

MathWorks  
**AUTOMOTIVE  
CONFERENCE 2023**  
India

## Master class - Driving Efficiency and Performance using Motor Control Workflow for Electric Vehicle

*Rahul Choudhary, MathWorks*

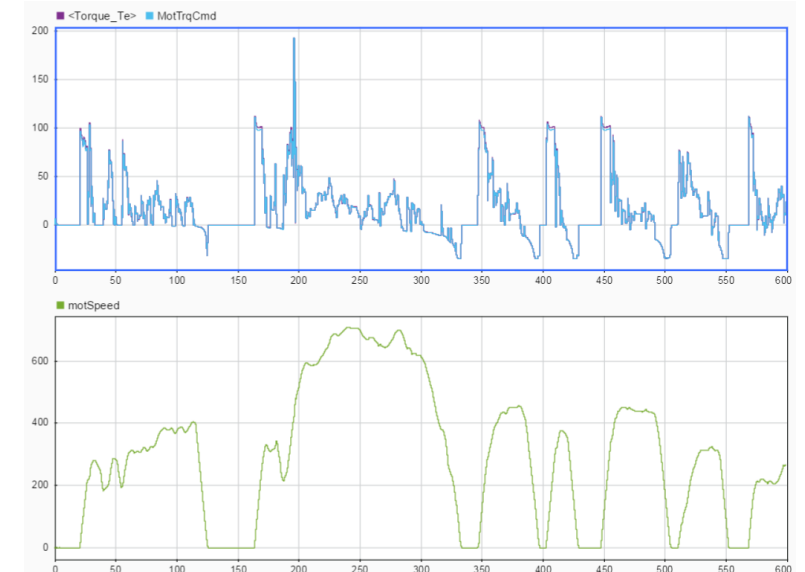
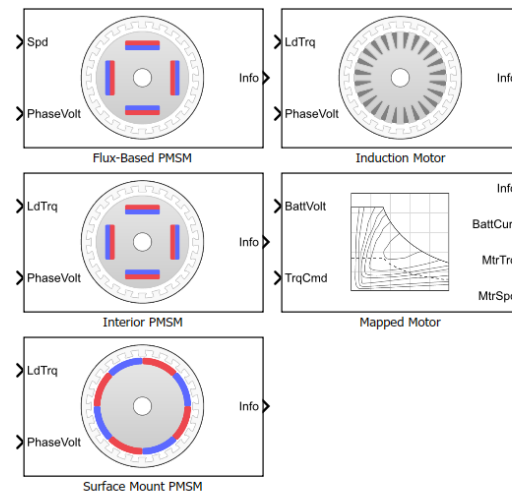
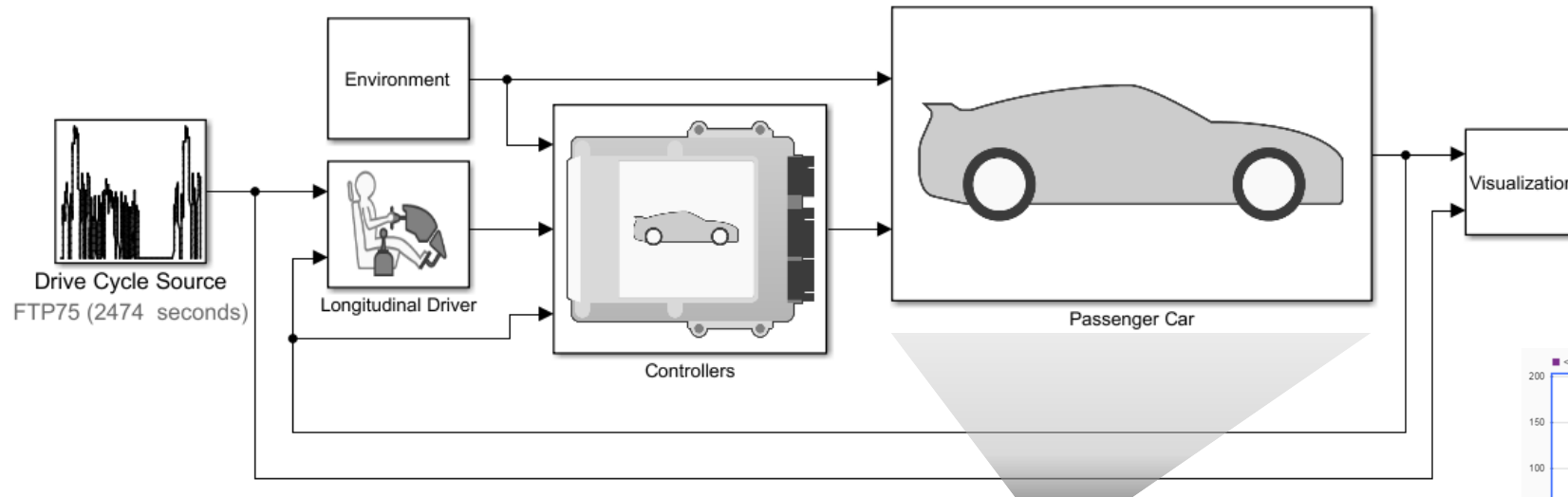


*Ananth Kumar, MathWorks*



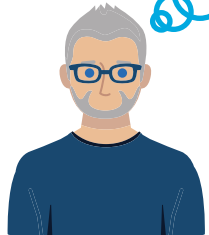
# The Heart of the Electric Powertrain: Motors

- Component Selection?
- Component Sizing?
- Trade-off Studies?
  
- Detailed Component Modeling and Control Design?



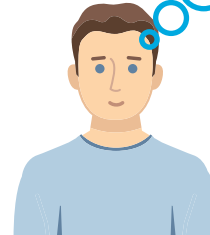
# Challenges in Motor Control Development for Electric Vehicle

What steps I should follow to implement efficient field-weakening control?



EV Start-up Engineer

How do I get motor parameters to develop my motor model?



Modeling Engineer

How do I generate efficient code and port to different hardware ?



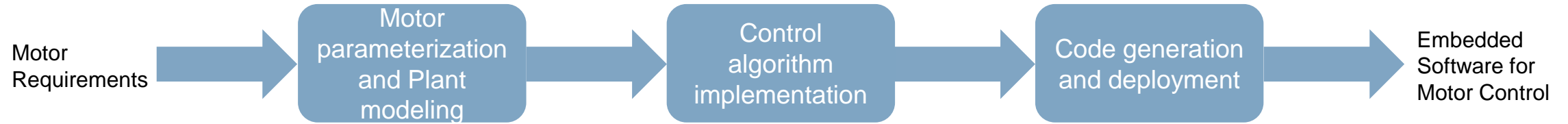
Embedded Engineer

How do I tune my Control loop gains to achieve the performance?

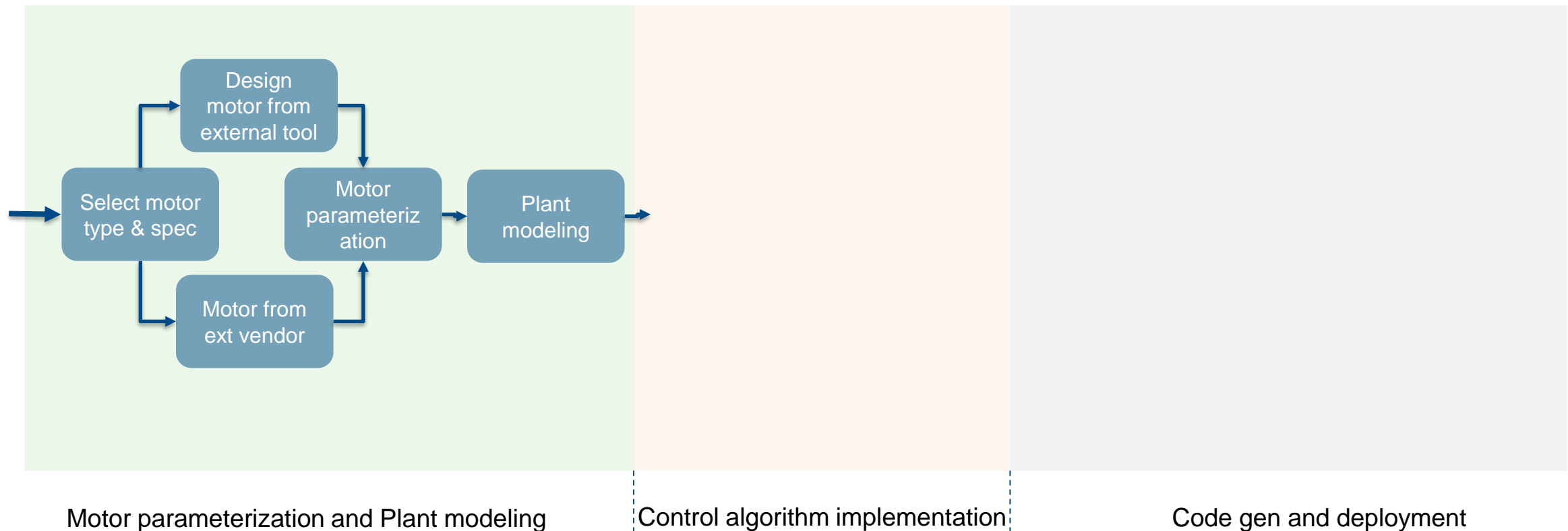


Control Engineer

# Proposed motor control software development workflow for EV

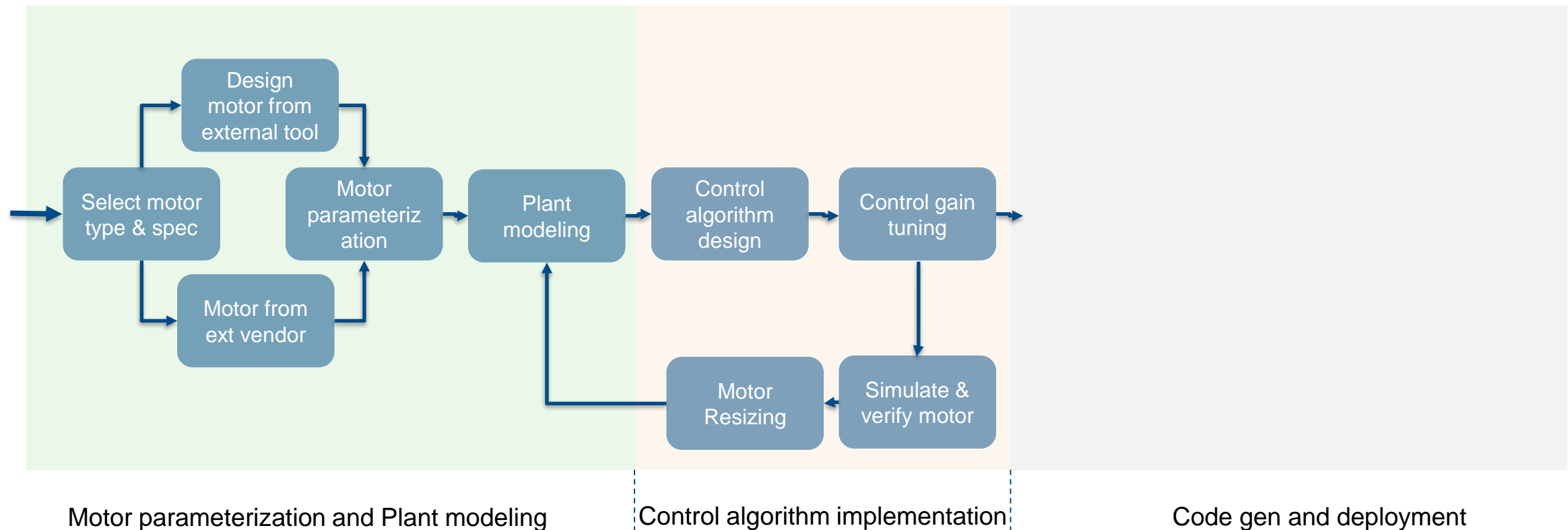


# Proposed motor control software development workflow for EV

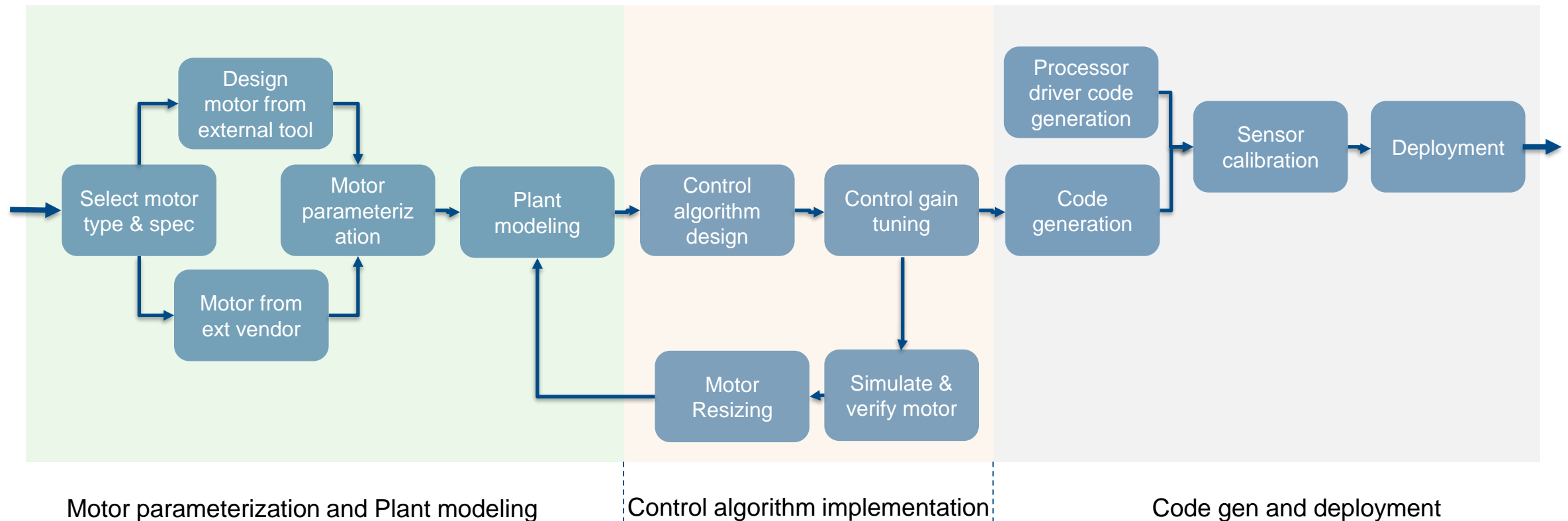




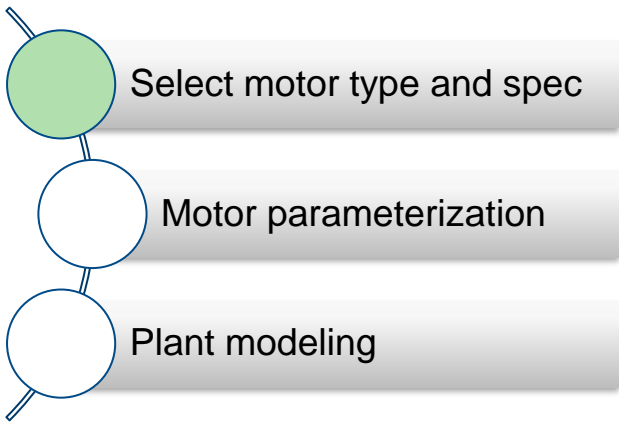
# Proposed motor control software development workflow for EV



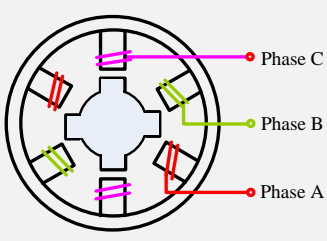
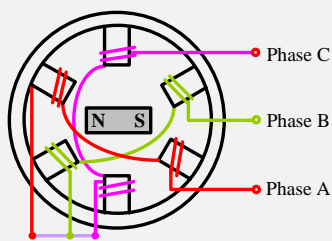
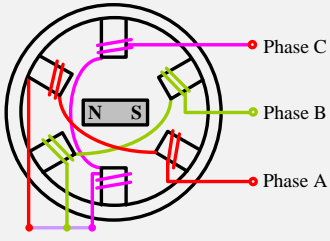
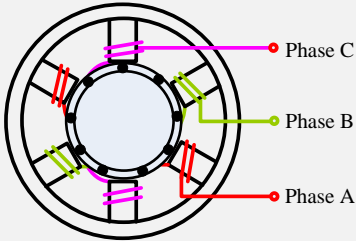
# Proposed motor control software development workflow for EV



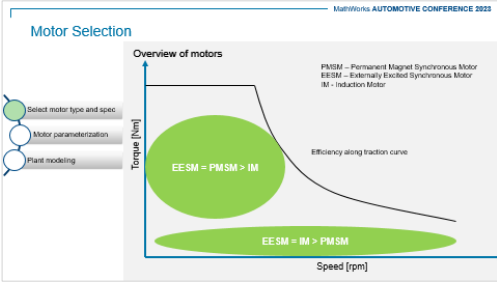
# Motor Selection



## Overview of motors



Squirrel Cage Induction Motor (IM)	Brushless DC Motor (BLDC)	Permanent Magnet Synchronous Motor (PMSM)	Switched Reluctance Motor (SRM)
<b>Advantages</b>			
<ul style="list-style-type: none"> <li>- Low material cost/ kg</li> <li>- Robust, reliable, Simple Control</li> <li>- Less maintenance</li> </ul>	<ul style="list-style-type: none"> <li>- Efficient, reliable</li> <li>- High Power Density and High torque at low speed</li> </ul>	<ul style="list-style-type: none"> <li>- Low noise, smooth operation</li> <li>- High performance and efficiency over operating range</li> </ul>	<ul style="list-style-type: none"> <li>- Simple and robust construction</li> <li>- Low cost, long constant power range</li> </ul>
<b>Disadvantages</b>			
<ul style="list-style-type: none"> <li>- Low efficiency, low power factor specially at light loads, Heavy, High Copper loss</li> </ul>	<ul style="list-style-type: none"> <li>- Higher torque ripples and cogging torque leads to vibration, noise and poor position control</li> </ul>	<ul style="list-style-type: none"> <li>- High Cost</li> <li>- Demagnetizing risk</li> <li>- Requires complex control</li> </ul>	<ul style="list-style-type: none"> <li>- High noise, torque ripples</li> <li>- Complex Control</li> <li>- High Switching losses</li> </ul>
<b>Industry Examples</b>			
Tesla model X, Toyota RAV4 EV, Renault Zoe	Hero Electric, Yamaha EC-03, Bounce	Tesla Model S, Chevrolet Bolt, Hyundai Kona Electric	Jaguar I-PACE Concept, Ford Fiesta EV Prototype





# Motor Selection

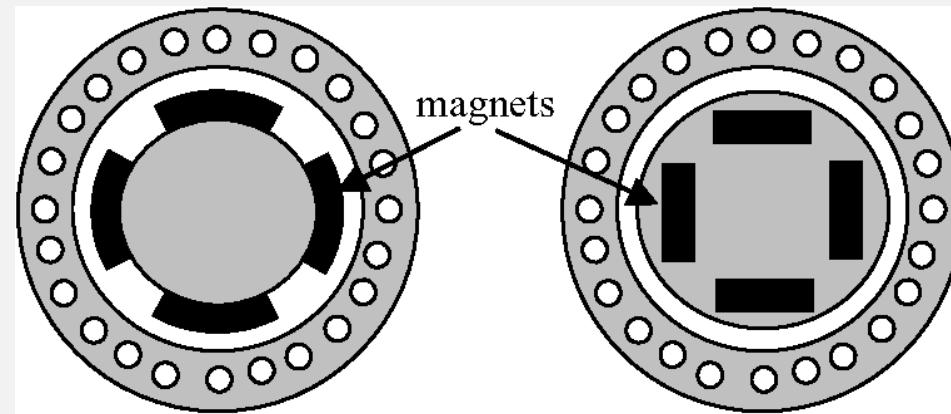
Select motor type and spec

Motor parameterization

Plant modeling

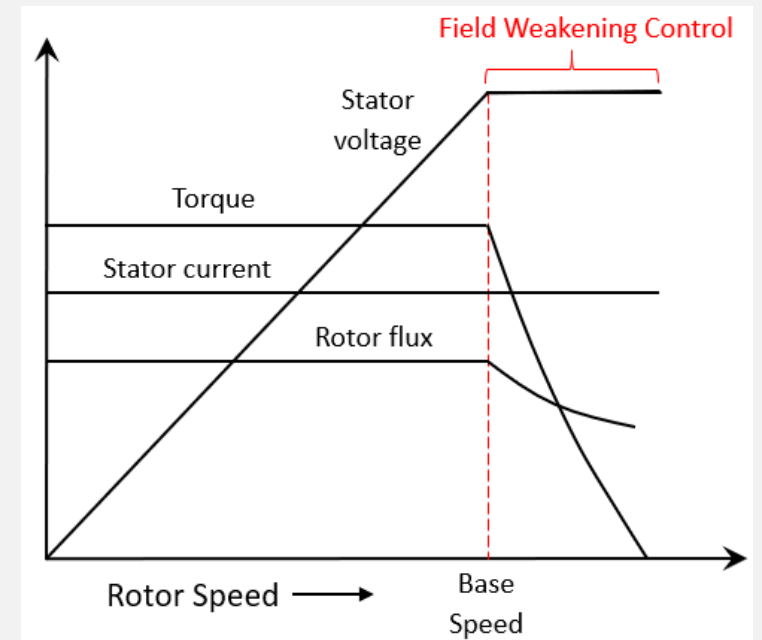
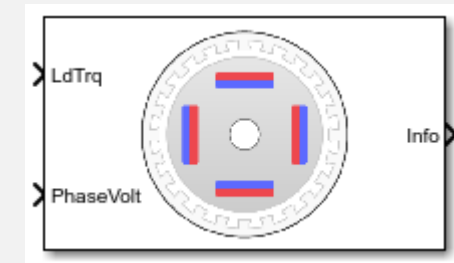
In this presentation, we will focus on Interior PMSM because of:

