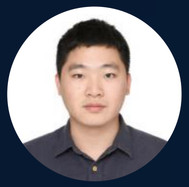
MATLAB EXPO 2021

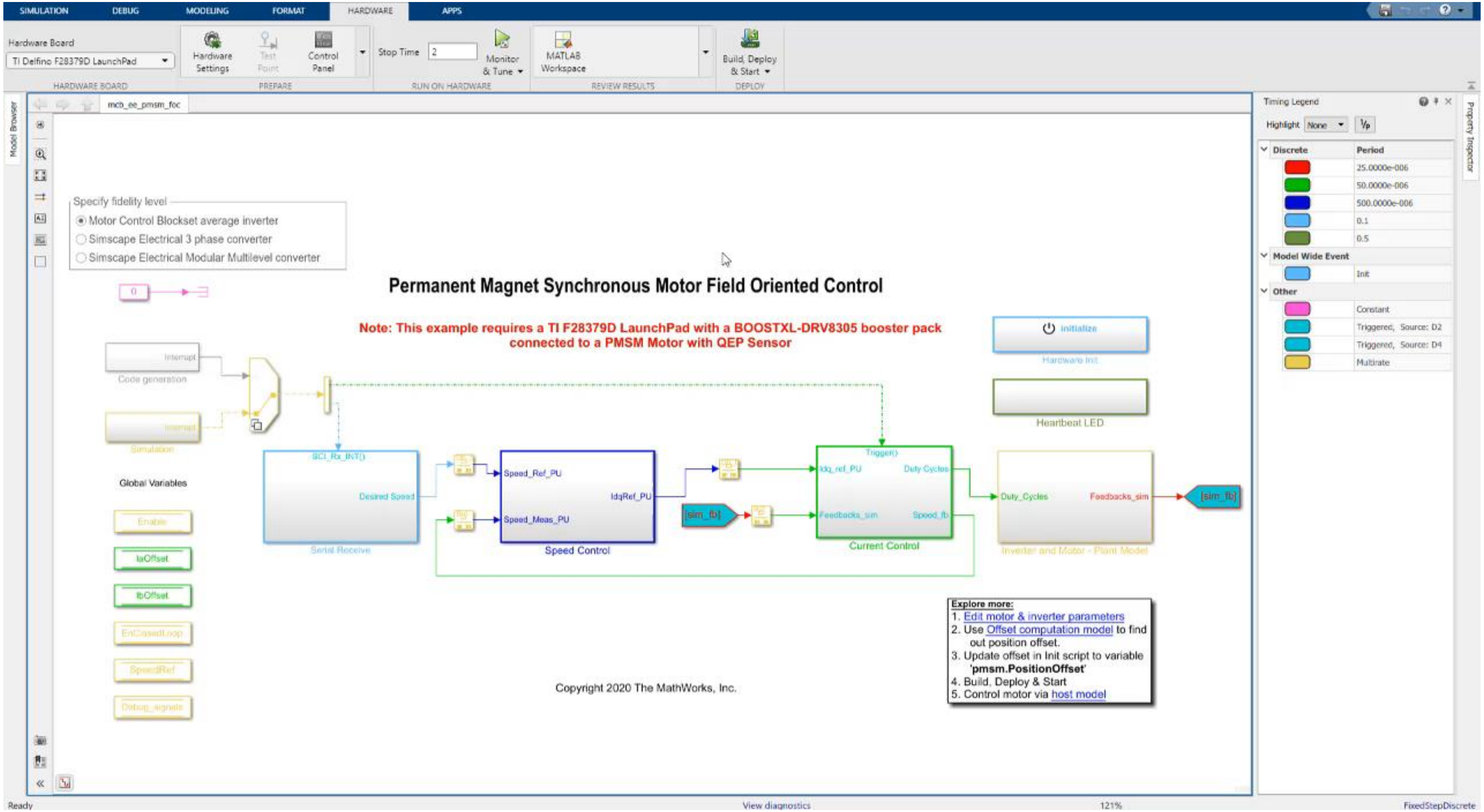
使用Simulink开发电机控制系统

樊朝祥, MathWorks

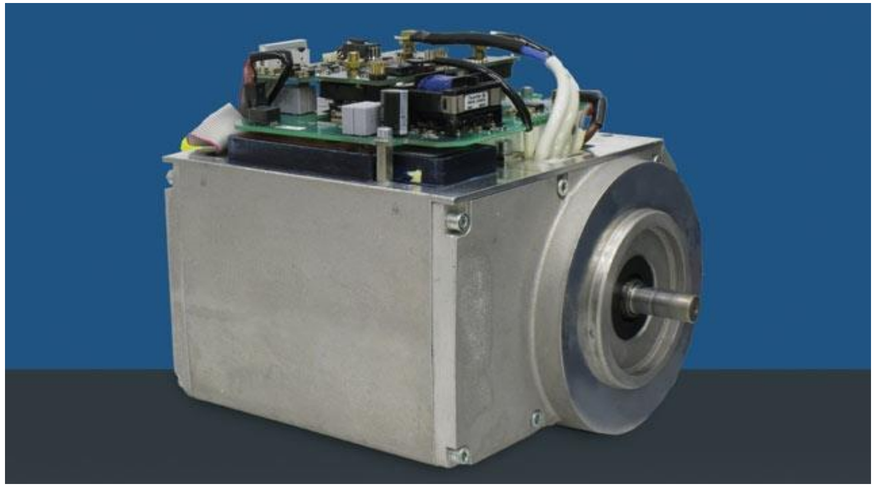


MathWorks

本讲实例 – Simulink 玩转无刷直流电机



无刷电机无处不在



电机控制开发面临的挑战 —— 用户案例

ITK Engineering 使用基于模型的设计开发符合 IEC 62304 标准的牙钻电机控制器

挑战

为牙钻无刷直流电机电机控制器开发无传感器的矢量控制FOC算法。

Solution

ITK 工程师使用基于模型的设计，对无传感器 BLDC 电机控制器以及被控对象电机本体进行建模、闭环仿真，优化、测试验证和自动生成代码。

Results

- 开发时间减半
- 在早期发现了硬件问题
- 赢得合同，树立了客户信心

配备 ITK Engineering 的无传感器、无刷电机控制功能的牙钻。

“使用 Simulink 的基于模型的设计，我们能够通过早期验证来降低成本和项目风险，缩短通过 IEC 62304 体系认证后的上市时间，交付一次性成功的高质量生产代码。”

- Michael Schwarz, ITK Engineering

Link to user story

电机控制开发面临的挑战

ITK Engineering 使用基于模型的设计开发符合 IEC 62304 标准的牙钻电机控制器

挑战

为牙钻无刷直流电机电机控制器开发无传感器的矢量控制FOC算法。

Solution

ITK 工程师使用基于模型的设计，对无传感器 BLDC 电机控制器以及被控对象电机本体进行建模、闭环仿真，优化、测试验证和自动生成代码。

Results

- 开发时间减半
- 在早期发现了硬件问题
- 赢得合同，树立了客户信心

配备 ITK Engineering 的无传感器、无刷电机控制功能的牙钻。

“使用 Simulink 的基于模型的设计，我们能够通过早期验证来降低成本和项目风险，缩短通过 IEC 62304 体系认证后的上市时间，交付一次性成功的高质量生产代码。”

