Hochschule
München
University of
Applied Sciences

Faculty of Electrical
Engineering and Information
Technology (FK04)

Institute for Sustainable
Energy Systems (ISES)

Artificial Intelligence for electrical drive systems

Compact Power Motion GmbH
Feringastraße 11a, D-85774 Unterföhring

Prof. Dr.-Ing. habil. Christoph M. Hackl
Niklas Monzen, M.Sc.

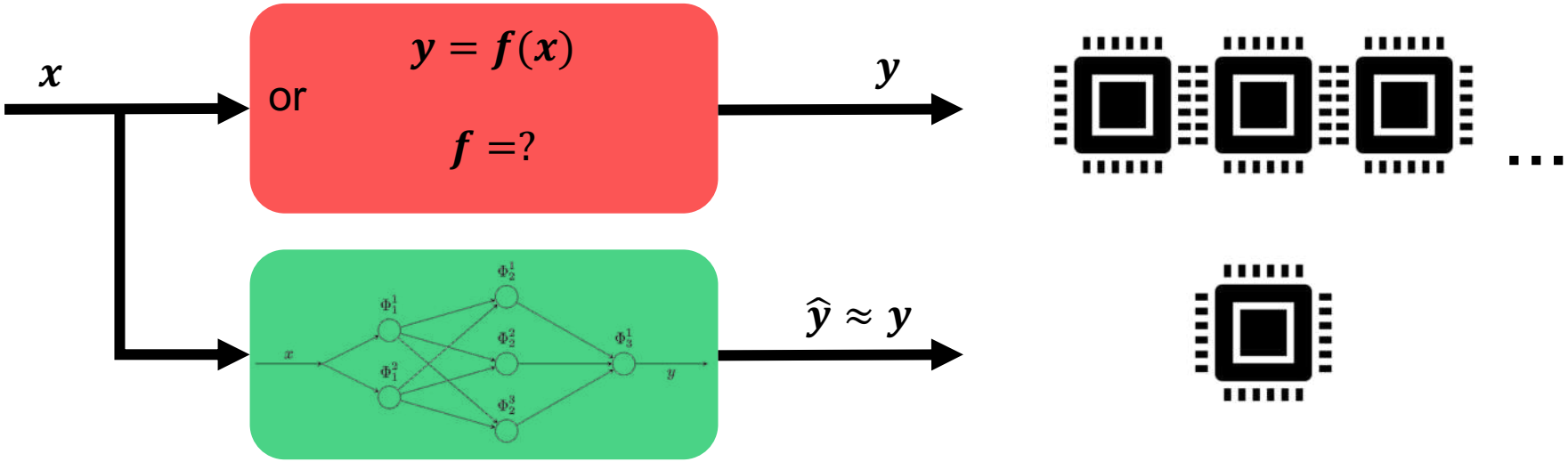
12.+13.06.2023

Agenda

- Motivation & applications
- Framework for design of ANN-based approches for electrical drive systems
- Hands-on example 1: System identification
- Hands-on example 2: ANN-based Maximum Torque per Losses (MTPL)
- Hands-on example 3: ANN-based Optimal Feedforward Torque Control (OFTC) of an IPMSM

Motivation
Reasons for Artificial Intelligence for electrical drive systems

- Problem:
  – Machine's nonlinearities lead to iterative and computationally intensive algorithms



- Idea:
  – Use ANNs as approximation (interpolation and extrapolation)
  – Fixed and feasible execution time
  – Accaptable storage requirement

Motivation

ANN-based OFTC for IPMSM

energies

MDPI

Article

Artificial Neural Network Based Optimal Feedforward Torque Control of Interior Permanent Magnet Synchronous Machines: A Feasibility Study and Comparison with the State-of-the-Art

Max A. Buettner †, Niklas Monzen † and Christoph M. Hackl *

Department of Electrical Engineering and Information Technology, Hochschule München (HM) University of Applied Sciences, Lothstr. 64, 80335 München, Germany; max.buettner@hm.edu (M.A.B.); niklas.monzen@hm.edu (N.M.)

* Correspondence: christoph.hackl@hm.edu

† These authors contributed equally to this work.

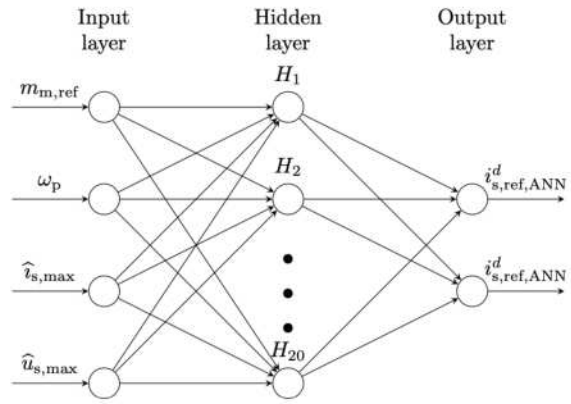
Abstract: A novel Artificial Neural Network (ANN) Based Optimal Feedforward Torque Control (OFTC) strategy is proposed which, after proper ANN design, training and validation, allows to analytically compute the optimal reference currents (minimizing copper and iron losses) for Interior Permanent Magnet Synchronous Machines (IPMSMs) with highly operating point dependent nonlinear electric and magnetic characteristics. In contrast to conventional OFTC, which either utilizes large look-up tables (LUTs; with more than three input parameters) or computes the optimal reference currents numerically or analytically but iteratively (due to the necessary online linearization), the proposed ANN-based OFTC strategy does not require iterations nor a decision tree to find the optimal operation strategy such as e.g., Maximum Torque per Losses (MTPL), Maximum Current (MC) or Field Weakening (FW). Therefore, it is (much) faster and easier to implement while (i) still machine nonlinearities and nonidealities such as e.g., magnetic cross-coupling and saturation and speed-dependent iron losses can be considered and (ii) very accurate optimal reference currents are obtained. Comprehensive simulation results for a real and highly nonlinear IPMSM clearly show these benefits of the proposed ANN-based OFTC approach compared to conventional OFTC strategies using LUT-based, numerical or analytical computation of the reference currents.

check for updates

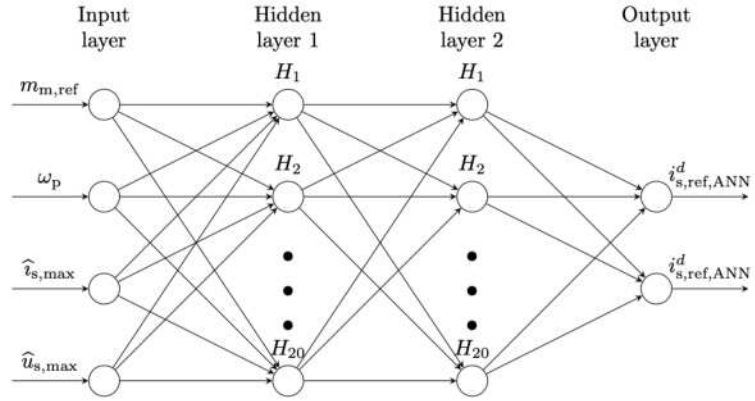
Citation: Buettner, M.A.; Monzen, N.; Hackl, C.M. Artificial Neural Network Based Optimal Feedforward Torque Control of Interior Permanent Magnet Synchronous Machines: A Feasibility Study and Comparison with the State-of-the-Art. *Energies* **2022**, *15*, 1838. https://doi.org/10.3390/en15051838

Academic Editors: Nicola Bianchi, Ludovico Ortombina and K.T. Chau

Keywords: electrical drive control system; operation management; optimal feedforward torque control; optimal reference current computation; transformer-like nonlinear machine model; artificial neural network; synchronous motor; interior permanent magnet synchronous machine; machine learning



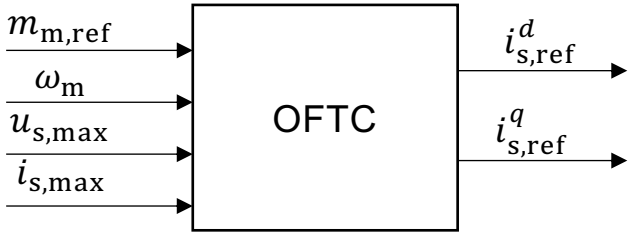
(a) Architecture (A_1).



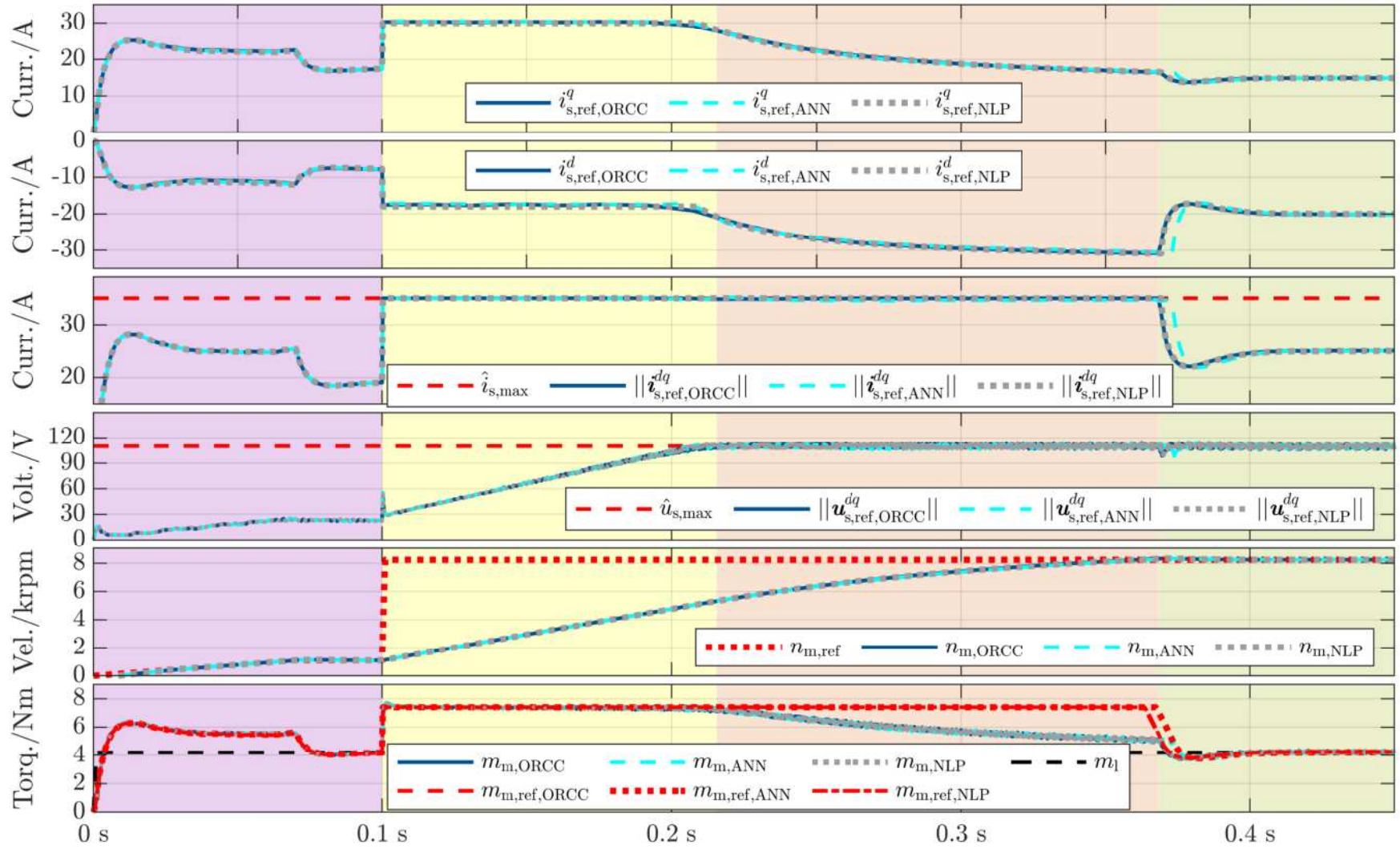
(b) Architecture (A_2).

Figure 9. Illustration of the two implemented ANN architectures: Each architecture has four in layer neurons and two output layer neurons; Architecture (A_1) has one hidden layer, whe Architecture (A_2) has two hidden layers with 20 hidden layer neurons each.