- High power density and efficiency
- Motor can spin more than base speed



Surface Mount PMSM

Interior PMSM

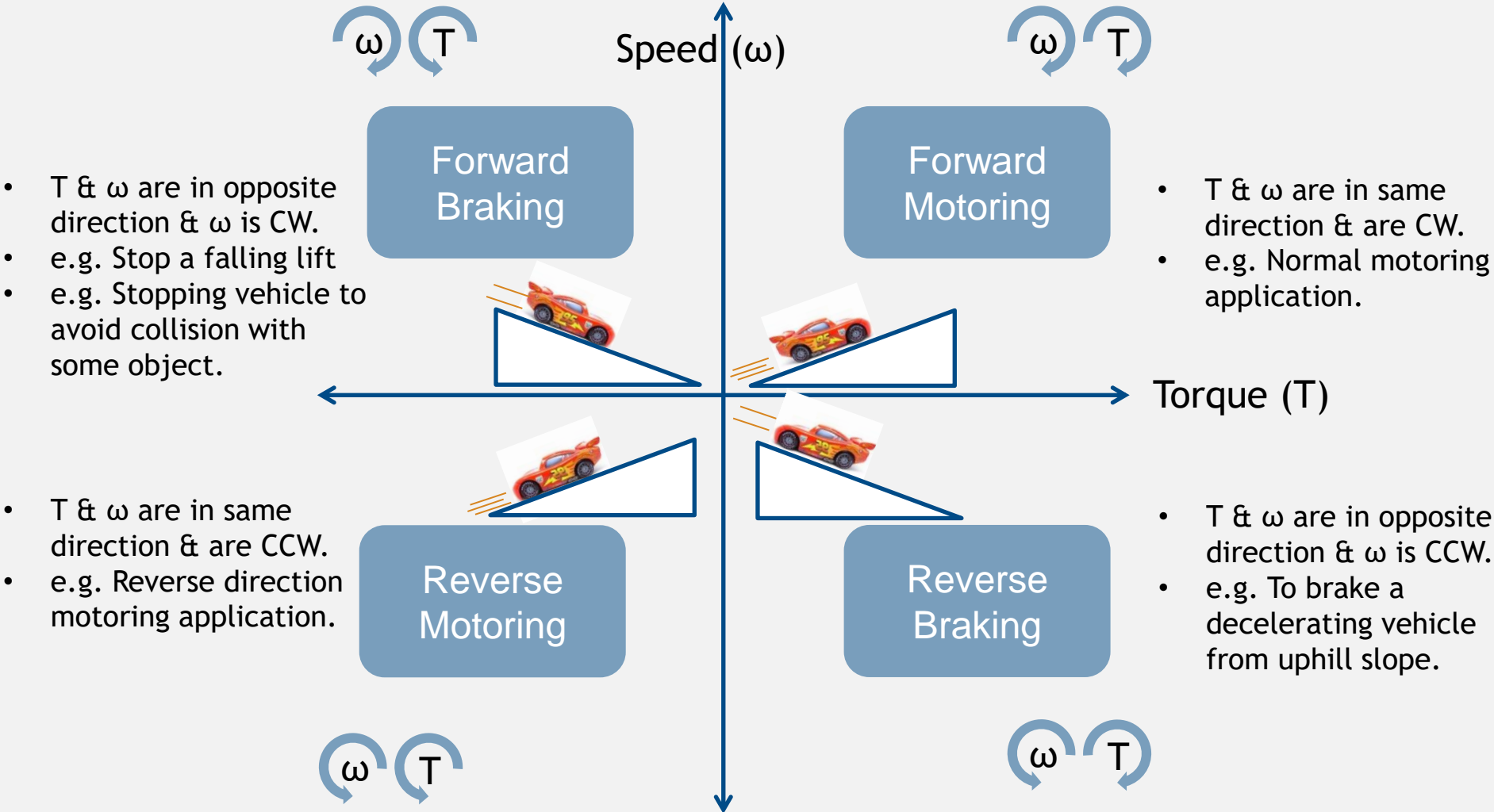


Motor characteristics

# Motor Selection

- Select motor type and spec
- Motor parameterization
- Plant modeling

## Motor functionality – Four quadrant operations

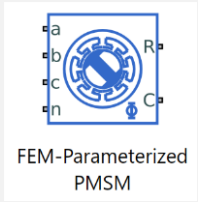
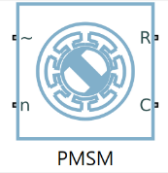
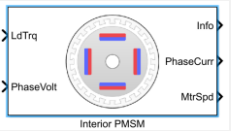
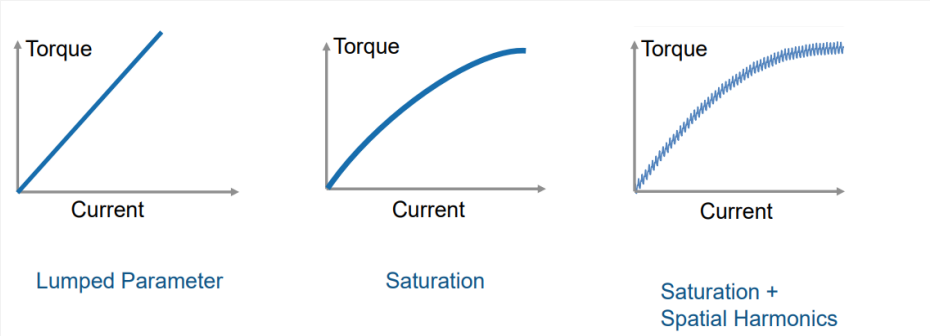


• Note: CW direction of T &  $\omega$  is taken as +ve ref for explanation only.

# Motor Parameterization and Model Fidelity

- Select motor type and spec
- Motor parameterization
- Plant modeling

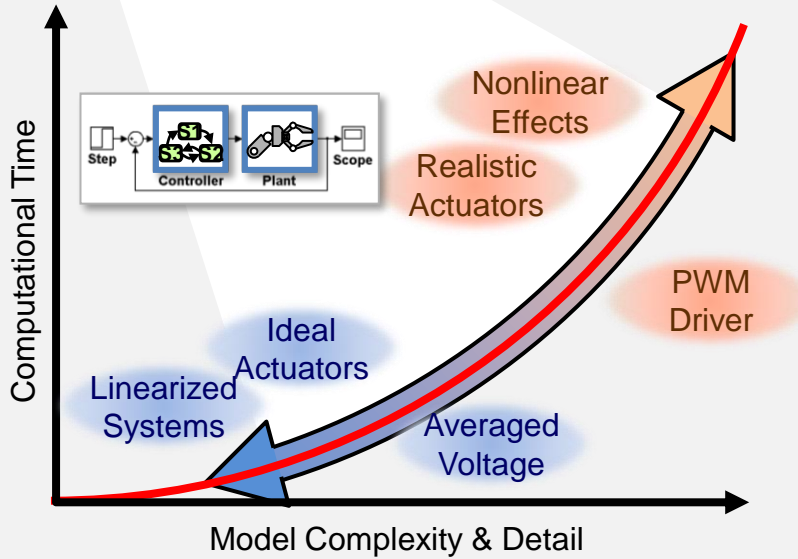
## Motor Model Fidelity Level



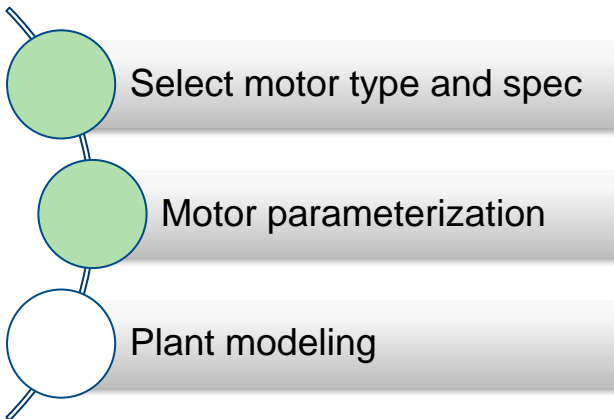
Motor Control Blockset

Simscape Electrical

## Computational Time vs. Model Complexity



# Motor Parameterization and Plant Modeling



## Motor Parameterization

If motor exist

**Estimate Motor Parameters Using Motor Control Blockset Parameter Estimation Tool**

Motor Control Blockset™ provides a parameter estimation tool that estimates the motor parameters accurately. Use the estimated motor parameters to simulate the motor model and design the control system. Therefore, the simulation response with the estimated parameters for the motor model is close to the behavior of the motor under test.

The parameter estimation tool determines these motor parameters for a Permanent Magnet Synchronous Motor:

Motor parameters	Units
Phase resistance ( $R_s$ )	Ohm
$d$ and $q$ axis inductances ( $L_d$ and $L_q$ )	Henry
Back-EMF constant ( $K_e$ )	Vpk_LL/krpm (where Vpk_LL is the peak voltage line-to-line measurement)
Motor inertia ( $J$ )	Kg.m <sup>2</sup>
Friction constant ( $F$ )	N.m.s

The parameter estimation tool accepts the minimum required inputs, runs tests on the target hardware, and displays the estimated parameters.

From datasheet or Instrumented tests running on the Hardware

Lumped Parameters

If motor and dyno test setup are available

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

From Dyno test data

Saturation

Motor doesn't exist but FEA data from motor design tool is available

**Import IPMSM Flux Linkage Data from ANSYS Maxwell**

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

[Open Model](#)

**Import IPMSM Flux Linkage Data from Motor-CAD**

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

[Open Model](#)




From FEA tools such as ANSYS Maxwell, JMAG, Motor-CAD

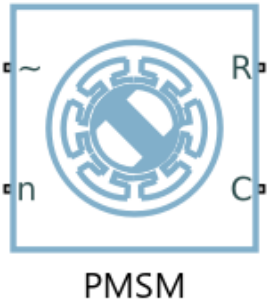
Saturation + Spatial Harmonics

Parameterization helps with Motor Modeling which captures motor dynamics and helps us with Control Design

# Motor Parameterization and Plant Modeling

## Motor Parametrization using Datasheet

-  Select motor type and spec
-  Motor parameterization
-  Plant modeling



Block Parameters: PMSM

Settings Description

NAME	VALUE
Modeling option	No thermal port
Selected part	<click to select>
<b>Main</b>	
Electrical connection	Composite three-phase ports
Winding type	Wye-wound
Modeling fidelity	Constant Ld, Lq, and PM
> Number of pole pairs	6
> Permanent magnet flux linkage parameterization	Specify flux linkage
> Permanent magnet flux linkage	0.03 Wb
> Stator parameterization	Specify Ld, Lq, and L0
> Stator d-axis inductance, Ld	0.00019 H
> Stator q-axis inductance, Lq	0.00025 H
> Stator zero-sequence inductance, L0	0.00016 H
> Stator resistance per phase, Rs	0.013 Ohm
Zero sequence	Include
Rotor angle definition	Angle between the a-phase magnetic axis
<b>Iron Losses</b>	
<b>Mechanical</b>	
<b>Initial Targets</b>	
<b>Nominal Values</b>	

Block Parameterization Manager: PMSM

SELECT FORMAT

Apply all Reset all

Manufacturer: All

PARAMETERIZE

Select part

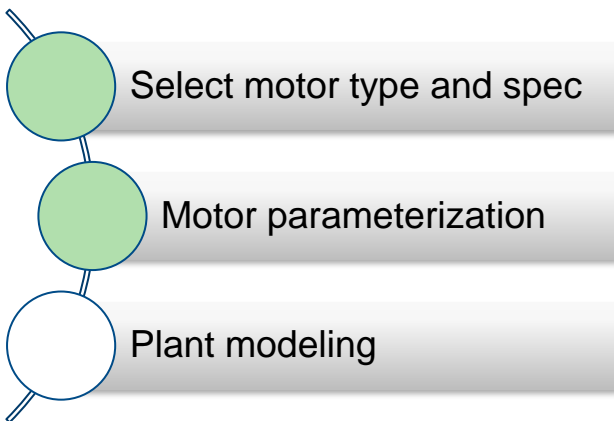
Part number	Manufacturer	Rated Speed, rpm	Pole Pairs
BSM132C-8200AA	ABB_BALDOR	1800	4
BSM33C-5177MHQ	ABB_BALDOR	1800	4
BSM50N-133	ABB_BALDOR	4000	2
BSM50N-275	ABB_BALDOR	2000	2
BSM63N-133	ABB_BALDOR	4000	2
HDS100-0206A	ABB_BALDOR	3000	5
HDS130-0817B	ABB_BALDOR	2000	5
HDS130-1829B	ABB_BALDOR	2900	18.0000
HDS180-2540B	ABB_BALDOR	4000	25.0000
HDS180-4876B	ABB_BALDOR	7600	48.0000
HDS185-0102A	ABB_BALDOR	1800	3000

Compare selected part with block

Parameter name	Parameterization	Override datasheet value	Part value:BSM132C
Main>Number of pole pairs	Datasheet derived	<input checked="" type="checkbox"/>	4
Main>Permanent magnet flux linkage	Datasheet derived	<input checked="" type="checkbox"/>	0.229230859562701
Main>Torque constant	Datasheet derived	<input checked="" type="checkbox"/>	0.916923438250803
Main>Back EMF constant	Datasheet derived	<input checked="" type="checkbox"/>	0.916923438250803
Main>Stator d-axis inductance, Ld	Datasheet derived	<input checked="" type="checkbox"/>	0.00115
Main>Stator q-axis inductance, Lq	Datasheet derived	<input checked="" type="checkbox"/>	0.00115
Main>Direct-axis current vector, iD	Parameter not set	<input type="checkbox"/>	[-200 0 200]
Main>Quadrature-axis current vector, iQ	Parameter not set	<input type="checkbox"/>	[-200 0 200]

# Motor Parameterization and Plant Modeling

## Motor parameters estimation from Instrumented tests



**Board Selection**  
DRV8305 and F28379D Launchpad

**Communication Port**  
Serial Setup

The COM port has to match your board  
For F28069 Launchpad, set Baudrate to 5.625e6  
For F28379D Launchpad, set Baud rate to 5e6

**Required Inputs**

Nominal Voltage: 24 V  
Nominal Current: 7.1 A (rms value)  
Nominal Speed: 4000 rpm  
Pole pairs: 4  
Input DC Voltage: 20 V  
Hall Offset: 0.28 Per Unit Position

Note: Press Ctrl+D to update the workspace

Hall Offset: For Hall offset calculation open required model for the hardware  
[mcb\\_pmsm\\_hall\\_offset\\_f28069m](#)  
[mcb\\_pmsm\\_hall\\_offset\\_f28379d](#)

Target Models: Click Build load and Run in required model for loading the target  
[mcb\\_param\\_est\\_f28069\\_DRV8312](#)  
[mcb\\_param\\_est\\_f28379D\\_DRV8305](#)

**Test Status**  
Run  Stop

**Estimated Motor Parameters**

Rs	0.4775	ohm
Ld	2.1052e-04	H
Lq	2.0117e-04	H
Bemf	--	Vpp/krpm
Motor Inertia	--	kg.m <sup>2</sup>
Friction constant	--	N.m.s

Save Parameters Open Model

**Signal from Target**

Running 95%

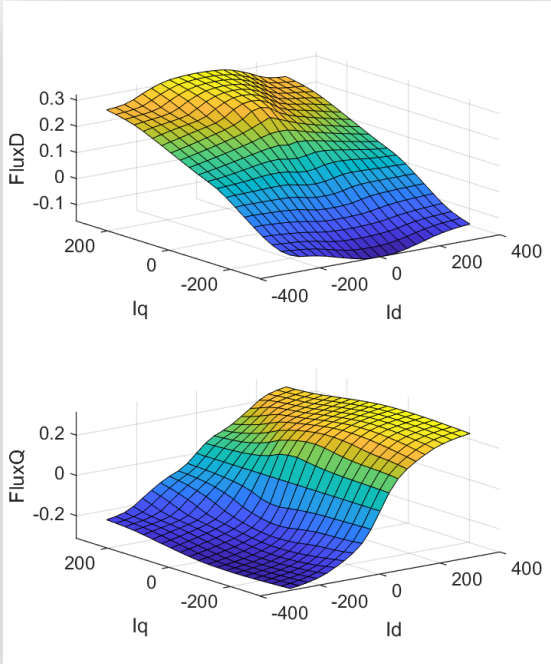
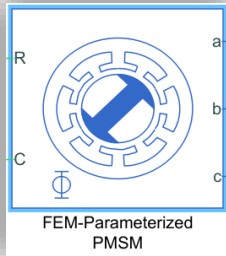
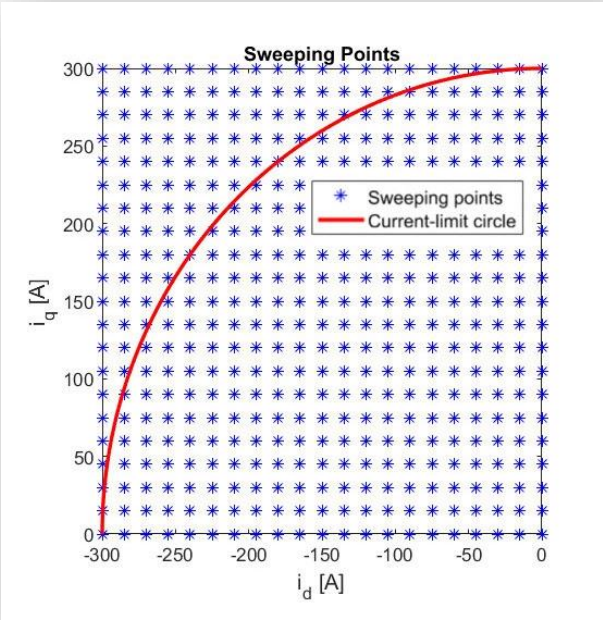
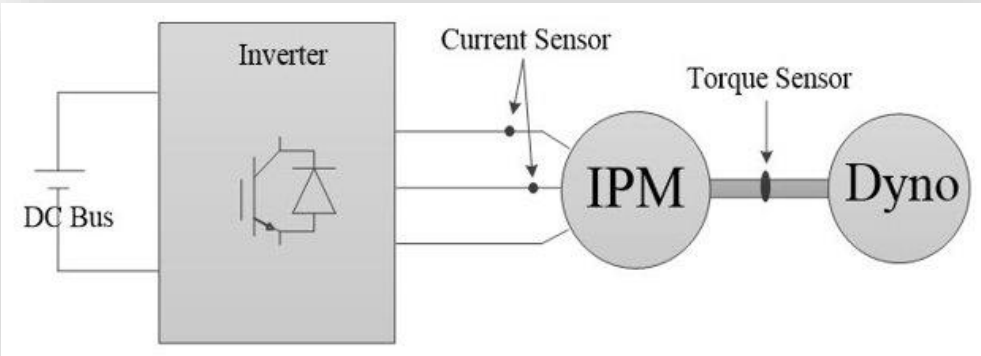
- Instrumented tests running on the target
- Sensor-based and Sensor-less modes available
- Supports PMSM and Induction Motor



# Motor Parameterization and Plant Modeling

- Select motor type and spec
- Motor parameterization
- Plant modeling

## Motor parameters from dyno test



Block Parameters: FEM-Parameterized PMSM

FEM-Parameterized PMSM (DQ0 flux data)

This block implements the electrical and mechanical characteristics of a permanent magnet synchronous motor for which magnetic flux linkage depends nonlinearly on currents and rotor angle. Right-click on the block and select Simscape block choices to access variant implementations.

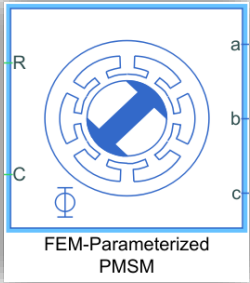
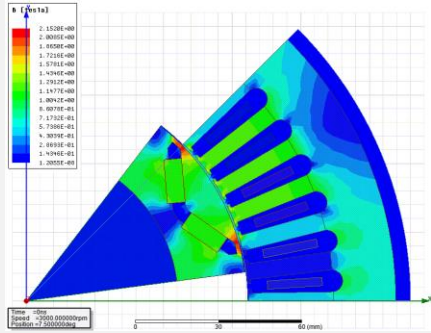
Settings

Electrical	Iron Losses	Mechanical	Variables
Flux linkage data format: D and Q axes flux linkages as a function of D-axis current (iD), Q-axis current (iQ), and rotor angle (theta)			
Winding type: Wye-wound			
Expose neutral port: No			
Number of pole pairs: N			
Park's convention for tabulated data: D leads Q, rotor angle measured from A-phase to D-axis			
Direct-axis current vector, iD:		idVec	A
Quadrature-axis current vector, iQ:		iqVec	A
Rotor angle vector, theta:		angleVec	deg
D-axis flux linkage, Fd(iD,iQ,theta):		fluxD	Wb
Q-axis flux linkage, Fq(iD,iQ,theta):		fluxQ	Wb
Torque matrix, T(iD,iQ,theta):		torque	N*m
Interpolation method: Linear			
Stator resistance per phase, Rs: 0.07 Ohm			

# Motor Parameterization and Plant Modeling

- Select motor type and spec
- Motor parameterization
- Plant modeling

## Motor parameters from motor design tool



```

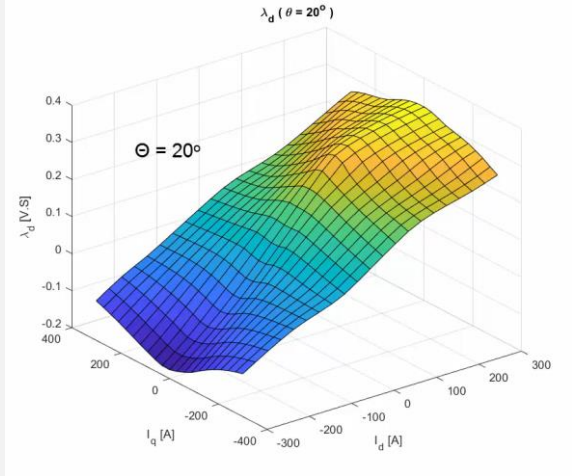
*****
* Copyright 2017-2019 ANSYS, Inc.  Unauthorized use, distribution, or
* ANSYS and all other ANSYS, Inc. product names and trademarks
* of ANSYS, Inc. or its subsidiaries in the United States or other
* countries is reproduced with permission of ANSYS, Inc.
*****
B_BasicData
  Version 1.0
  Poles 8
E_BasicData
B_PhaseImp 3
  PhaseA 1.0000000000e-003 1.0000000000e-003
  PhaseB 1.0000000000e-003 1.0000000000e-003
  PhaseC 1.0000000000e-003 1.0000000000e-003
E_PhaseImp
B_Sweepings
  Id_Iq (21: -300 -270 -240 -210 -180 -150 -120
        (21: -300 -270 -240 -210 -180 -150 -120
  Rotate (31: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
E_Sweepings
B_OutputMatrix DQ0

```


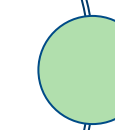

```

Editor - C:\Program Files\MATLAB\R2021a\toolbox\physmod\elec\eedemos\ee_ece_table.m
ee_ece_table.m x +
13
14 N = 8/2; % Number of pole pairs
15
16 %B_PhaseImp 3
17 % PhaseA 1.0000000000e-003 1.0000000000e-003
18 % PhaseB 1.0000000000e-003 1.0000000000e-003
19 % PhaseC 1.0000000000e-003 1.0000000000e-003
20 %E_PhaseImp
21
22 %B_Sweepings
23 idVec = [-300 -270 -240 -210 -180 -150 -120 -90 -60 -30 0 30 60 90 120 150 180 210 240 270 300];
24 iqVec = [-300 -270 -240 -210 -180 -150 -120 -90 -60 -30 0 30 60 90 120 150 180 210 240 270 300];
25 angleVec = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31];
26 %E_Sweepings
27
28 %B_OutputMatrix DQ0
29 data = [
30 % index fluxD
31 0 -9.2992778243e-002 -3.02

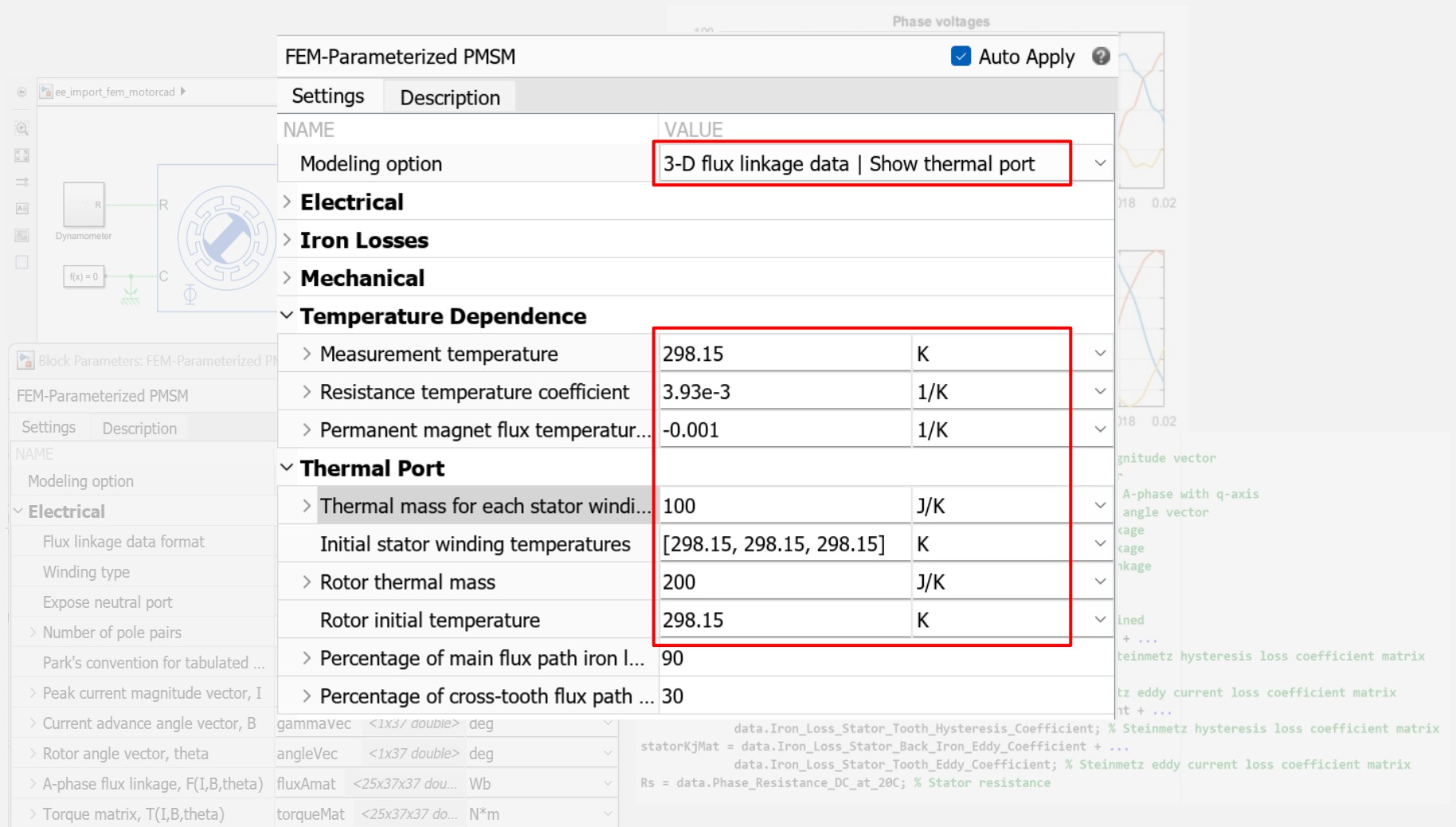
```



