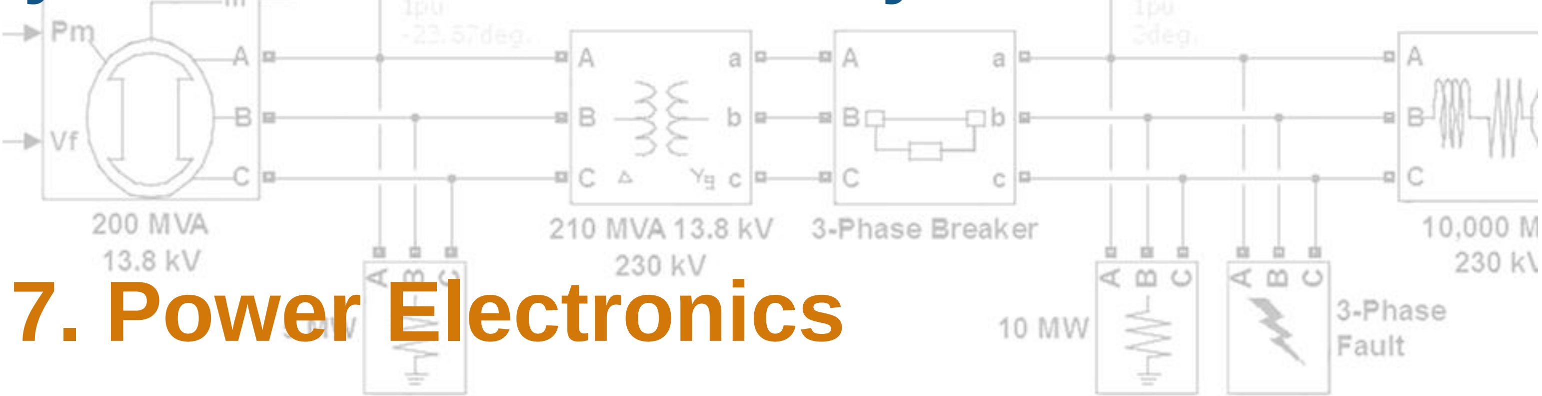
SimPowerSystems Hands-on Workshop:
Modeling and Simulation of Electrical Power Systems with SimPowerSystems™



7. Power Electronics

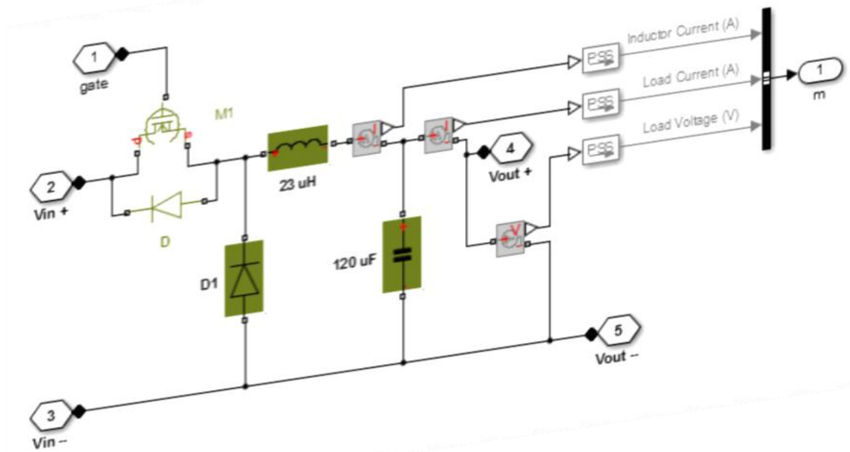
Carlos Osorio
Principal Application Engineer
MathWorks – Natick, MA

© 2014 The MathWorks, Inc.

Outline

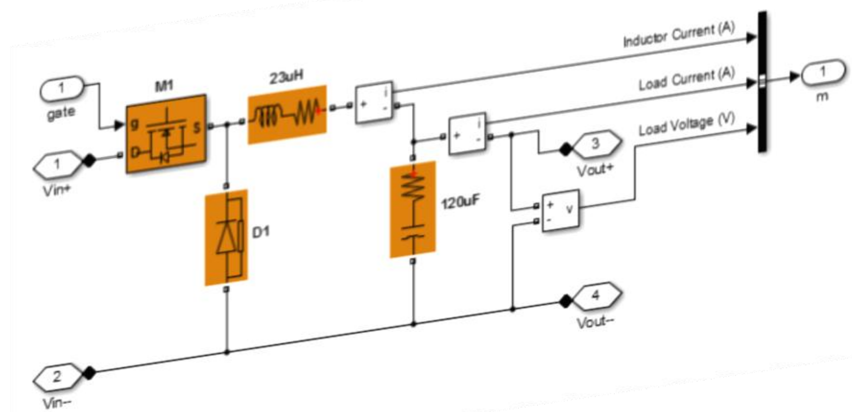
- SimElectronics or SimPowerSystems?
- Ideal switching algorithm
- Power quality and harmonic analysis
- Control design and linearization

SimElectronics or SimPowerSystems?



SimPowerSystems

Piecewise linear systems solution
Multiphase bridges and pulse generators
Detailed and average voltage models
Transient and harmonic analysis
Faster simulation



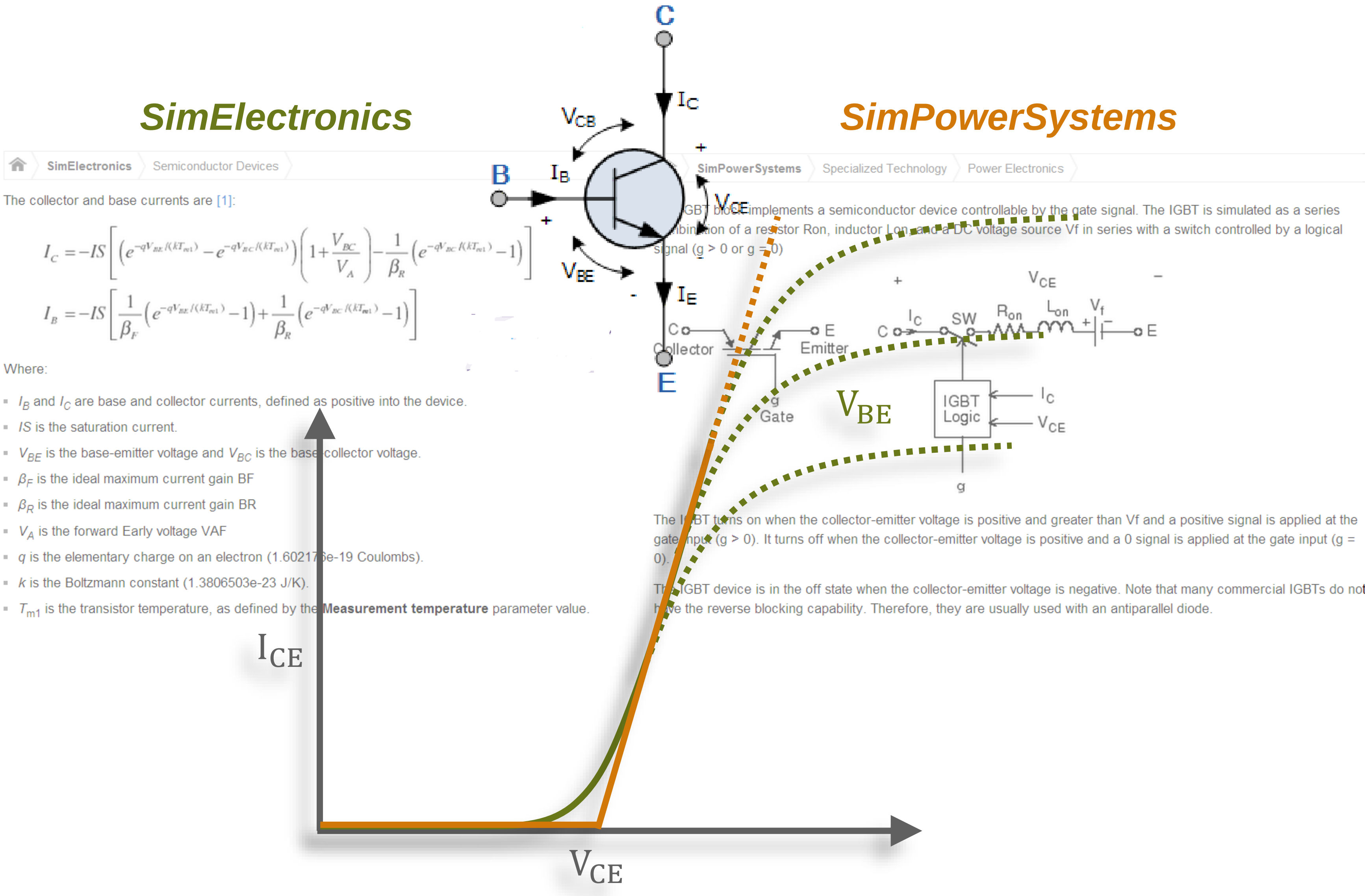
SimElectronics

Simultaneous nonlinear equations solution
SPICE level switching device models
Include switching losses
Include parasitic current effects
Include temperature effects
Higher fidelity simulation

```
>> se_dcdcbuckconverter
```

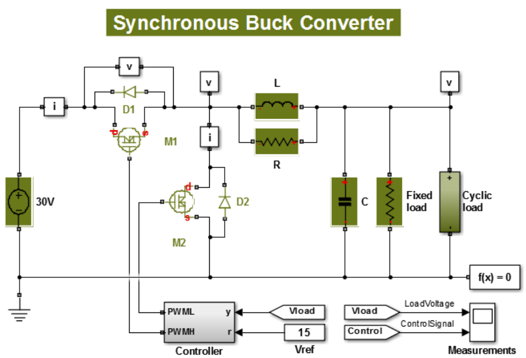
```
>> sps_dcdcbuckconverter
```

SimElectronics or SimPowerSystems?

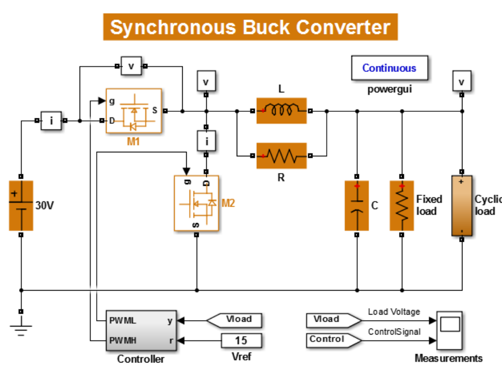


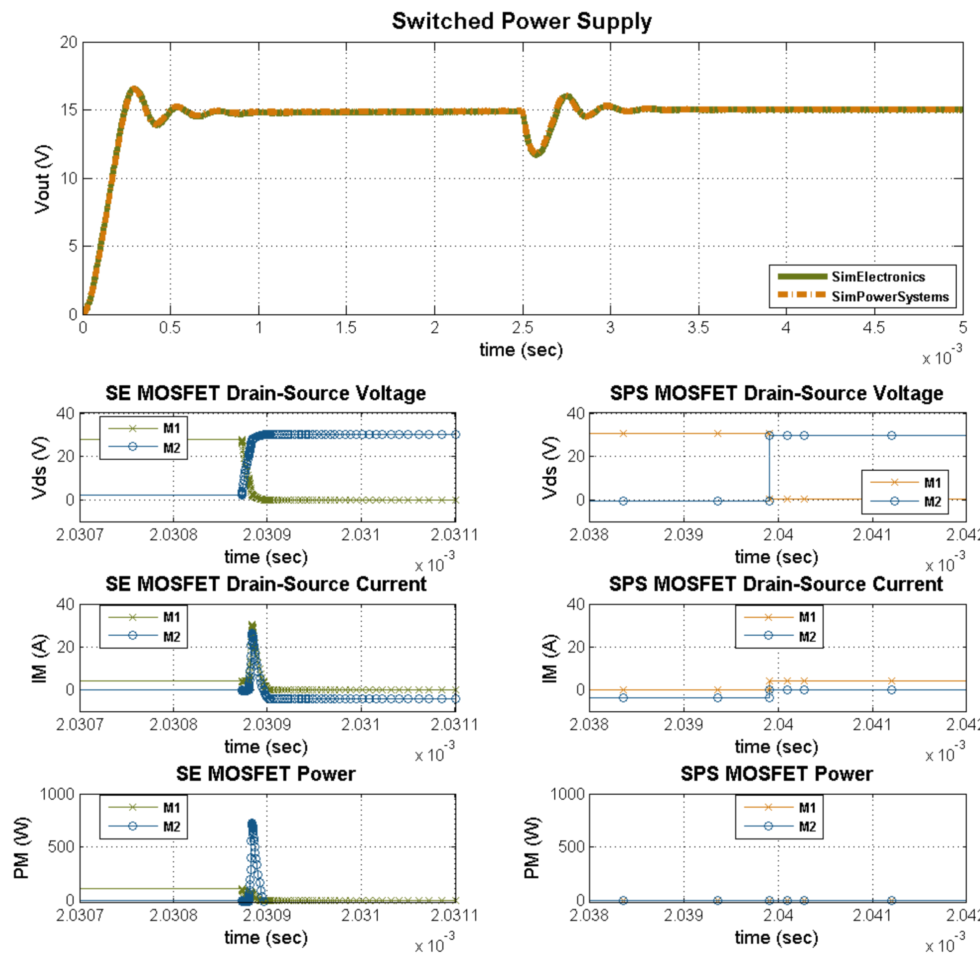
SimElectronics or SimPowerSystems?