Motivation
ANN-based OFTC for IPMSM



- ✅ OFTC for IPMSM feasible, considering
  - Nonlinear flux linkages
  - Copper losses and nonlinear iron losses
  - Voltage and current limits
- ! Only a few and simple ANN architectures trained
- → Execution time reduced from $440\ \mu s$ (MTPX) to $6\ \mu s$



Performance measure	X=OFTC$_{\mathrm{LUT}}$	X=OFTC$_{\mathrm{NUM}}$	X=OFTC$_{\mathrm{ANA}}$	X=OFTC$_{\mathrm{ANN}}$
$\bar{t}_{\mathrm{exec,X}}$	2 734.782 µs	448.745 µs	439.671 µs	5.855 µs

Motivation
ANN-based OFTC for EESM

Artificial neural network based optimal feedforward torque control of electrically excited synchronous machines

1st Niklas Monzen and 2nd Christoph M. Hackl
Laboratory for Mechatronic and Renewable Energy Systems (LMRES)
HM Munich University of Applied Sciences
Munich, Germany
{niklas.monzen, christoph.hackl}@hm.edu

***Abstract*—An Artificial Neural Network (ANN) based Optimal Feedforward Torque Control (OFTC) strategy for electrically excited synchronous machines (EESMs) is proposed. After design, data set creation, training and validation of the ANN, the analytical computation of the optimal stator and exciter currents is achieved which allows to minimize copper and iron losses and to produce the desired (or maximally feasible) machine torque. Voltage and currents constraints of stator and exciter are considered as well. In contrast to conventional OFTC, the proposed ANN-based OFTC strategy does not require iterations nor a decision tree to find the optimal current triple while machine nonlinearities, magnetic cross-coupling, saturation and speed-dependent iron losses are taken into account. In addition, the proposed ANN design procedure allows to consider *measurable* OFTC goals and computational resources that ensures a real-time capable implementation. Comprehensive simulation results for a real and nonlinear EESM clearly show these benefits by comparing the proposed ANN-based OFTC with results of a nonlinear optimization problem (NLP) solver (which cannot be used in real-time).**

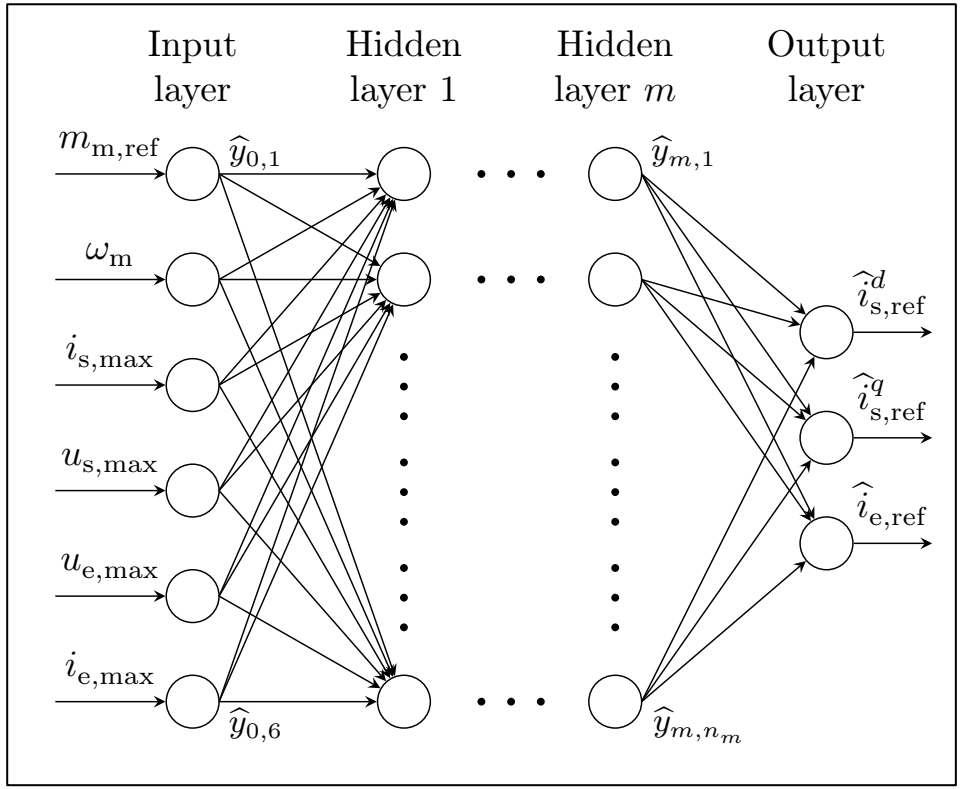
***Index Terms*—Artificial neural network (ANN), copper & iron losses, efficiency, electrically excited synchronous machine (EESM), machine learning, nonlinear machine model, optimal feedforward torque control (OFTC), optimal reference current computation, optimization.**

a unified theory was presented that covers different operating strategies such as Maximum-Torque-per-Ampere (MTPA; or Maximum-Torque-per-Current (MTPC)), Field Weakening (FW), Maximum Current (MC) and Maximum-Torque-per-Voltage (MTPV) and allows to compute the optimal stator reference currents for RSM and PMSM analytically without simplifying assumptions on flux linkage nonlinearities or stator resistances. However, iron losses were neglected. This has been overcome by the extended analytical approach presented in [17, 24]. But EESMs, with additional exciter current, are not covered.

In [28–30] numerical identification and search algorithms are proposed as *offline* OFTC approaches for EESM that are not suitable for a real-time implementation.

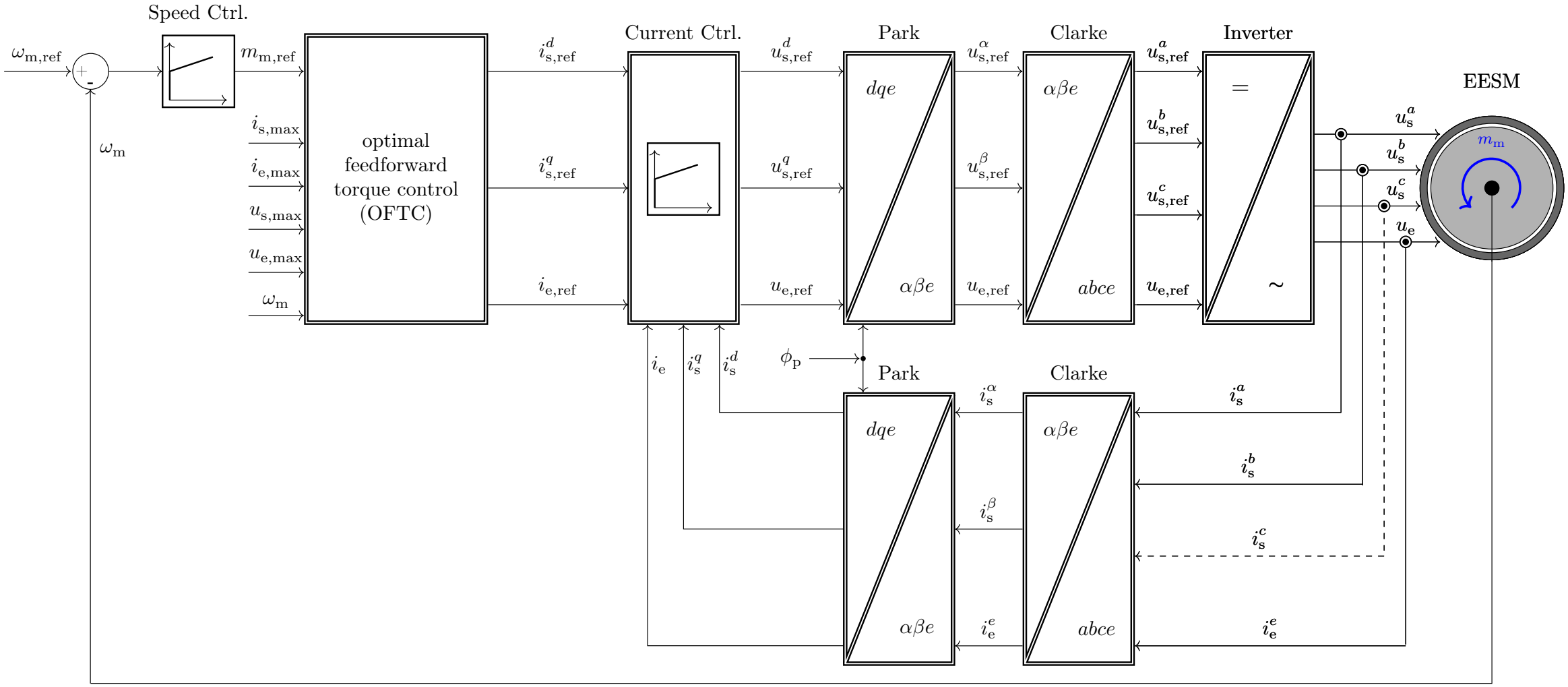
In [31], the MTPA problem for stator and exciter currents is solved by the use of a Lagrangian. However, electromagnetic cross-coupling and iron losses are not considered. Furthermore, the reference currents are computed offline and stored in look-up tables (LUTs) due to the computationally expensive solution of the MTPA problem.

In [32–36] online OFTC approaches are presented that are

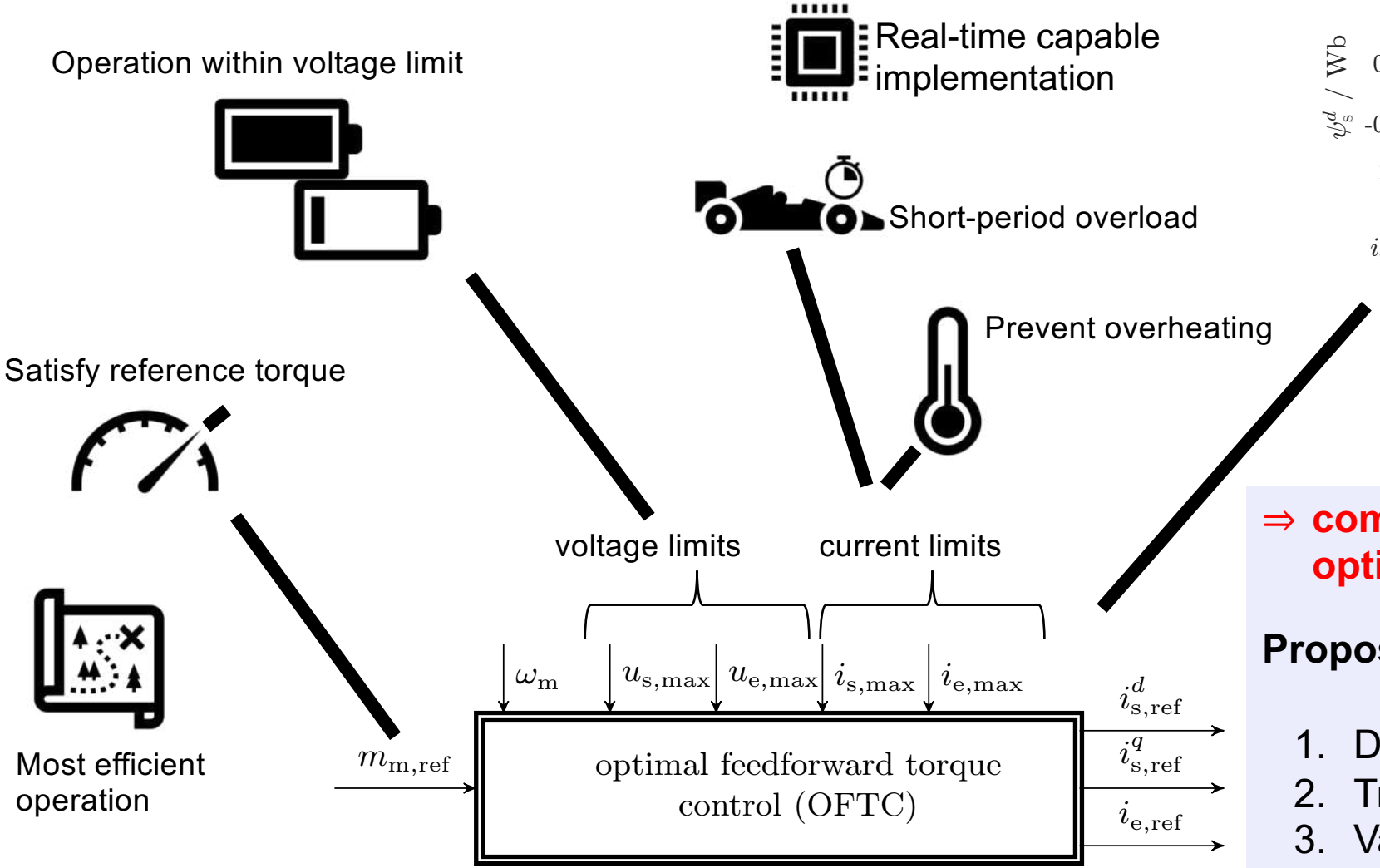


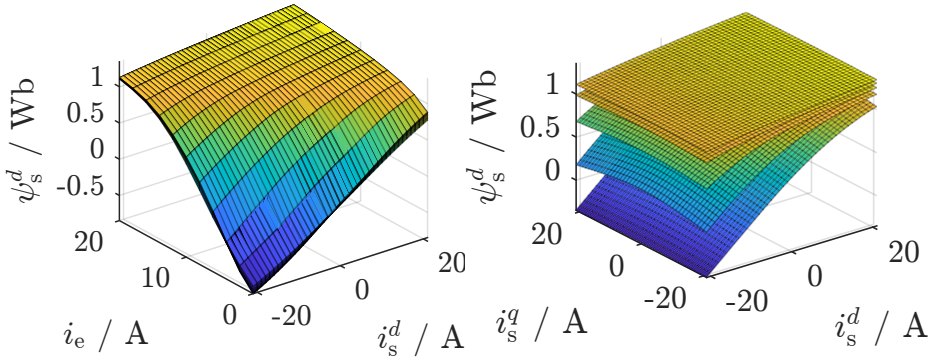
Motivation

Control of an electrically-excited synchronous machine (EESM) with optimal feedforward torque control (OFTC)



