Deep Learning-Based Control of Hydrogen Dual-Fuel Engines

Deep-Learning-basierte Regelung von Wasserstoff-Zweikraftstoffmotoren

Von der Fakultät für Maschinenwesen der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

genehmigte Dissertation

vorgelegt von

Alexander Emil Klaus Winkler

Berichter: Univ.-Prof. Dr.-Ing. Jakob Lukas Andert

Prof. Dr.Ir. Frank Willems

Tag der mündlichen Prüfung: 17. April 2026

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

Für Mama, Papa & Joni.

Foreword

This thesis was written during my time as a research associate at the Chair of Mechatronics in Mobile Propulsion (MMP) of RWTH Aachen University. My special thanks go to my doctoral supervisor Prof. Dr.-Ing. Jakob Andert. Dear Jakob, thank you for your continuous support, your advice, and the patience to challenge and encourage me while letting me grow freely as a scientist and engineer, including your great understanding for my urge to go to Canada, and for always having an open ear for my questions and ideas. I also sincerely thank Prof. Dr. Ir. Frank P.T. Willems for acting as the second examiner of this thesis. Dear Frank, I thoroughly enjoyed our exchanges at conferences and in Eindhoven. Furthermore, I thank Prof. Dr.-Ing. Niklas von der Aßen for chairing the examination board. Go Aachen Steelers!

The results of this work originated within the research unit FOR2401, funded by the German Research Foundation (DFG). A big thank you goes to the entire FOR2401 team, in particular to Jonathan and Katrin at IMTEK of the University of Freiburg, to Gio at ETH Zürich, and to Eugen and Xu at the Institute of Automatic Control of RWTH Aachen University. Eugen, you have supported me with your MPC knowledge ever since my master's thesis, thank you! A special mention naturally goes to my FOR2401 project team at MMP, Julian and Patrick: TP3! You welcomed me to the team from day one and showed me the ropes. Our trips to the World Congress in Japan and to Zürich, as well as the many FOR2401 project meetings, remain unforgettable. Thank you for the great time together.

All research presented in this thesis was carried out in close collaboration with the Department of Mechanical Engineering at the University of Alberta (UofA), under the guidance of Prof. Bob Koch and David Gordon. All experimental investigations were conducted at the UofA facilities during my research stays in 2022 and 2024. These stays were generously supported by the MITACS Globalink Research Internship and the RWTH Aachen University Junior Research Fellowship. David, from the day you first joined one of our group meetings, you effectively took on the role of a co-supervisor. I am incredibly grateful for your continuous guidance, your kindness, and our joint work at the engine, and for welcoming me and the students I brought along so warmly in Canada. You made my stays unforgettable. Say hi to the fam! Armin, thank you for your great ideas, your support, and the rewarding collaboration. Your ideas set our successful publication journey in motion, and I am grateful for everything we developed together along the way.

Working with the many students I had the pleasure of (co-)supervising was one of the highlights of my doctorate. I am grateful to all of you for your contributions through your theses and your work as student assistants at MMP, and would like to give special thanks to Pranav, Vasu, Michel, Johannes, and Neeraj. Pranav, thank you especially for your perseverance throughout our paper writing and for the dedication you brought to our work together. Vasu, my long-standing alpha student assistant and, by the end, an independent scientist: I am proud of what

we have achieved! Johannes, thank you for joining me in Canada for your bachelor's thesis, for continuing our work together through your master's thesis, and above all for the great friendship that we share. Pranav, Michel, and Neeraj, seeing you all go on to work at the institute makes me particularly proud. It has been a pleasure to accompany part of your journey and to see you continue your own paths in research as colleagues.

Dear colleagues and student assistants of the MMP team, a big thank you to all of you who supported me in preparing this work and accompanied me through this exciting time. Soyo and Markus, thank you for always being there with advice and support. A special thank you goes to my R224 companions, above all Momo, who was there from the very beginning. Momo, from our first semester in 2013 to the end of my PhD in 2026, you have been by my side; without you, my path would certainly have looked very different. Thank you for this great friendship! Lucas, thank you for the countless conversations across the desk, and for a friendship that goes beyond work. Christoph, Björn and Tobi, thank you for the many great moments we shared both in and outside the office. Together with Li, Chenye, Timm, and Michael, you all made R224 a place I always enjoyed coming to. Thanks to the best office! Fabian, Yannick, Andreas, and Chenye, developing and teaching HiLMA with you was one of the best parts of my time at MMP.

Max and Flo, thank you for your friendship and for making our "Diss Vacation" such a great memory: Flo, for sharing the suffering as someone who had been through a PhD himself, and Max, for your software engineering expertise and inspiration since the early days. Chris, my Unicup teammate on the ice, thank you for becoming my teammate throughout my PhD as well, every headache included. My thanks also go to the Doko group in Aachen, especially Bob and Sebi, for the many conversations, the exchange of ideas, and the encouragement along the way. Anni and Louis, thank you for being my best friends from home through all of this, and for patiently listening to things you were probably not that interested in.

Finally, my biggest thanks and all my love go to my family, Mama, Papa, and Joni, for your support on my way. Seeing you proud and feeling your backing has always driven me. And last but not least, thank you, dear Emely, for your support and your patience with a stressed Alex and for being a calm place to come back to.

Aachen, 16. Oktober 2026 Alexander Emil Klaus Winkler

Abstract

Medium-duty (MD) and heavy-duty (HD) transport contributes to a large share of CO_2 emissions due to high utilization and long vehicle lifetimes. These vehicles have long ranges and energy demands, making full electrification impractical in the near term. Retrofitting existing diesel engines for hydrogen-diesel dual-fuel (H2DF) operation offers a cost-effective, scalable way to cut carbon emissions by partially substituting diesel with hydrogen while using existing infrastructure. However, H2DF combustion's fast nonlinear dynamics and elevated nitrogen oxide (NO_x) emissions pose a challenge for conventional controllers. Therefore rapidly developed advanced control methods deployable on low-cost embedded hardware are needed to ensure safe, low-emission operation. This thesis addresses these needs by proposing a modular, data-driven control development workflow by leveraging deep learning techniques. A gated recurrent unit-based deep neural network (DNN) is trained on over 99,000 cycles of experimental H2DF engine data. The model accurately predicts key combustion outputs including indicated mean effective pressure (IMEP), NO_x concentration, particulate matter (PM) concentration, and maximum pressure rise rate (MPRR). As an additional step, automated machine learning (AutoML) is employed on high performance computing to automatically optimize network architectures, balancing accuracy and efficiency. The trained DNN is directly integrated as learned dynamics into a nonlinear model predictive control (MPC) using the open-source framework `acados`. The formulation includes augmented controls and handling of soft constraints via slack penalties. In engine experiments, tracking dynamic load reference trajectories, the MPC improves IMEP tracking by 27.8% compared to a baseline diesel-only controller from the engine manufacturer, reduces soot emissions by 61.1%, and achieves high hydrogen substitution. This enables a potential reduction in carbon emissions of 39.7% while satisfying all operational constraints. As expected, however, NO_x emissions increase substantially, with the mean value rising by 105.1%. The proposed *Raspberry Pi*-based toolchain enables end-to-end control development in under one day, significantly accelerating and simplifying the workflow compared to conventional methods. To overcome the real-time computational burden of MPC on constrained embedded targets, a behavior cloning (BC) approach is implemented. The cloned policy successfully replicates the MPC behavior on the test bench, with inference times below 2 ms including communication overhead, when deployed on the engine. Embedded profiling further confirms the feasibility of deploying the BC controller on resource-limited platforms such as the *ESP32*. This thesis presents a fast and cost-effective framework for developing advanced control strategies for retrofitted H2DF engines. The proposed approach supports safe, low-emission, and flexible engine operation, offering a practical solution for decarbonizing MD and HD transport where alternative propulsion methods such as electrification or the large-scale adoption of renewable fuels still face challenges to widespread implementation.

Zusammenfassung

Der mittlere (MD) und schwere (HD) Transportsektor trägt aufgrund hoher Auslastung und langer Fahrzeuglebensdauer wesentlich zu den CO_2-Emissionen bei, wodurch eine vollständige Elektrifizierung in naher Zukunft als unrealistisch gilt. Das Umrüsten bestehender Dieselmotoren auf Wasserstoff-Diesel-Zweikraftstoff-Betrieb (H2DF) bietet eine kosteneffiziente und skalierbare Möglichkeit zur CO_2-Reduktion durch partielle Dieselkraftstoffersetzung bei Nutzung bestehender Infrastruktur. Die schnellen, nichtlinearen Verbrennungsdynamiken sowie erhöhte NO_x-Emissionen im H2DF-Betrieb stellen für konventionelle Regelungen eine Herausforderung dar, sodass schnell entwickelbare, fortschrittliche Regelungsmethoden auf kostengünstigen Steuergeräten benötigt werden. Diese Arbeit adressiert diese Anforderungen durch einen modularen, datengetriebenen Entwicklungsprozess unter Nutzung von Deep Learning. Ein tiefes neuronales Netz (DNN) mit Gated Recurrent Units wird auf über 99.000 Zyklen experimenteller H2DF-Motordaten trainiert. Das Modell sagt zuverlässig zentrale Verbrennungsgrößen voraus, darunter den Indizierten Mitteldruck (IMEP), die NO_x-Konzentration, die Rußpartikelmassekonzentration (PM) und die maximale Druckanstiegsgeschwindigkeit (MPRR). Zusätzlich wird automatisiertes maschinelles Lernen (AutoML) auf Hochleistungsrechnern eingesetzt, um die Netzwerkarchitektur von DNN automatisiert hinsichtlich Prädiktionsqualität und Effizienz zu optimieren. Das trainierte DNN wird direkt als gelernte Dynamik in eine nichtlineare modellprädiktive Regelung (MPC) mithilfe der quelloffenen Software `acados` integriert. Die numerische Formulierung beinhaltet die Integrierung von Regelgrößen in den Zustandsvektor und nachgiebiger Beschränkungen über Strafterme. In Motorversuchen mit dynamischen Lastreferenzen verbessert die MPC die IMEP-Regelung um 27.8% gegenüber einem werkseitigen dieselbasierten Regler des Motorherstellers, reduziert Rußemissionen um 61.1% und ermöglicht eine hohe Wasserstoffsubstitution. Dies ermöglicht eine potentielle CO_2-Reduktion von 39.7% unter Einhaltung aller Nebenbedingungen. Erwartungsgemäß steigen jedoch die NO_x-Emissionen deutlich an; der Mittelwert erhöht sich um 105.1%. Die vorgestellte *Raspberry Pi*-basierte Werkzeugkette ermöglicht eine durchgängige Entwicklung von Datengenerierung bis zum Prüfstandeinsatz in weniger als einem Tag und beschleunigt den Entwicklungsprozess gegenüber konventionellen Methoden erheblich. Zur Überwindung des Echtzeit-Rechenaufwands der MPC auf ressourcenbeschränkter Hardware wird ein Behavior Cloning (BC)-Ansatz umgesetzt. Die geklonte Regelstrategie repliziert erfolgreich das Verhalten der MPC auf dem Prüfstand, mit Inferenzzeiten unter 2 ms inklusive Kommunikationslatenz bei Einsatz am Motor. Eine Laufzeitmessung bestätigt zudem die Umsetzbarkeit der BC-Regelung auf ressourcenarmen Plattformen wie dem *ESP32*. Diese Arbeit präsentiert einen effizienten Entwicklungsprozess für fortschrittliche Regelstrategien von H2DF-Motoren. Der vorgestellte Ansatz ermöglicht sicheren, emissionsarmen und flexiblen Motorbetrieb und bietet eine praxisnahe Lösung zur Dekarbonisierung des MD- und HD-Verkehrs, in dem alternative Antriebstechnologien wie Elektrifizierung oder der großflächige Einsatz erneuerbarer Kraftstoffe noch vor Herausforderungen bei der breiten Umsetzung stehen.

Contents

List of Figures xiii

List of Tables xvii

List of Algorithms xix

List of Symbols xxi

List of Acronyms xxv

1 Introduction 1

1.1 Motivation . . . 1
1.2 State-of-the-Art & Research Gap . . . 3
1.3 Solution Approach & Methodology . . . 6
1.4 Scientific Contributions . . . 7
1.5 Data & Code Availability . . . 8

2 Fundamentals 11

2.1 Hydrogen Dual-Fuel Internal Combustion Engines . . . 11
2.2 Control Systems & Model Predictive Control . . . 16
2.3 Deep Learning . . . 19

3 Experimental Setup 29

3.1 Engine & Test Bench Specifications . . . 29
3.2 Engine Outputs: Data Acquisition & Emission Measurements . . . 31
3.3 Engine Controls . . . 32
3.4 Combustion Characteristics & Challenges . . . 33

4 Deep Neural Network-based Engine Model 37

4.1 Introduction to the Chapter . . . 37
4.2 Model Features & Network Architecture . . . 39
4.3 Data Collection . . . 41
4.4 Model Training & Validation . . . 44
4.5 Automated Machine Learning . . . 48
4.6 Summary of the Chapter & Discussion . . . 51

5 Deep Nonlinear Model Predictive Control 53

5.1 Introduction to the Chapter . . . 53
5.2 Control Objective & Baseline Strategy . . . 56
5.3 Nonlinear Model Predictive Control Setup . . . 58
5.4 On-Target MiL Simulation and Embedded Deployment Feasibility . . . 63

5.5 Experimental Results on the Test Bench 65
5.5.1 Core MPC Variant 66
5.5.2 Variations of Cost Function & Model 71
5.5.3 Robustness Investigation 76
5.6 Summary of the Chapter & Discussion 80

6 Behavior Cloning of Expert Model Predictive Controller 81
6.1 Introduction to the Chapter 81
6.2 Control Concept & Approach 83
6.3 Deep Neural Network Policy Modeling 85
6.4 Experimental Results on the Test Bench 88
6.5 Embedded Runtime Benchmarking on Constrained Hardware 96
6.6 Summary of the Chapter & Discussion 97

7 Conclusion 99

References 103

Appendix 121

List of Figures

1.1 Emission reduction contributions by technology in 2050. 1
1.2 Projected global hydrogen demand by production method in 2050 under different scenarios. 2
1.3 Various shares of MD and HD trucks in Alberta (AB), Canada (2018). 3
1.4 Graphical abstract of the developments in this thesis. 9

2.1 Generic closed-loop control system. 16
2.2 Principle of model predictive control. 18
2.3 Perceptron. 20
2.4 Visualization of a generic feedforward neural network. 21
2.5 Recurrent neuron. 22
2.6 Long short-term memory (LSTM) unit. 22
2.7 Gated recurrent unit (GRU). 23

3.1 Experimental setup of the modified *Cummins QSB* 4.5L diesel engine. 30
3.2 Injection timing diagram for cylinder 1. 32
3.3 Cylinder pressure around TDC of H2DF and diesel combustion at fixed p_{mi}. 33
3.4 Qualitative cylinder pressure trend around TDC of H2DF and diesel combustion with increasing hydrogen injection duration. 34
3.5 Cylinder pressure trace with injection timing events. 35

4.1 Development process of DNN-based engine model for H2DF combustion engine. . . 39
4.2 DNN engine model architecture. 41
4.3 Experimental PRBS generated engine inputs and outputs. 42
4.4 PRBS generated data distribution for p_{mi} 43
4.5 Loss-epoch plot DNN training. 45
4.6 DNN-based engine plant model. Predicted outputs vs. experimental unseen test dataset. 46
4.7 DNN-based engine plant model. Predicted outputs in time domain. 47
4.8 DNN-based engine plant model. GRU hidden state activation on extract of test data. 48
4.9 `Optuna` AutoML multi-objective optimization results 51

5.1 Development process of online MPC for H2DF. 55
5.2 Diesel standard controller experimental results 57
5.3 Computational graph of the DNN based engine model, using GRU and FC layer. . . 59
5.4 Block diagram of MiL simulation for early MPC validation. 64
5.5 Block diagram of MPC and engine test bench communication setup. 66
5.6 *Core MPC*. Experimental results load tracking. 67
5.7 *Core MPC*. Experimental results. Predictions extract including cost. 69
5.8 *Core MPC*. Experimental results. Controls change rates and constraints. 70

5.9 MPC *slow steps* variant. Experimental results load tracking. Controls change rates extract including constraints. 73
5.10 MPC *no feedback* variant. Experimental results load tracking. 75
5.11 DNN-based engine model. Sensitivity matrix with heat-map. 76
5.12 *Core MPC*. Experimental results, tracking under high noise injection. 77
5.13 *Core MPC*. Experimental results, tracking with model-plant mismatch (engine speed variation). 79

6.1 Development process of online BC controller for H2DF based on previously developed expert MPC. 83
6.2 DNN-based BC controller. DNN architecture. 85
6.3 DNN-based BC controller. Data generation ramp reference trajectory for p_{mi} load tracking on the test bench by MPC expert. 86
6.4 DNN-based BC controller. Experimental data, distribution to datasets for p_{mi}. . . . 87
6.5 DNN-based BC controller. DNN predicted outputs in time domain. 87
6.6 DNN-based BC controller. *Core MPC* benchmark experimental tracking results. . . 89
6.7 DNN-based BC controller. Experimental tracking results. 91
6.8 DNN-based BC controller *no feedback* variant. Experimental tracking results. 93
6.9 DNN-based BC controller. Extrapolation capability. Extract of experimental tracking results. 95
6.10 Profiling results (maximum execution time) on low-cost embedded controller *ESP32*. 96

A.1 Activation functions 121
A.2 PRBS generated data distribution for $\mathrm{NO_x}$. 127
A.3 PRBS generated data distribution for soot. 128
A.4 PRBS generated data distribution for dp_{max}. 128
A.5 MPC model without feedback. Loss-epoch plot DNN training 129
A.6 MPC model without feedback. Predicted DNN outputs vs. actual 130
A.7 MPC model without feedback. Predicted DNN outputs time domain 131
A.8 `Optuna` AutoML single-objective optimization results. 131
A.9 Time series split method of cross-validation approach for $k = 5$ folds. 132
A.10 *Core MPC*. Experimental tracking results. Predictions extract including constraints. 134
A.11 MPC *fast steps* variant. Experimental results load tracking. Controls change rates extract including constraints. 135
A.12 MPC *high H_2 utilization* variant. Experimental results load tracking. 137
A.13 MPC *low H_2 utilization* variant. Experimental results load tracking. 138
A.14 *Core MPC*. Experimental results, load tracking under model-plant mismatch – low engine speed. 140
A.15 *Core MPC*. Experimental results, load tracking under model-plant mismatch – high engine speed. 141
A.16 DNN-based BC controller. Experimental data, engine inputs and outputs. 142

A.17 DNN-based BC controller. Experimental data, total data distribution for DNN outputs. 143
A.18 DNN-based BC controller. Loss-epoch plot of DNN training. 143
A.19 DNN-based BC controller. DNN predicted outputs vs. true data. 144
A.20 DNN-based BC controller *no feedback* variant. Loss-epoch plot of DNN training. . . 144
A.21 DNN-based BC controller *no feedback* variant. DNN predicted outputs vs. true data. 145
A.22 DNN-based BC controller *no feedback* variant. DNN predicted outputs in time domain. 146
A.23 Linear correlation of FLOPs and learnable parameters of multiple DNNs. 146

List of Tables

1.1 Overview of H2DF experimental studies (2015–2025) for CI engines with PFI of hydrogen. . . . 4

2.1 Comparison of hydrogen and diesel as fuels. . . . 12

3.1 Measured engine output variables for cylinder 1. . . . 31
3.2 Engine control input variables for cylinder 1. . . . 32

4.1 DNN-based engine plant model. Prediction performance on validation and unseen test datasets. . . . 45
4.2 AutoML study. Definition of search space for DNN engine model architectures. . . . 49

5.1 Diesel standard controller experimental results performance metrics. . . . 58
5.2 MPC constraint values (physical). . . . 63
5.3 *Core MPC*. Experimental results tracking performance metrics. . . . 68
5.4 Experimental results comparison of varying H_2 utilization settings of the *Core MPC*. 71
5.5 *no-feedback* vs. *Core MPC*: experimental comparison. . . . 74

6.1 DNN-based BC controller. DNN prediction performance metrics. . . . 88
6.2 Comparison of experimental results for BC and MPC controllers. . . . 94

A.1 *Cummins QSB* 4.5 Tier 3 engine parameter, as tested . . . 124
A.2 Rapid prototyping ECU Specifications . . . 125
A.3 Specifications of *PPS-M* emissions measurement system. . . . 125
A.4 *ECM NOxCANt* module specifications. . . . 126
A.5 Properties of cylinder pressure sensors and amplifiers. . . . 126
A.6 Input bounds for PRBS . . . 127
A.7 DNN-based engine plant model. Training hyperparameter settings. . . . 127
A.8 DNN (*no feedback* variant) prediction performance on validation and unseen test datasets. . . . 129
A.9 DNN architecture comparison . . . 129
A.10 *Core MPC*. `acados` solver settings. . . . 133
A.11 Experimental results comparison of varying controller aggressiveness settings of the *Core MPC* . . . 133
A.12 Experimental results comparison of *Core MPC* for varying engine speeds. . . . 139
A.13 Robust bounds of *Core MPC* to various disturbances in MiL simulation. . . . 139
A.14 DNN-based BC controller. DNN training hyperparameter settings. . . . 142
A.15 DNN-based BC controller *no feedback* variant. DNN prediction performance. . . . 143
A.16 *ESP32* (Espressif Systems' 32-bit microcontroller) microcontroller specifications. . . . 145

List of Algorithms

A.1 Pseudo-code. Training procedure for FFNN and RNN using SGD. 122
A.2 Pseudo-code. BC of an expert control policy. 122
A.3 Pseudo-code. Automated machine learning (AutoML) for DNNs. 123
A.4 Local sensitivity analysis of a DNN-based engine model. [119] 136

List of Symbols

$\oplus$	Element-wise addition
$\otimes$	Element-wise multiplication

Greek Symbols

α	Crank angle
α_{H2}	Start of hydrogen injection
α_{Pre}	Start of diesel pre-injection
α_{Main}	Start angle of main diesel injection
Δ	Difference operator (rate of change)
π_θ	Learned policy parameterized by θ
$\pi_{\theta,\mathrm{BC}}$	Behavioral Cloning policy mapping states to actions
π_{MPC}	Model predictive control policy
σ	Sigmoid function
σ	Standard deviation
θ	Trainable parameter set

