

MATLAB EXPO

November 13–14, 2024 | Online

From Idea to MCU Deployment: Applying Tiny Machine Learning on FOC for PMSMs

Danilo Pau, Technical Director, IEEE and ST Fellow

System Research and Applications, STMicroelectronics



Motivations for interest

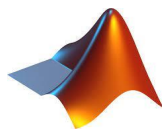


Developing Edge AI in the context of Motor Electrification poses challenges due to the well-known Field Oriented Control technique.

Introducing AI mandates to optimize accuracy, execution speed and energy efficiency, which requires a joint understanding of both AI and motor control systems.

The combination of MathWorks and STMicroelectronics AI methodologies and tools simplifies this process, easing efficient deployment of AI models to MCUs.

Let's review together how this can be achieved.



Key Words

IDEA

Interoperability

pmsm

Machine Learning

Methodology

Productivity

Development

Automation

Journey

FOC

Non Linearities

Embedded Systems



Methodology

D\$

M\$

O\$

MCU\$

DATA ACQUISITION

MODELING TINY NEURAL NETWORK

OPTIMIZATIONS

DEPLOY ON ST MCUs

ST Edge AI Developer Cloud

D1 Modelize the PMSM FOC control loop.

D2 Define Case Studies based on fast changing speed.

D3 Build datasets with PIDs highlighting their limits.

D4 Run the experiments to record enough data.

M1 Set data container to access the data correctly during training steps.

M2 Devise a feed forward NN for compensating PID errors.

M3 Measure the deployability of the network by ST Edge AI Developer Cloud.

M4 Run the training progress and the seek for the best model configuration during validation.

O1 Seek for the most accurate model on the test set.

O2 Perform Hyper Parameters tuning for the best compromise as model exploring configurations.

O3 Prune the model for low-cost deployment on the device.

O4 Optimize model performance monitoring in the control loop (add the NN to the PID).

MCU1 Import and PTQ model (ONNX) by the ST Edge AI Developer Cloud.

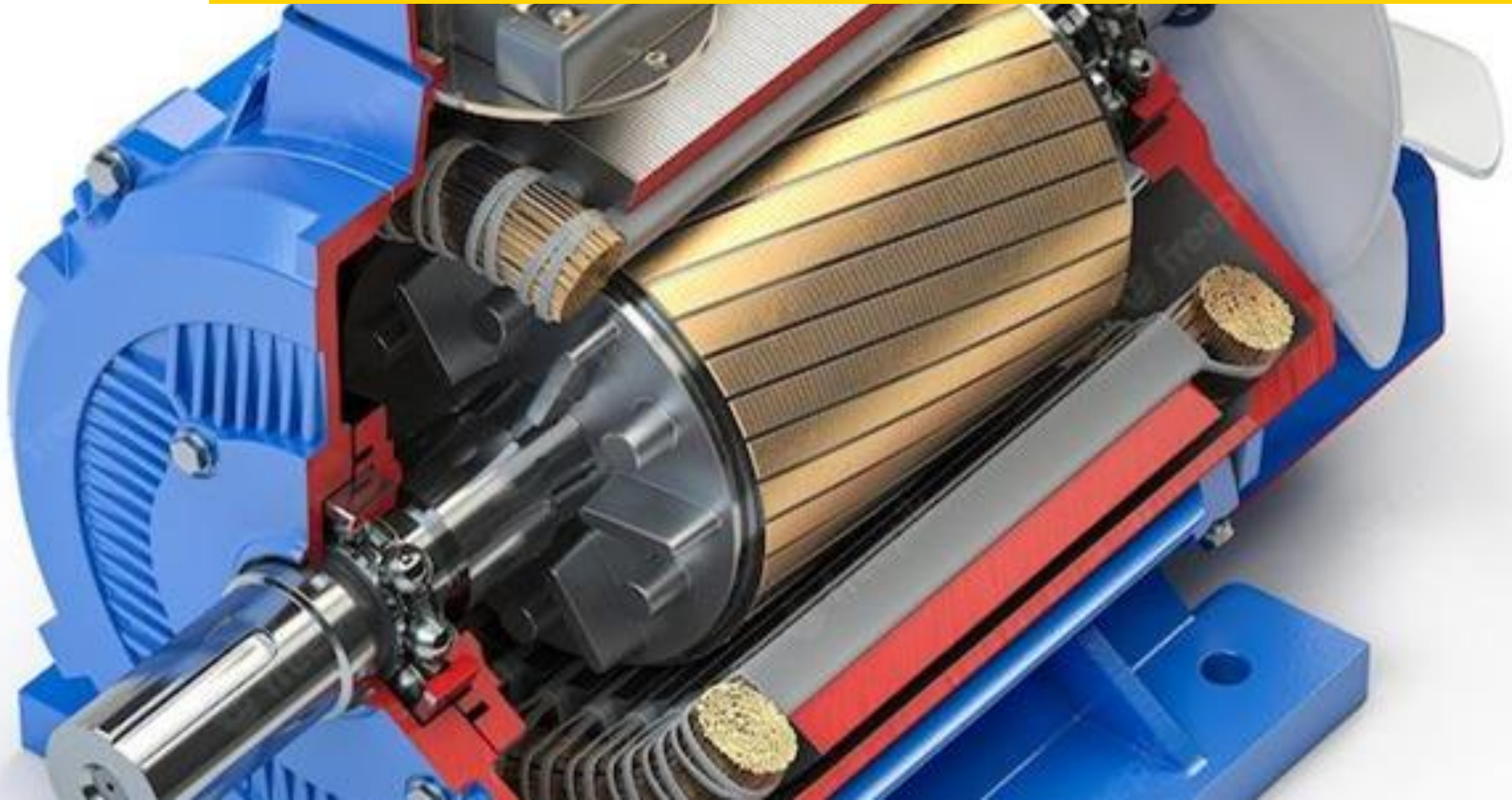
MCU2 Optimize the deployment (required RAM size w.r.t. inference time).

MCU3 Benchmark on ST MCUs and measure the inference time. Export detailed logs.

MCU4 Loop between PHASE 2, 3 and 4 until a satisfactory solution can be signed off.

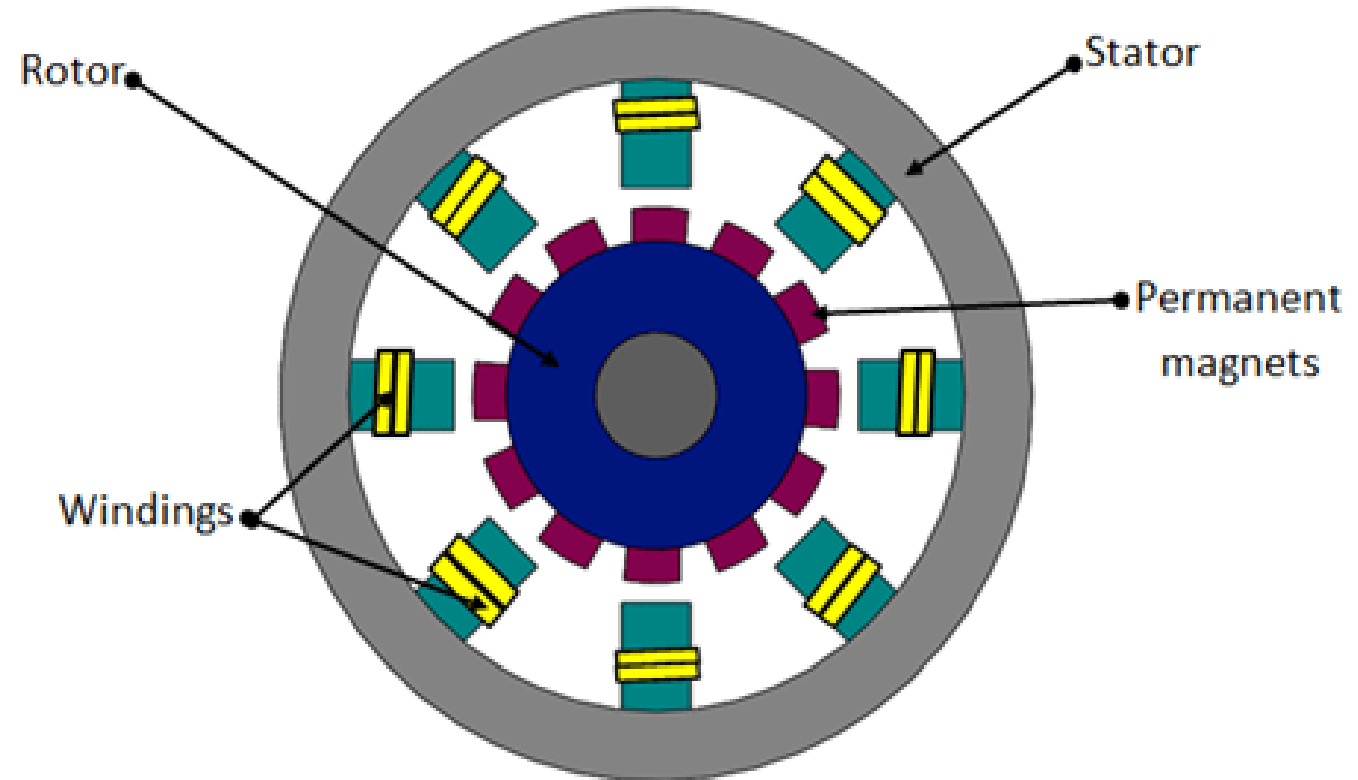
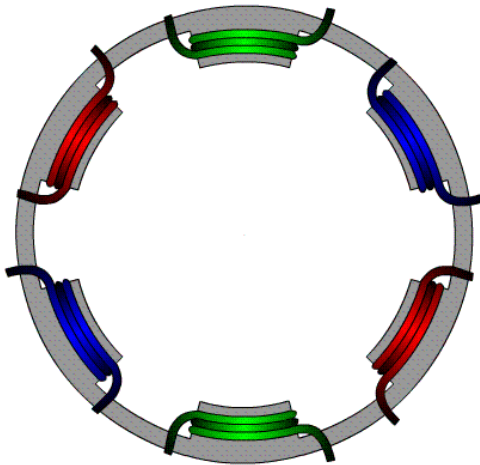