- Michael Schwarz, ITK Engineering

设计工作通常在具备电机硬件之前开始

为了符合行业标准需要进行大量的软件测试

一旦硬件条件具备需要尽快将软件部署到嵌入式控制器进行测试

高实时性复杂算法很难在短时间内完成部署

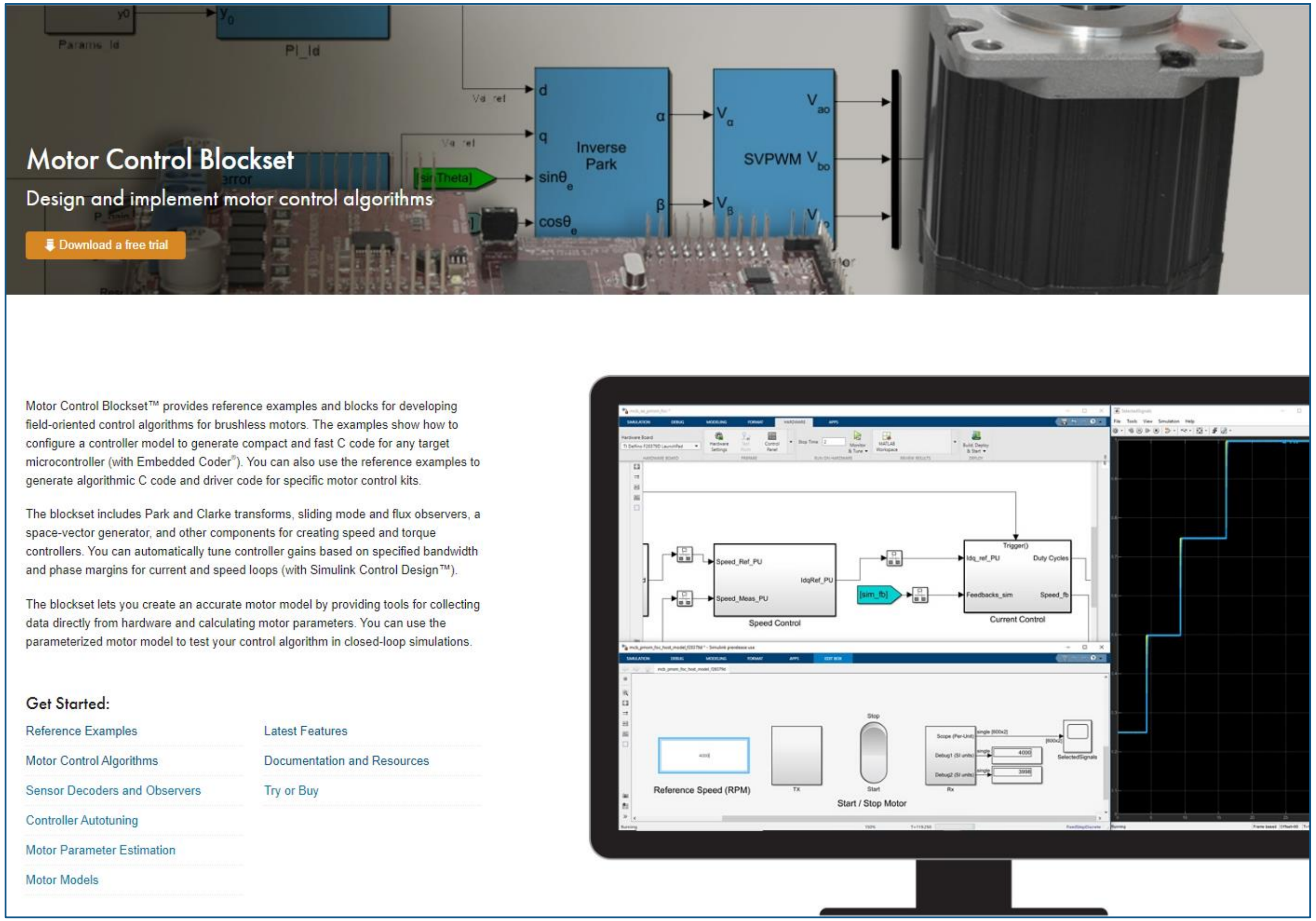
为什么使用Simulink开发电机控制

- 桌面仿真验证控制算法
- 从模型生成精简而高效的代码
- 利用参考设计最小化开发时间

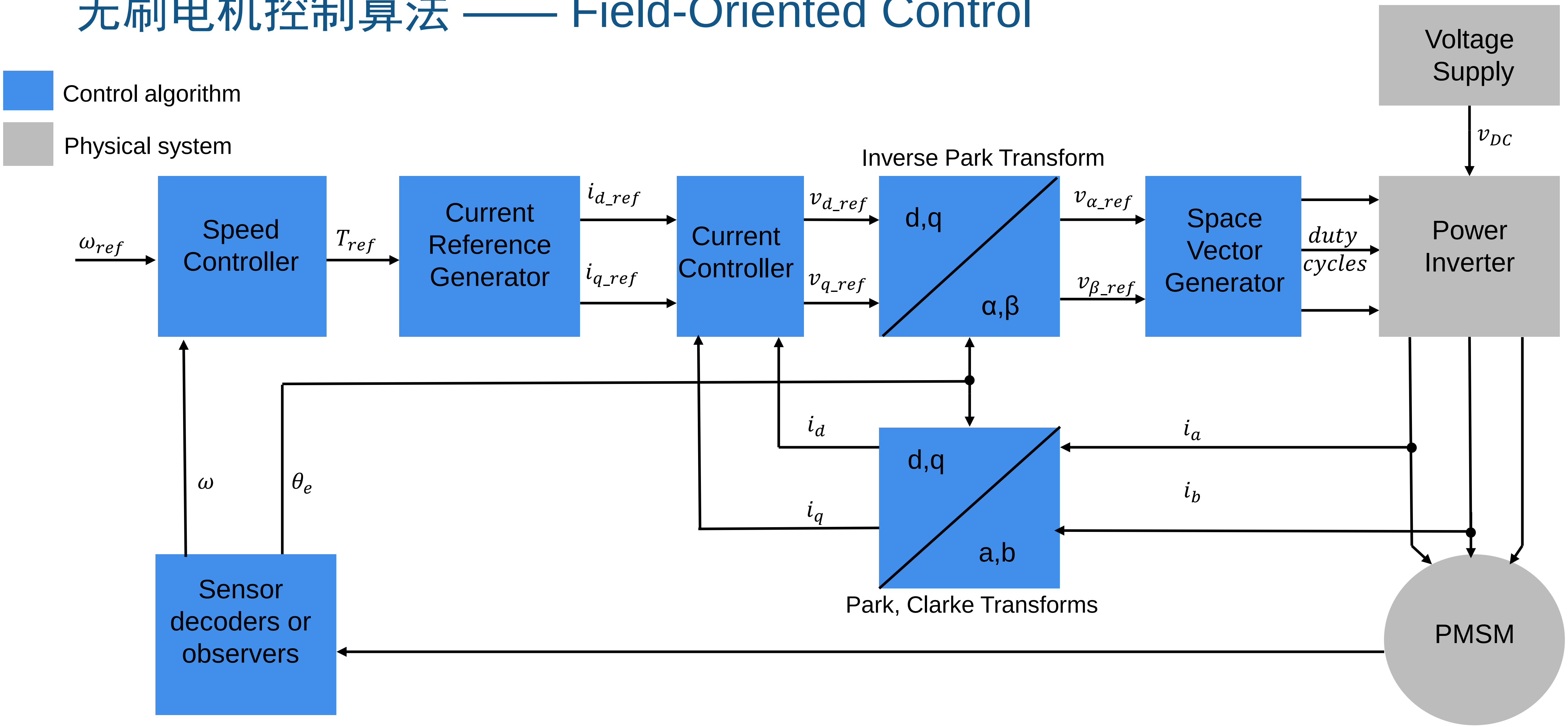
客户报告：将开发时间缩短 50% 以上

Motor Control Blockset 简化开发流程

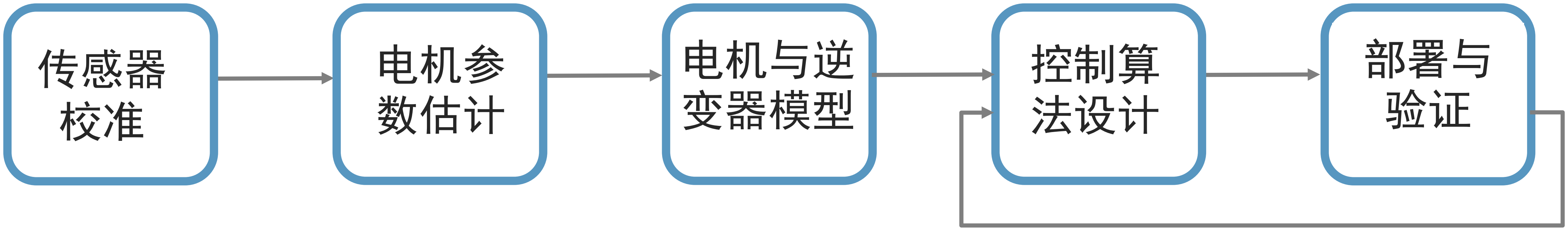
- 针对代码生成优化的算法模块
- 现成的编码器和磁链观察其等算法
- 电机参数估计
- 控制器自动调参
- 参考示例



