Simulink를 이용한 브러시리스 모터 제어 개발
강효석, MathWorks Korea
Spinning a Brushless Motor Using Simulink and Model-Based Design
Brushless Motors Are Everywhere
Developing Embedded Motor Control Software Has Its Challenges

ITK Engineering Develops IEC 62304–Compliant Controller for Dental Drill Motor with Model-Based Design

Challenge
Develop and implement field-oriented controller software for sensorless brushless DC motors for use in dental drills

Solution
Use Model-Based Design with Simulink, Stateflow, and Embedded Coder to model the controller and plant, run closed-loop simulations, generate production code, and streamline unit testing

Results
▪ Development time halved
▪ Hardware problems discovered early
▪ Contract won, client confidence established

“Model-Based Design with Simulink enabled us to reduce costs and project risk through early verification, shorten time to market on an IEC 62304–certified system, and deliver high-quality production code that was first-time right.”
- Michael Schwarz, ITK Engineering

Dental drills featuring ITK Engineering’s sensorless brushless motor control.
Developing Embedded Motor Control Software Has Its Challenges

- Design work needed to be started **before motor hardware** was available and needed extensive testing to comply with standards.

- Team needed to **rapidly implement control software** on embedded processor once more hardware became available.

- Complex algorithms running at **high sample rates** were difficult to implement in short amount of time.

---

**ITK Engineering Develops IEC 62304–Compliant Controller for Dental Drill Motor with Model-Based Design**

**Challenge**
Develop and implement field-oriented controller software for sensorless brushless DC motors for use in dental drills

**Solution**
Use Model-Based Design with Simulink, Stateflow, and Embedded Coder to model the controller and plant, run closed-loop simulations, generate production code, and streamline unit testing.

**Results**
- Development time halved
- Hardware problems discovered early
- Contract won, client confidence established

> “Model-Based Design with Simulink enabled us to reduce costs and project risk through early verification, shorten time to market on an IEC 62304–certified system, and deliver high-quality production code that was first-time right.”
> - Michael Schwarz, ITK Engineering
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
Field-Oriented Control (FOC) for Brushless Motors

- Control algorithm
- Physical system

- (Optional) Velocity Controller
- Current Reference Generator
- Current Controller
- SVPWM Generator
- Power Inverter

- Voltage Supply

- ω_ref → T_ref → i_{d,ref}
- i_{d,ref} → v_{d,ref}
- v_{d,ref} → α, β
- α, β → v_{α,ref}
- v_{α,ref} → SVPWM Generator

- Protection and auxiliary functions

- Position Sensor Signal Processing

- θ_e → ω → v_{dc}
- v_{dc} → i_d
- i_d → i_q
- i_q → a, b, c
- a, b, c → d, q

- Park, Clarke Transforms
- d, q → i_a, i_b, i_c

- PMSM
Workflow for Implementing Field-Oriented Control

1. Calibrate Sensors
2. Estimate Motor Parameters
3. Model Motor & Inverter
4. Design Control Algorithm
5. Deploy & Validate

MATLAB EXPO
We Will Use Texas Instruments Motor Control Kit

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

- Calibrate ADC offsets
- Calibrate position sensor offset
Sensor Calibration

ADC Offsets Calibration

Development Computer
ADC offset calibration model

Target Hardware

1. Code Generation
2. Compile and Link
3. Download and Ready to Run
Sensor Calibration
ADC Offsets Calibration

Development Computer
Host Model

Target Hardware

Calibrate Sensors
Estimate Motor Parameters
Model Motor & Inverter
Design Control Algorithm
Deploy & Validate

4 External mode Simulation
Sensor Calibration
ADC Offsets Calibration

Development Computer
Host Model

Target Hardware

4. External mode Simulation
5. Get offset for phase A and B

Calibrate Sensors → Estimate Motor Parameters → Model Motor & Inverter → Design Control Algorithm → Deploy & Validate

Serial connection
Sensor Calibration
ADC Offsets Calibration

Development Computer
Host Model

Target Hardware

4 External mode Simulation
5 Get offset for phase A and B
6 Enter these offsets into a setup script

Calibrate Sensors
Estimate Motor Parameters
Model Motor & Inverter
Design Control Algorithm
Deploy & Validate