Motivation
ANN-based OFTC for EESM





Nonlinear flux linkages, iron and copper losses

⇒ **computationally intensive nonlinear optimization problem (NLP)**

Proposed Solution:
ANN-based OFTC for EESM

1. Dataset creation
2. Training of suitable ANN-architectures
3. Validation and choice of an optimal ANN

ANN-based OFTC of EESM
Dataset creation

For the k-th input sample

$$\boldsymbol{x}[k] \coloneqq \left(m_{\mathrm{m,ref}}[k], \omega_{\mathrm{m}}[k], u_{\mathrm{s,max}}[k], i_{\mathrm{s,max}}[k], u_{\mathrm{e,max}}[k], i_{\mathrm{e,max}}[k]\right)^{\mathrm{T}}$$

the corresponding output vector

$$\boldsymbol{y}[k] \coloneqq \left(i^{d}_{\mathrm{s,ref}}[k], i^{q}_{\mathrm{s,ref}}[k], i_{\mathrm{e,ref}}[k]\right)^{\mathrm{T}}$$

is the solution of the nonlinear optimization problems (NLP)

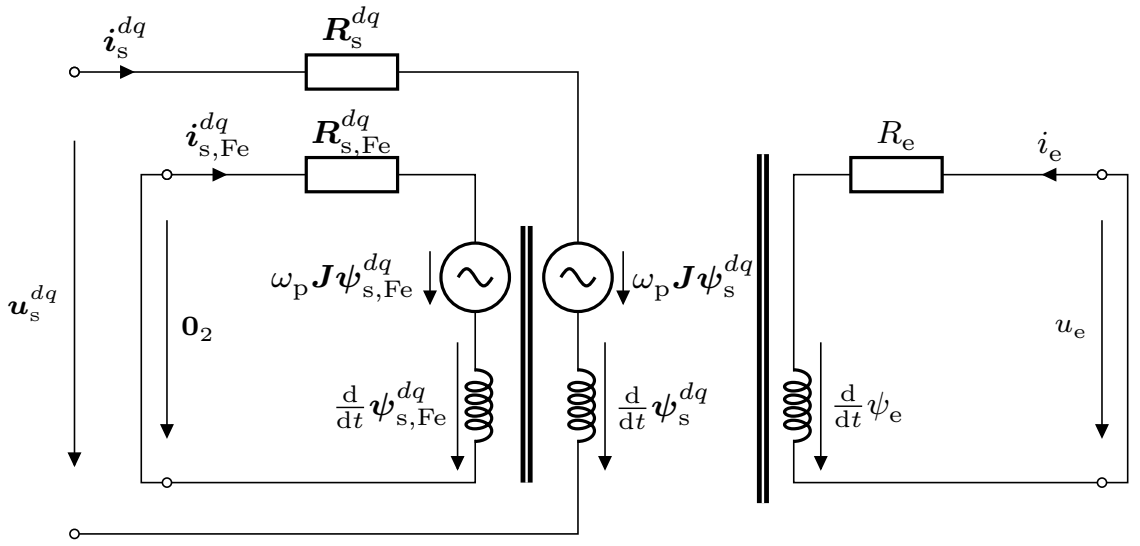
$$\boldsymbol{i}^{dqe}_{\mathrm{s,e,ref}}[k] = \underset{\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]}{\arg\min}\ p_{\mathrm{L}}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)$$

$$\begin{aligned}\text{s.t. } \left\|\boldsymbol{u}^{dq}_{s}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right\| &\le u_{\mathrm{s,max}}[k],\\ \left\|\boldsymbol{i}^{dq}_{s}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right\| &\le i_{\mathrm{s,max}}[k],\\ \left|u_{\mathrm{e}}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right| &\le u_{\mathrm{e,max}}[k],\\ |i_{\mathrm{e}}[k]| &\le i_{\mathrm{e,max}}[k],\\ m_{\mathrm{m}}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right) &= \underset{m_{\mathrm{m,max}}}{\mathrm{sat}}\ m_{\mathrm{m,ref}}[k]\end{aligned}$$

$$m_{\mathrm{m,max}}[k] = \max_{\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]}\ m_{\mathrm{m}}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)$$

$$\begin{aligned}\text{s.t. } \left\|\boldsymbol{u}^{dq}_{s}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right\| &\le u_{\mathrm{s,max}}[k],\\ \left\|\boldsymbol{i}^{dq}_{s}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right\| &\le i_{\mathrm{s,max}}[k],\\ \left|u_{\mathrm{e}}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right| &\le u_{\mathrm{e,max}}[k],\\ |i_{\mathrm{e}}[k]| &\le i_{\mathrm{e,max}}[k].\end{aligned}$$

→NLP can not be solved online in real-time!
→But offline to create a dataset with $k \in \{1, \ldots, K\}$ samples.



$$\boldsymbol{u}^{dq}_{\mathrm{s}} = \boldsymbol{R}^{dq}_{\mathrm{s}}\boldsymbol{i}^{dq}_{\mathrm{s}} + \boldsymbol{\omega}_{\mathrm{p}}\boldsymbol{J}\boldsymbol{\psi}^{dq}_{\mathrm{s}} + \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}^{dq}_{\mathrm{s}}$$

$$u_{\mathrm{e}} = R_{\mathrm{e}}i_{\mathrm{e}} + \frac{\mathrm{d}}{\mathrm{d}t}\psi_{\mathrm{e}}$$

$$\mathbf{0}_2 = \boldsymbol{R}^{dq}_{\mathrm{s,Fe}}\boldsymbol{i}^{dq}_{\mathrm{s,Fe}} + \boldsymbol{\omega}_{\mathrm{p}}\boldsymbol{J}\boldsymbol{\psi}^{dq}_{\mathrm{s}} + \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}^{dq}_{\mathrm{s}}$$

$$p_{\mathrm{L}} = \underbrace{\frac{2}{3\kappa^2}\left(\boldsymbol{i}^{dq}_{\mathrm{s}}\right)^{\mathrm{T}}\boldsymbol{R}^{dq}_{\mathrm{s}}\boldsymbol{i}^{dq}_{\mathrm{s}} + R_{\mathrm{e}}i^2_{\mathrm{e}}}_{\text{copper losses}} + \underbrace{\frac{2}{3\kappa^2}\left(\boldsymbol{i}^{dq}_{\mathrm{s,Fe}}\right)^{\mathrm{T}}\boldsymbol{R}^{dq}_{\mathrm{s,Fe}}\boldsymbol{i}^{dq}_{\mathrm{s,Fe}}}_{\text{iron losses}}$$

$$m_{\mathrm{m}} = \frac{2n_{\mathrm{p}}}{3\kappa^2}\left(\boldsymbol{i}^{dq}_{\mathrm{s}} + \boldsymbol{i}^{dq}_{\mathrm{s,Fe}}\right)^{\mathrm{T}}\boldsymbol{J}\boldsymbol{\psi}^{dq}_{\mathrm{s}}$$

ANN-based OFTC of EESM
ANN Training

Neuron's output (i-th neuron of the j-th layer):

$$\hat{y}_{j,i} = \Phi_{j,i}\left(\boldsymbol{w}_{j,i}^{\mathrm{T}}\boldsymbol{x}_j + b_{j,i}\right) \in \mathbb{R}$$

Layer's output (j-th layer):

$$\hat{\boldsymbol{y}}_j = \begin{pmatrix} \hat{y}_{j,1} \\ \vdots \\ \hat{y}_{j,n_j} \end{pmatrix} = \begin{pmatrix} \Phi_{j,1}\left(\boldsymbol{w}_{j,1}^{\mathrm{T}}\boldsymbol{x}_j + b_{j,1}\right) \\ \vdots \\ \Phi_{j,n_j}\left(\boldsymbol{w}_{j,n_j}^{\mathrm{T}}\boldsymbol{x}_j + b_{j,n_j}\right) \end{pmatrix} \in \mathbb{R}^{n_j}$$

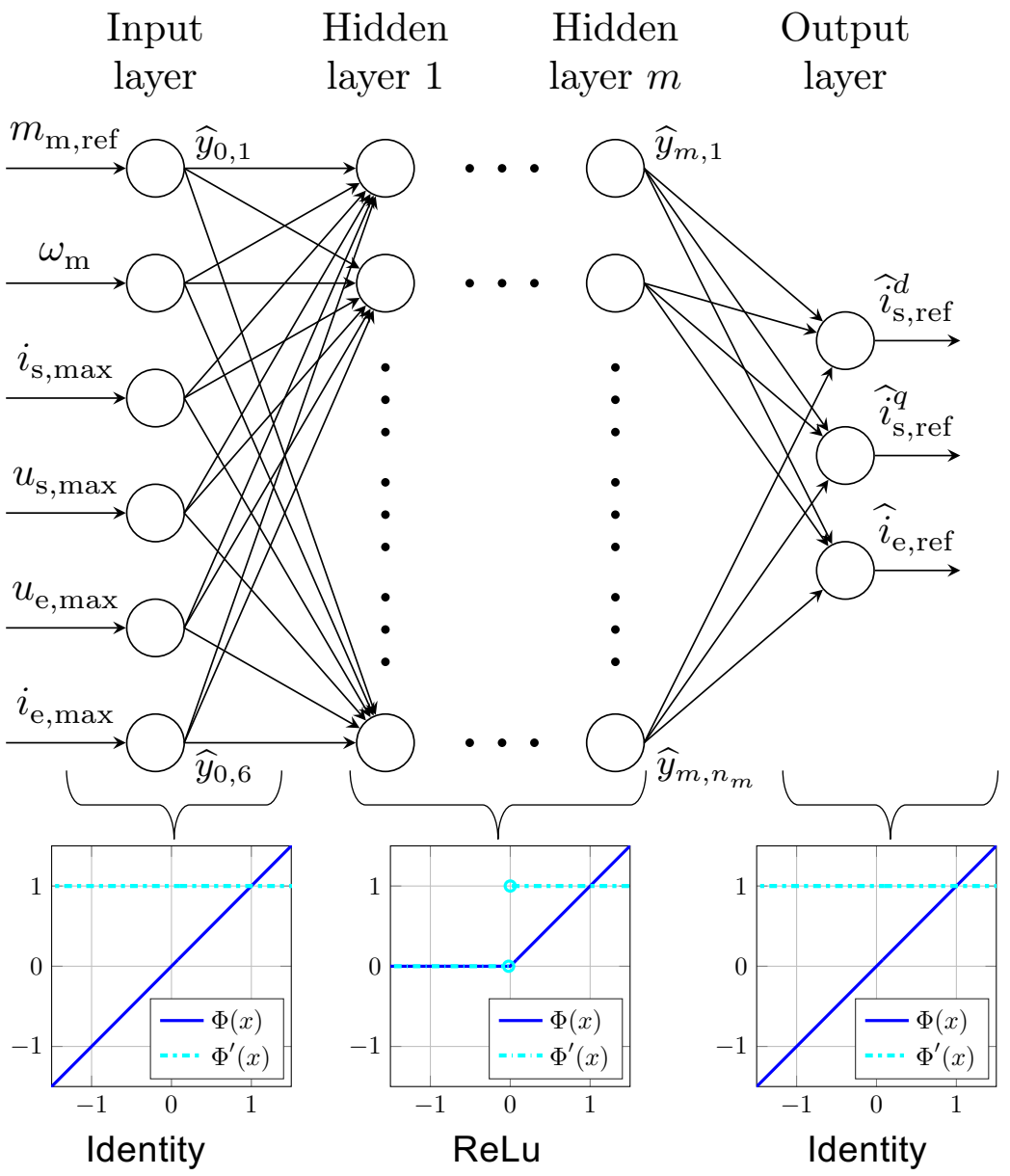
ANN output:

$$\hat{\boldsymbol{y}}[k] \coloneqq \left(\hat{\imath}_{\mathrm{s,ref}}^{d}[k], \hat{\imath}_{\mathrm{s,ref}}^{q}[k], \hat{\imath}_{\mathrm{e,ref}}[k]\right)^{\mathrm{T}} = f(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{b})$$

Training:

$$(\boldsymbol{w}^*, \boldsymbol{b}^*) = \underset{(\boldsymbol{w},\boldsymbol{b})}{\arg\min} \frac{1}{K}\sum_{k=1}^{K} \|\boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k]\|^2$$

Different ANN architectures can be trained
→ Which one is the *best* trained ANN?