Introduction

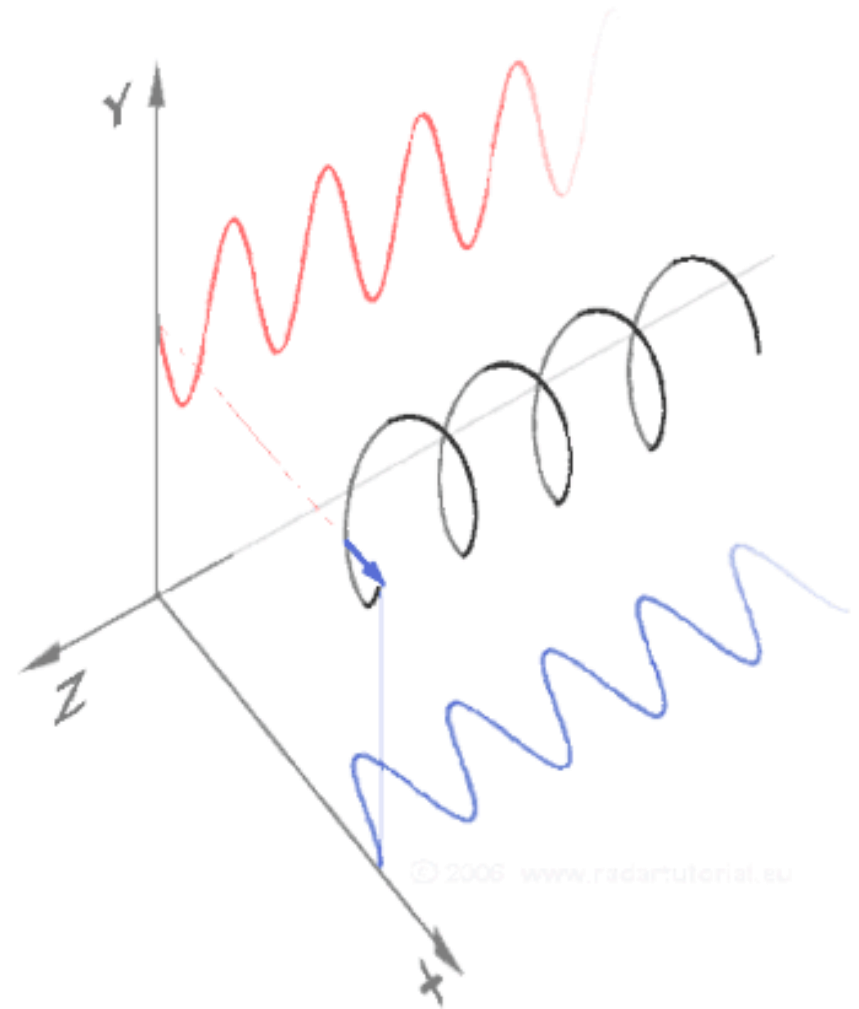
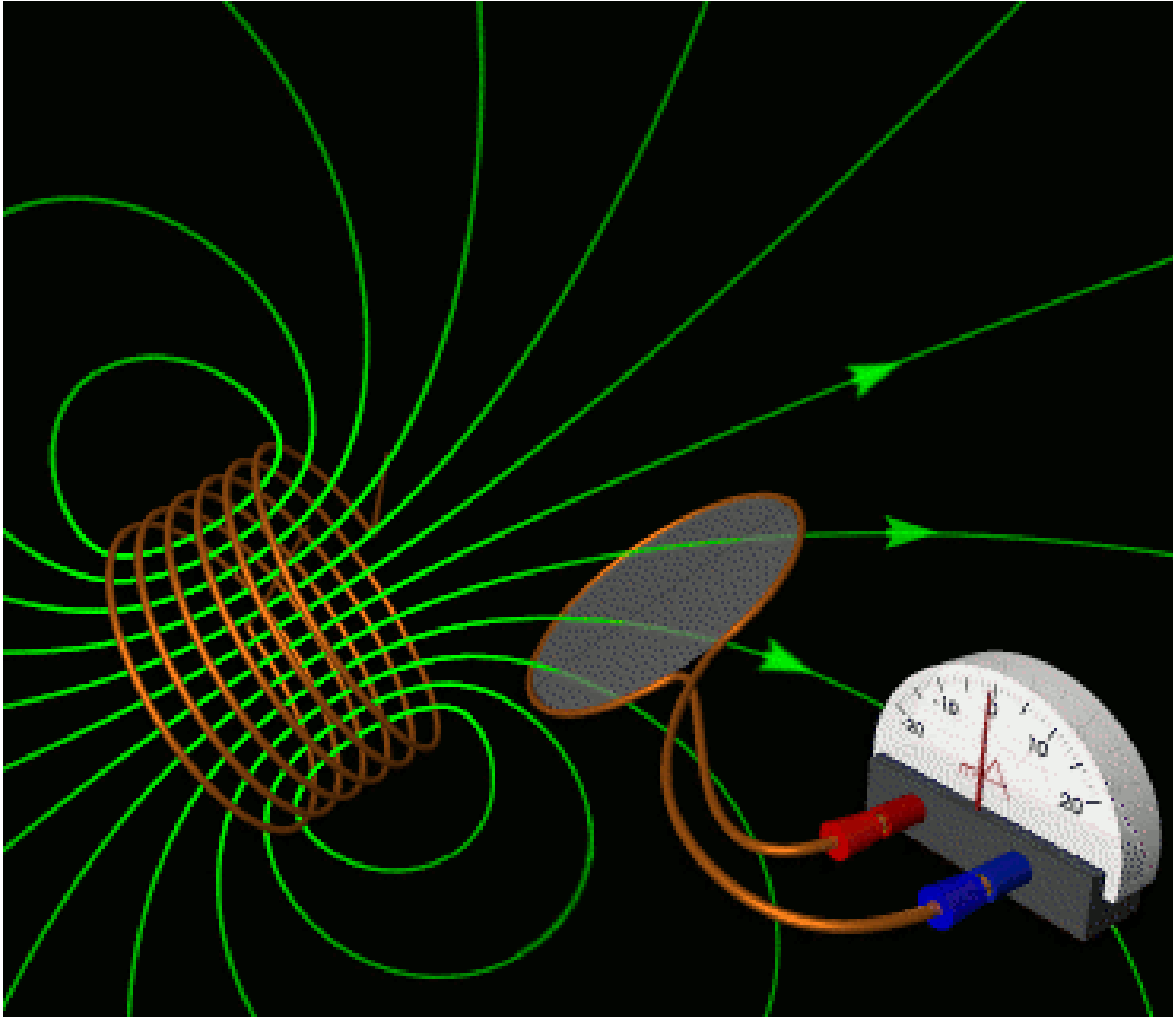


Permanent Magnet Synchronous Motor

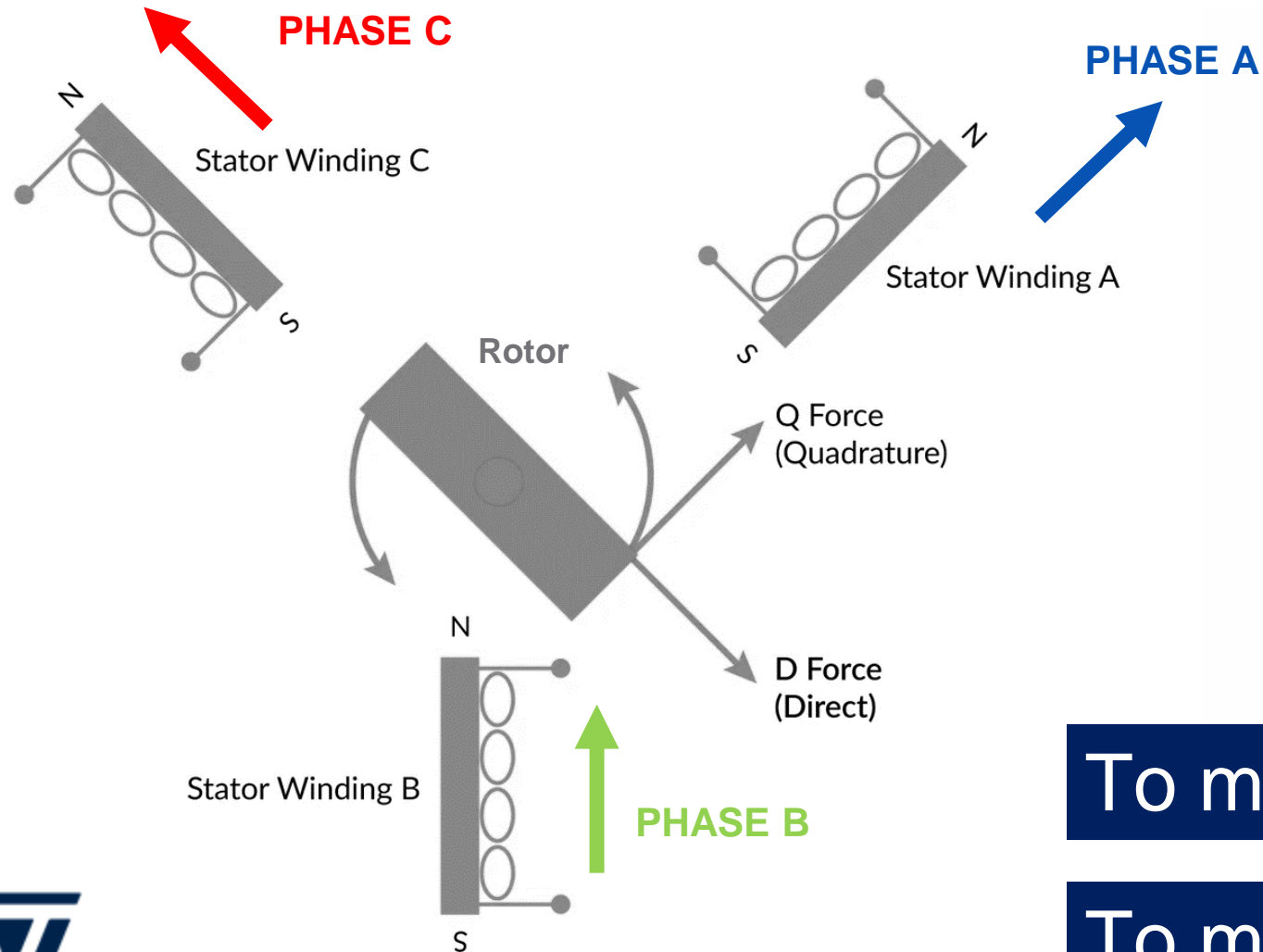
- The magnetic field of the permanent magnets placed on the rotor interacts with the one created by the synchronous sinusoidal alternating current in the stator windings.
- This interaction produces a torque, which causes the rotor to rotate.
- The EMF (Electromagnetic Field) force shall be controlled to produce the required torque over the time.



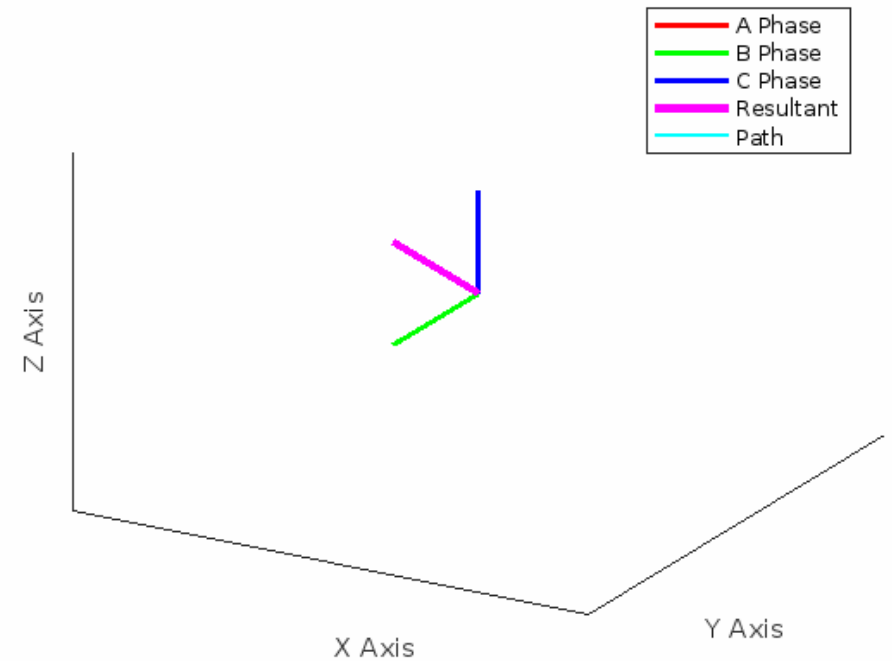
Time Varying Magnetic Field



Mission is to generate rotations



Visualization of 3-Phase Current Vector



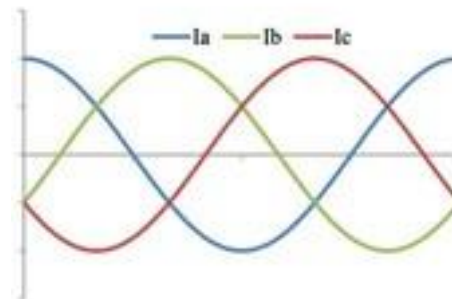
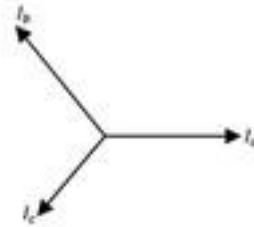
To maximize Q Force

To minimize D Force

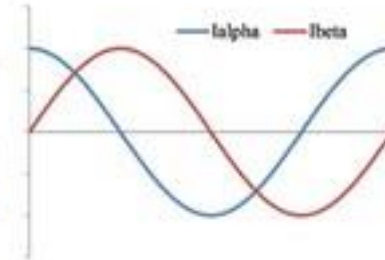
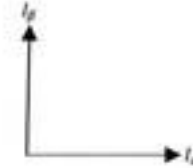
Field-Oriented Control

- In **FOC**, simplified, PID control based is required.
- To achieve that, Voltage and Current signals shall be no longer sinusoidal but direct so that the control loop occurs de-referenced from the 3D vector's rotation.
- This happens through the Clark and Park (and their inverse) transforms.
- The **time varying three-phase system** in rotor's ABC reference frame is transformed to **time invariant D Q** components.