SimElectronics



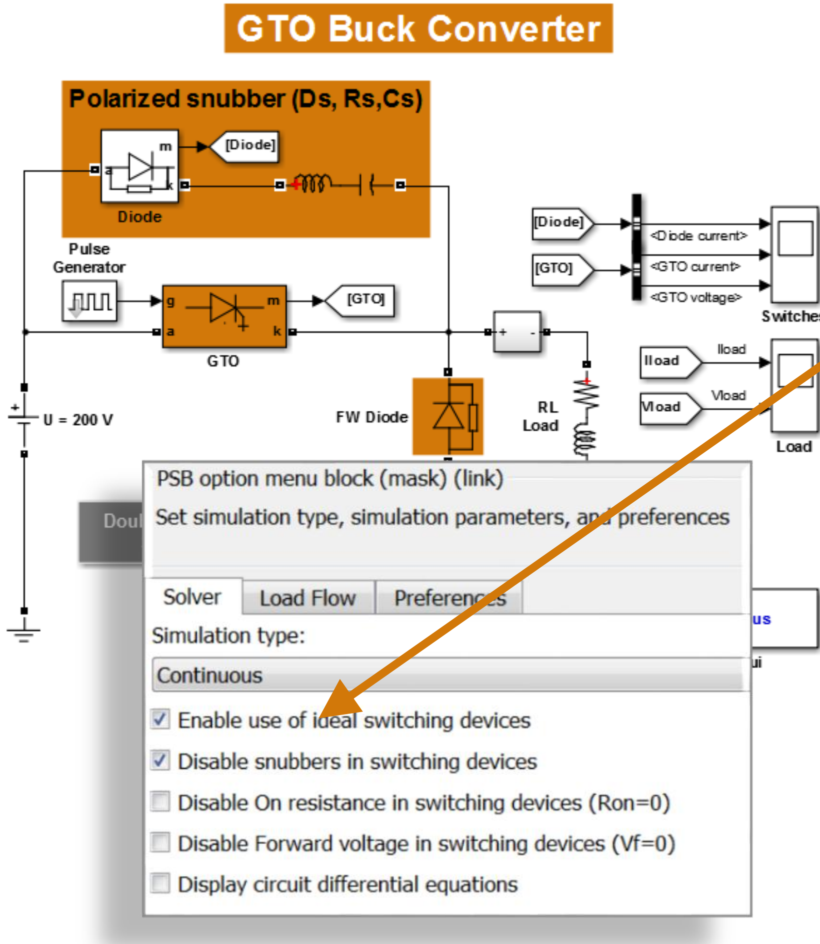
SimPowerSystems



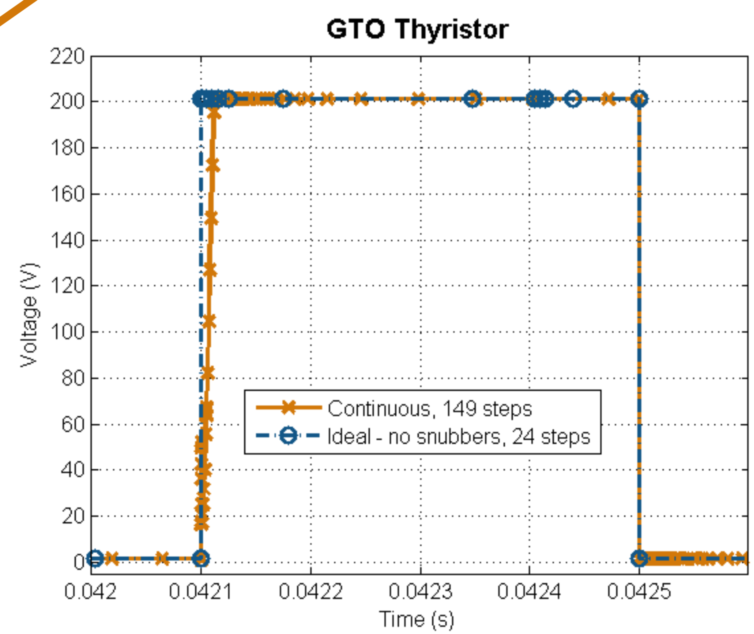



```
>> edit compare_powersupply
```

Ideal switching algorithm



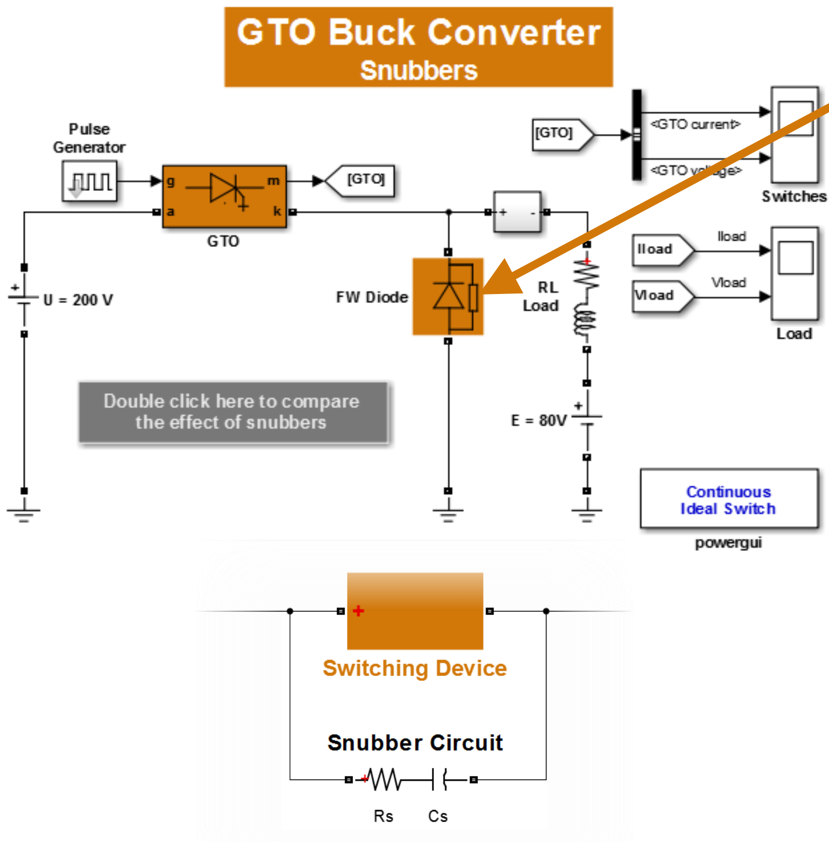
The ideal switching method can be selected from the solver configuration parameters tab when in Continuous simulation mode



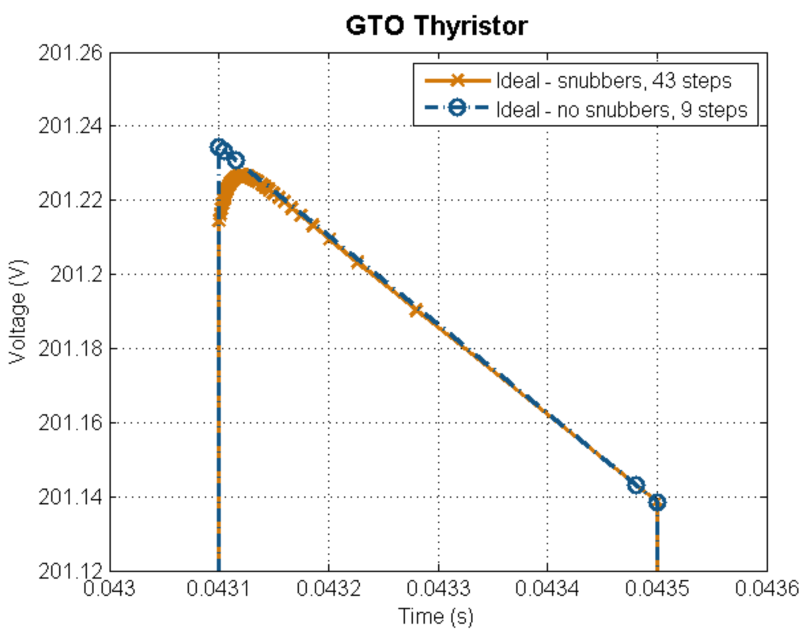

```
>> edit compare_idealswitching
```

Ideal switching algorithm

Effect of using snubber circuits



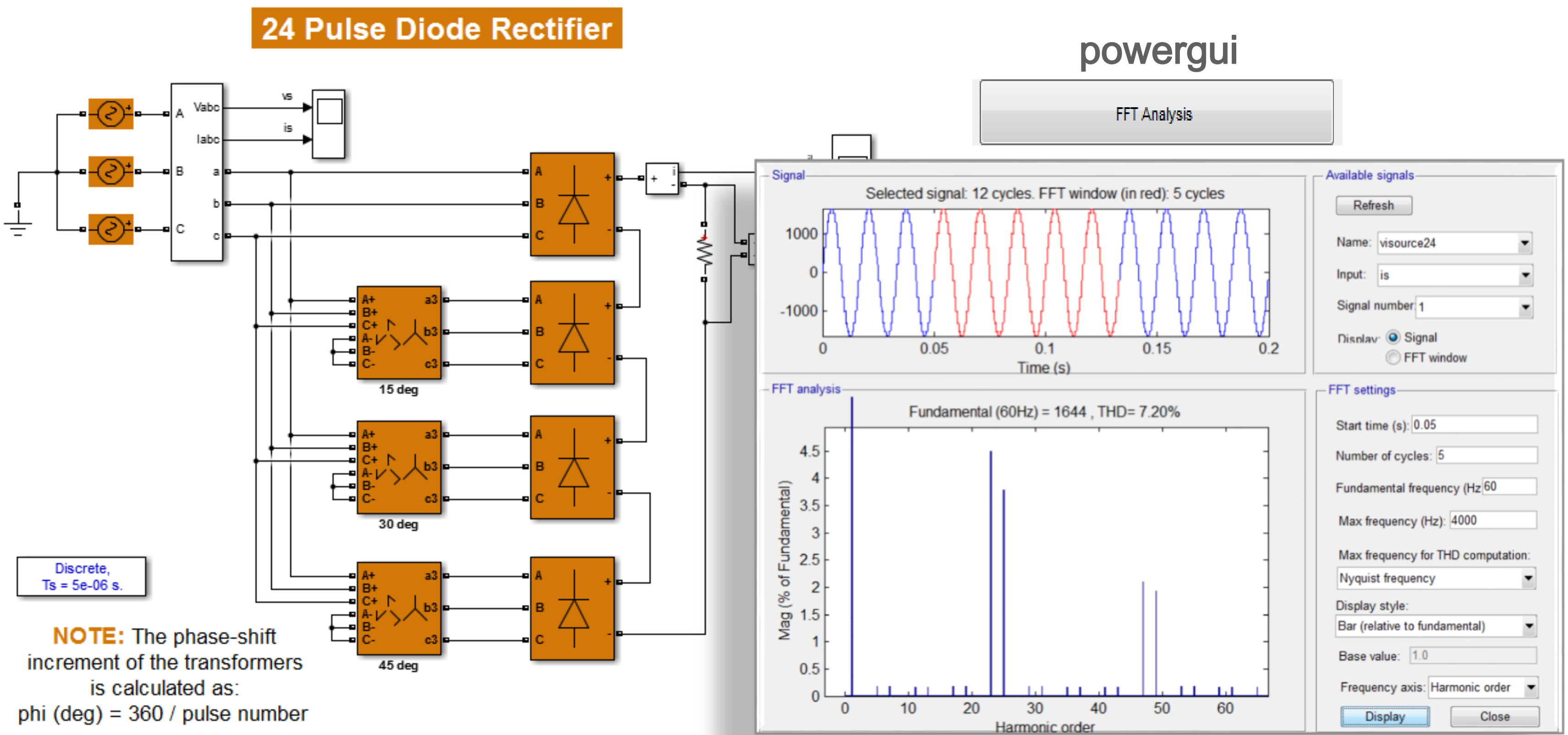
Snubbers provide an alternate path for discharging inductive elements and protecting switching devices in high power systems




```
>> edit compare_snubbers
```