Latin Symbols

$\mathcal{A}$	Behavioral Cloning action space
a	Control action predicted by BC policy
a	Activation function
a_i	Expert action corresponding to the ith demonstration
$\mathcal{A}_{\mathrm{noFb}}$	No-feedback BC action space (same as $\mathcal{A}$)
b	Bias term
c	Cell state
c_{NOx}	Nitrogen oxide concentration
c_{PM}	Particulate matter concentration
$\mathcal{D}$	Dataset of expert demonstrations
m_{D}	Mass of diesel fuel in kilograms
d	Disturbance
dp_{max}	Maximum pressure rise rate
e	Control deviation or error

f	Nonlinear state transition function
f	Forget gate
$F_{s,\tilde{x}}$	Slack variable selection matrix
$\tilde{f}$	Augmented nonlinear state transition function for MPC
$F_{\tilde{u}}$	Input constraint selection matrix
$F_{\tilde{x}}$	State constraint selection matrix
g	Candidate cell input
g	Equality constraint function
$\tilde{g}$	Augmented nonlinear output mapping function for MPC
h	Nonlinear output mapping function
$\hat{y}_i$	Predicted value for the ith sample
h	Hidden state
h	Inequality constraint function
m_{H2}	Mass of hydrogen fuel in kilograms
i	Input gate
J	Cost (stage and terminal)
J_{l}	Stage cost (Lagrange term)
J_{m}	Terminal cost (Mayer term)
k	Discrete time step index
k	Number of folds in cross-validation
$\mathcal{L}$	Loss function
$\mathrm{LHV_D}$	Lower heating value for diesel
$\mathrm{LHV_{H2}}$	Lower heating value for hydrogen
M	Number of outputs in a fully connected layer
N	Common horizon length
n_a	Dimension of the Behavioral Cloning action space
$n_{a,\mathrm{noFb}}$	Dimension of the no-feedback BC action space
N	Number of expert demonstrations
n_{Eng}	Engine speed in revolutions per minute
n_{GRU}	Number of units in the gated recurrent unit layer
N	Number of inputs in a fully connected layer
N_{p}	Prediction horizon length
N_{Param}	Total number of learnable parameters in a layer or network

n_s	Dimension of the Behavioral Cloning state space
$n_{s,\text{noFb}}$	Dimension of the no-feedback BC state space
N_u	Control horizon length
n_u	Dimension of the action space
n	Number of samples
n_x	Dimension of the state space
o	Output gate
p	Number of inputs
p_cyl	In-cylinder pressure
p_max	Maximum pressure
p_mi	Indicated mean effective pressure
$p_\text{mi,ref}$	Reference indicated mean effective pressure
p_Rail	Diesel rail fuel pressure
Q	State weighting matrix in the MPC cost function
q	Stage cost weighting scalar
R	Control weighting matrix in the MPC cost function
r	Control cost weighting scalar
r	Desired reference value
r	Reset gate
$\mathcal{S}$	Behavioral Cloning state space
s	Observed system state for BC
s	Slack variable vector combining lower and upper bound slacks
$\mathcal{S}_\text{noFb}$	Input state space for BC *no feedback* variant without feedback
s_i	Observed state of the ith demonstration
$s_{\text{max},\tilde{x}}$	Upper slack variable for state constraints
$s_{\text{min},\tilde{x}}$	Lower slack variable for state constraints
tanh	Hyperbolic tangent function
t_calc	Computation time available per cycle
$t_\text{comp,max}$	Maximum computation time per cycle
t_H2	Hydrogen duration of injection
t_Main	Main diesel duration of injection
t_P2M	Time between pre injection and main injection
t_Pre	Duration of pre-injection

$\mathcal{U}$	Action (control) space of the system
u	Controller actions (outputs)
$u_{\alpha,\mathrm{Main}}$	Start angle of main diesel injection (control input variable)
$u_{\mathrm{t,H2}}$	Hydrogen duration of injection (control input variable)
$\tilde{u}$	MPC input vector
$u_{\mathrm{t,Main}}$	Main diesel duration of injection (control input variable)
$u_{\mathrm{t,P2M}}$	Time between pre and main injection (control input variable)
V_{d}	Cylinder displacement volume
W_{i}	Indicated work done over one cycle
w	Weight associated with the ith input
$\mathcal{X}$	State space of the system
x	State vector
x_0	Initial state at timestep $k = 0$
X_{DNN}	Input features of a deep neural network
$\tilde{x}$	MPC state vector
y	Controller inputs or system outputs
$y_{\mathrm{c,NOx}}$	Concentration of nitrogen oxides (control output variable)
$y_{\mathrm{c,PM}}$	Concentration of particulate matter (control output variable)
Y_{DNN}	Output features of a deep neural network
$y_{\mathrm{dp,Max}}$	Maximum pressure rise rate (control output variable)
y_i	True value of the ith sample
$y_{\max}$	Maximum value of the dataset
$y_{\min}$	Minimum value of the dataset
$y_{\mathrm{p,mi}}$	Indicated mean effective pressure (control output variable)
$\tilde{y}$	MPC output vector
z	Weighted input sum to the activation function
z	Update gate

Notation Conventions: In this thesis, vectors are not highlighted in bold font and are denoted as $x(k)$ with k indicating the discrete time index; subscripts (x_k) are reserved for prediction horizon stages or optimization variables, following the convention established in Borrelli et al. [1]. Unless otherwise stated, all plots in this thesis represent theoretically discrete data, which are visualized as piecewise constant or continuous curves to improve readability.

List of Acronyms

AB	Alberta, Canada
Adam	Adaptive Moment Estimation
AF	Activation Function
AI	Artificial Intelligence
ANN	Artificial Neural Network
aTDC	After Top Dead Center
AutoML	Automated Machine Learning
BC	Behavior Cloning
BO	Bayesian Optimization
BO-TPE	Bayesian Optimization with Tree-structured Parzen Estimator
bTDC	Before Top Dead Center
C2C	Cycle-to-Cycle
CAD	Crank Angle Degree
CAN	Controller Area Network
CI	Compression Ignition
CNG	Compressed Natural Gas
CO	Carbon Monoxide
CO_2	Carbon Dioxide
DES	Diesel Energy Share
DI	Direct Injection
DL	Deep Learning
DNN	Deep Neural Network
DOI	Duration of Injection
dyno	Dynamometer
ECU	Engine Control Unit
EGR	Exhaust Gas Recirculation
FC	Fully Connected
FFNN	Feedforward Neural Network
FLOPs	Floating Point Operations
FPGA	Field Programmable Gate Array

GE	Generalization Error
GP	Gaussian Process
GRU	Gated Recurrent Unit
H2DF	Hydrogen Diesel Dual-Fuel
H2-ICE	Hydrogen Internal Combustion Engines
HC	Hydrocarbons
HCCI	Homogeneous Charge Compression Ignition
HD	Heavy-Duty
HES	Hydrogen Energy Share
HPC	High-Performance Computing
HPIPM	High Performance Interior Point Method
HPO	Hyperparameter Optimization
ICE	Internal Combustion Engine
IL	Imitation Learning
IMEP	Indicated Mean Effective Pressure
IRENA	International Renewable Energy Agency
LHV	Lower Heating Value
LSTM	Long Short-Term Memory
LUT	Look-Up Table
LX	Inline X-cylinder
MABX	dSPACE MicroAutoBox 2
MAE	Mean Absolute Error
MD	Medium-Duty
MiL	Model-in-the-Loop
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MPC	Model Predictive Control
MPRR	Maximum Pressure Rise Rate
MSE	Mean Squared Error
NAS	Neural Architecture Search
NLP	Nonlinear Programming Problem
NMAE	Normalized Mean Absolute Error
$\text{NMAE}_{\text{clipped}}$	Normalized Mean Absolute Error (clipped range)

NRMSE	Normalized Root Mean Square Error
$\text{NRMSE}_{\text{clipped}}$	Normalized Root Mean Square Error (clipped range)
OCP	Optimal Control Problem
P2M	Pre-to-Main Time
PFI	Port Fuel Injection
PI	Proportional-Integral
PM	Particulate Matter
ppm	parts per million
PRBS	Pseudo-Random Binary Sequence
QP	Quadratic Programming
RCP	Rapid Control Prototyping
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RPi	Raspberry Pi 400
RTI	Real-Time Iteration
RWTH	Rheinisch-Westfälische Technische Hochschule
SGD	Stochastic Gradient Descent
SI	Spark Ignition
SISO	Single-Input Single-Output
SOI	Start of Injection
SQP	Sequential Quadratic Programming
TDC	Top Dead Center
TPE	Tree-structured Parzen Estimator
UDP	User Datagram Protocol
UofA	University of Alberta
VKT	Vehicle-Kilometers Traveled

1 Introduction

1.1 Motivation

The continuous increase of atmospheric carbon dioxide (CO_2) concentrations, coupled with rising levels of harmful local pollutants, presents significant risks to global health and environmental stability [2], [3]. Human-driven activities, particularly in the transportation sector, substantially contribute to these environmental challenges, accounting for approximately 23% of global CO_2 emissions as of 2022 [4], [5]. Road transport alone accounts for about 69% of transport-related emissions, underscoring its critical role in current climate impacts and future mitigation strategies, with internal combustion engines (ICEs) responsible for the majority of this total [3].

Although electrification has gained traction in passenger vehicle markets, this technology remains rather unsuitable for logistics and freight transport due to current battery limitations, infrastructure constraints and high operational and initial cost [6], [7]. Thus, alternative solutions such as hydrogen propulsion technologies have been identified as significant contributors toward achieving net-zero emissions targets by 2050 [8], [9].

Fig. 1.1 shows the projected contributions of six key technological avenues to global CO_2 emission reductions by 2050, as published in the world energy transitions outlook 2024 by IRENA (International Renewable Energy Agency) [8]. Hydrogen is expected to contribute approximately 10% of the required reduction, highlighting its strategic role alongside renewables, energy efficiency, and electrification.

Additionally, Fig. 1.2 shows the projected global hydrogen demand as part of McKinsey's global energy perspective report (2023) [9]. It becomes clear the worldwide hydrogen demand in total is expected to increase by minimum 200%. The net zero scenario for 2050 anticipates a total demand of 585 Mt/a, entirely met by low-emission hydrogen (green and blue)[1], an increase of 550% compared to today. In contrast, the current trajectory scenario for 2050 projects a demand of 307 Mt/a, with a significant portion still supplied by gray hydrogen. This highlights

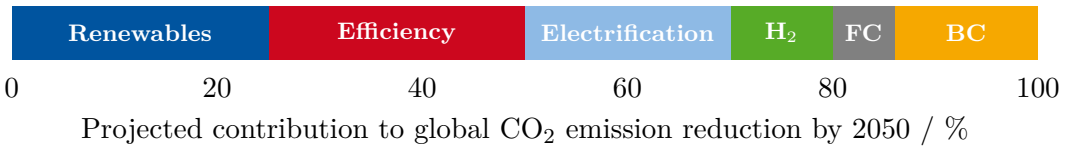


Figure 1.1: Projected contributions (2024) of six key technological avenues to global CO_2 emission reductions by 2050 (total mitigation 36.9 Gt) [8]. FC: fossil fuels with carbon capture and storage, BC: bioenergy with carbon capture and storage.

[1] The colour code refers to the production pathway: green hydrogen is produced by water electrolysis using renewable electricity, blue hydrogen from fossil fuels combined with carbon capture, utilization and storage, and gray hydrogen from fossil fuels – mainly steam methane reforming – without carbon capture [10].

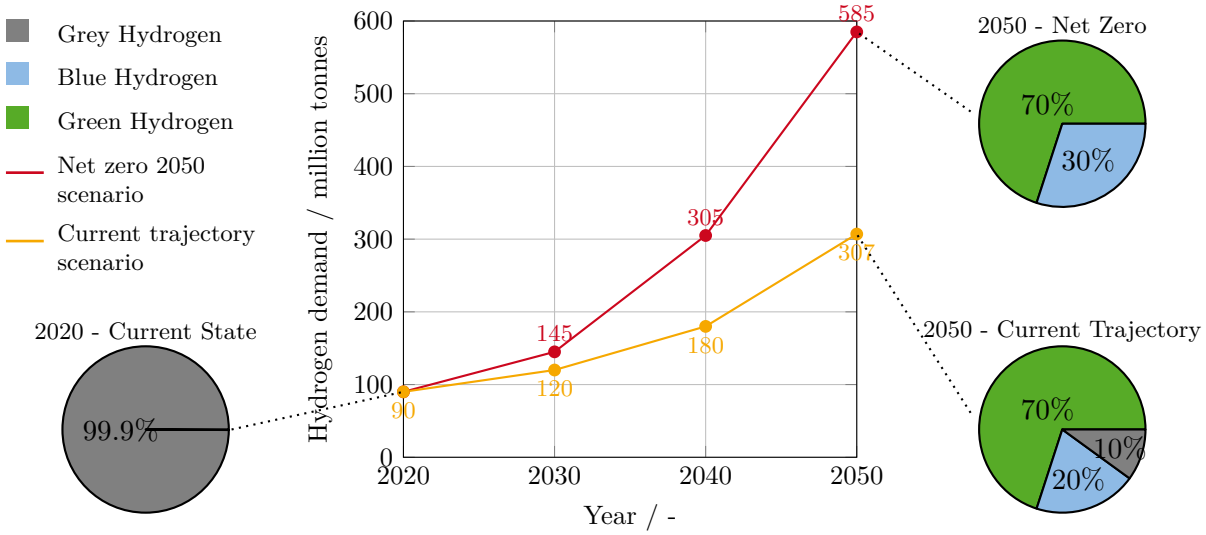


Figure 1.2: Projected global hydrogen demand by 2050, segmented by production method, under two scenarios from McKinsey's global energy perspective report (2023). [9]

the importance of not only expanding the use of hydrogen-driven technologies but also ensuring that hydrogen production itself becomes environmentally sustainable. Alarmingly, the vast majority of hydrogen produced today (over 99.9%) is gray hydrogen, generated from fossil sources without carbon capture, offering no enhanced climate benefit. [9]

These projections underscore the strategic importance of hydrogen and hydrogen-powered propulsion systems in achieving global net-zero targets by 2050, as emphasized in these large-scale international studies [8], [9]. To realize this potential, a parallel effort is required: scaling up green hydrogen production while advancing its end-use applications. While both aspects of the hydrogen value chain are critical, this thesis focuses on the latter – specifically, the development and control of hydrogen-powered drivetrains for heavy-duty (HD) transport.

However, the adoption of fuel cell vehicles and hydrogen-ICE (H2-ICE) is currently impeded by their high (initial) costs, technological immaturity, and the need for extensive infrastructure development [11], [12]. In contrast to hydrogen fuel cells, however, H2-ICE technology benefits from lower hydrogen purity requirements, which reduces fuel production costs [13], [14].

Specifically in regions such as Alberta (AB), Canada, medium-duty (MD) and HD vehicles disproportionately contribute to transport emissions due to their high utilization rates and longer service life [15]. Fig. 1.3 illustrates that although HD trucks accounted for only 12% of combined MD and HD vehicle sales and 20% of the registered fleet in Alberta (2018), they consumed 55% of the fuel, making them the dominant source of carbon dioxide emissions in this sector [15]. This disproportionate impact underscores the urgent opportunity to target HD drivetrains for emissions reduction. Given the high cost and long turnover cycles associated with replacing these vehicles with zero-emission alternatives, there is a pressing need for transitional solutions that can deliver rapid and scalable emission reductions.

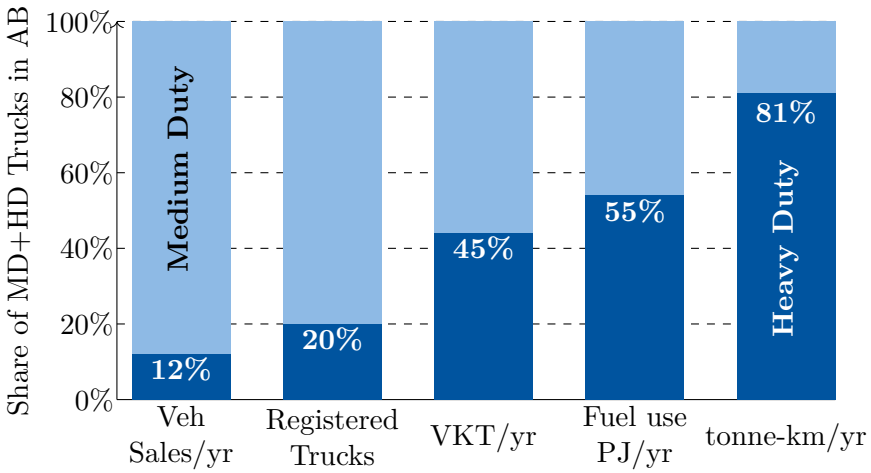


Figure 1.3: Various shares of MD and HD trucks in Alberta (AB), Canada (2018). MD trucks (3.9 t to 15 t) in light blue, HD trucks (> 15 t) in dark blue. VKT: vehicle-kilometers traveled, yr: year. [15]

One of such promising short- to medium-term alternatives is the retrofitting of existing diesel compression-ignition (CI) engines in HD trucks to operate as hydrogen–diesel dual-fuel (H2DF) systems. This approach enables significant reductions in CO_2 emissions by using hydrogen injected to the air intake port, substantially lowering diesel consumption while leveraging existing engine infrastructure with minimal modifications [16], [17], [18]. Such retrofits not only offer immediate environmental benefits by utilizing blue or preferably green hydrogen but also provide an economically feasible pathway toward emissions reduction in the crucial MD and HD transport sectors [15], [19]. This is particularly relevant for HD engines in applications such as large mining trucks, which are very costly and designed for long operational lifespans, making immediate replacement impractical. The multi-fuel capability of H2DF retrofits ensures operability even without a dedicated hydrogen supply, offering energy security and independence from emerging infrastructure – especially valuable in remote regions. [19], [20], [21], [22]

1.2 State-of-the-Art & Research Gap

H2DF engines have emerged as a promising approach for reducing carbon emissions in HD applications. Over the past decade, several experimental studies have evaluated the impact of hydrogen addition on engine performance, emissions, and control. Despite significant gains in CO_2 and soot reduction, challenges remain in terms of NO_x emissions and the need for sophisticated control strategies, especially under transient conditions.

Several experimental studies have investigated H2DF operation in CI engines using port fuel injection (PFI) of hydrogen, as summarized in Table 1.1.

Table 1.1: Overview of H2DF experimental studies (2015–2025) for CI engines with PFI of hydrogen. IMEP: indicated mean effective pressure, LX: inline X-cylinder.

Study	**Engine type & operating conditions**	**Control method**	**Key findings**
Suzuki et al. [23] (2015)	5.2L L4 HD; steady-state (1500 min^{-1}, 4.2 bar, 7.5 bar and 10.5 bar IMEP)	Up to 29% EGR; Unmodified ECU	HES: up to 80%. ↓CO_2, ↑NO_x (no EGR), ↑soot (with EGR)
Jhang et al. [24] (2016)	5.9L L6 HD; steady-state (800–1840 min^{-1}, 25%–75% load); onboard electrolysis	Unmodified ECU	HES: ~5%. ↓CO_2/CO, ↑NO_x at high load, ↓NO_x at low load & idle
Tsujimura and Suzuki [13] (2017)	1.3L single-cyl.; steady-state (600–1500 min^{-1}, 3–9 bar IMEP)	Supercharger, no EGR; manual SOI control	HES: ~50%. ↓CO_2/PM, ↑NO_x
Dimitriou et al. [25] (2018)	5.2L L4 HD; steady-state (1500 min^{-1}, 20 & 40 kW load)	EGR + SOI timing (manual tuning)	HES: 85%@40 kW. ↓CO_2/PM, ↑NO_x (mitigated by EGR)
Guan et al. [17] (2024)	14.8L HD truck; dyno (1000–1400 min^{-1}, 10%–75%% load)	Commercial retrofit; unmodified ECU	HES: up to 30%. ↓CO_2, ↓NO_x @med.–high load, ↑NO_x/CO/soot @low load (due to O_2 displacement)
Wang et al. [26] (2023)	3.6L L4; JE05 cycle & const. speeds (40–80 km/h); dyno	Unmodified ECU	HES: up to 22%. ↓CO_2/PM/CO, ↑NO_x, ↓transient efficiency (50% HES)
Liu et al. [27] (2023)	3.0L L4 turbo; steady-state (1800 min^{-1}, 30–90% load)	15% EGR + SOI retard	HES: 60% @30% load, 15% @90% load. ↓CO_2 35%, ↓NO_x 63% with EGR, ↑NO_x 20%% @90% load

Suzuki et al. [23] (2015) examined a 5.2L HD engine at steady-state, achieving up to 80% hydrogen energy share (HES) and significantly reducing CO_2 emissions. However, NO_x increased without exhaust gas recirculation (EGR), and soot rose when EGR (up to 29%) was applied. No changes were made to the engine control unit (ECU). Similarly, Jhang et al. [24] (2016) evaluated a 5.9L engine with onboard hydrogen from electrolysis, reaching ∼5% HES. They reported CO_2 and carbon monoxide (CO) reductions, while NO_x increased at high loads and decreased at idle and low loads. Again, no active control strategy was implemented. In contrast, Tsujimura and Suzuki [13] (2017) conducted fundamental studies on a single-cylinder 1.3L engine using a supercharger and manual start-of-injection (SOI) control. At ∼50% HES, reductions in CO_2 and particulate matter (PM) were observed, but NO_x emissions increased, and no EGR or

adaptive control was used. Dimitriou et al. [25] (2018) tested a 5.2L multi-cylinder engine at 20 and 40 kW loads, achieving up to 85% HES. Emissions of CO_2 and PM were reduced, and NO_x increases were effectively mitigated through manually tuned EGR and retarded SOI, though transient behavior was not explored. In real-world scenarios, Guan et al. [17] (2024) tested a 14.8L HD engine on a dynamometer (dyno) with up to 30% HES. CO_2 and NO_x decreased at medium to high loads, but low-load operation led to higher NO_x, CO, and soot due to oxygen displacement and the absence of coordinated air-path control; the ECU remained unmodified. Likewise, Wang et al. [26] (2023) assessed a 3.6L engine on a chassis dyno during transient driving cycles. With up to 22% HES and no ECU changes, they observed reductions in CO_2, PM, and CO, but a significant NO_x increase and only partial utilization of the hydrogen energy input, underlining the need for better injection and EGR strategies. Lastly, Liu et al. [27] (2023) conducted steady-state tests on a 3.0L turbocharged engine, varying HES from 15% to 60%. 15% EGR and SOI retardation reduced emissions at low load, but at 90% load, NO_x rose by 20%, underscoring the need for load-dependent advanced control.