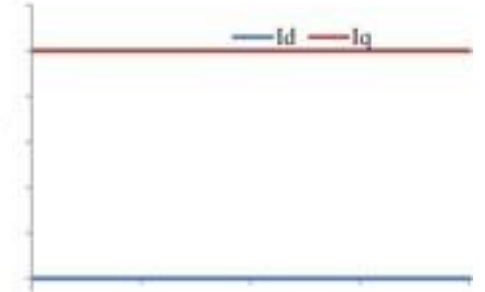
Three Phase 120° Reference Frame



Two Phase Reference Frame



Rotating Reference Frame



FOC for PMSM

Higher top speed

High energy efficiency

>> 97% to 99.5% <<

Essential for BEVs

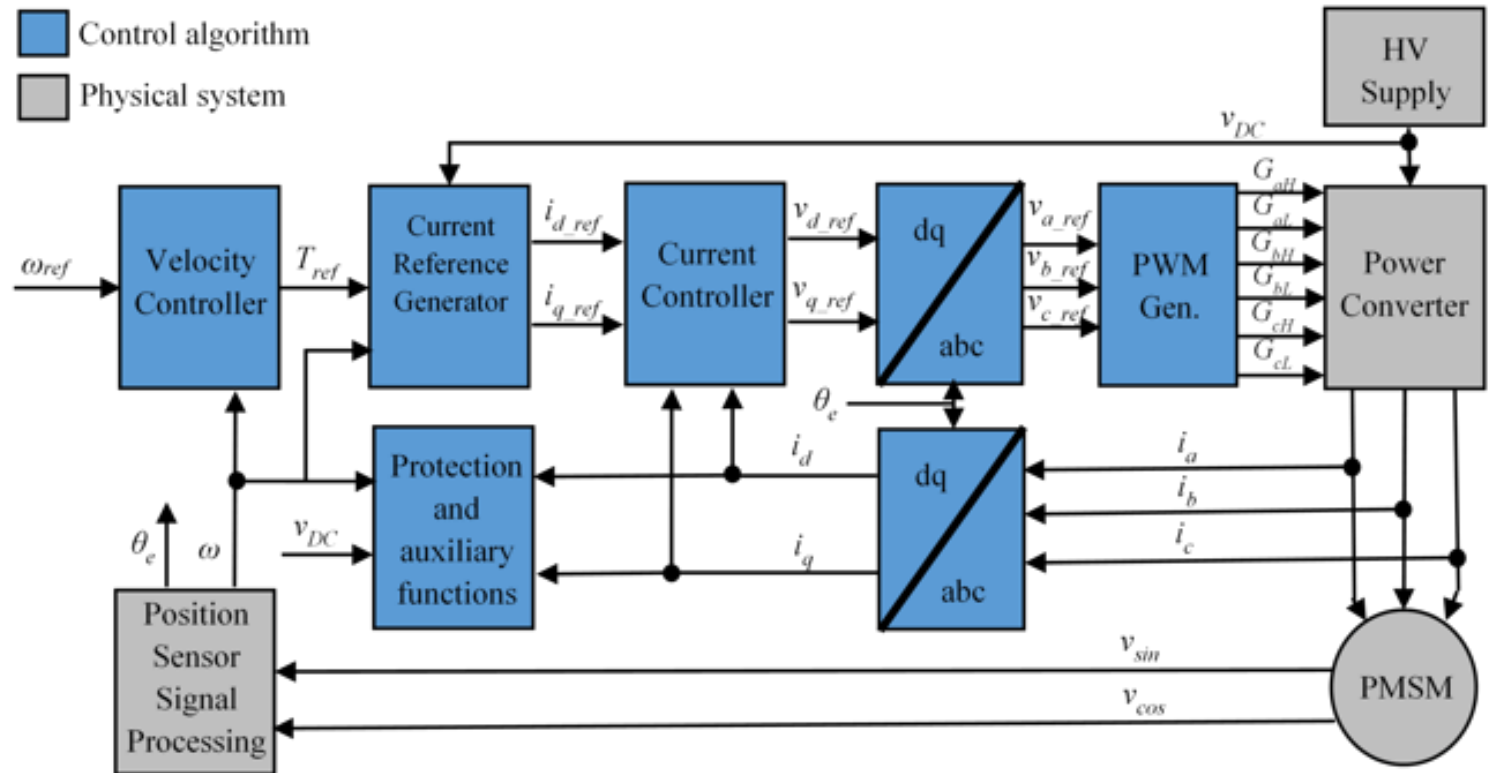


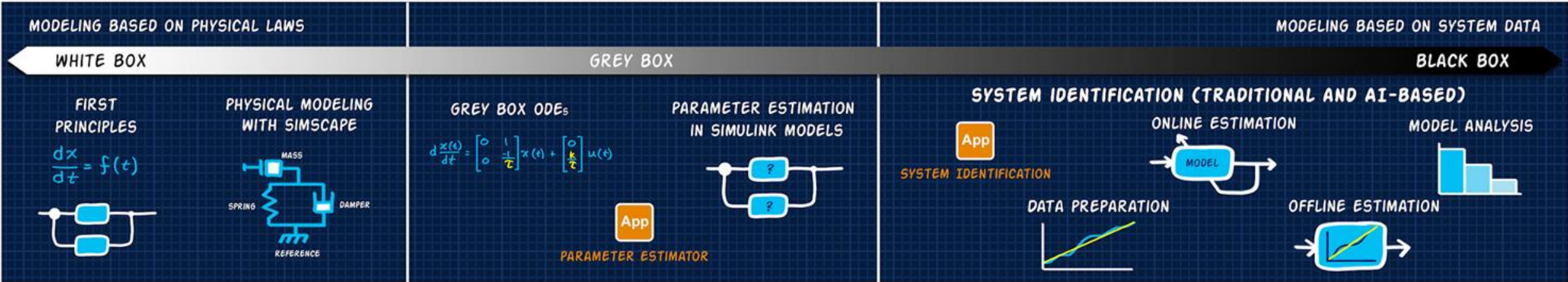
Image Source: <https://it.mathworks.com/help/sps/ref/pmsmfieldorientedcontrol.html>

Simulink Modeling



Model Parameters

Determine model parameters through first principles, grey box, and data-driven methods.



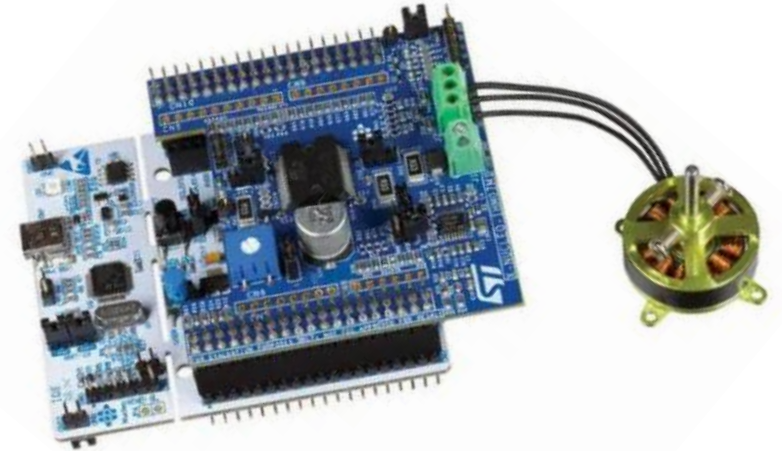
Model Manipulation

Modify models through transformation, linearization, and order reduction methods.



Exemplary Motor

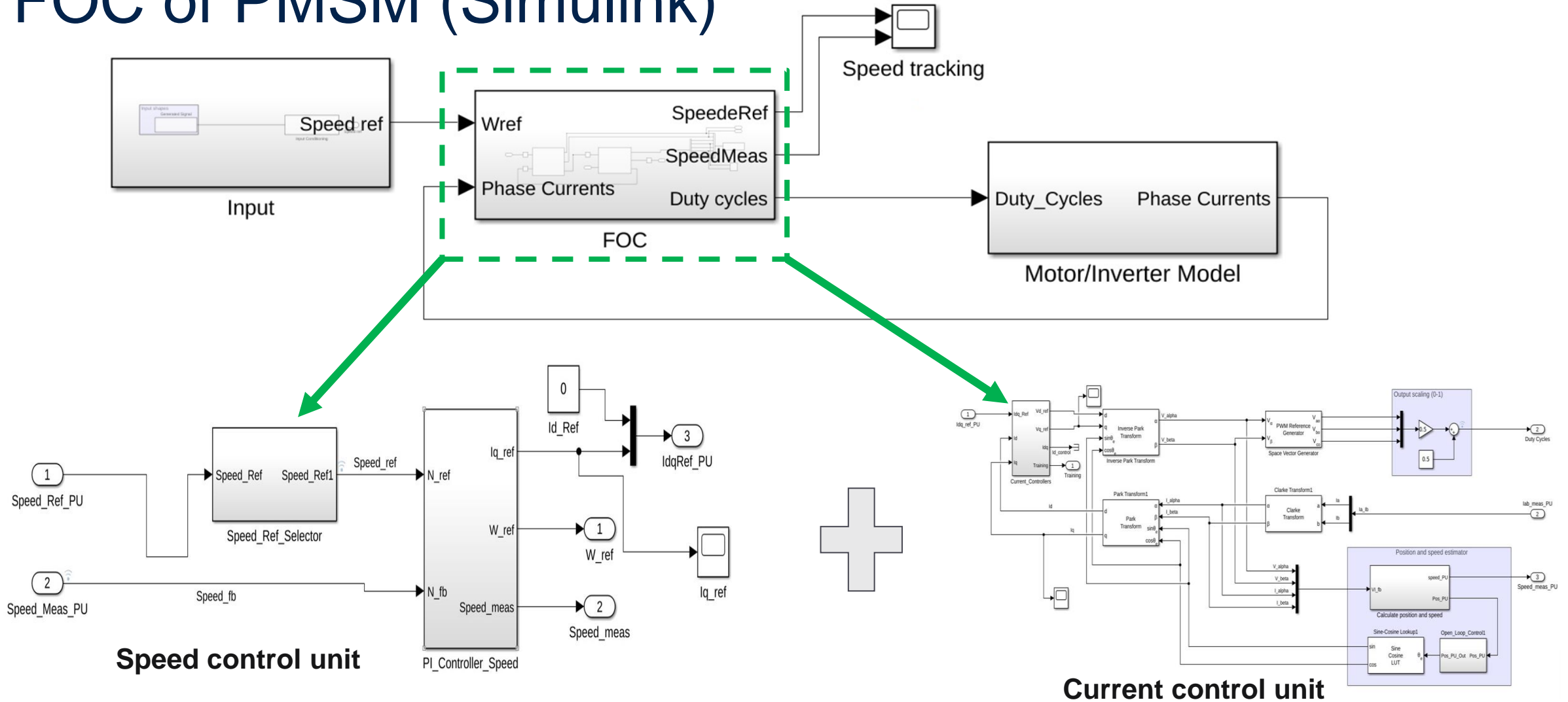
- The **BR2804-1700KV** motor operates at a nominal voltage of 11.1V, within the **X-NUCLEO-IHM07M1**'s 8-48V range.
- Additionally, the motor's maximum current of 5A aligns closely with the board's 2.8A output peak current per phase, making it a **safe and effective for educational purposes**.
- The motor's 7 pole pairs are well-suited for FOC, which is efficiently handled by the **X-NUCLEO-IHM07M1** board, ensuring high torque and smooth operation, crucial for precision control applications.
- This motor was used to parametrize the Simulink model



<https://www.st.com/en/evaluation-tools/p-nucleo-ihm001.html>

Model: Bull-Running model **BR2804-1700 kV**
Nominal voltage: **11.1 V DC** (battery up to 3S)
Maximum DC current: **5 A**
Poles: **7 pole pairs**
Max speed: **19,000 RPM**

FOC of PMSM (Simulink)



A Wide Set of Resources



Prodotti Soluzioni Università Assistenza Community Eventi

Act

Help Center

Search Help Center

INDICE

« Documentation Home

« Control Systems

« Motor Control Blockset

« Applications

« Types of Motors

Category

Permanent Magnet Synchronous Motors (PMSM)

Brushless DC (BLDC) Motors

Induction Motors

Switched Reluctance Motors (SRM)

Synchronous Reluctance Motors (SynRM)

Documentation Examples Blocks Videos Answers

Trials

Permanent Magnet Synchronous Motors (PMSM)

Motor control reference examples for PMSM

Utilize the reference examples to implement sensor-based and sensorless motor control algorithms ranging from conventional to advanced techniques for PMSM.