Power quality and harmonic analysis

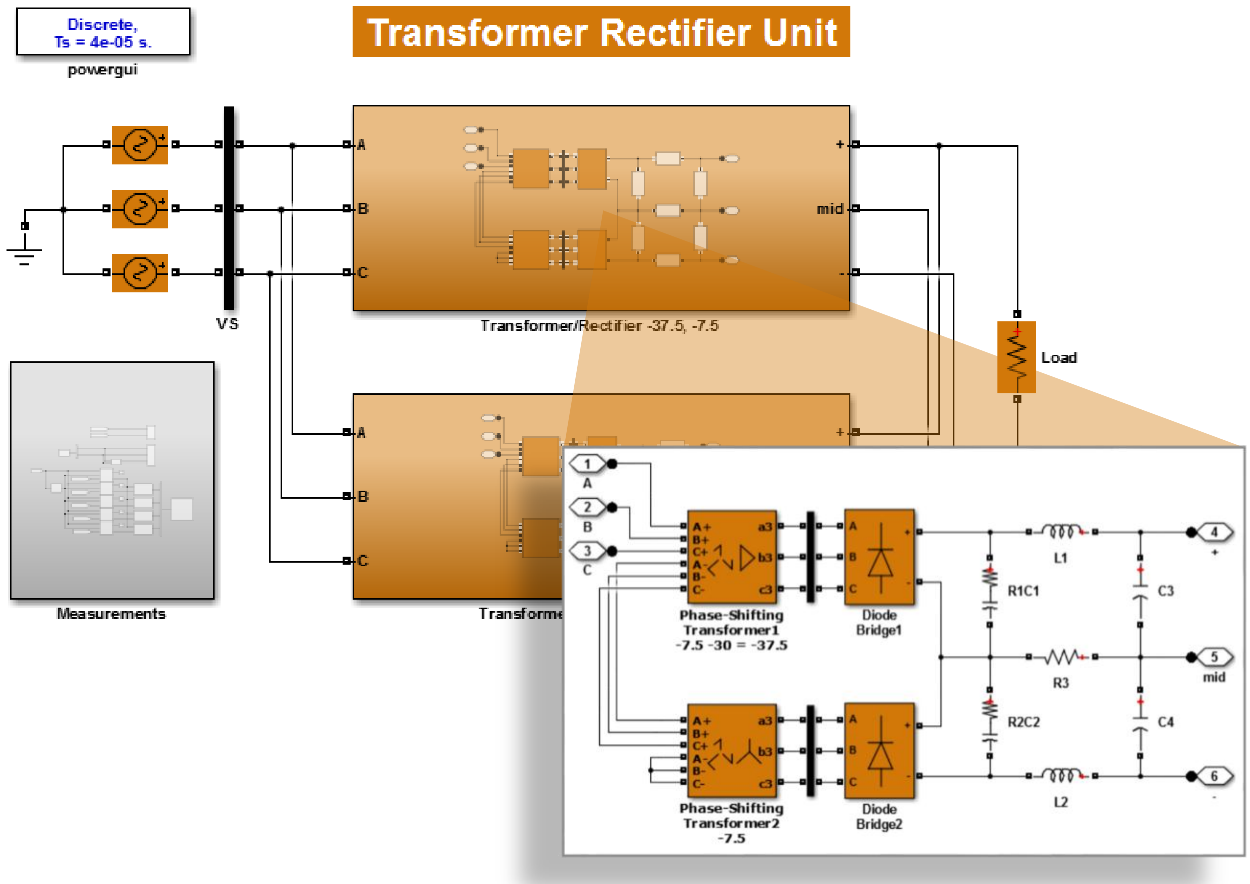
Multi-pulse rectification



NOTE: The phase-shift increment of the transformers is calculated as:
phi (deg) = 360 / pulse number

```
>> rectifier_6n12pulse
>> rectifier_24pulse
```

Power quality and harmonic analysis
Transformer rectifier units (TRUs)

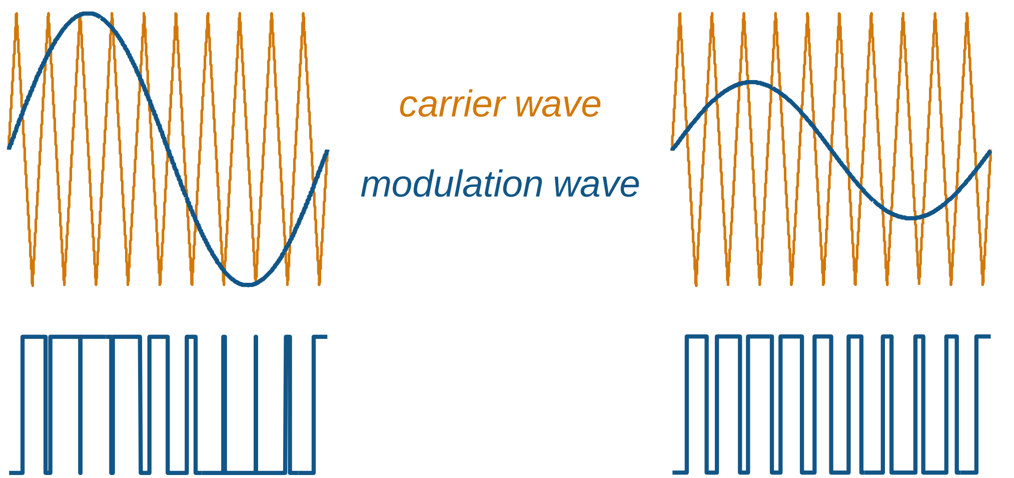



```
>> transformer_rectifier_unit
```

Power quality and harmonic analysis

Switching strategies – Pulse-Width Modulation (PWM)

- The aim of PWM is to construct a high frequency carrier wave which contains an underlying, lower frequency modulation wave
- The modulation wave is the waveform that you want the electrical system to ‘see’
- Higher order harmonics will depend on the ratio of the carrier wave frequency and the modulation wave frequency

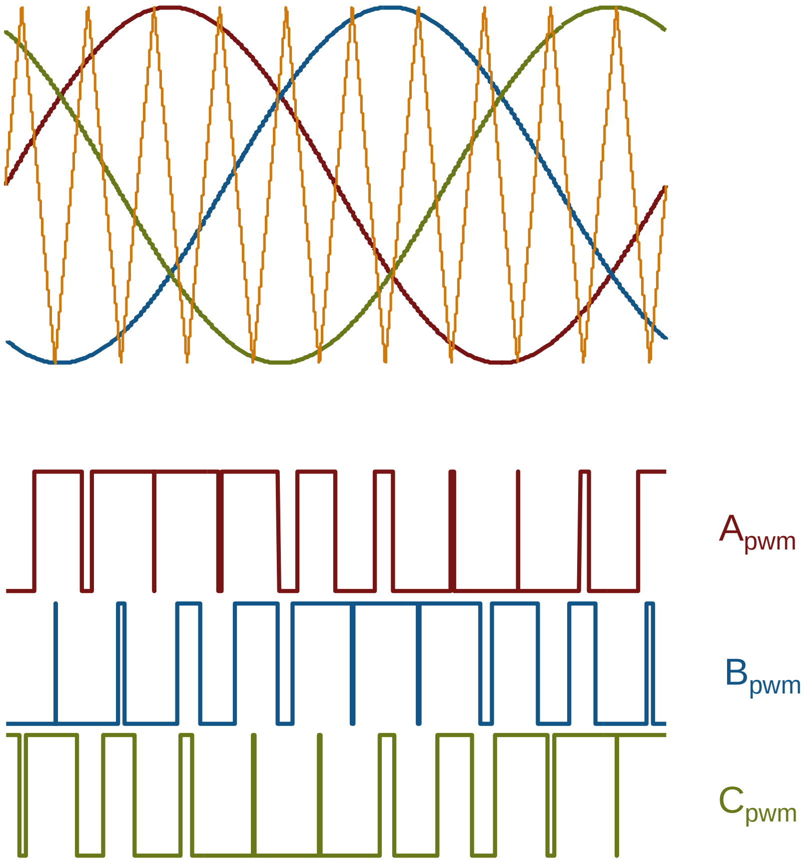


PWM Duty Cycle
for full amplitude modulation

PWM Duty Cycle
for 50% amplitude modulation

Power quality and harmonic analysis

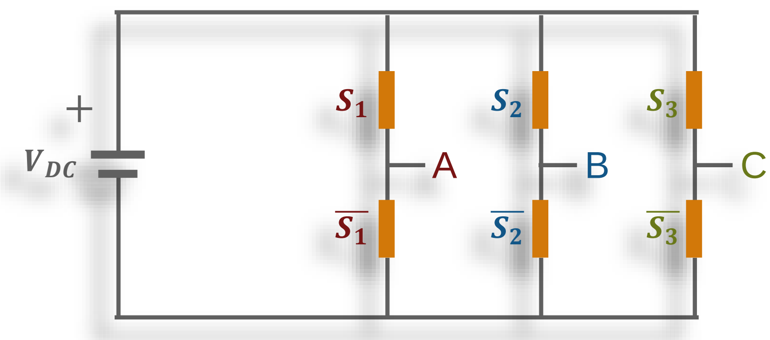
Switching strategies – Pulse-Width Modulation (PWM)



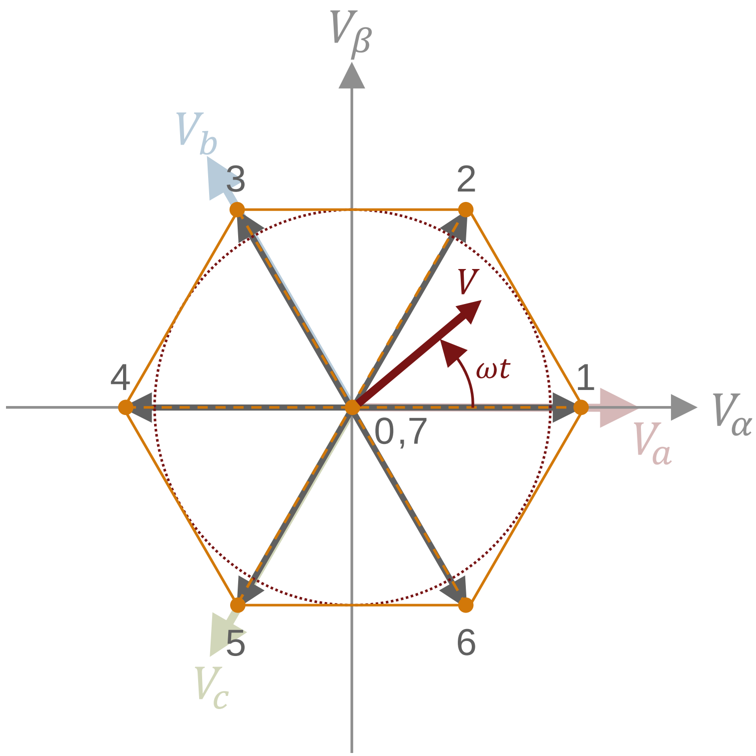
Power quality and harmonic analysis

Switching strategies – Space Vector Modulation (SVM)