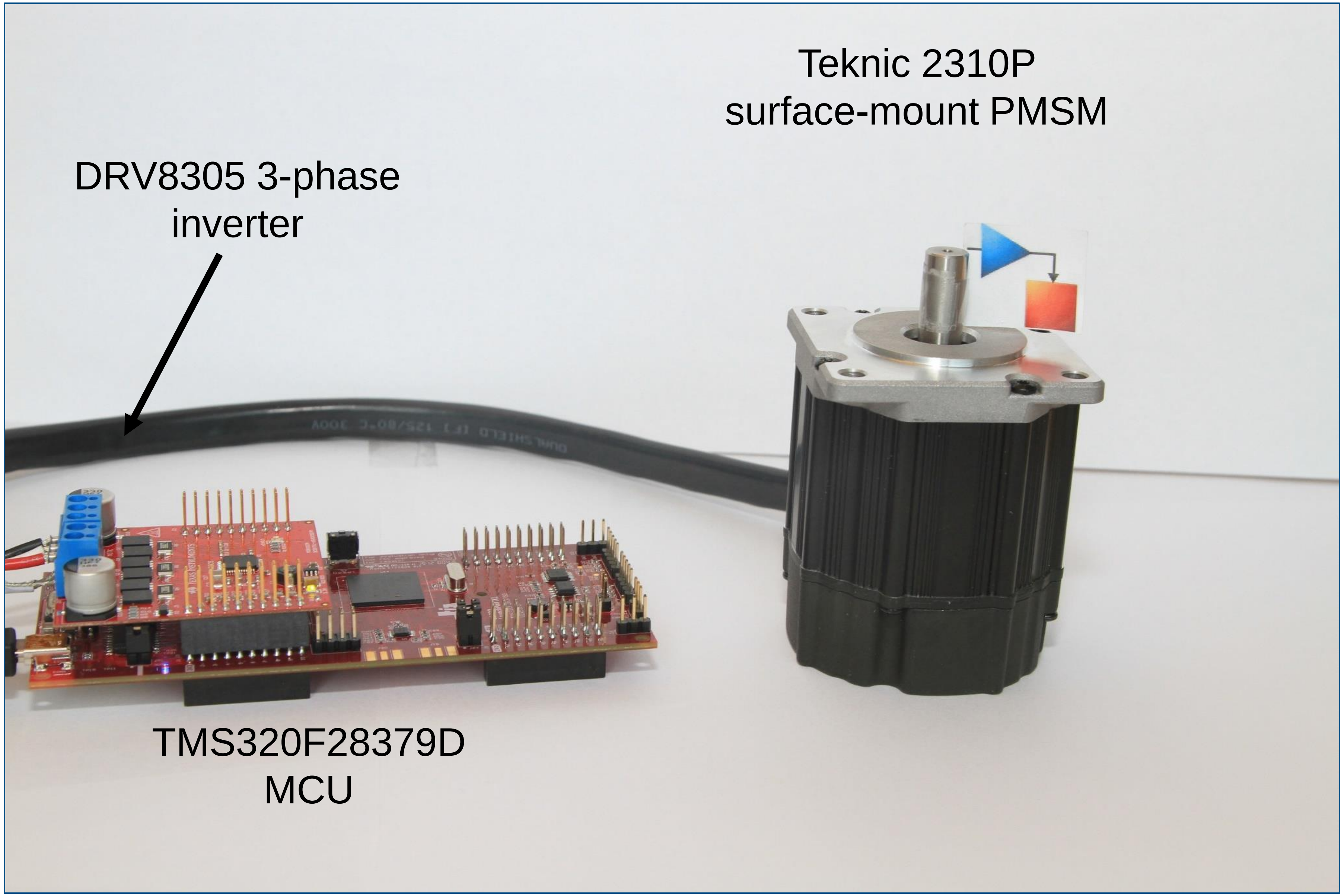
无刷电机控制算法 —— Field-Oriented Control



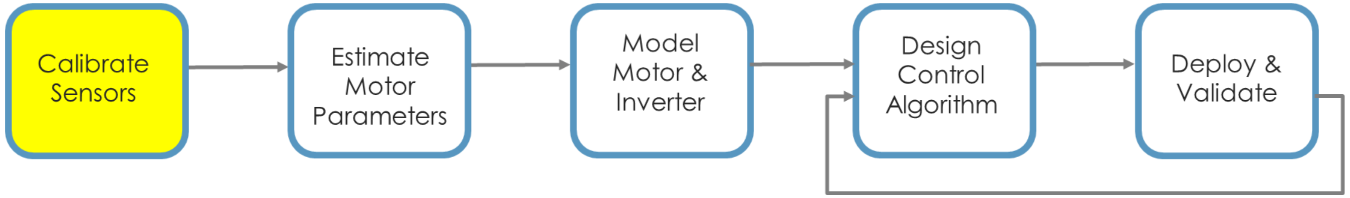
FOC 开发工作流



本例中使用的TI套件

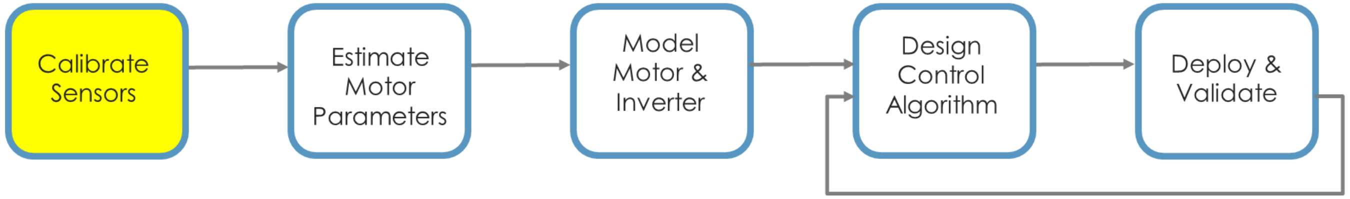

Teknic 2310P
surface-mount PMSM
DRV8305 3-phase
inverter
TMS320F28379D
MCU

传感器校准



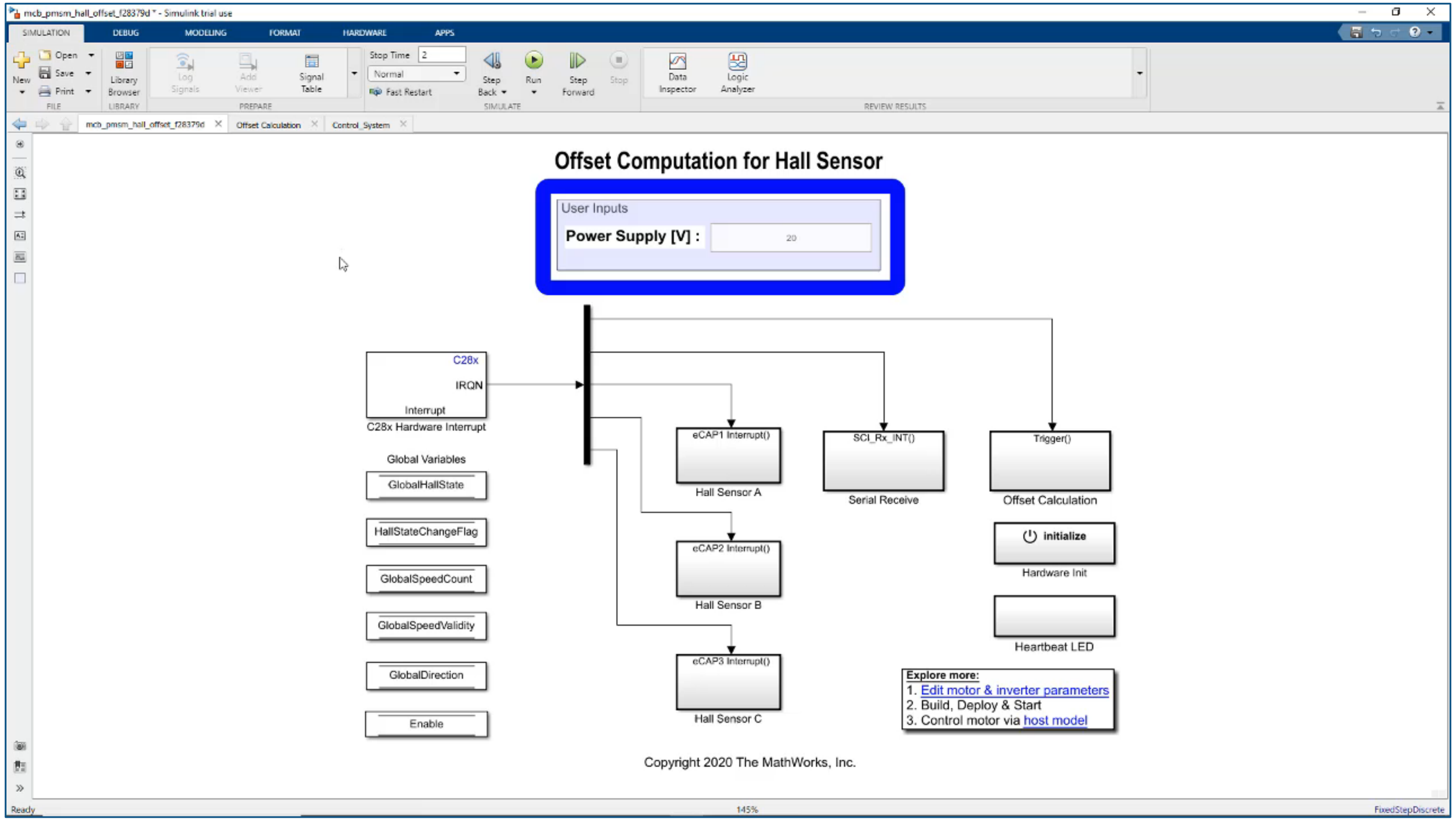
- ADC 校准
- 位置传感器校准

传感器校准

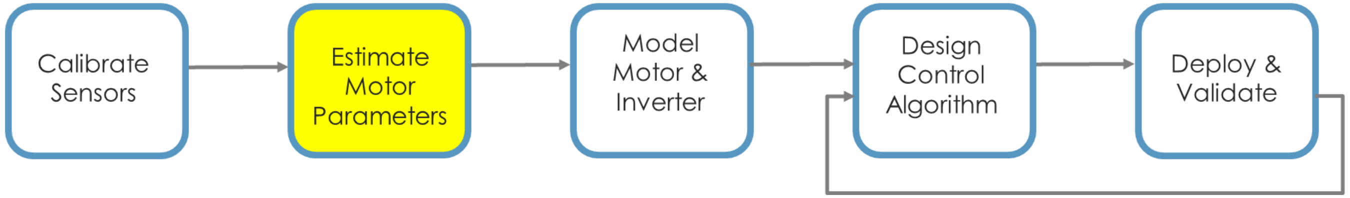


- ADC校准

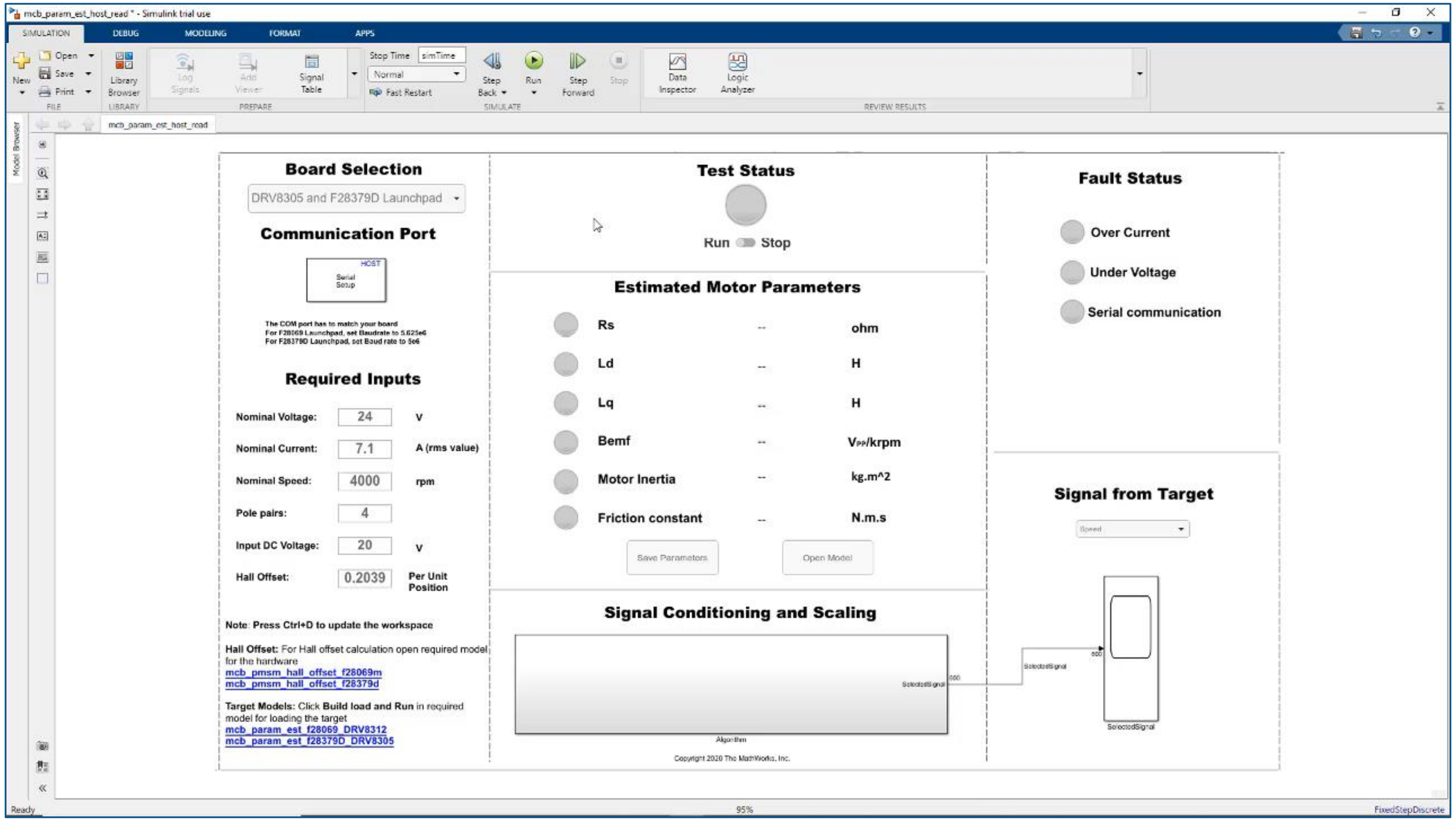
- 位置传感器校准



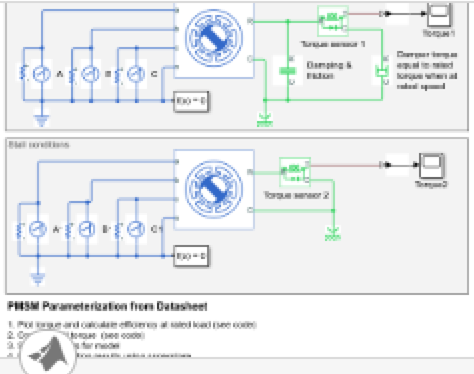
参数估计



- 在目标硬件上运行
- 从 Host 模型控制与估计电机参数