Featured Examples

Open Loop Control of 3-phase motors
Note: This example requires a TI F2800xM Control Card with DRV9312 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

Parameter Estimation Algorithm

Estimate PMSM Parameters Using Recommended Hardware

Determines the parameters of a permanent magnet synchronous motor (PMSM) using the recommended Texas Instruments®

Parameter Estimation Algorithm

Estimate PMSM Parameters Using Custom Hardware

Includes an algorithm to determine the parameters of a permanent magnet synchronous motor (PMSM) using any custom motor-control

PMSM Parameter Estimation
Note: This example requires a TI F2800xM Control Card with a BOOSTx12 C-Blockset pack connected to a PMSM Motor with DSP Service

Estimate PMSM Parameters Using Parameter Estimation Blocks

Uses the parameter estimation blocks provided by Motor Control Blockset™ to estimate these parameters of a permanent magnet

PMSM Parameter Estimation on Real-Time Target
Note: This example requires a TI F2800xM Control Card with a BOOSTx12 C-Blockset pack connected to a PMSM Motor with DSP Service

Estimate PMSM Parameters Using Parameter Estimation Blocks on Real-Time...

Uses the parameter estimation blocks provided by Motor Control Blockset™ to estimate these parameters of a permanent magnet

Estimate PMSM Parameters Using FPGA-Based Motor Control Development Kit

Estimate PMSM Parameters Using FPGA-Based Motor Control Development Kit

Estimate parameters of a permanent magnet synchronous motor (PMSM) using blocks from Motor Control Blockset™ on an FPGA device (Trenz

Permanent Magnet Synchronous Motor Field Oriented Control
Note: This example requires a TI F2800xM Control Card with a BOOSTx12 C-Blockset pack connected to a PMSM Motor with DSP Service

Sensorless Field-Oriented Control of PMSM

Offset Computation with Hall sensor
Note: This example requires a TI F2800xM Control Card with a BOOSTx12 C-Blockset pack connected to a PMSM Motor with DSP Service

Hall Offset Calibration for PMSM

Field-Oriented Control of PMSM Using Hall Sensor

Field-Oriented Control of PMSM Using Hall Sensor

Quadrature Encoder Offset Calibration for PMSM

Quadrature Encoder Offset Calibration for PMSM

Field-Oriented Control of PMSM Using Quadrature Encoder

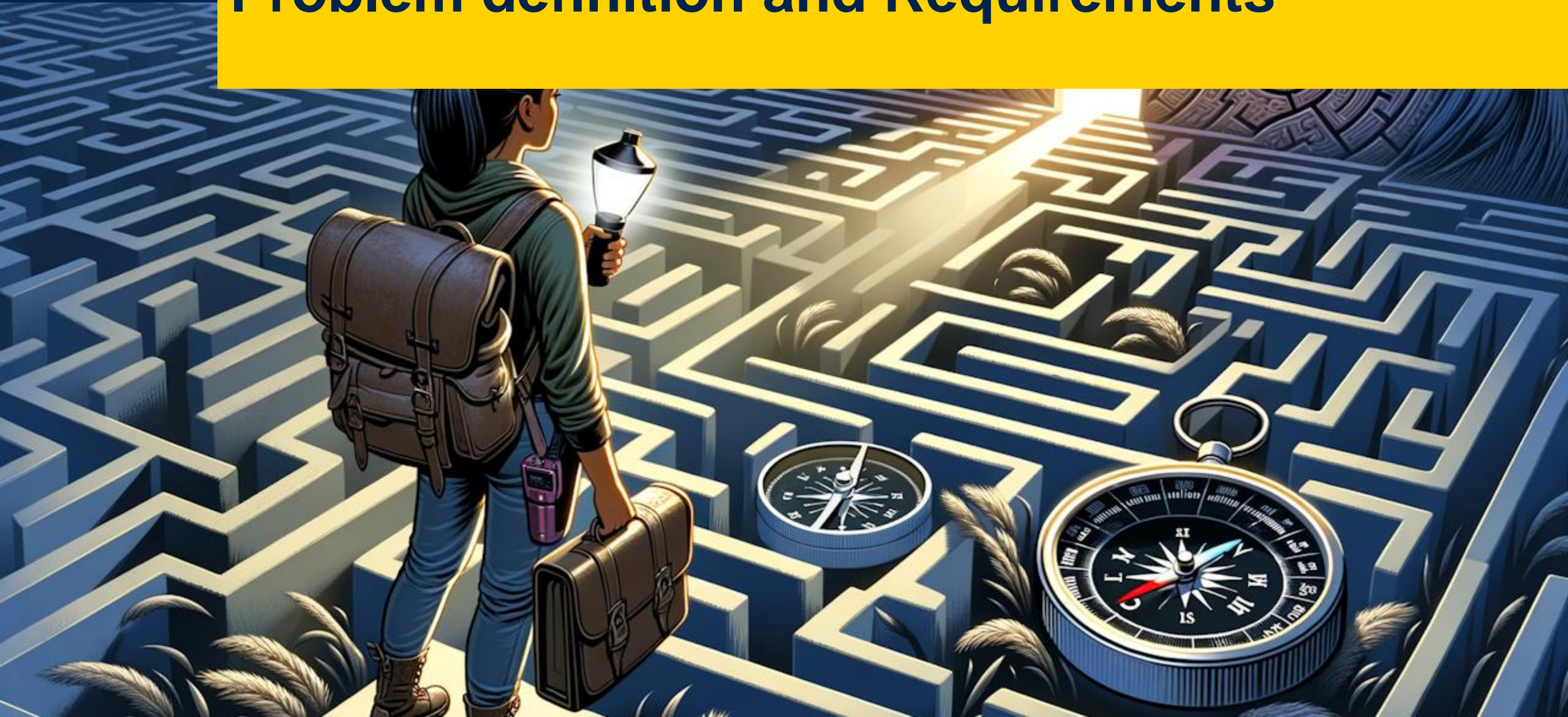
Field-Oriented Control of PMSM Using Quadrature Encoder

Field-Weakening Control (with MTPA) of PMSM

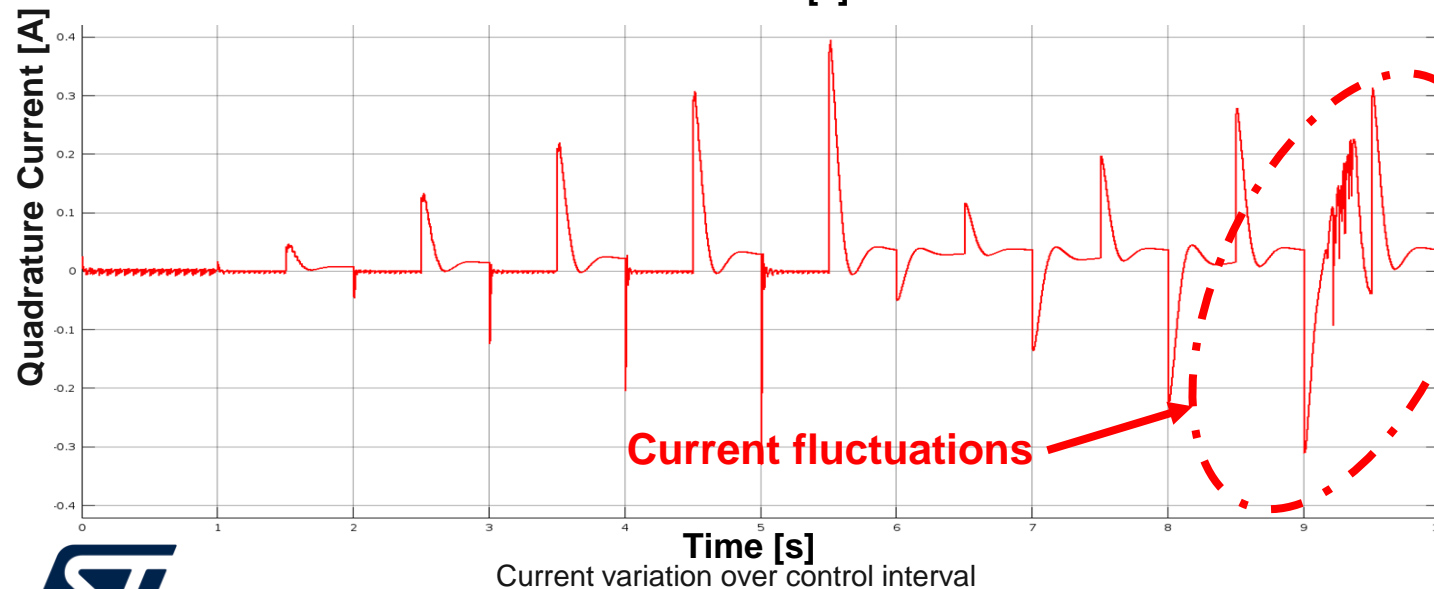
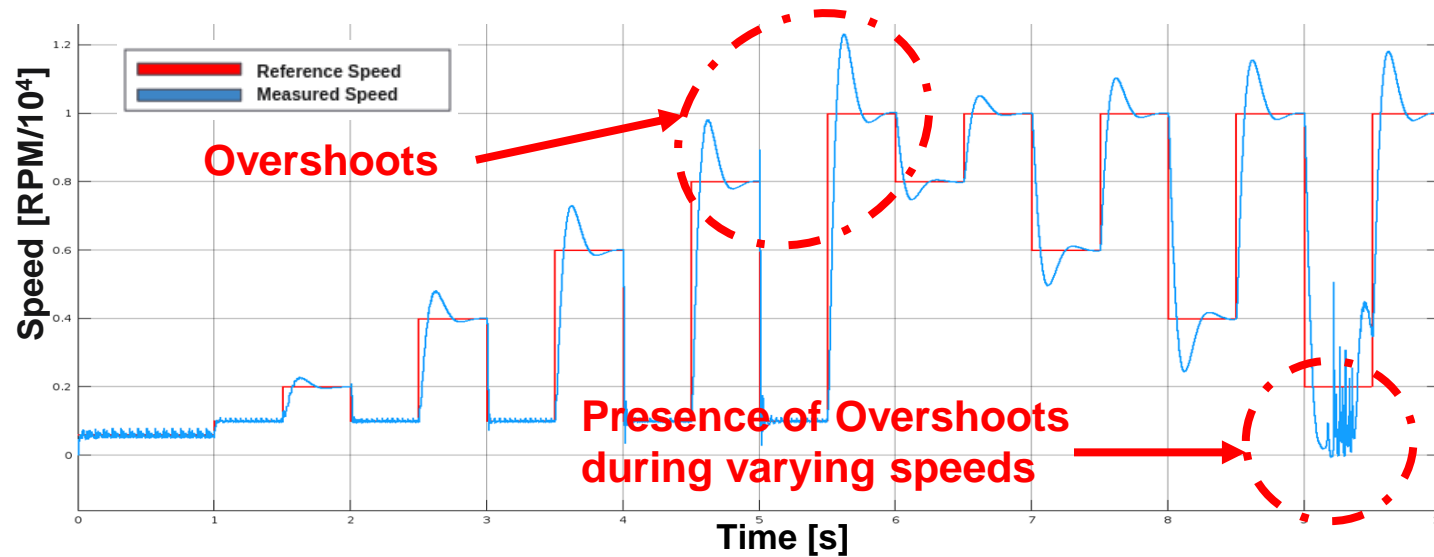
Field-Weakening Control (with MTPA) of PMSM