- The aim of SVM is to approximate the reference voltage vector instantaneously by combining the switching states corresponding to eight basic vectors



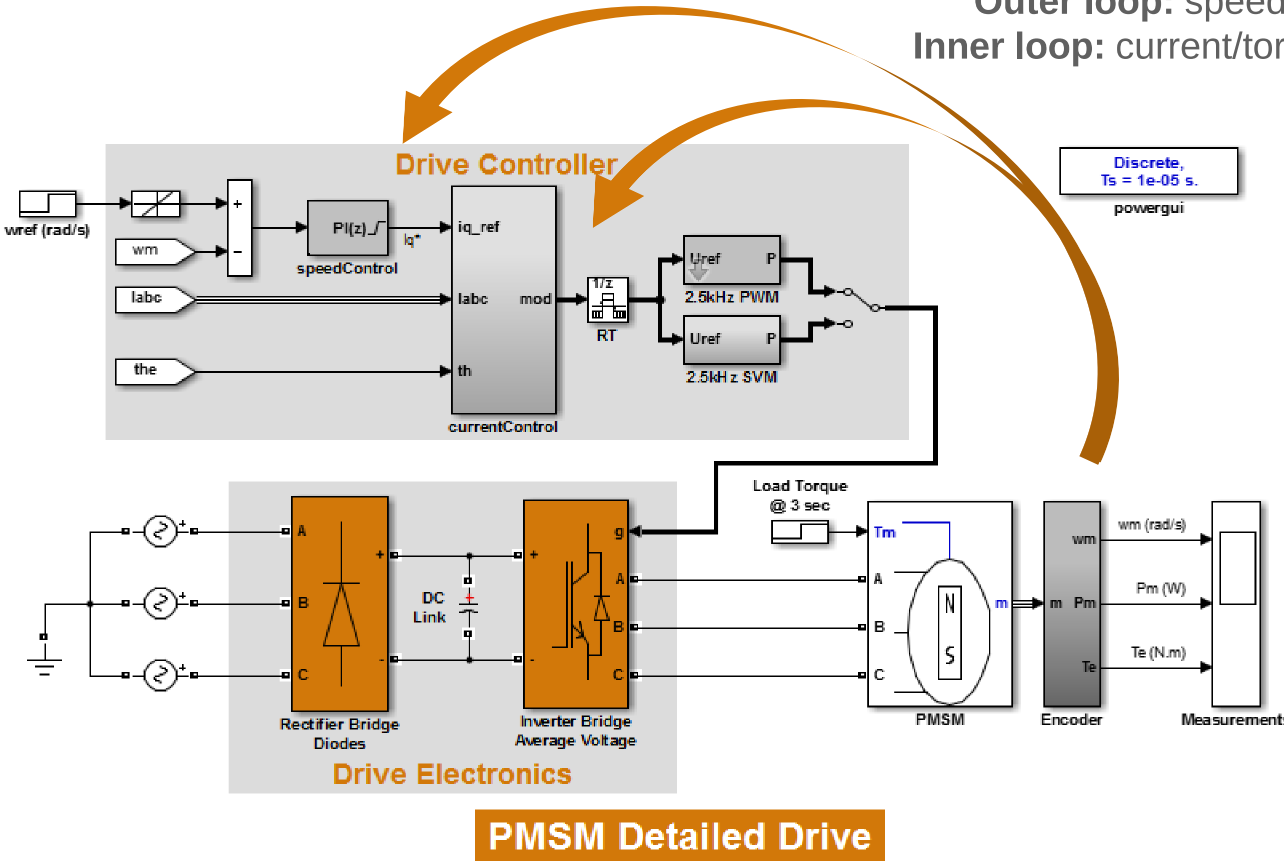
	0	1	2	3	4	5	6	7
S_1	0	1	1	0	0	0	1	1
S_2	0	0	1	1	1	0	0	1
S_3	0	0	0	0	1	1	1	1



Power quality and harmonic analysis

Comparing switching strategies

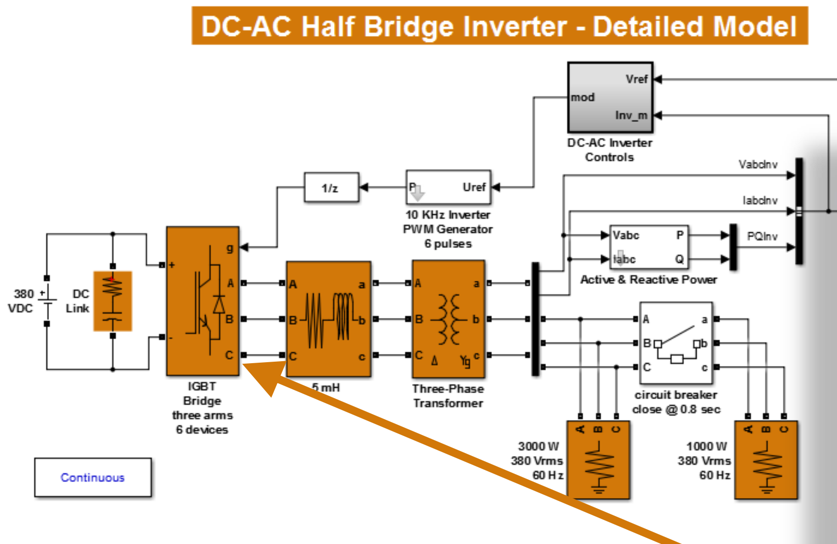
Outer loop: speed control
Inner loop: current/torque control

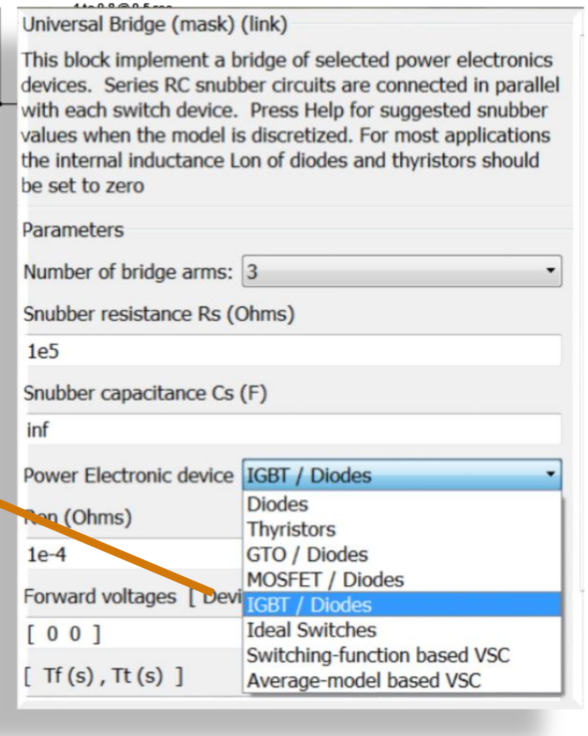



```
>> pmsm_detailed_drive
```

Control design and linearization

Detailed and average transistor bridges





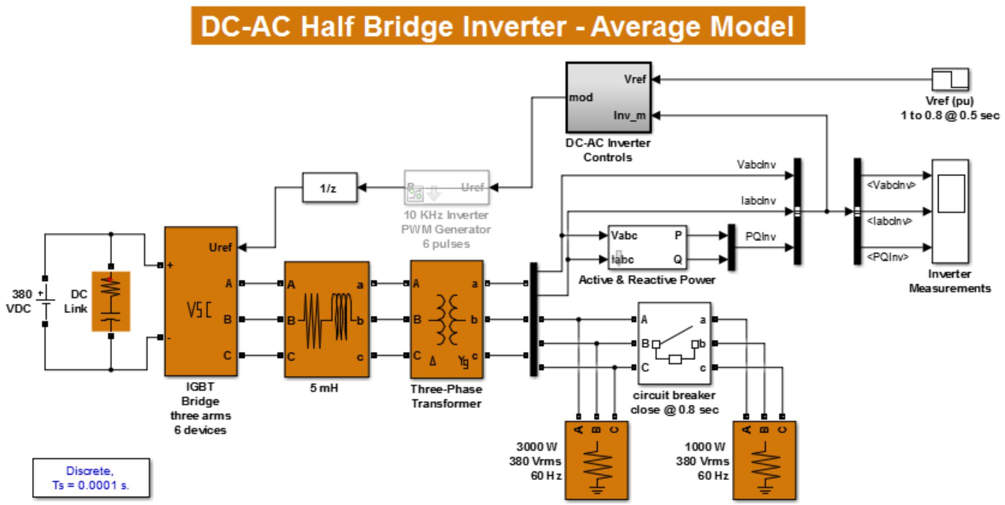
1

Switch the inverter bridge to an average voltage model and comment through the PWM signal generator

```
>> igbt_power_inverter_det
```

Control design and linearization

Detailed and average transistor bridges



2

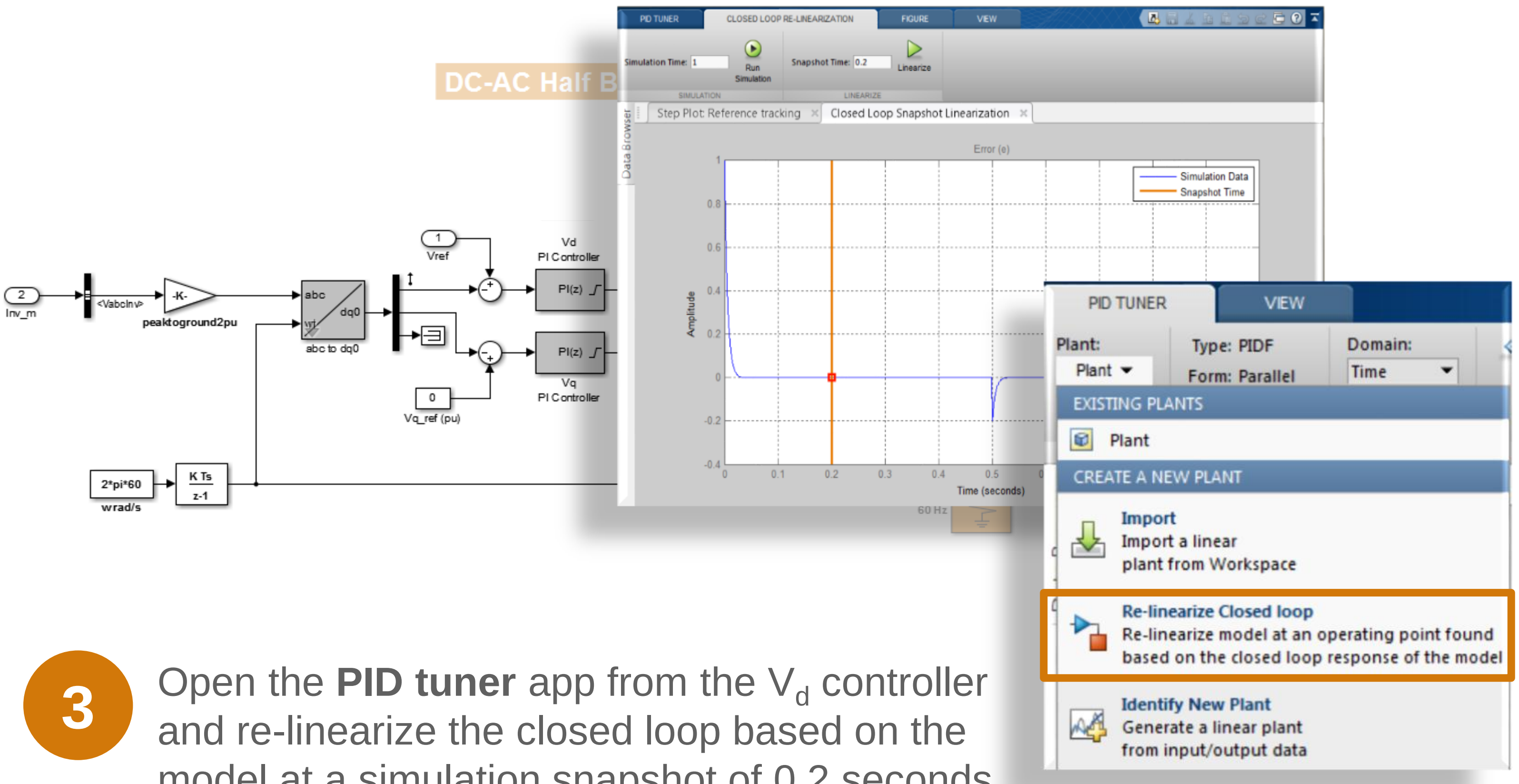
Adjust the simulation mode and sampling rates appropriately