ANN-based OFTC of EESM

Choice and validation of an optimal ANN design

$$e_{u_s} = \frac{1}{K}\sum_{k=1}^{K} \max\left(0, \left\|\hat{u}_s^{dq}[k]\right\| - u_{s,max}[k]\right)^2$$

$$e_{u_e} = \frac{1}{K}\sum_{k=1}^{K} \max\left(0, |u_e[k]| - u_{e,max}[k]\right)^2$$

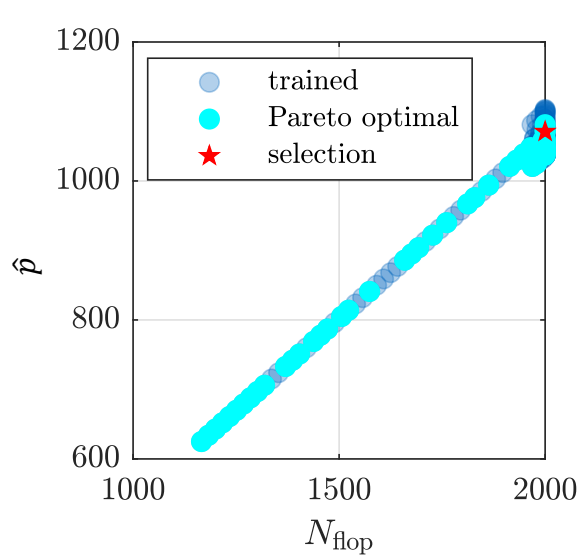
$$e_{i_s} = \frac{1}{K}\sum_{k=1}^{K} \max\left(0, \left\|\hat{i}_s^{dq}[k]\right\| - i_{s,max}[k]\right)^2$$

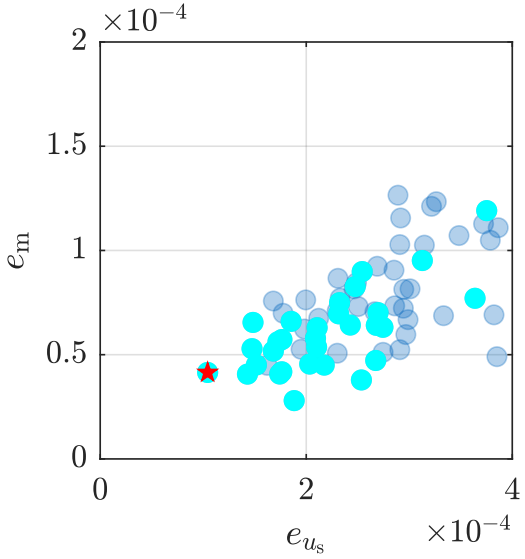
$$e_{i_e} = \frac{1}{K}\sum_{k=1}^{K} \max\left(0, |i_e[k]| - i_{e,max}[k]\right)^2$$

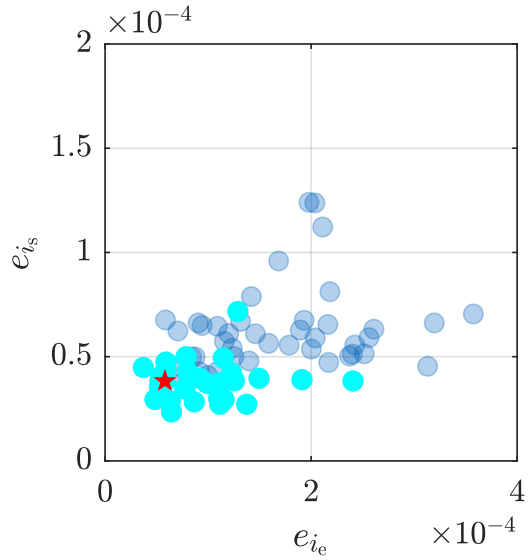
$$e_m = \frac{1}{K}\sum_{k=1}^{K} \left(\hat{m}_m[k] - \underset{m_{m,max}}{\text{sat}}\, m_{m,ref}[k]\right)^2$$

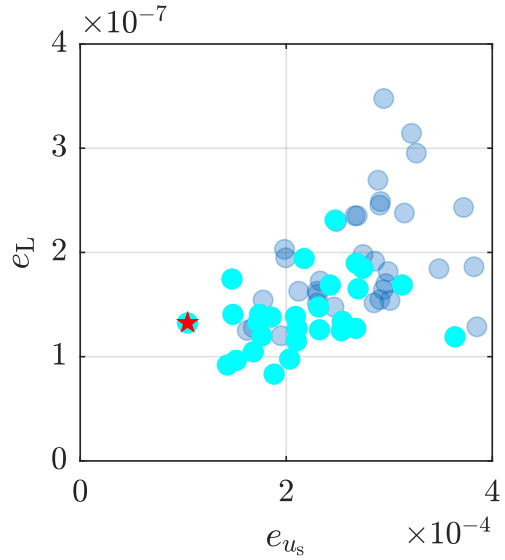
$$e_L = \frac{1}{K}\sum_{k=1}^{K} (\hat{p}_L[k] - p_L[k])^2$$

Number of parameters: $\hat{p}$
Number of floating-point-operations: N_{flop}





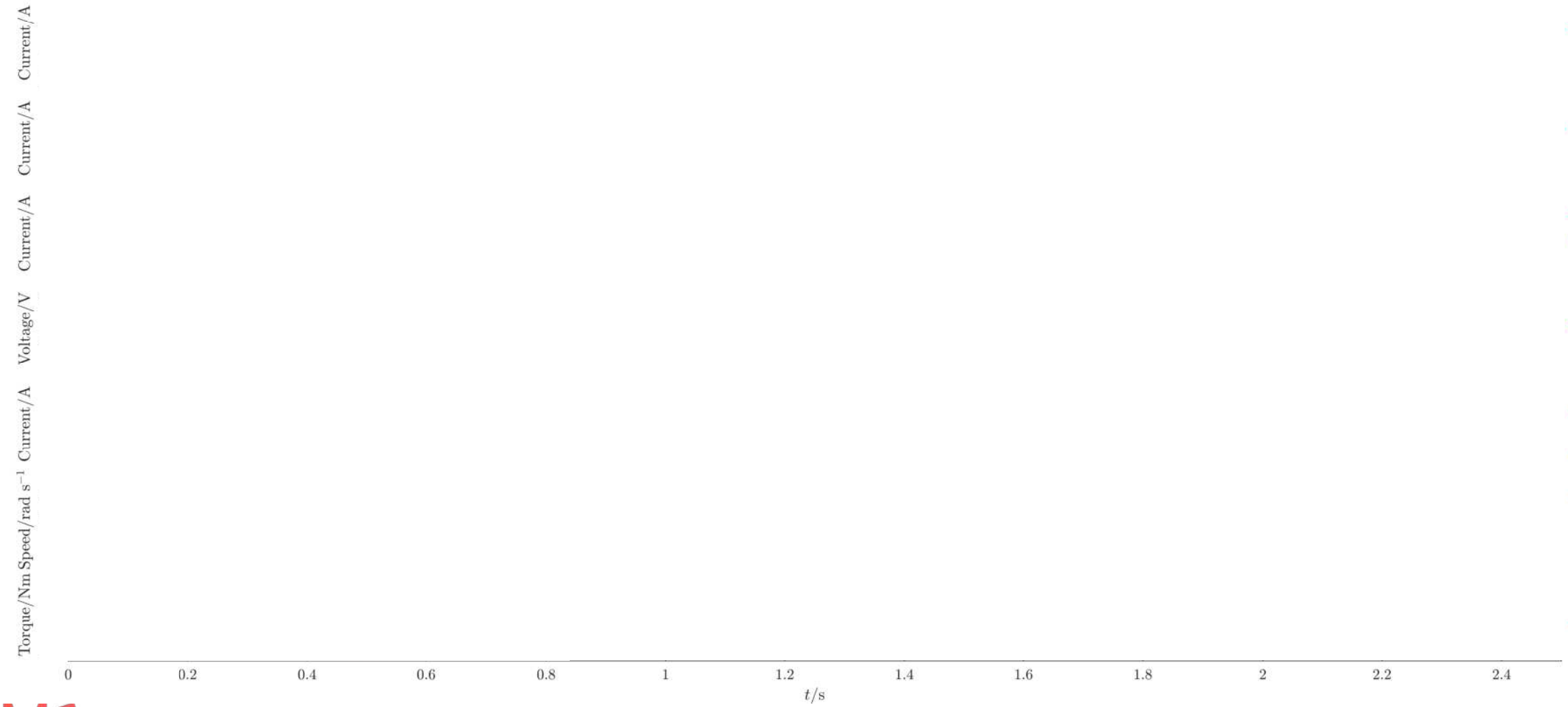




ANN-based OFTC of EESM

Simulation results

no voltage and current limitation	current limitation	stator voltage and current limitation	voltage limitation

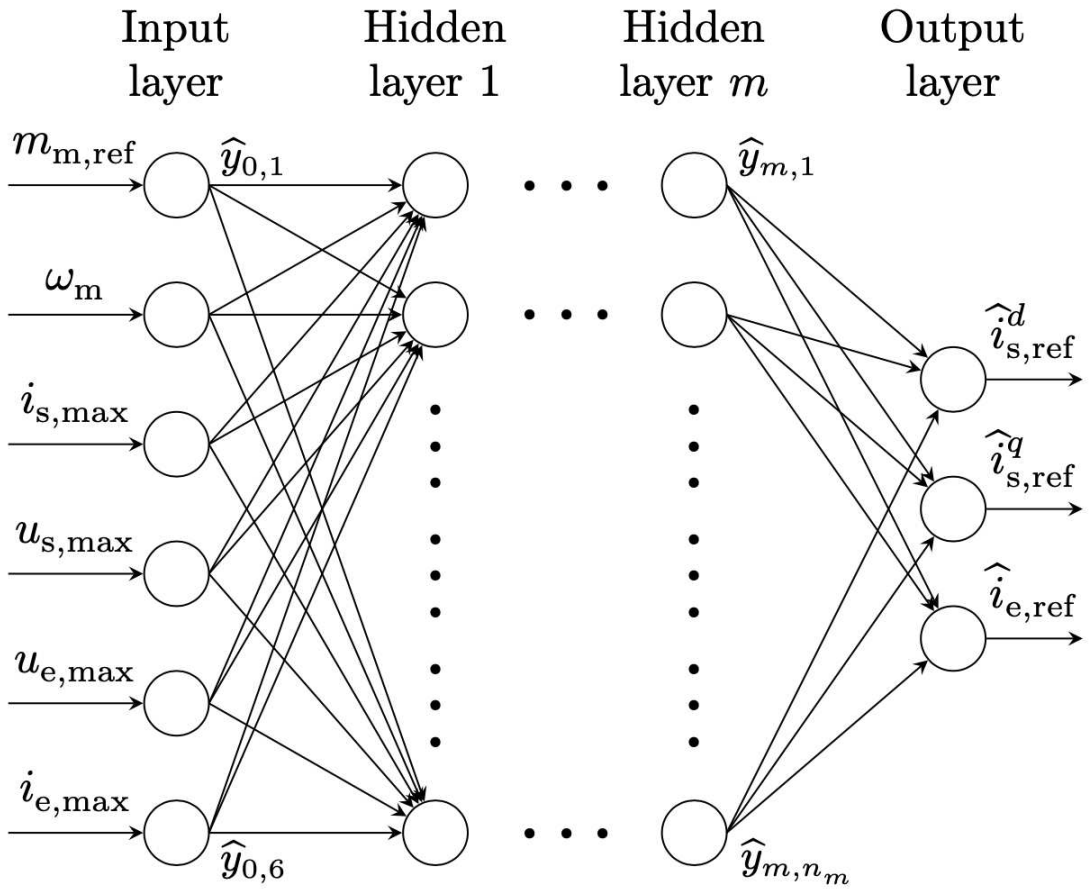


Motivation
ANN-based OFTC for EESM

- ✅ ANN-based OFTC for EESM feasible, considering
  – machine nonlinearities, Iron and copper losses
  – stator and exciter voltage and current limits
  – real-time capable implementation