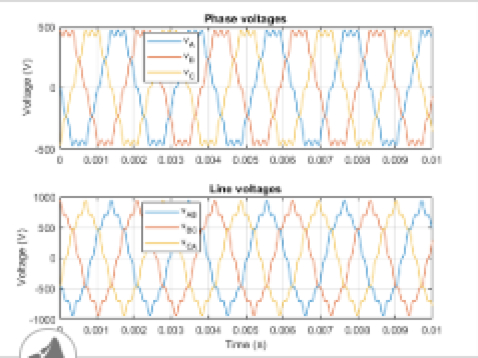
其他电机参数化方法



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

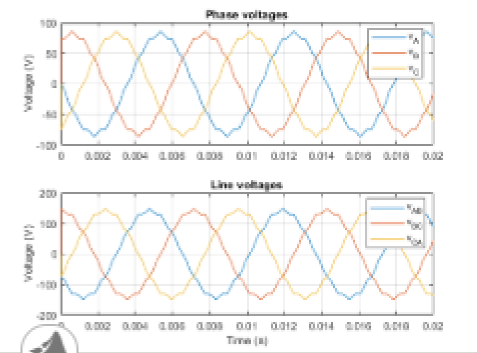
Open Model



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

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
Step 1: Load and Preprocess Data	Load and preprocess this nonlinear motor flux data from dynamometer testing or finite element analysis (FEA): • *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 `gridfit` 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.

