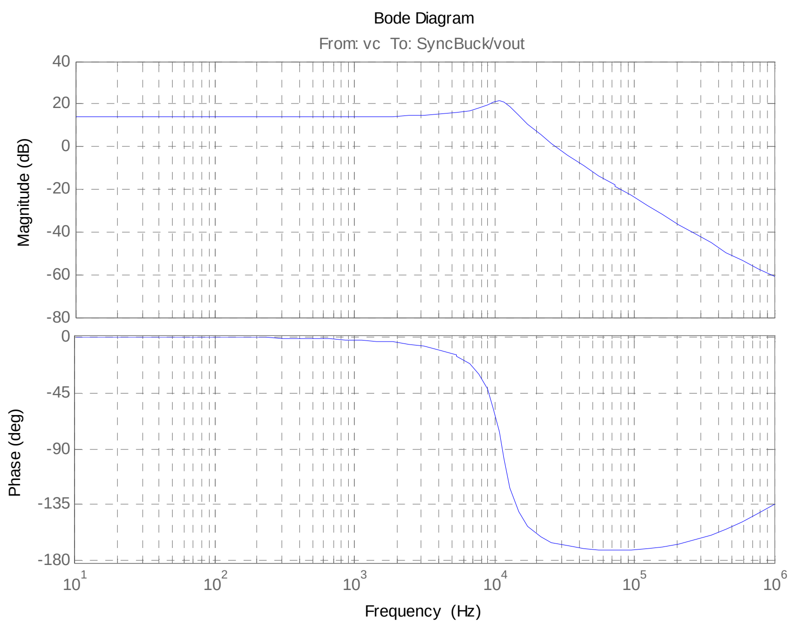
There are many ways to perform small-signal linearization, including a graphical user interface with many options (Tools, Control Design, Linear Analysis…). Instead, we show here how simple tasks such as plotting magnitude and phase responses can be automated by writing MATLAB scripts.

From the main MATLAB window, do File, New, Script, save the file as BodePlotter_script and enter the script as follows


Editor - C:\Users\Dragan Maksimovic\Documents\Courses\5807\Spring2011\MATLAB\work\BodePlotter_script.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
script
Ln 13 Col 2
OVR

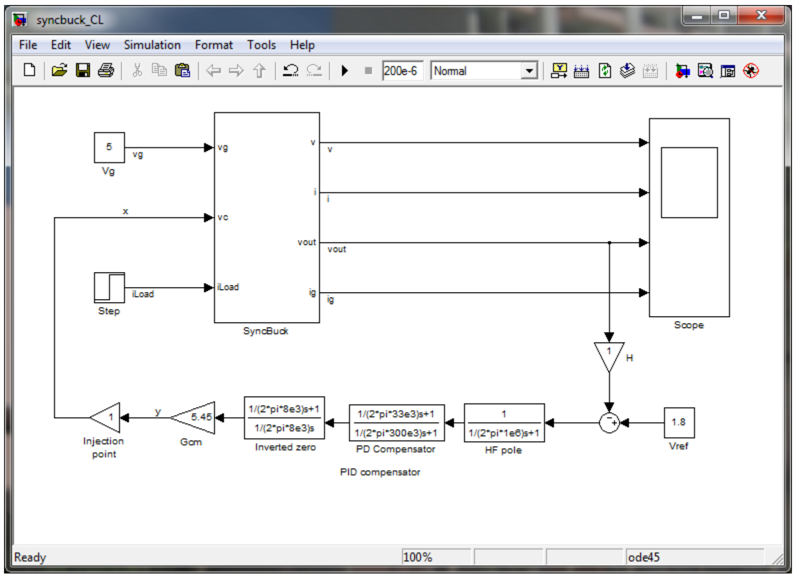
```
%% Bode plotter using linearization tool
% requires Simulink Control Design toolbox
%
%
model = 'syncbuck_OL'; % set to file name of simulink model. Must have i/o points set within this model
io = getlinio(model)   % get i/o signals of model
op = operspec(model)
op = findop(model,op)   % calculate model operating point
ssm = linearize(model,op,io)   % compute state space model of linearized system
%
%
ltiview('bode',ssm)    % send linearized model to LTI Viewer tool
%
```

Click on the Run button, or in the MATLAB widow type BodePlotter_script and enter. An LTI (linear time invariant) viewer windows opens and shows the converter open-loop control-to-output magnitude and phase responses. The LTI Viewer has many options, e.g. to adjust the scales, display other properties of the linearized model, etc. The Bode Diagram shows the familiar magnitude and phase responses of the buck converter control-to-output transfer function.