- ✅ Complete framework to design ANN-based OFTC
  – dataset creation based on the nonlinear machine model
  – training (and validation) of suitable ANN architectures
  – optimal ANN design and choice as result of trade-off between the OFTC goals

- ❗ Meta-study:
  – Number of hidden layers: 1…4
  – Numbers of neurons per hidden layer: 2…200
  – FLOPs limit: 1000…2000
  →1.282.176 ANN architectures
  →*only* 200 ANN architectures chosen (50 with maximum number of FLOPs per number of hidden layers)



Motivation
ANN-based OFTC for RSM

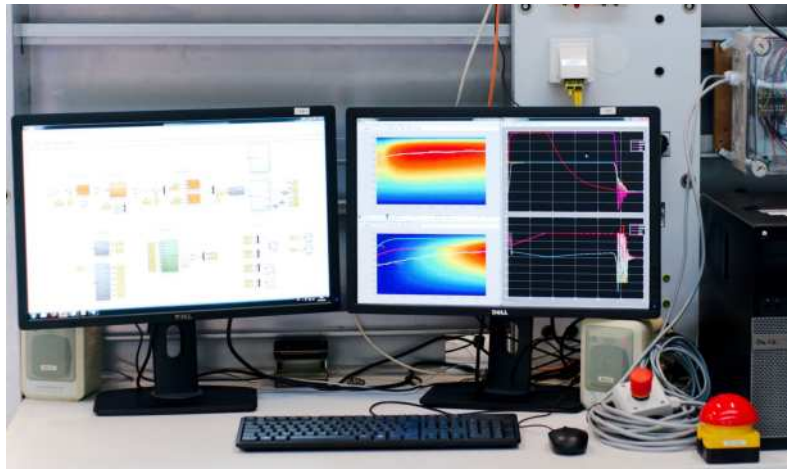
Multi-objective optimization for artificial neural network based optimal feedforward control with application to optimal feedforward torque control of nonlinear synchronous machines

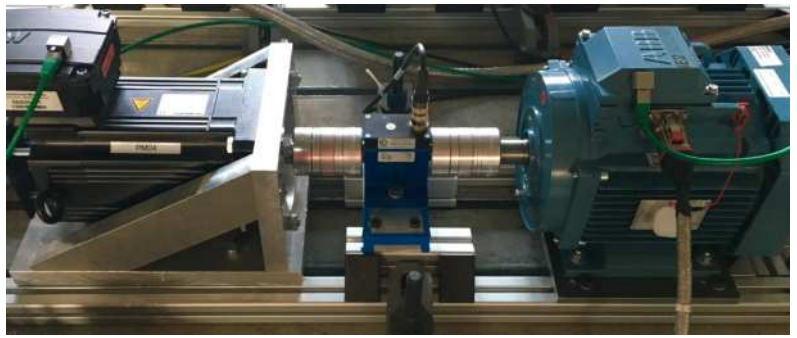
Work in progress…

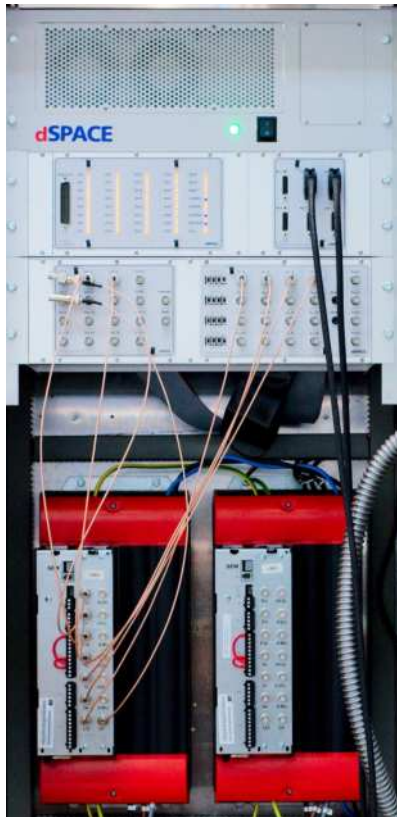
Motivation
ANN-based OFTC for RSM

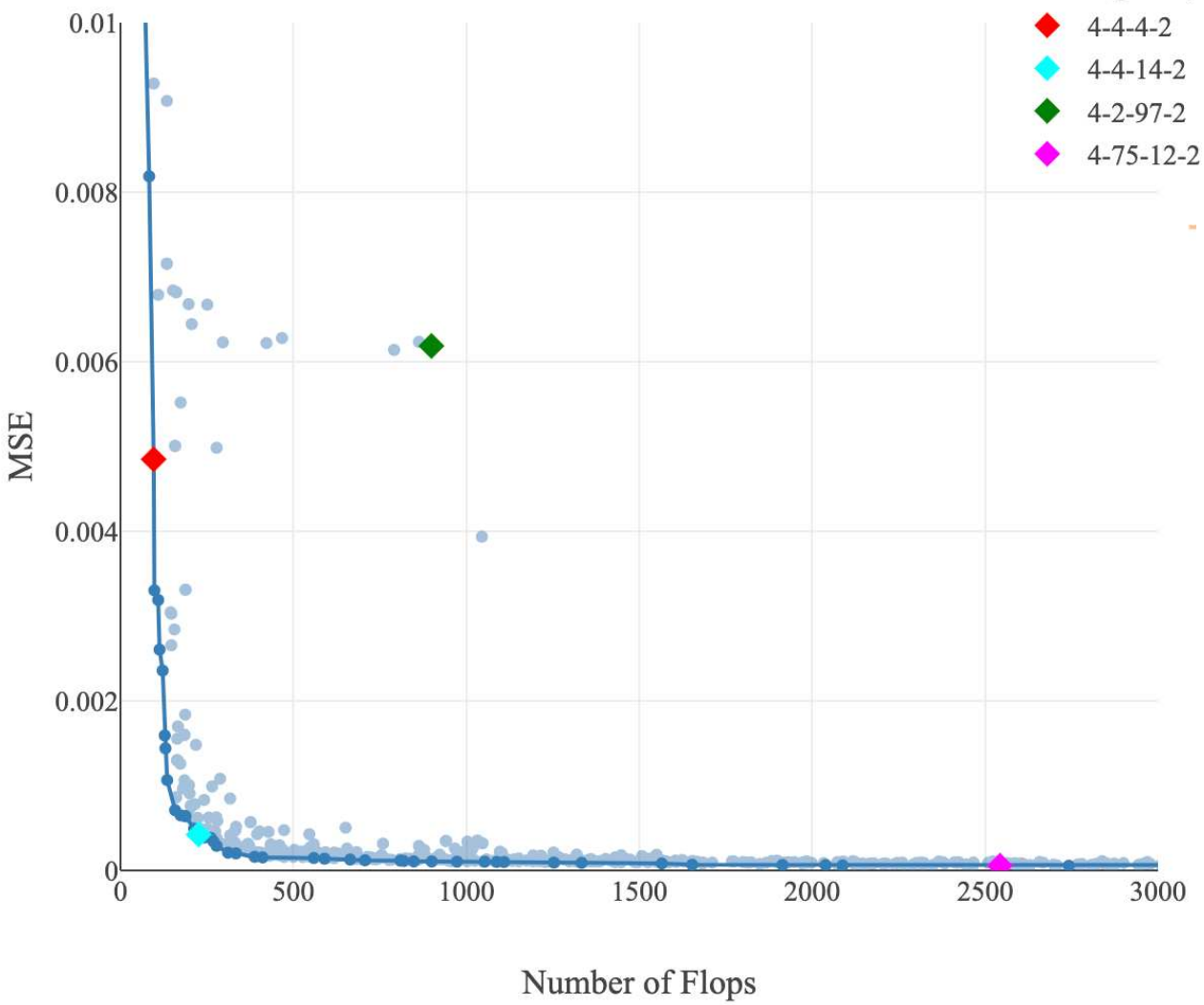
- Multi-Objective Optimization (MOO)
- Target space:
  – Number of FLOPs
  – MSE of outputs
- Hyperparameters:
  – Numbers of neurons per hidden layers
- Goal:
  – minimize all targets simultaneously