Across all studies, CO_2 and particulate emissions have been shown to decrease nearly proportionally with increasing hydrogen substitution. However, controlling NO_x formation remains a key challenge, particularly under medium to high loads and during transient operation. Most existing research focuses on steady-state conditions, with few studies addressing dynamic transients or actual reference tracking performance in closed- or open-loop conditions. Furthermore, these control systems are typically based on stock ECUs that rely on feedforward look-up tables (LUTs), proportional-integral (PI) controllers, or static injection maps [28]. While these methods have been adequate for basic operation, they offer limited robustness and adaptability, often resulting in sub-optimal performance under real-world uncertainties. Additionally, significant test bench calibration time utilizing experienced engineers is necessary to acquire the data, slowing the overall control development process [29].

Despite the promising environmental benefits of H2DF combustion, current literature reveals a lack of advanced control strategies capable of handling its unique challenges, particularly the elevated NO_x emissions and highly nonlinear combustion dynamics. Effective control requires migrating from conventional single-fuel strategies to new dual-fuel approaches while accounting for rapid hydrogen combustion and system constraints like the maximum cylinder pressure and pressure rise rates. To unlock the full potential of H2DF technology, there is a pressing need for real-time capable control and modeling algorithms that can operate on low-cost embedded hardware.

This context motivates the two central research questions explored in this thesis:

1. **How can advanced control systems be designed to enable low-emission and safe operation of H2DF engines under real-world conditions?**
   This question addresses the challenge of controlling H2DF combustion systems, which are inherently nonlinear, highly dynamic, and subject to safety-critical constraints. Ensuring

both low emissions and safe operation under real-world variability requires control strategies that can handle transient behavior, meet strict physical limits, and adapt to varying engine conditions. The broader context includes the urgent need to reduce CO_2 emissions using scalable technologies that can be deployed immediately through retrofitting, without relying on long-term infrastructure development.

2. **How can advanced control system development be accelerated to enable sustainable powertrain innovation on low-cost, readily available embedded hardware?**
   This question arises from the increasing demand for rapid control development in an industry facing cost pressure, shorter innovation cycles, and a shortage of skilled engineers. At the same time, there is a growing need to implement advanced control algorithms on affordable embedded platforms. The key challenge is to reduce development effort while ensuring real-time capability on production-grade hardware, enabling the near-term adoption of sustainable powertrain technologies.

1.3 Solution Approach & Methodology

To address the two central research questions and close the identified research gap, this thesis adopts a structured methodology that combines experimental analysis, machine learning (ML) enabled modeling, and advanced control strategies, systematically developed and presented across the subsequent chapters. The graphical abstract in Fig. 1.4 provides a high-level overview of the methodology and development process for a safe and efficient controller for a H2DF engine on low-cost embedded systems.

The development process begins with the experimental setup (Chapter 3), where the fast, nonlinear dynamics of the H2DF combustion process are studied, and sensor configurations and output signals are defined. Based on this setup, a comprehensive dataset is generated and used to train a deep neural network (DNN) model (Chapter 4). DNNs are powerful function approximators capable of capturing the complex, nonlinear behavior of the H2DF engine. In this work, the DNN serves as a data-driven surrogate model for subsequent control design, to simplify and accelerate the system identification process. To reduce manual tuning efforts, automated ML (AutoML) techniques (Section 4.5) are applied to improve model accuracy and computational efficiency.

This DNN model is then integrated into a model predictive control (MPC) framework (Chapter 5). MPC is a model-based advanced control technique solving a finite-horizon optimal control problem (OCP) at each time step, taking system constraints into account to compute optimal control inputs in a receding horizon manner. It is particularly suitable for multi-variable systems with nonlinear dynamics and operational limits, as present in H2DF combustion. The open-source solver `acados` [30] is used for online optimization, and the resulting controller is

manually tuned and evaluated to identify the best-performing configuration. Robustness analyses (Section 5.5.3) are carried out to test performance under noise and uncertainties, while configuration studies explore controller variants for specific use cases.

Despite its promising performance, the MPC's computational demands exceed the capabilities of typical low-cost embedded systems. To bridge this gap, a behavior cloning (BC) approach is applied (Chapter 6). BC is a subset of imitation learning (IL) and a supervised learning method that trains a DNN to reproduce expert behavior – here, the control actions generated by the MPC. This results in a fast, inference-only policy model that preserves the expert controller's behavior while significantly reducing real-time computational costs. The final BC-based controller is deployed on embedded hardware and validated experimentally, including an additional deployment of a feedback-independent variant. Additionally, multiple DNN architectures are profiled for their runtime performance and suitability on embedded platforms (Section 6.5).

This workflow aims to demonstrate that advanced nonlinear and data-driven control strategies can be successfully implemented in resource-constrained environments, enabling the rapid development of a safe and adaptable prototype controller suited to diverse real-world constraints. Each of the three main chapters (Chapter 4 on DNN-based engine modeling, Chapter 5 on real-time MPC implementation, and Chapter 6 on data-driven control via BC) includes a dedicated literature review to position the contribution of each chapter within the current state of the art. Furthermore, every chapter concludes with a summary and discussion of the results. This structure supports a clear presentation of the individual contributions while also highlighting their interconnection within the overarching goal of this thesis.

1.4 Scientific Contributions

Parts of this thesis have been published or are currently under review. Where applicable, direct links to the corresponding publications are indicated in the introduction of the respective chapters. Corresponding publications authored by the candidate, often in collaboration with the teams at RWTH Aachen University, University of Alberta (UofA) and University of Freiburg, are [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48]. A complete list of the author's publications can be found on the author's profiles on ResearchGate and ORCiD.

The scientific contributions of this work are as follows:

- Developed a DNN model using gated recurrent unit (GRU) layers to capture the nonlinear dynamics of H2DF combustion. [43], [47]
- Applied AutoML to optimize the trade-off between prediction accuracy and computational efficiency, running 27,000 trials on high performance computing (HPC) resources. [46]
- Designed and implemented a nonlinear MPC framework leveraging the trained DNN model, formulated as an OCP with augmented controls and soft constraints, solved using `acados`. [32], [33], [38], [47]

- Demonstrated the controller's ability to safely track unseen dynamic load profiles while balancing emissions, fuel flexibility, and mechanical constraints. [32], [35], [42], [47]
- Enabled real-time execution and rapid prototyping by deploying MPC on a *Raspberry Pi*, streamlining the pipeline from data generation to embedded control without altering main engine software. [33], [47]
- Presented a BC approach to imitate expert MPC behavior, achieving a 250% speedup over the MPC on the test bench, and demonstrated real-time feasibility on the low-cost embedded system *ESP32* via embedded profiling, with all profiled networks below 6 ms per engine cycle (18 ms budget). [34], [48]
- Demonstrated the potential of the H2DF retrofit approach to enable immediate carbon emission reductions using existing diesel engine platforms. [42], [43], [45], [47]

1.5 Data & Code Availability

In the interest of transparency and reproducibility, several datasets and code repositories associated with this thesis have been made publicly available:

- Main engine experimental dataset and DNN training scripts: `doi.org/10.5281/zenodo.16902579`, [49].
- Nonlinear MPC implementation: `doi.org/10.5281/zenodo.16902939`, [50].
- AutoML optimization scripts: `doi.org/10.5281/zenodo.16903152`, [51].
- BC controller code and experimental MPC expert demonstration dataset: `doi.org/10.5281/zenodo.16902947`, [52].

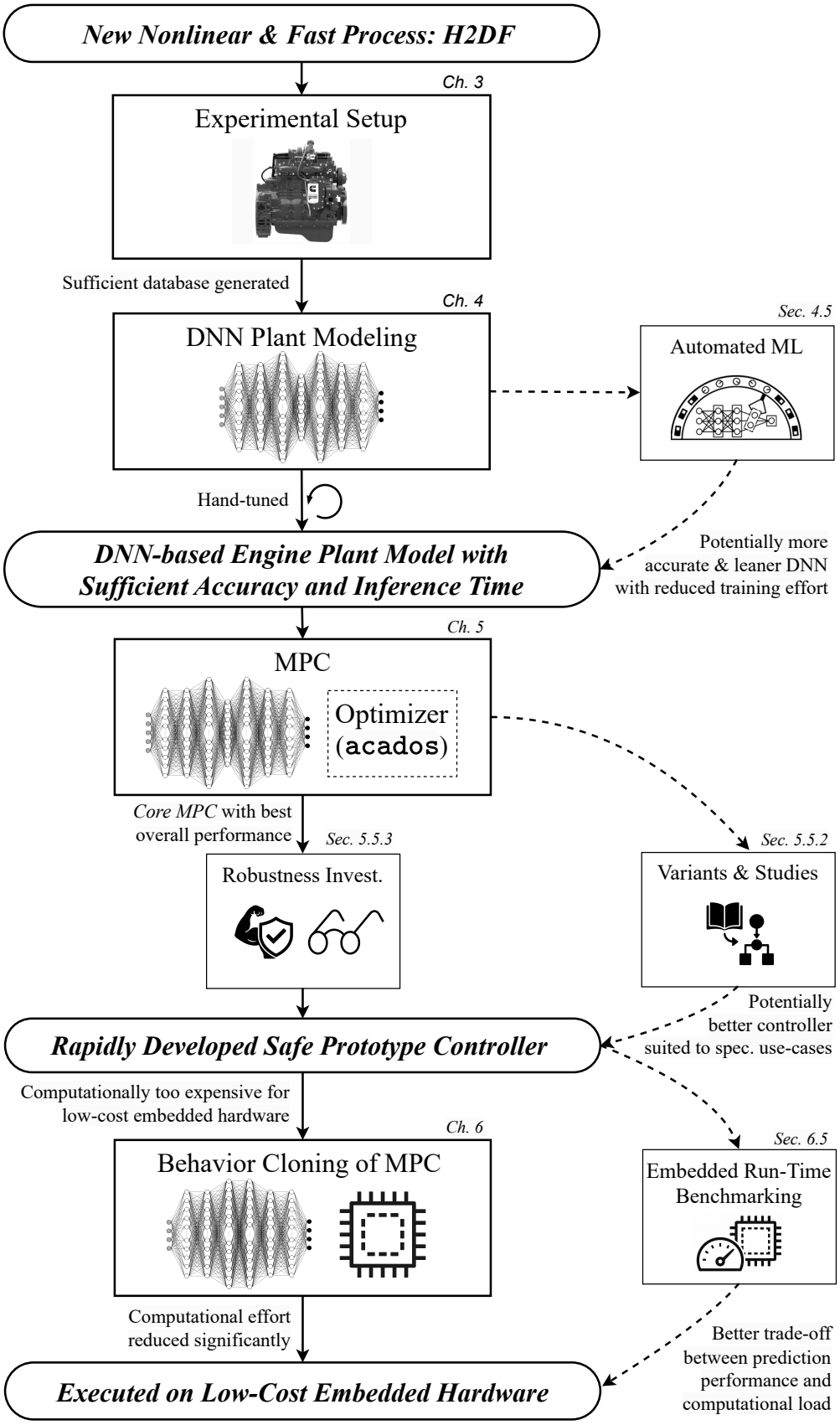


Figure 1.4: Graphical Abstract of the developments in this thesis. AutoML logo taken with permission from [31].

2 Fundamentals

This chapter provides the necessary background to understand the work presented in this thesis. Section 2.1 covers the fundamentals of hydrogen diesel dual-fuel (H2DF) combustion and compares hydrogen and diesel fuels. Section 2.2 outlines control system basics and model predictive control (MPC). Section 2.3 introduces the key deep learning (DL) concepts used in this thesis.

2.1 Hydrogen Dual-Fuel Internal Combustion Engines

Hydrogen and Diesel as Fuel

Hydrogen is a clean, high-energy-density fuel that can be used in internal combustion engines (ICEs) and fuel cells. It has a wide flammability range, enabling lean combustion, which allows to enhance efficiency and reduce NO_x emissions [16]. Hydrogen has a high auto-ignition temperature (585 °C) and a fast flame speed, leading to improved combustion characteristics. However, storage and distribution challenges arise due to its low volumetric energy density, requiring high-pressure tanks or cryogenic storage [53]. Additionally, hydrogen embrittlement can degrade engine materials, and pre-ignition can occur due to its low ignition energy [54].

Diesel is a high-energy-density fossil fuel widely used in compression ignition (CI) engines due to its high cetane number, which ensures efficient auto-ignition and combustion [55]. It has a high volumetric energy content and is liquid at atmospheric pressure, making it ideal for HD applications. Diesel combustion, however, generates particulate matter (PM) and NO_x emissions, requiring after-treatment systems like diesel particulate filters and selective catalytic reduction for emission control [56]. While diesel engines offer high torque and fuel efficiency, they contribute significantly to carbon emissions and air pollution [57].

Table 2.1 shows the fuel characteristics of hydrogen and diesel in a comparison including information about storage and sourcing of the fuels. Hydrogen offers strong theoretical advantages as a fuel, including high gravimetric energy density, zero CO_2 emissions at the point of use and broad applicability across spark-ignition engines, dual-fuel CI concepts and fuel-cell systems [16], [54]. In practice, however, its deployment is constrained far more by sourcing and infrastructure than by combustion or fuel-cell technology, as outlined in Chapter 1.

Global hydrogen production is still overwhelmingly fossil based, with less than 1% classified as low-emission hydrogen [8]. The climate benefit of using hydrogen in a vehicle therefore depends almost entirely on the production pathway: fossil hydrogen largely shifts emissions upstream, while green hydrogen competes with direct electrification for scarce renewable electricity [63].

Storage and distribution further amplify these constraints. Hydrogen's volumetric energy density is extremely low, so storing it requires either high pressure (350 bar to 700 bar) or liquefaction at cryogenic temperatures, both with high energy penalties and heavy composite tanks [60].

Table 2.1: Comparison of hydrogen and diesel as fuels. [16], [53], [54], [55], [58], [59], [60], [61], [62]

Fuel property	**Hydrogen**	**Diesel**
Lower heating value (LHV)	120 MJ/kg (high)	42.5 MJ/kg (low)
Volumetric energy density	1.8 MJ/L @ 200 bar, 4.8 MJ/L @ 700 bar (low)	35.87 MJ/L (high)
Ignition temperature	585 °C (high, reduces knock tendency)	210 °C (lower, easier ignition)
Combustion type	SI or CI (dual-fuel)	CI
Engine-out emissions	Zero CO_2, potential for high NO_x	High CO_2, NO_x, and PM emissions
Flame speed	High (faster combustion)	Lower (slower combustion)
Storage technology	Compressed high-pressure gas or cryogenic liquid hydrogen, high energy demand for compression or liquefaction	Liquid fuel at ambient conditions, low conversion energy, low handling and safety risk
Tank characteristics	Heavy and large composite pressure vessels, high cost, sensitive to impact and fatigue	Low cost metal or plastic tanks, high structural robustness, well understood failure modes
Sourcing	Produced via renewable-powered electrolysis (green), natural gas reforming (grey) with carbon capture (blue), cost and carbon intensity strongly depend on electricity source	Mature global extraction and refining chain, stable cost structure and supply, high direct carbon intensity when derived from non-biogenic sources

Diesel, in contrast, is a dense ambient-temperature liquid with a mature, globally available logistics chain. The infrastructure reflects this disparity: by the end of 2024, only about 1 160 hydrogen refueling stations were in operation worldwide, far below what would be required for widespread adoption [64].

From a propulsion-technology perspective, hydrogen engines and fuel-cell systems face the same system-level limitation: without accessible and affordable low-emission hydrogen, their technical maturity is of limited practical value. Hydrogen engines offer robustness, compatibility with existing manufacturing and promising efficiency potential, but still require careful management of NO_x, backfire and mixture formation. Fuel cells provide higher efficiency and essentially zero tailpipe pollutants, yet remain costly and sensitive to hydrogen purity [65], [66]. Across the literature, hydrogen emerges less as a universal transport fuel and more as a targeted option for applications where batteries are impractical and where hydrogen can be supplied reliably and at scale, for example HD corridors, industrial clusters or specific off-road use cases. In the following, the focus is on H2DF operation of CI engines, which leverages existing diesel engine hardware while partially replacing diesel with hydrogen.

Diesel Internal Combustion Engine Basics

Diesel ICE (named in 1883 after Rudolf Diesel [67]) operate based on CI, where fuel is injected into highly compressed air, causing spontaneous ignition. Unlike spark ignition (SI) engines, diesel ICE rely on precise fuel injection strategies to regulate combustion timing and subsequently power output and engine-out emissions [55]. Diesel powertrains have dominated medium and heavy road freight transport for decades because they offer high energy efficiency, robust long-term durability, and a globally established fuel and service infrastructure. Their broad availability and reliable load-carrying capability continue to make them the central backbone of commercial transport applications in many regions. [68]

Modern diesel engines employ common-rail direct injection (DI), which allows high-pressure fuel injection independent of engine speed. Nowadays, multiple injection strategies are commonly used [55]:

- **Pilot injection**: a small quantity of fuel is injected before the main injection to reduce ignition delay and control pressure rise, enhancing smooth combustion. Typically at 20 CADbTDC to 10 CADbTDC (crank angle degree before top dead center[1]).
- **Main injection**: the primary fuel dose is injected, determining the main heat release and thus the engine's work output. Typically at 8 CADbTDC to 2 CADbTDC.
- **Post injection**: additional fuel is injected after the main injection to improve PM oxidation. Typically at 20 CADaTDC to 50 CADaTDC (crank angle degree after top dead center).

Advanced injection strategies include combinations of pilot, main and post injections, often with multiple splits and injection-rate shaping to control combustion phasing and mixture formation. Multiple splits modulate the heat-release profile by dividing the total fuel mass into several precisely timed events (e.g. pilot 1, pilot 2), each influencing the thermodynamic conditions that govern ignition and subsequent heat release. Injection-rate shaping, in contrast, tailors the injection rate within a single injection event by manipulating needle-lift dynamics or rail-pressure evolution, thereby affecting ignition delay, the combustion dynamics, and the resulting pressure-rise behavior. Together, these mechanisms enable more precise control over efficiency and emission formation compared to single-injection strategies. [69], [70]

NO_x formation occurs due to high combustion temperatures and the presence of oxygen. In contrast, soot (a primary component of PM) forms as a result of incomplete combustion, especially in fuel-rich regions of the combustion chamber. In this thesis, the terms soot and PM are used interchangeably unless otherwise specified. [55]

ICE are commonly analyzed in research using in-cylinder pressure measurements obtained from e.g. piezoelectric sensors. From the cylinder pressure trace p_{cyl}, the indicated mean effective

[1] Top dead center (TDC) refers to the position of the piston when it is at the highest point in the cylinder during the compression stroke.

pressure p_{mi} (IMEP), can be calculated. It is a measure of the average (hypothetical) pressure in the cylinder over an engine cycle [71]:

$$p_{\mathrm{mi}} = \frac{W_{\mathrm{i}}}{V_{\mathrm{d}}} = \frac{1}{V_{\mathrm{d}}} \oint p_{\mathrm{cyl}}\, dV, \tag{2.1}$$

where W_{i} denotes the indicated work done over a cycle, V_{d} equals the cylinder displacement volume and the integral $\oint p_{\mathrm{cyl}}\, dV$ represents the area enclosed by the pressure–volume trace over one engine cycle. Neglecting frictional losses and normalizing over the number of cylinders, p_{mi} is directly proportional to the engine's indicated output power at fixed speed. As a result, it serves as a standardized metric for evaluating engine performance. [71]

The maximum pressure rise rate dp_{max} (MPRR) in (bar/CAD) is calculated as the maximum value of the derivative of the in-cylinder pressure p_{cyl} with respect to crank angle α [71]:

$$dp_{\mathrm{max}} = \max\Big(\frac{dp_{\mathrm{cyl}}}{d\alpha}\Big). \tag{2.2}$$

Maximum pressure rise rate dp_{max} correlates with mechanical stress: a high dp_{max} can cause engine noise and durability concerns [55]. When developing prototype engines such as in this study, dp_{max} has to be monitored in particular to avoid knocking and to guarantee safe engine operation.

Hydrogen-Diesel Dual-Fuel Internal Combustion Engine

H2DF operation has gained attention as a practical, retrofit-focused and immediately deployable strategy for reducing tank-to-wheel CO_2 and soot emissions in MD and HD transport. In contrast to dedicated hydrogen engines, fuel cells or battery-electric solutions, existing CI platforms are upgraded with hydrogen supply and mixture preparation systems so that a fraction of the diesel energy is substituted by hydrogen. This makes H2DF particularly attractive for high-value assets with long service life, such as mining trucks, stationary power systems and freight vehicles in remote regions where grid access is limited and hydrogen infrastructure will only expand gradually. In such contexts, retrofits deliver these reductions on relatively short time scales without the cost and downtime of full powertrain replacement, while the unchanged diesel path preserves operability whenever hydrogen supply is constrained. [15], [16], [17], [18], [19], [20], [21], [22].

From a system perspective, the overall decarbonization potential of H2DF is constrained by structural limitations. Since hydrogen refueling infrastructure is sparse in many regions, especially off-grid or remote areas, achievable hydrogen utilization in daily operation may be lower than the levels demonstrated in controlled research studies. Additionally, as stated in detail in Section 2.1, on-board hydrogen utilization has multiple drawbacks compared to conventional diesel ICE such as storage, on-board tanks and safety. Consequently, H2DF must be regarded as a transitional strategy that bridges current diesel technology and future zero-carbon propulsion

systems. Its advantages lie in low entry cost, rapid deployability and compatibility with existing fleets rather than in full decarbonization. [19], [20], [21], [22]

Converting a conventional diesel engine into an H2DF engine primarily affects the fuel injection and mixture preparation system. Hydrogen introduction is typically achieved through port fuel injection (PFI) or direct injection (DI). PFI introduces hydrogen into the intake manifold upstream of the intake valves where it mixes with the intake air. This method is simple and cost efficient and therefore well suited for retrofit configurations [16]. However, premixing hydrogen in the intake system raises the risk of pre-ignition or backfire at elevated hydrogen utilization [72]. Furthermore, gaseous hydrogen displaces intake air and reduces volumetric efficiency, limiting full-load capability unless additional boost pressure is supplied. DI strategies address several of these limitations by injecting hydrogen directly into the cylinder during late compression. This reduces pre-ignition risk, increases controllability of mixture stratification and allows higher hydrogen share [16]. The drawback is increased system complexity and cost because DI requires new injector hardware, high-pressure hydrogen supply and more extensive integration effort. This reduces its suitability for simple retrofit applications and places it closer to clean-sheet engine development.

