

Introduction to Motor Control Blockset (R2020a)

Dakai Hu, Ph.D
MathWorks Sr. Application Engineer
5/27/2020

Brushless motors are everywhere



Why Simulink for motor control?

- Verify control algorithm with desktop simulation
- Generate compact and fast code from models
- Minimize development time using reference examples

Customers routinely report 50% faster time to market

Motor Control Blockset simplifies the workflow

- Control blocks optimized for code generation
- Sensor decoders and observers
- Motor parameter estimation
- Controller autotuning
- Reference examples

Motor Control Blockset
Design and implement motor control algorithms

[Download a free trial](#)

Motor Control Blockset™ provides reference examples and blocks for developing field-oriented control algorithms for brushless motors. The examples show how to configure a controller model to generate compact and fast C code for any target microcontroller (with Embedded Coder™). You can also use the reference examples to generate algorithmic C code and driver code for specific motor control kits.

The blockset includes Park and Clarke transforms, sliding mode and flux observers, a space-vector generator, and other components for creating speed and torque controllers. You can automatically tune controller gains based on specified bandwidth and phase margins for current and speed loops (with Simulink Control Design™).

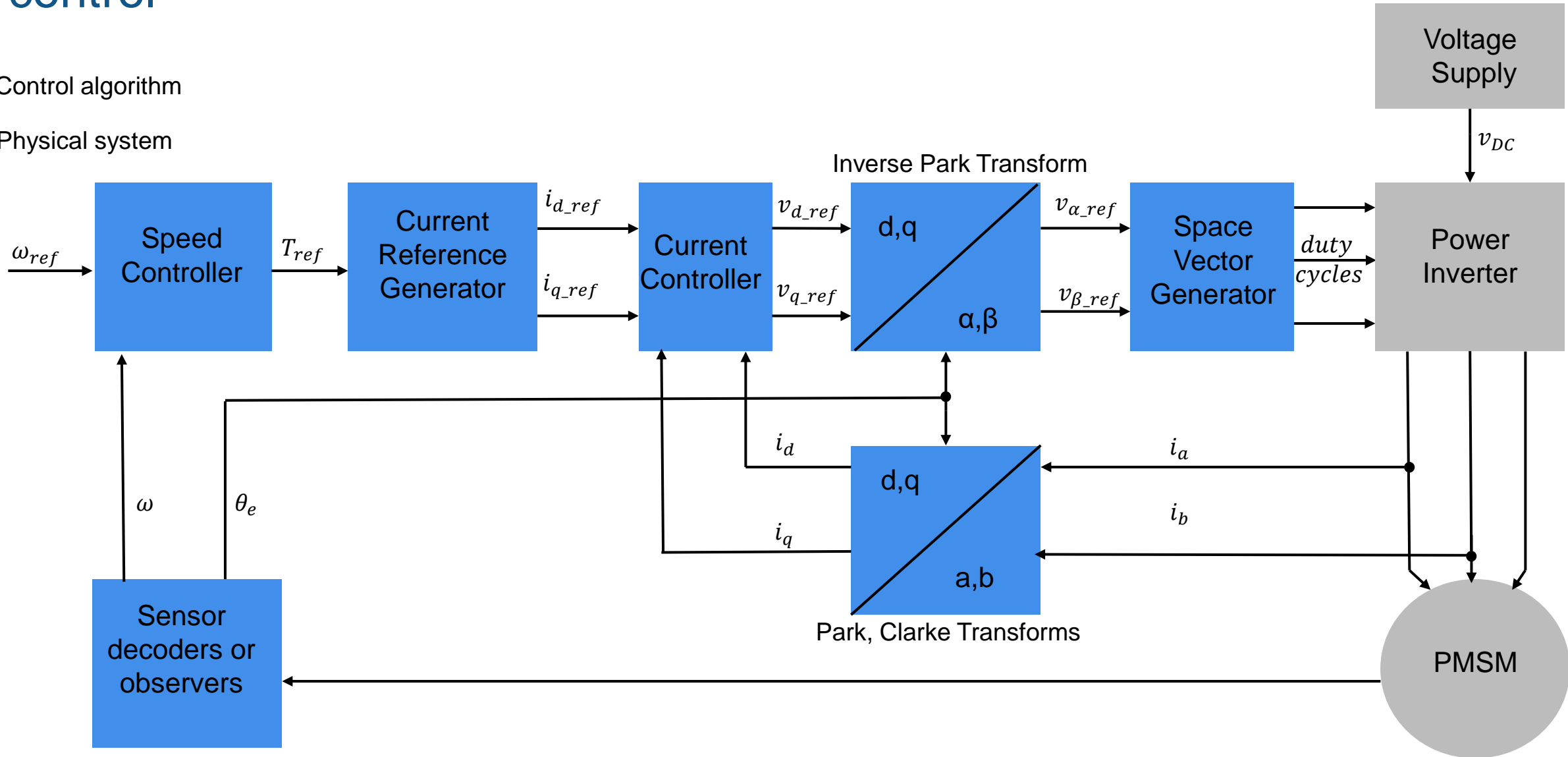
The blockset lets you create an accurate motor model by providing tools for collecting data directly from hardware and calculating motor parameters. You can use the parameterized motor model to test your control algorithm in closed-loop simulations.

Get Started:

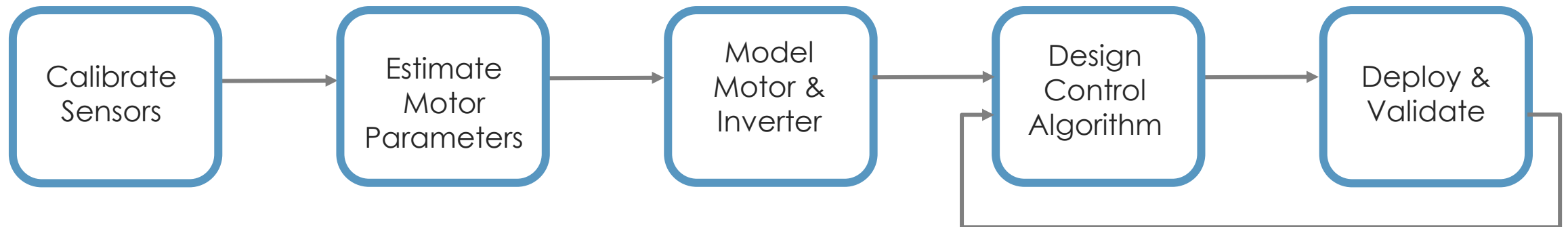
Reference Examples	Latest Features
Motor Control Algorithms	Documentation and Resources
Sensor Decoders and Observers	Try or Buy
Controller Autotuning	
Motor Parameter Estimation	
Motor Models	