Notice that the controller runs at a 10 kHz rate

```
>> igbt_power_inverter_avg
```

Control design and linearization

Detailed and average transistor bridges

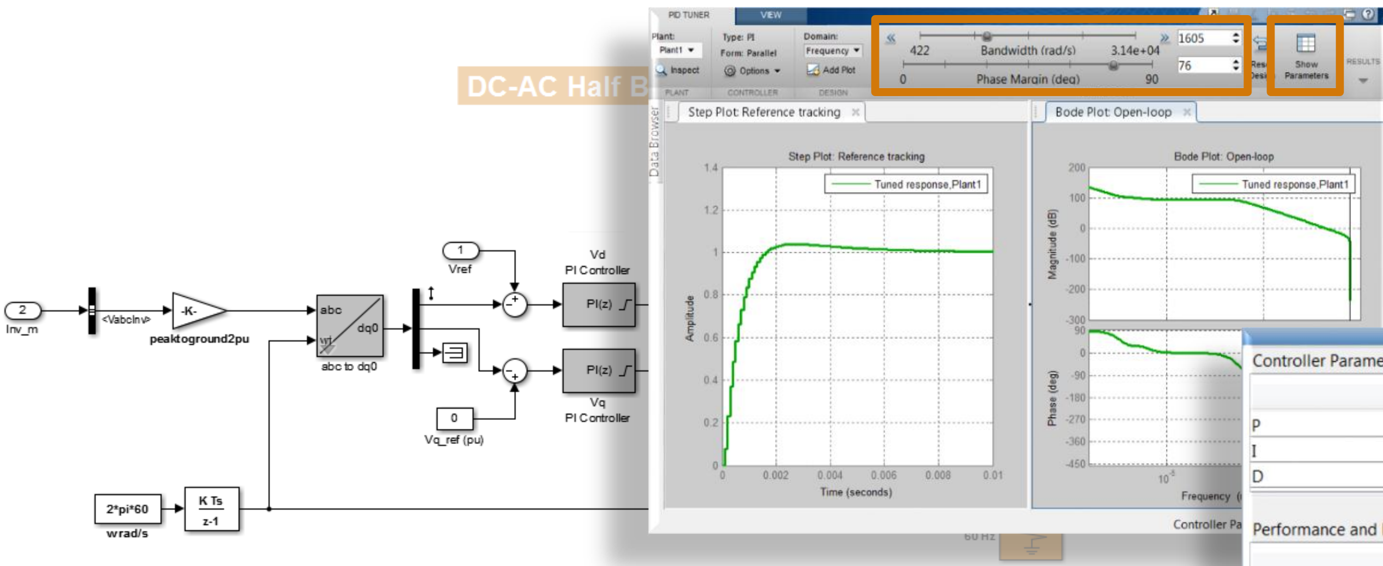


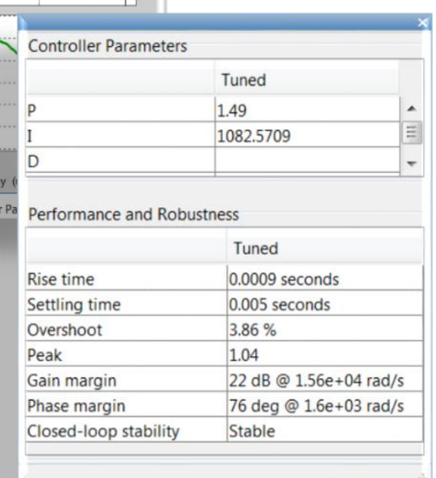
3 Open the **PID tuner** app from the V_d controller and re-linearize the closed loop based on the model at a simulation snapshot of 0.2 seconds

```
>> igbt_power_inverter_avg
```

Control design and linearization

Detailed and average transistor bridges





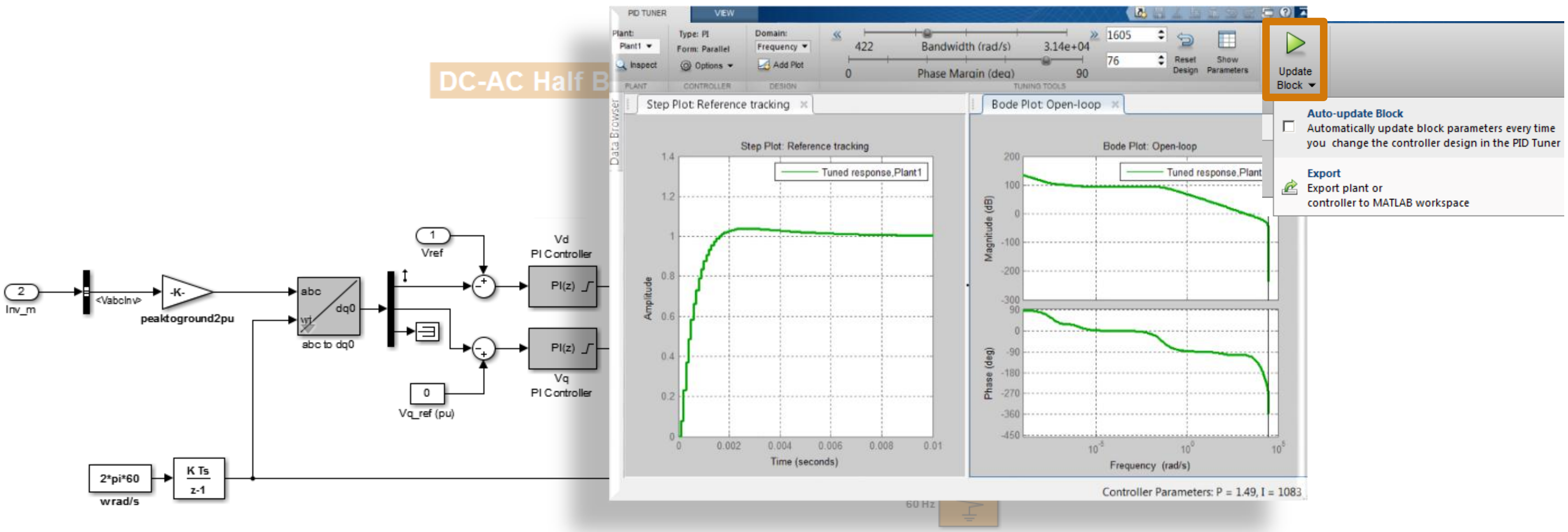
4 Use the sliders in the app to tune the V_d controller for the following performance targets:

- bandwidth > 200 Hz
- settling time < 5 milliseconds
- overshoot < 5%

```
>> igbt_power_inverter_avg
```

Control design and linearization

Detailed and average transistor bridges



5 Click **Update Block** and verify the performance of the controller against the detailed switching inverter model in simulation

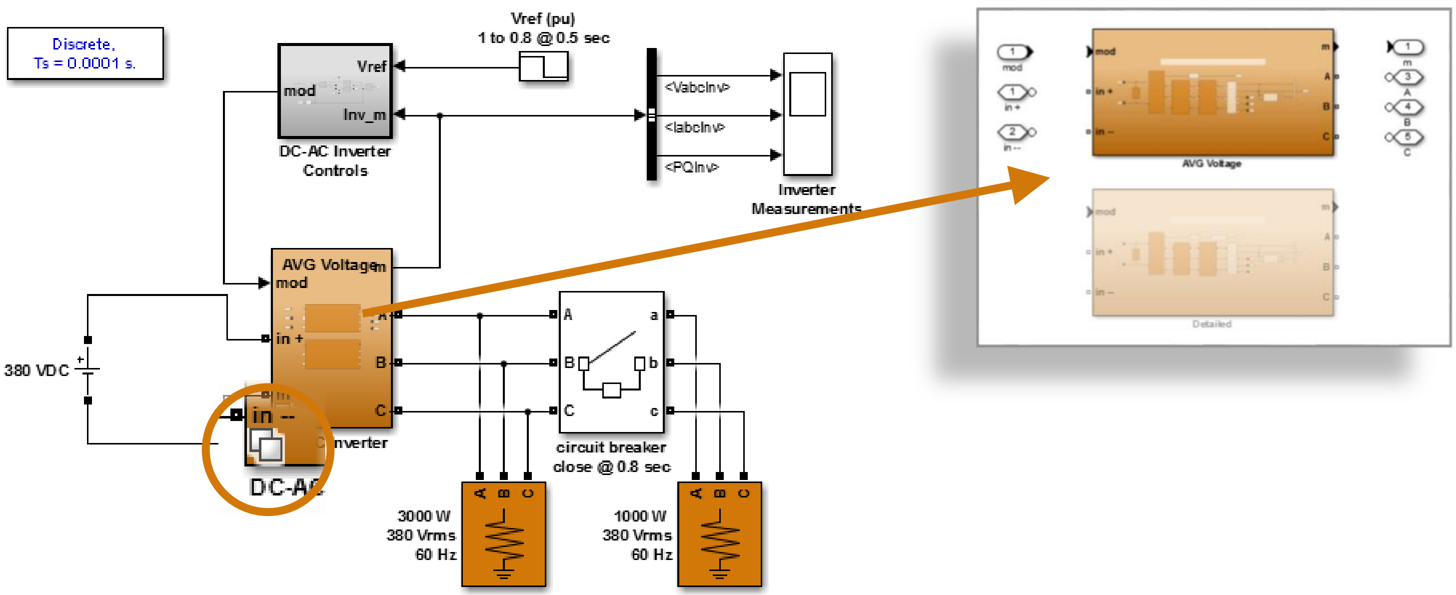
For simplicity, we will use the same newly computed PI gains for the V_q controller

```
>> igbt_power_inverter_avg
```

Control design and linearization

Detailed and average transistor bridges

6 Combine both versions of the inverter in a single model using a **Variant Subsystem**
Subsystem & Model Reference → Convert Subsystem to…