数据手册

Simscape Electrical

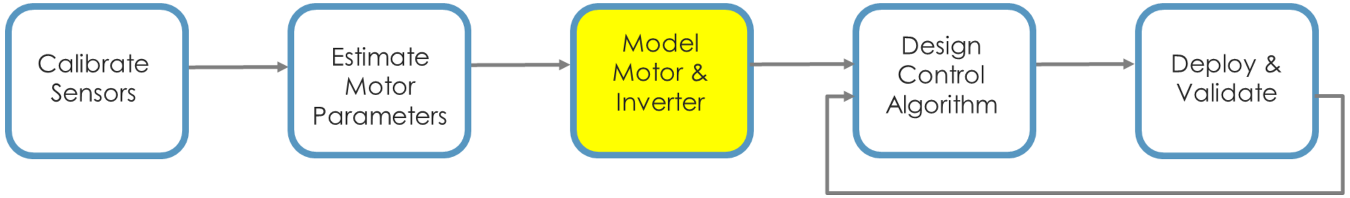
ANSYS Maxwell, JMAG,
Motor-CAD 等有限元工具

Simscape Electrical

功率计数据

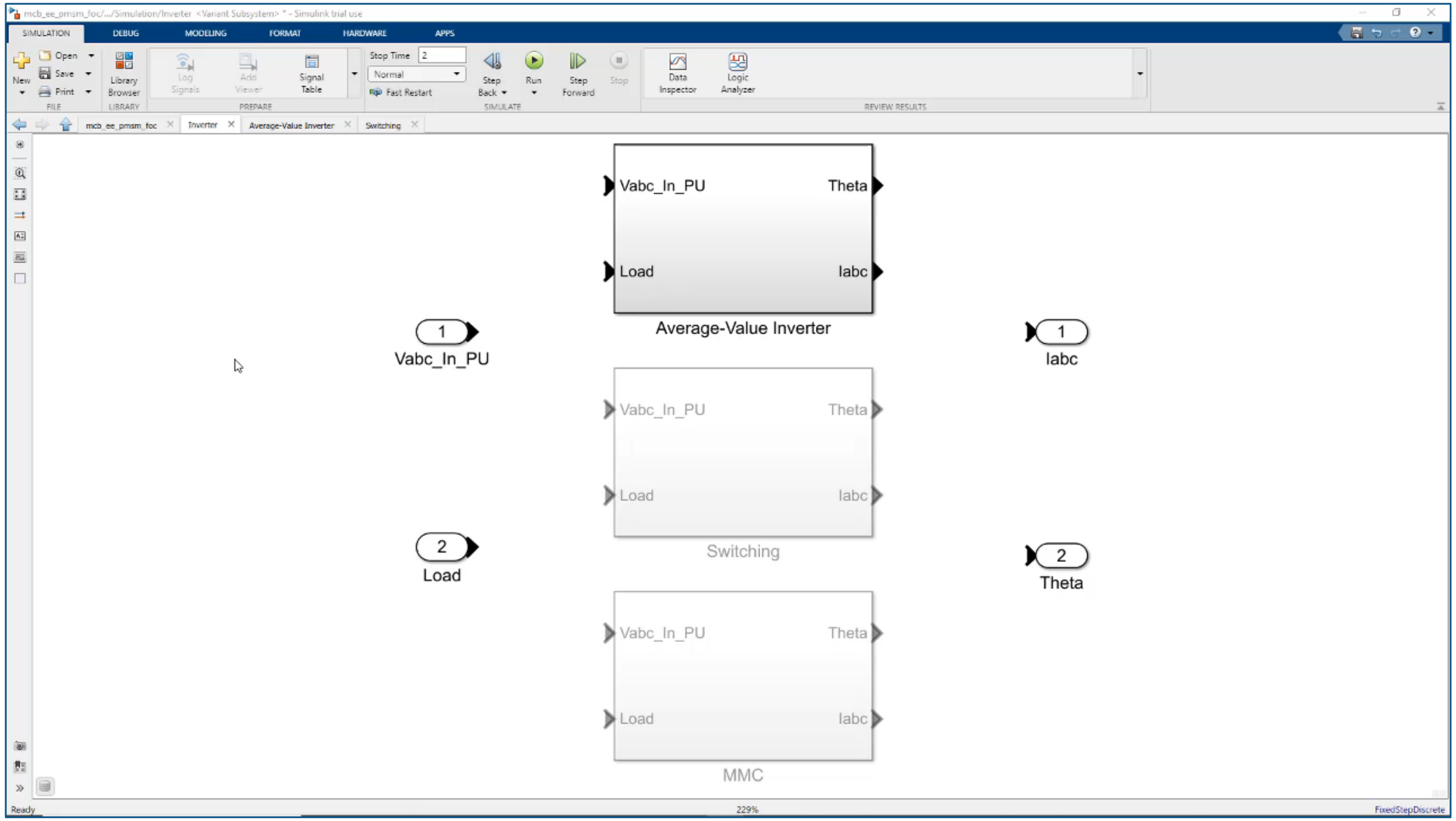
Powertrain Blockset

电机与逆变器建模

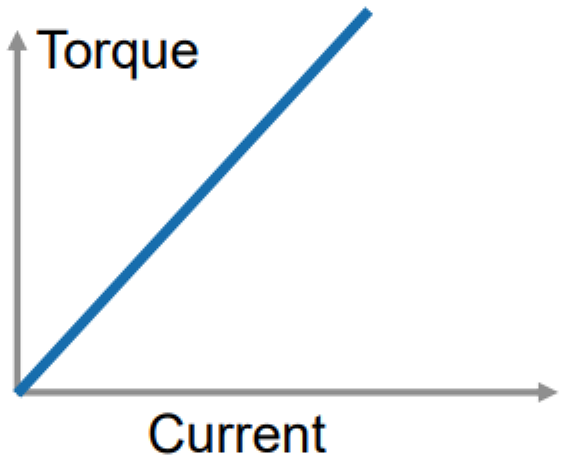


- 使用线性集总参数模型

- 使用平均值逆变器模型或 Simscape Electrical 中的开关模型

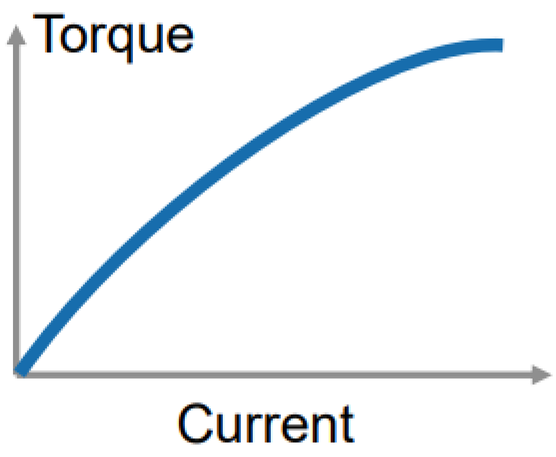


根据需要建立不同精度的模型



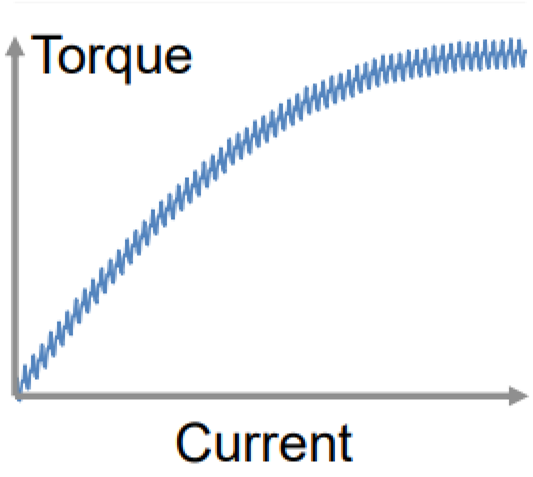
Lumped Parameter

Motor Control Blockset
Simscape Electrical



Saturation

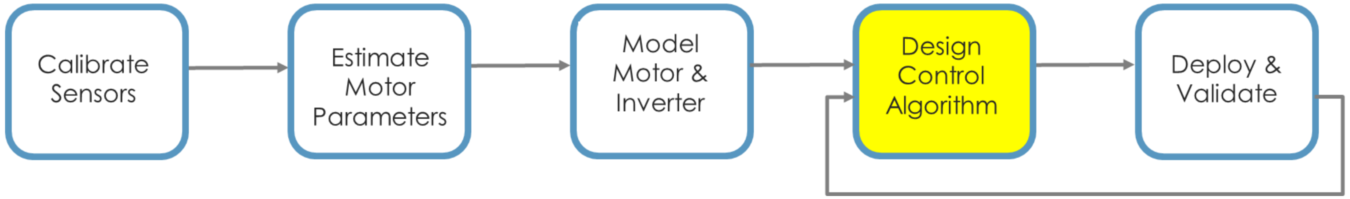
Simscape Electrical



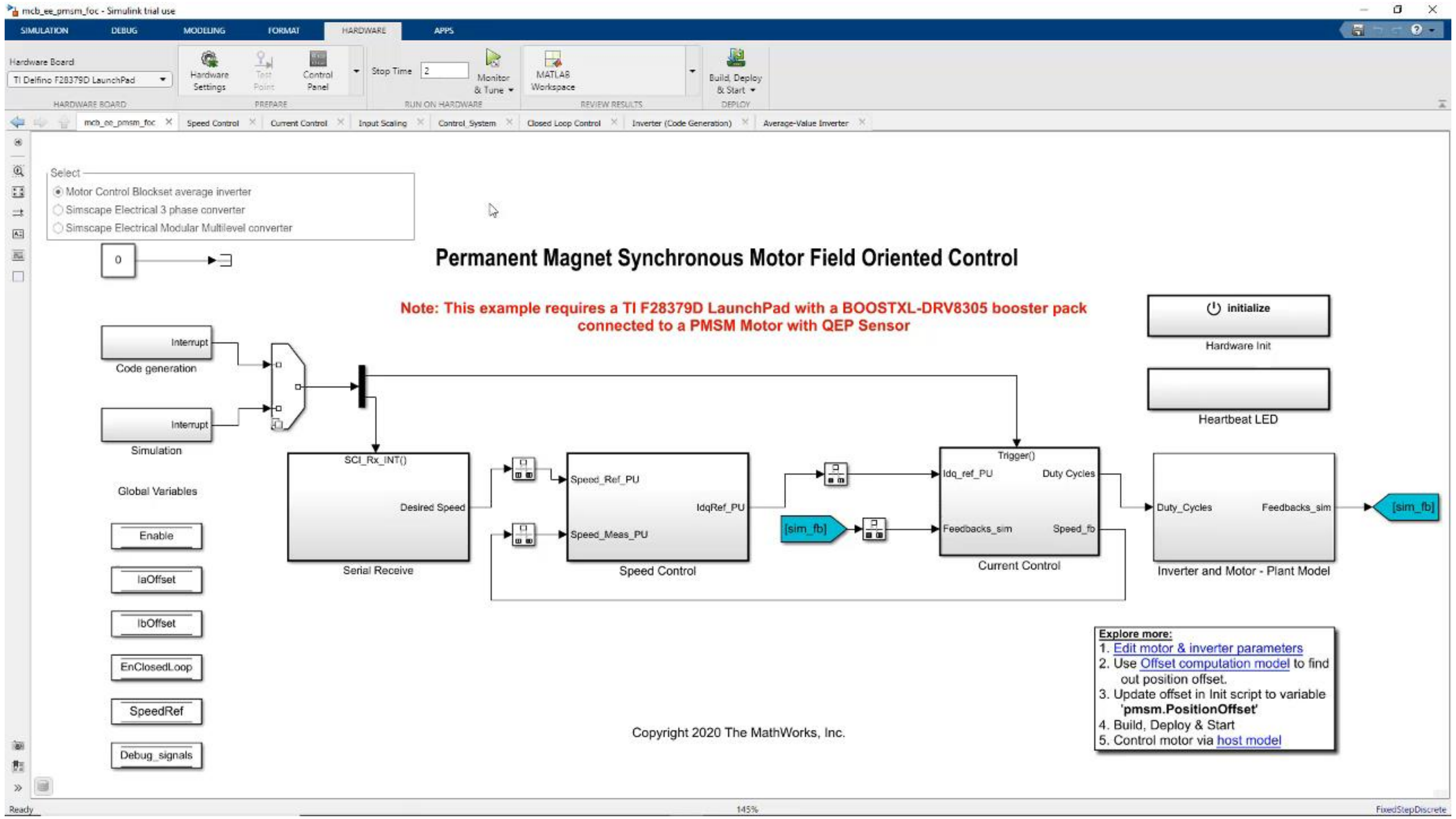
Saturation +
Spatial Harmonics

Simscape Electrical

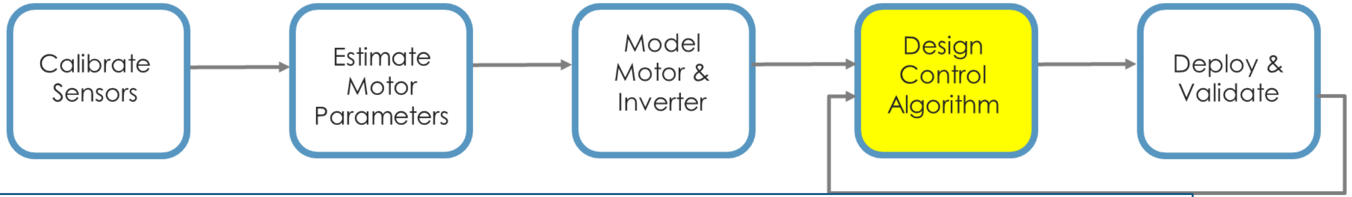
控制算法设计



- FOC算法建模
- 解码器或无传感器观测器建模
- 调节闭环参数
- 闭环仿真验证



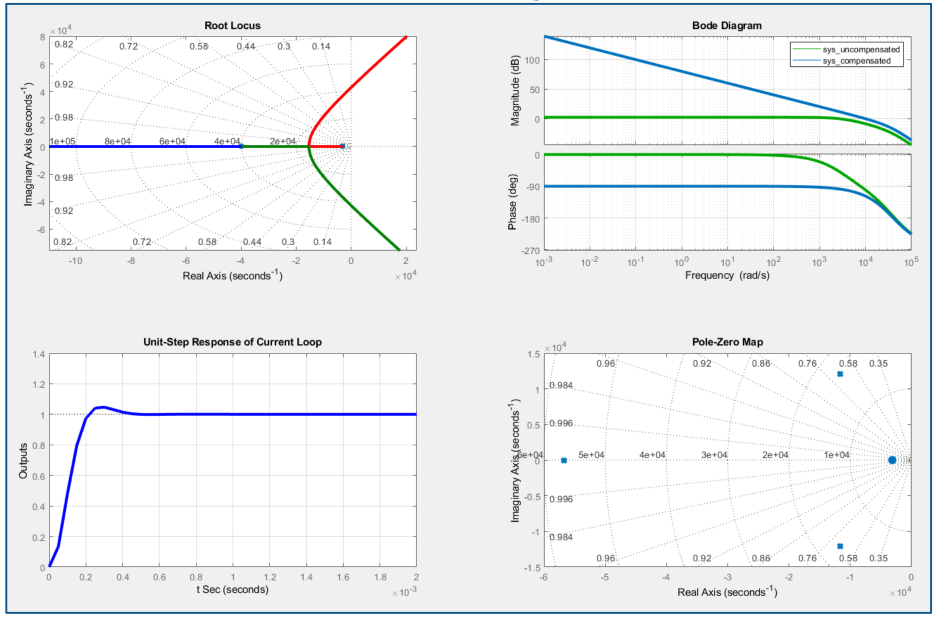
控制算法设计



- FOC算法建模
- 解码器或无传感器观测器建模
- **调节闭环参数**
- 闭环仿真验证

```
%% 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



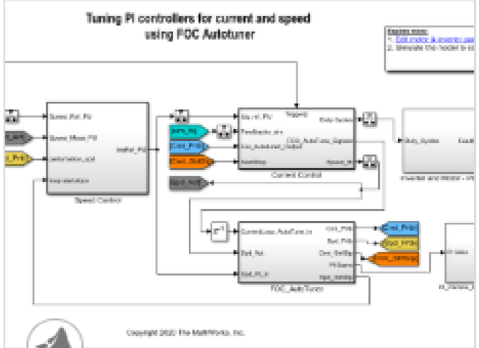
Control System Toolbox

多种控制调参技术