Brushless motors require complex algorithms – field-oriented control

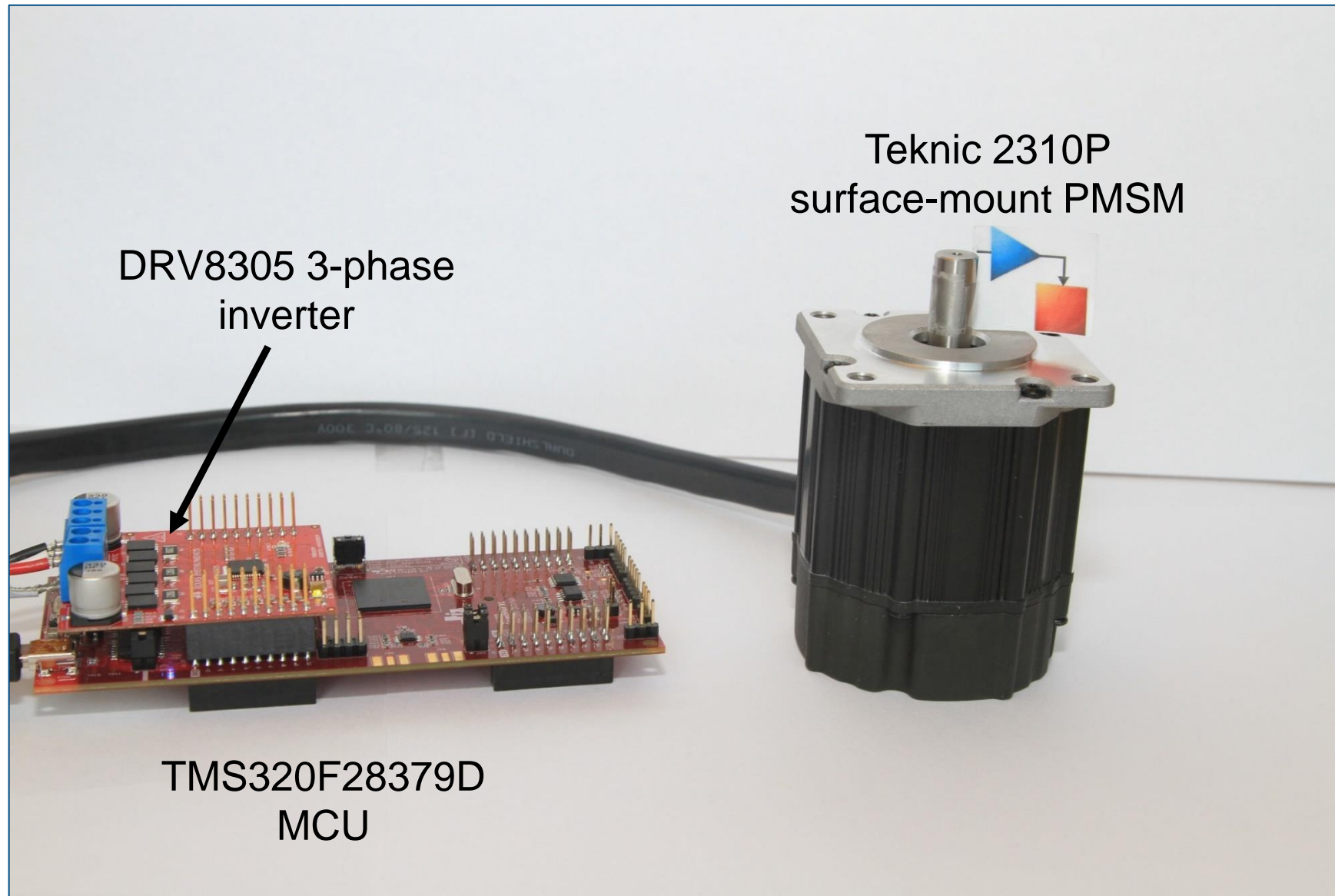
Control algorithm
 Physical system



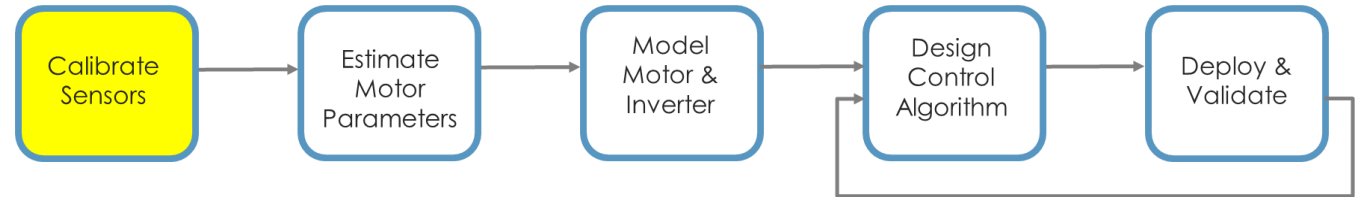
Workflow for implementing field-oriented control



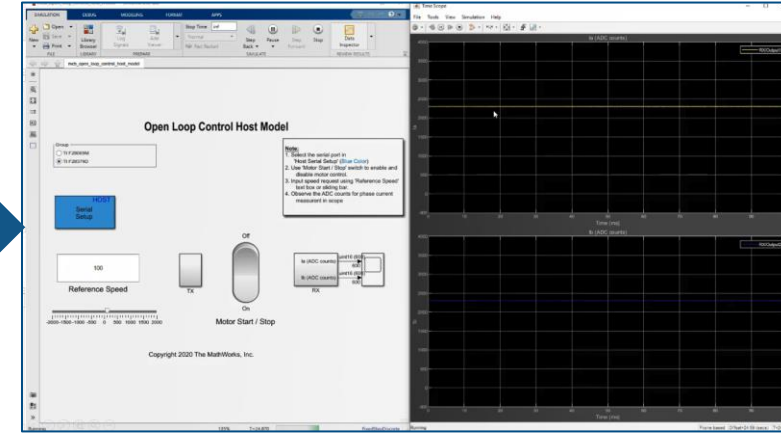
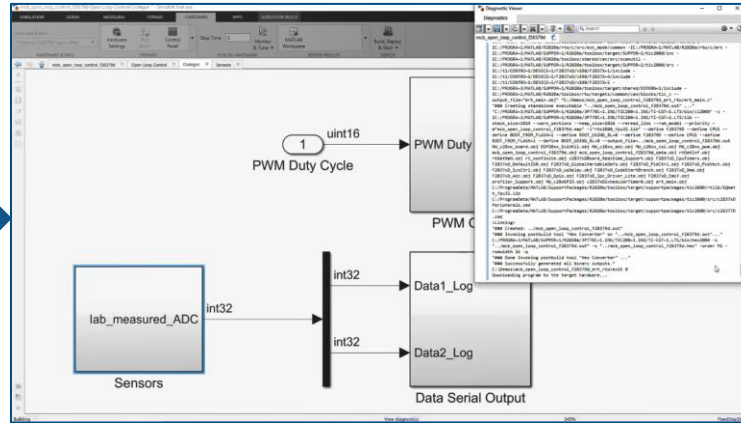
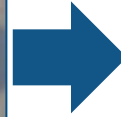
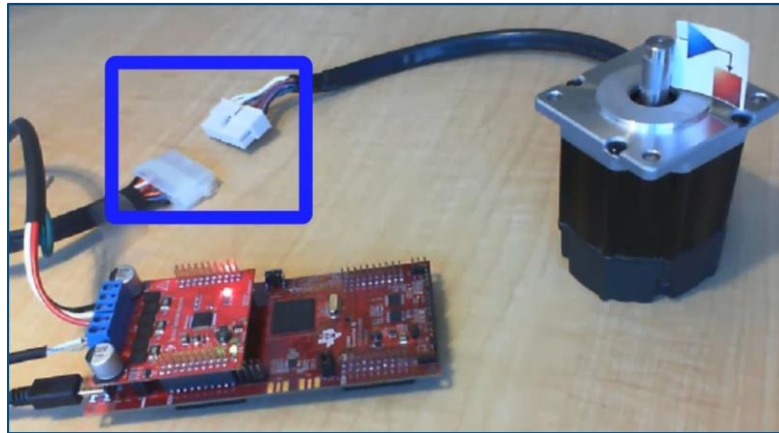
We will use Texas Instruments motor control kit