Motivation
ANN-based OFTC for RSM



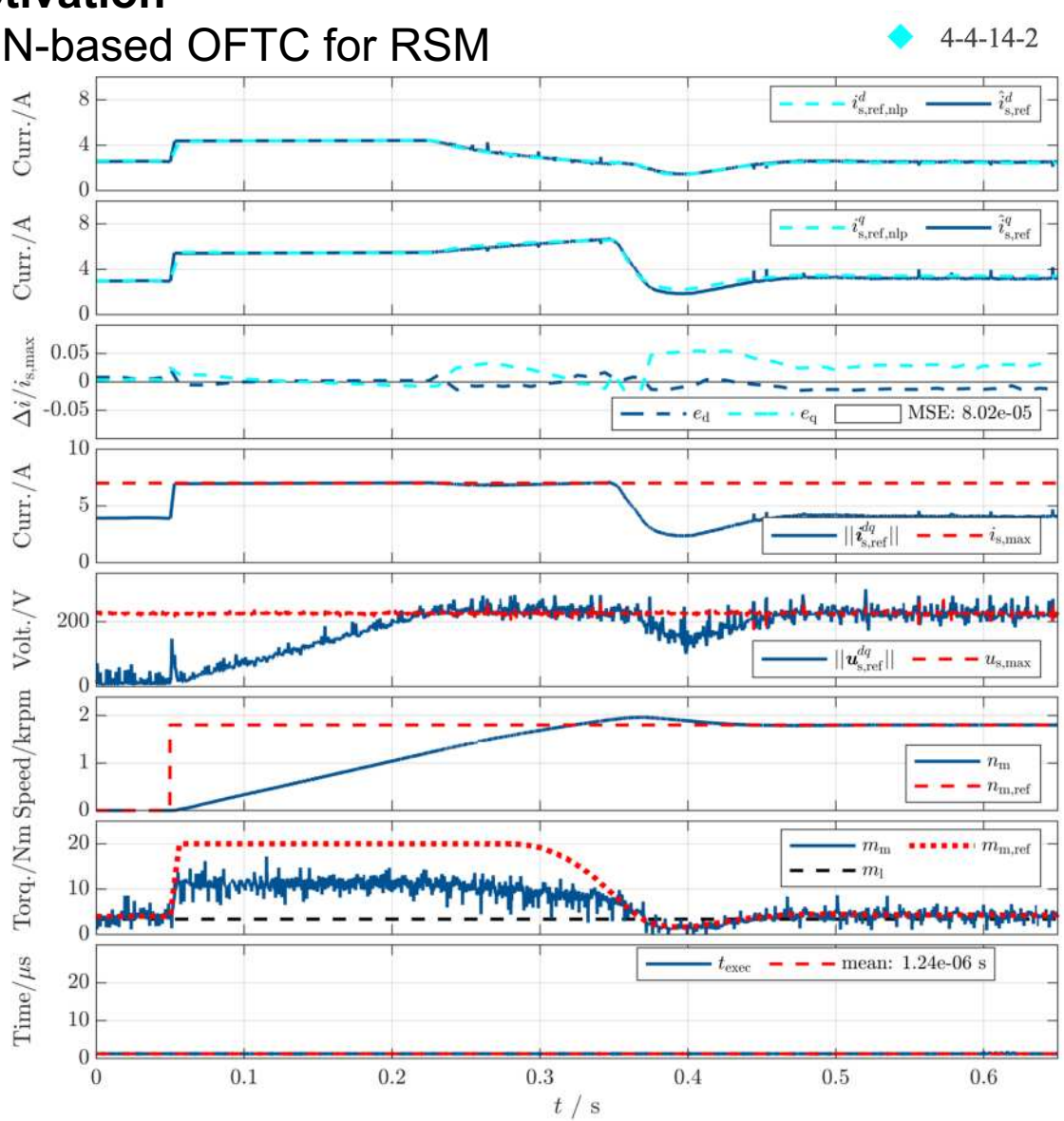


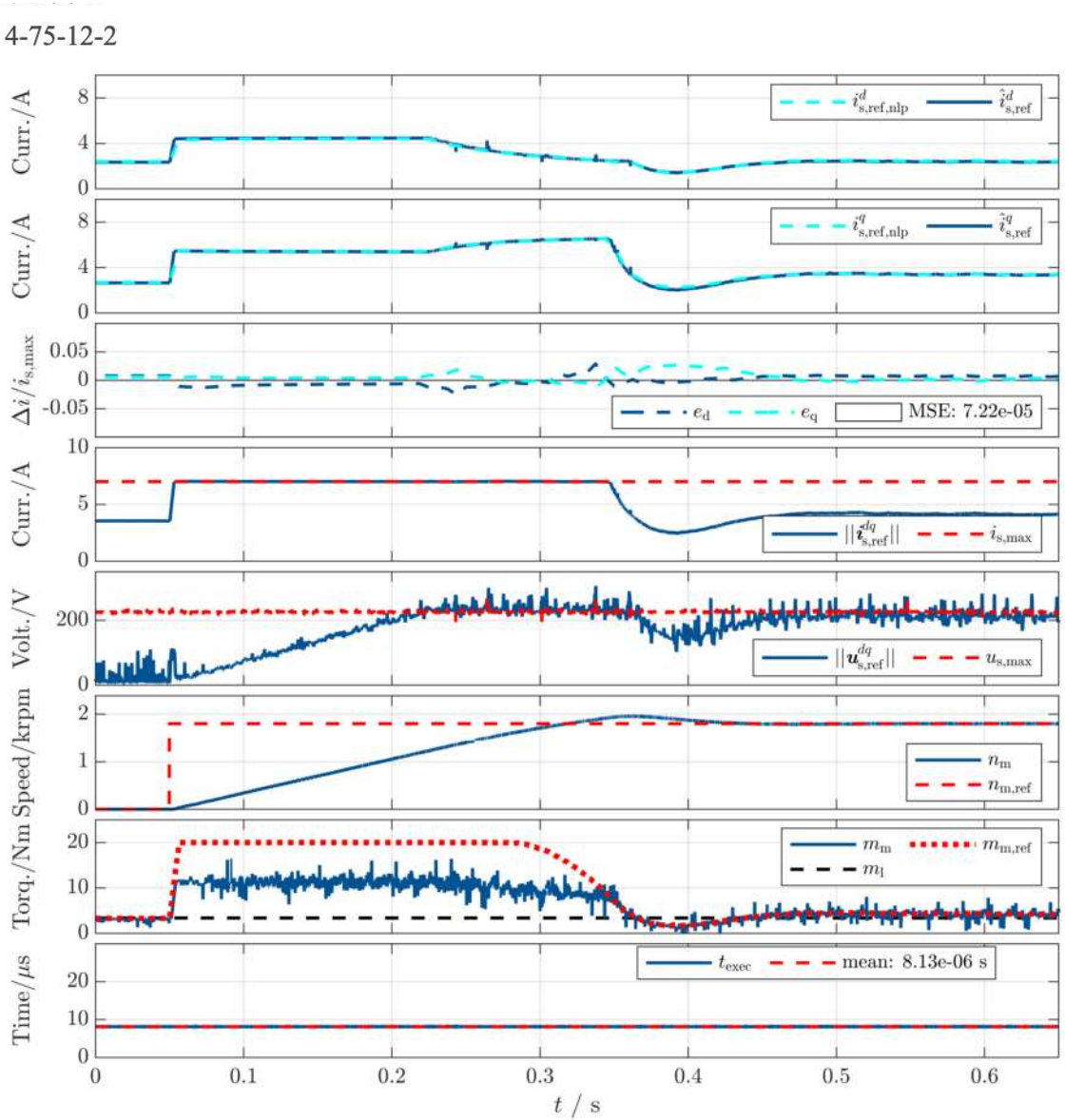




Motivation

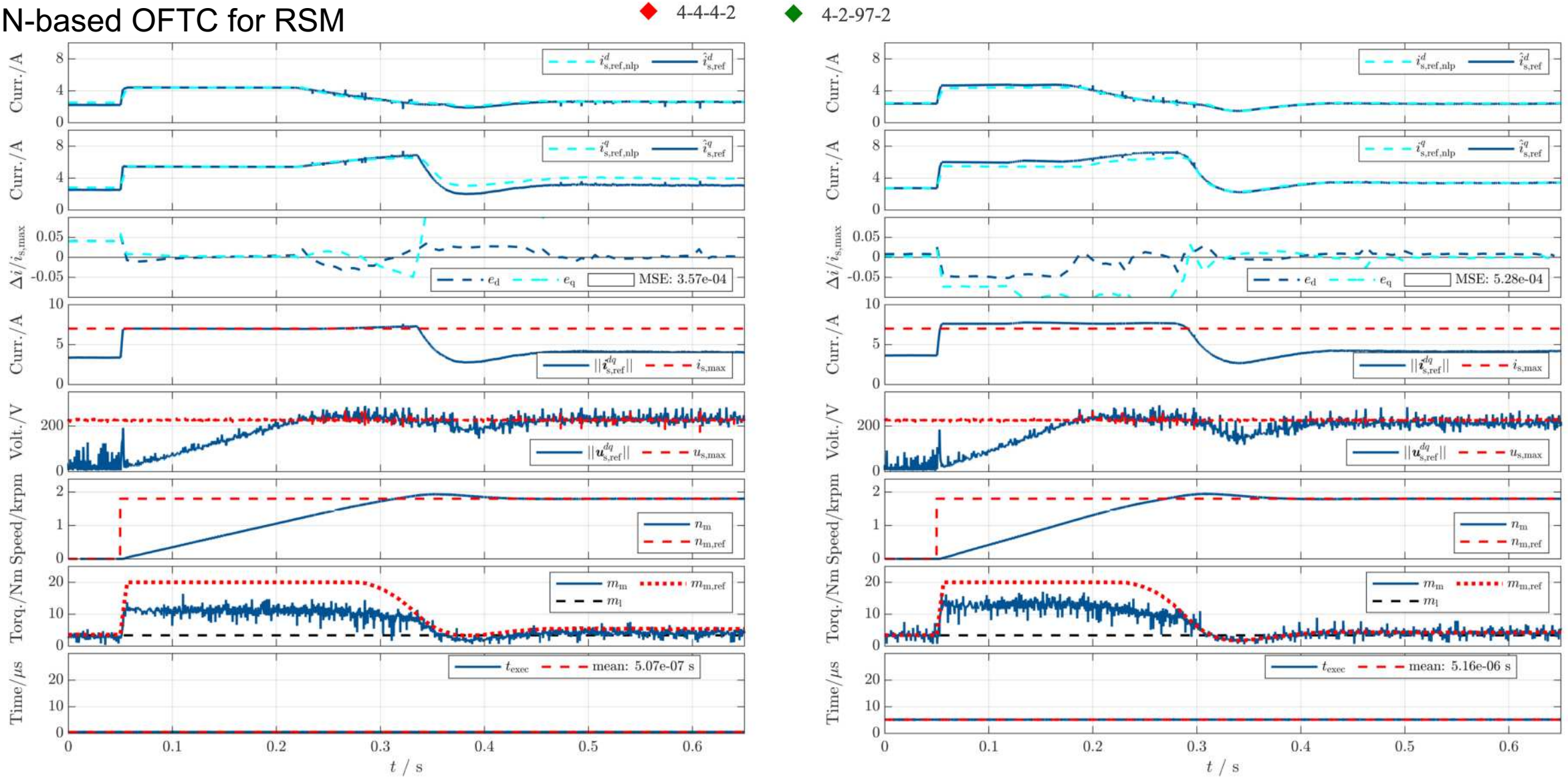
ANN-based OFTC for RSM



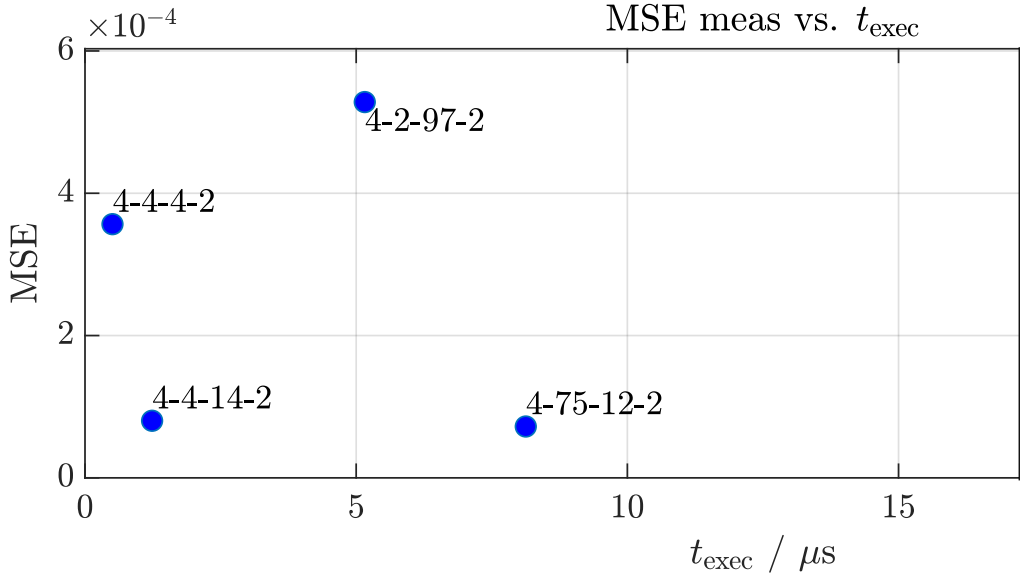


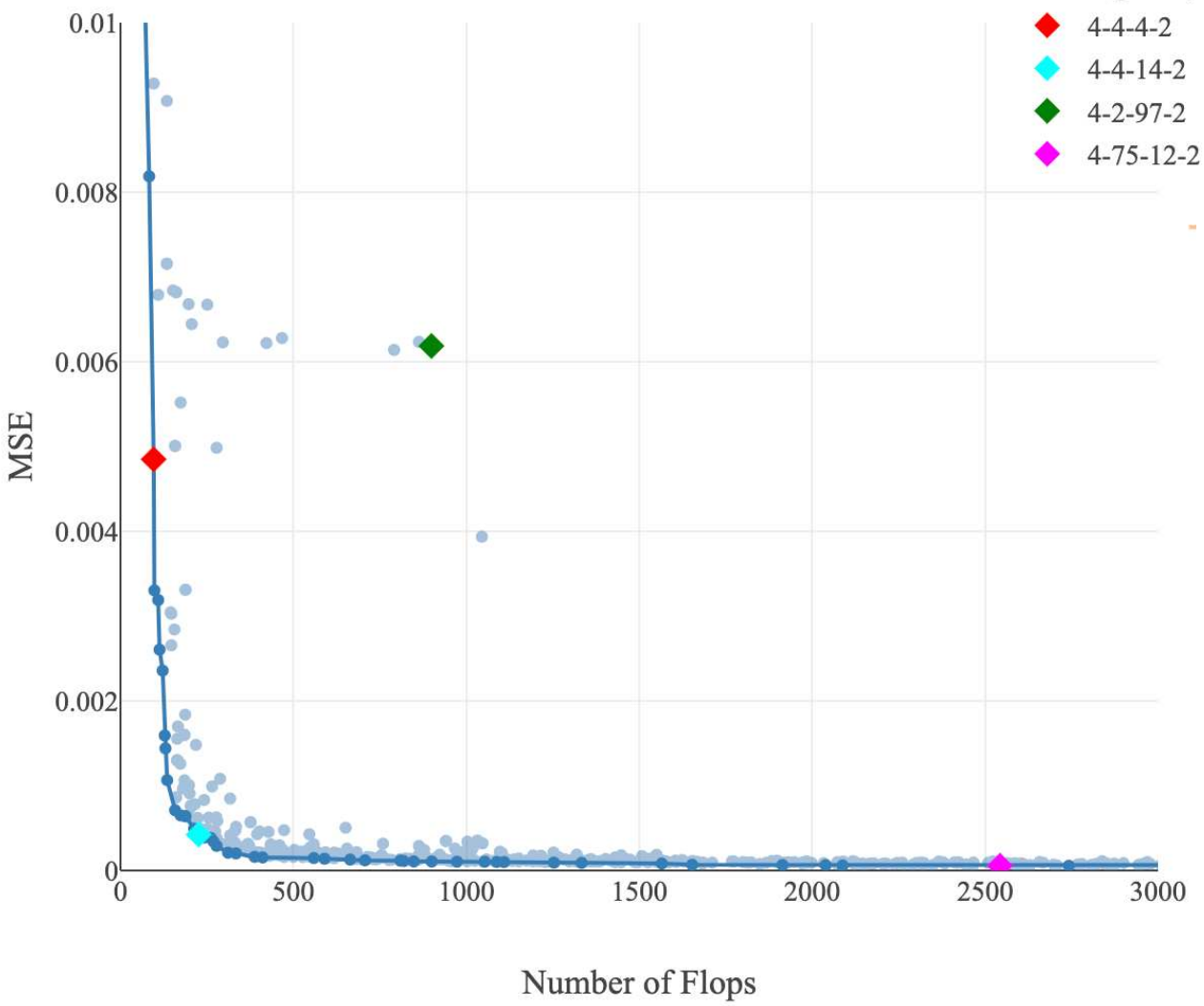
Motivation

ANN-based OFTC for RSM



Motivation
ANN-based OFTC for RSM





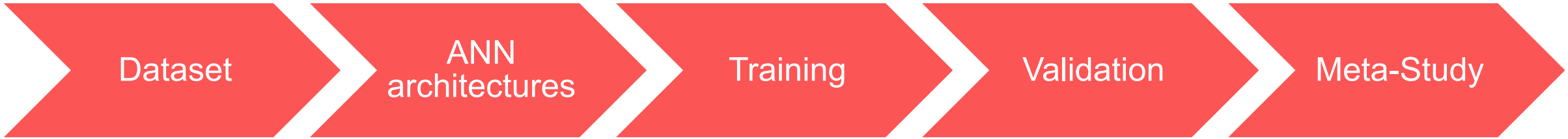
→Experimental validation of ANN-based OFTC for RSM
→best ANN architecture is trade-off between execution time and MSE
→execution time and MSE is significantly lower compared to MTPX-approach