Problem definition and Requirements

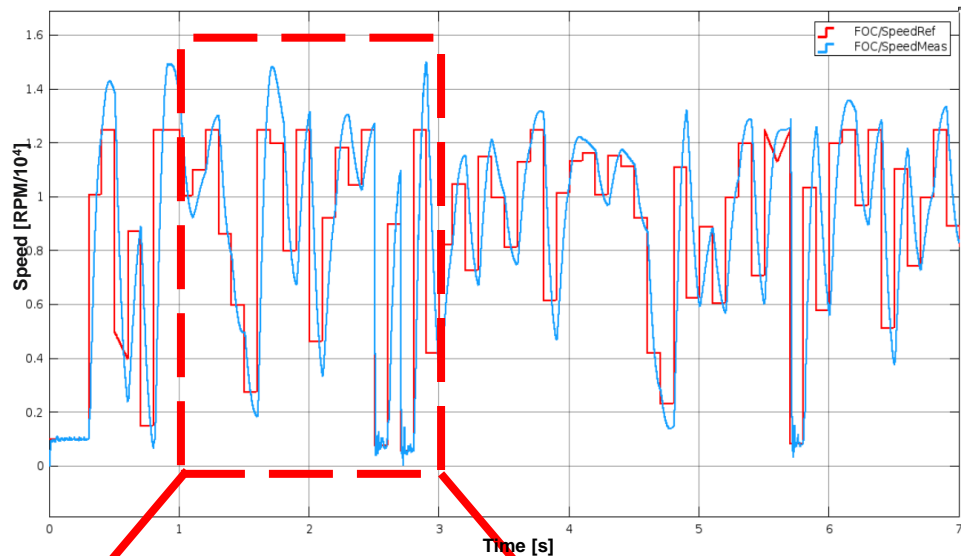


Case Study 1

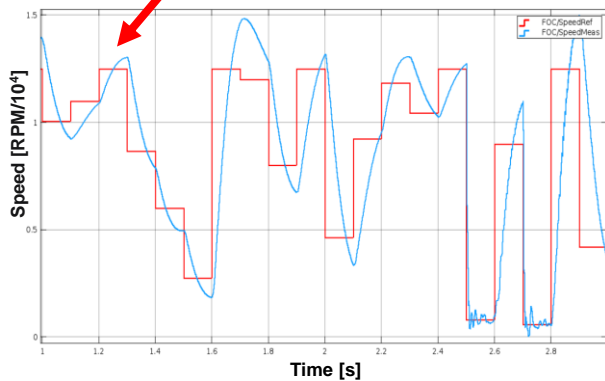


- **Case 1** introduced a speed signal with **2 transitions per second**.
- The **PI(D) controller struggled** to quickly adapt to rapid changes in the reference speed leading to **poor dynamic performances** and **sluggish responses**.
- Moreover, it produces **significant (0.81) deviation** and **longer settling times**, impacting the precision and stability of motor control.

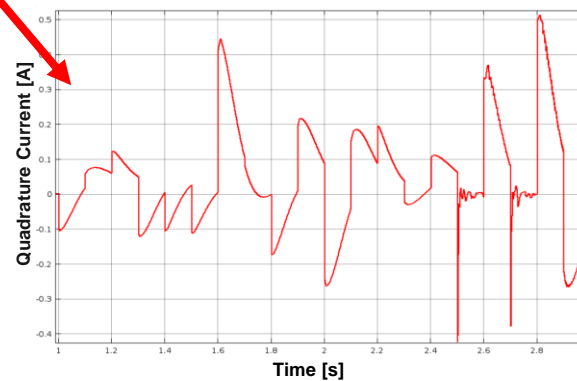
Case Study 2



PI(D) speed control over a 7 seconds interval, with about 10 Hz speed variations



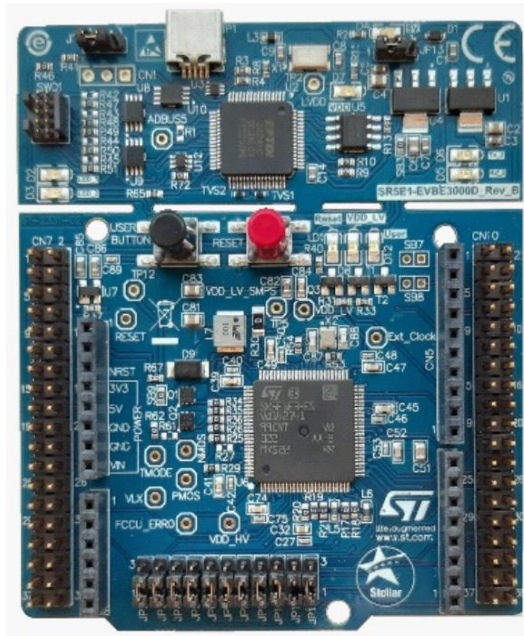
Poor control over each interval



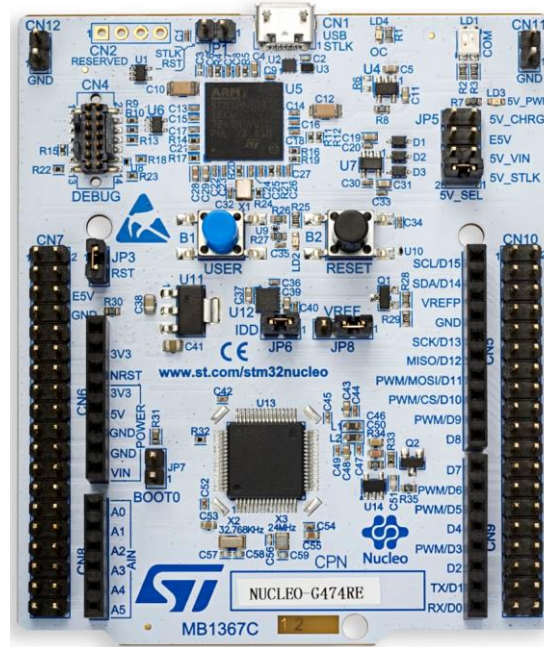
Reference current display over 2 seconds interval

- **Case 2** introduced even more transitions (**10**) in one second.
- At even faster transitions, though the PI(D) controller follows the overall speed trend, it **significantly fails** to stabilize around the desired speed (for each interval).
- This since the calculated reference (quadrature **q**) current generated by the speed PI(D) controller contains deviations (errors) for most of the time steps

Embedded MCU Targets



Board: **SR5R1-EVBE3000D**
Processor Speed: 300 MHz
Internal RAM: 256 KiB
Internal Flash: 1920 KiB



Board: **NUCLEO-G474RE**
Processor Speed: 170 MHz
Internal RAM: 128 KiB
Internal Flash: 512 KiB

- Correcting PI(D) signals requires **extra computations**.
- These approaches shall be deployable on tiny MCUs.
- Two ST MCU boards, automotive (Stellar) and IoT (STM32), have been considered.

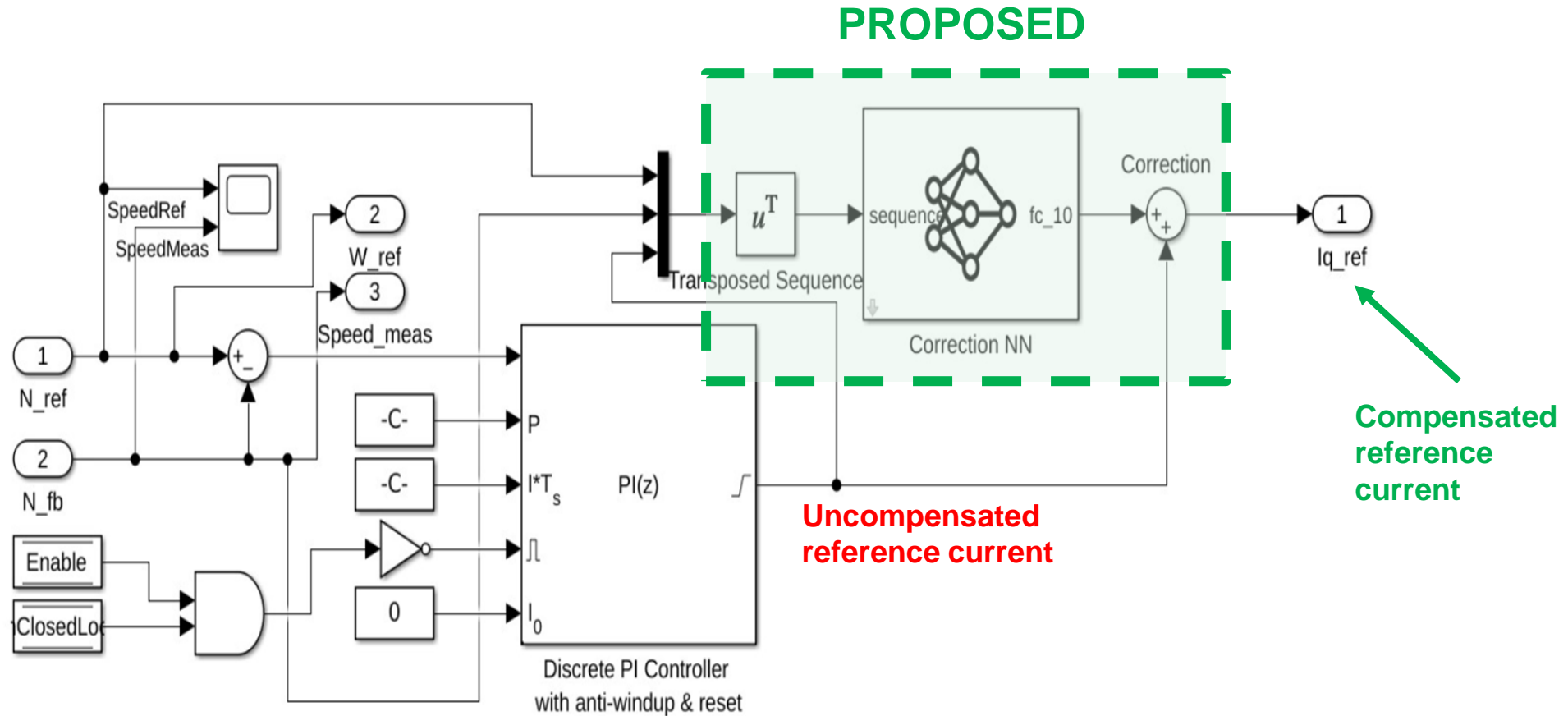
Approach devised

A chalkboard with a dark green surface and a wooden frame. Two large, hand-drawn white arrows are on the board. The top arrow points to the right and contains the text "Old Way". The bottom arrow points to the left and contains the text "New Way".

Old Way

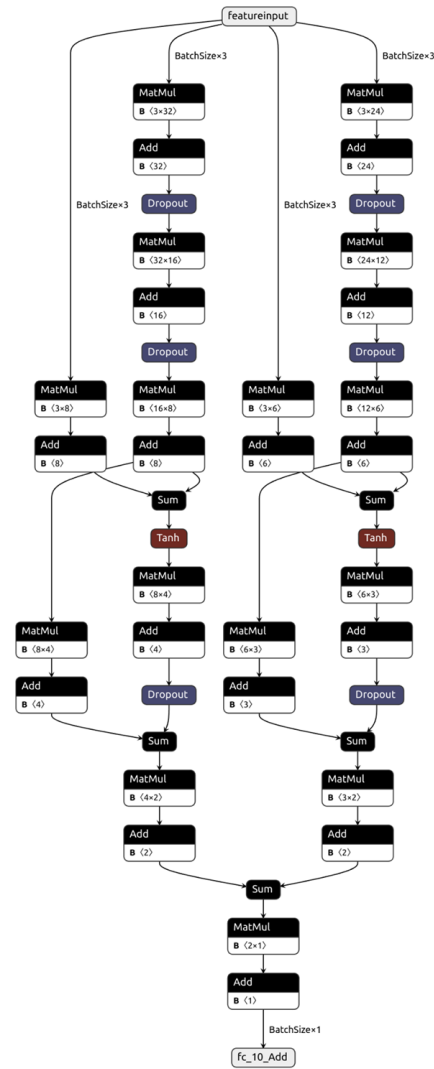
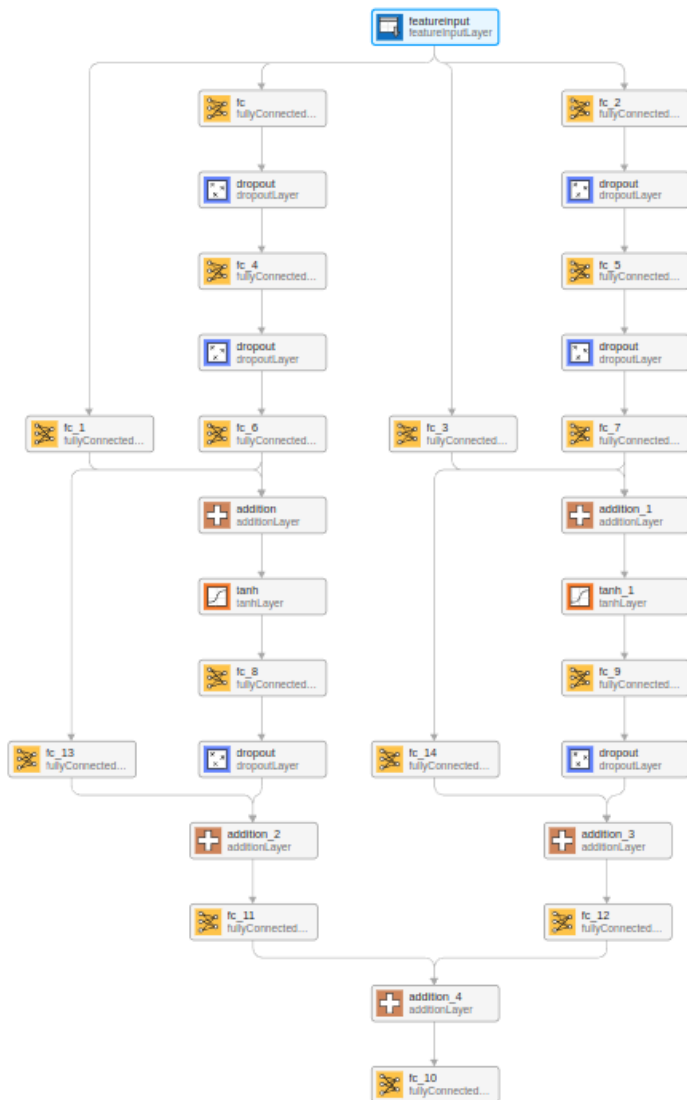
New Way