Sensor calibration

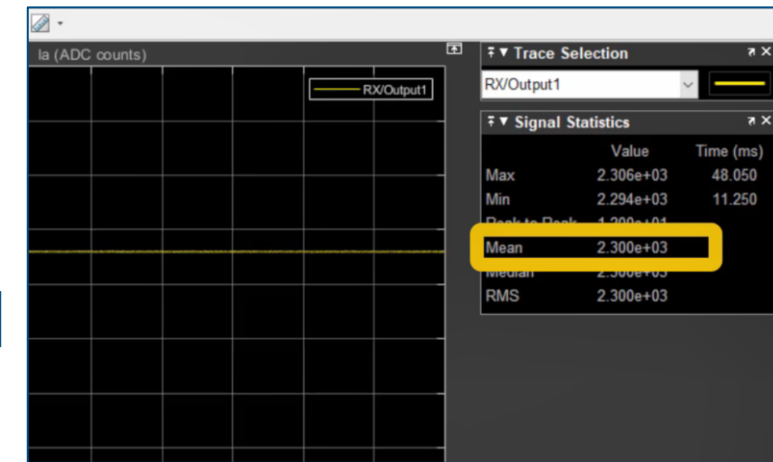


- Calibrate ADC offsets

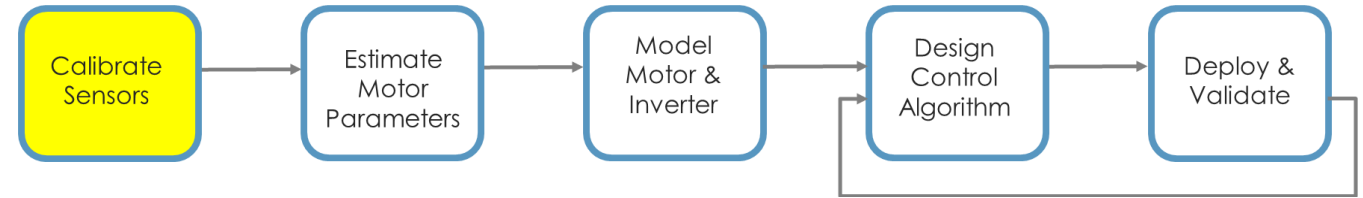


```

case 'BoostXL-DRV8305'
    inverter.model = 'BoostXL-DRV8305'; % Model
    inverter.sn = 'INV_XXXX'; % Serial Number
    inverter.V_dc = 24; %V // DC Bus Voltage
    inverter.I_max = 19.3; %Amps // Max Current
    inverter.I_trip = 10; %Amps // Trip Current
    inverter.Rds_on = 2e-3; %Ohms // MOSFET Rds(on)
    inverter.Rshunt = 0.007; %Ohms // Shunt Resistor
    inverter.MaxADCCnt = 4095; %Counts // Max ADC Counts
    inverter.CtSensAOffset = 2300; %Counts // ADC Offset A
    inverter.CtSensBOffset = 2303; %Counts // ADC Offset B
    inverter.ADCGain = 1; % // ADC Gain
  
```



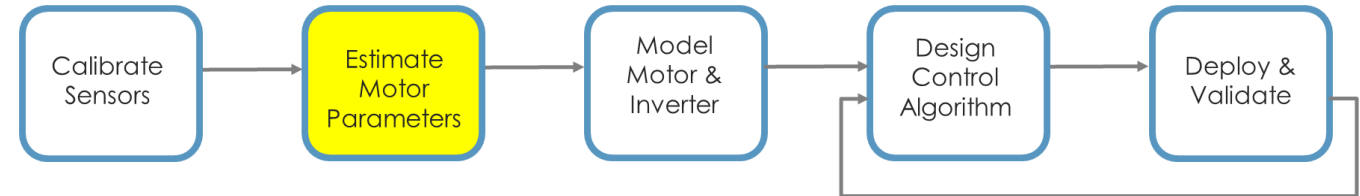
Sensor calibration



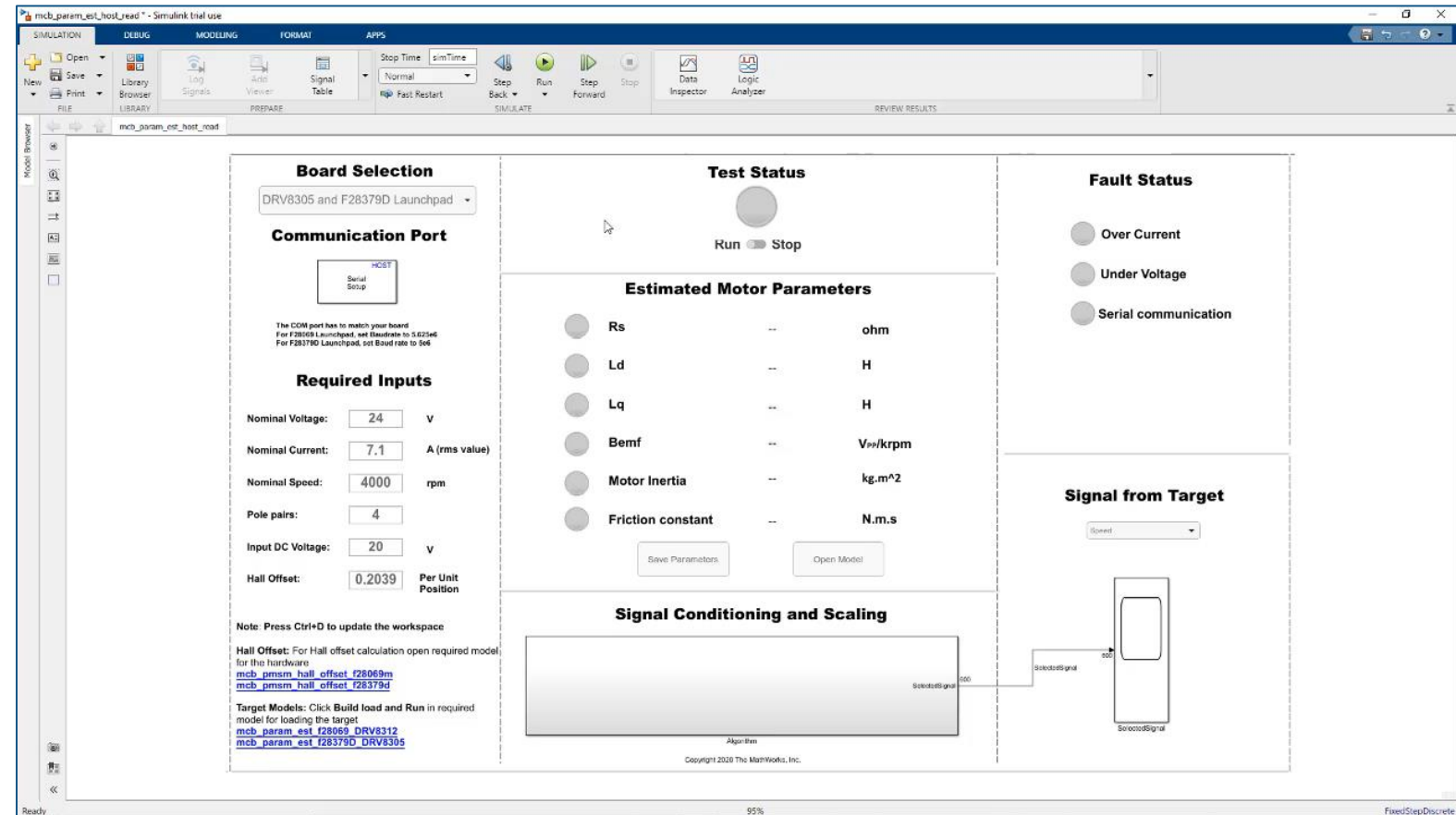
- Calibrate ADC offsets
- Calibrate position sensor offset

Copyright 2020 The MathWorks, Inc.