Agenda

- Motivation & applications
- **Framework for design of ANN-based approches for electrical drive systems**
- Hands-on example 1: System identification
- Hands-on example 2: ANN-based Maximum Torque per Losses (MTPL)
- Hands-on example 3: ANN-based Optimal Feedforward Torque Control (OFTC) of an IPMSM

Design of ANNs for electrical drive systems

Overview



Design of ANNs for electrical drive systems

Overview

Dataset → ANN architectures → Training → Validation → Meta-Study

- The data must be created by measurements, FEA or model-based computations.
- Collecting all individual input-output sample sets

$$(\boldsymbol{x}[k]; \boldsymbol{y}[k]) \quad \text{for all } k \in \{1, \dots, K\}$$

- Leads to the overall, trainings and validation input-output dataset

$$(\mathbb{X}, \mathbb{Y}) = (\mathbb{X}_\mathrm{T}, \mathbb{Y}_\mathrm{T}) \cup (\mathbb{X}_\mathrm{V}, \mathbb{Y}_\mathrm{V})$$

- Normalization: scaling to a range of usually 0…1 (or sometimes -1 to +1)

$$x_\mathrm{N} = \frac{x - x_\mathrm{min}}{x_\mathrm{max} - x_\mathrm{min}}$$

For the k-th input sample

$$\boldsymbol{x}[k] := \left(m_{\mathrm{m,ref}}[k], \omega_\mathrm{m}[k], u_{\mathrm{s,max}}[k], i_{\mathrm{s,max}}[k], u_{\mathrm{e,max}}[k], i_{\mathrm{e,max}}[k]\right)^\mathrm{T}$$

the corresponding output vector

$$\boldsymbol{y}[k] := \left(i^d_{\mathrm{s,ref}}[k], i^q_{\mathrm{s,ref}}[k], i_{\mathrm{e,ref}}[k]\right)^\mathrm{T}$$

is the solution of the nonlinear optimization problems (NLP)

$$\boldsymbol{i}^{dqe}_{\mathrm{s,e,ref}}[k] = \arg\min_{\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]} p_\mathrm{L}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)$$

$$\text{s.t.} \quad \begin{aligned} \left\|\boldsymbol{u}^{dq}_s\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right\| &\le u_{\mathrm{s,max}}[k], \\ \left\|\boldsymbol{i}^{dq}_s\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right\| &\le i_{\mathrm{s,max}}[k], \\ \left|u_\mathrm{e}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right| &\le u_{\mathrm{e,max}}[k], \\ |i_\mathrm{e}[k]| &\le i_{\mathrm{e,max}}[k], \\ m_\mathrm{m}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right) &= \operatorname{sat}_{m_{\mathrm{m,max}}} m_{\mathrm{m,ref}}[k] \end{aligned}$$

$$m_{\mathrm{m,max}}[k] = \max_{\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]} m_\mathrm{m}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)$$

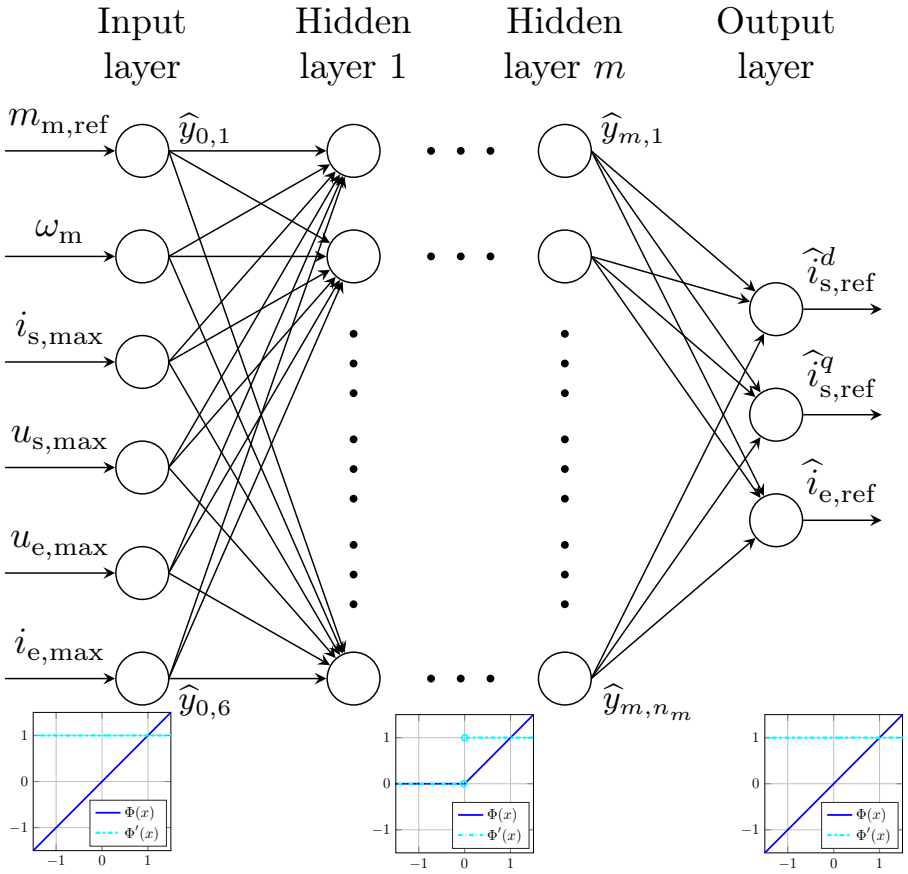
$$\text{s.t.} \quad \begin{aligned} \left\|\boldsymbol{u}^{dq}_s\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right\| &\le u_{\mathrm{s,max}}[k], \\ \left\|\boldsymbol{i}^{dq}_s\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right\| &\le i_{\mathrm{s,max}}[k], \\ \left|u_\mathrm{e}\left(\boldsymbol{i}^{dqe}_{\mathrm{s,e}}[k]\right)\right| &\le u_{\mathrm{e,max}}[k], \\ |i_\mathrm{e}[k]| &\le i_{\mathrm{e,max}}[k]. \end{aligned}$$

Design of ANNs for electrical drive systems

Overview

Dataset | **ANN architectures** | Training | Validation | Meta-Study

- The ANN architecture is defined by
  - Number of input neurons
  - Number of output neurons
  - Number of hidden layers
  - Numbers of neurons per hidden layers
  - Activation functions



Layer's output (j-th layer):

$$\widehat{\boldsymbol{y}}_j = \begin{pmatrix} \hat{y}_{j,1} \\ \vdots \\ \hat{y}_{j,n_j} \end{pmatrix} = \begin{pmatrix} \Phi_{j,1}\left(\boldsymbol{w}_{j,1}^{\mathsf{T}} \boldsymbol{x}_j + b_{j,1}\right) \\ \vdots \\ \Phi_{j,n_j}\left(\boldsymbol{w}_{j,n_j}^{\mathsf{T}} \boldsymbol{x}_j + b_{j,n_j}\right) \end{pmatrix} \in \mathbb{R}^{n_j}$$

ANN output: $\widehat{\boldsymbol{y}}[k] = f\left(\boldsymbol{x}[k], \widehat{\boldsymbol{\theta}}\right)$

Design of ANNs for electrical drive systems

Overview

Dataset | ANN architectures | **Training** | Validation | Meta-Study

- Define the error function (or loss function)

$$E(\hat{\boldsymbol{\theta}})$$

- Optimization

$$\hat{\boldsymbol{\theta}}^* = \underset{\hat{\boldsymbol{\theta}}}{\operatorname{argmin}}\, E(\hat{\boldsymbol{\theta}})$$

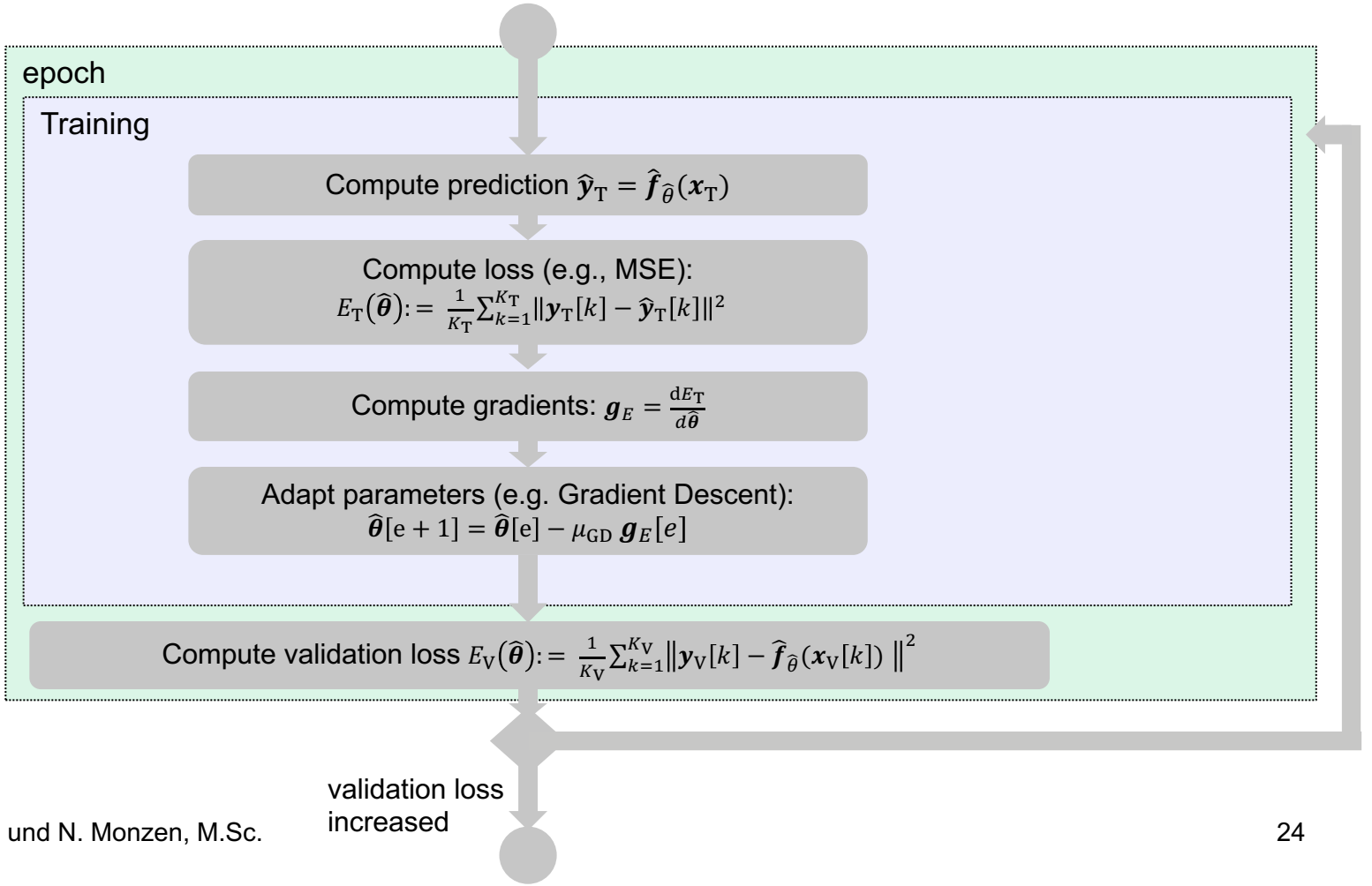
- Compute gradients (backpropagation)

gradients: $\boldsymbol{g}_E = \frac{\mathrm{d}E}{d\hat{\boldsymbol{\theta}}}$

- Iterative optimization of error

$$\hat{\boldsymbol{\theta}}[b+1] = \hat{\boldsymbol{\theta}}[b] + \boldsymbol{H}_{\hat{\boldsymbol{f}}}^{-1} \boldsymbol{J}_{\hat{\boldsymbol{f}}}^{\top} \boldsymbol{e}[b]$$