MATLAB EXPO
Sensor Calibration

- Calibrate ADC offsets
- Calibrate position sensor offset
Parameter Estimation

- Instrumented tests running on the target
- Host model to start and control parameter estimation
Bonus: Other Techniques to Parameterize Motor Models

From datasheet
Simscape Electrical

From ANSYS Maxwell, J MAG, Motor-CAD FEA tools

From dyno data
Powertrain Blockset

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 `CreatingIDTable.m` calls `gridfit` to model the current surface using scattered or semi-scattered flux data.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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. |
Modeling Motor and Inverter

- Use linear lumped-parameter motor model
- Model inverter as an average-value inverter or model switching with Simscape Electrical
Bonus: Modeling at Needed Level of Fidelity

- **Lumped Parameter**
  - Motor Control Blockset
  - Simscape Electrical

- **Saturation**
  - Simscape Electrical

- **Saturation + Spatial Harmonics**
  - Simscape Electrical
Bonus: Motor Modeling Using Simscape Electrical
Nonlinear PMSM Model

- Define PMSM behavior using d- and q-axis flux linkage

- Parameterization option is directly compatible with Maxwell, JMAG and Motor-CAD data
  - With a few changes to text file, MATLAB variables that match block parametrization can be generated
Control Algorithm Design

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

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

Input:
- pmsm: Motor object
- inverter: Inverter object
- PU_System: Per-Unit System
- T_pwm: PWM switching time period
- Ts: Sample time for current controllers
- T_speed: Sample time for speed controller
Control Algorithm Design

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

mcb_getControlAnalysis(pmsm,inverter,PU_System PI_params Ts,Ts_speed);
Bonus: Several Techniques to Tune Loop Gains

Empirical Computation
Motor Control Blockset

FOC Autotuner
Motor Control Blockset and Simulink Control Design

Classic Control Theory
Simulink Control Design
Control Algorithm Design

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

- 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
Deployment

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

- Generate code (floating and fixed-point)
- Use host model to control and debug
- Validate on hardware
Verify and Profile Code Using Processor-In-the-Loop (PIL) Testing

Code Execution Profiling Report for mcb_pmsm_foc_sim_v2/Current Controller

1. Summary

Total time
Unit of time
Command
Timer frequency (ticks per second)
Profiling data created

2. Profiled Sections of Code

Section | Maximum Execution Time in ns | Average Execution Time in ns | Maximum Execution Time in ns
--- | --- | --- | ---
[+] Current_initialize | 2260 | 2260 | 
Current_step [5e-05 0] | 5135 | 5067 | 
Current.terminate | 540 | 540 | 

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%
Bonus: Code Generation for MCU/FPGA/SoC

- MCU or FPGA: C or HDL Code Generation through Coders
- SoC: Need to consider interface between ARM and FPGA (AXI-Bus)

Functions Supported for HDL Code Generation
- HDL Code Generation from Simulink — Blocks

Embedded Coder: C/C++
HDL Coder: VHDL/Verilog
HDL Coder Automatically Generates the AXI-interface

ARM Processor
- Mode Select
- Safety Function
- Velocity Control
- Encoder Calibration

FPGA
- Conversion
- Current Control
- Current Limiter
- Angle - Velocity Estimation
- PWM

Driver Circuit
- Isolated
- Inverter Module
- Encoder
Workflow for Implementing Field-Oriented Control

1. Calibrate Sensors
2. Estimate Motor Parameters
3. Model Motor & Inverter
4. Design Control Algorithm
5. Deploy & Validate

The process involves calibrating sensors, estimating motor parameters, modeling the motor and inverter, designing the control algorithm, and finally deploying and validating the system.
ATB Technologies Cuts Electric Motor Controller Development Time by **50%** Using Code Generation for TI’s C2000 MCU

**Challenge**
Develop control software to maximize the efficiency and performance of a permanent magnet synchronous motor

**Solution**
Use MathWorks tools for Model-Based Design to model, simulate, and implement the control system on a target processor

**Results**
- Development time cut in half
- Design reviews simplified
- Target verification and deployment accelerated

“MathWorks tools enabled us to verify the quality of our design at multiple stages of development, and to produce a high-quality component within a short time frame.”
- Markus Schertler, ATB Technologies
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, sensor calibration, built-in algorithmic blocks, automated parameter estimation, and gain-tuning
Learn More

- Visit mathworks.com/products/motor-control and mathworks.com/solutions/power-electronics-control

- 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
Thank You !!