Parameter estimation



- Instrumented tests running on the target
- Host model to start and control parameter estimation



Bonus: you can use other techniques to parameterize motor models

PMSM Parameterization from Datasheet

1. Pick torque and calculate efficiency at rated load (see code)
2. Compare torque sensor to rated torque sensor at rated speed
3. Compare torque sensor to rated torque sensor at rated speed

PMSM Parameterization from Datasheet

Two test harnesses that add confidence that a PMSM is correctly parameterized from a datasheet. It also calculates motor efficiency at

[Open Model](#)

From datasheet

Import IPMSM Flux Linkage Data from ANSYS Maxwell

Import a motor design from ANSYS® Maxwell® into a Simscape™ simulation.

[Open Model](#)

From ANSYS Maxwell, JMAG, Motor-CAD FEA tools

Import IPMSM Flux Linkage Data from Motor-CAD

Import a motor design from Motor-CAD into a Simscape™ simulation.

[Open Model](#)

From dyno data

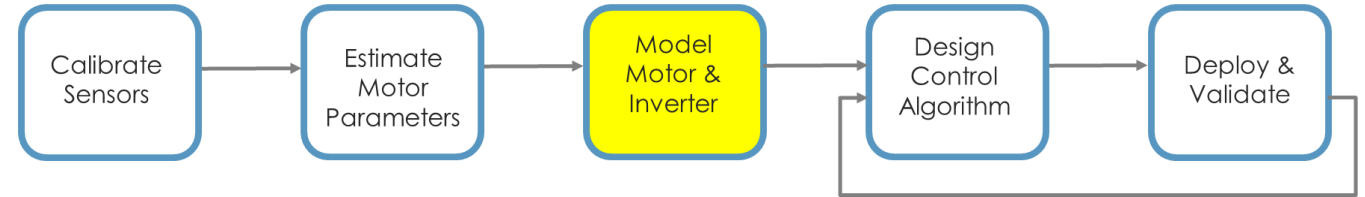
Generate Parameters for Flux-Based PMSM Block

Using MathWorks tools, you can create lookup tables for an interior permanent magnet synchronous motor (PMSM) controller that characterizes the d -axis and q -axis flux.

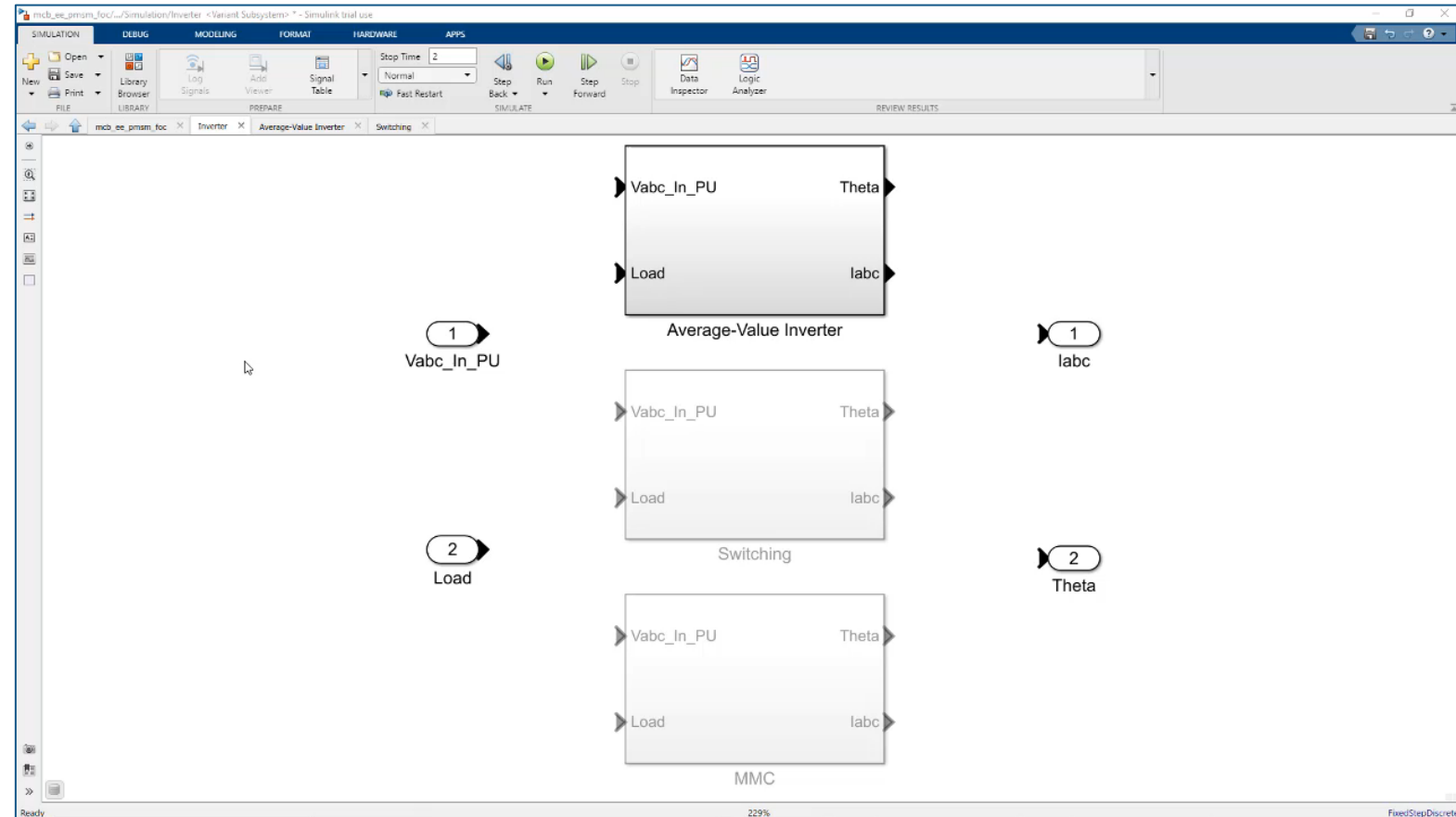
To generate the flux parameters for the Flux-Based PMSM block, follow these workflow steps. Example script `CreatingIdqTable.m` calls `gridfit` to model the current surface using scattered or semi-scattered flux data.

Workflow	Description
Step 1: Load and Preprocess Data	Load and preprocess this nonlinear motor flux data from dynamometer testing or finite element analysis (FEA): <ul style="list-style-type: none"> d- and q- axis current d- and q- axis flux Electromagnetic motor torque
Step 2: Generate Evenly Spaced Table Data From Scattered Data	Use the <code>gridfit</code> function to generate evenly spaced data. Visualize the flux surface plots.
Step 3: Set Block Parameters	Set workspace variables that you can use for the Flux-Based PM Controller block parameters.