Closed-loop modeling and simulations

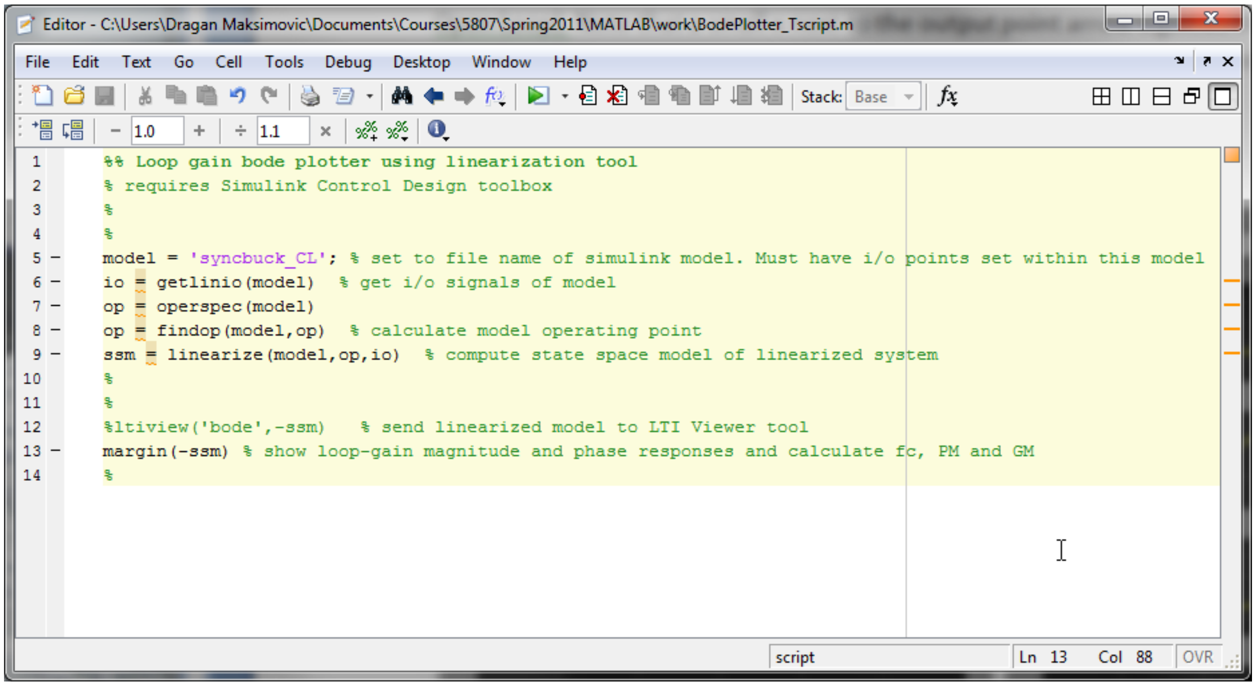
Next, save the model syncbuck_OL as syncbuck_CL, and add feedback components to close the voltage-mode control loop using the PID compensator design example described in class notes.



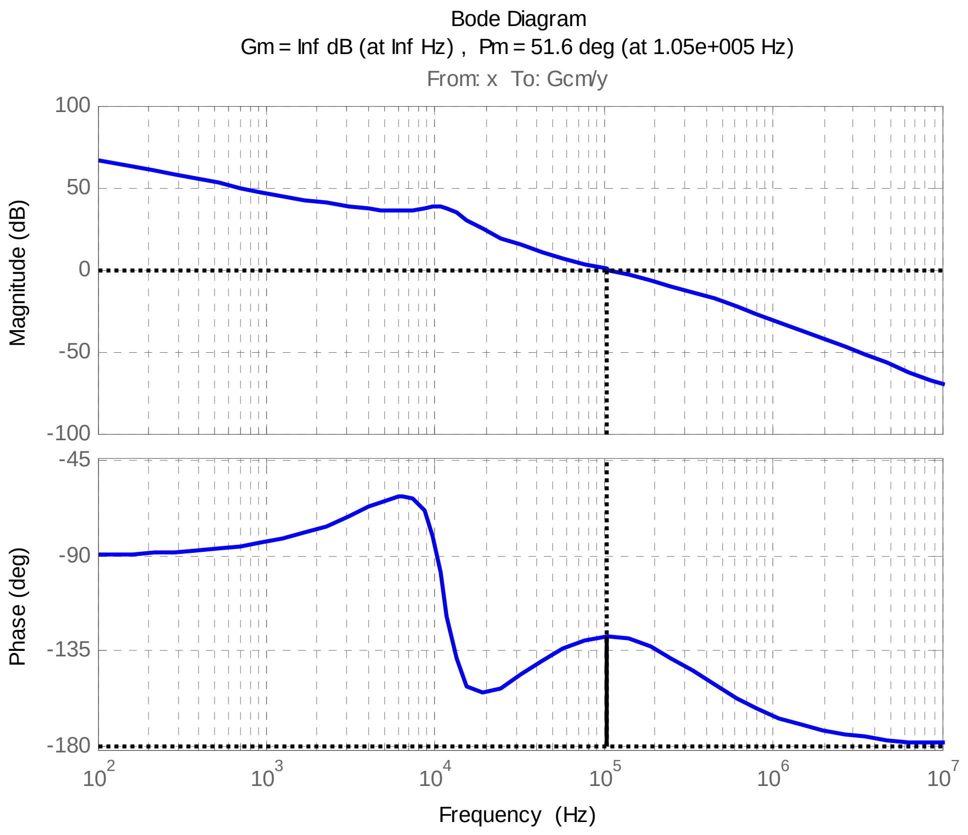
In Simulink, the compensator transfer function can be realized in a number of different ways. For clarity, the realization shown above consists of three Transfer Fcn blocks (from Continuous) and a Gain block. The sensor gain is H = 1. The loop includes an additional Gain block with a gain of 1. This block serves as the injection point for evaluation of the loop-gain (following the injection approach discussed in the ECEN5797/5807 Textbook). To perform linearization, add an input point at the output of the injection point Gain block, and an output point at the input of the injection point Gain block. To get the loop gain, make sure that you select Open Loop for the output point, (right-click, Linearization Points, Open Loop).