AI Augmented FOC



Speed control set-up, with TinyNN to predict the PI(D)'s deviations

Network Topology



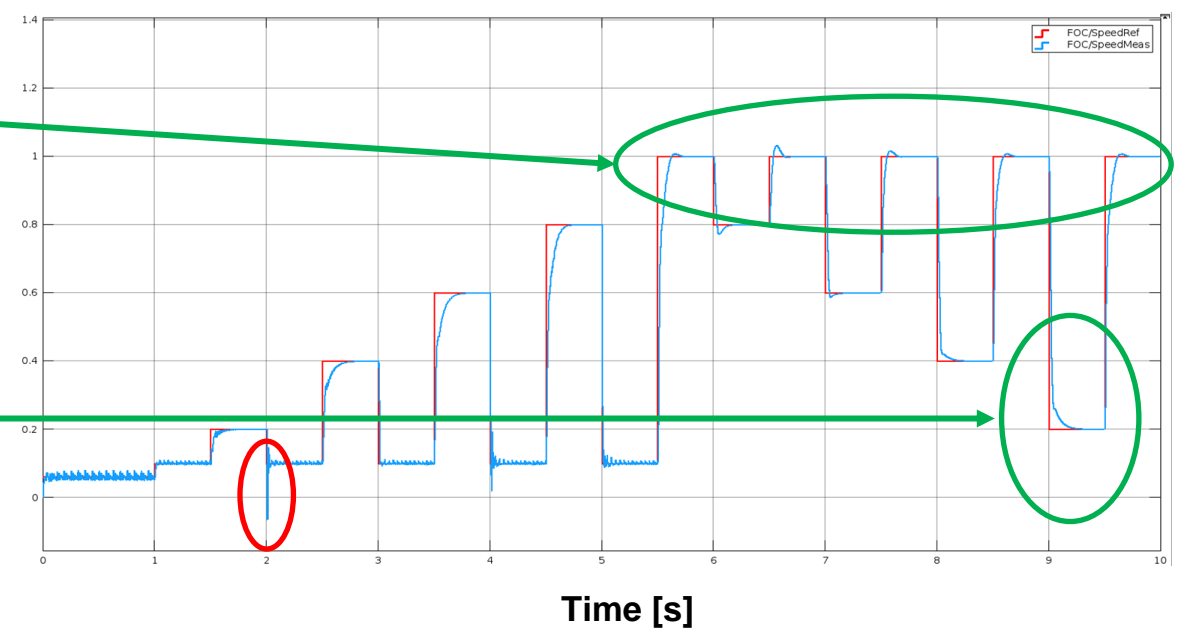
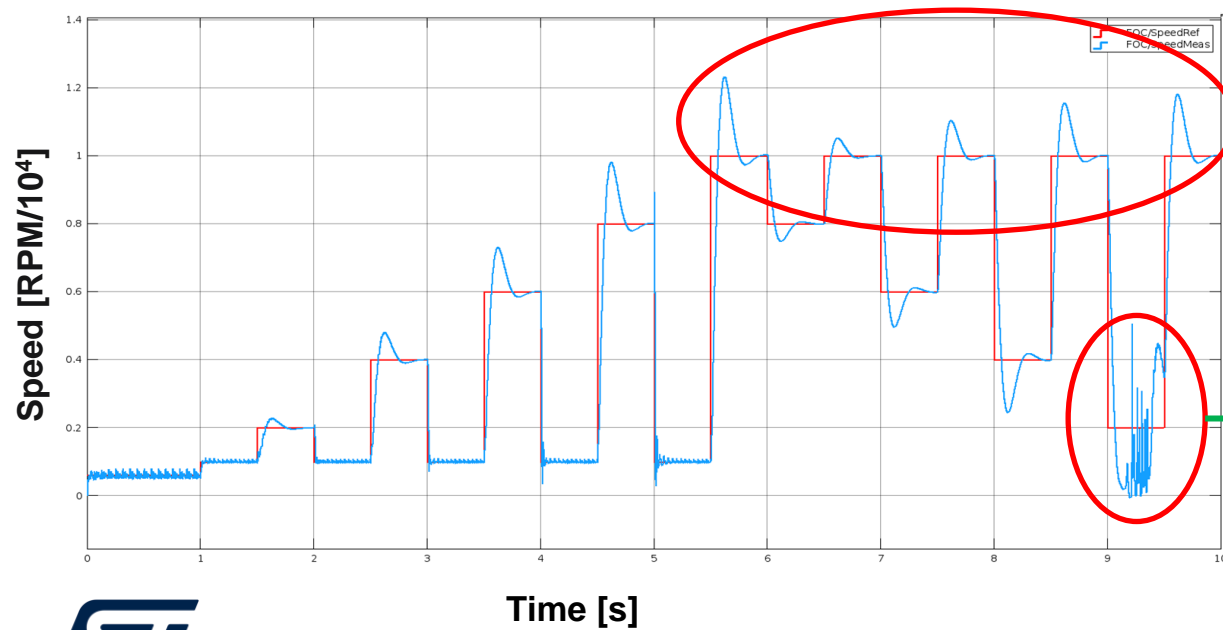
- The proposed model (w/o spatial dilation) was a **1.4 K (weights)** model size, moderately deep with residual connections.
- Ratio training samples and weights = **171.4**

Experimental Results



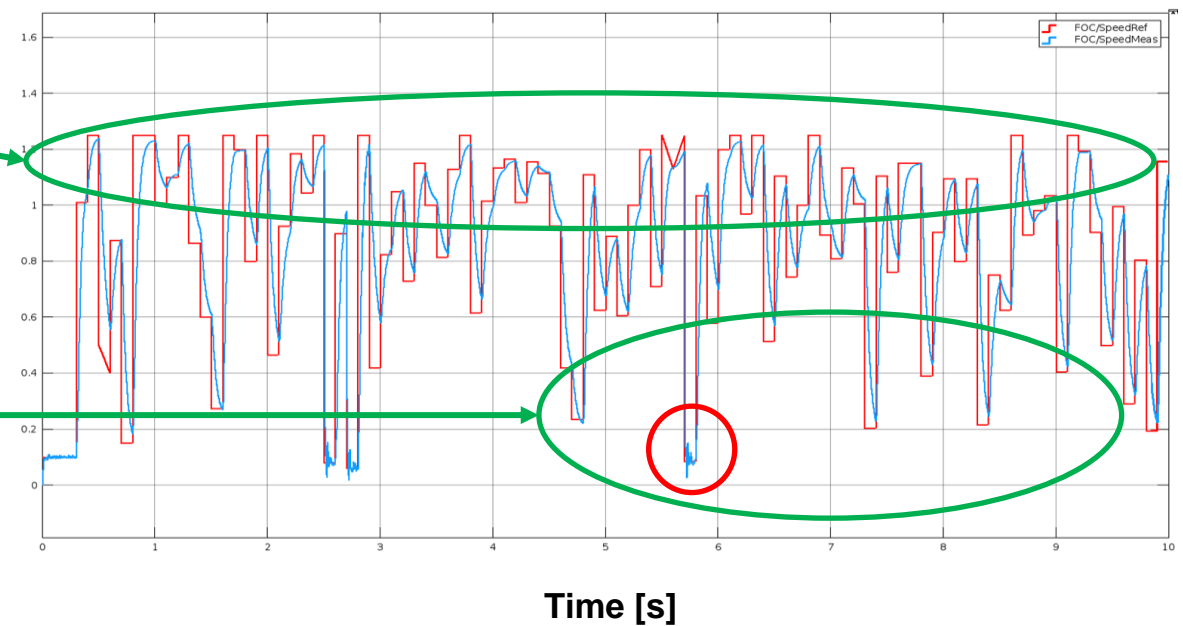
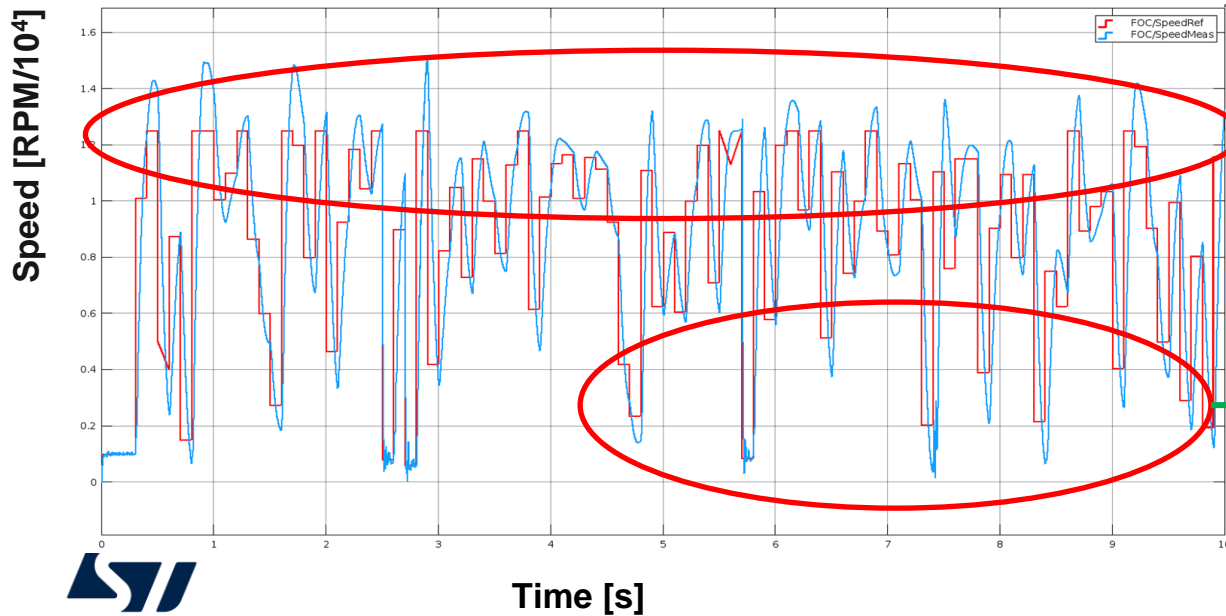
Case Study 1

	Max Deviation	Average Deviation	Max overshoot
PI(D)	0.81	0.05	0.24
PI(D) + TinyNN	0.89	0.02	0.03
Percentage Change (%)	+10	- 60	- 87.5