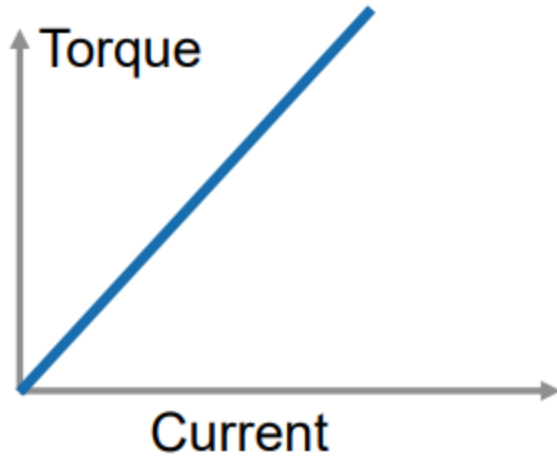
Modeling motor and inverter



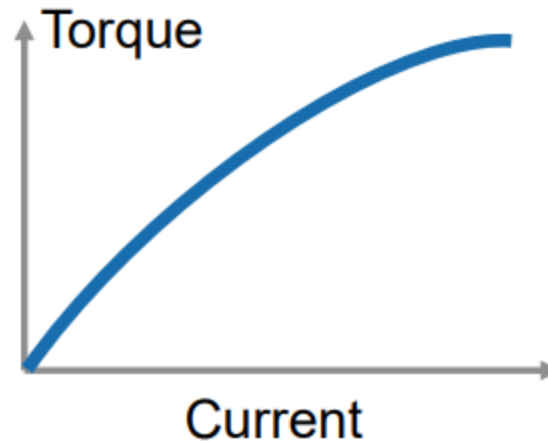
- Use linear lumped-parameter motor model
- Model inverter as an average-value inverter or model switching with Simscape Electrical



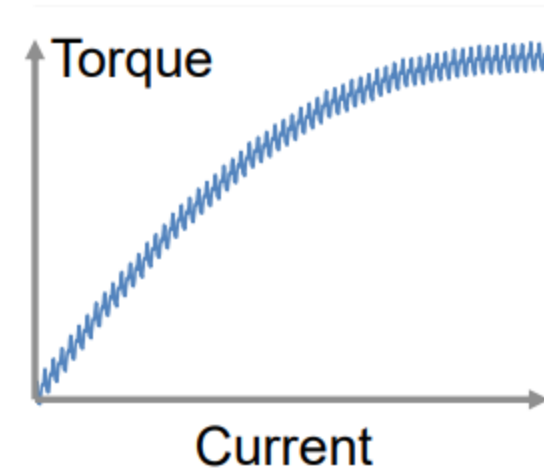
Bonus: you can model at needed level of fidelity



Lumped Parameter



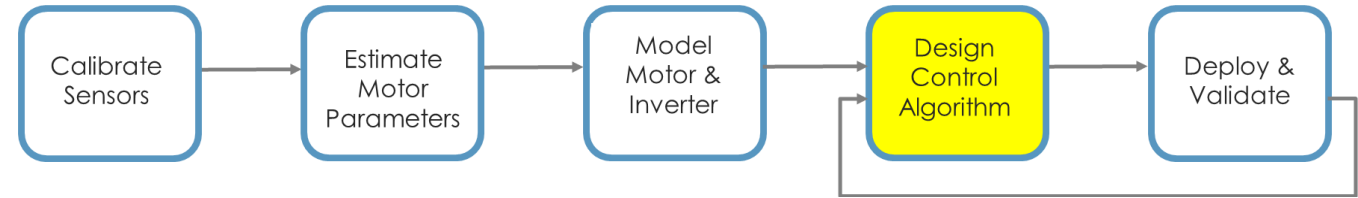
Saturation



Saturation +
Spatial Harmonics

Control algorithm design

- Model field-oriented control algorithm
- Model sensor decoders or sensorless observers
- Tune loop gains
- Verify in closed-loop simulation



Permanent Magnet Synchronous Motor Field Oriented Control

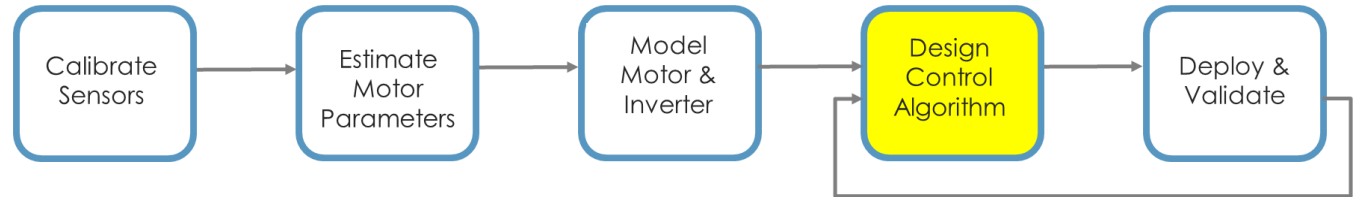
Note: This example requires a TI F28379D LaunchPad with a BOOSTXL-DRV8305 booster pack connected to a PMSM Motor with QEP Sensor

Explore more:
 1. Edit motor & inverter parameters
 2. Use Offset computation model to find out position offset.
 3. Update offset in Init script to variable 'pmsm.PositionOffset'
 4. Build, Deploy & Start
 5. Control motor via host model

Copyright 2020 The MathWorks, Inc.

Control algorithm design

- Model field-oriented control algorithm
- Model sensor decoders or sensorless observers
- Tune loop gains
- Verify in closed-loop simulation



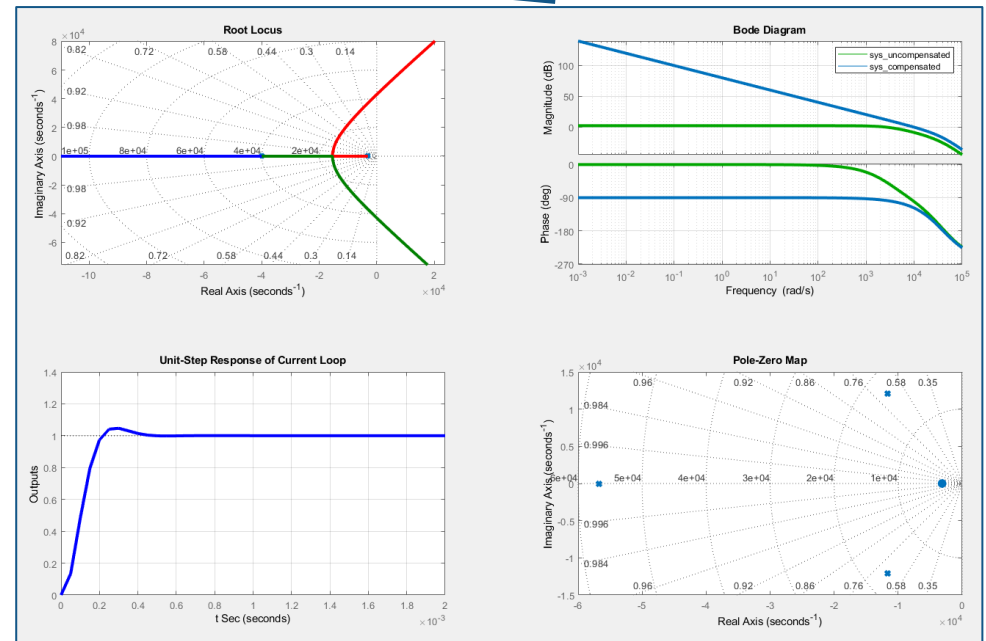
```

%% Controller design // Get ballpark values!

PI_params = mcb.internal.SetControllerParameters(pmsm, inverter, PU_System, T_pwm, Ts, Ts_speed);

%Updating delays for simulation
PI_params.delay_Currents = int32(Ts/Ts_simulink);
PI_params.delay_Position = int32(Ts/Ts_simulink);
PI_params.delay_Speed = int32(Ts_speed/Ts_simulink);
PI_params.delay_Speed1 = (PI_params.delay_IIR + 0.5*Ts)/Ts_speed;
mcb_getControlAnalysis(pmsm, inverter, PU_System, PI_params, Ts, Ts_speed);
  