```
%% 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);
```

经验计算

Motor Control Blockset



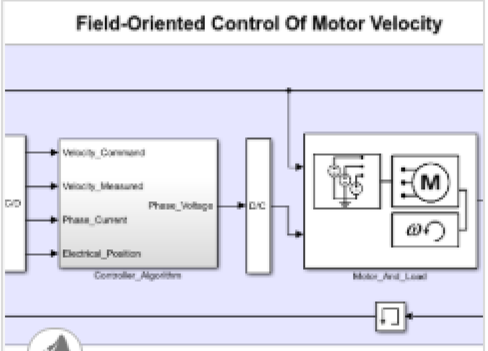
Tune PI controllers by Using Field Oriented Control (FOC) Autotuner

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



Tune Field-Oriented Controllers Using SYSTUNE

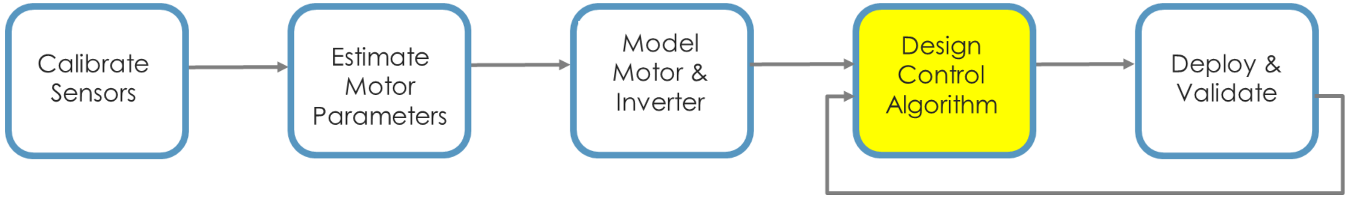
Tune a field-oriented controller for an asynchronous machine in one simulation.

Open Script

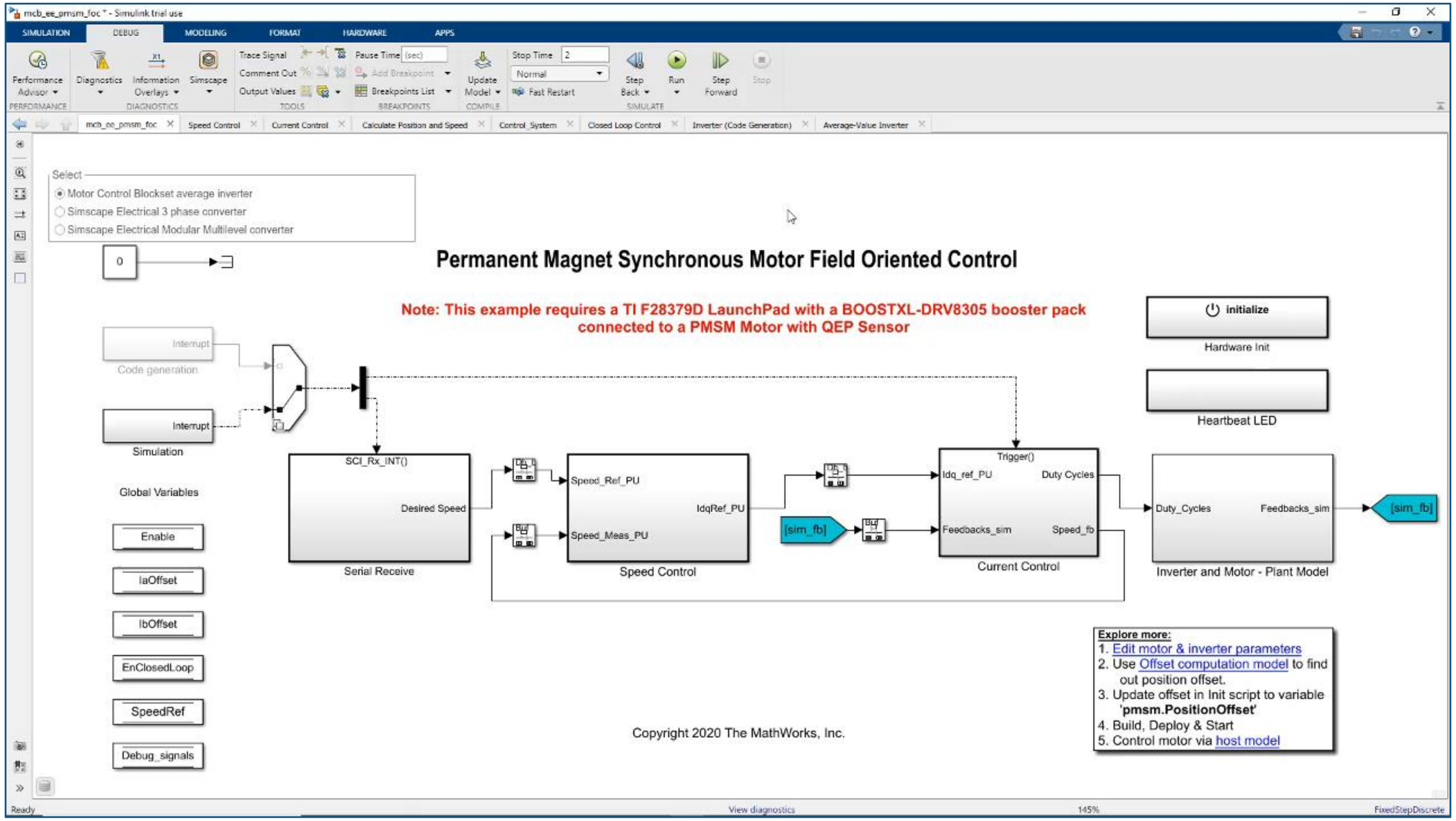
经典控制理论

Simulink Control Design

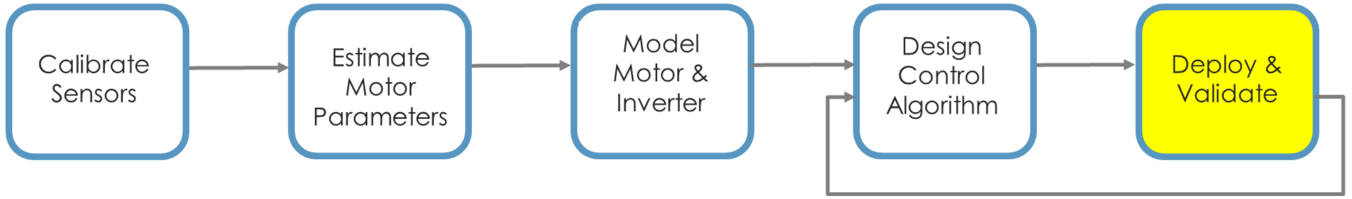
控制算法设计



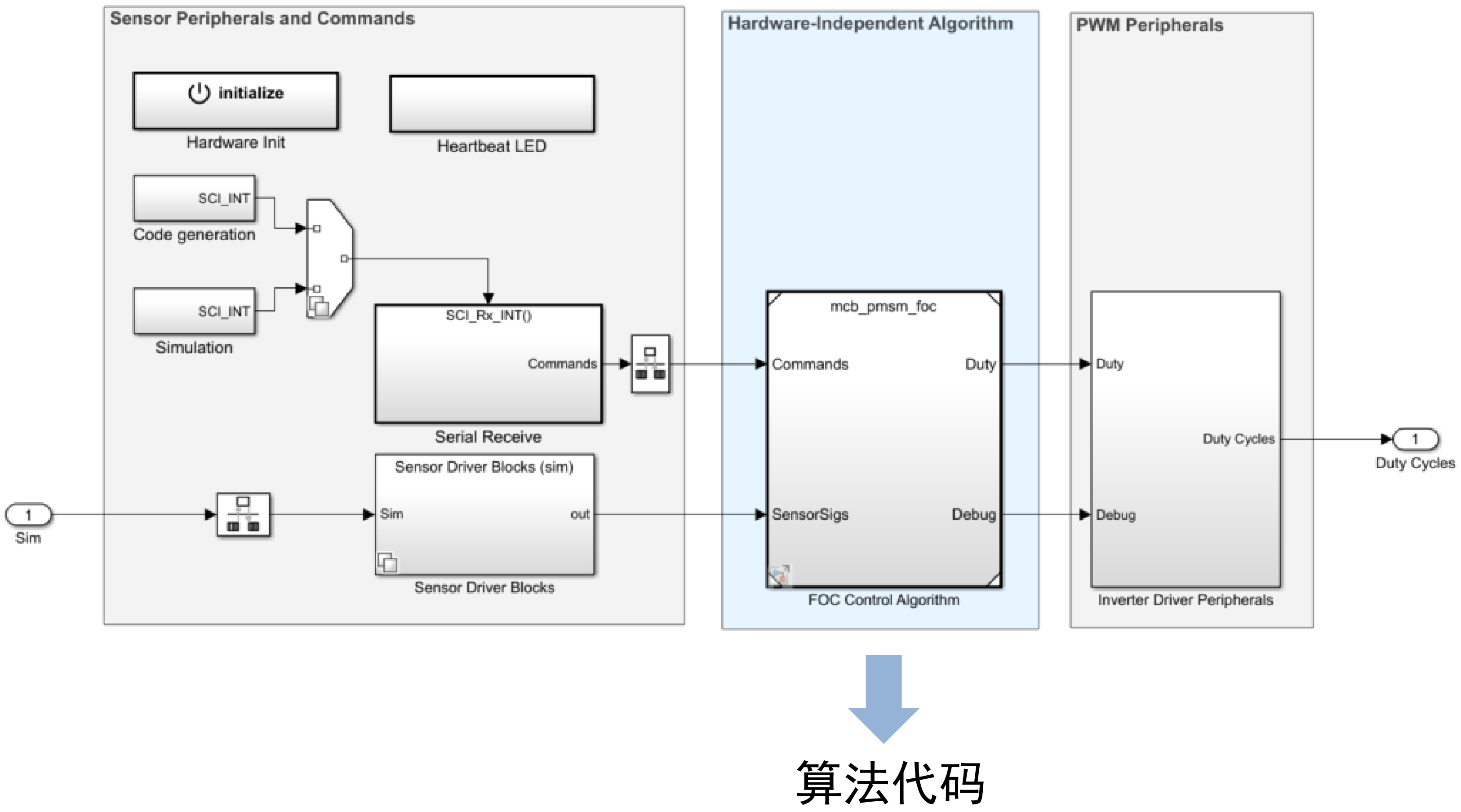
- FOC算法建模
- 解码器或无传感器观测器建模
- 调节闭环参数
- 闭环仿真验证



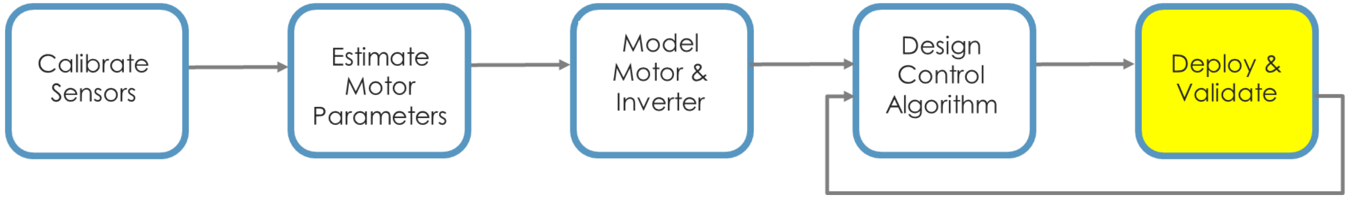
软件部署



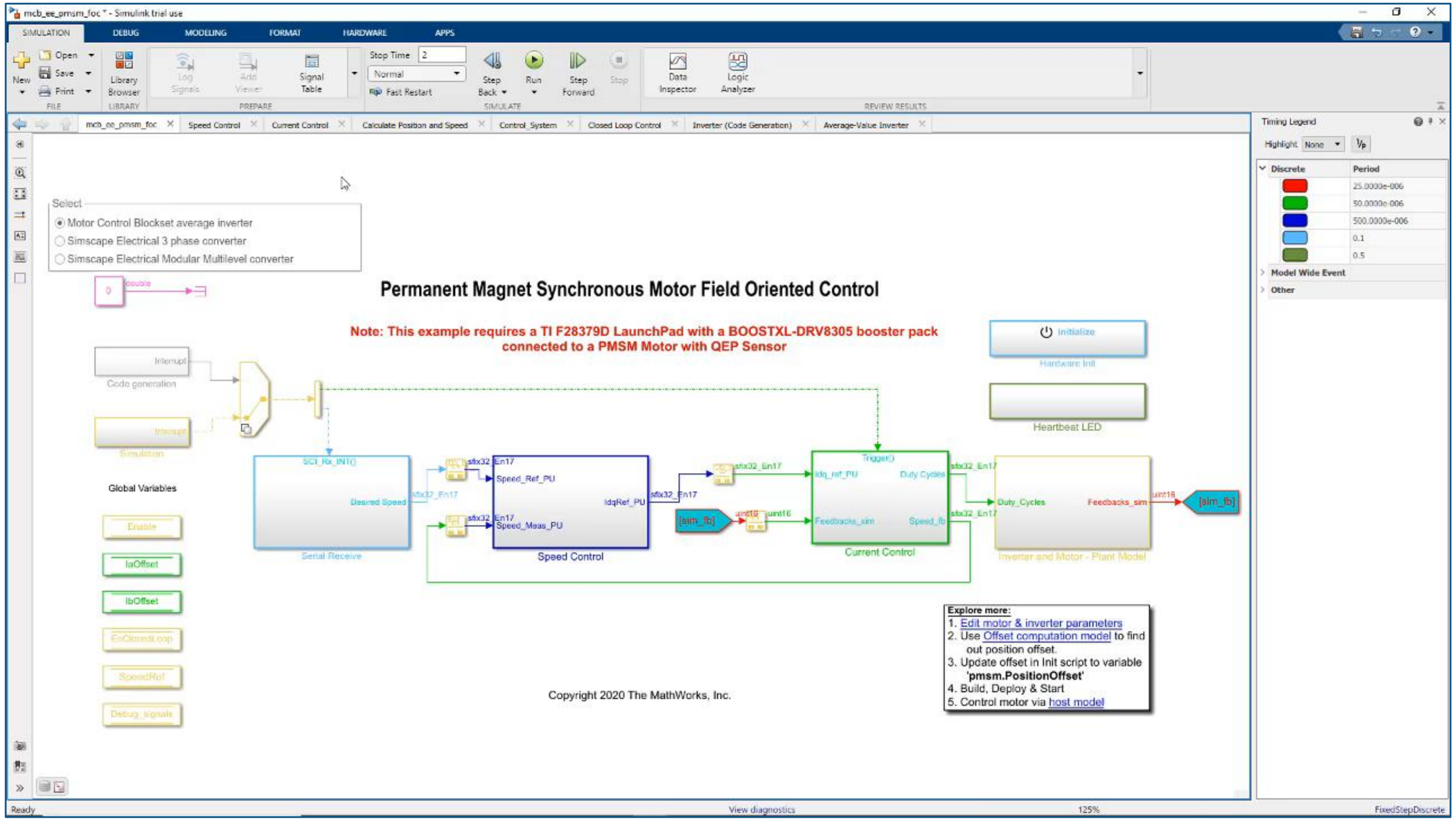
- 针对任意硬件的ANSI C标准代码
- 将模型划分为算法和驱动的示例
- 生成嵌入式应用算法



软件部署



- 生成浮点或定点算法
- 使用host 模型进行调试
- 在硬件上验证



MATLAB EXPO

使用PIL验证代码和效率

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	report(executionProfile, 'Units', 'seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%0.0f');
Timer frequency (ticks per second)	2e+08
Profiling data created	16-Jan-2020 18:09:48

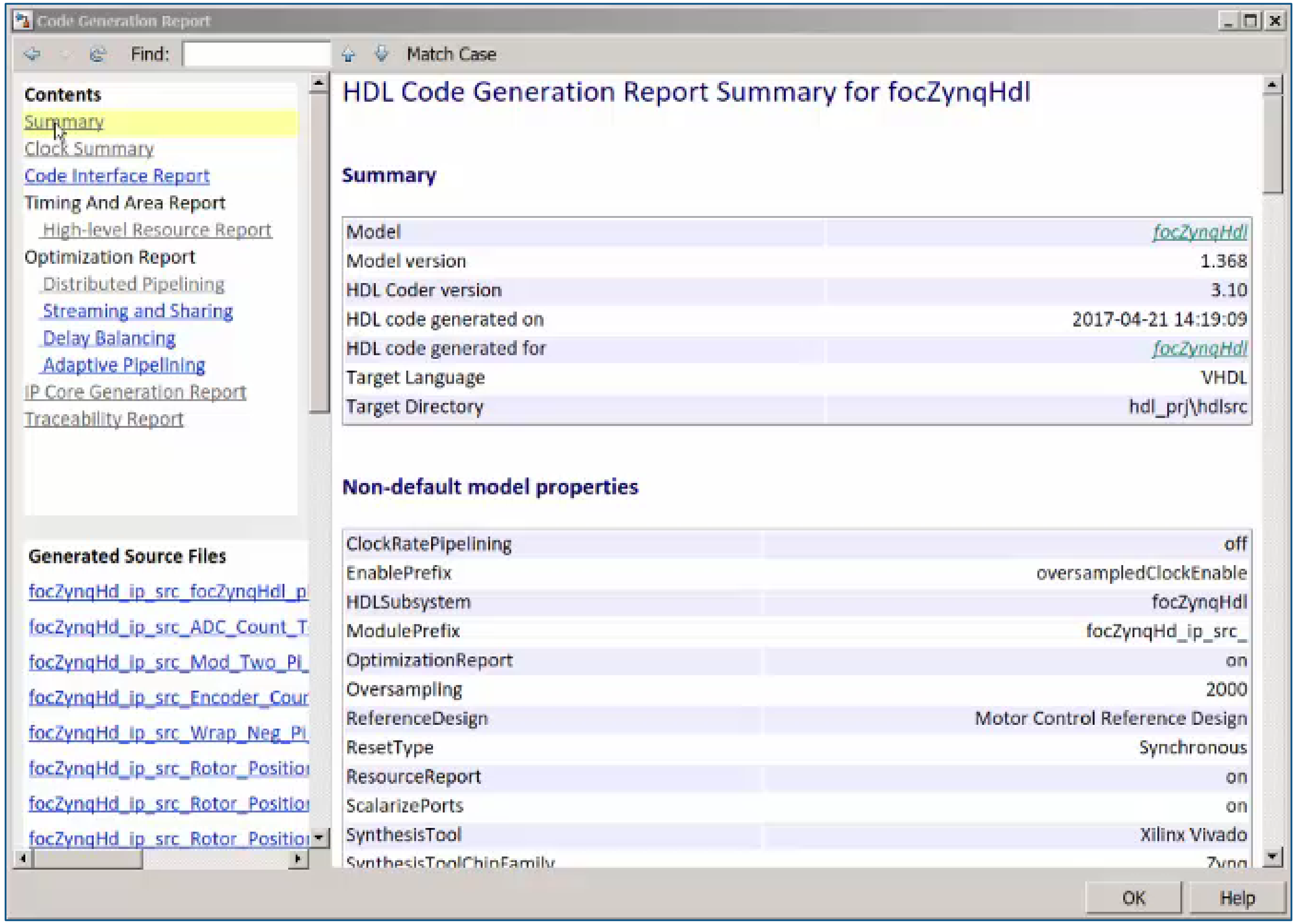
2. Profiled Sections of Code

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%