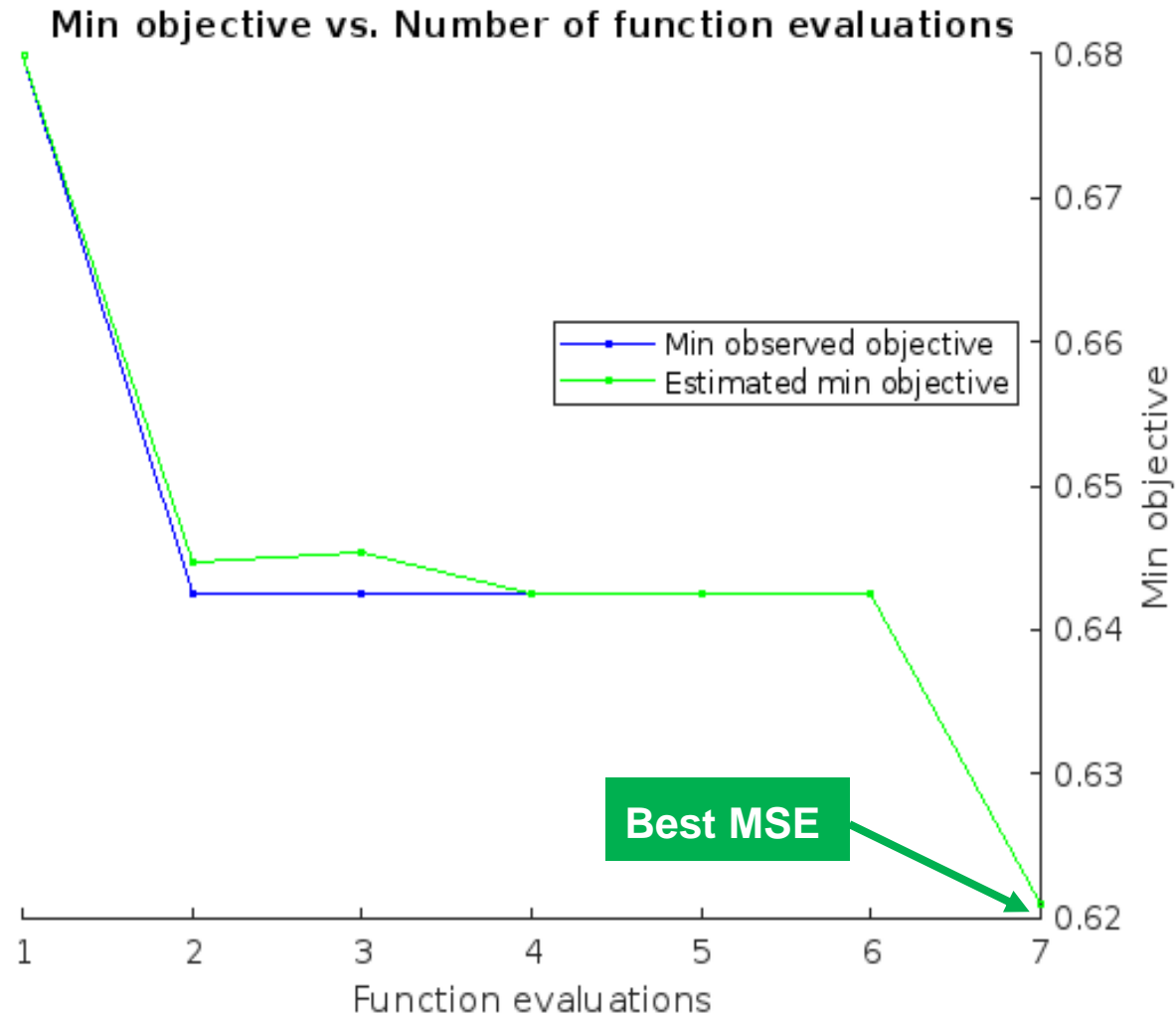
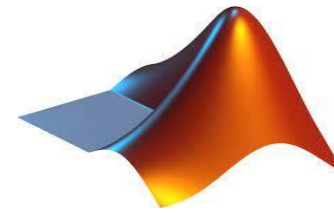


Case Study 2

	Max Deviation	Average Deviation	Max overshoot
PI(D)	1.21	0.18	0.25
PI(D) + TinyNN	1.19	0.15	0.08
Percentage Change (%)	- 1.65	- 16.7	- 68



Hyper Parameters Optimization

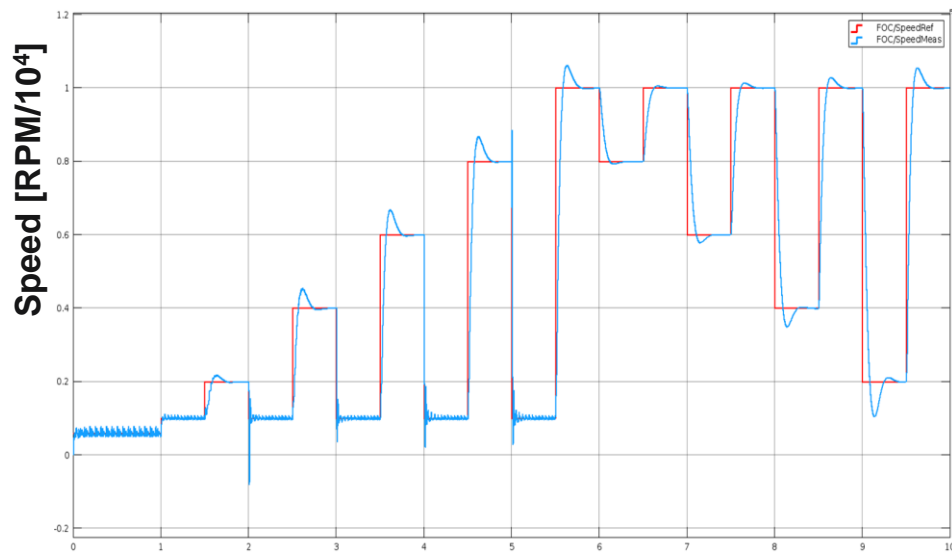


- The objective was to further reduce the number of trainable parameters.
- Obtained the best model iterating through several model configurations.
- Each model was trained over the same number of iterations as the initial model, and the model with least MSE was found.

HPO results

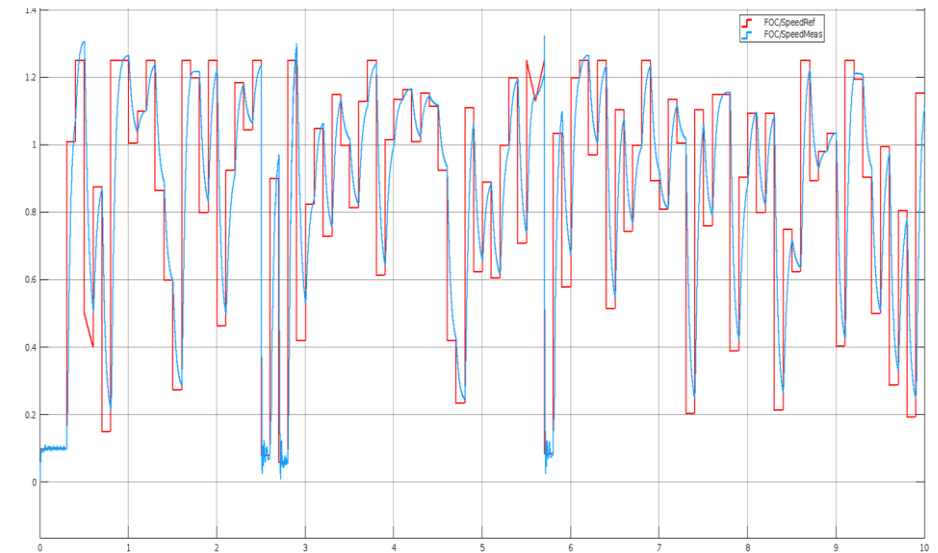
	Trainables	Max Deviation	Average Deviation	Max overshoot
PI(D) + TinyNN	1.4 K	0.89	0.02	0.03
PI(D) + HPO TinyNN	0.67 K	0.896	0.0316	0.06
Percentage Change (%)	-52.1	+0.67	+58	+100

Case 1



	Trainables	Max Deviation	Average deviation	Max overshoot
PI(D) + TinyNN	1.4 K	1.19	0.15	0.08
PI(D) + HPO TinyNN	0.34 K	1.23	0.15	1.24
Percentage Change (%)	-75.7	+3.36	No change	+1450

Case 2



Time [s]

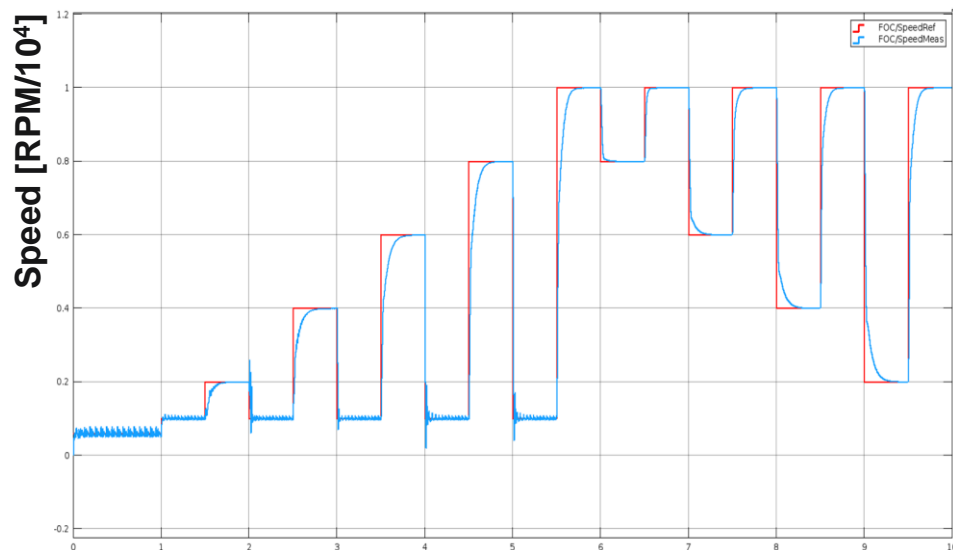
Pruning results

	Trainables	Max Deviation	Average Deviation	Max overshoot
PI(D) + TinyNN	1.4 K	0.89	0.02	0.03
PI(D) + Pruned TinyNN	0.873 K	0.9	0.02	~ 0.0
Percentage Change (%)	-37.64	+ 1.12	No change	-100

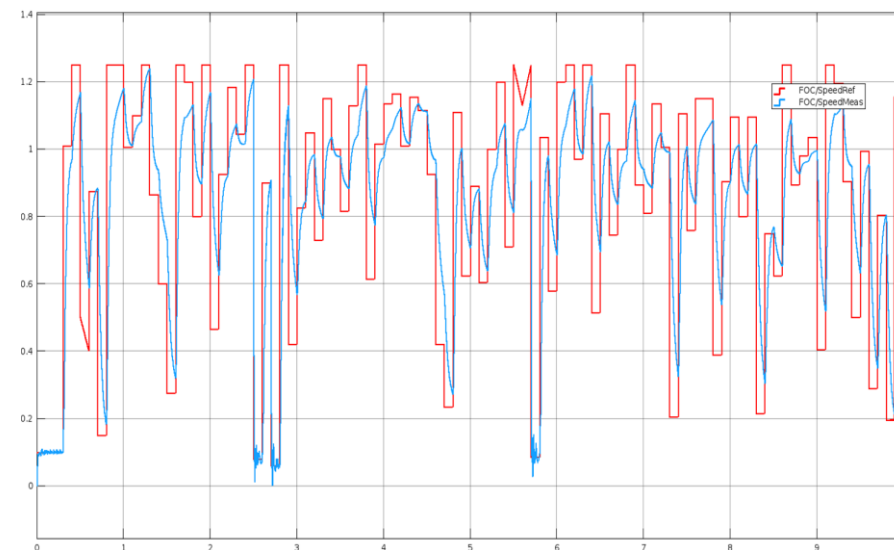
Case 1

	Trainables	Max Deviation	Average deviation	Max overshoot
PI(D) + TinyNN	1.4 K	1.19	0.15	0.08
PI(D) + Pruned TinyNN	0.6 K	1.20	0.16	0.03
Percentage Change (%)	-57.14	+0.84	+6.67	-62.5

Case 2



Time [s]



Deployability on MCU



ST Edge AI Developer Cloud

The screenshot shows the ST Edge AI Developer Cloud interface. At the top, there are tabs for 'INPUT', 'OUTPUT', and 'MODEL TYPE'. Below these, there are five hardware options, each with a 'Select' button and a version indicator:

- STM32 MCUs**: STM32 Microcontroller units. Start with General Purpose STM32 Discovery Kits and Nucleos. Latest (9.0.0).
- STM32 MCU with Neural-AI™**: STM32 Microcontrollers embedding Neural Processing Unit (NPU). Start with STM32 MCUs including Neural-AI™ to accelerate your AI applications. 10.0.0.
- STM32 MPUs**: STM32 Microprocessors Units (MPU). Start with STM32 Microprocessors embedding Cortex-A loaded with X-LINUX-AI. 10.0.0.
- Stellar-E MCUs**: Stellar-E Microcontroller units. Start with Stellar electrification (E) to empower neural network architectures on automotive MCUs. Latest (9.0.0).
- MEMS Sensors with ISPU**: MEMS sensors featuring an embedded intelligent sensor processing unit (ISPU). Start with MEMS Sensors embedding ISPU, an ultralow power, computationally efficient, high-performance programmable core that can execute signal processing and AI algorithms in the edge. Latest (9.0.0).