```

Field	Value
T1	5.0000e-05
T2	5.0000e-04
sigma	5.0000e-05
Ti_i	3.1922e-04
Ti_id	3.3273e-04
damping	0.7071
Kp_i	2.5778
Ki_i	8.0752e+03
Kp_id	2.6869
Ki_id	8.0752e+03
Ki_texas	0.1566
Ki_d_texas	0.1503
delta	0.0263
delay_IIR	0.0200
x	1.2000
Ti_speed	0.0378
Kp_speed	0.9231
Ki_speed	24.4215



Bonus: you can use several techniques to tune loop gains

```

%% Set PWM Switching frequency
PWM_frequency = 20e3; %Hz // converter s/w freq
T_pwm = 1/PWM_frequency; %s // PWM switching time period

%% Set Sample Times
Ts = T_pwm; %sec // sample time for controller
Ts_simulink = T_pwm/2; %sec // simulation time step for model simulation
Ts_motor = T_pwm/2; %Sec // simulation sample time
Ts_inverter = T_pwm/2; %sec // simulation time step for average value inverter
Ts_speed = 10*Ts; %Sec // sample time for speed controller

%% Set data type for controller & code-gen
dataType = fixdt(1,32,17); % Fixed point code-generation
dataType = 'single'; % Floating point code-generation

%% System Parameters // Hardware parameters
pmsm = mcb_SetPMSMMotorParameters('BLY171D');

%% Parameters below are not mandatory for offset computation
inverter = mcb_SetInverterParameters('DRV8312-C2-KIT');
inverter.ADCOffsetCalibEnable = 1; % Enable: 1, Disable:0
target = mcb_SetProcessorDetails('F28069M', PWM_frequency);

%% Derive Characteristics
pmsm.N_base = mcb_getBaseSpeed(pmsm, inverter); %rpm // Base speed of motor at given Vdc
% mcb_getCharacteristics(pmsm, inverter);

%% PU System details // Set base values for pu conversion
PU_System = mcb_SetPUSystem(pmsm, inverter);

%% Controller design // Get ballpark values!
PI_params = mcb.internal.SetControllerParameters(pmsm, inverter, PU_System, T_pwm, Ts, Ts_speed);
    
```

Empirical Calculation

Tuning PI controllers by Using Field Oriented Control (FOC) Autotuner

Computes the gain values of the PI controllers within the speed and current controllers by using the Field Oriented Control Autotuner block.

[Open Example](#)

FOC Autotuner

Tune Field-Oriented Controllers Using SYSTUNE

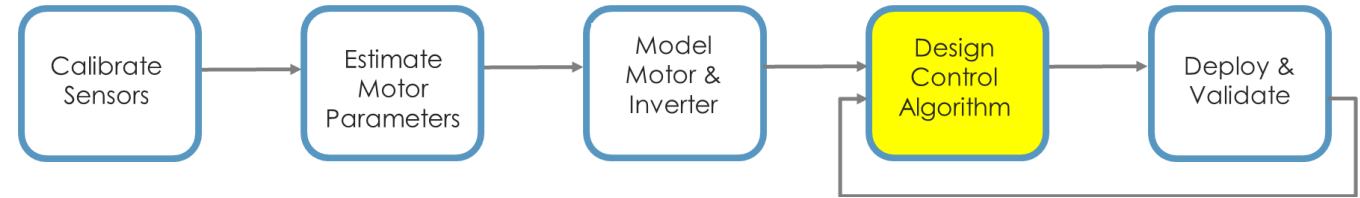
Tune a field-oriented controller for an asynchronous machine in one simulation.

[Open Script](#)

Classic Control Theory

Control algorithm design

- Model field-oriented control algorithm
- Model sensor decoders or sensorless observers
- Tune loop gains
- Verify in closed-loop simulation



Permanent Magnet Synchronous Motor Field Oriented Control

Note: This example requires a TI F28379D LaunchPad with a BOOSTXL-DRV8305 booster pack connected to a PMSM Motor with QEP Sensor

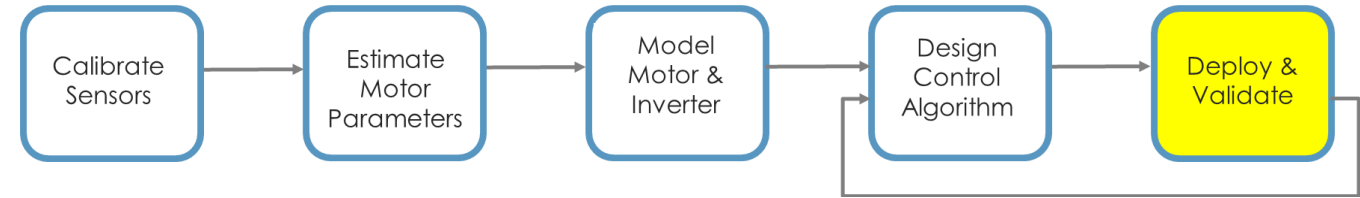
Explore more:

1. Edit motor & inverter parameters
2. Use Offset computation model to find out position offset.
3. Update offset in Init script to variable 'pmsm.PositionOffset'
4. Build, Deploy & Start
5. Control motor via host model

Copyright 2020 The MathWorks, Inc.

Deployment

- Generate code (floating and fixed-point)
- Use host model to control and debug
- Validate on hardware



Permanent Magnet Synchronous Motor Field Oriented Control

Note: This example requires a TI F28379D LaunchPad with a BOOSTXL-DRV8305 booster pack connected to a PMSM Motor with QEP Sensor