Hydrogen significantly influences the combustion process. Its wide flammability range, low ignition energy and high laminar flame speed accelerate ignition and heat release. Thus, knock and pre-ignition become more likely at elevated hydrogen utilization or high engine load [13]. Because hydrogen promotes rapid premixed combustion, $dp_{\max}$ is often significantly higher than in conventional diesel operation [13]. Managing $dp_{\max}$ is essential to maintain acceptable combustion noise and mechanical stress levels. Hydrogen slip is another important phenomenon. At low load conditions, incomplete oxidation can allow unburned hydrogen to pass into the exhaust system [54], [73]; for the engine used in this work, these emissions were quantified by [74]. This results in energy and efficiency losses and has indirect climate impacts because hydrogen influences atmospheric chemistry [54], [75], [76]. Despite these challenges, hydrogen utilization generally increases thermal efficiency. The potential for leaner combustion and reduced heat losses often yields efficiency gains relative to neat diesel operation [16], [54].

Quantifying the degree of hydrogen replacement requires the hydrogen energy share (HES). It is defined as the hydrogen energy fraction, typically reported in percent for readability:

$$\mathrm{HES} = \frac{m_{\mathrm{H2}}\,\mathrm{LHV}_{\mathrm{H2}}}{m_{\mathrm{H2}}\,\mathrm{LHV}_{\mathrm{H2}} + m_{\mathrm{D}}\,\mathrm{LHV}_{\mathrm{D}}}, \tag{2.3}$$

where m_{H2} and m_{D} represent the injected masses per cycle and $\mathrm{LHV}_{\mathrm{H2}}$ and $\mathrm{LHV}_{\mathrm{D}}$ are the respective lower heating values. Since injected mass depends on the duration of injection (DOI), injector characterization is required for accurate estimation. Relevant fuel properties are summarized in Table 2.1.

Hydrogen utilization reduces carbon-based local emissions. Across experimental studies, significant reductions in CO, CO_2, PM and unburned hydrocarbons (HC) are consistently observed.

The main exception is NO_x. Because hydrogen increases local flame temperatures and accelerates heat release, NO_x emissions often rise, particularly at medium and high loads. Some very lean, low-load operating points exhibit reduced NO_x, but the operating window in which CO, CO_2, PM and unburned HC can be strongly reduced while avoiding NO_x increases is narrow and sensitive to both load and HES. A detailed survey on H2DF engines in literature is provided in Section 1.2. [16], [72]

The following Section introduces the relevant control fundamentals and methods employed for H2DF engine control.

2.2 Control Systems & Model Predictive Control

A control system regulates the behavior of a system to achieve specific objectives by processing inputs y and generating outputs u, also called control actions. The system dynamics to be controlled can be divided up into multiple-input multiple-output (MIMO) and single-input single-output (SISO) systems. MIMO systems are characterized by several inputs and outputs, where the inputs and outputs may have cross-couplings [29]. Moreover, control systems are traditionally categorized as open-loop or closed-loop systems [77], [78].

Open-loop systems operate without feedback, using predefined control actions that are effective when input-output relationships are well understood and disturbances are minimal. However, they cannot adapt to dynamic changes like disturbances [79].

Closed-loop systems, or feedback systems, measure the system's actual output y and compare it with the desired reference value r to compute the control deviation or error e. The feedback controller thus uses this error to adjust the control signal u that reduces the discrepancy, ensuring the system tracks the reference signal. This feedback mechanism improves robustness and disturbance d rejection but requires accurate observers like sensors and more complex designs compared to open-loop controllers [77], [80]. A generic closed-loop control system operating in the discrete time domain, with the timestep denoted as k, is illustrated in Fig. 2.1. The output of the controlled system is fed back as $y(k-1)$ to be considered for calculating the controls of the next time step [80], [81].

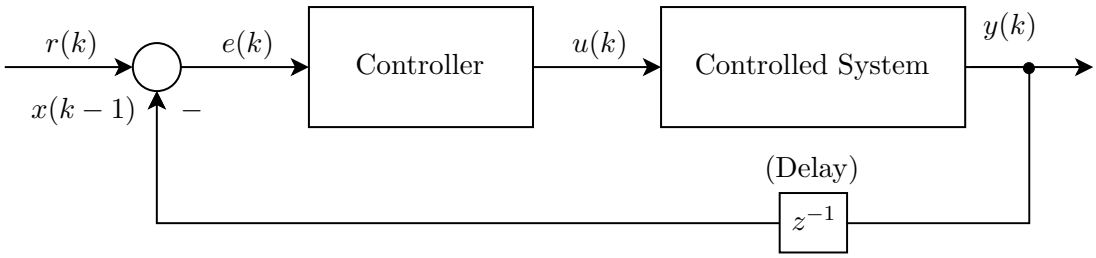


Figure 2.1: Generic closed-loop control system in discretized time domain. u: controls, y: system output, r: reference, e: control deviation, k: timestep.

It is noteworthy that all relevant system outputs in this thesis are either directly measured or derived from sensor data in real time, eliminating the need for a state observer.

The dynamics of a system can be further classified as linear or nonlinear. Linear systems are characterized by dynamics that can be described by linear differential or difference equations, where the system response depends linearly on the initial conditions and control inputs. This linearity makes such systems analytically solvable and easier to control. In contrast, nonlinear systems exhibit dynamics governed by nonlinear equations, which often arise in real-world applications due to phenomena such as saturation, friction, or complex interactions. ICE systems exhibit nonlinear behavior that must be accounted for in control design, as linearized representations are often insufficient [29]. Subsequently, the corresponding time-invariant, nonlinear and discretized model in a state-space representation including the initial condition is defined as follows [80]:

$$x(k+1) = f\big(x(k),\, u(k)\big), \tag{2.4a}$$

$$y(k) = h\big(x(k),\, u(k)\big), \tag{2.4b}$$

$$x(0) = x_0, \tag{2.4c}$$

where $x(k)$ is the state vector and f and h are nonlinear functions describing the state transition and output mapping, respectively. x_0 denotes the initial condition at timestep $k = 0$.

Traditionally, control systems have predominantly relied on proportional-integral (PI) controllers or look-up-tables (LUTs) due to their simplicity and effectiveness. However, the presence of highly nonlinear dynamics as in ICE, necessitates the use of advanced control strategies such as MPC. [29]

Model Predictive Control

MPC is an advanced control method that enables multi-objective optimization by leveraging a process model and solving an optimization problem. It uses a linear or nonlinear dynamic model to predict optimal control inputs over a finite prediction horizon. The two core features of MPC are the explicit incorporation of a system model and the repeated solution of an optimal control problem (OCP). At each time step, only the first control input of the computed sequence is applied to the system, while the remaining predicted inputs span future time steps. In the next time step, the entire optimization is repeated with updated state information; this approach is known as the receding horizon principle. [80], [81]

A schematic representation of this MPC tracking strategy is shown in Fig. 2.2. The controller chooses control actions $u(\cdot \mid k)$ at timestep k for the system output $y(\cdot \mid k)$ to reach the desired reference trajectory $r(\cdot \mid k)$. The notation $(\cdot \mid k)$ includes all predicted steps starting from the time k over the whole horizon length. Due to the control horizon N_u and prediction horizon N_p being identical in the applications of this study, the common horizon length $N = N_\mathrm{u} = N_\mathrm{p}$ is defined for simplification of the notation. [80]

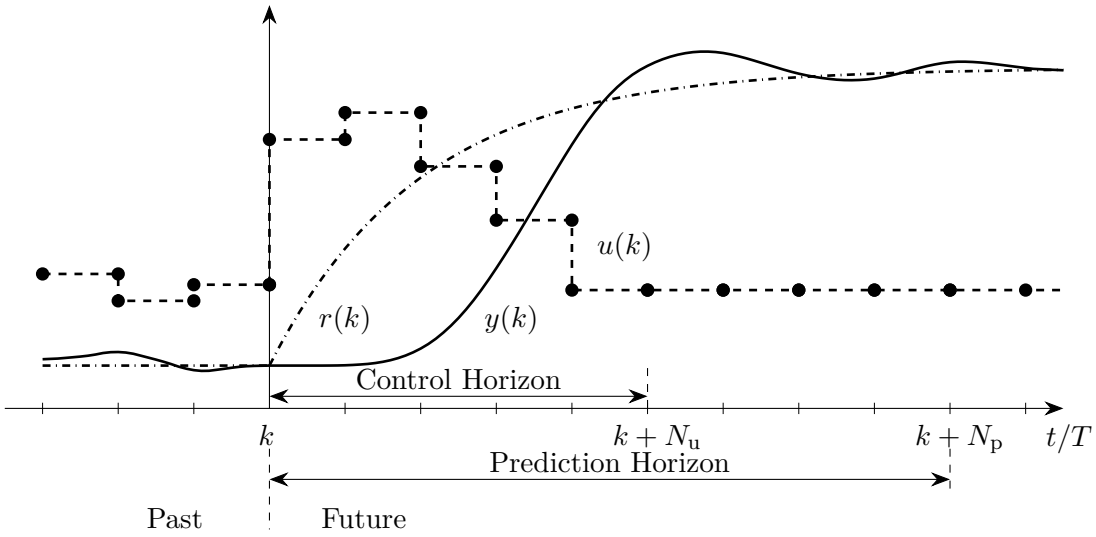


Figure 2.2: Principle of model predictive control. u: control trajectory, y: output trajectory, r: reference trajectory, k: timestep. Based on [28].

Subsequently, Eq. 2.5 defines the finite-dimensional time-invariant OCP [1], [80]:

$$
\begin{aligned}
\underset{\substack{u_0,\dots,u_{N-1}\\ x_0,\dots,x_N}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \underbrace{J_\mathrm{l}(x_k, u_k)}_{\text{stage cost / lagrange term}} + \underbrace{J_\mathrm{m}(x_N)}_{\text{terminal cost / mayer term}} \\
\text{subject to} \quad & x_0 = x(k), \\
& x_{k+1} = f(x_k, u_k) \quad \forall k \in \mathbb{N} \setminus N, \\
& g(x_k, u_k) = 0 \quad \forall k \in \mathbb{N} \setminus N, \\
& h(x_k, u_k) \leq 0 \quad \forall k \in \mathbb{N}
\end{aligned}
\tag{2.5}
$$

where $\mathbb{N} = \{0, 1, \dots, N\}$. In addition, the OCP incorporates nonlinear equality constraints $g(x, u)$ and inequality constraints $h(x, u)$, which contain state and action bounds as well as general path constraints. The system dynamics $f(x_k, u_k)$ were previously defined as nonlinear in Eq. 2.4a. The cost function to be minimized for penalizing the system's behavior consists of the stage cost lagrange term $J_\mathrm{l}(x_k, u_k)$ and the terminal cost mayer term $J_\mathrm{m}(x_N)$. The subscript k in x_k indicates that the variables are part of the predictions $x(\cdot \mid k)$, whereas $x(k)$ denotes the current state at timestep k.

The OCP in Eq. 2.5 can also be denoted as finite-dimensional OCP-structured nonlinear programming problem (NLP). This formulation is derived from a general time-continuous OCP through the direct multiple shooting method. This involves incorporating the gap-closing constraints $x_{k+1} = f(x_k, u_k)$ and $x_0 = x(k)$ and augmenting the dependent variables x_k to the optimization variables. Although this approach increases the complexity, it helps to address steep derivatives and sensitivity to the initial condition in gradient-based optimization, particularly for nonlinear systems. This is achieved by splitting the trajectory into smaller segments, each with its own initial conditions. Single shooting in contrast optimizes the trajectory based

only on the initial state, with subsequent states computed by forward integration of the dynamics, optimizing the entire trajectory as a single continuous path. Each stage between the shooting nodes can be independently adjusted and optimized, resulting in improved numerical stability, better convergence properties and being more robust towards disturbances. As a result, direct multiple shooting involves higher computational costs but often achieves better convergence and requires fewer iterations compared to simpler shooting methods like single shooting. [80], [82]

The resulting OCP-structured NLP has a block-sparse structure which makes it well-suited for numerical optimization algorithms like sequential quadratic programming (SQP) [83]. SQP iteratively approximates the NLP by solving local quadratic sub-problems for the objective and linearizing constraints around the current solution. The method efficiently exploits the block-sparse structure of the OCP to reduce computational complexity and accelerate convergence. One of the strengths of SQP algorithms is their warm-starting capability, which allows the previous solution to be the initial guess for subsequent optimization steps. This is especially effective when the sampling time is chosen such that the actual system's behavior does not deviate much from the model's prediction [83]. Under these conditions, optimization can often be completed in a single iteration per timestep, a strategy known as the SQP real-time iteration (RTI) scheme by Diehl et al. [84].

One of the main contributions of this thesis is the utilization of learned, data-driven models within the developed controllers, including MPC. These models are detailed in the following section.

2.3 Deep Learning

Machine learning (ML) is a subset of artificial intelligence (AI) in which learning algorithms create models based on data without being explicitly programmed. In comparison to rule-based systems, ML models are capable of learning patterns from the data and improve their performance over time. Due to the rise of available data and increased computational power ML has gained significant attention in recent years [85], [86].

ML consists of different learning approaches, each suited to specific problems and data types. These subsets are defined as the following [87]:

- **Supervised learning:** models are trained using labeled input-output data to recognize patterns and make future predictions. Examples include linear regression, logistic regression, and artificial neural networks (ANN).
- **Unsupervised learning:** works with unlabeled data, aiming to identify patterns, such as clustering (e.g., k-means clustering).
- **Reinforcement learning (RL):** an agent is trained through a reward system, reinforcing desired actions while penalizing undesired behaviors.

This thesis primarily focuses on a subset of ML: DL and supervised learning, which are explained in the following more deeply. Additionally, AutoML as a combination of ML and automation, is explained briefly. It is noteworthy, that the work in this thesis is bounded to time-series regression tasks and subsequently, classification problems are ignored.

Deep Neural Networks

DL utilizes multi-layered ANNs to automatically extract features from data. Deep neural networks (DNNs), defined as deep ANNs with multiple layers, are particularly suited for time regression tasks and are therefore discussed in more detail due to their relevance for the developed control concepts. [88]

The core element of DNNs is the single neuron, also referred to as perceptron. The perceptron is based on the principle of the human brain and was introduced by Rosenblatt [89] in 1958. The functional structure of a single neuron is shown in Fig. 2.3. The output y is calculated as described in Eq. 2.6:

$$y = a(z) = a\left(\sum_{i=1}^{p} w_i\, u_i + b\right). \tag{2.6}$$

For this purpose, the inputs u_i, weighted by the weights w_i, and a bias b are summed. The bias adds an offset that frees the neuron from having its output anchored at the origin, allowing it to represent a wider and more useful set of functions. The output is finally processed by an activation function (AF) $a(z)$ depending on the calculated sum. [88], [89]

AFs introduce nonlinearity into DNNs and map inputs to specific output ranges, enabling them to learn complex relationships. Common AFs include [88]:

- **Sigmoid** (σ): maps inputs to $(0, 1)$. Often used in binary classification but can cause vanishing gradients.
- **Tanh** (tanh): outputs in $(-1, 1)$, zero-centered, but still prone to saturation.
- **ReLU** (rectified linear unit): outputs zero for negative inputs and linear for positive ones. Efficient and widely used in hidden layers.

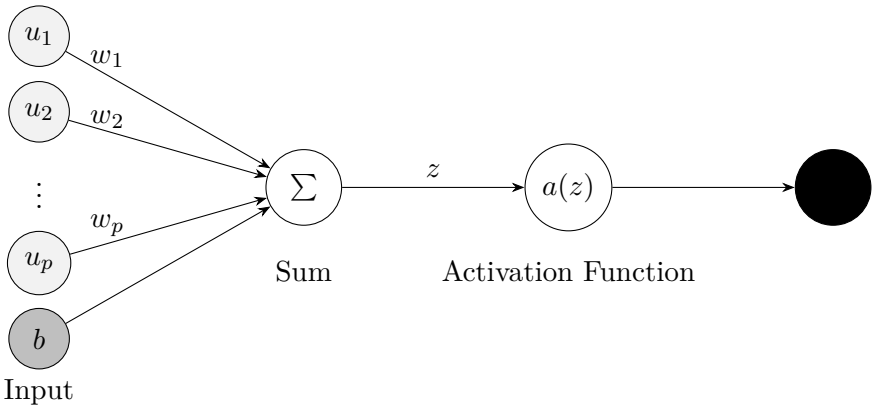


Figure 2.3: Perceptron. Principle of a single network neuron.

- **Unit step**: binary output (0 or 1) based on sign of input; rarely used due to lack of differentiability.

The selection of AF is typically guided by empirical performance and task-specific requirements [86], [88]. Fig. A.1 shows their respective behavior graphically.

Feedforward Neural Networks

Feedforward neural networks (FFNN) represent a fundamental class of ANNs, where information flows in one direction (from input to output) without feedback connections. They are composed of stacked layers of perceptrons. A basic FFNN consists of an input layer, one or more hidden layers (often FC) to each other and an output layer. Each layer receives input from the previous layer and passes its output to the next, forming a sequential chain of transformations. This structure is illustrated in Fig. 2.4. [89]

The network's behavior is governed by trainable parameters $\theta = \{w_i, b_i\}$, where w_i and b_i denote the weights and biases of each layer as presented in Eq. 2.6. The number of hidden layers defines the depth of the network, while each layer contains multiple neurons, also referred to as units, perceptrons or nodes. Since FFNNs lack internal feedback, they are inherently static and unable to model temporal dependencies or sequential data without lagged inputs, which distinguishes them from the recurrent architectures discussed in the following. [88], [89]

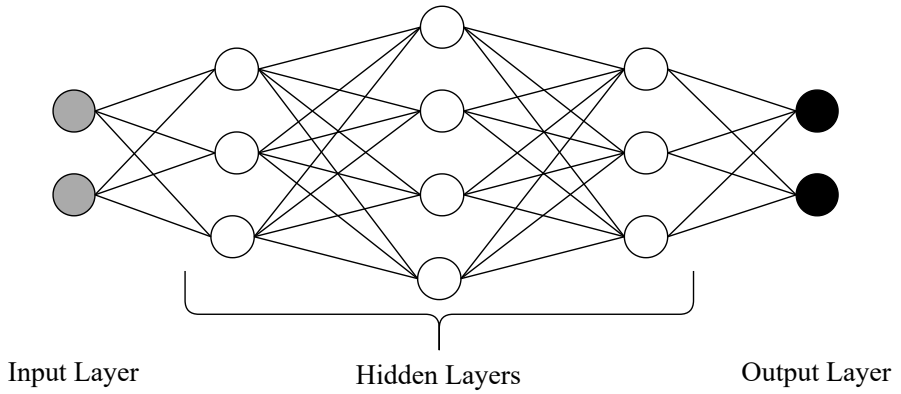
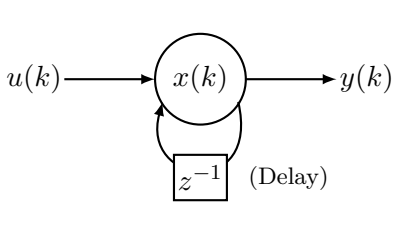


Figure 2.4: Visualization of a generic feedforward neural network.

Recurrent Neural Networks

In principle, recurrent neural networks (RNNs) can also be considered as FFNN, extended by a feedback, as for the simple recurrent neuron shown in Fig. 2.5.

The output $y(k)$ is now determined as a function of the input $u(k)$, the parameters θ and the previous states $x(k-1)$. The current states $x(k)$ now also depend on the previous states $x(k-1)$. There are various RNNs with different types of feedback, such as internal or external feedback. Due to this feedback, RNNs are well suited for the consideration of sequential data,



$$y(k) = f\big(u(k),\, x(k),\, \theta\big)$$
$$x(k) = f\big(u(k),\, x(k-1),\, \theta\big)$$

Figure 2.5: Recurrent neuron. u: input, y: output, x: state, θ: parameter.

such as time-series data, and are therefore also referred to as sequence models. Two of the major recurrent units are explained briefly below. [88], [90]

Long Short-Term Memory Unit: The Long Short-Term Memory (LSTM) is one of the most successful and widely used RNNs. The principle of an LSTM cell was developed by Hochreiter and Schmidhuber [91] in 1997 and further improved by Gers et al. [92]. The cell is characterized by its ability to account for long-term dependencies, while handling the vanishing gradient problem encountered for other RNNs. [91], [92]

The schematic structure of an LSTM cell is shown in Fig. 2.6. As a gated RNN, the LSTM regulates information flow using three gates: the input gate $i(k)$, the forget gate $f(k)$, and the output gate $o(k)$. Each gate applies a sigmoid (σ) AF to control which information is stored, forgotten, or exposed. The candidate cell input $g(k)$, computed via a hyperbolic tangent tanh AF, proposes new content for the memory and is regulated by the input gate $i(k)$. The internal memory of the LSTM is maintained in the cell state $c(k)$, which captures long-term dependencies via additive updates. The forget gate determines which portions of the previous cell state $c(k-1)$ are retained, while the input gate governs how much of the new candidate $g(k)$ is written to memory. The hidden state $h(k)$, which serves as both the network output

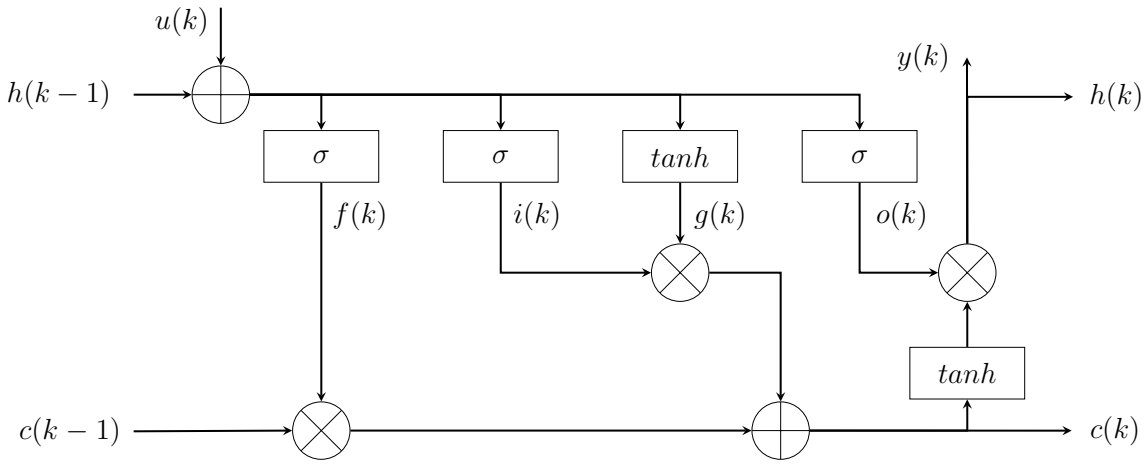


Figure 2.6: Long short-term memory (LSTM) unit. $\oplus$ element-wise addition, $\otimes$ element-wise multiplication, u: input, y: output, c: cell state, h: hidden state, σ: sigmoid function, tanh: hyperbolic tangent function.

and external feedback, is derived from the updated cell state and regulated by the output gate. The complete behavior of the LSTM cell is described mathematically in Eq. 2.7 [88].

$$i(k) = \sigma\left(w_{u,i}^{\top}\, u(k) + w_{h,i}^{\top}\, h(k-1) + b_i\right), \tag{2.7a}$$

$$o(k) = \sigma\left(w_{u,o}^{\top}\, u(k) + w_{h,o}^{\top}\, h(k-1) + b_o\right), \tag{2.7b}$$

$$f(k) = \sigma\left(w_{u,f}^{\top}\, u(k) + w_{h,f}^{\top}\, h(k-1) + b_f\right), \tag{2.7c}$$

$$g(k) = \tanh\left(w_{u,g}^{\top}\, u(k) + w_{h,g}^{\top}\, h(k-1) + b_g\right), \tag{2.7d}$$

$$c(k) = f(k) \otimes c(k-1) + i(k) \otimes g(k), \tag{2.7e}$$

$$h(k) = o(k) \otimes \tanh\big(c(k)\big), \tag{2.7f}$$

$$y(k) = h(k). \tag{2.7g}$$

Gated-Recurrent Unit: The GRU is a variant of the RNN and was introduced by Cho et al. [93] in 2014. Similar to LSTMs, the GRU is designed to capture long-term dependencies and mitigate the vanishing gradient problem found in traditional RNNs.