- Prevent overfitting



Design of ANNs for electrical drive systems

Overview

Dataset | ANN architectures | **Training** | Validation | Meta-Study

- Define the error function (or loss function)

$$E(\hat{\boldsymbol{\theta}})$$

- Optimization

$$\hat{\boldsymbol{\theta}}^* = \underset{\hat{\boldsymbol{\theta}}}{\operatorname{argmin}}\, E(\hat{\boldsymbol{\theta}})$$

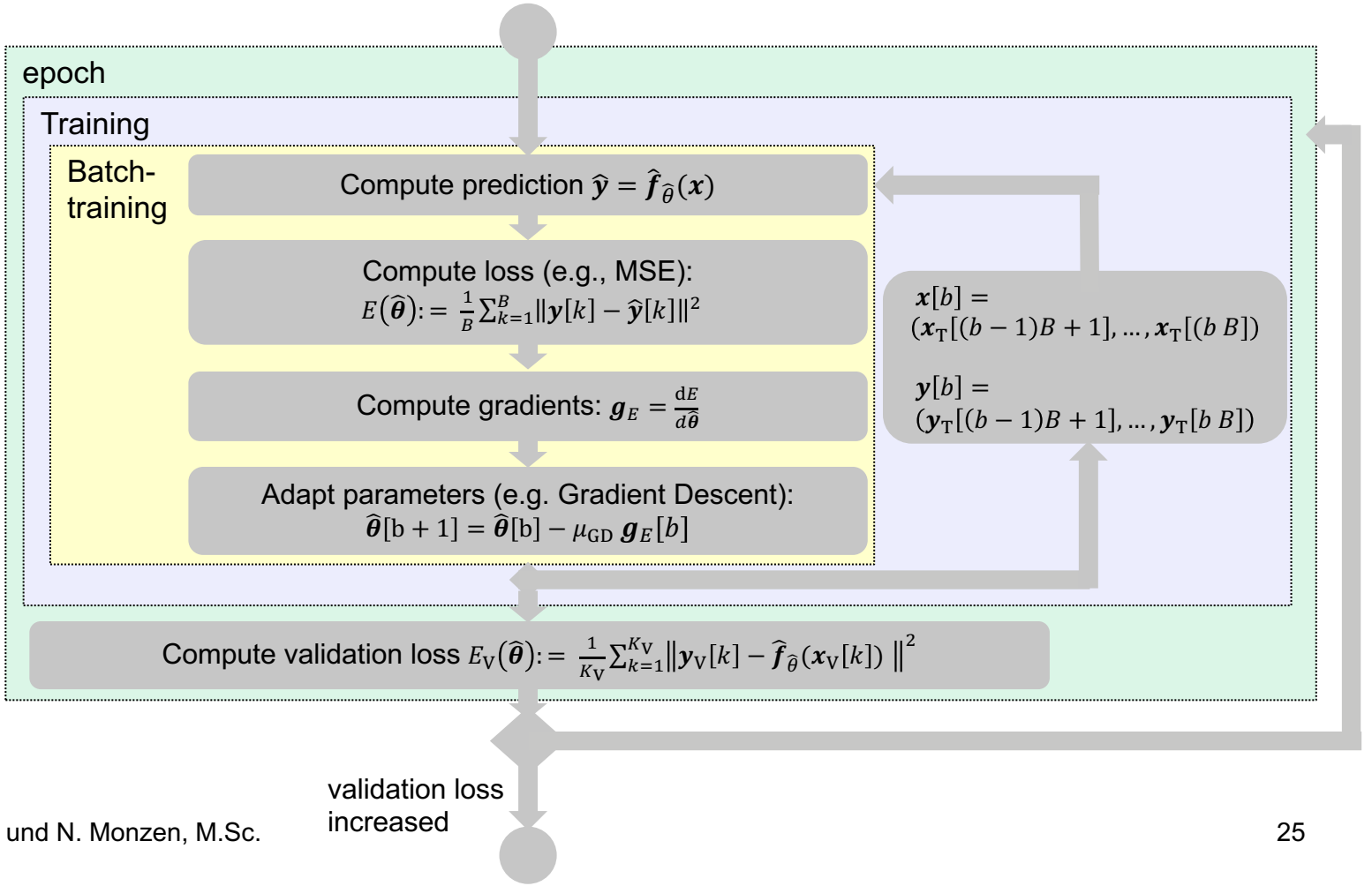
- Compute gradients (backpropagation)

$$\text{gradients: } \boldsymbol{g}_E = \frac{\mathrm{d}E}{\mathrm{d}\hat{\boldsymbol{\theta}}}$$

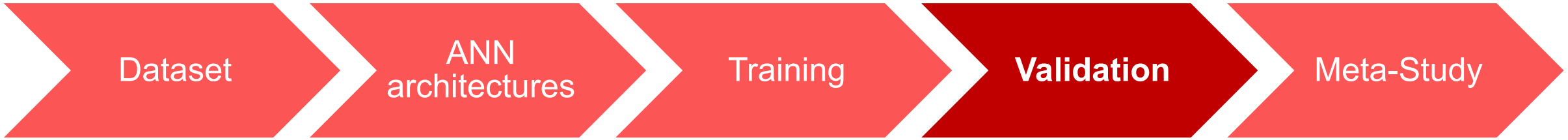
- Iterative optimization of error

$$\hat{\boldsymbol{\theta}}[b+1] = \hat{\boldsymbol{\theta}}[b] + \boldsymbol{H}_{\hat{\boldsymbol{f}}}^{-1} \boldsymbol{J}_{\hat{\boldsymbol{f}}}^{\top} \boldsymbol{e}[b]$$

- Prevent overfitting
- Batched-training with batch-size B



Design of ANNs for electrical drive systems
Overview



- Compute MSE of test dataset
- Computational resources:
  – Number of parameters $\hat{p}$

$$\hat{p} := \sum_{l=1}^{L} \hat{p}_l = \sum_{l=1}^{L} m_l \left(m_{l-1} + 1\right)$$

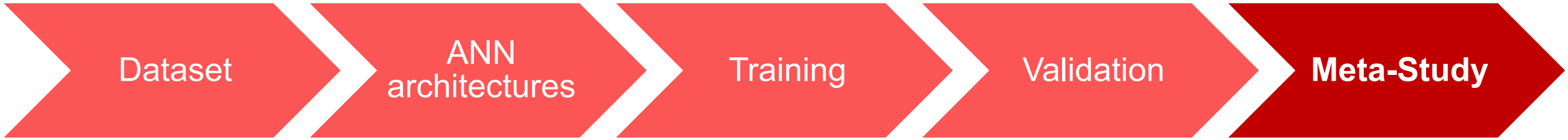
  – Number of floating-point-operations: N_{flop}

$$N_{\mathrm{flop}} = 2\,n_0 + 2\,n_{m+1}\,n_m + \sum_{k=1}^{m} n_k \left(2\,n_{k-1} + 1\right)$$

- Dynamic simulation
- Experimental validation

Design of ANNs for electrical drive systems

Overview



- Parameter Study
- Multi-objective Optimization

- Targets:
  - MSE of all outputs
  - MSE of each output
  - Application motivated MSE (e.g., torque, voltage, current)
  - Number of parameters: N_{param}
  - Number of floating-point-operations: N_{flop}

- Hyperparameters
  - architectures
    - Numbers of neurons per hidden layers
    - Number of hidden layers
    - Activation functions
  - Optimization
    - Initialization of parameters
    - Optimizer & optimizer parameters
  - Dataset
    - Sampling (e.g., grid, latin-hypercube sampling)
    - Normalization

Design of ANNs for electrical drive systems

Tools

Pros:
Distributed training, Azure integration, multi-language support (Python, C#, C++), production deployment.

Cons:
Smaller community, limited resources, Azure-centric, advanced customization

Microsoft Cognitive Toolkit

Pros:
Distributed computing and GPU acceleration, high/low-level APIs (e.g. Keras)

Cons:
Steeper learning curve, compatibility issues.

TensorFlow

PyTorch

Pros:
Dynamic computation graph allows flexibility, GPU acceleration, Pythonic syntax

Cons:
Less optimized for production

MATLAB SIMULINK

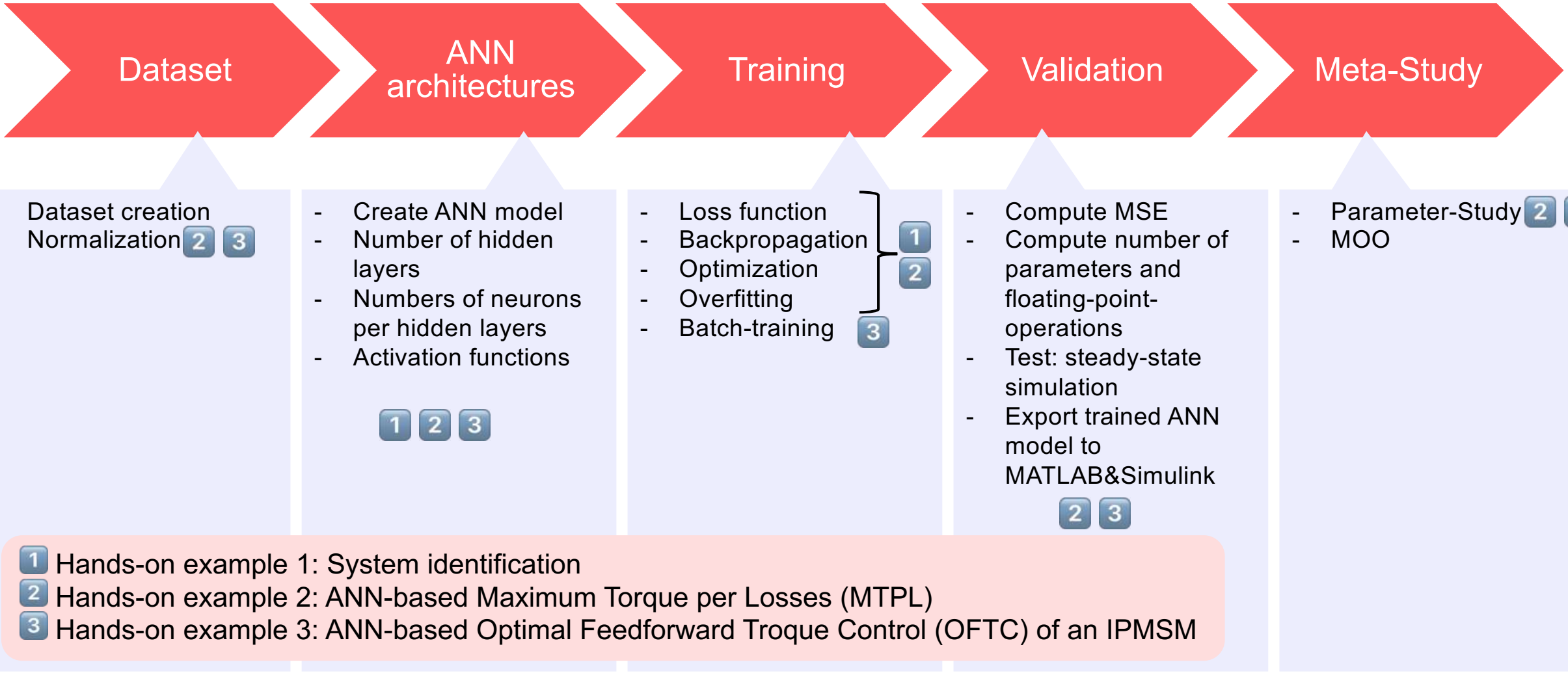
Deep Learning Toolbox

Pros:
Wide range of built-in functions, user-friendly interface.

Cons:
Can be slow, limited support for distributed computing, limited advanced customization, proprietary software

Design of ANNs for electrical drive systems

Overview



Hands-on examples
Getting started

Install Python 3.X: https://www.python.org/downloads/

Download and extract Workshop files:

https://monzen.pages.gitlab.lrz.de/data-sharing/Workshop_CPM_LMRES.zip

Set up virtual environment [optional]
- Open a terminal in `Workshop_CPM_LMRES` folder.
- Create a virtual environment `env`:

```
python -m venv env
```

- and activate the environment by

```
source env/bin/activate  // MAC
env/Scripts/activate.bat // In CMD (WINDOWS)
env/Scripts/Activate.ps1 // In Powershell (WINDOWS)
```

Installation of python packages:

```
pip install -r requirements.txt
```

Start Jupyter Notebook `jupyter notebook`

Alternative: Google Colab
- Google account required
- Open the shared notebooks

Hands-on example 1: System identification

Hands-on example 2: ANN-based Maximum Torque per Losses (MTPL)

Hands-on example 3: ANN-based Optimal Feedforward Torque Control (OFTC) of an IPMSM