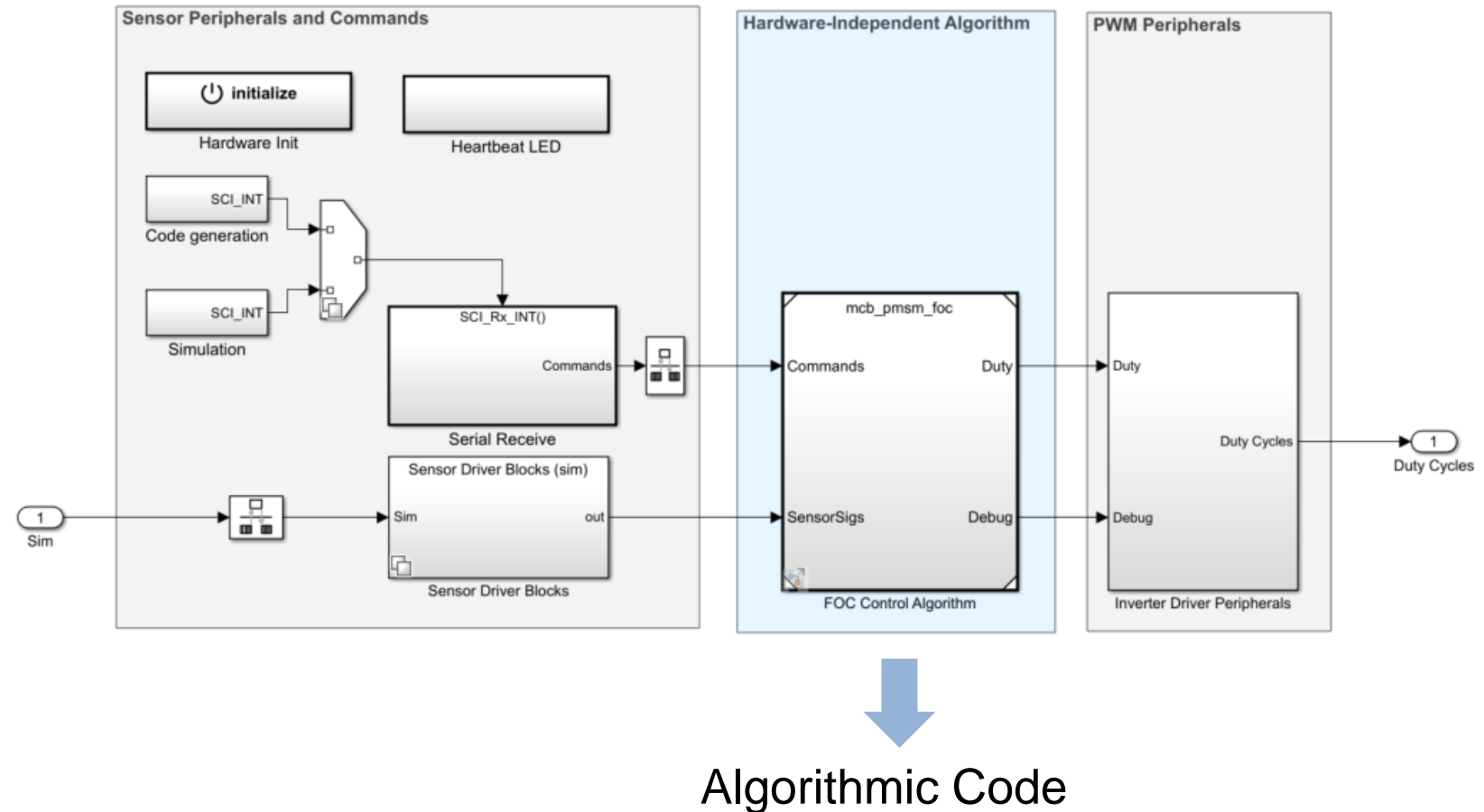
Copyright 2020 The MathWorks, Inc.

Explore more:

1. Edit motor & inverter parameters
2. Use Offset computation model to find out position offset
3. Update offset in Init script to variable `'pmsm.PositionOffset'`
4. Build, Deploy & Start
5. Control motor via host model

You can generate code for a custom target

- Target any processor with ANSI C code
- Use provided example to partition the model into algorithmic and hardware-specific parts
- Generate algorithmic code for integration into embedded application



You can verify and profile code using Processor-In-the-Loop testing




Code Execution Profiling Report for `mcb_pmsm_foc_sim_v2/Current Control1`

The code execution profiling report provides metrics based on data collected from a SIL or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the SIL or PIL test harness or inside the code generated for each component. See [Code Execution Profiling](#) for more information.

1. Summary

Total time	50681790
Unit of time	ns
Command	<code>report(executionProfile, 'Units', 'seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%0.0f');</code>
Timer frequency (ticks per second)	2e+08
Profiling data created	16-Jan-2020 18:09:48

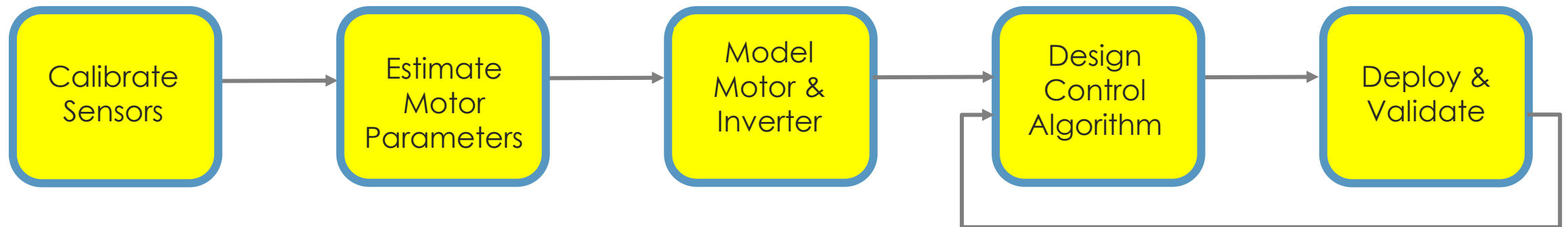
2. Profiled Sections of Code

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls	
[+] Current_initialize	2260	2260	1365	1365	1	
Current_step [5e-05 0]	5135	5067	5135	5067	10001	
Current_terminate	540	540	540	540	1	

3. CPU Utilization

Task	Average CPU Utilization	Maximum CPU Utilization
Current_step [5e-05 0]	10.13%	10.27%
Overall CPU Utilization	10.13%	10.27%

Workflow for implementing field-oriented control

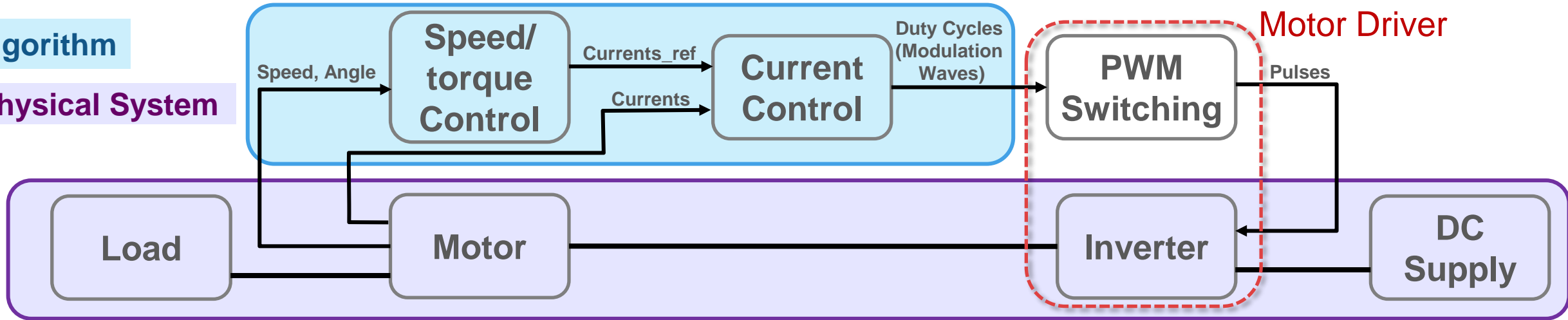


Motor Control Blockset & Simscape Electrical

Simscape Electrical	Motor Control Blockset	Key Differentiator
<ul style="list-style-type: none"> No single precision, no fixed point 	<ul style="list-style-type: none"> ❖ <u>Fast, compact code</u> ❖ <u>All precisions</u> 	<ul style="list-style-type: none"> Will you be deploying code to a microcontroller? <p>YES → You need Motor Control Blockset</p>