The schematic structure of a GRU cell is shown in Fig. 2.7. Unlike LSTMs, the GRU does not have a separate cell state. Instead, it controls the flow of information using only the hidden state $h(k)$ and three gating mechanisms: the update gate $z(k)$, the reset gate $r(k)$ and the candidate

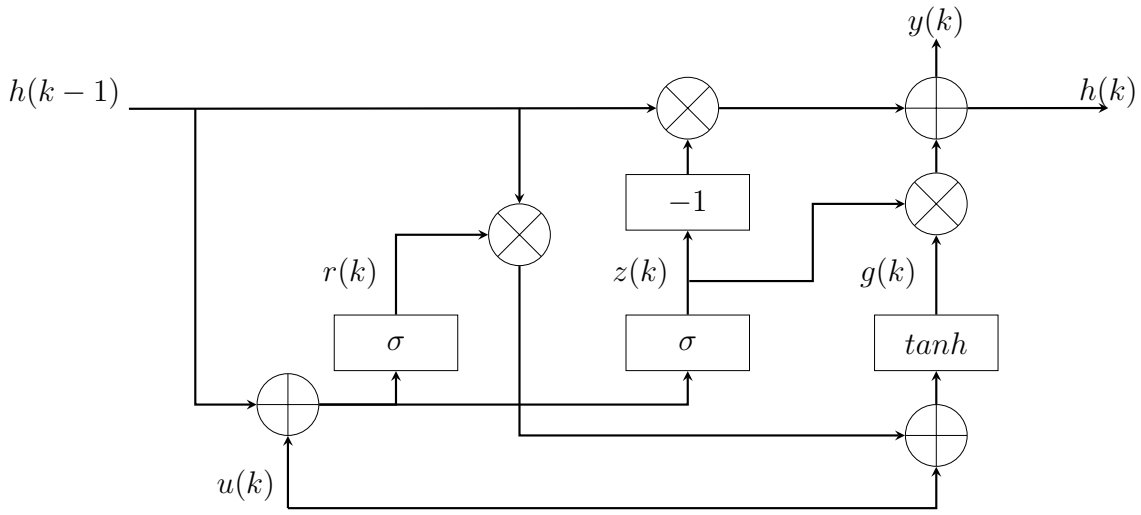


Figure 2.7: Gated recurrent unit (GRU). $\oplus$ element-wise addition, $\otimes$ element-wise multiplication, u: input, y: output, h: hidden state, σ: sigmoid function, tanh: hyperbolic tangent function.

gate g. The complete GRU cell is mathematically described as follows [88]:

$$z(k) = \sigma\left(w_{u,z}^{\top} u(k) + w_{h,z}^{\top} h(k-1) + b_z\right), \tag{2.8a}$$

$$r(k) = \sigma\left(w_{u,r}^{\top} u(k) + w_{h,r}^{\top} h(k-1) + b_r\right), \tag{2.8b}$$

$$g(k) = \tanh\left(w_{u,g}^{\top} u(k) + w_{h,g}^{\top} \left(r(k) \otimes h(k-1)\right) + b_g\right), \tag{2.8c}$$

$$h(k) = (1 - z(k)) \otimes h(k-1) + z(k) \otimes g(k), \tag{2.8d}$$

$$y(k) = h(k). \tag{2.8e}$$

Compared to LSTMs, GRUs have fewer parameters because they do not include a separate cell state. This often results in faster training times while maintaining strong performance for sequence-based tasks. In addition, when integrated into MPC, the computational overhead of GRU compared to LSTM units is reduced significantly by halving the number of dynamic states coming from the unit itself as well as the number of operations when forward-propagating. [94], [95]

Supervised Learning: Model Training

Training a DNN involves finding the optimal set of weights and biases that minimize a loss function. This is achieved through optimization algorithms, which update the model parameters iteratively based on the gradient of the loss function. The two most commonly used optimization techniques are stochastic gradient descent (SGD) and adaptive moment estimation (Adam). SGD is a fundamental optimization method that updates model parameters using small batches of data rather than the entire dataset. It is computationally efficient but can be unstable due to noisy updates. Despite its simplicity, it remains a strong baseline for training DNNs. Adam is an adaptive optimization algorithm that adjusts learning rates individually for each parameter based on past gradients. It combines the benefits of momentum, leading to faster convergence and better performance in many cases. However, it requires careful hyperparameter tuning [96]. The process of training a DNN is illustrated in pseudo-code in Algorithm A.1. [88]

Assessing the performance of DNNs requires appropriate metrics [85], [97]. The root mean squared error (RMSE) measures the average magnitude of prediction errors, penalizing larger errors more heavily than smaller ones. It is defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{2.9}$$

where n is the number of samples, y_i represents the true value and $\hat{y}_i$ is the predicted value. To calculate the normalized RMSE (NRMSE), the RMSE is normalized with its range ($y_{\max} - y_{\min}$) helping to compare model performance across different scales:

$$\mathrm{NRMSE} = \frac{\mathrm{RMSE}}{y_{\max} - y_{\min}}. \tag{2.10}$$

For reference tracking, the range of the reference trajectory is used, and the range applied is stated wherever NRMSE values are reported. The NRMSE can also be normalized using different ranges to be more robust towards outliers in the data. These can be a multiple of the standard deviation (σ) or a so-called clipping with percentiles (clipped), e.g. removing the first and 99th percentiles. This avoids outliers that would otherwise inflate the normalization range and artificially improve results. Finally, the mean absolute error (MAE) measures the average absolute difference between actual and predicted values. Unlike RMSE, it does not heavily penalize large errors:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{2.11}$$

Furthermore, several hyperparameters influence the training dynamics and final performance of DNNs. The batch size determines the number of samples processed before the model updates its parameters. A small batch size allows for better generalization, while a large batch size leads to faster training but increases the risk of overfitting. Epochs define how many times the entire dataset is passed through the neural network during training. More epochs improve model learning but may lead to overfitting if training continues for too long. Additionally, the learning rate controls how much the model updates its weights with each optimization step. Moreover, a high learning rate speeds up training but may cause instability, while a low learning rate ensures smooth convergence but may be too slow. Subsequently, the initial learning rate is an important hyperparameter as well, since the training can diverge and get unstable right from the beginning. In practice, the learning rate is often scheduled to be reduced in certain steps for better convergence behavior. [88], [98]

Behavior Cloning

While MPC requires an explicitly defined and accurate model for the system dynamics, obtaining such a model can be challenging or computationally inefficient for many real-world systems. An alternative approach is to learn control policies directly from expert demonstrations, bypassing the need for an explicit dynamics model and avoiding the expensive iterative computations inherent to optimization-based MPC methods. [99], [100]

One such approach is behavior cloning (BC), a fundamental technique in imitation learning (IL), where a model or agent learns a policy by mimicking expert demonstrations. In BC, a direct mapping from observed states to actions using supervised learning techniques is learned: given a dataset of expert demonstrations $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N}$, where s_i represents the observed state and a_i the corresponding expert action, the goal is to learn a policy $\pi_\theta(s)$ that approximates the expert's decision-making process:

$$\pi_\theta(s) = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}(\pi_\theta(s_i),\, a_i), \tag{2.12}$$

where $\mathcal{L}$ represents one of the loss functions defined in Section 2.3 (performance metrics, e.g.

RMSE). An equivalent pseudo-code representation of the BC training process for a DNN is provided in Algorithm A.2. [99], [100]

BC is advantageous when sufficient expert demonstrations from existing controllers are available, making it sample-efficient and quick to deploy. However, it suffers from covariate shift, lack of robustness, and dependence on high-quality expert data. To address these issues, techniques like dataset aggregation [101], regularization methods, and hybrid approaches combining BC with RL have been proposed. [88], [101], [102]

Automated Machine Learning for Deep Neural Networks

Automated machine learning (AutoML) is an approach in AI, aiming to simplify and accelerate the ML workflow. It automates various stages of model development, including preprocessing, feature engineering, model selection or neural architecture search (NAS), hyperparameter optimization (HPO), and in some variants even deployment. By doing so, AutoML opens up ML, allowing non-experts to develop robust models while improving productivity for experts. [103], [104]

The AutoML process for DNNs involves the following key steps [104]:

1. **Defining the search space:** specify the range of possible architectures, hyperparameters, and training configurations to explore.
2. **Search strategy:** employ algorithms to efficiently navigate the search space, balancing exploration and exploitation.
3. **Evaluation:** train and evaluate candidate models using validation datasets, employing performance metrics.
4. **Optimization:** use results from evaluations to refine the search strategy and converge to optimal configurations.

A pseudo-code representation of the process is presented in Algorithm A.3.

Due to the complexity and large search spaces underlying in DNN design, two primary areas of AutoML for DNN are HPO and NAS. [105]

HPO: Finding the best set of training hyperparameters to maximize model and training performance is the goal of HPO. As described in Section 2.3, hyperparameters are settings that control the training process and model structure, such as learning rate, batch size, and number of layers. Effective HPO can significantly improve a DNN's accuracy and its efficiency in training. [103]

NAS: The goal of NAS is to automate the design of the network architecture itself. It aims to discover architectures that balance high prediction performance with further metrics such as computational efficiency of the DNN in both model training and deployment.[105]

Hardware-aware NAS considers device constraints, such as memory or latency, to optimize architectures for deployment on specific hardware [105]. While an integration of the target hardware into the AutoML process or a simulation of those can be either inefficient or complex, simple metrics to evaluate the computational efficiency of a DNN are as follows [88]:

- **FLOPs (floating point operations):** FLOPs are calculated as the total number of multiplications and additions required for a single forward pass through the network. E.g., for a FC layer with N vector inputs, M outputs (see Eq. 2.6) and bias additions, and a following ReLU activation, the FLOPs can be expressed as [106]:

$$\text{FLOPs}_{\text{FC+ReLU}} = \underbrace{2\,N\,M}_{\substack{\text{matrix–vector} \\ \text{multiplications \& additions}}} + \underbrace{M}_{\text{bias additions}} + \underbrace{M}_{\text{ReLU activations}} . \tag{2.13}$$

- $N_{\textbf{Param}}$ **(number of learnable parameters):** represents the model's capacity. This includes the total weights and biases (parameters) in a network. For an FC layer with N inputs and M outputs (see Eq. 2.6), the total number of parameters is:

$$N_{\text{Param}} = \underbrace{N \times M}_{\text{weights}} + \underbrace{M}_{\text{biases}} . \tag{2.14}$$

There is a correlation between FLOPs and the number of learnable parameters: higher numbers of learnable parameters often result in increased FLOPs, as larger layers or more complex architectures require more computations. However, certain optimizations, such as sparse connections or pruning, can reduce FLOPs without significantly affecting the number of parameters. [107] While FLOPs and parameter count are useful indicators of model complexity, they do not fully capture a model's efficiency or performance. More advanced metrics such as memory access patterns, latency, and energy consumption, can offer deeper insights, but are omitted here for brevity and simplicity.

When integrating HPO and NAS, the overall search space can be large and complex. To solve this optimization problem, several techniques can be deployed:

- **Random search:** randomly samples hyperparameters from predefined distributions. It is simple to implement and often effective in high-dimensional search spaces. [108]
- **Grid search:** exhaustively explores all possible combinations of hyperparameters. While thorough, it becomes computationally prohibitive as the number of hyperparameters increases. [109]
- **Bayesian optimization (BO):** uses probabilistic models, such as gaussian processes (GP) or tree-structured parzen estimators (TPE), to predict promising hyperparameter configurations based on prior evaluations, reducing the need for extensive trials. GPs

model the objective function as a distribution, providing uncertainty estimates that guide the selection of the next configuration via acquisition functions like Expected Improvement. TPE, a scalable alternative, uses density estimation to distinguish between promising and non-promising configurations, making it effective for high-dimensional or discrete spaces. [110]

- **Hyperband pruning:** an efficient method that combines random sampling with adaptive resource allocation. It uses early stopping to discard poorly performing configurations and reallocates resources to more promising candidates. [111]

Significant challenges persist when deploying AutoML methods in computing environments. The substantial computational demands of large-scale HPO and NAS often result in high energy consumption and thus potentially increased carbon emissions, raising ecological concerns. Furthermore, the scalability of these techniques is constrained by the availability of HPC resources, as the search space expands exponentially with model complexity. [112]

3 Experimental Setup

This chapter outlines the experimental setup used in this work. It describes the engine, test bench, sensor data, and control system with its dependencies. The combustion behavior in hydrogen diesel dual-fuel (H2DF) operation is compared to conventional diesel operation based on experimental data. Key motivations emerge: the need for a nonlinear control strategy and one that meets real-time engine constraints.

All experiments were conducted at the mechanical engineering department of the UofA in Edmonton, Alberta, Canada.

The content of this chapter is partially based on [33], [35], [113], [47].

3.1 Engine & Test Bench Specifications

A modified four-cylinder *Cummins QSB* 4.5L diesel engine with a compression ratio of 17.2:1 is used both for data collection to train the controllers presented in this work and for experimental validation. The experimental setup is shown in Fig. 3.1. Table A.1 describes the engine parameters in detail. This engine is a direct injection (DI) common-rail diesel engine without exhaust gas recirculation (EGR), now naturally aspirated after removal of its original turbocharger. The main hardware modification to the engine is the addition of a hydrogen port fuel injection (PFI) (*HANA H2200A*) in the intake manifold of cylinder 1. The exhaust gas leaving cylinder 1 has been separated from the other three cylinders to allow for independent analysis. Hydrogen is stored in K-tanks[1] with a nominal pressure of 200 bar. A single-stage high flow hydrogen rated stainless steel pressure regulator set to supply hydrogen at a static pressure of 8.3 bar is utilized. Additionally, the production engine control unit (ECU) has been replaced on the engine to allow for fully flexible engine control development in `MATLAB/Simulink`. The rapid control prototyping (RCP) system *dSPACE MicroAutoBox II* (MABX) and *RapidPro* power stage have been used in this work. Additional details regarding the experimental setup can be found in Table A.2, in [113] and [35].

The *HANA H2200A* injector used is originally designed as a compressed natural gas (CNG) port injector for industrial engines with a cylinder volume of approximately 2.5L. As hydrogen has less than 30% the volumetric energy density when compared to CNG, the injector flow rates are adequate for the single 1.12L cylinder. However, the purity of hydrogen results in reduced injector lubrication and thus accelerated wear which can be seen in the results presented.

It is noteworthy that this engine is not equipped with an EGR system, which is known to significantly reduce NO_x emissions by lowering peak combustion temperatures [55].

[1] A K-tank is a standardized industrial gas cylinder, typically used for compressed gases like hydrogen, with a water capacity of approximately 50L and a nominal working pressure of 200 bar.

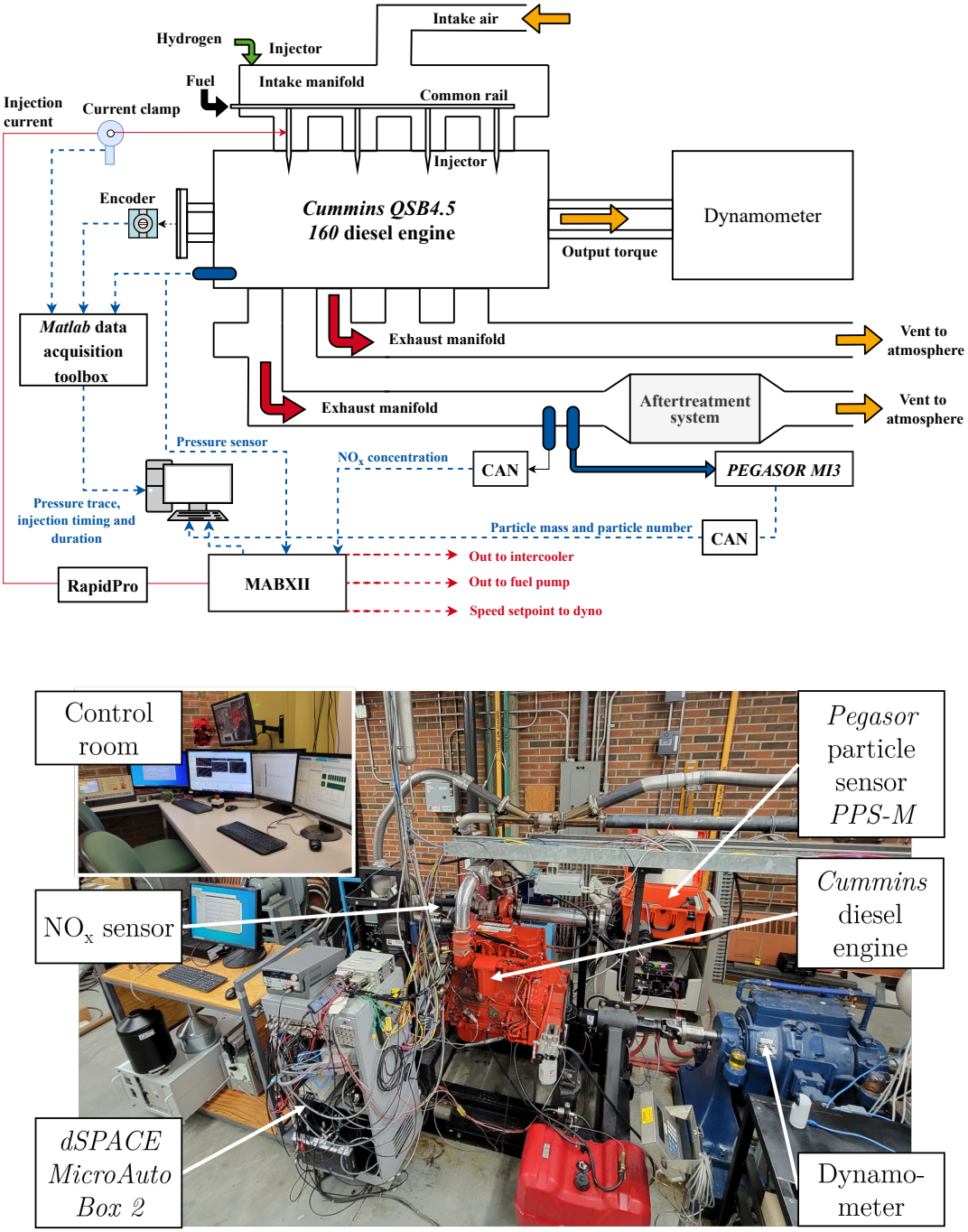


Figure 3.1: Experimental setup of the modified *Cummins QSB* 4.5L diesel engine used in this work, located at the UofA. The engine is retrofitted with a hydrogen PFI system at cylinder 1. Top: schematic overview. Bottom: photograph.

3.2 Engine Outputs: Data Acquisition & Emission Measurements

In-cylinder pressure is measured using a *Kistler 6124A* piezoelectric pressure transducer, which is mounted between the exhaust valves (detailed specifications can be found in A.5). Using the measured pressure of cylinder 1, the indicated mean effective pressure p_{mi} and maximum pressure rise rate dp_{max} can be calculated on a *Xilinx Kintex-7* field programmable gate array (FPGA) within the MABX as described in Section 2.1. Additional information can be found in [114].

Engine-out particulate matter c_{PM} emissions and NO_x emissions c_{NOx} are measured directly after the exhaust valve of cylinder 1, without any additional aftertreatment. A *Pegasor PPS-M* particle sensor is used to measure the engine-out particulate emissions, providing both particle number (in particles/m^3) and particle mass (in mg/m^3). PM emissions are difficult to accurately measure and the *PPS-M* sensor is specifically designed for monitoring vehicle emissions with a relatively quick transient response with a response time of 0.2 s. However, the sensor's performance is highly sensitive to particle size distribution and maintenance quality, with measurement errors reaching up to 50% according to the manufacturer [115], and in some cases significantly exceeding this threshold based on internal calibration data at the UofA. The *Pegasor* data is transmitted via controller area network (CAN) and logged by the control host computer. The NO_x volume concentration in parts per million (ppm) in the exhaust stream is measured using an *ECM NOxCANt* measurement module with a calibrated NO_x sensor [116]. The measured concentration is then transmitted over the CAN network to the MABX where the data is both saved and provided to the controllers. Additional specifications of both emission sensors can be found in Table A.3 and Table A.4. It should be noted that combustion phasing and center of heat release, and thus overall thermal efficiency, are not within the scope of this work and were not considered during engine operation and analysis.