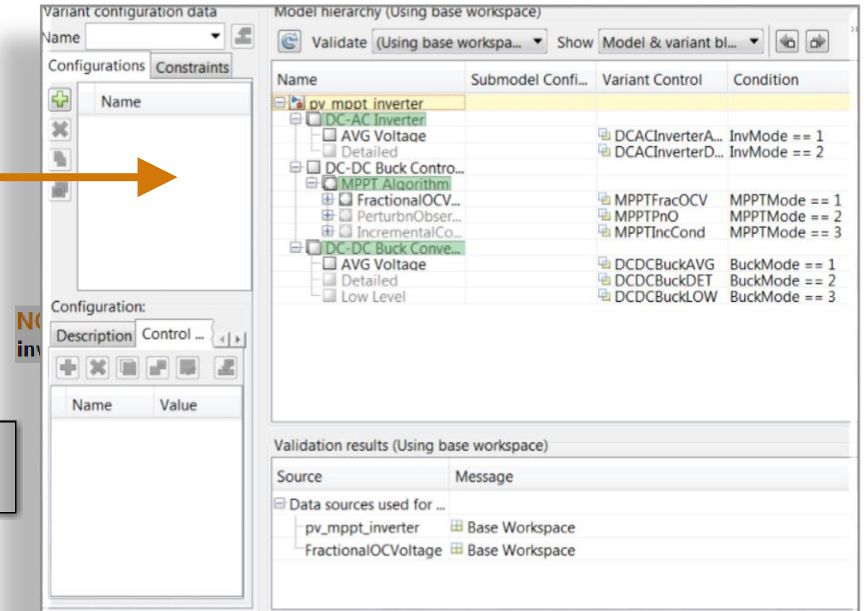

```
>> igbt_power_inverter
```

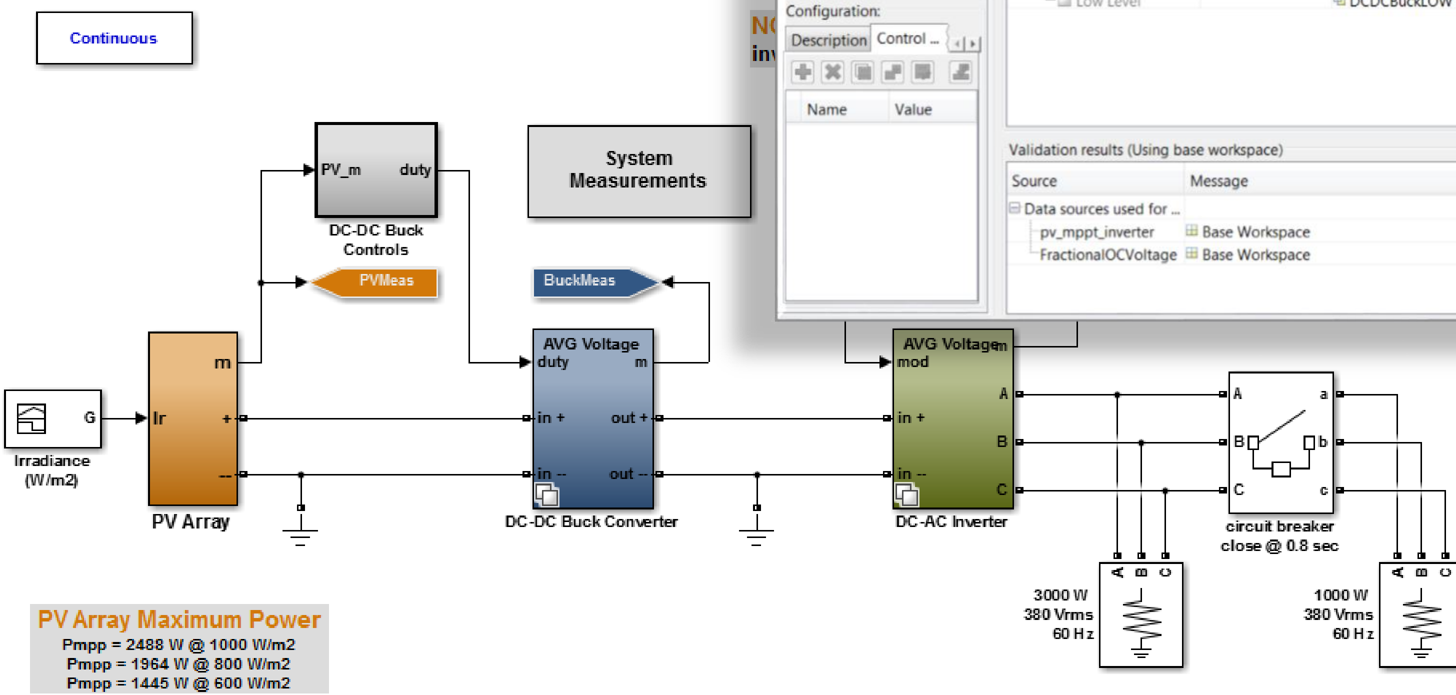
Control design and linearization

Using subsystem variants

Use the **Variant Manager** to set multiple variant configurations for simulation and testing

Variant → Open in Variant Manager

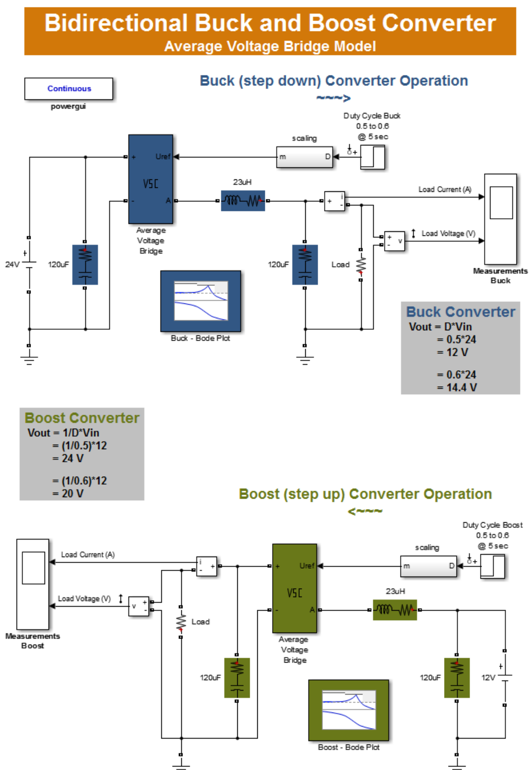





```
>> pv_mppt_inverter
```

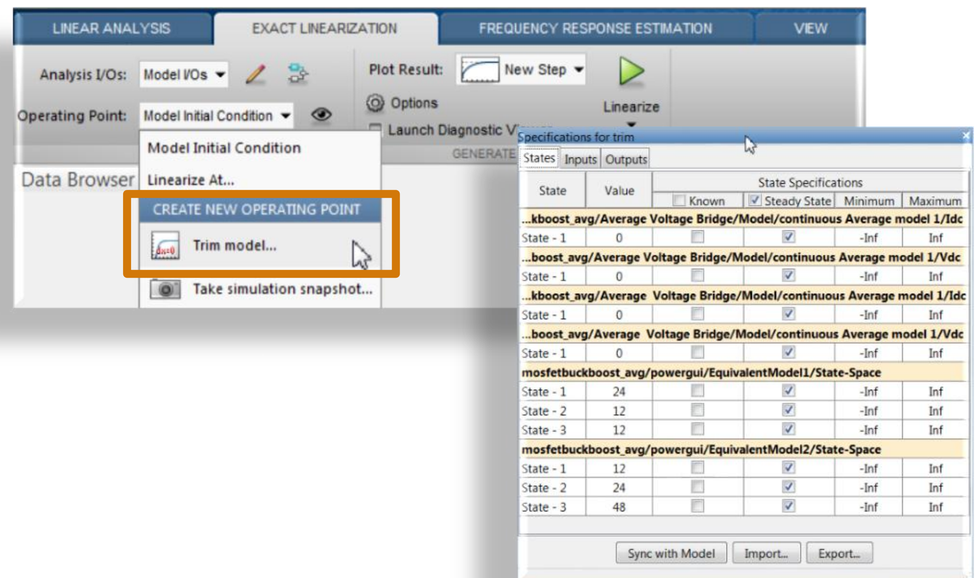
Control design and linearization

Initialization and trimming



1 Use the **Trim model** option in the Linear Analysis tool

Analysis → Control Design → Linear Analysis…



2 Check the **Specifications** for the output voltages as **known** and trim the model

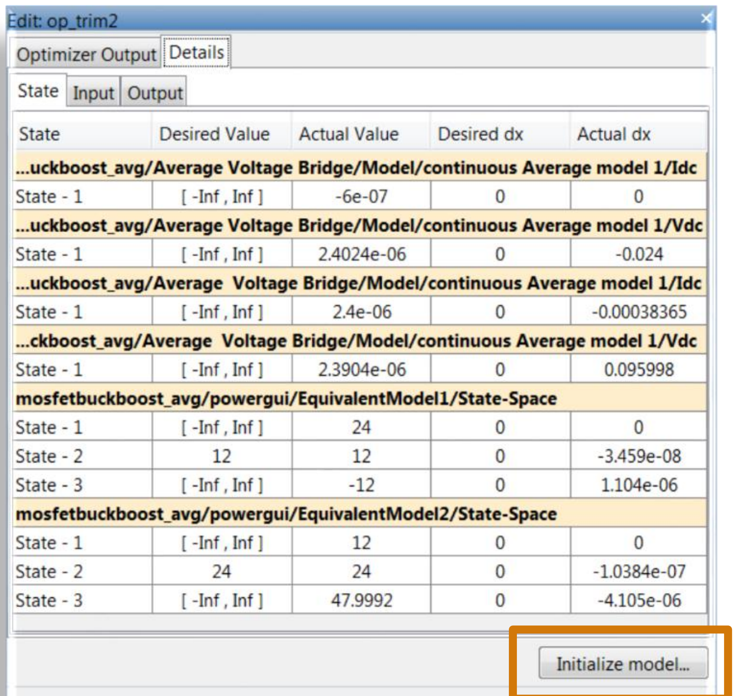
Change optimization method to **nonlinear least squares**

```
>> mosfetbuckboost_avg
```

Control design and linearization

Initialization and trimming

3 Verify trimmed operating condition and initialize model



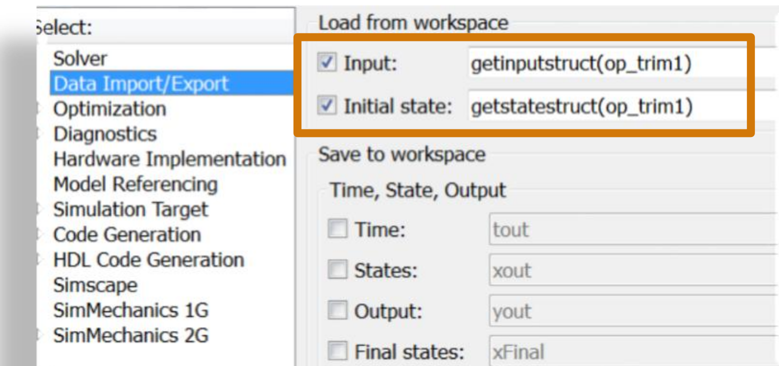
Edit: op_trim2

Optimizer Output | Details

State | Input | Output

State	Desired Value	Actual Value	Desired dx	Actual dx
...uckboost_avg/Average Voltage Bridge/Model/continuous Average model 1/Idc				
State - 1	[-Inf , Inf]	-6e-07	0	0
...uckboost_avg/Average Voltage Bridge/Model/continuous Average model 1/Vdc				
State - 1	[-Inf , Inf]	2.4024e-06	0	-0.024
...uckboost_avg/Average Voltage Bridge/Model/continuous Average model 1/Idc				
State - 1	[-Inf , Inf]	2.4e-06	0	-0.00038365
...ckboost_avg/Average Voltage Bridge/Model/continuous Average model 1/Vdc				
State - 1	[-Inf , Inf]	2.3904e-06	0	0.095998
mosfetbuckboost_avg/powergui/EquivalentModel1/State-Space				
State - 1	[-Inf , Inf]	24	0	0
State - 2	12	12	0	-3.459e-08
State - 3	[-Inf , Inf]	-12	0	1.104e-06
mosfetbuckboost_avg/powergui/EquivalentModel2/State-Space				
State - 1	[-Inf , Inf]	12	0	0
State - 2	24	24	0	-1.0384e-07
State - 3	[-Inf , Inf]	47.9992	0	-4.105e-06

Initialize model...