# Motor Parameterization and Plant Modeling

-  Select motor type and spec
-  Motor parameterization
-  Plant modeling

## Motor parameters from motor design tool



**FEM-Parameterized PMSM**  Auto Apply

NAME	VALUE
Modeling option	3-D flux linkage data   Show thermal port
<b>&gt; Electrical</b>	
<b>&gt; Iron Losses</b>	
<b>&gt; Mechanical</b>	
<b>&gt; Temperature Dependence</b>	
> Measurement temperature	298.15 K
> Resistance temperature coefficient	3.93e-3 1/K
> Permanent magnet flux temperatur...	-0.001 1/K
<b>&gt; Thermal Port</b>	
> Thermal mass for each stator windi...	100 J/K
Initial stator winding temperatures	[298.15, 298.15, 298.15] K
> Rotor thermal mass	200 J/K
Rotor initial temperature	298.15 K
> Percentage of main flux path iron l...	90
> Percentage of cross-tooth flux path ...	30

```

data.Iron_Loss_Stator_Tooth_Hysteresis_Coefficient; % Steinmetz hysteresis loss coefficient matrix
statorKjMat = data.Iron_Loss_Stator_Back_Iron_Eddy_Coefficient + ...
data.Iron_Loss_Stator_Tooth_Eddy_Coefficient; % Steinmetz eddy current loss coefficient matrix
Rs = data.Phase_Resistance_DC_at_20C; % Stator resistance
    
```

# Motor Constraint Curves and Characteristics

- Display Motor Constraints such as
  - MTPA curve (Maximum Torque per Ampere)
  - MTPV curve (Maximum Torque per Voltage)
  - Voltage Limit curve
  - Current Limit

- Exploration Motor Characteristics

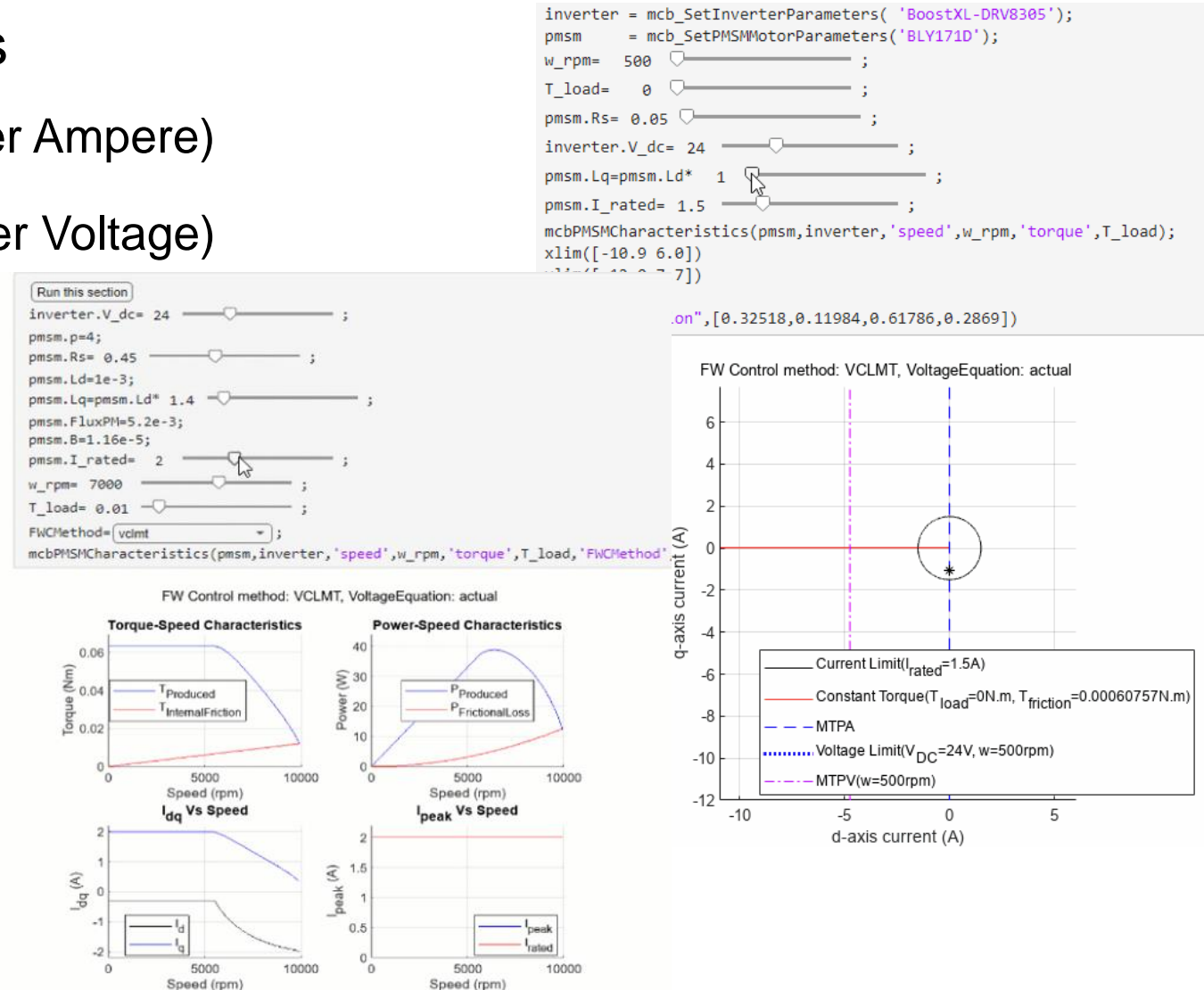
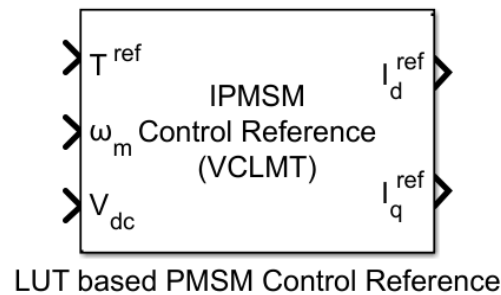
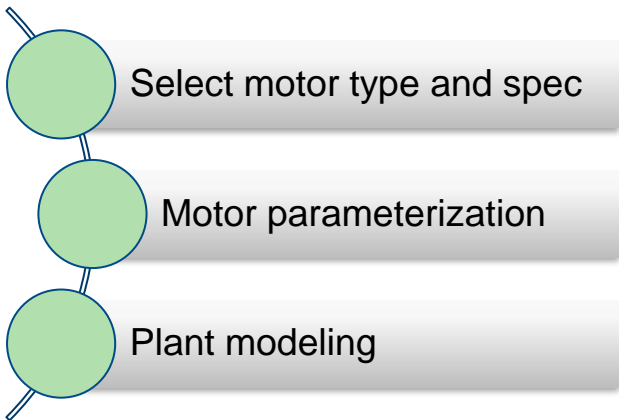
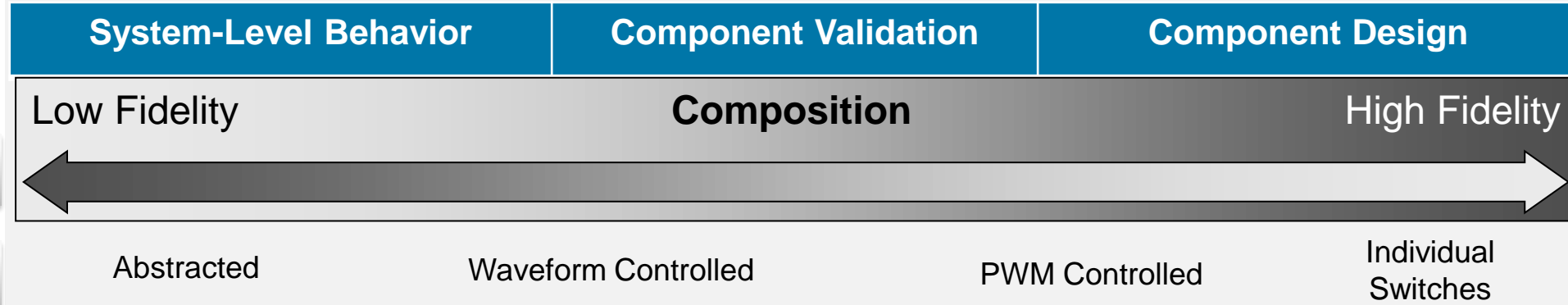


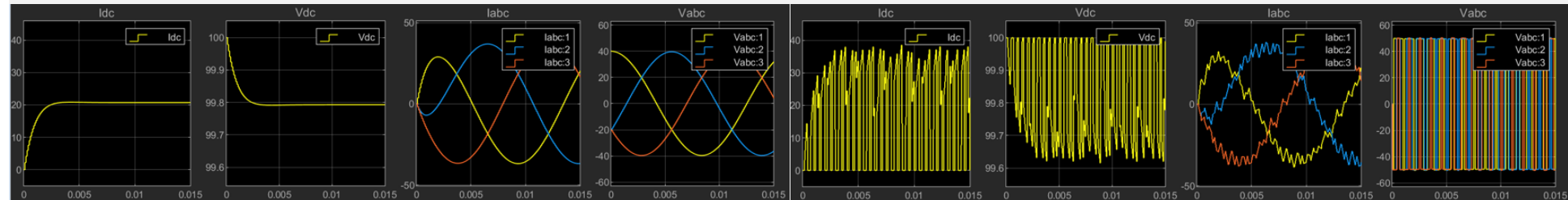
Figure : Dependency of curves and characteristics on parameters changes

# Motor Parameterization and Plant Modeling

## Power Converter Model Fidelity



## Semiconductor Converter Models



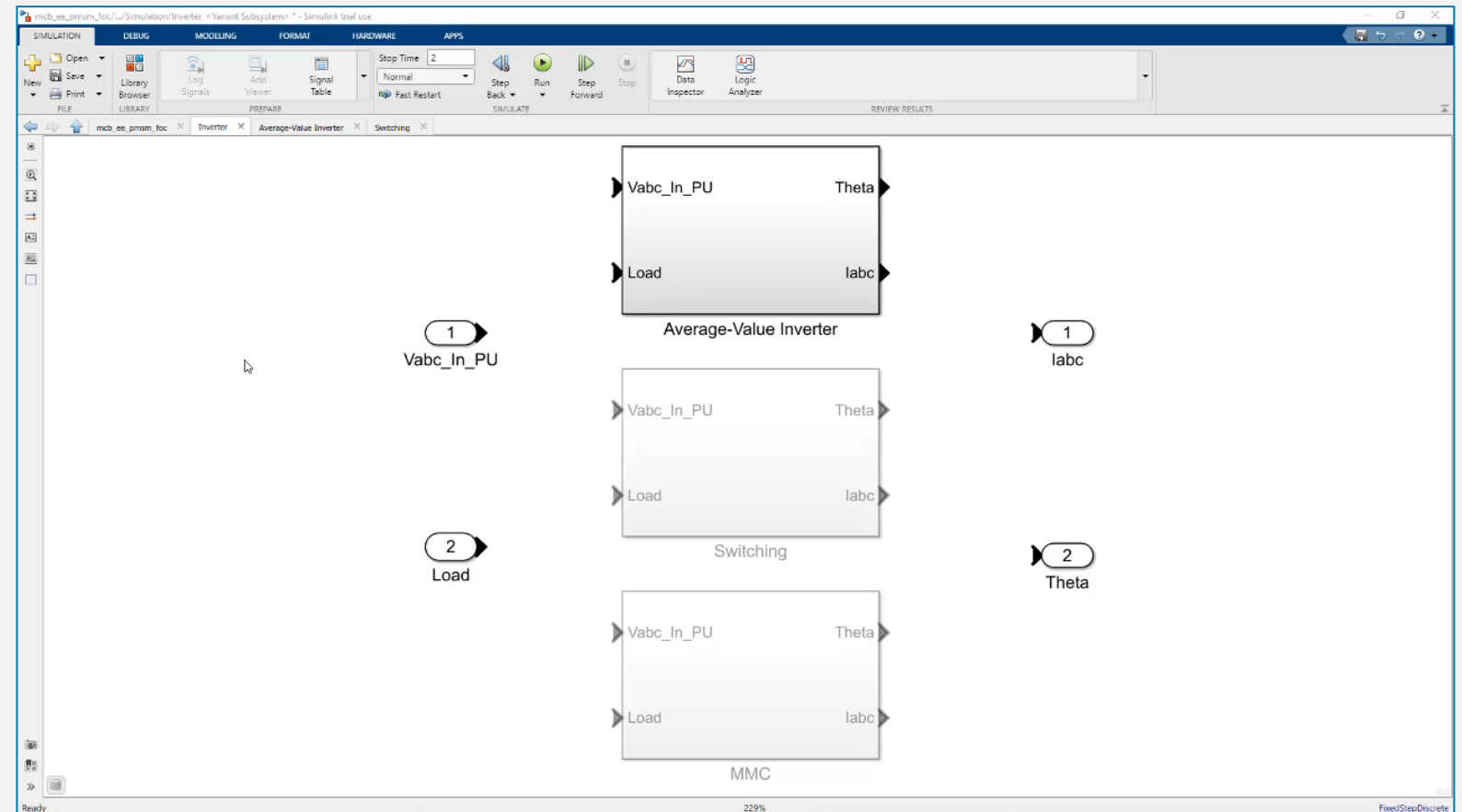
# Motor Parameterization and Plant Modeling

Variant approach for different plant fidelity level

Select motor type and spec

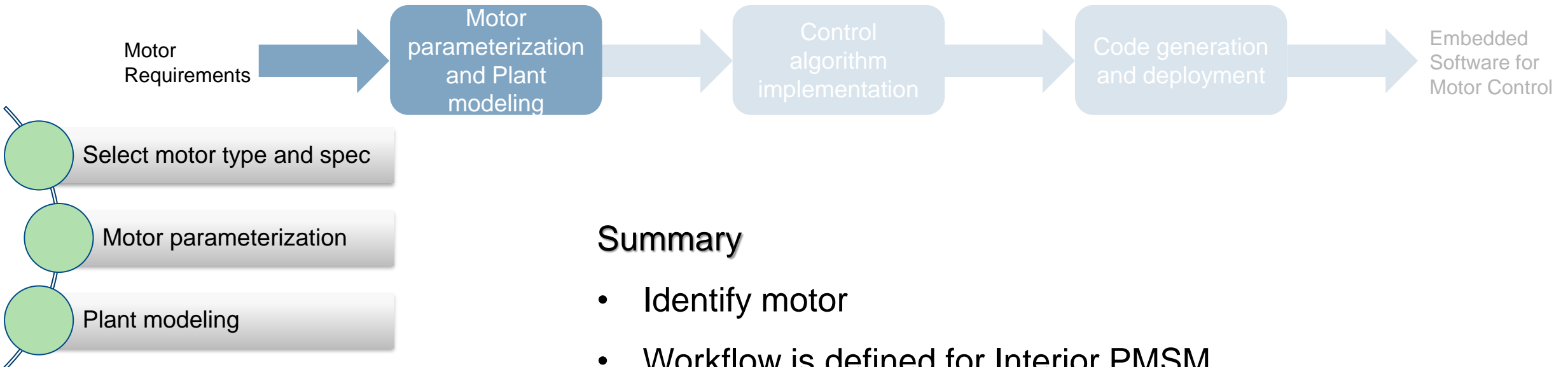
Motor parameterization

Plant modeling





# Motor Parameterization and Plant Modeling



## Summary

- Identify motor
- Workflow is defined for Interior PMSM
- Steps to parameterize Motor using datasheet, running instrumented tests, FEA tool or dyno test data
- Model motor and inverter with different fidelity levels

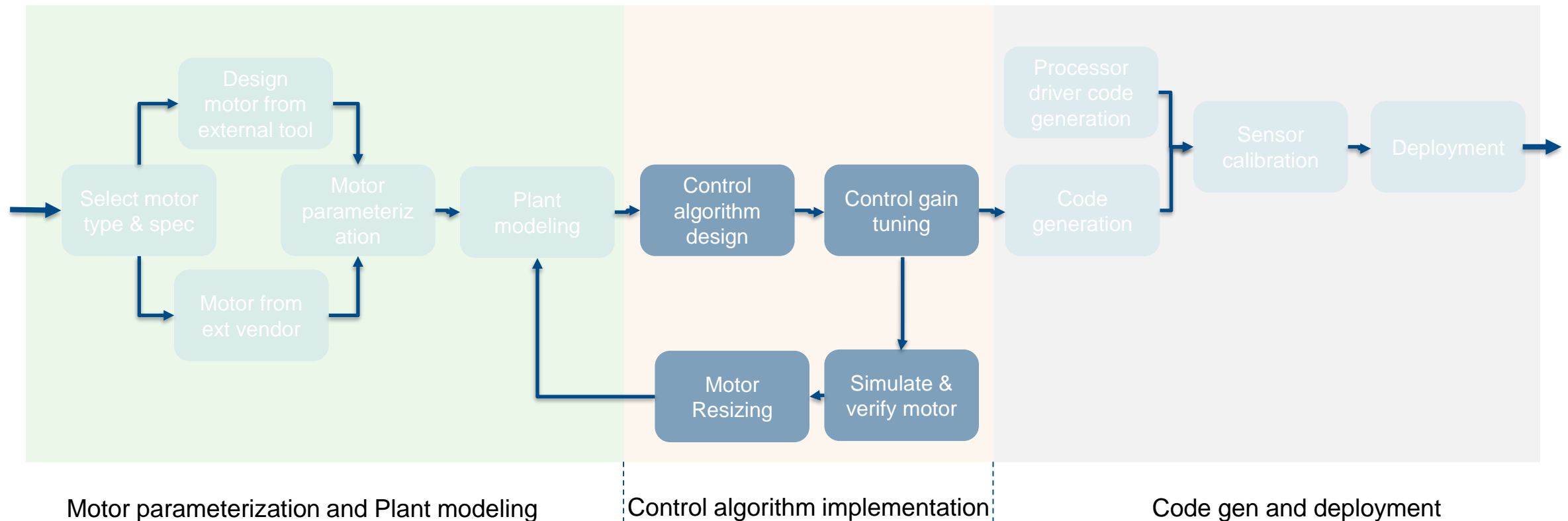
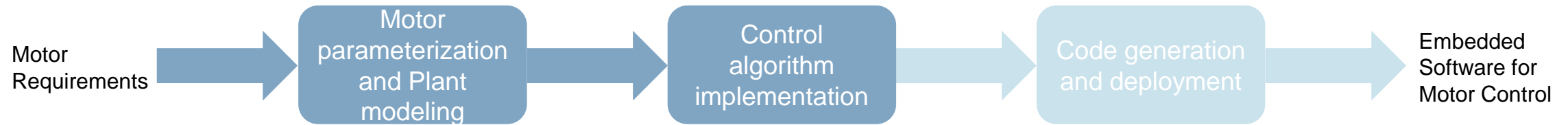


Modeling  
Engineer



How do I get motor parameters to develop my motor model?  
🔧 Use instrumented test or dyno test or FEA data for parameterizing the motor

# Proposed motor control software development workflow for EV



# Control Algorithm Implementation

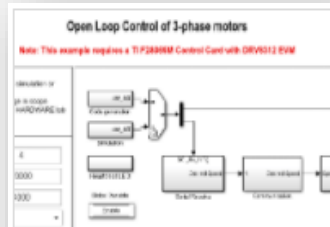
## Control algorithm design - Different control strategies

Control algorithm design

Control-loop gain tuning

Simulate & Verify

Motor resizing

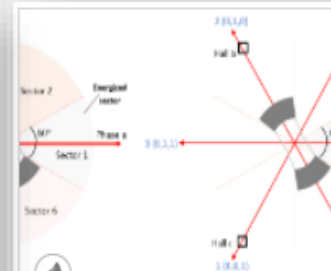


**Open Loop Control of 3-phase motors**  
 Note: This example requires a T1F2800M Control Card with DRV5012 EVM

Run 3-Phase AC Motors in Open-Loop Control and Calibrate ADC Offset

Uses open-loop control (also known as scalar control or Volts/Hz control) to run a motor. This technique varies the stator voltage

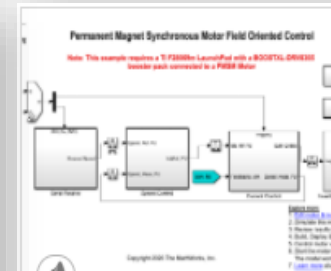
V/F control



**Six-Step Commutation of BLDC Motor Using Sensor Feedback**

Uses 120-degree conduction mode to implement the six-step commutation technique to control speed and direction of rotation of a


Six-step commutation for BLDC with Hall sensor



**Sensorless Field-Oriented Control of PMSM**

Implements the field-oriented control (FOC) technique to control the speed of a three-phase permanent magnet synchronous

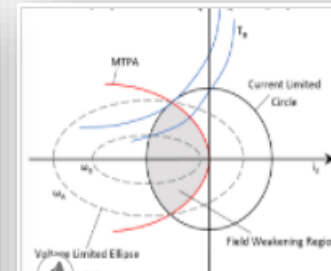
Field-oriented control



**Direct Torque Control of PMSM Using Quadrature Encoder or Sensorless Flux...**

Implements direct torque control (DTC) technique to control the speed of a three-phase permanent magnet synchronous motor (PMSM). Direct