In summary, there are four engine-out (cylinder 1) outputs measured and relevant for this work as depicted in Table 3.1:

Table 3.1: Measured engine output variables for cylinder 1.

Output variable	**Symbol**	**Unit**
Indicated mean effective pressure (IMEP)	p_{mi}	bar
Nitrogen oxide concentration (NO_x)	c_{NOx}	ppm
Particulate mass concentration (PM, soot)	c_{PM}	mg/m^3
Maximum pressure rise rate (MPRR)	dp_{max}	bar/CAD

3.3 Engine Controls

In the experiments presented in this thesis many of the engine operating parameters have been conditioned to remove additional variables that are not the focus of this research and to reduce the complexity of the control problem. Specifically, the following parameters are externally controlled: the engine coolant temperature, oil temperature, engine speed, and hydrogen fuel pressure have all been kept constant. Additionally, some parameters have also been set to a constant value to reduce the complexity of the control problem including: limiting diesel injection to two events (pre and main), fixed duration of pre-injection $t_{\mathrm{Pre}} = 0.17\,\mathrm{ms}$ which amounts to 3 mg of diesel fuel, diesel rail fuel pressure $p_{\mathrm{Rail}} = 970\,\mathrm{bar}$, start of injection (SOI) for the hydrogen injection $\alpha_{\mathrm{H2}} = 360\,\mathrm{CADbTDC}$ (crank angle degrees before top dead center) and intake air pressure (naturally aspirated).

Subsequently, there are four free engine controls available in this work as stated in Table 3.2:

Table 3.2: Engine control input variables for cylinder 1.

Input variable	**Symbol**	**Unit**
Duration of main diesel injection (DOI main)	t_{Main}	ms
Pre-to-main time (P2M)	t_{P2M}	µs
Start of main diesel injection (SOI main)	α_{Main}	CADbTDC
Duration of hydrogen injection (DOI hydrogen)	t_{H2}	ms

It is noteworthy, that the SOI of the diesel pre-injection α_{Pre} depends on the pre-to-main-time t_{P2M}. Furthermore, t_{P2M} is calculated using the engine speed n_{Eng} in min^{-1} as:

$$t_{\mathrm{P2M}}(k) = \left(\frac{\alpha_{\mathrm{Pre}}(k) - \alpha_{\mathrm{Main}}(k)}{6 \cdot n_{\mathrm{Eng}}(k)} - t_{\mathrm{Pre}}(k)\right). \tag{3.1}$$

Using t_{P2M} instead of the start of the pre injection α_{Pre} simplifies the control task by defining pre-injection timing relative to the main injection. This avoids nonlinear effects caused by varying α_{Main}, which can make calibration and prediction more difficult when using α_{Pre} directly. These engine controls and the dependencies between them are schematically visualized in Fig. 3.2.

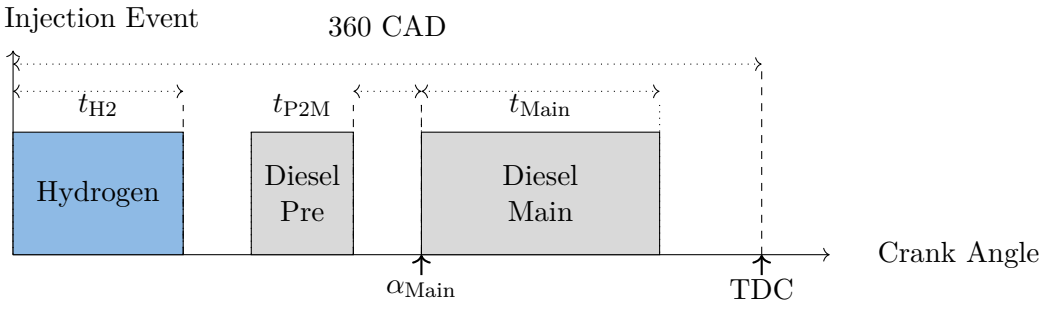


Figure 3.2: Injection timing diagram of the selected controls for cylinder 1.

3.4 Combustion Characteristics & Challenges

Fig. 3.3 compares the diesel and H2DF combustion of the *Cummins* engine cylinder 1, highlighting the impact of hydrogen addition. It shows pressure traces from cylinder 1 ($p_{\mathrm{cyl},1}$) calculated using Eq. 2.1 for 20 combustion cycles at a fixed p_{mi} of 6 bar, focused on the region around TDC (−30 CADaTDC to 60 CADaTDC). This view emphasizes the cycle-to-cycle (C2C) behavior during the peak combustion event near TDC.

The start of the pressure rise for H2DF is slightly delayed compared to diesel, occurring just before TDC. Furthermore, H2DF shows a higher dp_{max} and maximum cylinder pressure (p_{max}), which increases mechanical stress and engine wear. These effects can be attributed to hydrogen's high flame speed, which leads to rapid combustion once ignition is triggered near TDC. While H2DF exhibits a sharp pressure peak around TDC, the diesel process has a longer burn duration with moderately elevated pressure over a wider crank angle range during the expansion stroke, where dV is larger. As a result, the integral of $p\,\mathrm{d}V$ and therefore the overall p_{mi} remains essentially identical for diesel and H2DF operation, even though the zoomed view may suggest a higher value for H2DF. Because the heat release in the H2DF case is more concentrated around TDC, the process is closer to constant-volume combustion and thus can achieve a higher indicated efficiency, requiring less fuel energy for the same p_{mi}. Additionally, the H2DF trace exhibits greater C2C variation and thus combustion instability, likely due to the sensitivity of hydrogen ignition and mixing, contributing to increased combustion noise and long-term durability concerns.

Fig. 3.4 shows pressure traces of various engine runs with varying hydrogen injection duration and thus the hydrogen energy share (HES), while keeping the diesel injection timing constant. A zoomed-in view of the single traces between −10 CADaTDC to 30 CADaTDC is provided to better capture the combustion behavior near TDC. The HES ranges between 0% (pure diesel)

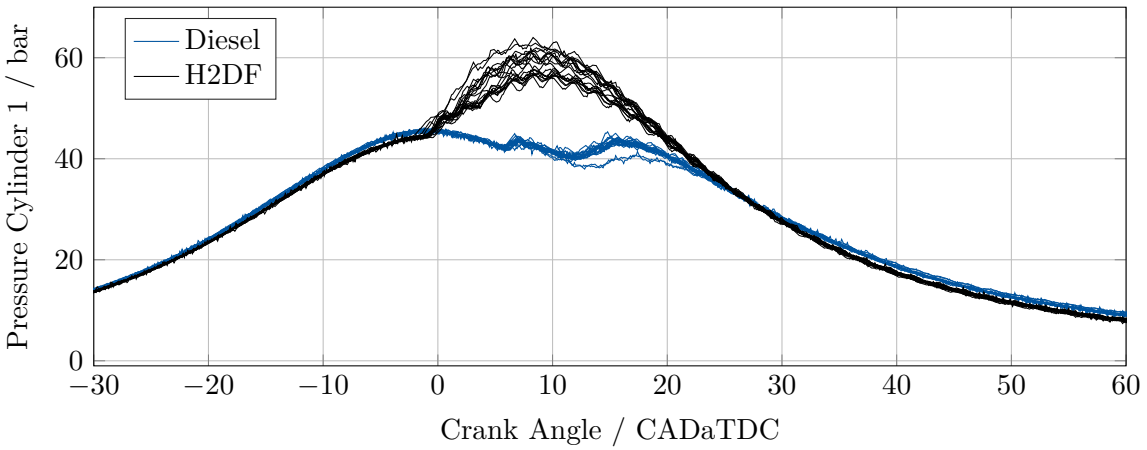


Figure 3.3: Cylinder pressure around TDC of H2DF and diesel combustion for 20 cycles at fixed p_{mi} (6 bar), $n_{\mathrm{Eng}} = 1500\,\mathrm{min}^{-1}$.

and 62% and is calculated according to Eq. 2.3. The required injected fuel masses are determined using injector models developed by McNally [113].

As partially observed in Fig. 3.3, the data in Fig. 3.4 further demonstrate that increasing the HES leads to substantial increases in $p_{\max}$, p_{mi}, and $dp_{\max}$. At lower HES levels (0% to 25%), the combustion increases more gradually, resulting in moderate peak pressures. In contrast, at higher HES levels (40% and above), the pressure rise becomes significantly steeper, and peak pressures increase sharply, indicating faster and more intense combustion occurring closer to TDC.

These observations underline the importance of precise control in dual-fuel operation to maintain performance and stability, while keeping the cylinder pressure within safe limits. Moreover, the nonlinear behavior observed reinforces the need for fast, C2C capable control systems that can effectively manage these dynamic effects. Following this C2C capability, it is important to evaluate the available calculation time for such a control system.

Fig. 3.5 thus visualizes the injection timings and control setpoints over one combustion cycle. The combustion metrics, calculated in real-time on the FPGA as described in Section 2.1, are completed by 165 CADaTDC [117]. While the setpoint for the diesel injection is scheduled relatively late at 650 CADaTDC, the setpoint for the hydrogen PFI occurs much earlier, at 330 CADaTDC. For this work, the engine is primarily operated at a constant speed of $n_{\mathrm{Eng}} = 1500\,\mathrm{min}^{-1}$, corresponding to an engine cycle duration of 80 ms. Consequently, only 165 CAD or approx. 18 ms are available to compute the control variables within each cycle (t_{calc}). This tight time window presents a major challenge for the computational efficiency of the controllers developed in this work. Advanced control strategies such as nonlinear model predictive control (MPC), which rely on iterative optimization, are particularly demanding given the limited

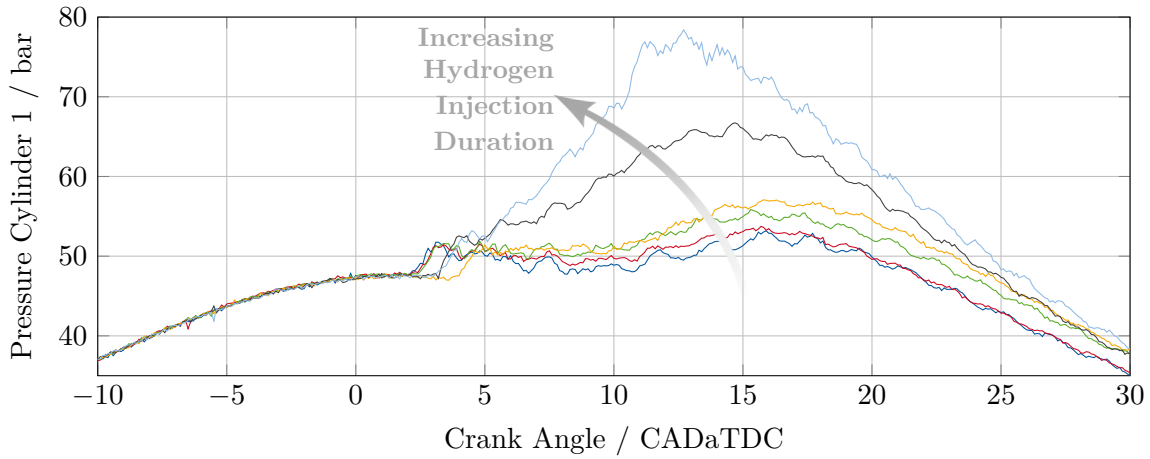


Figure 3.4: Cylinder pressure evolution around TDC for H2DF and diesel operation at $n_{\mathrm{Eng}} = 1500\,\mathrm{min}^{-1}$ with qualitatively increasing hydrogen injection duration, while diesel timing remained constant. This qualitative illustration is not energy- or load-neutral.

computational resources available in automotive ECUs.

The following chapters present a data-driven control framework for H2DF engines, starting in the next chapter with the development of a deep neural network (DNN)-based engine model for future integration into a nonlinear MPC. The overarching goal is to tackle two key challenges identified in this chapter: enabling real-time implementation and ensuring safe operation in the presence of complex, highly nonlinear engine dynamics of H2DF combustion.

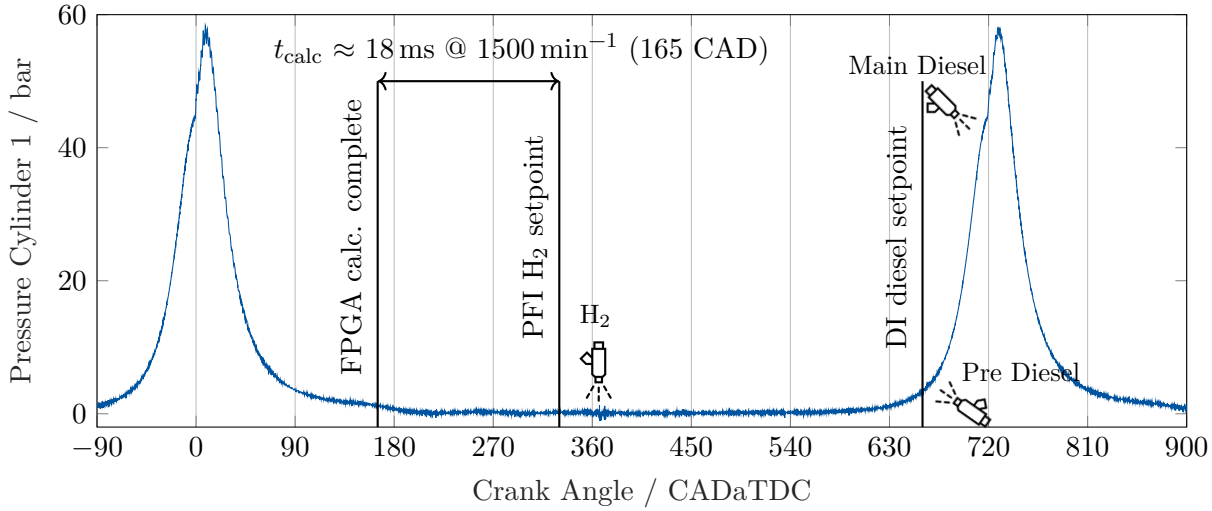


Figure 3.5: Cylinder pressure trace with injection timing events at $n_{\text{Eng}} = 1500\,\text{min}^{-1}$

4 Deep Neural Network-based Engine Model

This chapter presents the development of a deep neural network (DNN)-based engine plant model, intended for future integration into a model predictive control (MPC). The goal is to address the two main challenges identified in previous chapters: achieving real-time capability and enabling safe control of the highly nonlinear dynamics of the hydrogen diesel dual-fuel (H2DF) combustion. Meeting these demands requires a trade-off between computational efficiency and prediction accuracy, limiting the DNN's depth and architecture.

The following section provides a brief literature review on internal combustion engine (ICE) modeling approaches in both academia and industry, and outlines the structure and contribution of this chapter.

The chapter ends with a brief summary and a concluding discussion that positions the presented DNN-based modeling approach within the broader scope of the thesis.

This chapter is partially based on [32], [33], [35], [38], [43], [44], [46], [118], [119], [47].

4.1 Introduction to the Chapter

Despite decades of academic progress in combustion modeling and model-based engine control, production engines continue to rely heavily on conventional strategies such as look-up-tables (LUT) and proportional-integral (PI) controllers. Industrial engine control typically combines feedforward and feedback strategies, where LUT serve as calibration maps and PI controllers dominate feedback loops [28], [120]. These approaches are robust and well-understood but require extensive manual calibration and can fail to fully exploit engine potential [29]. Powerful modeling options which are used for offline ICE development are high-fidelity simulation tools such as *GT-Power* [121] or three-dimensional computational fluid dynamics-based tools [122], [123] provide detailed insights but are computationally expensive and thus impractical for real-time deployment.

Academic research, by contrast, has advanced dynamic, model-based techniques – including mean-value engine models by Hendricks and Sorenson [124] (1990) and Saerens et al. [125] (2008), and more recently, machine learning (ML) by Norouzi et al. [32] (2022). Although these advanced methods offer improved adaptability and performance, their adoption in industry remains limited due to the simplicity and proven reliability of traditional solutions [28]. Additionally, a major barrier to deploying ML-based methods in industry is that certification frameworks for ML systems are still lacking [126].

In this context, ML offers a promising alternative. ML models, particularly artificial neural networks (ANNs), enable the development of fast, accurate, and compact surrogate models that can reduce calibration effort, save test bench time, and thus lower associated carbon emissions. Crucially, these models also reduce dependency on experienced combustion engineers.

Recent research has demonstrated that ANNs can effectively model engine performance and emissions: Rezaei et al. [127] (2015) employed radial basis functions and feedforward neural networks (FFNN) on experimental homogeneous charge compression ignition (HCCI) engine data to predict key combustion metrics across several steady-state point, while Syed et al. [128] (2017) applied a FFNN to capture brake thermal efficiency and emissions in a H2DF engine. More recently, Molina et al. [129] (2021) trained a ANN on laminar flame speeds to simulate methane–hydrogen spark ignition (SI) engine combustion within a virtual engine, and Shin et al. [130] (2020) developed a DNN to simultaneously predict knock occurrence alongside performance and emission outputs in a gasoline engine.

However, these applications remain largely confined to offline tasks due to concerns regarding real-time performance constraints. The author and the research groups at RWTH and the UofA have applied recurrent neural networks (RNNs), particularly long short-term memory (LSTM) architectures, for modeling both HCCI and diesel engines [32], [33], [118]. These models integrated into MPC reached real-time feasibility but still faced limitations due to their structural complexity and computational load.

Building on these efforts, the current work introduces a control-oriented engine model tailored for the H2DF combustion process. Owing to the fast and highly nonlinear nature of hydrogen combustion, conventional modeling approaches are insufficient. To address this, a gated recurrent unit (GRU)-based architecture is proposed. As discussed in Section 2.3, GRUs provide similar prediction accuracy to LSTMs but are computationally more efficient due to their reduced number of internal memory states.

The overall development workflow is outlined schematically in Fig. 4.1.

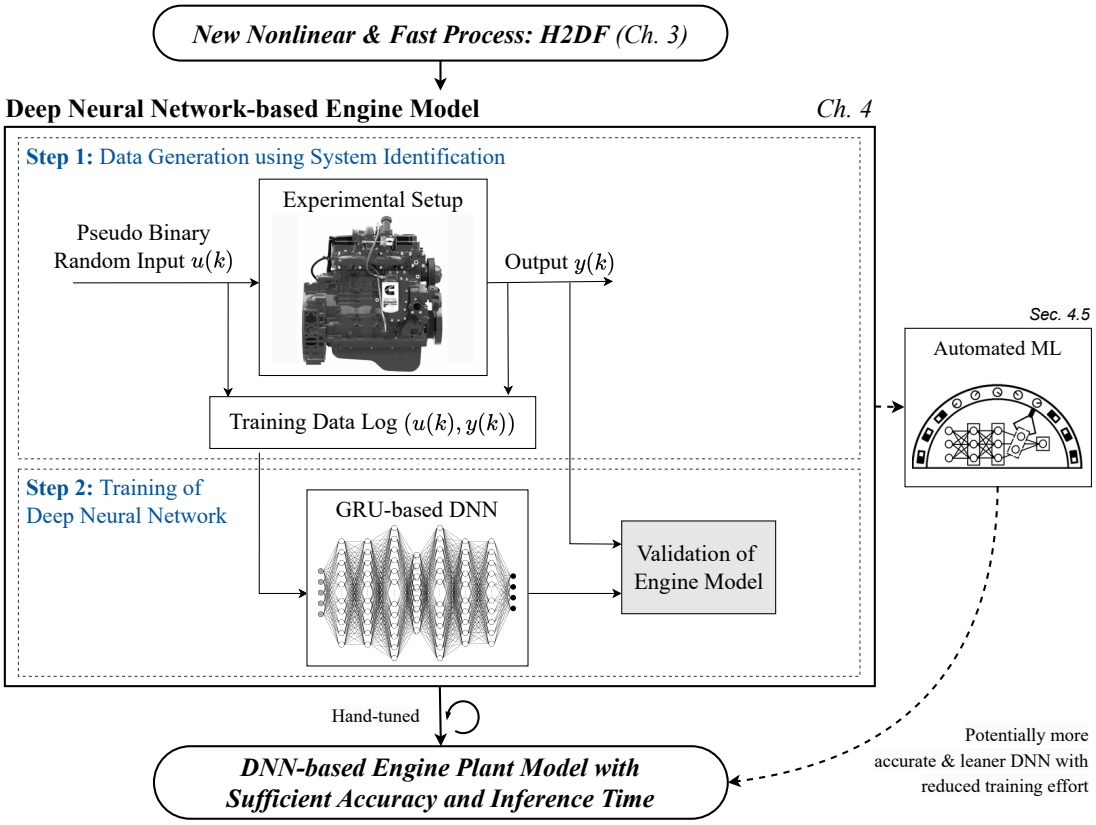


Figure 4.1: Development process of DNN-based engine model for H2DF combustion engine. AutoML logo taken with permission from [31].

The process begins with the selection of input and output features and the presentation of the empirically identified DNN architecture. This is followed by data collection using a pseudo-random binary sequence (PRBS) to excite the system and capture relevant engine behavior and dynamics. The collected data is then used for model training including dataset distributions and validation. Finally, the use of automated machine learning (AutoML) is demonstrated as an additional method to efficiently address the multi-objective task of finding a network that is both lightweight and accurate while using optimized training hyperparameters.

4.2 Model Features & Network Architecture

In the following section, the DNN's input and outputs features and its network architecture are presented.

The input features of the DNN, denoted as $X_{\mathrm{DNN}}(k)$, are defined as:

$$X_{\mathrm{DNN}}(k) = \left[t_{\mathrm{Main}}(k), t_{\mathrm{P2M}}(k), \alpha_{\mathrm{Main}}(k), t_{\mathrm{H2}}(k), p_{\mathrm{mi}}(k-1)\right]^T, \tag{4.1}$$
$$X_{\mathrm{DNN}}(k) \in \mathbb{R}^5.$$

To model the engine as a dynamic system, the DNN uses the engine control variables $u(k)$, as defined in Table 3.2, as its primary inputs. These are further augmented by the indicated mean effective pressure value p_{mi} from the previous timestep, which serves as a feedback signal for the most relevant tracking variable. This addition enables the developed controllers to operate in a closed-loop feedback configuration, which is expected to improve tracking performance. To reduce model complexity, as motivated in Section 3.3, only a subset of available control variables is included.