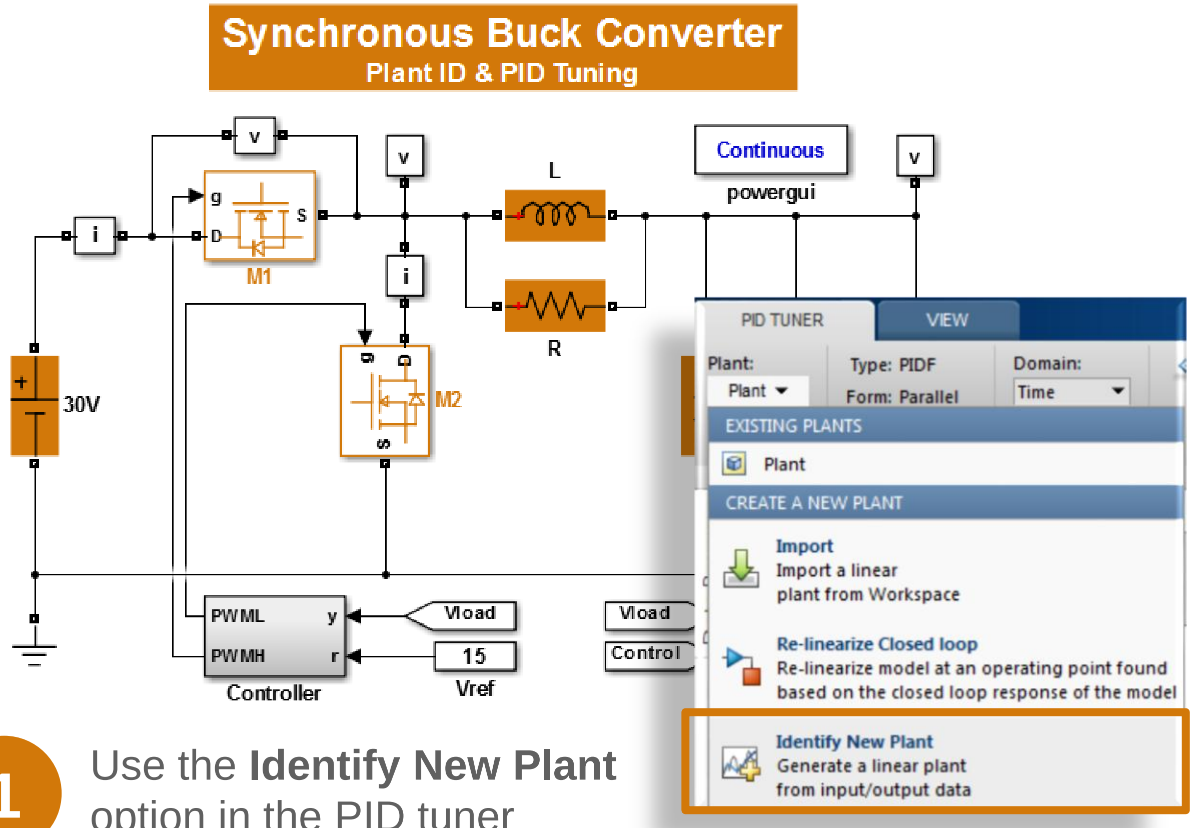


4 Run simulation with the computed operating condition object as initial states

```
>> mosfetbuckboost_avg
```

Control design and linearization

Plant identification from simulation data

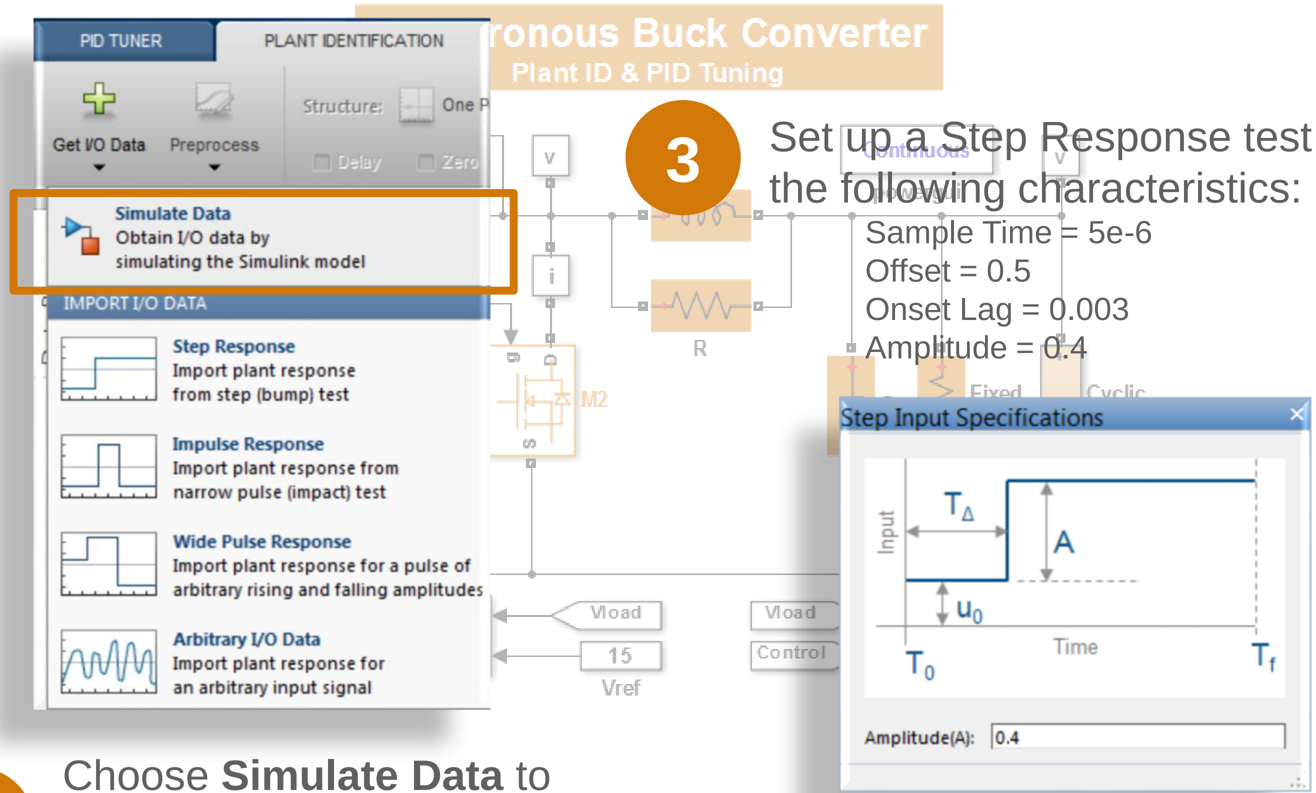


1 Use the **Identify New Plant** option in the PID tuner

```
>> sps_powersupply_sid
```

Control design and linearization

Plant identification from simulation data



3 Set up a Step Response test with the following characteristics:

- Sample Time = 5e-6
- Offset = 0.5
- Onset Lag = 0.003
- Amplitude = 0.4

2 Choose **Simulate Data** to obtain I/O results from the simulation model

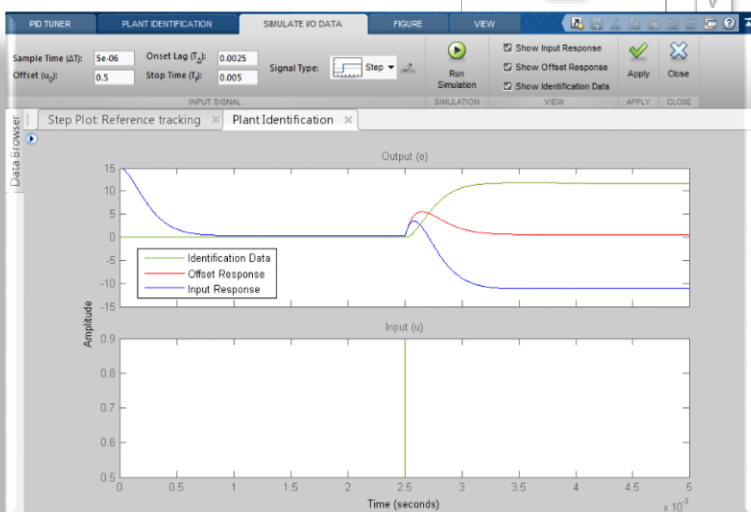
```
>> sps_powersupply_sid
```

Control design and linearization

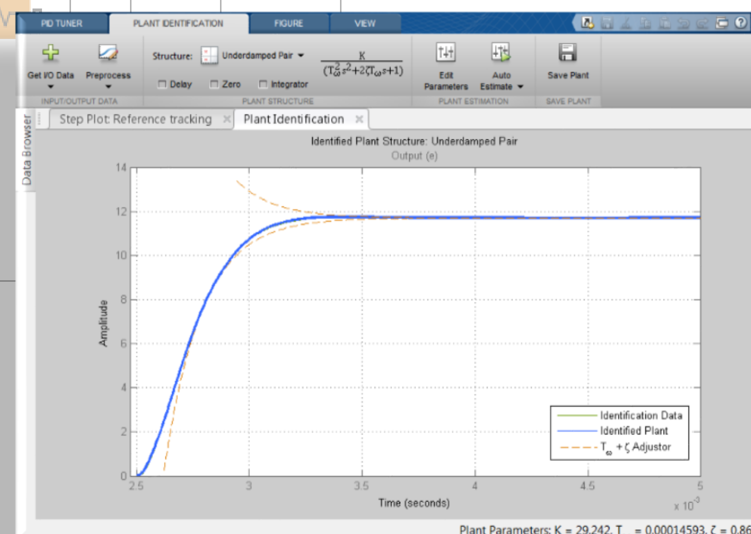
Plant identification from simulation data

Synchronous Buck Converter

4 Run the simulation to obtain the identification data



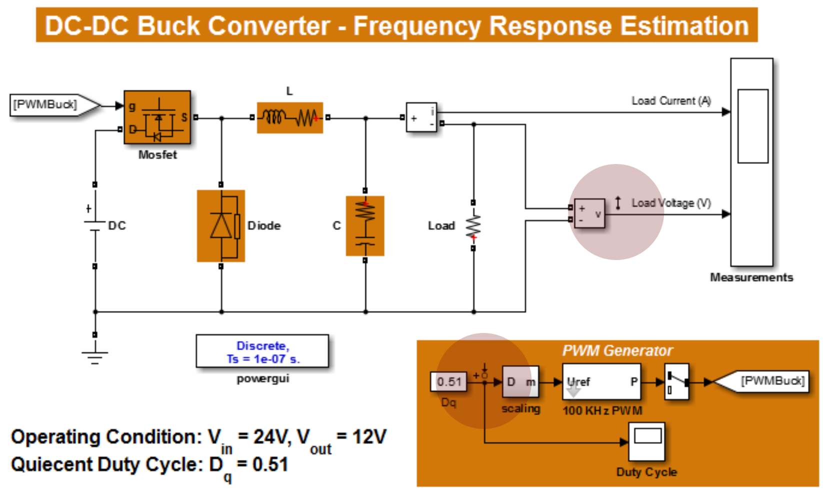
5 Select the model structure and use the interactive handles to match the model to the data




```
>> sps_powersupply_sid
```