Direct torque control



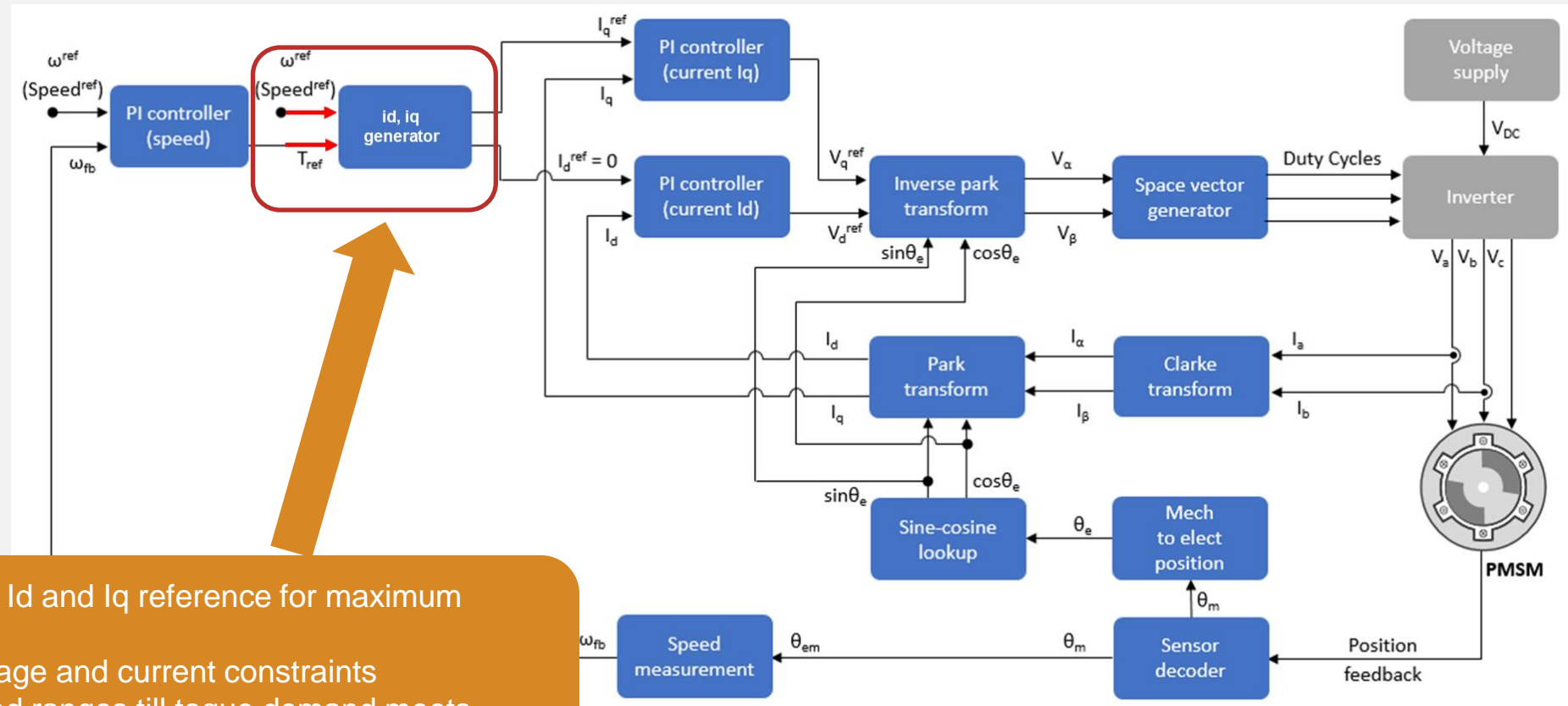
**Field-Weakening Control (with MTPA) of PMSM**

Implements the field-oriented control (FOC) technique to control the torque and speed of a three-phase permanent magnet

Field-weakening control

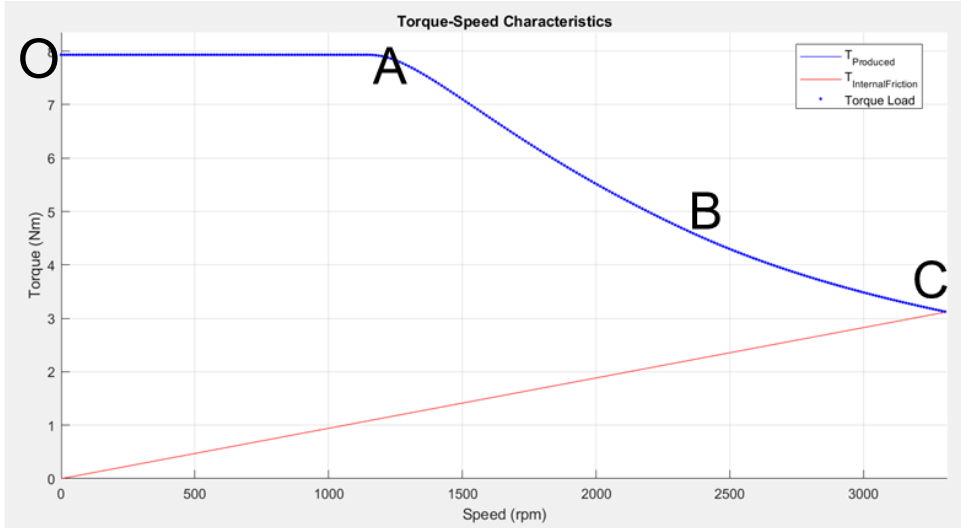
# Field-weakening control architecture with block to generate optimum control reference

- Control algorithm design
- Control-loop gain tuning
- Simulate & Verify
- Motor resizing

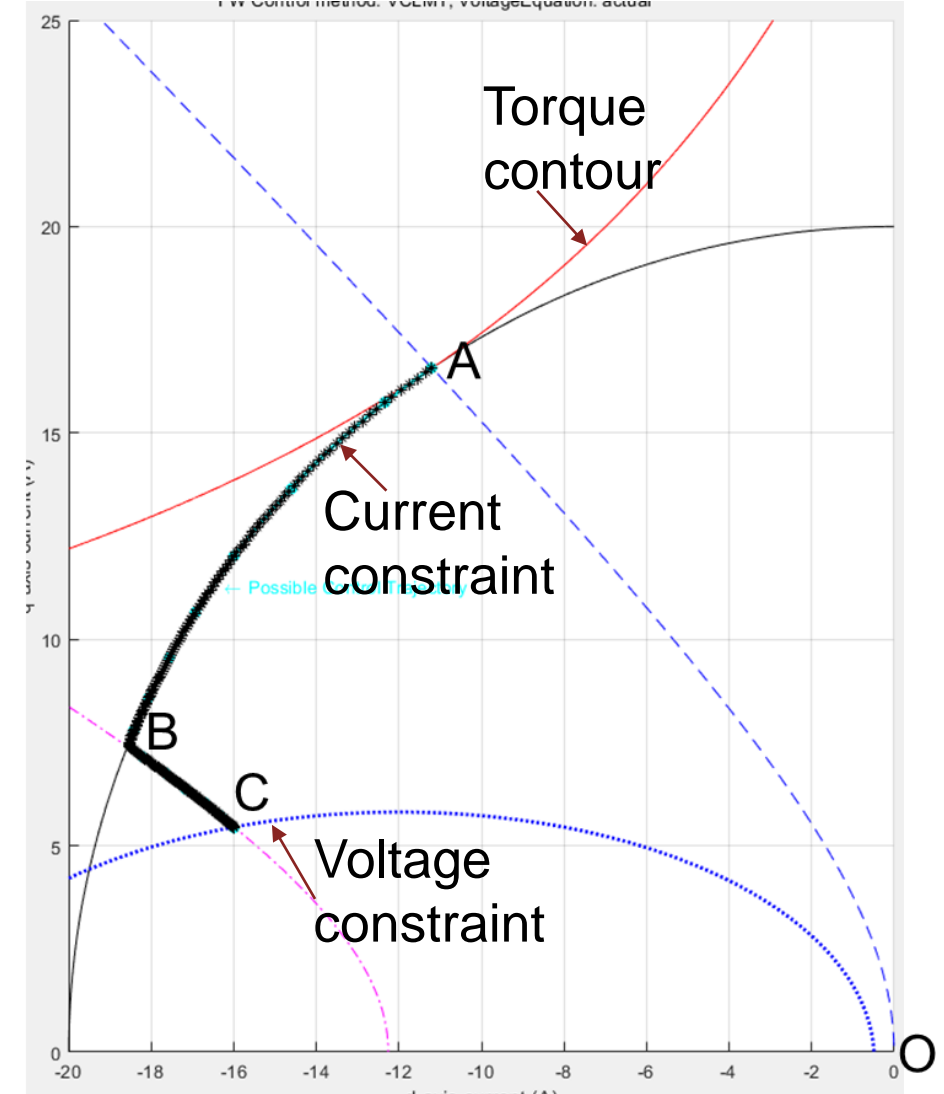


1. Output right  $I_d$  and  $I_q$  reference for maximum torque
2. Honour voltage and current constraints
3. Output speed ranges till torque demand meets frictional torque

# Compare critical operating points in Torque-Speed characteristic curve and $I_d$ \_ref, $I_q$ \_ref plot



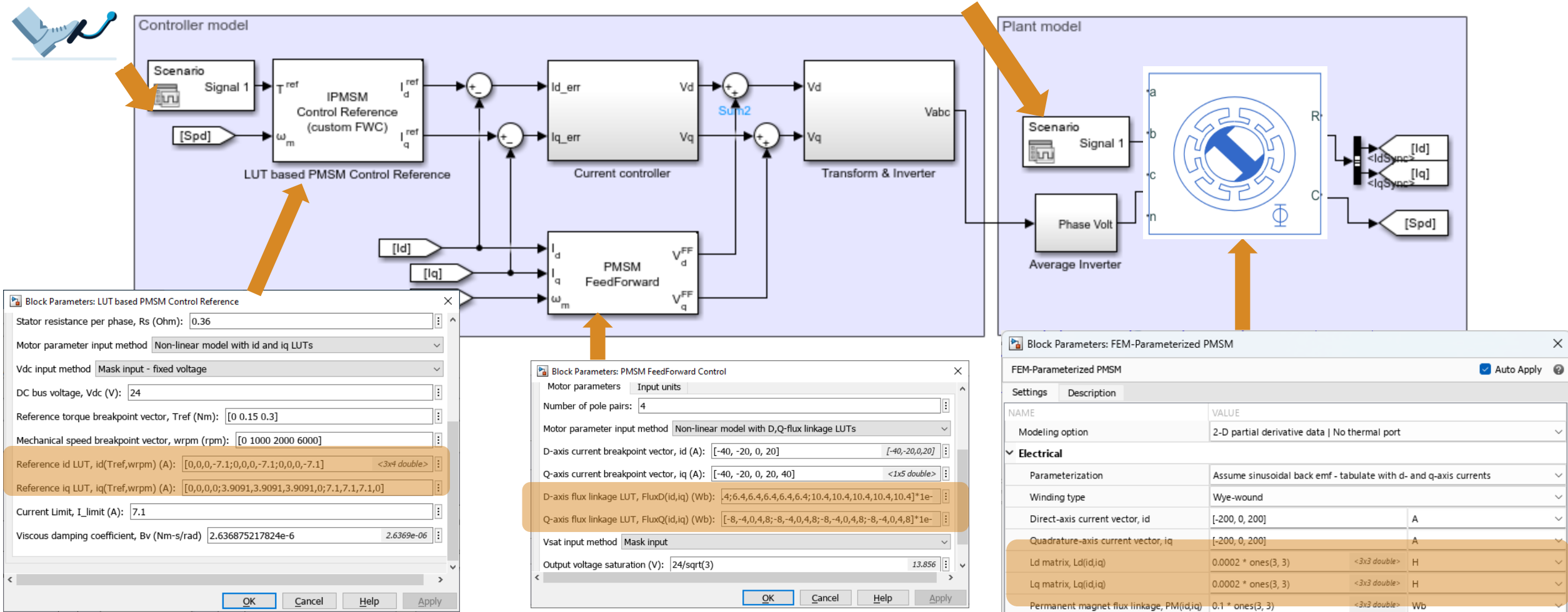
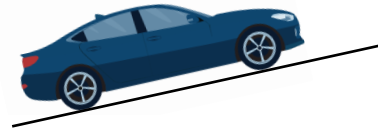
q-axis current



D-axis current

Region	
OA	Maximum torque per Ampere (MTPA)
AB	Field-weakening control (Beyond base speed)
BC	Field-weakening control honoring Maximum Torque per volt (MTPV)
CO	Field-weakening control honoring Voltage constraint limit
C	C is max achievable speed where frictional torque equals motor deliverable torque

# Implementing Field-weakening control with motor-dyno test data



**Block Parameters: LUT based PMSM Control Reference**

Stator resistance per phase,  $R_s$  (Ohm): 0.36

Motor parameter input method: Non-linear model with id and iq LUTs

Vdc input method: Mask input - fixed voltage

DC bus voltage, Vdc (V): 24

Reference torque breakpoint vector,  $T_{ref}$  (Nm): [0 0.15 0.3]

Mechanical speed breakpoint vector,  $w_{rpm}$  (rpm): [0 1000 2000 6000]

Reference id LUT,  $i_d(T_{ref}, w_{rpm})$  (A): [0,0,0,-7.1;0,0,0,-7.1;0,0,0,-7.1] <3x4 double>

Reference iq LUT,  $i_q(T_{ref}, w_{rpm})$  (A): [0,0,0,0;3.9091,3.9091,3.9091,0;7.1,7.1,7.1,0]

Current Limit,  $I_{limit}$  (A): 7.1

Viscous damping coefficient,  $B_v$  (Nm-s/rad): 2.636875217824e-6 2.6369e-06

OK Cancel Help Apply

**Block Parameters: PMSM FeedForward Control**

Motor parameters Input units

Number of pole pairs: 4

Motor parameter input method: Non-linear model with D,Q-flux linkage LUTs

D-axis current breakpoint vector,  $i_d$  (A): [-40, -20, 0, 20] [-40,-20,0,20]

Q-axis current breakpoint vector,  $i_q$  (A): [-40, -20, 0, 20, 40] [-40,-20,0,20,40]

D-axis flux linkage LUT,  $FluxD(i_d, i_q)$  (Wb): 4;6.4,6.4,6.4,6.4,6.4;10.4,10.4,10.4,10.4,10.4\*1e-6

Q-axis flux linkage LUT,  $FluxQ(i_d, i_q)$  (Wb): [-8,-4,0,4,8;-8,-4,0,4,8;-8,-4,0,4,8;-8,-4,0,4,8]\*1e-6

Vsat input method: Mask input

Output voltage saturation (V): 24/sqrt(3) 13.856

OK Cancel Help Apply

**Block Parameters: FEM-Parameterized PMSM**

FEM-Parameterized PMSM  Auto Apply

Settings Description

NAME	VALUE	
Modeling option	2-D partial derivative data   No thermal port	
<b>Electrical</b>		
Parameterization	Assume sinusoidal back emf - tabulate with d- and q-axis currents	
Winding type	Wye-wound	
Direct-axis current vector, $i_d$	[-200, 0, 200]	A
Quadrature-axis current vector, $i_q$	[-200, 0, 200]	A
Ld matrix, $L_d(i_d, i_q)$	0.0002 * ones(3, 3)	<3x3 double> H
Lq matrix, $L_q(i_d, i_q)$	0.0002 * ones(3, 3)	<3x3 double> H
Permanent magnet flux linkage, $PM(i_d, i_q)$	0.1 * ones(3, 3)	<3x3 double> Wb



# Field-weakening control for Interior PMSM

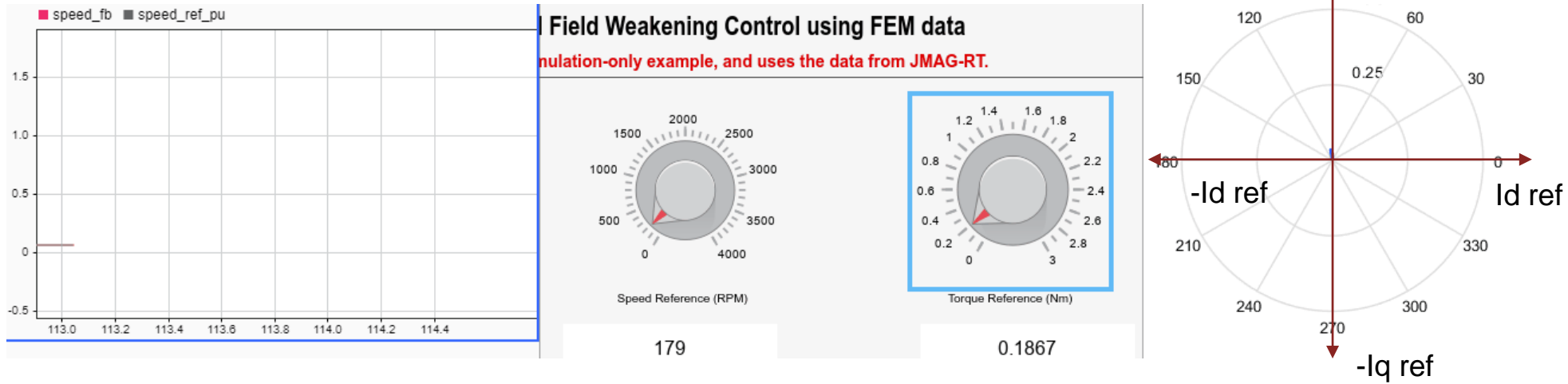
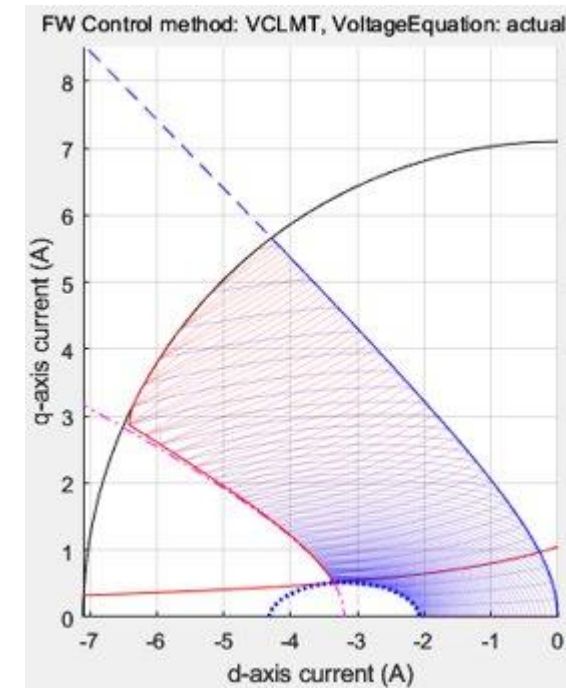
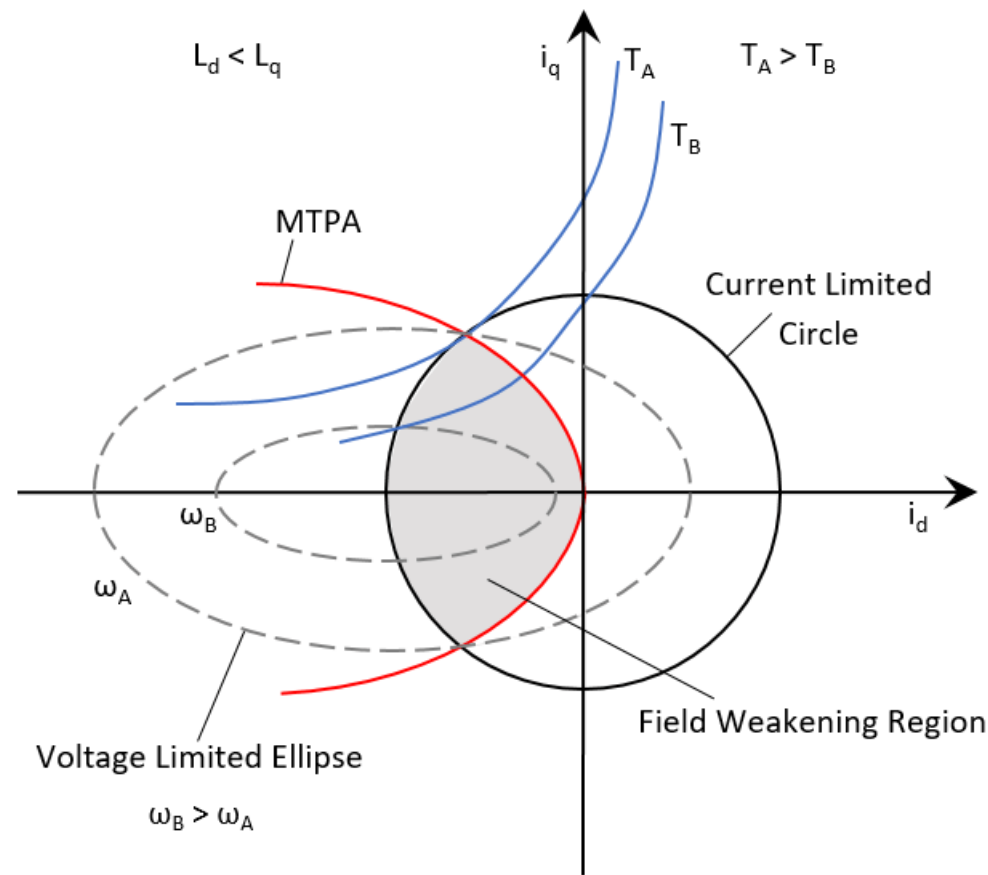
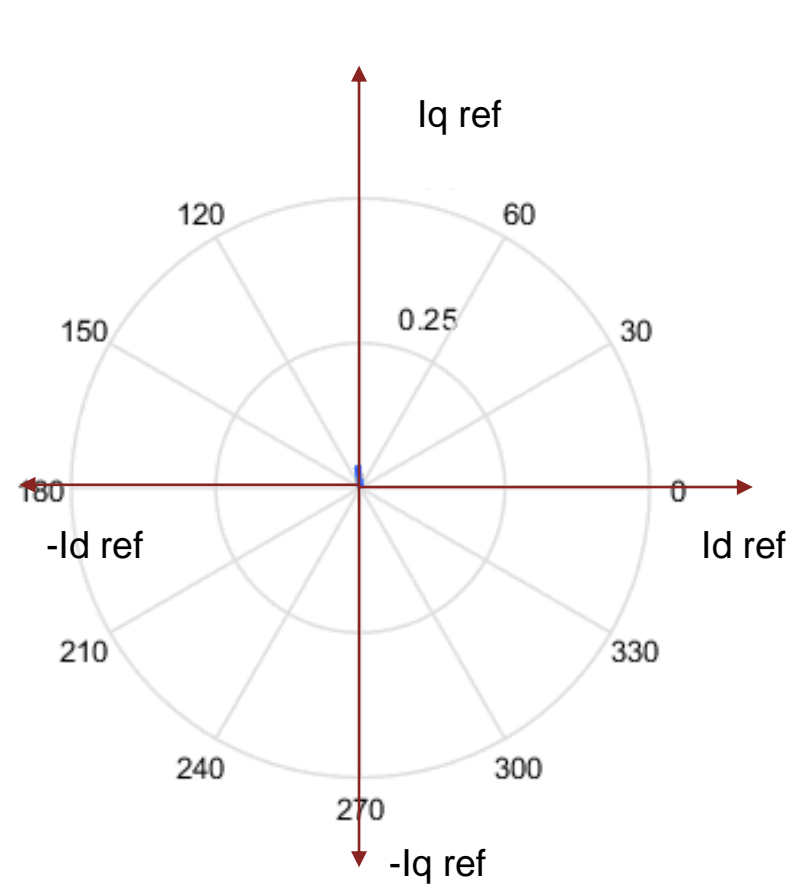


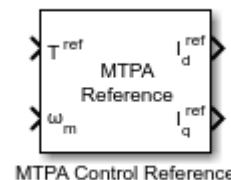
Fig. shows speed ref and measured speed with varying load and speed ref in Dashboard simulation

$I_d$  and  $I_q$  plot in  $dq$  axis shows the current trajectory with increase in speed

# Field-weakening control for Interior PMSM



Id and Iq plot in dq axis shows the MTPA path



Generates Id\_ref and Iq\_ref for MTPA, field-weakening control and MTPV

mcb\_pmsm\_fw\_host\_model\* - Simulink

SIMULATION DEBUB MODELING FORMAT APPS SWITCH

Stop Time inf

Normal

Step Back Run Step Forward Stop

Data Inspector

FILE LIBRARY PREPARE SIMULATE REVIEW RESULTS

# PMSM Field Weakening Control Host

HOST  
Serial Setup

**Note:**

1. Update workspace with variables used in [target model](#)
2. Select the serial port in 'Host Serial Setup' (Blue Color)
3. Use 'Motor Start / Stop' switch to control motor.
4. Input speed request using 'Reference Speed' block.
5. Observe the debug signals in scope.