The DNN's output features $Y_{\mathrm{DNN}}(k)$ are defined following Table 3.1:

$$\begin{aligned} Y_{\mathrm{DNN}}(k) &= \left[p_{\mathrm{mi}}(k), c_{\mathrm{NOx}}(k), c_{\mathrm{PM}}(k), dp_{\max}(k)\right]^T, \\ Y_{\mathrm{DNN}}(k) &\in \mathbb{R}^4. \end{aligned} \tag{4.2}$$

As discussed in Section 2.1 and Section 3.4, the most important outputs of the engine are p_{mi} for load tracking and the maximum cylinder pressure rise rate $dp_{\max}$ to avoid excessive engine wear and prevent knocking or high noise. In addition, to reduce the emissions output, the concentration of c_{NOx} and particulate matter c_{PM} are predicted by the DNN engine model.

In previous works, LSTM-based architectures were used successfully [32]. As discussed in Section 2.3, GRUs can capture temporal dependencies with fewer parameters than LSTM units by retaining a history across hidden states. Motivated by these advantages, this work adopts a GRU-based architecture to model the process dynamics. Fig. 4.2 shows the architecture of the DNN.

The empirically identified network consists of multiple layers: the input layer, several fully connected (FC) layers utilizing rectified linear unit (ReLU) activation functions, surrounding a GRU layer of size $n_{\mathrm{GRU}} = 8$ in an encoder-decoder structure, and the output layer. Incorporating FC layers improve the DNN's capabilities to predict nonlinear behavior without increasing the number of sample and computationally inefficient internal states of the GRU.

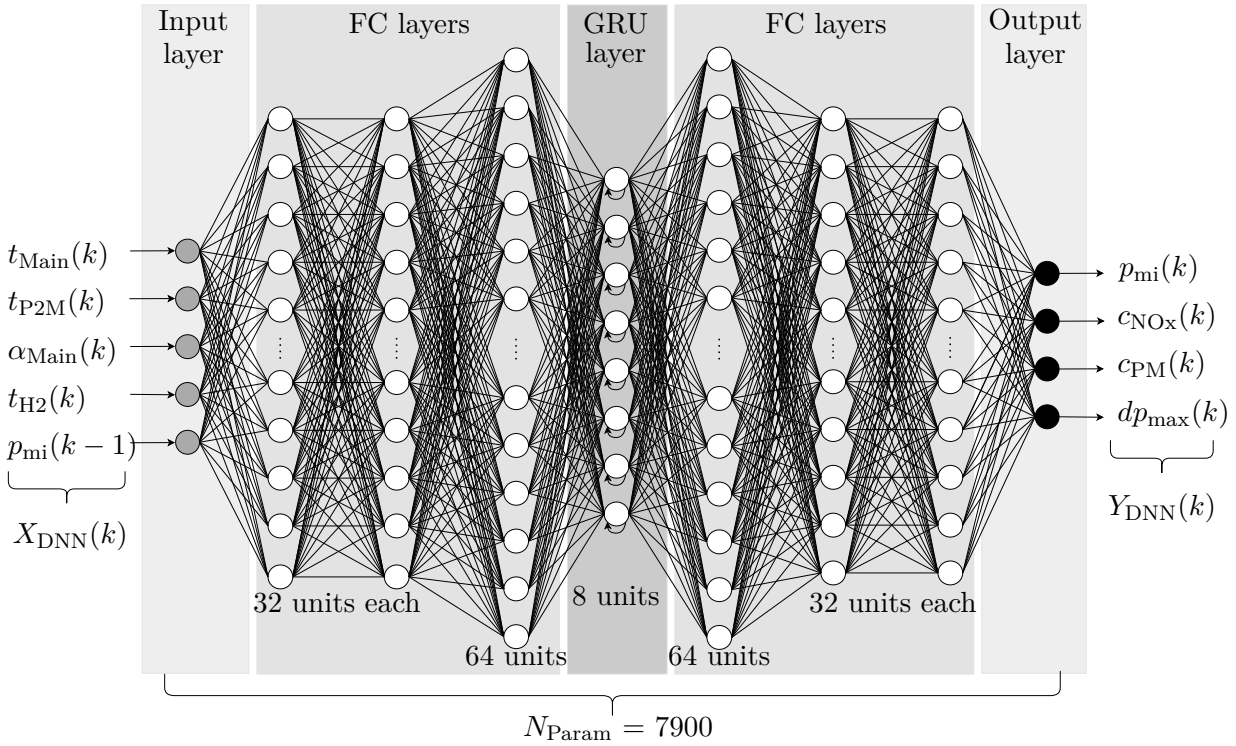


Figure 4.2: DNN engine model architecture with feedback. The fully connected (FC) layers are using rectified linear unit (ReLU) activation functions.

4.3 Data Collection

By collecting experimental data on the test bench, a dataset comprising 99,800 consecutive engine cycles or samples N is generated. During these engine cycles, all four engine control inputs defined in Eq. 4.1 are varied randomly following the PRBS. A bounded random signal is employed to modify both the amplitude and frequency of these four inputs. System-level safety constraints on dp_{max} and p_{max} ensure that the random actuations do not cause engine damage. The boundaries and settings of the randomizer are depicted in Table A.6. It is worth noting that the engine used in this work is considered robust; such (bounded) random process actuation might not be safe or feasible with other real-world systems such as HCCI.

Fig. 4.3 shows the measured engine inputs and outputs in the time-domain:

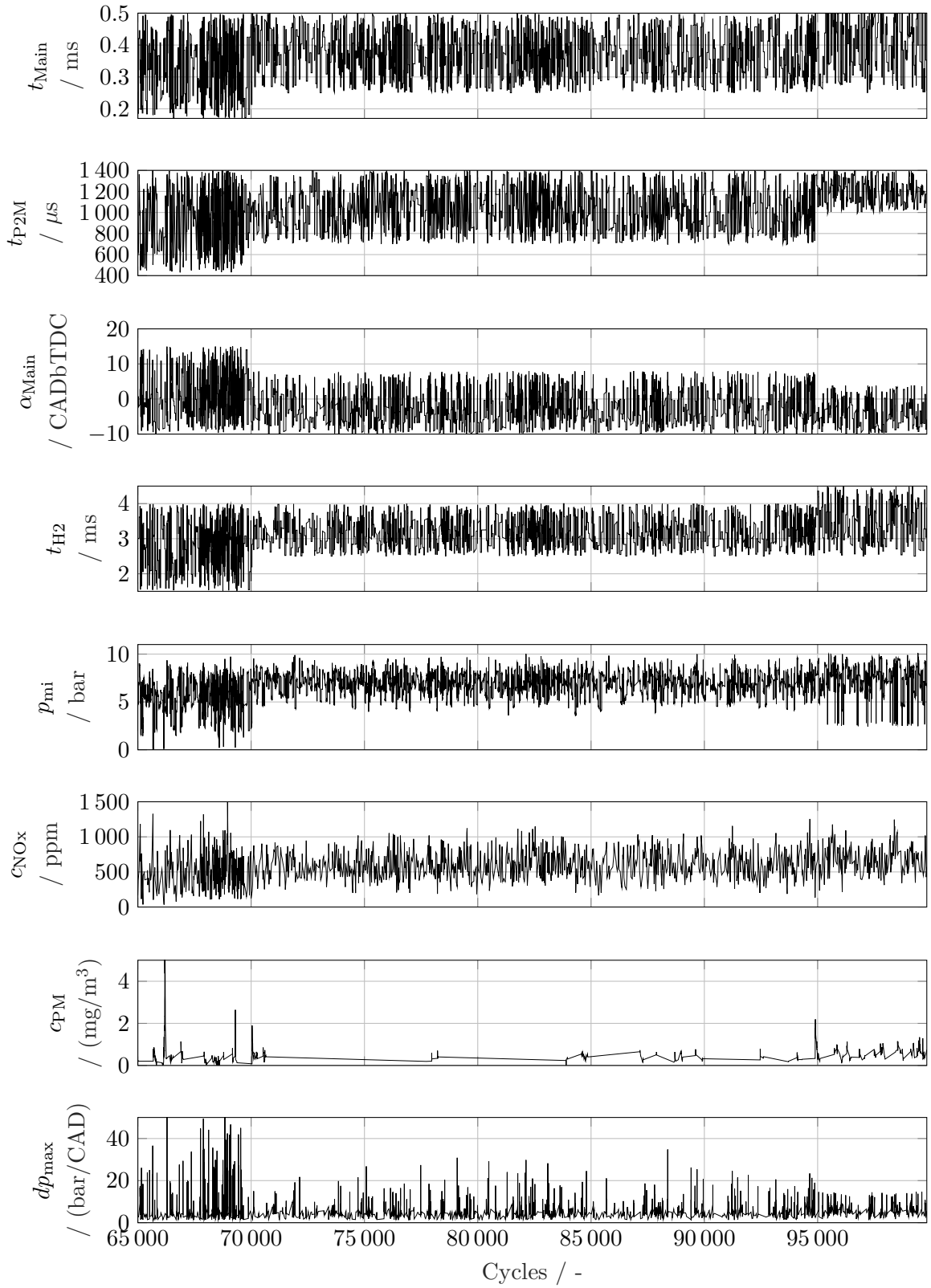


Figure 4.3: Experimental PRBS generated engine inputs in time-domain at $n_{\mathrm{Eng}} = 1500\,\mathrm{min}^{-1}$. 99,800 total engine cycles. Only cycles from 65,000 onward are shown here due to the large data volume.

Due to the large data volume, only cycles from 65,000 onward are shown in Fig. 4.3. The data collection required a net engine runtime of approximately 130 min across three separate measurement runs.

To achieve an evenly distributed coverage of the dataset, the PRBS input boundaries were adjusted over the course of the three runs. In particular, the lower bounds of the main duration of injection t_{Main} and the hydrogen duration of injection t_{H2} were increased to produce more samples in higher p_{mi} regions over 7.5 bar. To ensure stable combustion and avoid knocking at these higher load levels, the upper limit of the main start of injection (SOI, α_{Main}) was reduced, while the lower bound of the pre-to-main time t_{P2M} was increased simultaneously. Subsequently, to better capture varying system dynamics, the update frequency of the PRBS inputs was increased during the final measurement run.

These iterative adjustments reflect the prototypical and exploratory nature of this work, highlighting one advantage of data-driven controller development. With little process knowledge and time, the data generation using this approach can be changed online while running the engine, ensuring varying dynamics and output distributions.

Each of the three measurement runs was divided into training, validation, and test datasets using a fixed split of 80%, 15%, and 5%, respectively. Since GRU units rely on temporal patterns, the sequence of data had to be preserved – so no shuffling of the data was conducted. The resulting subsets from all runs were then concatenated to form the final training, validation, and test datasets used for model development.

The distribution of the IMEP output (p_{mi}), which represents the primary variable for load tracking, is shown in Fig. 4.4.

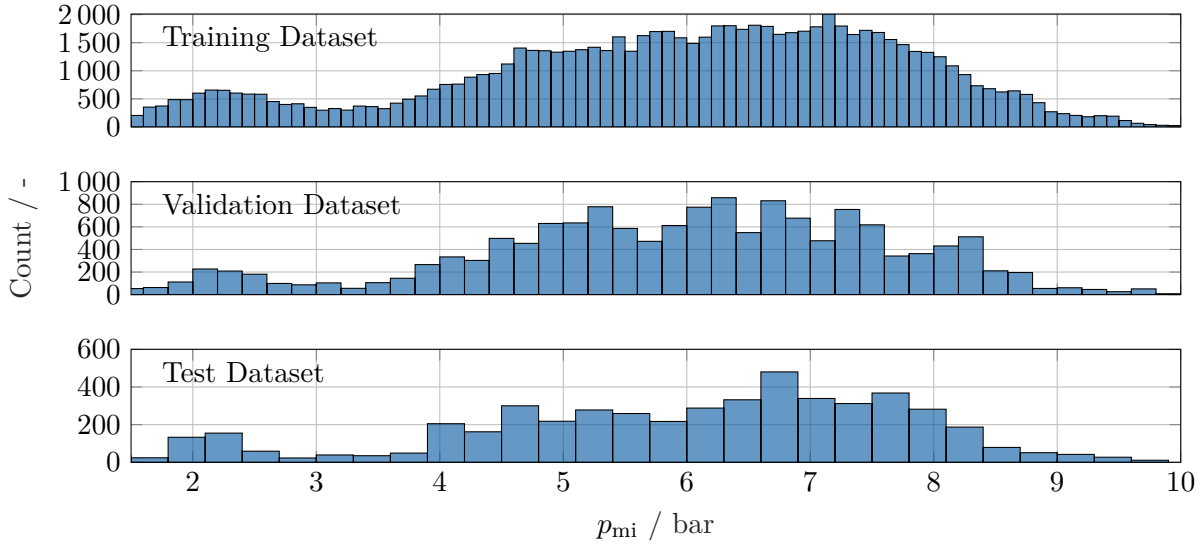


Figure 4.4: Experimental PRBS generated engine data distribution for p_{mi} at $n_{\mathrm{Eng}} = 1500\,\mathrm{min}^{-1}$. Upper plot: training dataset (80% of total data), mid plot: validation dataset (15% of total data), lower plot: unseen test dataset (5% of total data).

The figure demonstrates that the distributions across the different datasets are consistent over the full p_{mi} range, ensuring reliable comparability for training and evaluation. This distribution supports robust model generalization and tracking performance across varying operating conditions. Naturally, due to the PRBS approach, within the p_{mi} range there are areas present with less data, such as at approx. 3 bar, while the majority of data lies within the range of 5 bar to 8 bar. The dip at approx. 3 bar can additionally be attributed to the diesel injector operating in its ballistic region at short main-injection durations. The datasets exhibit two normal distributions in p_{mi}, one slightly above 2 bar and another near 6 bar. This bimodality results from the random variation of the PRBS approach of hydrogen injection durations between 1.5 ms and 4.5 ms. Due to hardware limitations, injections shorter than approximately 2 ms did not always fully open the valve, leading to additional samples of diesel-only operation. Hence, the lower-pressure cluster corresponds to pure diesel combustion, while the higher-pressure cluster represents dual-fuel operation.

The data for the remaining outputs (see Eq. 4.2) is of similar distribution and depicted in Fig. A.2, Fig. A.3 and Fig. A.4.

4.4 Model Training & Validation

After data collection and distribution into training, validation, and test sets, model training is conducted exclusively on the training set, while the validation set is used for performance monitoring and model selection. The final model is then evaluated on the independent test set to assess its generalization capability. The training is conducted in a `MATLAB/Simulink` environment using the `Deep Learning Toolbox` with the hyperparameters as referenced in Table A.7. Further explanations to the training hyperparameters are detailed in Section 2.3. These hyperparameters were identified manually, relying on extensive empirical testing and experience.

The RMSE is chosen as the evaluation metric or training objective to be minimized (Eq. 2.9). The entire dataset is normalized with each variable's range, rather than standardized (more common in ML), to improve numerical properties when integrated into the MPC. Besides, all results and values in the following are displayed denormalized in the physical domain for better interpretation.

The resulting loss-epoch plot of the training is shown in Fig. 4.5. The DNN demonstrates stable convergence with minor oscillations, as both training and validation losses decrease over time. Starting around epoch 2000, a slight increase in validation loss indicates the start of overfitting. To prevent this, the final model is selected based on the lowest recorded validation loss at epoch 2000, ensuring good generalization performance.

For the training-independent validation of the model on the unseen test dataset, multiple performance or error metrics are introduced in Table 4.1. For improved physical interpretability (as explained in Section 2.3), both the MAE and RMSE (Eq. 2.11, Eq. 2.9) are displayed. Additionally, the clipped NRMSE ($\mathrm{NRMSE}_{\mathrm{clipped}}$, Eq. 2.10) and normalized MAE using a clipped

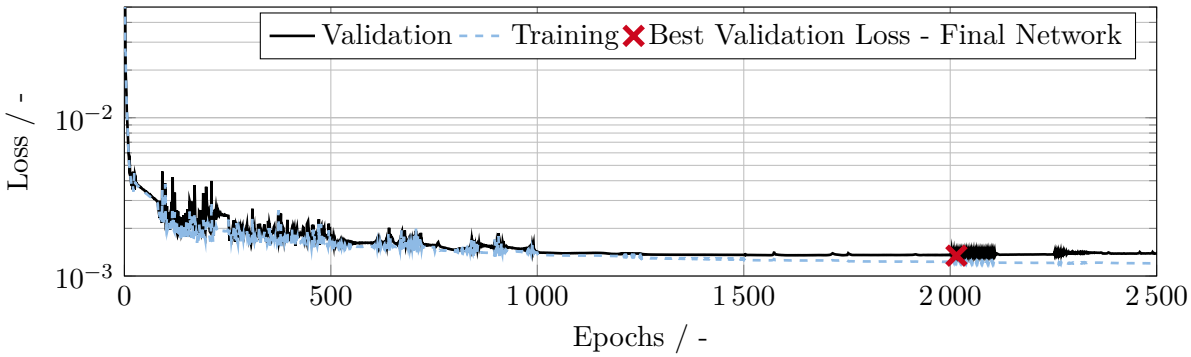


Figure 4.5: Loss-epoch plot for the DNN training. The final network is chosen according to the best validation loss.

range (NMAE, $\text{NMAE}_{\text{clipped}}$) are used to compare prediction accuracy across different output variables on a consistent scale. Here, the clipping is applied by limiting values to the first and 99th percentiles, removing outliers, which would otherwise inflate the normalization range and artificially enhance the apparent model performance. The model shows strong accuracy for p_{mi}, with low MAE and RMSE values, indicating reliable tracking performance across both validation and test sets. Predictions for c_{NOx} also achieve good accuracy, with consistent $\text{NRMSE}_{\text{clipped}}$ below 5%, suggesting the model captures the emission trends well. Soot prediction performance is more variable with an $\text{NRMSE}_{\text{clipped}}$ of 9.2% on the unseen test dataset. This behavior is attributed to the low absolute sensor accuracy of the *Pegasor* soot sensor. The sensor is specifically focused on catching trends in soot emissions, while generally an accurate absolute measurement of soot emissions is considered difficult. Furthermore, the *Pegasor* in general is quite prone to larger reading errors as described in Section 3.2, making accurate learning and generalization challenging for this variable. Nevertheless, its ability to capture transients with its low response time made this sensor a good choice for this work, due to the reliance of control-oriented models

Table 4.1: DNN-based engine plant model. Prediction performance on validation and unseen test datasets. Clipped values are normalized by the range between the first and 99th percentiles to avoid outliers.

Metric	**Validation**	**Test**
p_{mi}		
MAE / bar	0.14	0.12
$\text{NMAE}_{\text{clipped}}$ / –	1.8%	1.6%
RMSE / bar	0.35	0.23
$\text{NRMSE}_{\text{clipped}}$ / –	4.5%	3.0%
c_{NOx}		
MAE / ppm	25	26
$\text{NMAE}_{\text{clipped}}$ / –	2.5%	2.6%
RMSE / ppm	35	40
$\text{NRMSE}_{\text{clipped}}$ / –	3.5%	4.0%

Metric	**Validation**	**Test**
c_{PM}		
MAE / mg/m^3	0.13	0.10
$\text{NMAE}_{\text{clipped}}$ / –	8.6%	6.5%
RMSE / mg/m^3	0.34	0.14
$\text{NRMSE}_{\text{clipped}}$ / –	22.6%	9.2%
dp_{max}		
MAE / bar/CAD	0.86	0.89
$\text{NMAE}_{\text{clipped}}$ / –	6.7%	6.9%
RMSE / bar/CAD	1.89	1.56
$\text{NRMSE}_{\text{clipped}}$ / –	14.7%	12.1%

(especially in MPC) on precise gradient prediction. Furthermore, the $dp_{\max}$ predictions show moderate accuracy, with higher errors and lower values compared to other outputs. Nonetheless, the absolute MAE values remain below 1 bar/CAD, which is still acceptable considering the large range.

The prediction results of the final network on the unseen test dataset are additionally depicted in a parity plot in Fig. 4.6, which confirm the metrics shown in Table 4.1. This visualization illustrates the prediction accuracy and regression performance of the DNN model. Ideally, all data points would lie along the 45-degree line (in blue, "ideal fit"), which is closely followed by the predictions for p_{mi} and c_{NOx} emissions. In contrast, the predictions for soot c_{PM} and $dp_{\max}$ exhibit greater scatter and deviate more noticeably from the ideal fit line. For c_{PM} in particular, the model struggles to predict higher values accurately. This can be attributed to the limited number of data samples in those regions (see Fig. A.3), as well as the lower accuracy of the soot measurement sensor (see Section 3.2).