FPGA代码生成



HDL Code Generation

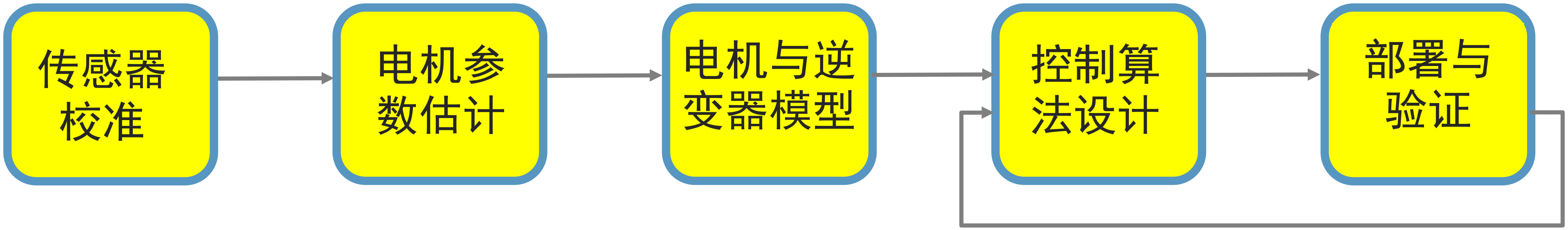
邦奇动力(Punch Powertrain)基于SoC开发复杂电机驱动控制系统

- 任务：开发新一代电动和混动车的动力总成系统
- 目标：降低成本的同时提升能量密度和效率
  - 集成电机和电力电子部件到传动系统
- 方案：采用新的开关磁阻电机设计
  - 电机最高转速是以前的两倍
  - 新的硬件平台Xilinx® Zynq® SoC 7045
- 挑战：没有FPGA设计经验!

- ✓ 实现了电驱系统的机电软集成设计
- ✓ 开发了四种不同的控制策略
  - 2名工程师，18个月
- ✓ 得到了可重用于验证的模型
- ✓ 建立了无缝的集成验证流程
  - 在硬件生产前进行充分验证
  - 大大减少了试验台架验证时间

Link to video of presentation

FOC 开发工作流



ATB Technologies 通过使用适用于 TI C2000 MCU 的代码生成使电机控制器的开发时间缩短了一半

Challenge

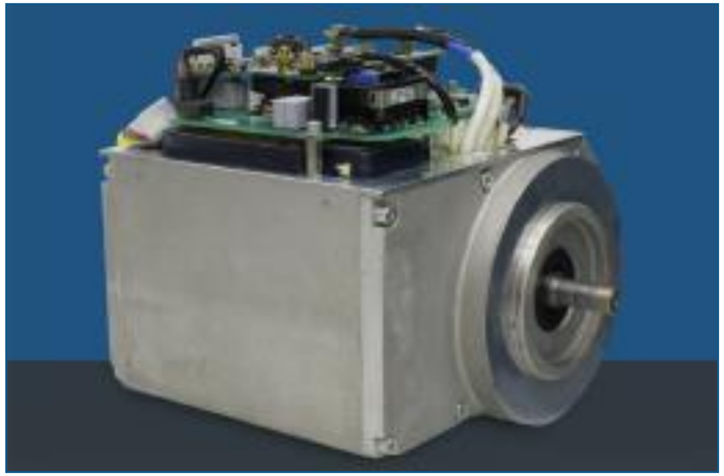
开发控制软件，最大程度实现永磁同步马达的效 率和性能

Solution

使用 MathWorks 基于模型的设计工具，对控 制系统进行建模和仿真，并将其实现在目标处 理器上

Results

- 缩短了一半的开发时间
- 简化了设计评审
- 加速了目标验证和部署



ATB Technologies 的永磁同步马达

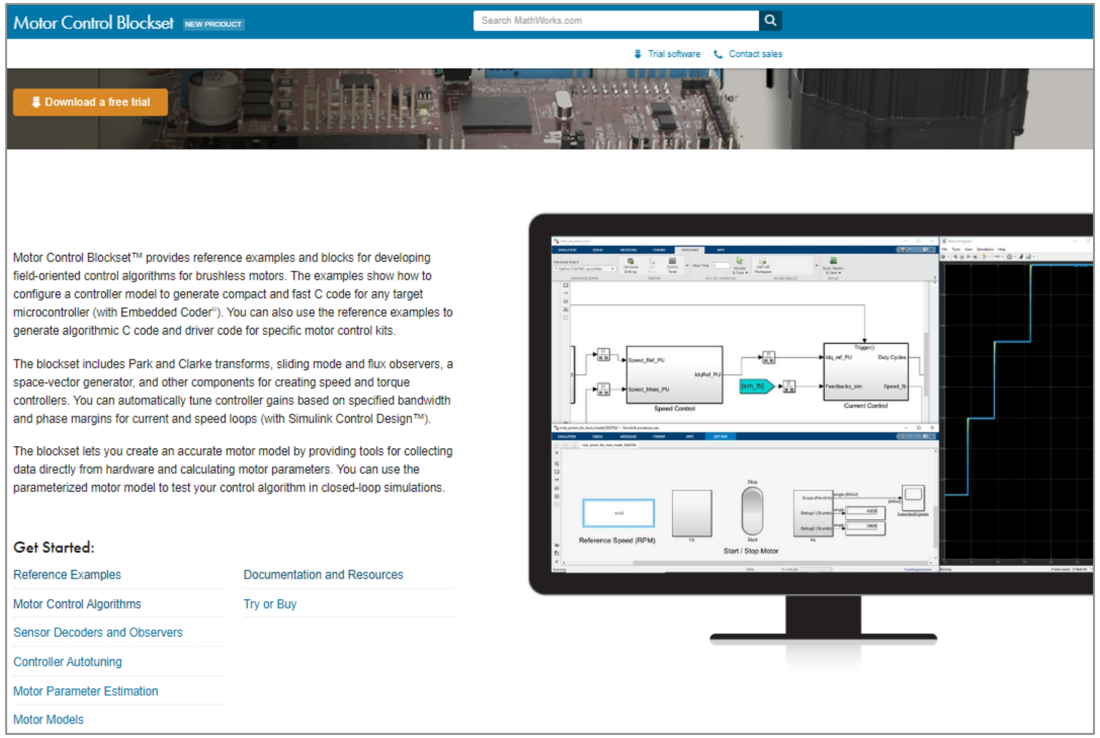
“MathWorks 工具让我们可以在开发的多个阶段验证设计的质量，并在较短时间期限内制造出高质量的元件。”

- Markus Schertler, ATB Technologies

Link to user story

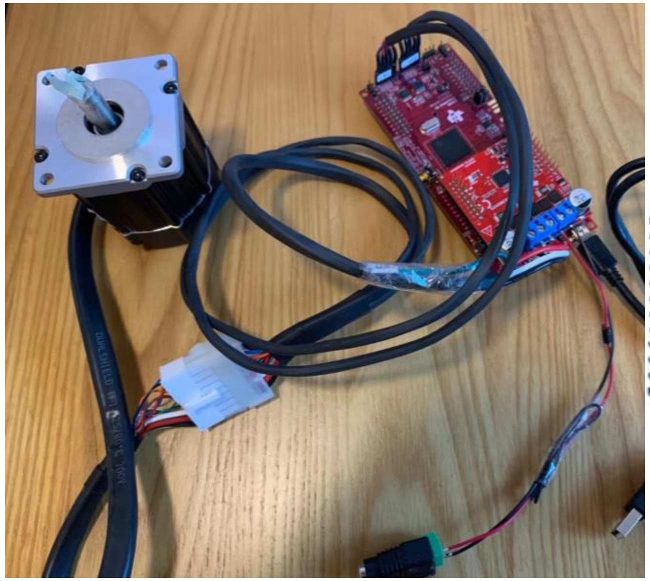
使用基于模型的设计方法开发你的下一个电机控制产品

- 桌面仿真验证控制算法
- 从模型生成精简且高效的代码
- 利用参考模型、内建算法、自动参数估计和控制调参等多种手段缩短开发周期



电机培训 - 5天实战课程

- **课程交付内容**
  - Simulink电机本体建模
  - Simulink电机控制器设计和优化-从算法到模型
  - 电机控制算法代码生成和硬件部署-从模型到C/C++代码
  - 控制算法的确认验证（V&V）



Day 1

- Introduction
- Motors and Motor Control Algorithms
- DC Motor and Controls
- BLDC Plant Modeling