Start / Stop  
Field Weakening Control

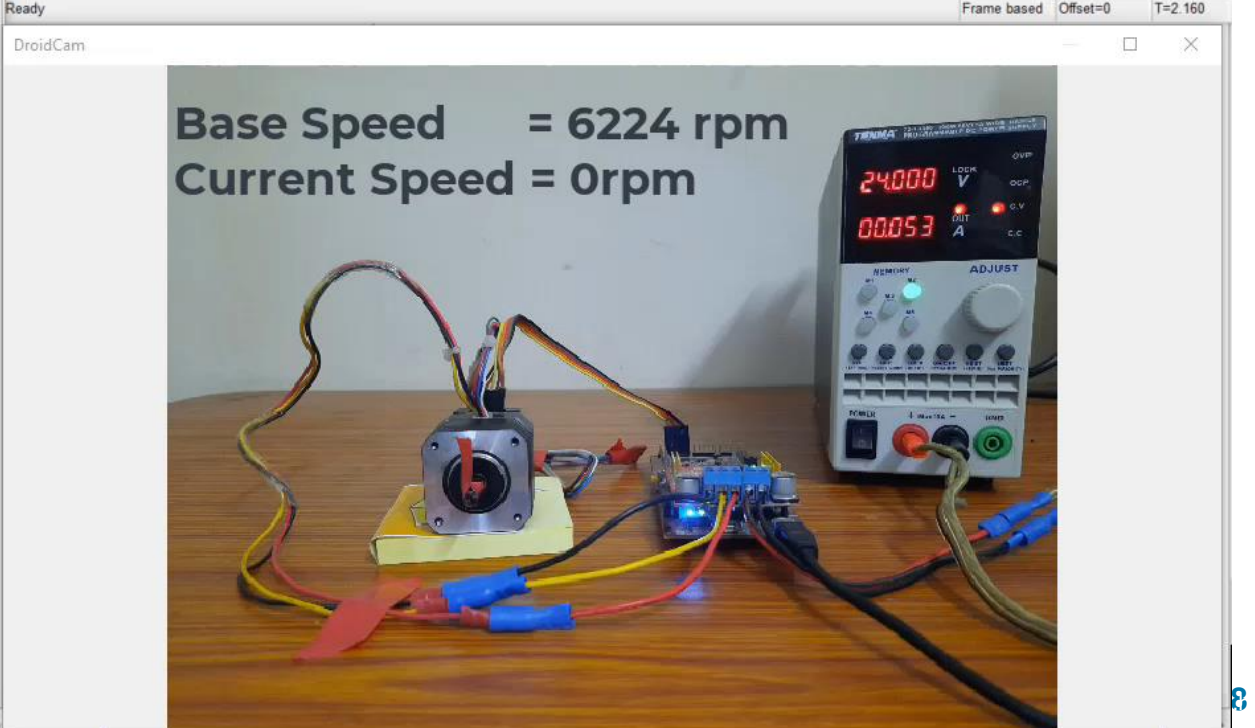


Start / Stop Motor

- Debug signals
- Speed\_ref & Speed\_feedback
  - Id\_ref & Id\_feedback
  - Iq\_ref & Iq\_feedback
  - Torque & Power
  - Ia & Ib



Copyright 2020 The MathWorks, Inc.



# Control Algorithm Implementation

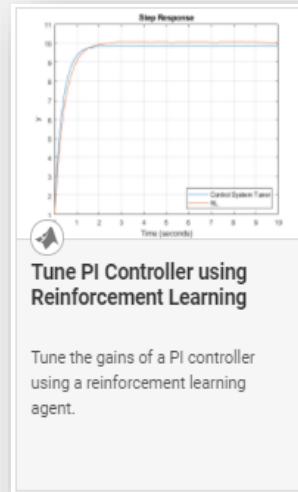
## Control algorithm design - Novel control strategies

Control algorithm design

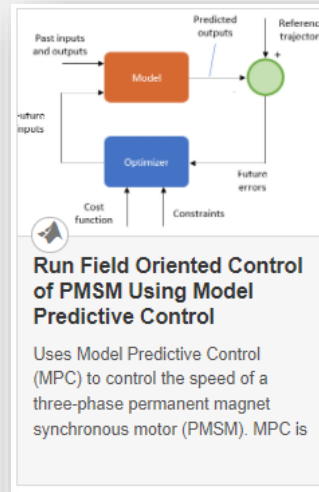
Control-loop gain tuning

Simulate & Verify

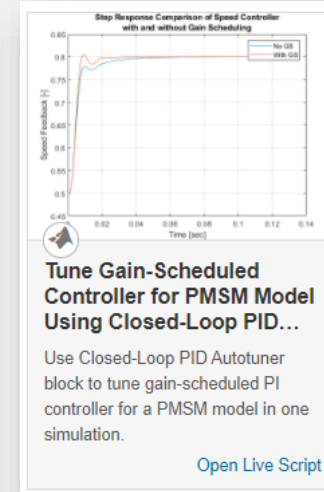
Motor resizing



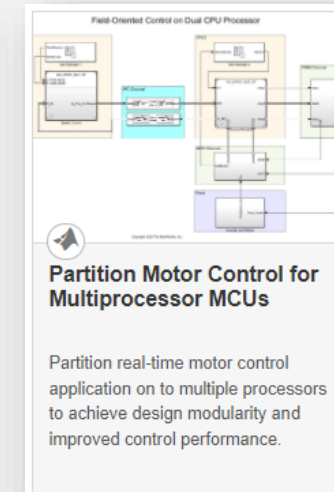
Reinforcement learning for PMSM



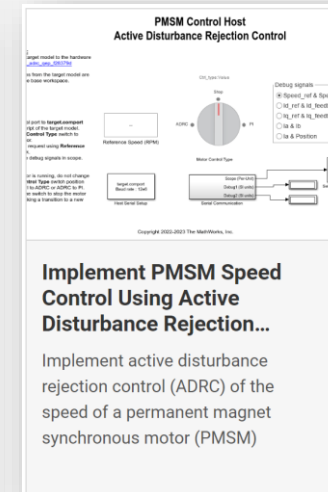
Model predictive control



Gain scheduling example







Motor control simulation for Multiprocessor MCUs

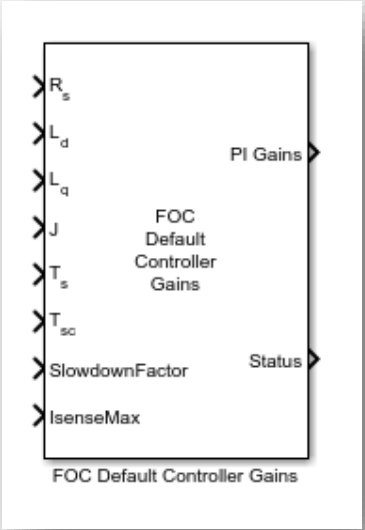


Active Disturbance Rejection Control for PMSM

# Control Algorithm Implementation

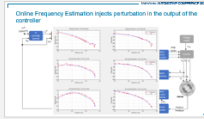
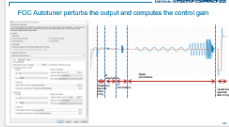
-  Control algorithm design
-  Control-loop gain tuning
-  Simulate & Verify
-  Motor resizing

## Control-loop gain tuning



Empirical Computation

Motor Control Blockset



**Tune 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

Motor Control Blockset and Simulink Control Design

**PID Tuner App with plant frequency response from hardware**

Online frequency estimation and PID tuner app

**Tune Field-Oriented Controllers Using SYSTUNE**

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

[Open Script](#)

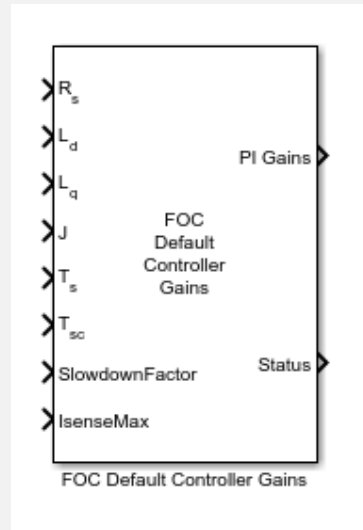
Classical Control Theory

Simulink Control Design


# Control Algorithm Implementation

## Control-loop gain tuning

- Control algorithm design
- Control-loop gain tuning
- Simulate & Verify
- Motor resizing



Empirical Computation



**Control Engineer**

How do I tune my Control loop gains to achieve the performance?

Use any of the methods listed here for tuning the control gains

Control Blockset

**Tune 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

Motor Control Blockset and Simulink Control Design

**PID Tuner App with plant frequency response from hardware**

Online frequency estimation and PID tuner app

Online frequency estimation and PID tuner app

**Field-Oriented Control Of Motor Velocity**

**Tune Field-Oriented Controllers Using SYSTUNE**

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

[Open Script](#)

Classical Control Theory

Simulink Control Design



# Control Algorithm Implementation

## Simulate & Verify

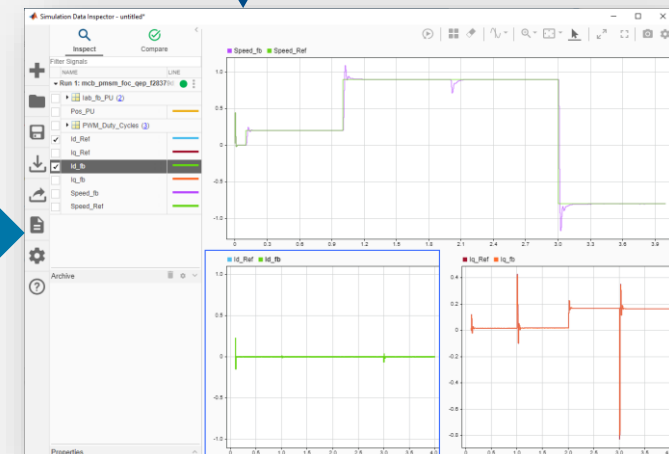
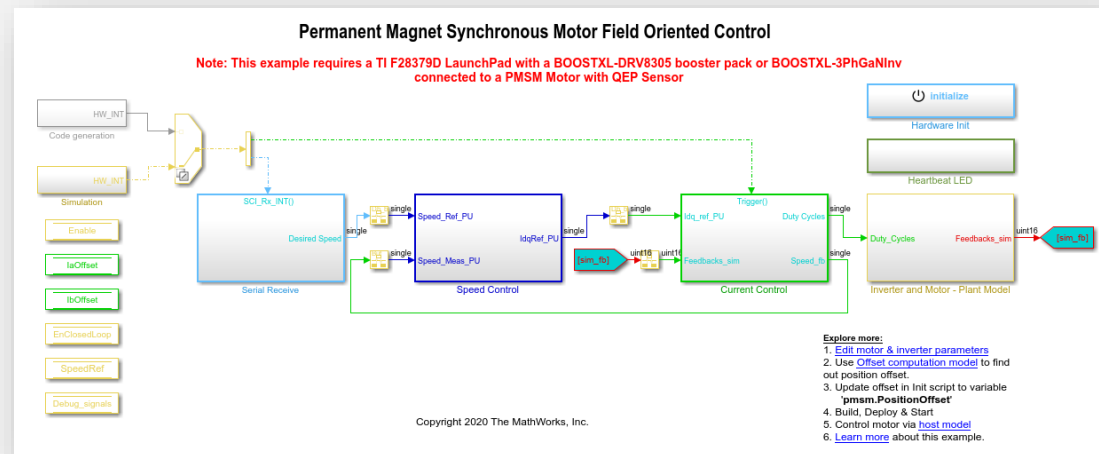
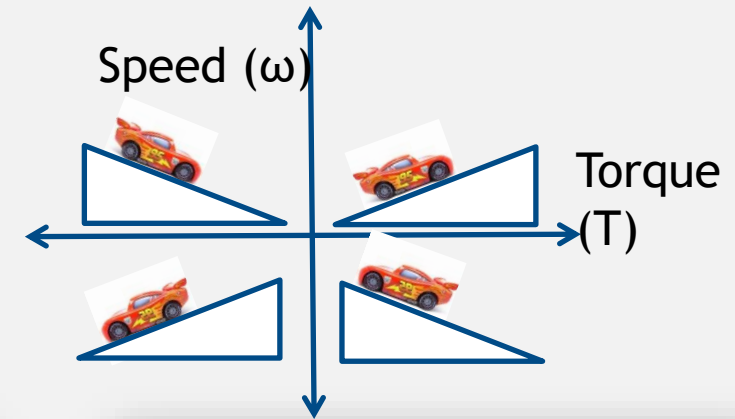
Simulate on 4-quadrant operations and validate the motor and control characteristics.

Control algorithm design

Control-loop gain tuning

Simulate & Verify

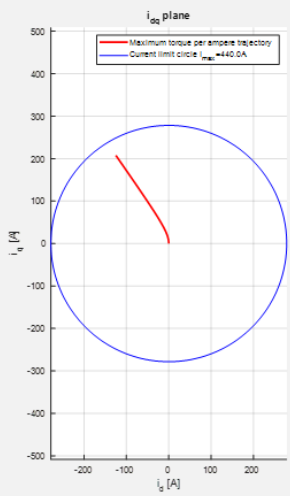
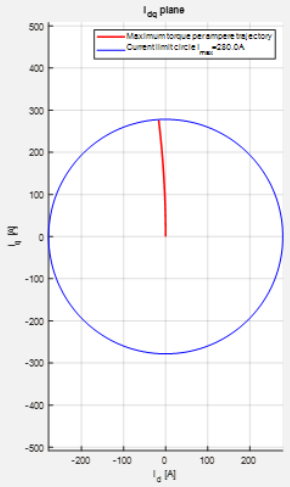
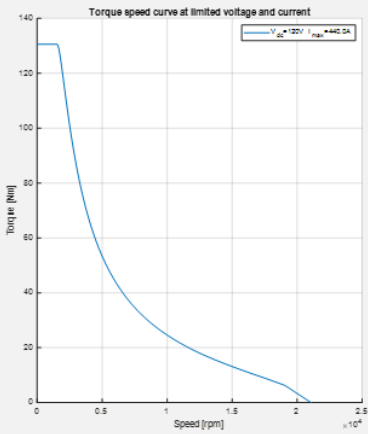
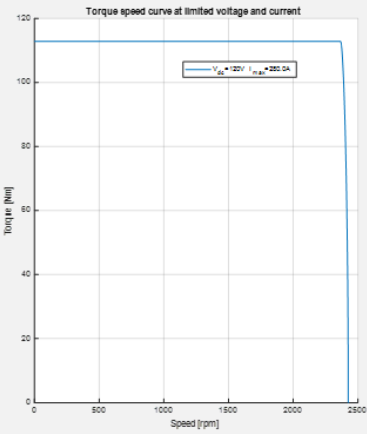
Motor resizing



# Control Algorithm Implementation

- Control algorithm design
- Control-loop gain tuning
- Simulate & Verify
- Motor resizing

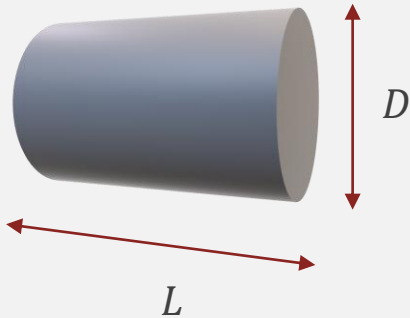
## Motor Resizing



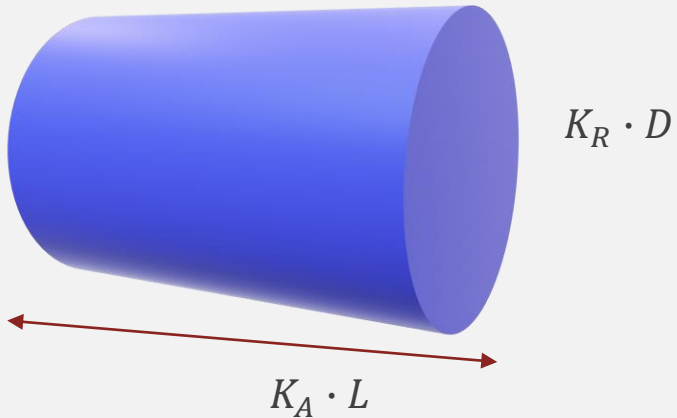
Existing characteristics

Proposed characteristics

### Initial Design



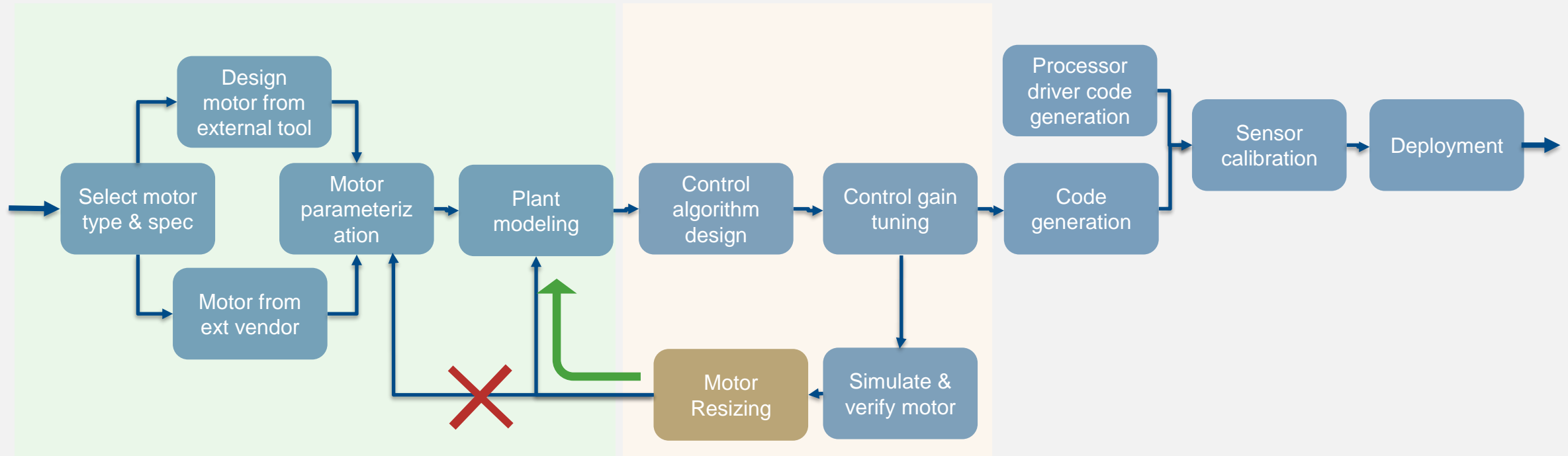
### Resized Design



- Where,
- o  $K_A$  is the axial resize factor
  - o  $K_R$  is the radial resize factor
  - o  $K_W$  is the rewinding factor
  - o  $D$  is the diameter of motor
  - o  $L$  is the length of motor

# Control Algorithm Implementation

## Motor Resizing

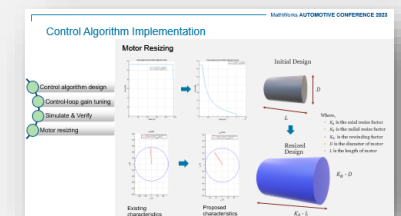


Motor parameterization and Plant modeling

Control algorithm implementation

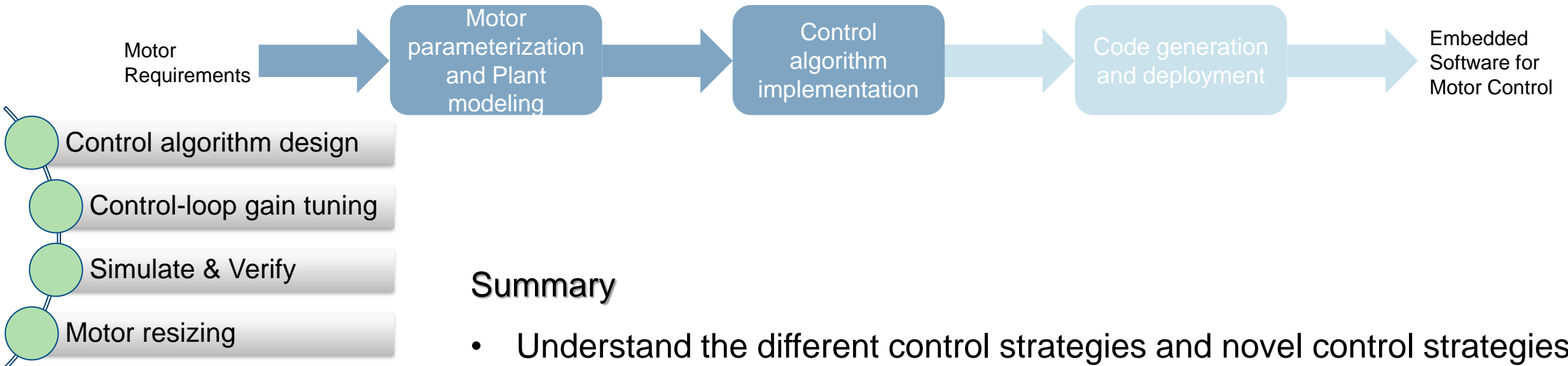
Code gen and deployment

- For resizing motor, No need to run dyno test again.
- Reuse the motor parameters from dyno test with a factor.
- This saves laboratory test time and money.