8bits Post Training Quantization (PTQ)

The screenshot shows the PTQ configuration interface. It includes a 'Powered by Onnx Runtime' badge and an information icon. There are two main sections:

- Disable per channel quantization**: A checkbox that is currently unchecked.
- Load a dataset to check the accuracy obtained after quantization**: A section with a 'Load file (.npz)' button, an information icon, and a note: 'If no quantization file is provided, quantization will occur with random values'. A 'Launch quantization' button is located at the bottom right of this section.

Below these sections is a 'Quantized models' list:

- netProjCase2_PerChannel_quant_random_1.onnx**: Content Length: 48.92 KiB, Last Modified: 9/1/24, 2:47 PM. It has a download icon, a trash icon, and a 'Select' button.

<https://stm32ai-cs.st.com/home>

Case 1

	Number of Parameters	MACC	FLASH (KiB)	RAM (KiB)	Execution Time (us)
Fp32 Tiny NN	1400	1620	Weights: 5.68 Library STM32: 15 Library Stellar-E: 16	Activations: 0.199 Library: 6	NUCLEO-G474RE 207.4 SR5E1-EVBE3000D 92.8
HPO Model	670	764	Weights: 2.61 Library STM32: 15 Library Stellar-E: 16	Activations: 0.152 Library: 6	NUCLEO-G474RE 144.8 SR5E1-EVBE3000D 69.4
Pruned Model	873	1034	Weights: 3.28 Library STM32: 20 Library Stellar-E: 20	Activations: 0.148 Library: 10	NUCLEO-G474RE 232.2 SR5E1-EVBE3000D 102.4
8 bits Quantized Pruned NN on ST Edge AI Dev Cloud	873	910	Weights: 1.37 Library STM32: 32 Library Stellar-E: 30	Activations: 0.383 Library: 13	NUCLEO-G474RE 371.7 SR5E1-EVBE3000D 167.9



Case 2

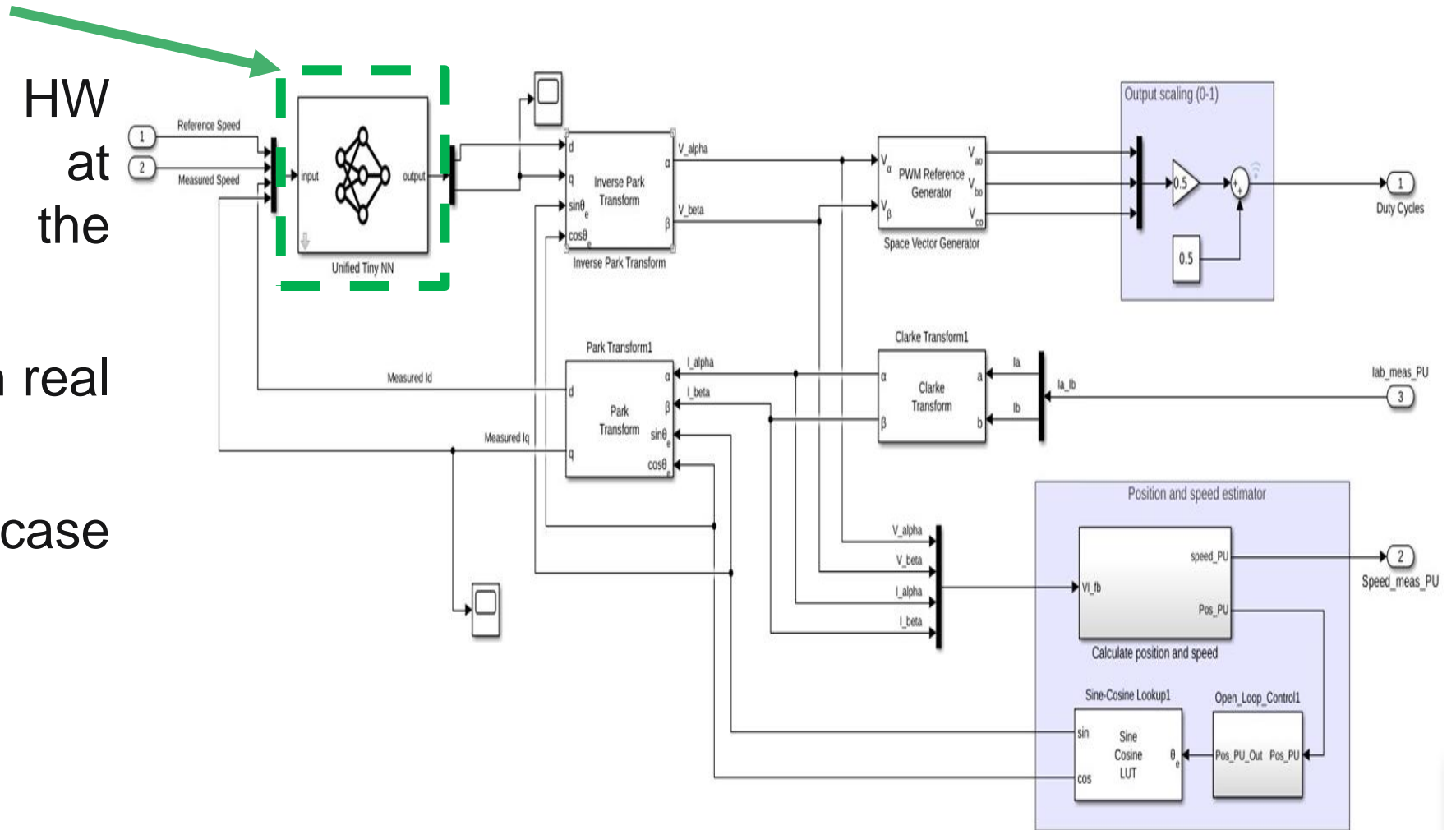
	Number of Parameters	MACC	FLASH (KiB)	RAM (KiB)	Execution Time (us)
Fp32 Tiny NN	1400	1620	Weights: 5.68 Library STM32: 15 Library Stellar-E: 16	Activations: 0.199 Library: 6	NUCLEO-G474RE 207.4 SR5E1-EVBE3000D 92.8
HPO Model	340	470	Weights: 1.33 Library STM32: 15 Library Stellar-E: 16	Activations: 0.105 Library: 6	NUCLEO-G474RE 127.6 SR5E1-EVBE3000D 61.2
Pruned Model	600	760	Weights: 2.26 Library STM32: 20 Library Stellar-E: 20	Activations: 0.145 Library: 10	NUCLEO-G474RE 215.2 SR5E1-EVBE3000D 96.20
8 bits Quantized Pruned NN on ST Edge AI Dev Cloud	600	640	Weights: 1.07 Library STM32: 32 Library Stellar-E: 32	Activations: 0.360 Library: 13	NUCLEO-G474RE 361.4 SR5E1-EVBE3000D 112.1

Future Work



Future Works

- Extend AI approach.
- Study TinyNN acceleration. E.g. 100KHz to close control loop in $10\mu\text{s}$.
- Tests on the field with real PMSM motors.
- Introduce new case studies.
- Explore quantization.



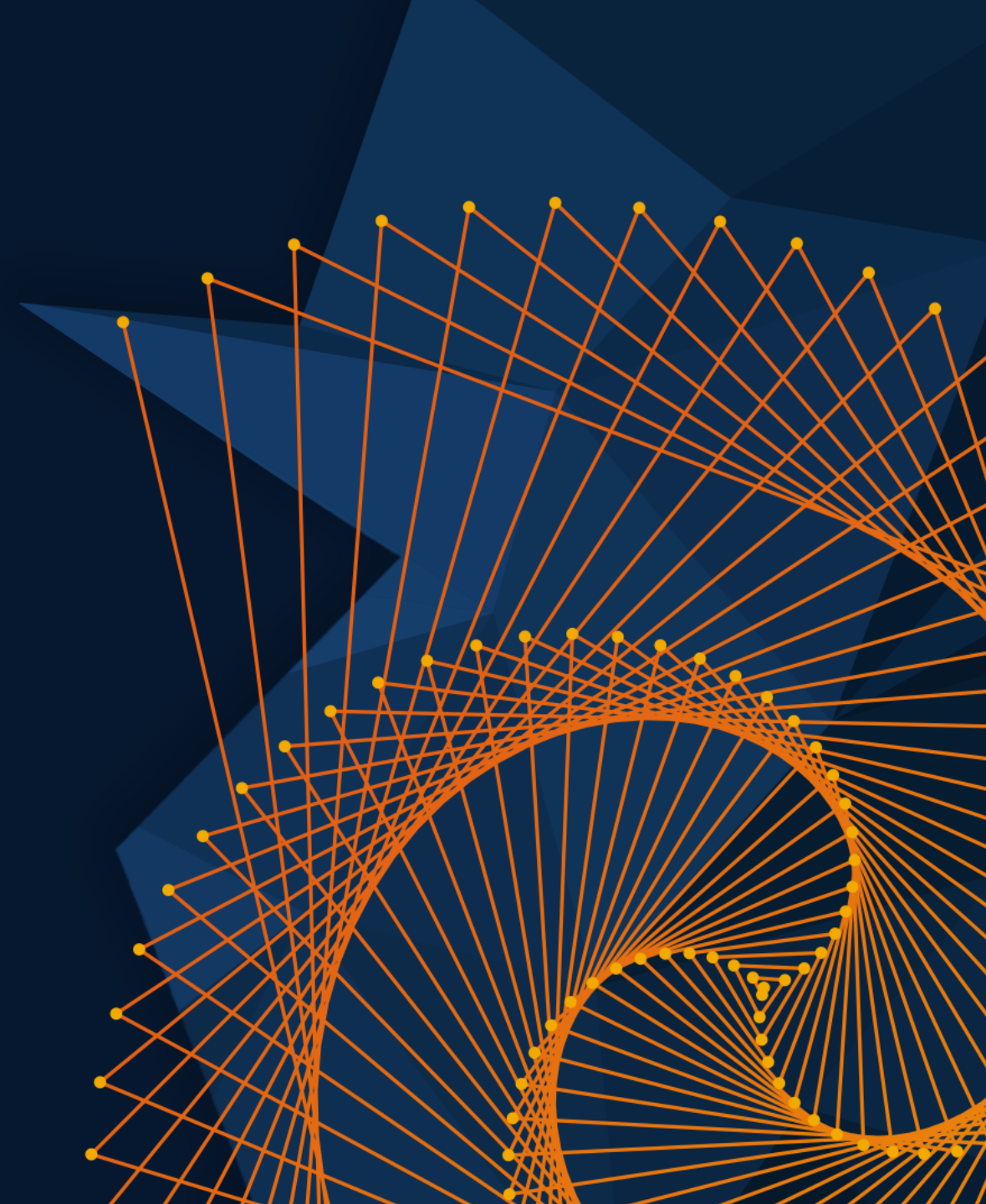
MATLAB EXPO

Thank you

danilo.pau@st.com



© 2024 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [mathworks.com/trademarks](https://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.





MathWorks ✓

@MathWorks

Share the EXPO experience

#MATLABEXPO