Control design and linearization

Frequency response estimation



1 Open the **Frequency Response Estimation** tool

Analysis → Control Design → Frequency…

2 Create a new input **Fixed Sample Time Sinestream** with **T_s = 1e-7**

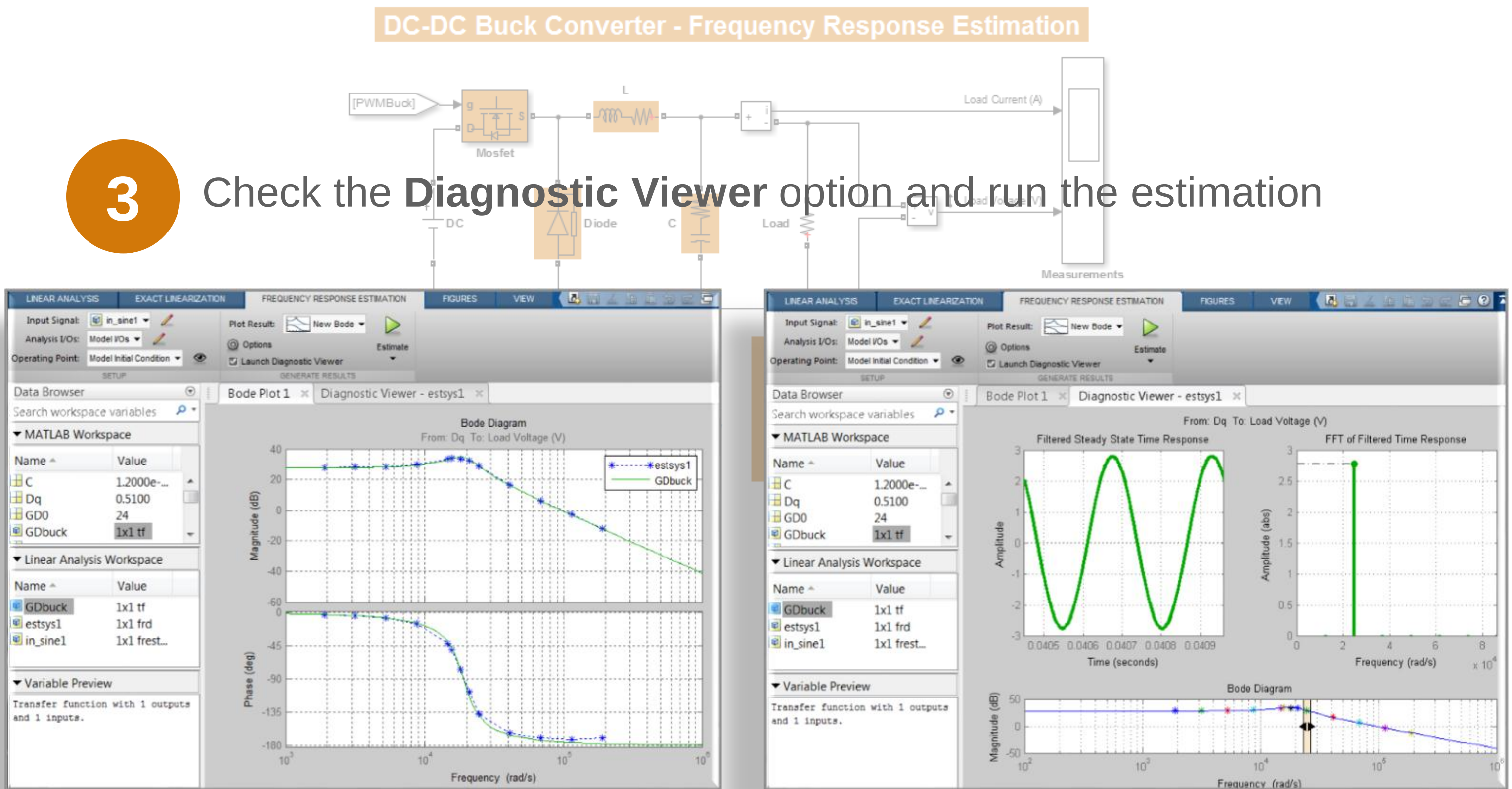
Use the pre-computed analytic transfer function of the converter **GDBuck** as a reference for selecting the test frequencies and increase the amplitude of the test sine waves to **1e-1**

```
>> mosfetbuck_fre
```

Control design and linearization

Frequency response estimation

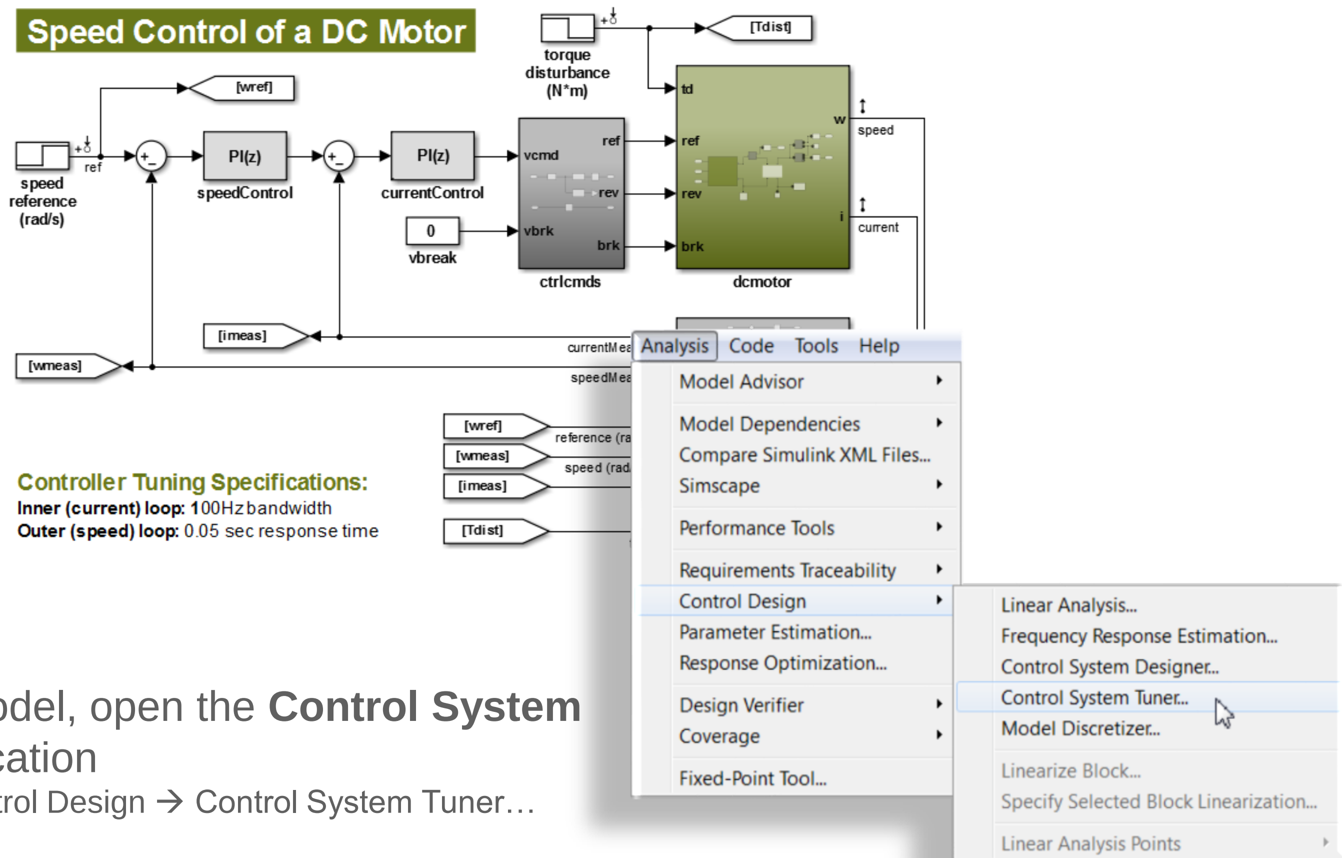
3 Check the **Diagnostic Viewer** option and run the estimation




```
>> mosfetbuck_fre
```

Control design and linearization

Multi-input, multi-output control tuning



1 From the model, open the **Control System Tuner** application

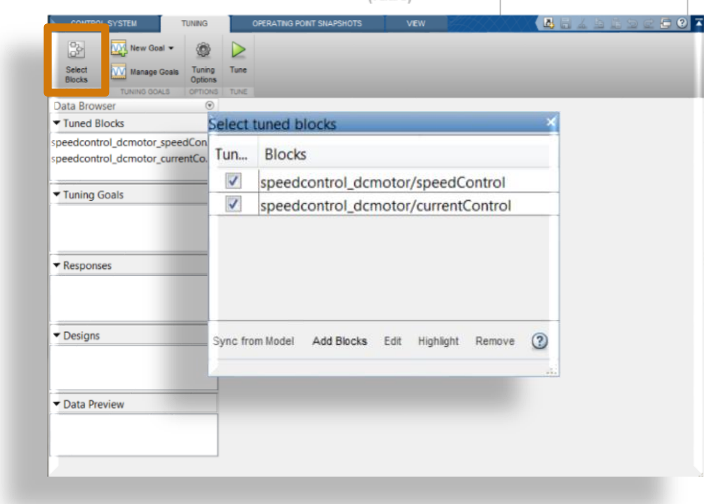
Analysis → Control Design → Control System Tuner…

```
>> speedcontrol_dcmotor
```

Control design and linearization

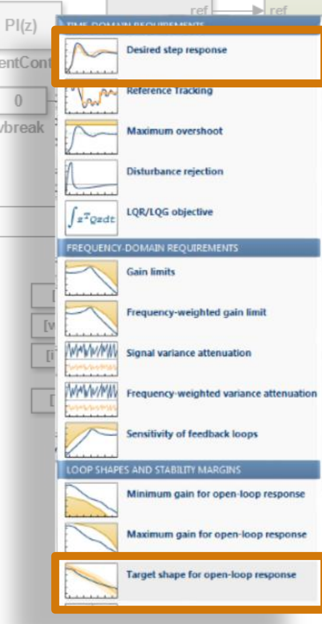
Multi-input, multi-output control tuning

2 In the **Control System** tab, capture a new operating point by taking a simulation snapshot at 10 seconds



3 In the **Tuning** tab, select the controller blocks to be tuned

4 In the **Tuning** tab, select the two new desired tuning goals



Goal #1: Desired step response
Input: speed reference
Output: speed measurement
Time constant = 0.05 seconds

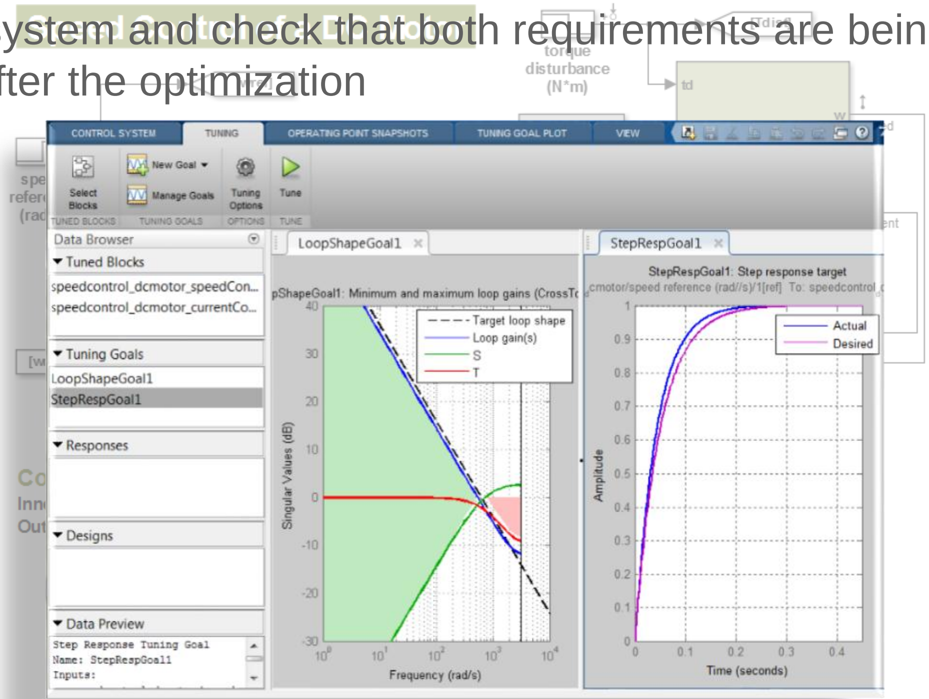
Goal #2: Target shape for open loop response
Response at current measurement
Open loop at the speed measurement
Crossover frequency = 100*2*pi rad/s

```
>> speedcontrol_dcmotor
```

Control design and linearization

Multi-input, multi-output control tuning

5 Tune the system and check that both requirements are being adequately satisfied after the optimization



6 In the **Control System** tab, select **Update Blocks** and verify the performance of the controller against the nonlinear model in simulation

```
>> speedcontrol_dcmotor
```