Motor Resizing

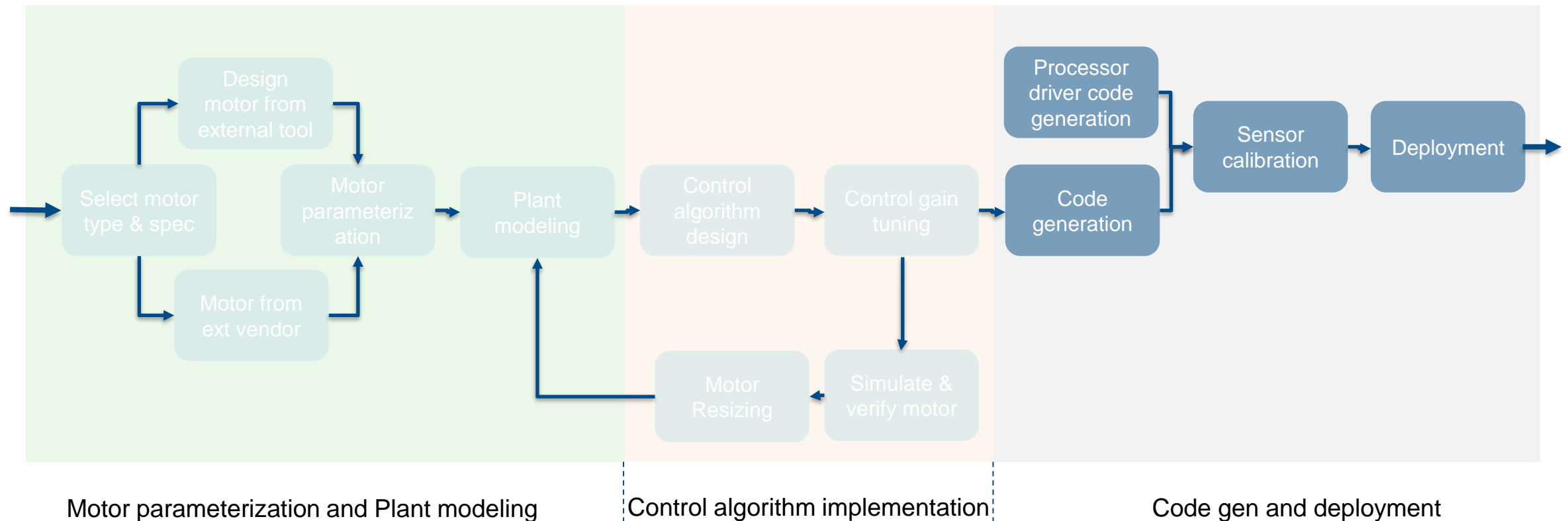
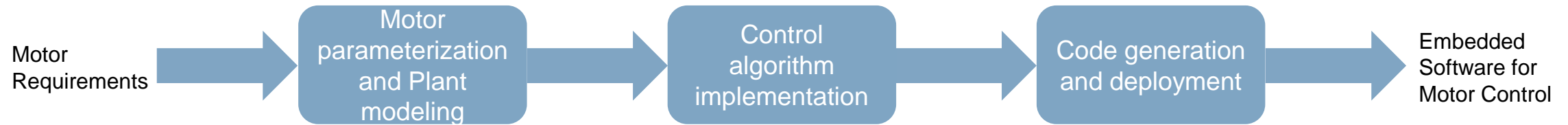
# Control Algorithm Implementation



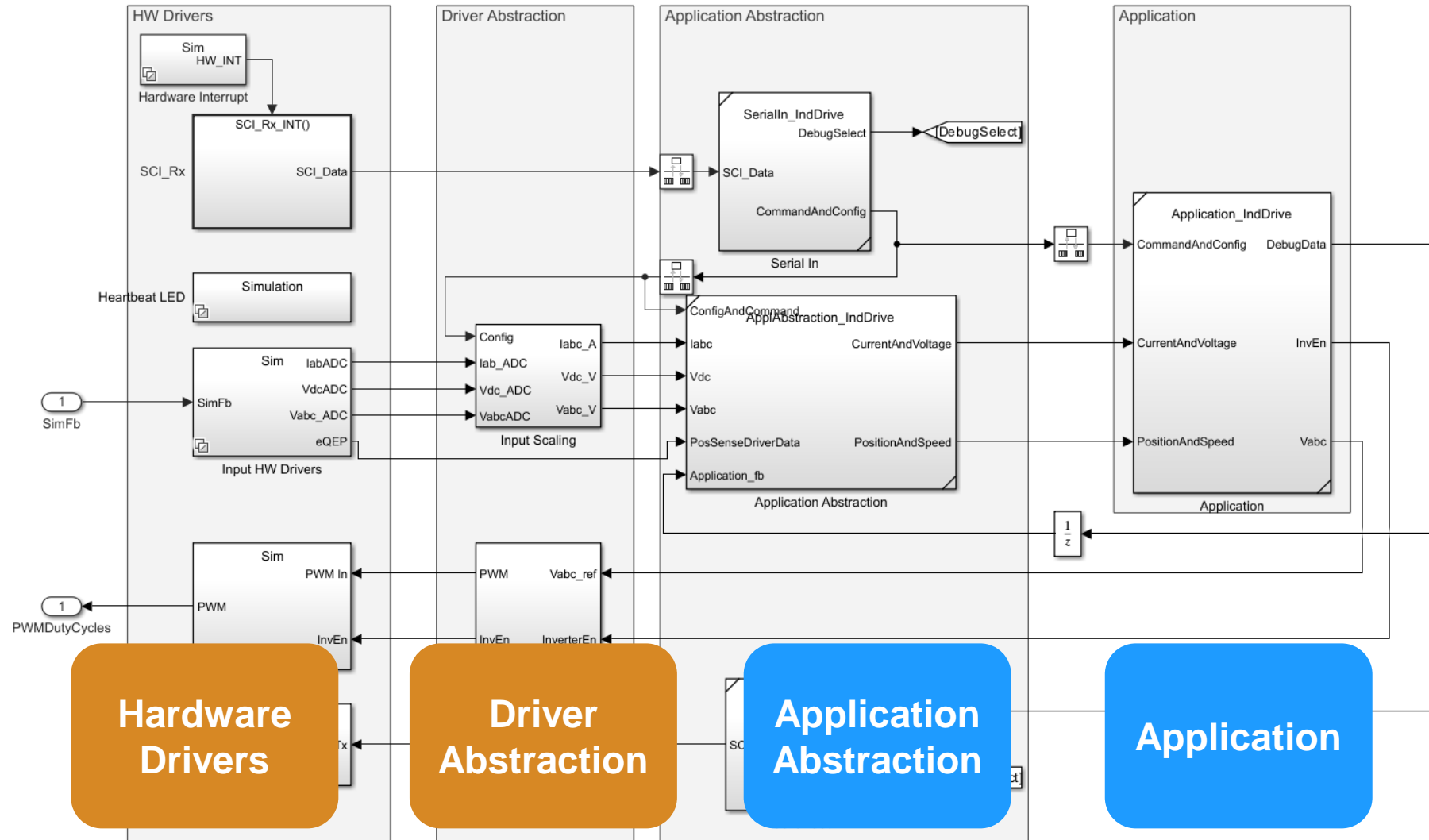
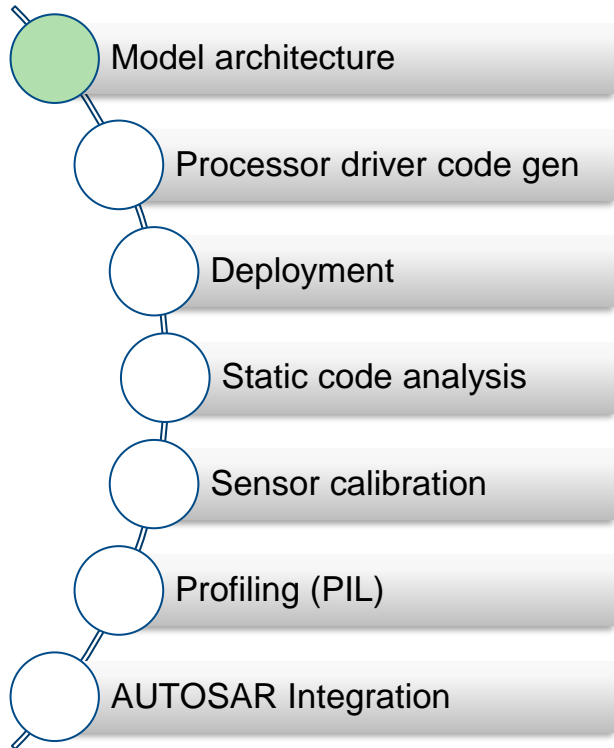
## Summary

- Understand the different control strategies and novel control strategies
- Discussed the Field-weakening control (MTPA, MTPV)
- Discussed the control gain tuning strategies
- Discussed the motor resizing

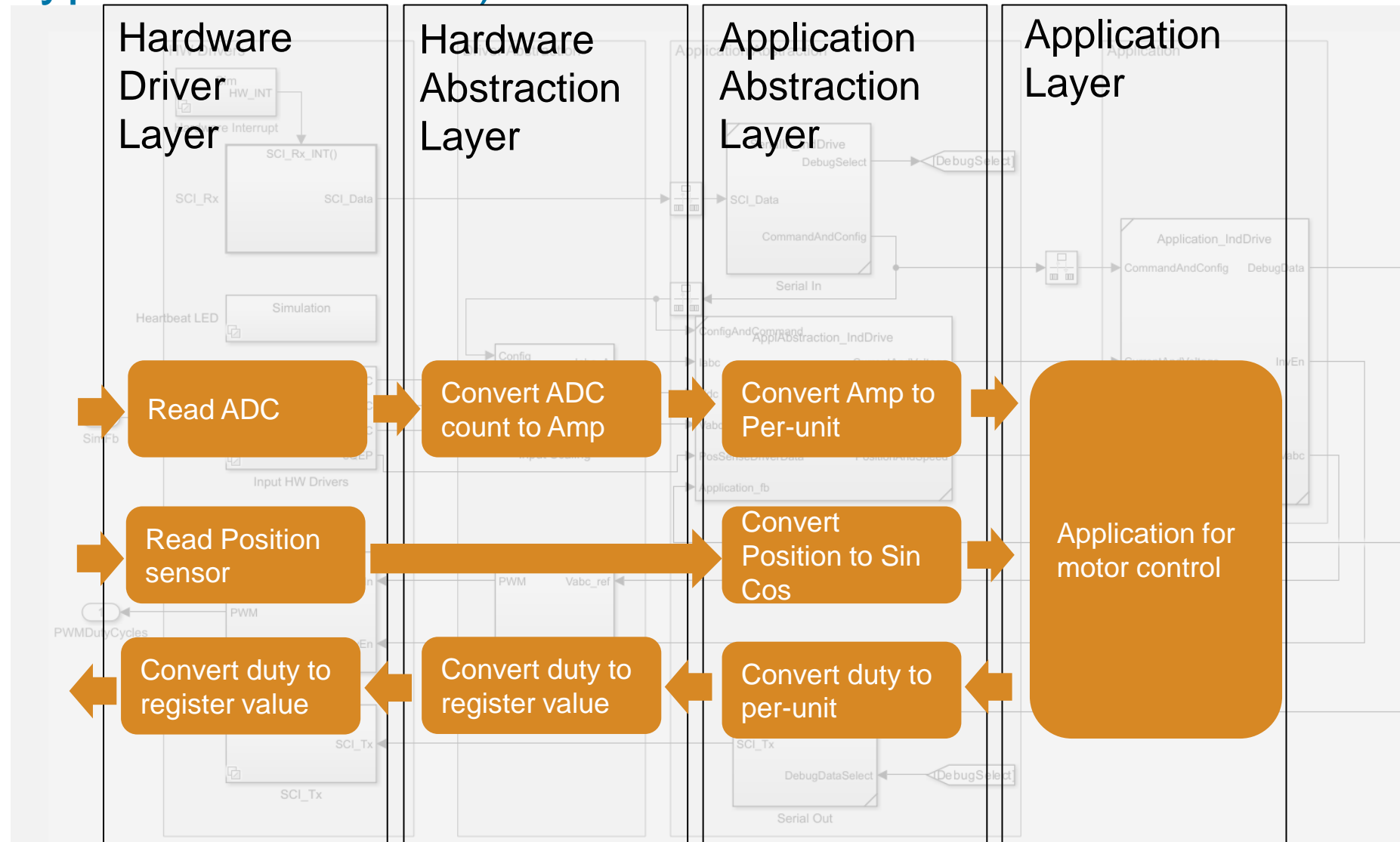
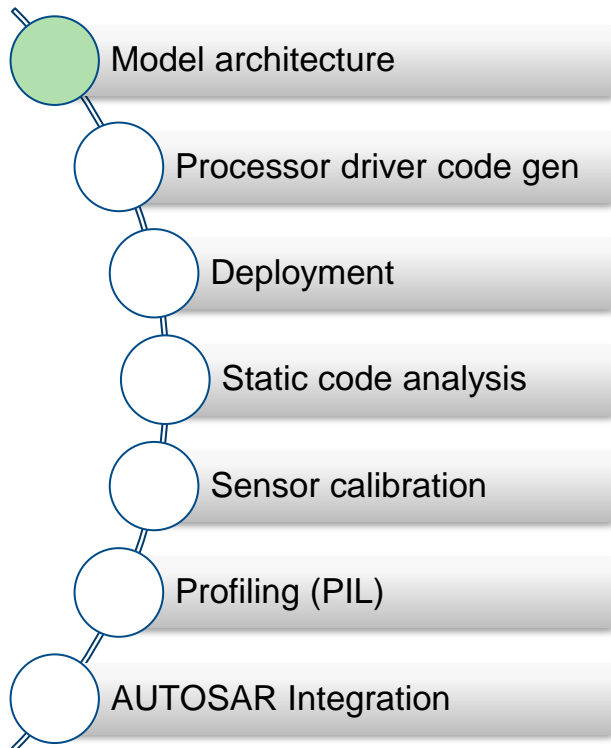
# Proposed motor control software development workflow for EV



# Model architecture with layers to assist porting algorithm between different hardware (Proto-type to Production)



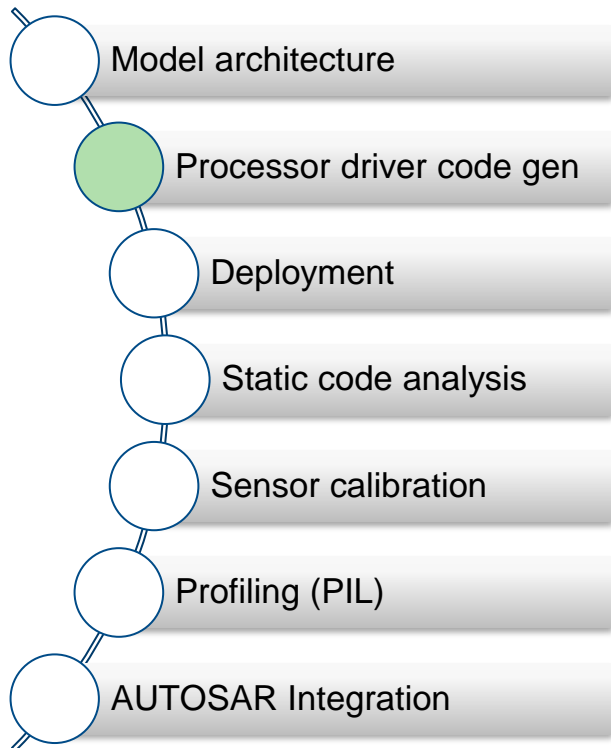
# Model architecture to assist porting algorithm between different hardware (Proto-type to Production)





# Processor driver code from Hardware support package or Driver block

ADC, PWM, Interrupt trigger, Serial communication blocks



Simulink Library Browser

instrument control

C2000 Microcontroller Blockset/F2837xD

C2806x	F2837x/07x/38x A_IN0 ADC	C28x f0 CAN RCV Msg CAN Receive	C28x Msg CAN XMT CAN Transmit	CLAMath sin(u) Y CLA Math	function 1 CLA Subsystem
C280x	F2837x/07x CLATaskTrigger CLA Task	C28x STS CMPSS1H CMPSS	C28x DAC-A DAC	F2837x GPIOx Digital Input	F2837x GPIO DO Digital Output
C281x	C28x TS eCAP	F2837x/07x/004x ePWM1 ePWM	C28x qposcnt eQEP	C28x RD I2C RCV	C28x WD I2C XMT
C2833x	C28x Out IPC Receive	C28x Status IPC Transmit	C28x Data SCI RCV	C28x Data SCI XMT	F2837x/07x DFSTS SDFM-1 SDFM
C2834x	C28x Sw Int Trigger PIEIFR12.INT8	C28x Tx Controller Transfer Chip select: SPICS	C28x Rx SPI Receive Chip select: SPICS	C28x Tx SPI Transmit Chip select: SPICS	C28x Status Watchdog

Software Interrupt Trigger

SPI Controller Transfer

SPI Receive

SPI Transmit

Watchdog

Block Parameters: ADC

ADC Type 3-5 (mask) (link)

Configures the Type 3 to Type 5 ADC to output data collected from the ADC pins on the processor.  
SOC: Start of Conversion  
EOC: End of Conversion

SOC Trigger  Input Channels

ADC Module **A**

ADC Resolution **12-bit (Single-ended input)**

SOC trigger number **SOC0**

SOCx acquisition window  
15

SOCx trigger source **Software**

ADCINT will trigger SOCx **No ADCINT**

Sample time:  
0.001

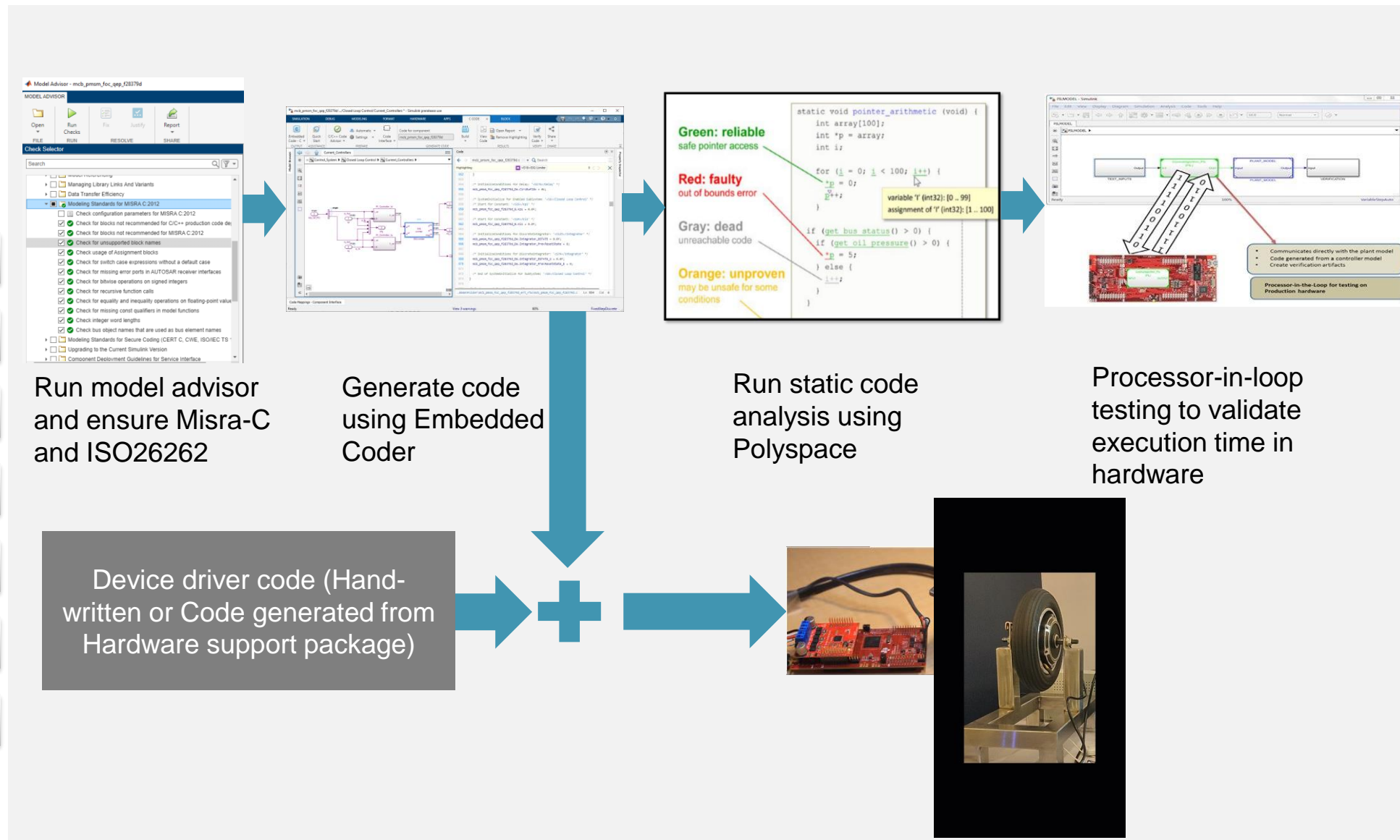
Data type: **uint16**

Post interrupt at EOC trigger

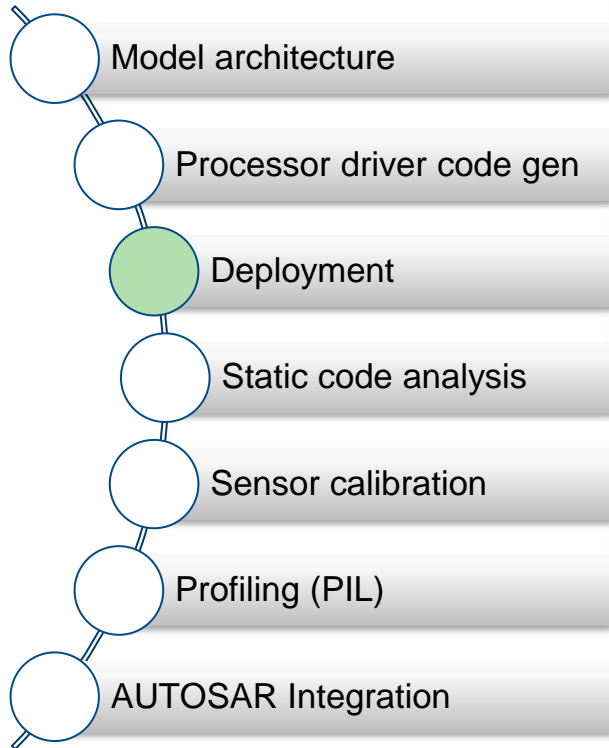
OK Cancel Help Apply

# Generate c-code or hdl-code and integrated with peripheral driver code

- Model architecture
- Processor driver code gen
- Deployment
- Static code analysis
- Sensor calibration
- Profiling (PIL)
- AUTOSAR Integration



# Processor driver code from Hardware support package or Driver block



## Processor peripheral driver code generation (ADC, PWM, Interrupt trigger, Serial communication)



**Embedded Coder Support Package for Texas Instruments C2000 Processors** by MathWorks Embedded Coder Team **STAFF**

Generate code optimized for C2000 MCU.  
Embedded Coder® Support Package for Texas Instruments C2000™ Processors enables you to run Simulink® models on TI C2000 MCUs. Embedded Coder automatically generates C code for your algorithms and



TI



**Field-Oriented Control of PMSM Using NXP™ S32K144 Kit** version 1.0 by Shivaprasad Narayan **STAFF**

The workflow demonstrates Field Oriented Control of a Permanent Magnet Synchronous motor using NXP™ MCSPT1AK144: S32K144 Development Kit  
FOC-of-PMSMField-Oriented Control of Permanent Magnet Synchronous Motor Using NXP™ S32K144 Development kitThis example implements a motor control system using the NXP™ MCSPT1AK144 hardware. The

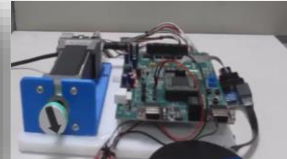


NXP



**Demo for Motor Control Deployment on Microchip Controllers** version 1.0.0 by Brian McKay **STAFF**

Demo used in MathWorks-Microchip joint webinar: Deploying Motor Control Algorithms on Microchip dsPIC, PIC32, and SAM Controllers.  
which includes a dsPIC33E Digital Signal Controller.The demo also require you to download and install the free add-on MPLAB Device Blocks for Simulink: dsPIC, PIC32, and SAM MCU's.View the webinar for a



Microchip



**Embedded Coder Support Package for STMicroelectronics STM32 Processors** by MathWorks Embedded Coder Team **STAFF**

Generate code optimized for STMicroelectronics STM32 Processor boards



STM



**Embedded Coder Support Package for Infineon AURIX TC4x Microcontrollers**

by MathWorks Embedded Coder Team **STAFF**  
Generate code optimized for Infineon AURIX TC4x Microcontrollers



Infineon



**HDL Coder Support Package for Xilinx Zynq Platform** by MathWorks HDL Coder Team **STAFF**

Generate code for the FPGA portion of the Zynq-7000 SoC.  
HDL Coder™ Support Package for Xilinx® Zynq™-7000 Platform supports the generation of IP cores that can be used in FPGA designs using Xilinx Vivado® or Xilinx ISE. When used in combination  
Hardware Support



Xilinx  
FPGA



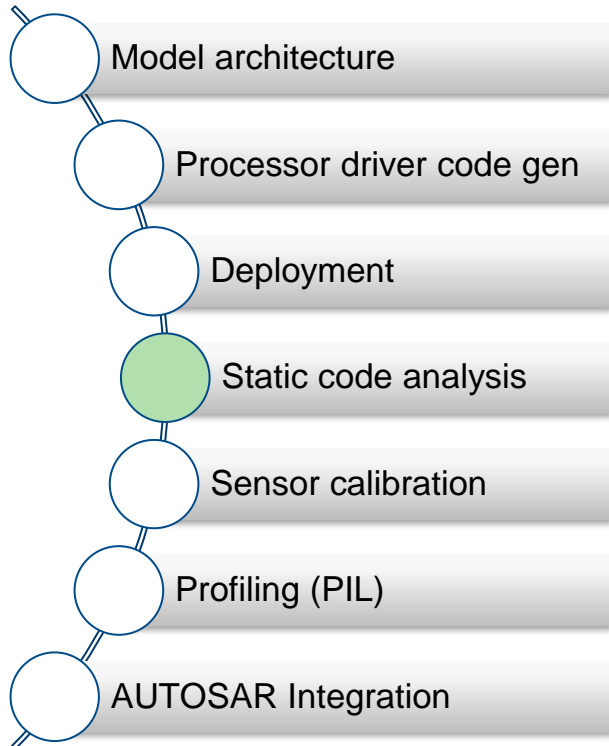
**Simulink Real-Time Target Support Package**

by MathWorks Simulink Real Time Target Team **STAFF**  
Tools to compile a real-time application that runs on a Speedgoat target computer



Speedgoat

# Perform static-code analysis tool using Polyspace



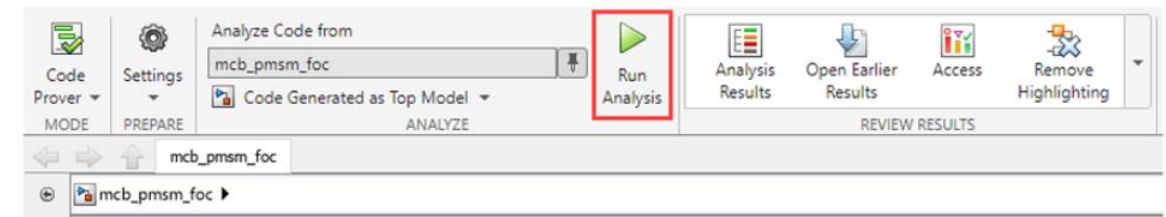
## Analyze and Verify Motor Control Algorithms Using Polyspace R2023b

This example uses the Polyspace® static code analysis tools to analyze and verify Simulink® models containing motor control algorithms. Static code analysis is a software verification technique that analyzes source code for quality, reliability, and security without executing the code. This approach uses robust error detection routines (that include checks for critical run-time errors) to identify bugs and defects. This ensures compliance with common coding standards.