A small x next to the output point arrow signifies that the system is linearized in open-loop (effectively breaking the small-signal loop at the injection point, while still keeping the loop closed in operating point calculations).

The script to plot the loop-gain magnitude and phase responses is a small modification of the BodePlotter_script. Instead of the LTI viewer, MATLAB function margin is used to plot the loop-gain magnitude and phase responses, and to calculate the cross-over frequency, phase margin and gain margin. Note that a minus sign is added in front of the small-signal model ssm, so that the loop gain is correctly displayed as T = -vy/vx.

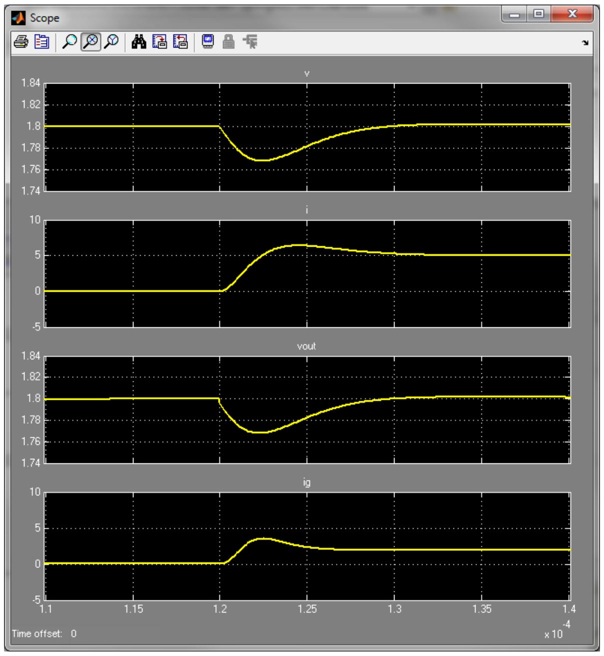

Editor - C:\Users\Dragan Maksimovic\Documents\Courses\5807\Spring2011\MATLAB\work\BodePlotter_Tscript.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
script
Ln 13 Col 88
OVR

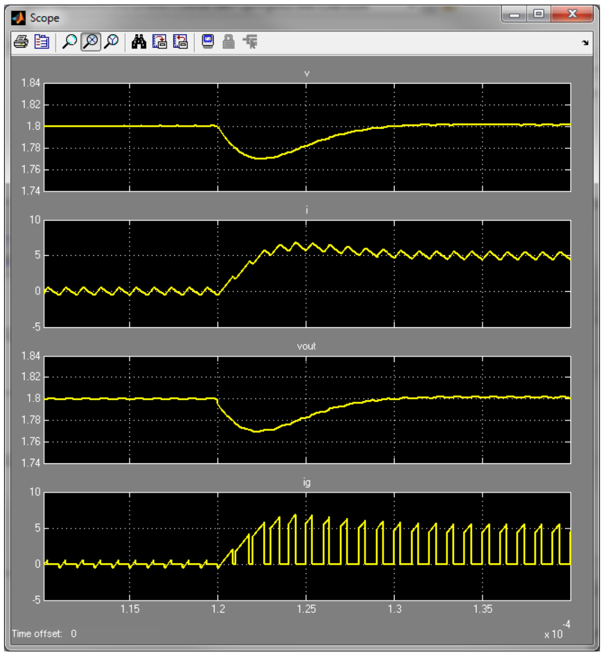
```
%% Loop gain bode plotter using linearization tool
% requires Simulink Control Design toolbox
%
%
model = 'syncbuck_CL'; % set to file name of simulink model. Must have i/o points set within this model
io = getlinio(model)  % get i/o signals of model
op = operspec(model)
op = findop(model,op)  % calculate model operating point
ssm = linearize(model,op,io)  % compute state space model of linearized system
%
%
%ltiview('bode',-ssm)   % send linearized model to LTI Viewer tool
margin(-ssm) % show loop-gain magnitude and phase responses and calculate fc, PM and GM
%
```



The results confirm the designed cross-over frequency and the phase margin.

Finally, you may perform transient simulations with the averaged model or with the switching model to verify performance under step-load transients or other disturbances.


Scope
v
1.84
1.82
1.8
1.78
1.76
1.74
i
10
5
0
-5
vout
ig
1.1
1.15
1.2
1.25
1.3
1.35
1.4
x 10^-4
Time offset: 0


Scope
v
1.84
1.82
1.8
1.78
1.76
1.74
i
10
5
0
-5
vout
ig
1.15
1.2
1.25
1.3
1.35
x 10^-4
Time offset: 0