Algorithm

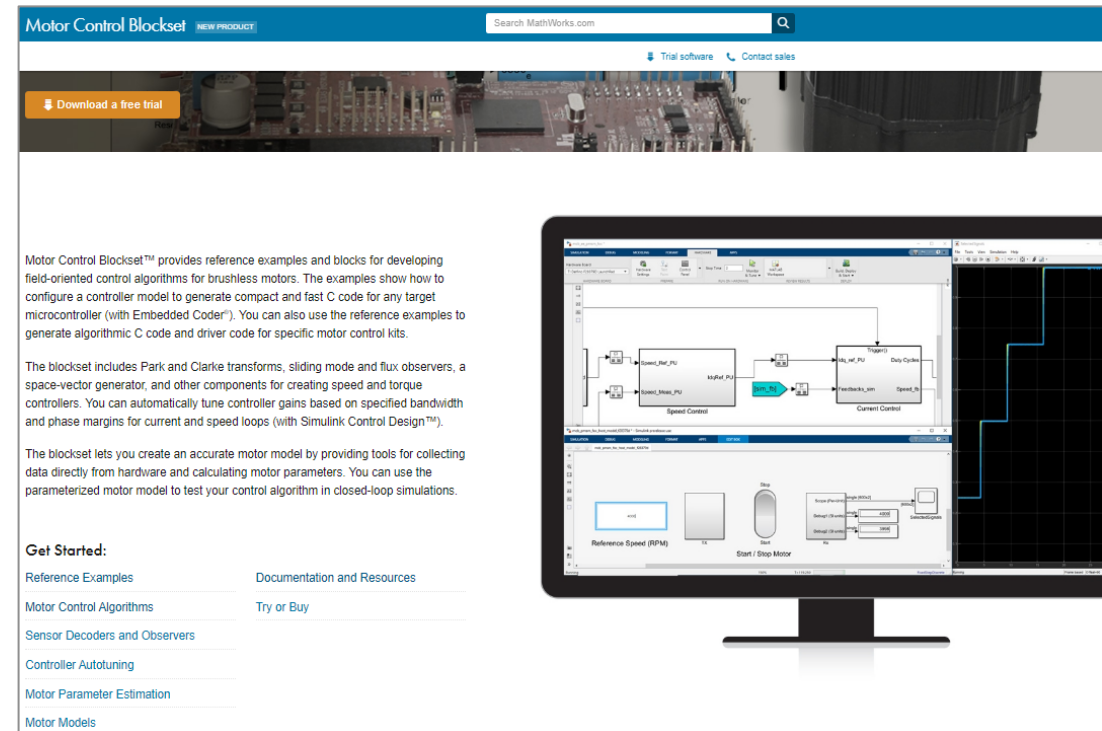
Physical System



Simscape Electrical	Motor Control Blockset	Key Differentiator
<ul style="list-style-type: none"> ❖ Switching (Motor Driver) ❖ All fidelity ranges ❖ Physical networks 	<ul style="list-style-type: none"> Average value (No switching) Linear motor with fixed L_d / L_q Signal-based 	<ul style="list-style-type: none"> Are switching effects important to you? Do you need nonlinear motor behavior? Is your motor part of a larger system (electrical, thermal network)? <p>YES → You need Simscape Electrical</p>

Use Model-Based Design for your next motor control project!

- Verify control algorithm with desktop simulation
- Generate compact and fast code from models
- Minimize development time using reference examples, built-in algorithmic blocks, automated parameter estimation, and gain-tuning



Motor Control Blockset NEW PRODUCT! [Trial software](#) [Contact sales](#)

[Download a free trial](#)

Motor Control Blockset™ provides reference examples and blocks for developing field-oriented control algorithms for brushless motors. The examples show how to configure a controller model to generate compact and fast C code for any target microcontroller (with Embedded Coder®). You can also use the reference examples to generate algorithmic C code and driver code for specific motor control kits.

The blockset includes Park and Clarke transforms, sliding mode and flux observers, a space-vector generator, and other components for creating speed and torque controllers. You can automatically tune controller gains based on specified bandwidth and phase margins for current and speed loops (with Simulink Control Design™).

The blockset lets you create an accurate motor model by providing tools for collecting data directly from hardware and calculating motor parameters. You can use the parameterized motor model to test your control algorithm in closed-loop simulations.

Get Started:

- [Reference Examples](#)
- [Documentation and Resources](#)
- [Motor Control Algorithms](#)
- [Try or Buy](#)
- [Sensor Decoders and Observers](#)
- [Controller Autotuning](#)
- [Motor Parameter Estimation](#)
- [Motor Models](#)

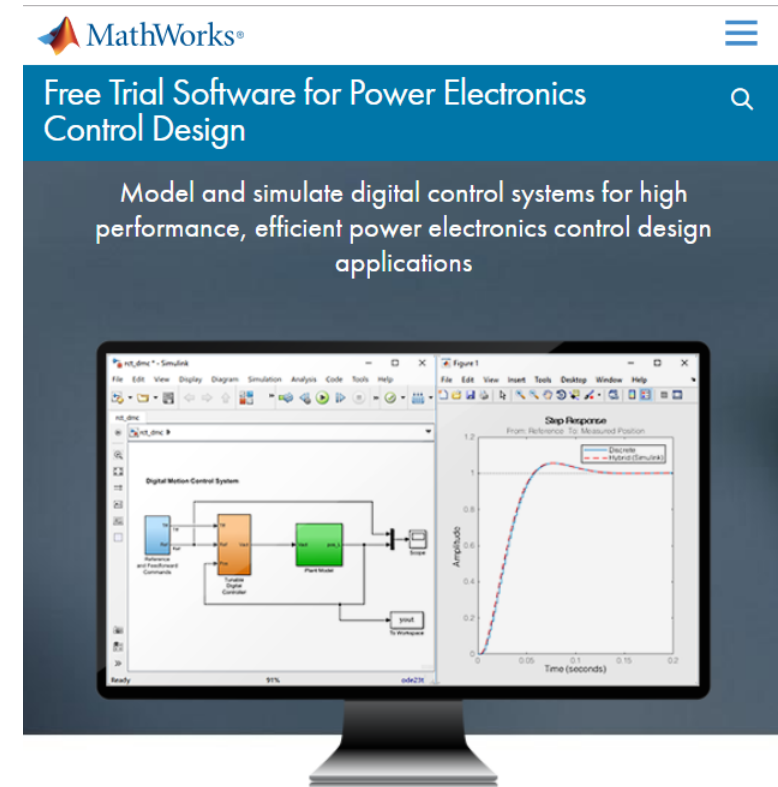
The image also shows a Simulink model on a monitor. The model includes blocks for 'Speed Control', 'Current Control', and 'Deep Cycles'. A graph on the right shows a step function for speed reference.

Takeaways

- Motor Control Blockset helps customers adopt production code generation for developing embedded motor control software
- It also provides an onramp to MBD for motor control engineers new to Simulink
- Reference applications is a key capability of the product
- There is intentional overlap with Simscape Electrical, many customers will benefit from using both products together

Learn More

- Visit <https://www.mathworks.com/products/motor-control.html>
- And <https://www.mathworks.com/solutions/power-electronics-control.html>
- Get [power electronics control design trial package](#) with necessary tools for desktop modeling, simulation, control design, and production code generation of your next motor control project



The screenshot shows the MathWorks website interface for the 'Free Trial Software for Power Electronics Control Design'. The page features a blue header with the MathWorks logo and a search icon. Below the header, the main content area has a dark blue background with white text that reads: 'Model and simulate digital control systems for high performance, efficient power electronics control design applications'. The central focus is a computer monitor displaying two Simulink windows. The left window, titled 'Digital Motion Control System', shows a block diagram with components like 'Digital Motor Controller' and 'Plant Model'. The right window, titled 'Step Response', shows a plot of 'Amplitude' versus 'Time (seconds)'. The plot contains two curves: a solid blue line for 'Desired' and a dashed red line for 'Measured (Simulink)', both showing a step response that rises from 0 to 1.0 over approximately 0.05 seconds.

START TODAY. Download and install the trial software package.