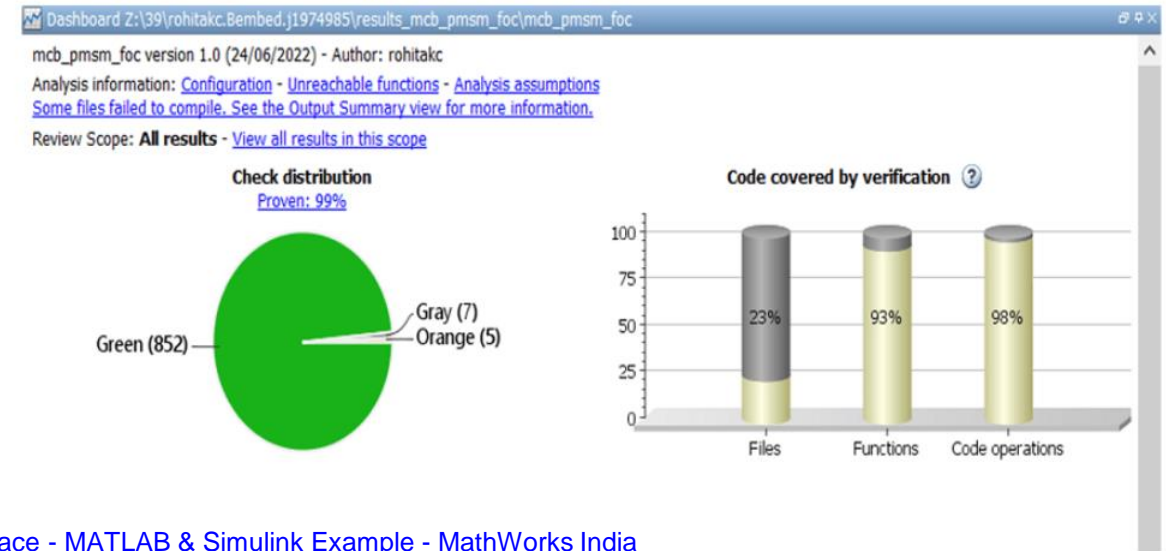
This example uses:

- Polyspace Bug Finder
- Polyspace Code Prover
- Motor Control Blockset

3. Click **Run Analysis** to start running the Code Prover tool.



4. After the tool execution completes, click **Polyspace > Analysis Results** to open the code analysis results in the Polyspace app.





# Calibrate position sensors by spinning the motor in hardware using reference examples

Model architecture

Processor driver code gen

Deployment

Static code analysis

Sensor calibration

Profiling (PIL)

AUTOSAR Integration

**Offset Computation for QEP**

Note: This example requires a TI F28069m controller card connected to a PMSM Motor with QEP.

**Quadrature Encoder Offset Calibration for PMSM Motor**

Calculates the offset between the d-axis of the rotor and encoder index pulse position as detected by the quadrature encoder sensor. The

Quadrature encoder calibration

**Hall Sensor Sequence Calibration of BLDC Motor**

Calculates the Hall sensor sequence with respect to position zero of the rotor in open-loop control.

Identify Hall sensor sequence

**Offset Computation with Hall sensor**

Note: This example requires a TI F28069m controller card connected to a PMSM Motor with Hall Sensor.

**Hall Offset Calibration for PMSM Motor**

Calculates the offset between the rotor direct axis (d-axis) and position detected by the Hall sensor. The field-oriented control (FOC)

Calibrate Hall sensor

**Monitor Resolver Using Serial Communication**

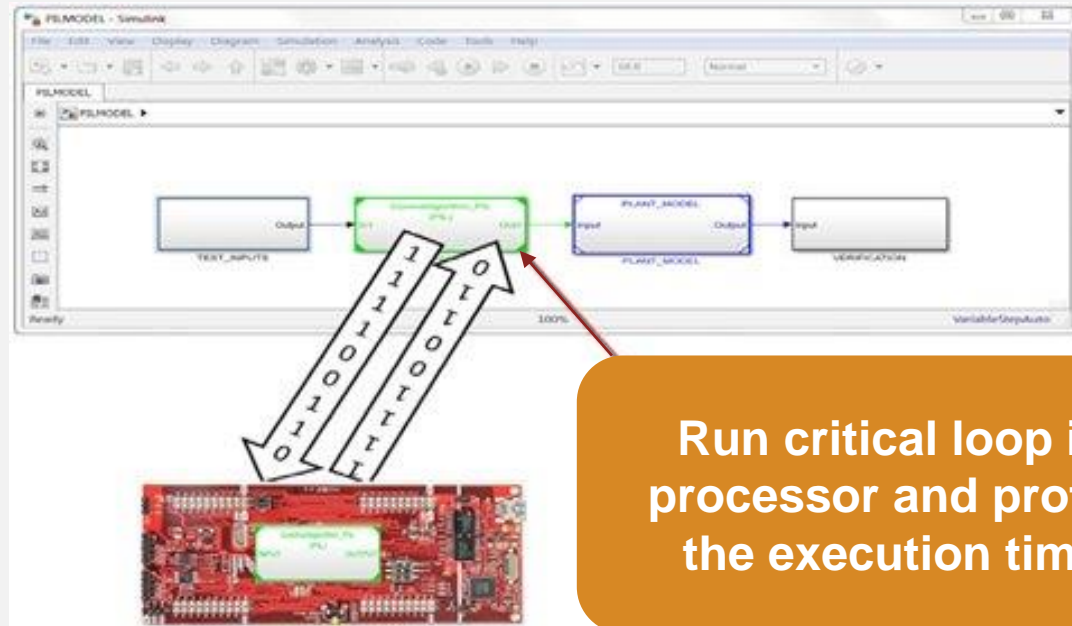
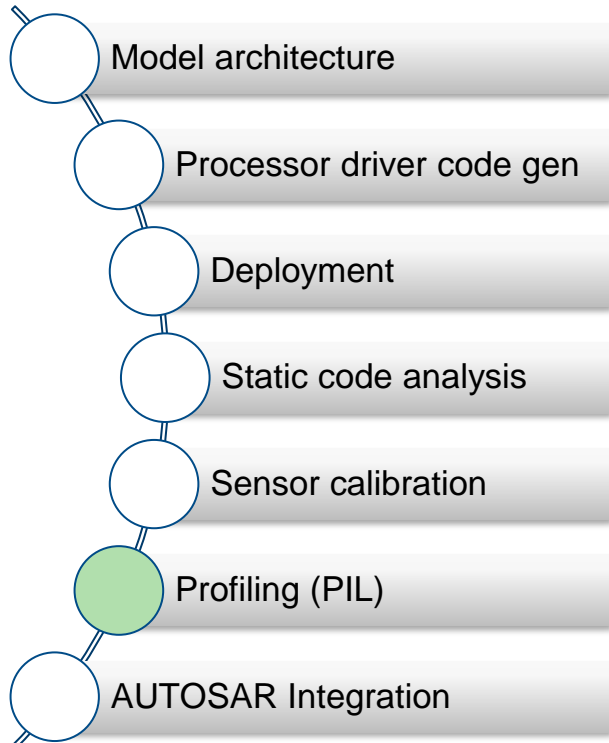
Operates the resolver sensor to measure the rotor position. The resolver consists of two orthogonally placed stator windings

Resolver with Square-pulse carrier frequency

<https://in.mathworks.com/help/mcb/gs/quadrature-encoder-offset-calibration-pmsm-motor.html>

<https://in.mathworks.com/help/mcb/gs/hall-sensor-sequence-calibration-blcd-motor.html>

# Perform profiling test and measure the execution time of critical control loop



## Report for mcb\_pmsm\_foc\_sim/Current

based on data collected from a SIL or PIL execution. Execution times are calculated from data collected from a SIL or PIL test harness or inside the code generated for each component. See [Code Execution](#)

[Profiling](#) for more information.

### 1. Summary

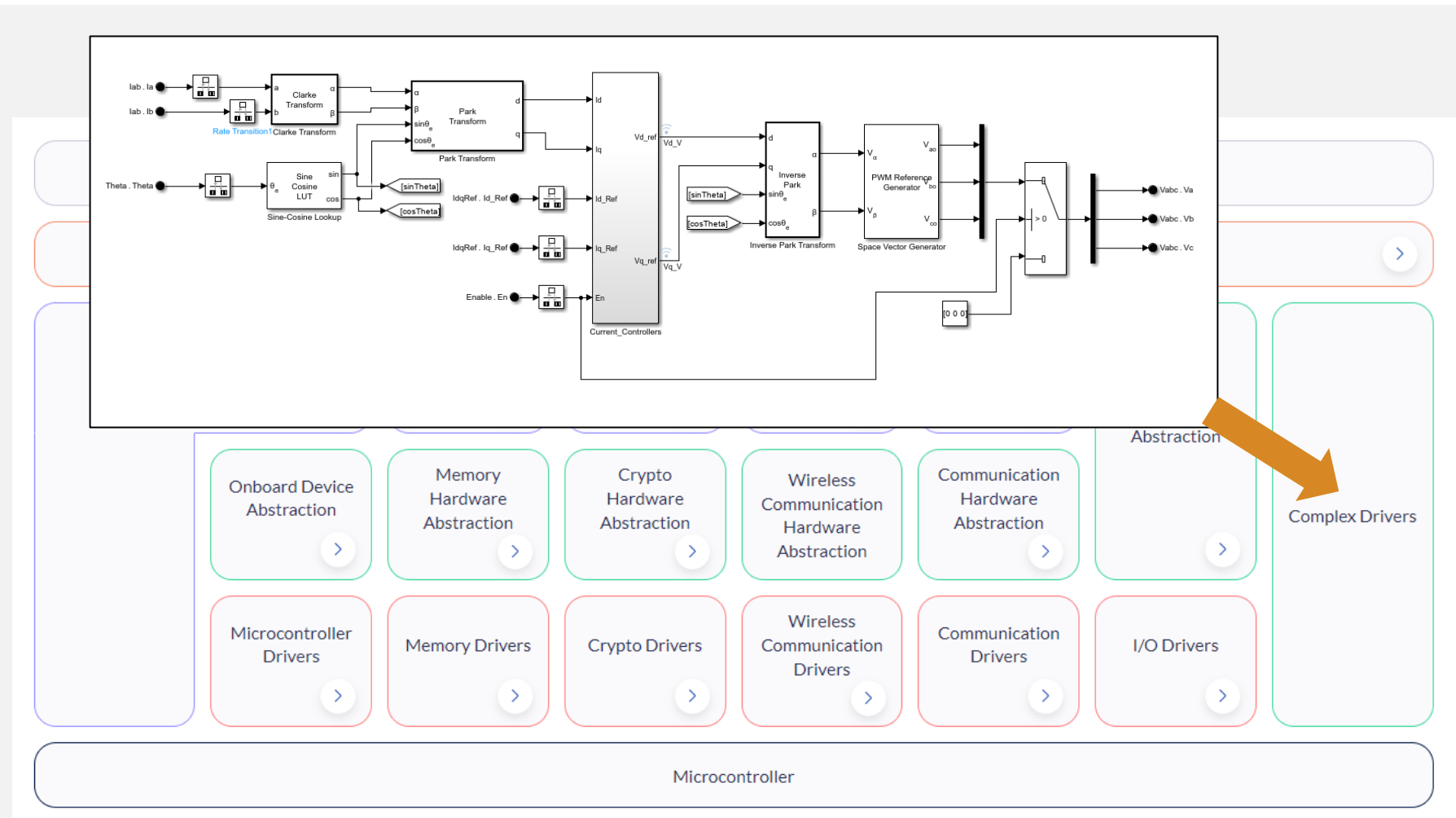
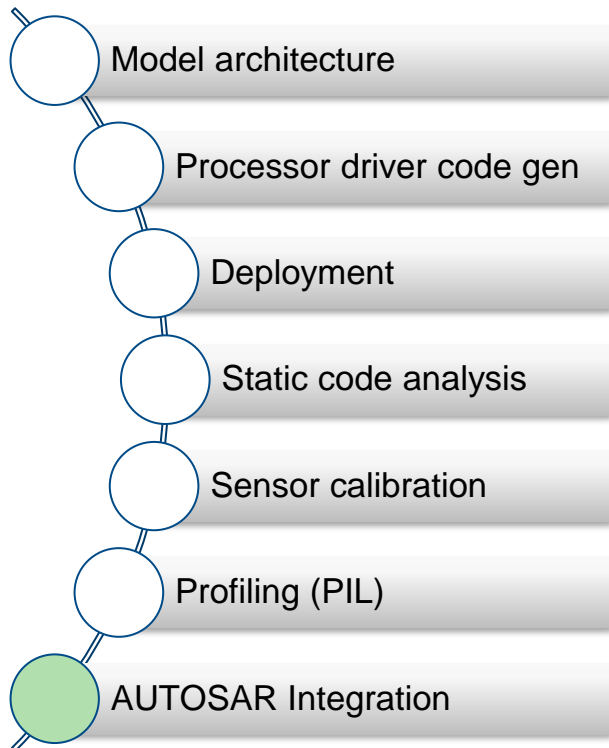
Total time	54531910
Unit of time	ns
Command	report(executionProfile, 'Units', 'seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%0.0f');
Timer frequency (ticks per second)	2e+08
Profiling data created	15-Jan-2021 13:00:17

### 2. Profiled Sections of Code

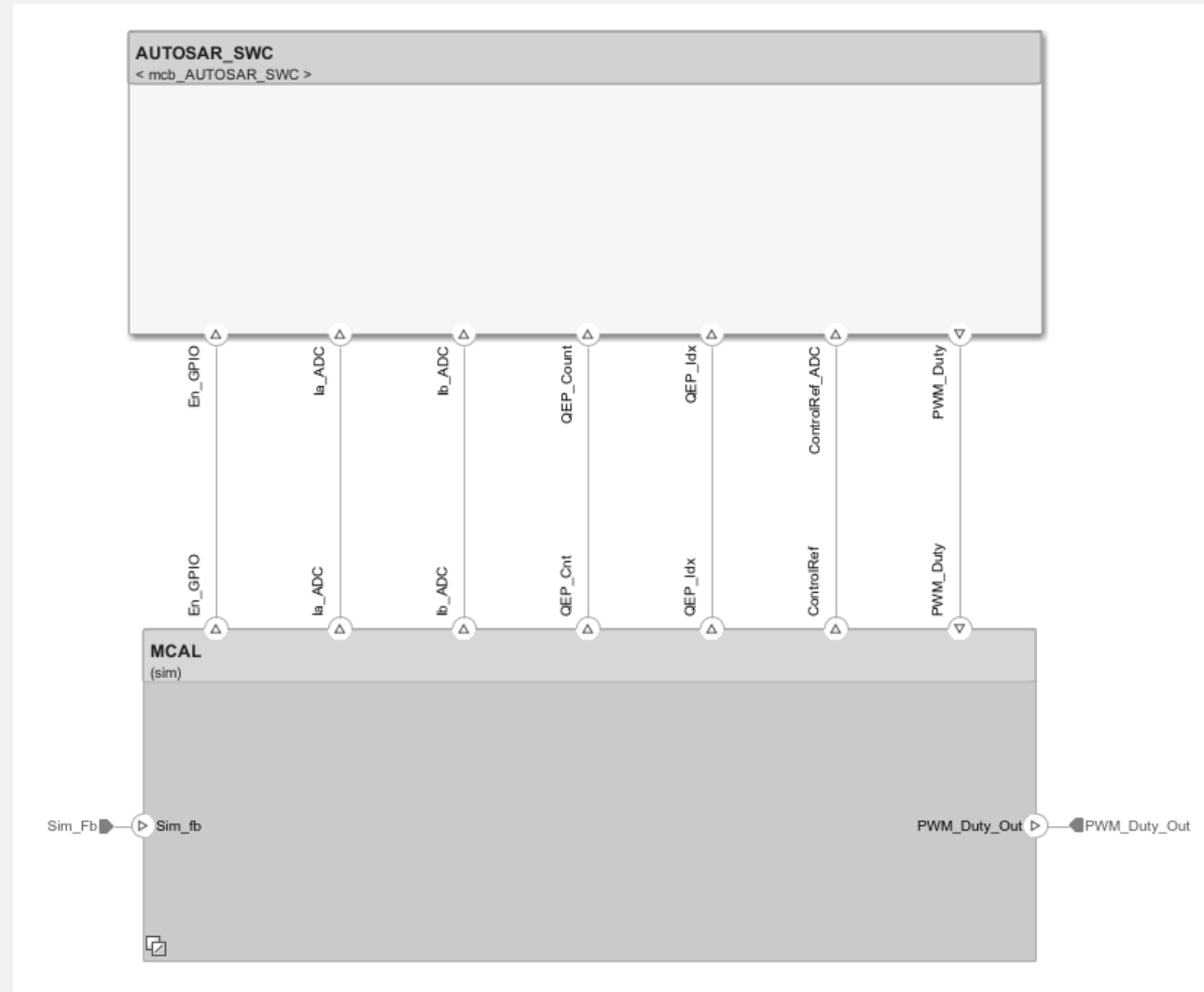
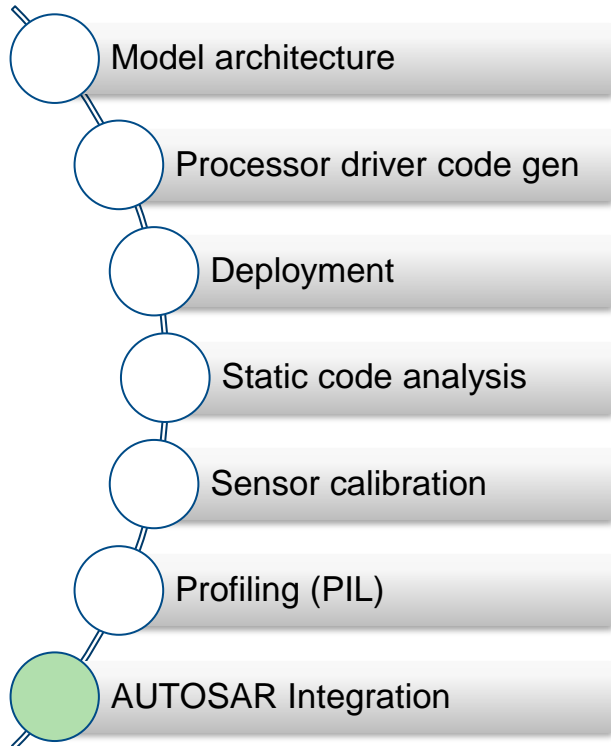
Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls
[+] <a href="#">Current_initialize</a>	1935	1935	1010	1010	1
[+] <a href="#">Current_step [5e-05 0]</a>	5560	5452	580	580	10001
<a href="#">Current_terminate</a>	140	140	140	140	1



# Integrate motor control loop in AUTOSAR architecture

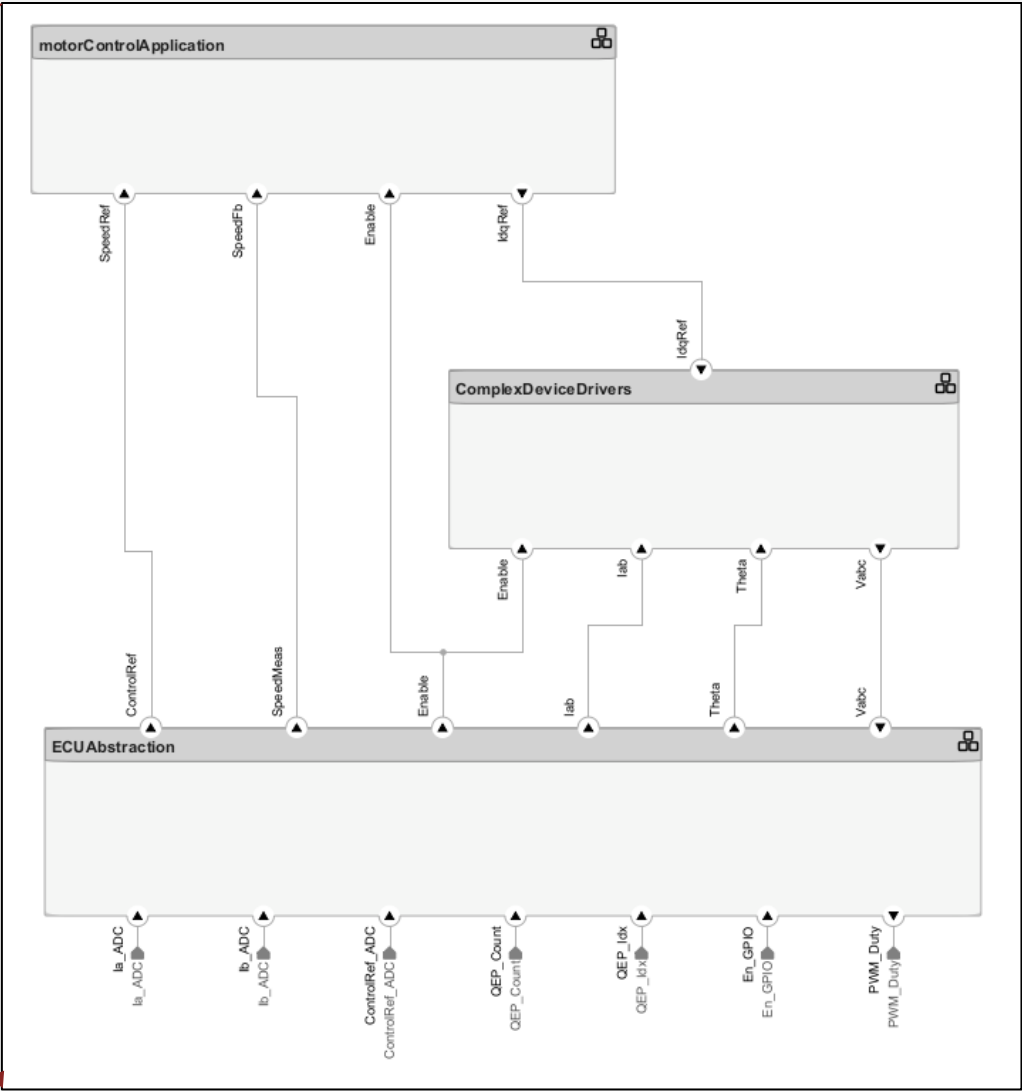
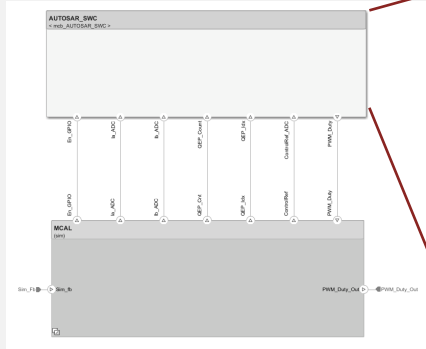