- PMSM Plant Modeling

Day 2

- Open-loop Control of PM
- Closed-loop Control of PM
- Sensors
- BLDC Controls
- PMSM Controls

Day 3

- Control Tuning
- Analyze and Evaluate Uncertainties
- Fault Diagnostics and Protection
- Sensorless Control
- Rapid Control Prototype
- Code Generation

Day 4

- Code Optimization
- Model-in-the-Loop Testing
- SIL and PIL Testing and Profiling
- Code Deployment

Day 5

- HIL Testing
- Standards Compliance
- Conclusions

Appendices

- Software Environment Setup
- Hardware Environment Setup
- Model Configuration Parameters
- Fixed-point Precision Control
- Speedgoat Target Machine Setup
- App Designer

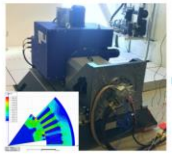
工程服务 - 弱磁标定

工程支持

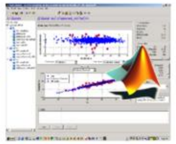
- 流程讨论
- 方法教学

标定工具

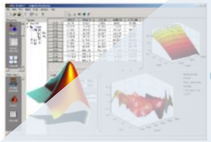
- 用户界面
- 按需定制



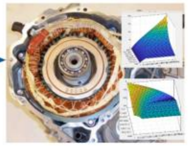
数据采集



数据建模

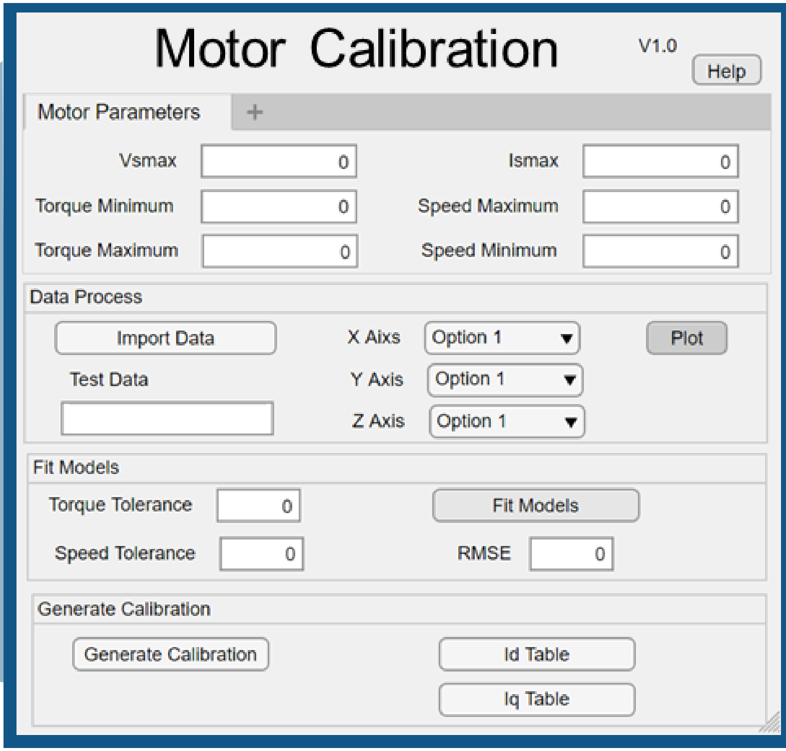


虚拟标定

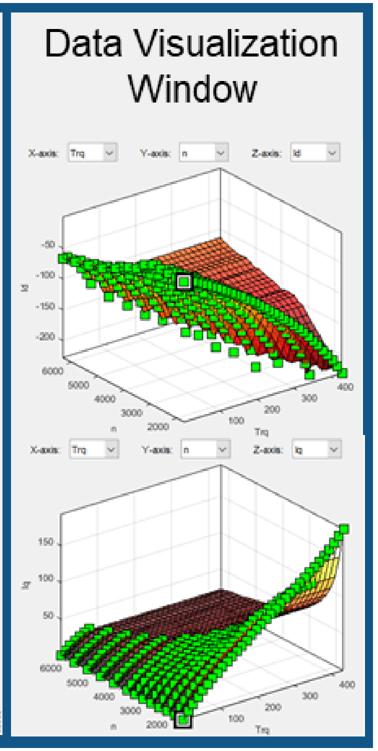


算法实现

标定流程工具

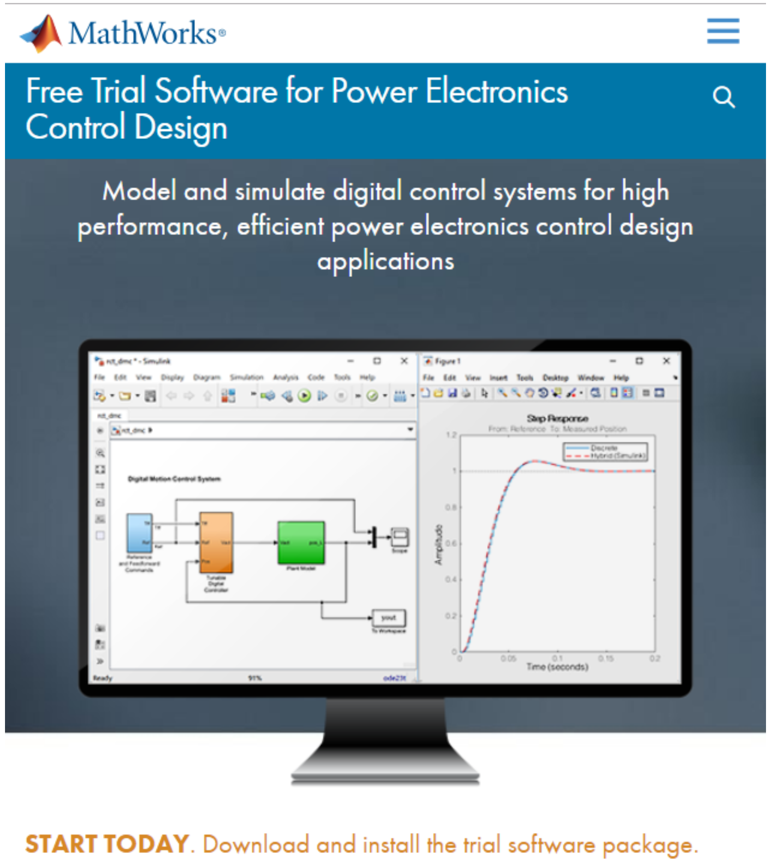


数据可视化



更多内容

- 访问 mathworks.com/products/motor-control 和 mathworks.com/solutions/power-electronics-control
- 参加MathWorks其他电力电子讲座
- 试用电力电子设计工具包，利用桌面建模、仿真、控制设计和产品级代码生成进行电机控制开发



MATLAB EXPO

2021

Thank you

MathWorks®