# Integrate motor control loop in AUTOSAR architecture

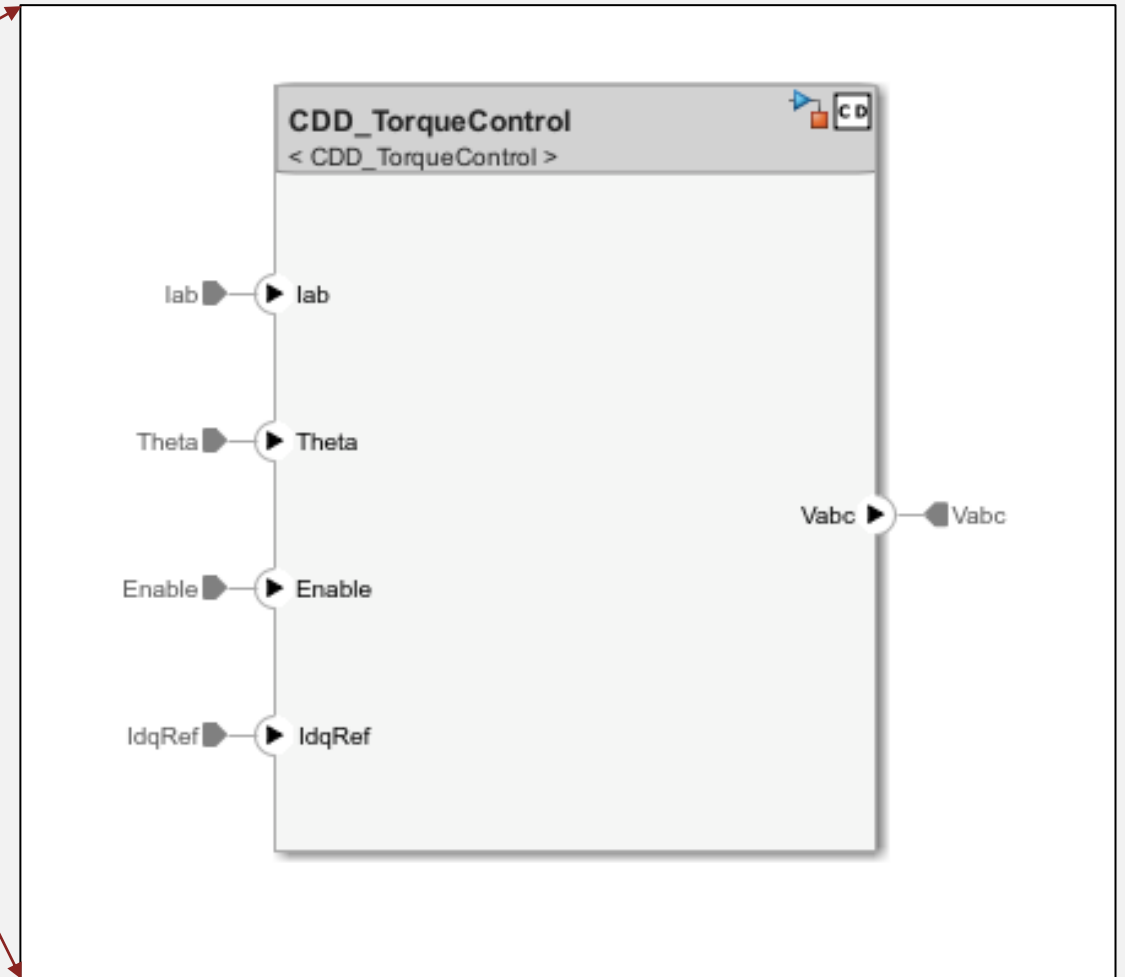
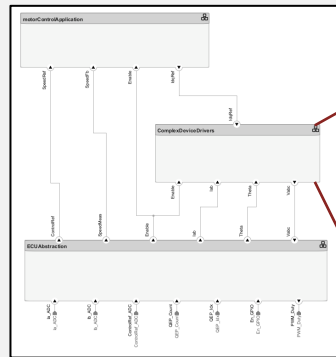
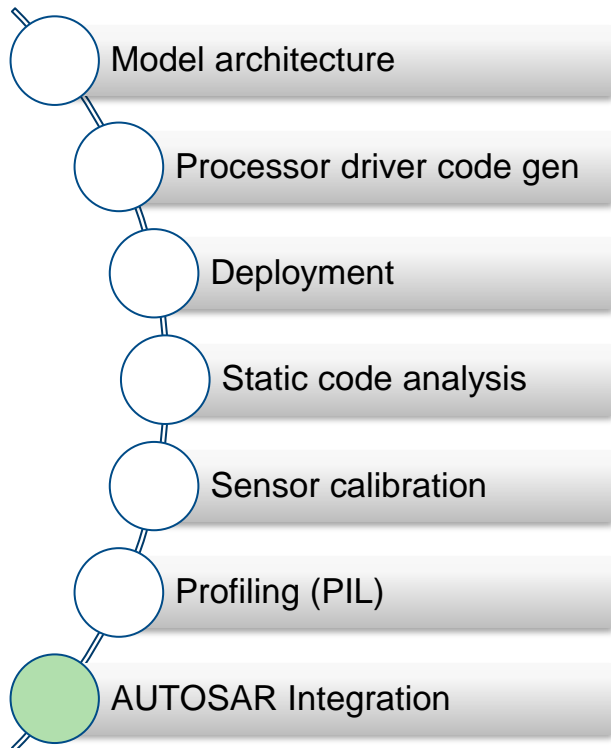


# Integrate motor control loop in AUTOSAR architecture

- Model architecture
- Processor driver code gen
- Deployment
- Static code analysis
- Sensor calibration
- Profiling (PIL)
- AUTOSAR Integration**



# Integrate motor control loop in AUTOSAR architecture



# Integrate motor control loop in AUTOSAR architecture

Model architecture

Processor driver code gen

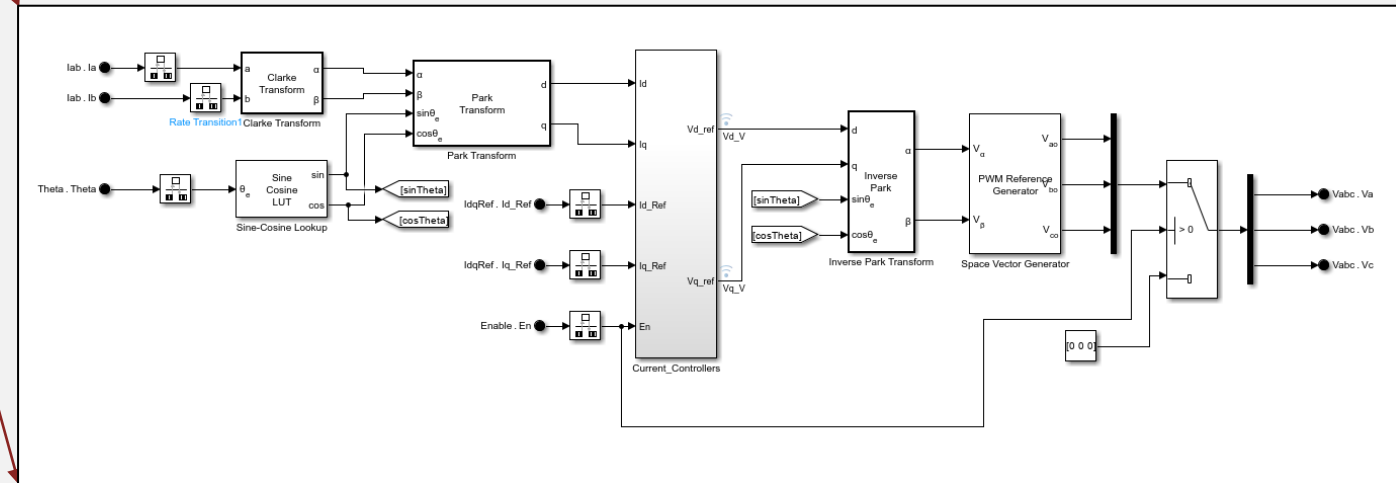
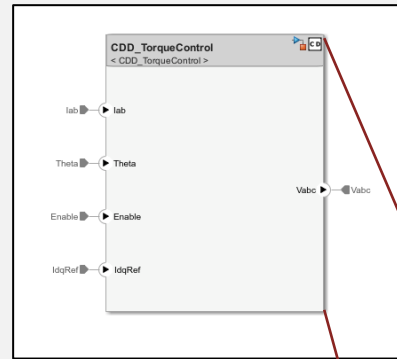
Deployment

Static code analysis

Sensor calibration

Profiling (PIL)

**AUTOSAR Integration**



# Generate AUTOSAR compliant code and ARXML file for motor control component

Model architecture

Processor driver code gen

Deployment

Static code analysis

Sensor calibration

Profiling (PIL)

AUTOSAR Integration

The screenshot displays the MATLAB/Simulink workspace for a motor control component. On the left, a file browser shows the project structure, including folders for stubs, ARXML files, Windows Batch files, C source files, DMR files, and MATLAB files. The main workspace is divided into three panes:

- ARXML Component Description:** Shows the XML metadata for the component 'CDD\_TorqueControl'. Key details include:
  - Version: 1.17
  - Generated on: Wed Sep 27 11:39:19 2023
  - UUID: 1941239435 3344221716 3681281373 1919767531
  - XML Schema: `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`
  - Component Type: `<COMPLEX-DEVICE-DRIVER-SW-COMPONENT-TYPE UUID="2312b9f1-dde3-5f-91-000000000000">`
  - Ports: `<R-PORT-PROTOTYPE UUID="bb3386e7-96a4-5fbc-0719-705f4c-000000000000">` (Theta)
  - Required-Com-Specs: `<NONQUEUED-RECEIVER-COM-SPEC DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE" HANDLE-OUT-OF-RANGE NONE USES-END-TO-END-PROTECTION false ALIVE-TIMEOUT 0/>`
- Generated C Code:** Shows the header file `Rte_CDD_TorqueControl.h` with the following content:
 

```

ARXML schema: "R20-11"
File generated on: "27-Sep-2023 11:39:16" */

#ifndef Rte_CDD_TorqueControl_h
#define Rte_CDD_TorqueControl_h
#include "Rte_Type.h"
#include "Compiler.h"

/* Data access functions */
#define Rte_IRead_CDD_TorqueControl_Step_Theta_Theta Rte_IRead_CDD_TorqueControl_Step_Theta_Theta
Float Rte_IRead_CDD_TorqueControl_Step_Theta_Theta(void);

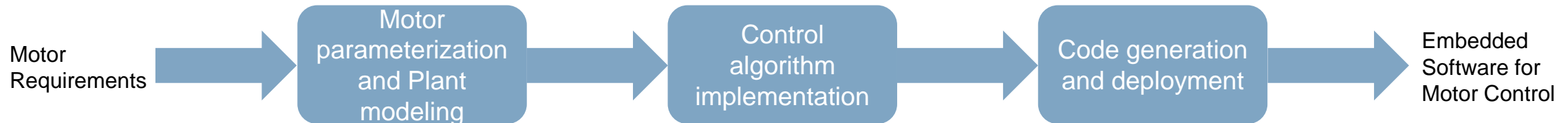
#define Rte_IRead_CDD_TorqueControl_Step_Enable_En Rte_IRead_CDD_TorqueControl_Step_Enable_En
Boolean Rte_IRead_CDD_TorqueControl_Step_Enable_En(void);

#define Rte_IRead_CDD_TorqueControl_Step_Iab_Ia Rte_IRead_CDD_TorqueControl_Step_Iab_Ia
Float Rte_IRead_CDD_TorqueControl_Step_Iab_Ia(void);

#define Rte_IRead_CDD_TorqueControl_Step_IdqRef_Id_Ref Rte_IRead_CDD_TorqueControl_Step_IdqRef_Id_Ref
Float Rte_IRead_CDD_TorqueControl_Step_IdqRef_Id_Ref(void);
      
```



# Summary



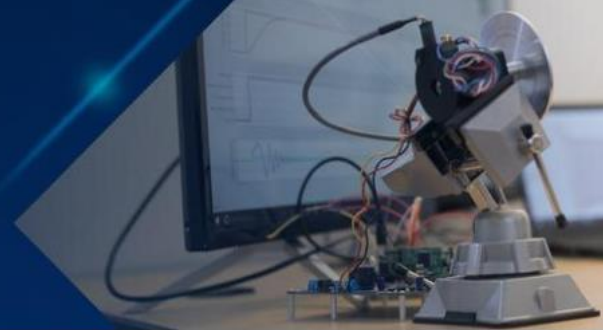
- ✓ Discussed on the motor control workflow and reference examples available for each of the steps
- ✓ Different methods to parameterize the motor and different fidelity levels in modeling
- ✓ Different control strategies and its control loop gain tuning
- ✓ Deploy to the hardware and Validate

# Learn More

## MATLAB and Simulink for Motor Drives and Traction Motors

Develop algorithms and embedded software for motor-inverter control systems

Free trial



### How It Works



Design motor control algorithms



Test motor control algorithms



Implement motor control algorithms on hardware

## Simulate Motor Control Algorithms

Use MATLAB® and Simulink® to build motor models from libraries of motors, inverters, sources, and loads. Choose the level of fidelity in motor and inverter modeling based on your requirements and simulate motor control algorithms.

- Implement linear lumped-parameter motor models and use average value inverters with Motor Control Blockset™ for fast simulations
- Model and simulate nonlinear motor dynamics and ideal or detailed switching in the inverter using Power Systems Simulation Onramp
- Parametrize motor models to capture motor dynamics with the help of instrumented tests or import parameters from a database or finite element analysis



Motor Control Design with MATLAB and Simulink

## MathWorks Videos

Suggest a video



Understanding Field-Oriented Control | Motor Control, Part 4



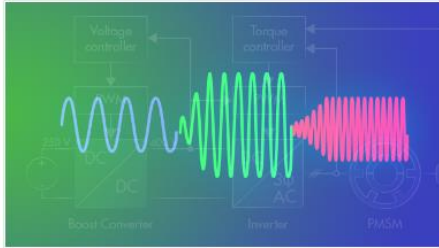
Reinforcement Learning for Developing Field-Oriented Control



Motor Control, Part 3: BLDC Speed Control Using PWM

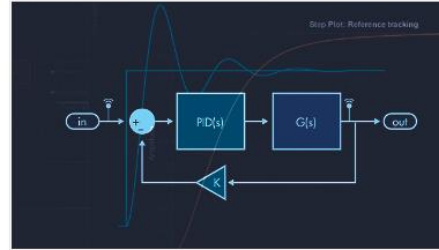
» View all MathWorks videos

# Enable Your Team on Motor Control



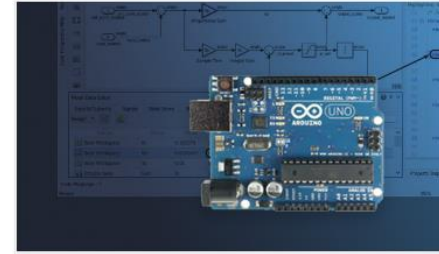
## Power Electronics Control Design with Simulink and Simscape

Learn to model power electronic systems in the Simulink environment using Simscape Electrical™ and to design control with Simulink Control Design.



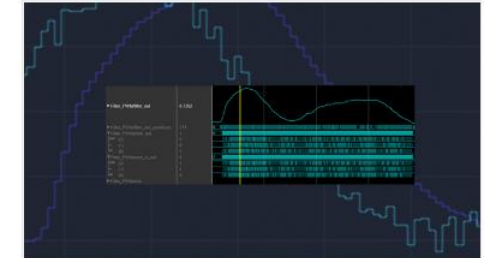
## Control System Design with MATLAB and Simulink

Learn to design and model control systems with Simulink. Topics include system identification, parameter estimation, control system analysis, and response optimization.



## Embedded Coder for Production Code Generation

Develop Simulink models for deployment in embedded systems. Topics include code structure and execution, code generation options and optimizations, and deploying code to target hardware.



## Generating HDL Code from Simulink

Learn to prepare Simulink models for HDL code generation, generate HDL code and testbench for a compatible Simulink model, and perform speed and area optimizations.



## Power Electronics Simulation Onramp

5 modules | 1 hour | [Languages](#)

Learn the basics of simulating power electronics converters in Simscape.



## Circuit Simulation Onramp

7 modules | 2 hours | [Languages](#)

Learn the basics of simulating electrical circuits in Simscape.



MathWorks  
**AUTOMOTIVE  
CONFERENCE 2023**  
India

**Thank you**



# FOC Autotuner perturbs the output and computes the control gain

Block Parameters: Field Oriented Control Autotuner

FOCAutoTuner (mask) (link)

This block automatically and iteratively tunes multiple control loops used in Field Oriented Control applications. The optional loops to tune are the quadrature axis (q-axis) current, direct axis (d-axis) current, speed and flux loops. Use "Help" button for more information regarding general tuning workflow.

Parameters

Tuned Loops

Tune D-axis current loop  Tune Q-axis current loop

Tune speed loop  Tune flux loop

Loop Settings

Use same settings for current loop controllers (D-axis + Q-axis)

Use same settings for outer loop controllers (Speed + Flux)

Experiment Sample Time

Experiment sample time (sec) -1

Tuning Experiment Block

Loop Tuning Settings

Tuning sample time (sec)  $Ts\_tuning$  0.005  Use different sample time for tuning

Current Loops (D-axis + Q-axis)

Controller

Type: PI

Form: Parallel

Discrete Time Settings

Controller sample time (sec)  $Ts$   $5e-05$

Integrator method Forward Euler

Filter method Forward Euler

Tuning Goals

Target bandwidth (rad/sec)  $PI\_params.CurrentBW$  2000

Target phase margin (degrees)  $PI\_params.CurrentPhaseMargin$  80

Speed Loop

Controller

Type: PI

Form: Parallel

Discrete Time Settings

Controller sample time (sec)  $Ts\_speed$  0.0005

Integrator method Forward Euler

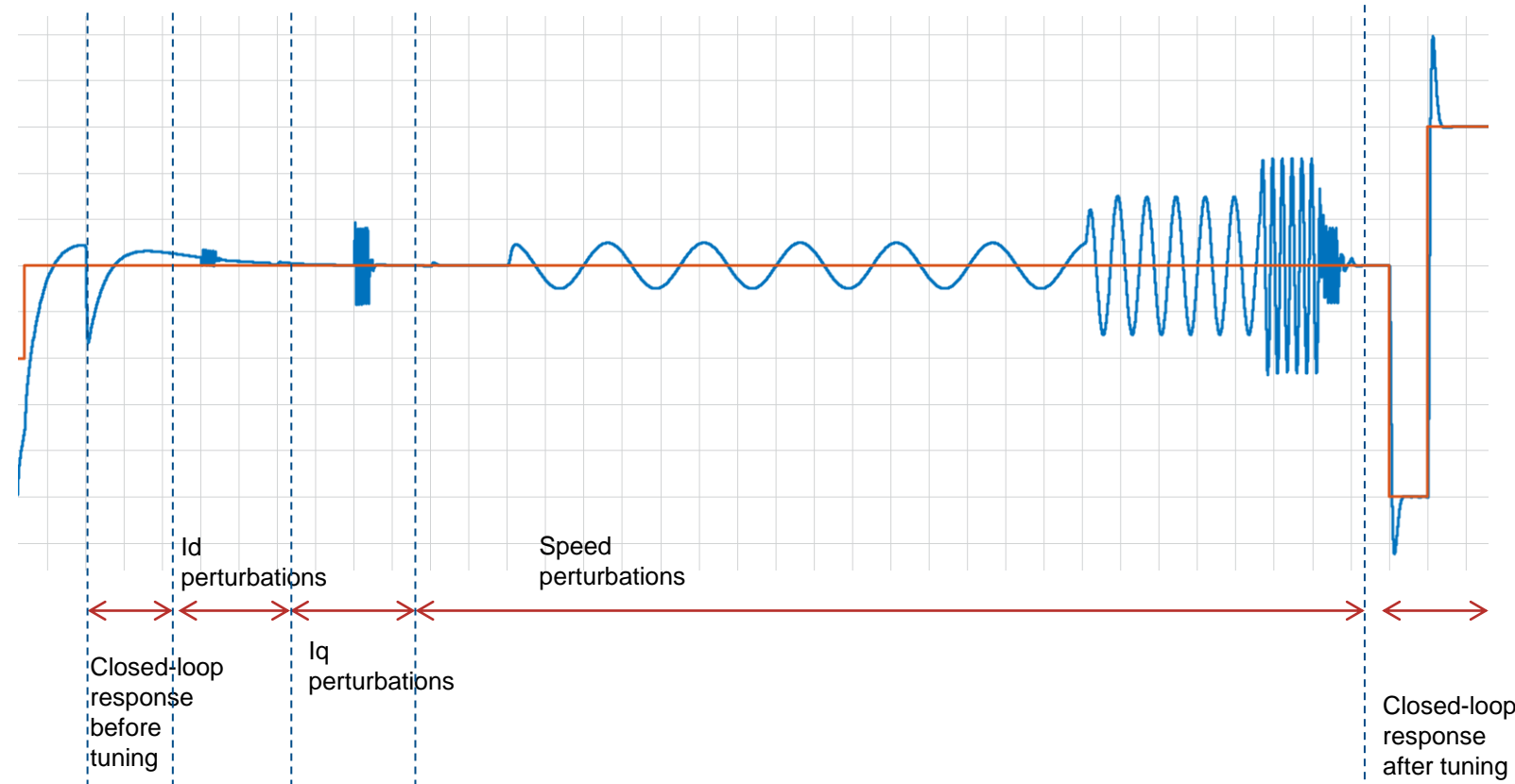
Filter method Forward Euler

Tuning Goals

Target bandwidth (rad/sec)  $PI\_params.SpeedBW$  50

Target phase margin (degrees)  $PI\_params.SpeedPhaseMargin$  30

OK Cancel Help Apply



# Online Frequency Estimation injects perturbation in the output of the controller