The presentation of the network predictions in the time-domain on the unseen test dataset in Fig. 4.7 underlines the good performance of the chosen DNN. In contrast to parity plots, which offer no insight into how predictions evolve over time, this time-domain visualization shows that the DNN is able to follow the overall trends of all outputs well – an essential requirement for use in MPC. This becomes especially clear for the c_{PM} and $dp_{\max}$ signals, where the network manages to capture the dynamic behavior well, even if the point-wise accuracy is lower, as

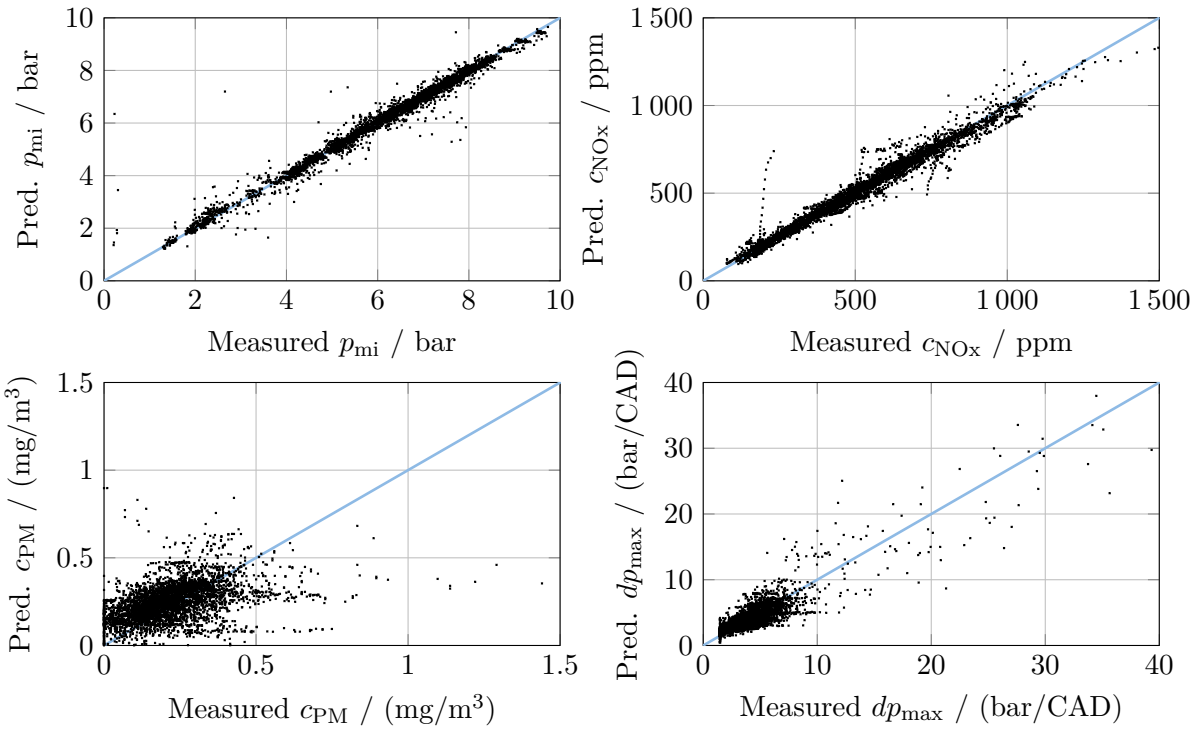


Figure 4.6: DNN-based engine plant model. Predicted outputs vs. experimental unseen test dataset.

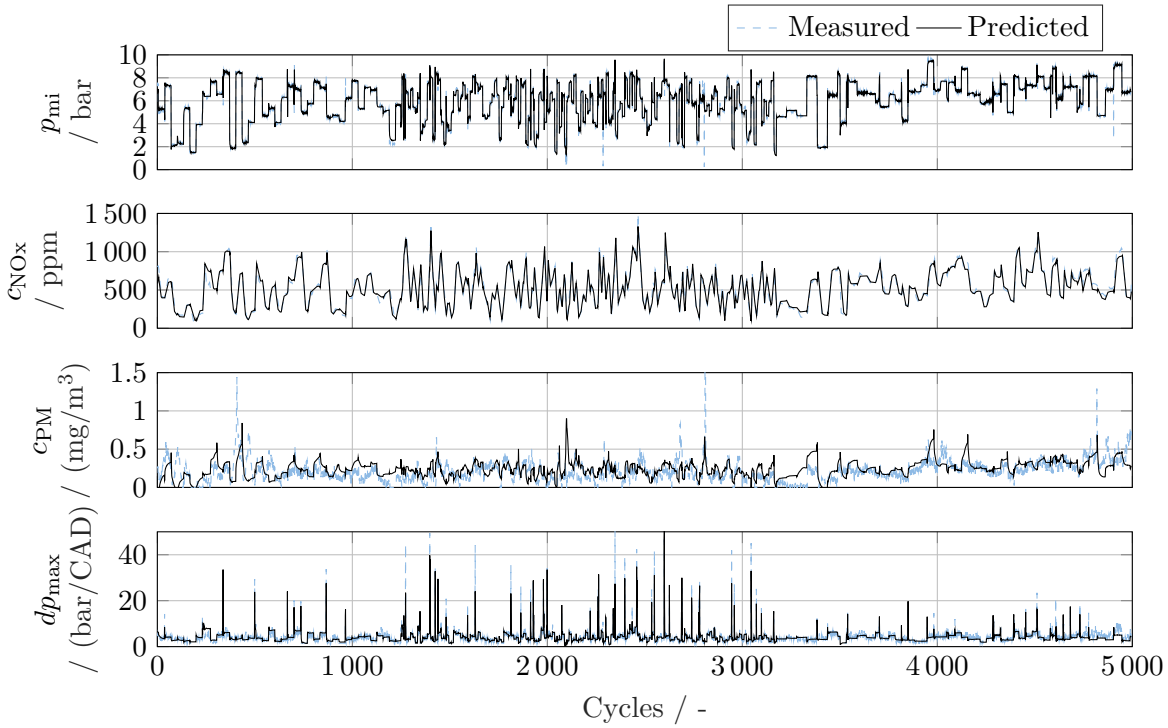


Figure 4.7: DNN-based engine plant model. Predicted outputs on the experimental unseen test dataset in time domain.

discussed previously.

The DNN model used in this work features hidden states that evolve with the current engine output, allowing the dynamics model to adapt to changing engine conditions. To further visualize this, the evolution of those hidden states is shown in Fig. 4.8. These states do not represent physical quantities but serve as internal memory elements arising from the model's structure, enabling the network to capture temporal dependencies. Unlike static FFNN structures, the model dynamically updates its hidden states based on the current inputs. This behavior highlights both the adaptive nature and the high dimensionality of the model.

The development of the DNN model proved to be a time-consuming and iterative process. The network architecture and hyperparameters were selected manually, based on experience and trial-and-error, without any optimization. Consequently, the final model size and subsequently, the expected computational load was estimated intuitively, without any foresight regarding runtime performance when integrated into the MPC or deployed on an embedded system.

This challenging and manual search for a suitable DNN architecture can be significantly improved using automated methods such as hyperparameter optimization. The following section introduces AutoML as one such approach. The aim is to demonstrate how this process can be streamlined by systematically exploring different DNN architectures and training hyperparameters, with the goal of balancing prediction accuracy and computational efficiency.

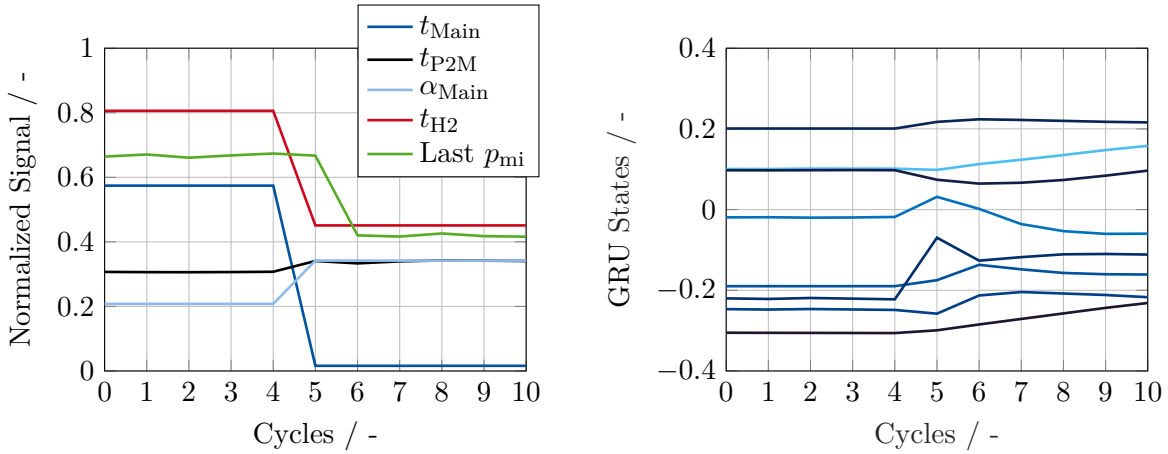


Figure 4.8: DNN-based engine plant model. GRU hidden state activation on extract of test data. Left: DNN inputs (controls), Right: GRU states under input changes.

4.5 Automated Machine Learning

Following the fundamentals outlined in Section 2.3, this section presents the application of AutoML for optimizing the DNN engine model development introduced earlier. Instead of relying solely on manual and empirical tuning, the AutoML process is used to automatically explore different network architectures and training hyperparameters in a systematic and efficient way. The goal is to enhance the model development workflow by replacing expert knowledge and trial-and-error with a structured, automated search.

The focus in this section firstly lies on the study setup and design, where the optimization algorithm, the corresponding search space and the data preprocessing are defined. Furthermore, the relevant performance metrics are selected and the hardware and software setup are described.

The experiments are conducted using the same dataset and baseline DNN model as in the manual development process.

Study Design: Search Space & Algorithm

Identifying an appropriate search space and strategy is essential for AutoML. Human expertise is employed to manually tune key parameters, narrowing the search space to balance exploration and exploitation effectively. Table 4.2 summarizes the selected search space. It distinguishes between network- and training-related hyperparameters, with the former defining the architectural constraints for the neural architecture search (NAS). In this study, the previously used model structure utilizing the encoder-decoder principle shown in Fig. 4.2 was used as a baseline. While the GRU layer was fixed in the center of the DNN (apart from the number of GRU units), the number of FFNN layers and units before and after the GRU layer were set as network-related hyperparameter. Training-related hyperparameter further restrict the scope of the hyperparameter optimization (HPO). To reduce search space and focus on key aspects, parameters were

Table 4.2: AutoML study. Definition of search space for DNN engine model architectures.

Hyperparameter	Variable type	Range
Network related		
Number of hidden layers before GRU	Categorical	{1, 2, 3, 4}
Number of hidden layers after GRU	Categorical	{1, 2, 3, 4}
Number of hidden units in FC layers	Categorical	{8, 16, 32, 64}
Number of hidden units in GRU layer	Categorical	{2, 4, 6, 8}
Training related		
Learning rate	Real number	[1e-3, 1e-4], 50% drop every 500 epochs

fixed to previously used values: the Adam optimizer was selected, and the L2 regularization factor was set to 0.1, while the mini-batch size was set to 512. The learning rate was the only free training-related hyperparameter. A learning rate scheduler was implemented: an initial value, chosen as a real number between 1e-3 and 1e-4, was reduced by 50% every 500 epochs. An early stopping with a patience of 500 epochs and hyperband pruning (see Section 2.3 for background) were applied to efficiently allocate resources, terminating stagnant models early while allowing promising candidates to continue training. The total size of the hyperparameter search space can thus be determined by considering all independent combinations of layer counts and units. Since the number of hidden units could be selected individually for each hidden layer before and after the GRU block, the size grows exponentially. Thus, the total search space amounts to 462,400, excluding the continuous initial learning rate (see Eq. A.2 for details).

Exploring such a large search space presents a significant challenge, making the selection of an efficient search strategy crucial. As discussed in Section 2.3, bayesian optimization (BO) with a tree-structured parzen estimator (TPE, in combination: BO-TPE) is particularly well suited for this task.

Hardware & Software Configuration

The AutoML tasks were performed on the *CLAIX-2023*-HPC cluster at RWTH Aachen University[1] [131]. The configuration consisted of four compute nodes, each equipped with two *Intel Xeon 8468 Sapphire* processors (48 cores each, totaling 96 cores, 512 GB of main memory). Runtime, memory usage, energy consumption, and performance metrics were continuously monitored via an integrated `Grafana` [132]-based monitoring system.

The DNNs were implemented and trained using `TensorFlow` [133] and `Keras` [134] in `Python` [135] within `apptainer` [136] containers. Multi-threaded job submissions and resource management were handled via batch scripts and the `SLURM` [137] workload manager. For hyperparameter optimization, the `Optuna` framework [138] was used, leveraging its `JournalStorage` feature to

[1]Computational resources and support were provided by the IT Center of RWTH Aachen University under project number `rwth1479`, which is gratefully acknowledged.

enable distributed optimization across multiple nodes. `Optuna's` flexibility, scalability, and ease of integration made it particularly suitable for the complex, high-dimensional search spaces encountered in this study.

Data & Feature Selection & Preprocessing

The dataset used for the AutoML study corresponds to the same data described previously in Chapter 4.3. Instead of providing three static datasets as in earlier sections, a ($k = 5$)-fold cross-validation strategy is employed to better assess model generalization. Cross-validation partitions the data into k equally sized folds, using $(k-1)$-folds for training and the remaining fold for validation, cycling through all partitions. This method provides a more robust evaluation of model performance at the expense of significantly increased computational overhead. An explanatory schematic of the cross-validation setup is shown in Fig. A.9.

The feature and model structure selection remains identical to that outlined in Chapter 4. The full list of input and output variables is given in Eq. 4.1 and Eq. 4.2, respectively.

Performance Metrics & Evaluation Criteria

The evaluation of the DNN architectures in this study is based on two primary performance metrics: the prediction performance and the DNN's computational efficiency.

Prediction performance is quantified by the generalization error (GE), an advanced metric implemented in `TensorFlow`. GE is the gap between true expected loss (here: using the average RMSE) and empirical training loss, quantifying predictive performance on unseen data already in training [133]. An advantage of this metric is that it directly measures a DNN's ability to generalize beyond its training data [139].

In addition to prediction accuracy, network computational efficiency is considered by evaluating the number of floating point operations (FLOPs) required for a forward pass (see Eq. 2.13 in Section 2.3). Together, these two metrics establish a multi-objective optimization problem, aiming to balance model accuracy and computational complexity.

Results & Analysis

The AutoML run was conducted using four cluster nodes of the `CLAIX-2023` cluster. Each core executed one instance of a DNN training job, resulting in up to 384 parallel training processes.

Fig. 4.9 shows the results of the multi-objective optimization study, plotting GE against FLOPs. Each black dot represents a single solution from the total set of 27,000 evaluated configurations. The Pareto front is highlighted, along with the ideal points for each objective individually and the best Pareto-optimal solution, selected based on the shortest Euclidean distance to both ideal points (FLOPs: 4024 & GE: 0.00375). The inset provides a zoomed-in view of the region around the Pareto front, illustrating the trade-off between accuracy and computational cost in detail. See Table A.9 for the values and detailed network architecture. For comparison, the empirically

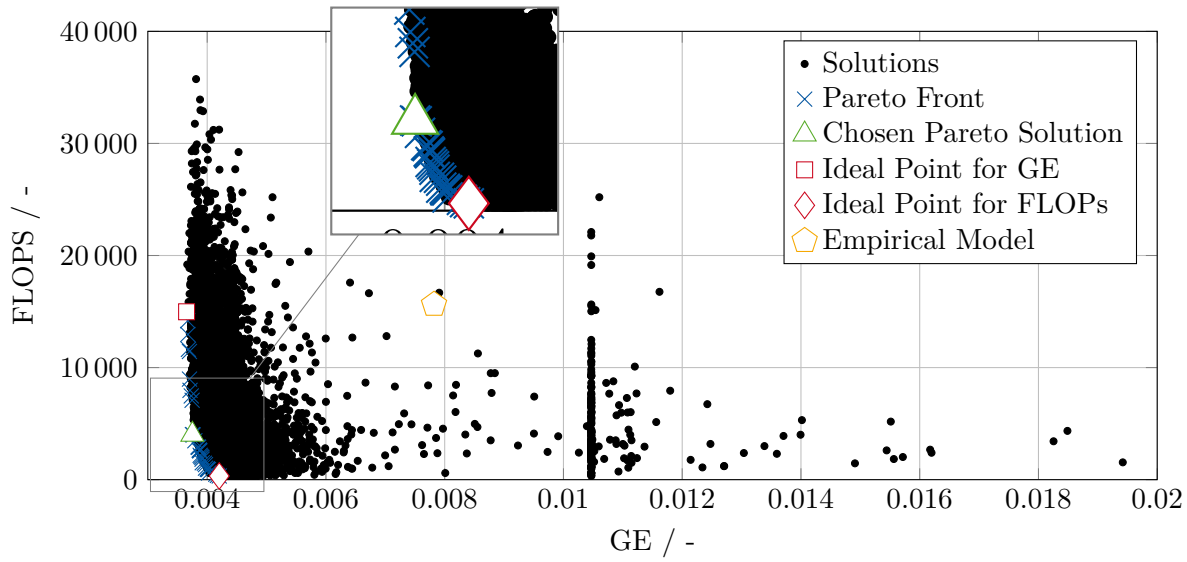


Figure 4.9: `Optuna` AutoML multi-objective optimization results. Prediction performance (GE) vs. network computational performance (FLOPs).

designed DNN model of Section 4.2 is also depicted: the chosen Pareto-optimal solution achieves a 52% better GE while requiring 74% fewer FLOPs than the empirical model, underlining the effectiveness of the AutoML-based architecture search.

The initial manual designs employed multiple FC layers before the GRU for dimensionality extraction. In contrast, the AutoML study consistently favored architectures with only a single pre-GRU layer, suggesting that deeper preprocessing may impair temporal prediction capabilities.

The full AutoML experiment ran for approximately 74 h, consuming around 203 kW h of energy, while a direct comparison to manual tuning on local machines is difficult; a simple estimate shows that continuous operation at 150 W for three weeks results in approximately 75 kW h. Thus, despite the scale and additional overhead from cross-validation, the AutoML study achieved efficient energy use.

4.6 Summary of the Chapter & Discussion

This chapter addressed the challenge of accurately modeling the highly nonlinear dynamics of the H2DF combustion, for which no established modeling approaches exist. The goal was to develop a computationally lean yet sufficiently accurate engine plant model, suitable for future integration into an MPC.

The proposed approach follows a structured data-driven workflow: starting with a selection of input and output features, followed by the proposal of the DNN model architecture using FFNN and GRU layers in an encoder-decoder structure. Data was collected from over 99,000 engine cycles using a PRBS excitation scheme, ensuring diverse operating conditions. The

model demonstrated high prediction accuracy across key engine outputs, including the engine load p_{mi}, the NO_x and PM concentrations c_{NOx} and c_{PM}, and the maximum pressure rise rate dp_{max}. While slightly reduced accuracy was observed for outputs such as PM and dp_{max}, the model proved particularly effective in capturing dynamic behavior and gradient information on all outputs: an essential requirement for control-oriented applications such as MPC.

To address the additional requirement of the DNN model's computational efficiency (crucial for real-time control), AutoML was applied. This allowed efficient navigation of the multi-objective trade-off between prediction accuracy and inference cost of the DNN, running 27,000 trials on the HPC of the RWTH Aachen University.

Overall, this chapter demonstrates that deep learning can provide accurate and adaptive plant models for novel combustion systems and highlights the role of automated methods in streamlining model development for embedded control applications.

While the presented modeling approach successfully enables accurate prediction of the complex engine dynamics, several limitations remain. Model performance, particularly for p_{mi} and dp_{max}, could likely be further improved with more extensive and better-distributed training data. Nevertheless, considering the rapid model development, the results underline the potential of the proposed approach and highlight the inherent complexity of the underlying dynamics. Moreover, the pseudo-random input strategy was feasible here due to the robustness of the diesel engine. For more sensitive systems, such as HCCI combustion, such excitation may pose risks [140]. Lastly, although compact by deep learning standards, the resulting networks are still considerably larger than classical models, complicating integration into resource-constrained or safety-critical applications. In addition, AutoML proved effective in identifying lean and accurate networks, yet it risks disconnecting the model from physical intuition and expert validation – making interpretability and system-level analysis more difficult.

In the implemented real-time setup, the manually designed DNN model was used on the test-bench rather than the AutoML-selected architecture. This decision was driven by existing software integrations and the non-sequential development of the test-bench environment, where replacing the model would have required substantial re-integration effort. Building on this foundation, the next chapter integrates the DNN into an MPC framework and evaluates the resulting controller on the H2DF engine test-bench.

5 Deep Nonlinear Model Predictive Control

This chapter presents a nonlinear model predictive control (MPC) framework using a deep neural network (DNN) as model tailored for hydrogen diesel dual-fuel (H2DF) operation. It addresses the two key challenges identified previously: achieving real-time capability and ensuring safe control of the highly nonlinear combustion dynamics.

A brief literature review on MPC for internal combustion engine (ICEs) control follows, emphasizing current strategies and their limitations. This is complemented by an overview of the chapter's structure and its contribution.

This chapter is partially based on [32], [33], [35], [38], [47].

5.1 Introduction to the Chapter

Early applications of MPC to experimentally validated ICE control focused on leveraging reduced-order physical models. Bengtsson et al. [141] (2006) demonstrated real-time MPC on a six-cylinder homogeneous charge compression ignition (HCCI) engine, while Di Cairano et al. [142] (2012) implemented MPC for idle speed regulation in spark ignition (SI) engines, achieving notable gains in drivability and fuel efficiency. Iwadare and Ueno [143] (2009) applied MPC to diesel engine air-path control, enabling fast, independent actuation of multiple variables. Widd et al. [144] (2014) further showcased the use of explicit MPC for managing exhaust recompression in HCCI combustion.

The first efforts to integrate machine learning (ML) into MPC frameworks emerged with the aim of reducing modeling effort and computational complexity. Wang et al. [145] (2006) introduced an artificial neural network (ANN)-based MPC for air-fuel ratio control in SI engines, accelerating online optimization using a reduced Hessian approach. Janakiraman et al. [146] and Janakiraman et al. [147] (2015, 2016) used extreme learning machines to model HCCI combustion dynamics, enabling accurate control. Irdmousa et al. [148] (2019) implemented a data-driven linear-parameter-varying-MPC for reactivity controlled compression ignition engines, focusing on combustion phasing. Norouzi et al. [32] (2022) embedded DNNs into a nonlinear MPC framework for diesel engines using `acados`, showing real-time capability and notable NO_x reduction. Gordon et al. [33] (2024) extended this architecture to HCCI engines, demonstrating stable combustion phasing and accurate load tracking, despite the process's strong cycle-to-cycle (C2C) dependencies. In both studies, the DNNs served as the predictive model within the optimal control problem (OCP).

In summary, MPC for ICE control has progressed from conventional models to advanced data-driven strategies that enable sufficient performance across complex combustion regimes. However, several challenges remain, including the trade-off between model depth and real-time feasi-

bility, the high demand for high-quality training data, and the extension to broader application domains.

This chapter addresses parts of this gap by presenting recent work from the author and the research teams at RWTH and the UofA, where LSTM-based architectures were employed to model HCCI and diesel engines [32], [33], [35]. When integrated into MPC, these models demonstrated real-time capability using the `acados` framework.

While building on the general modeling and control concepts established in these earlier works, the approaches presented in this chapter and thesis represent an independent line of investigation. The focus shifts toward increasing modeling capacity and robustness through a revised architecture and controller design. Structural complexity and computational load are addressed by replacing the long short-term memory (LSTM) with the more efficient gated recurrent unit (GRU) architecture and by increasing the depth and width of the DNN, as motivated in Chapter 4. Numerical robustness is significantly enhanced through a redesigned cost function, which incorporates slack variables and control-rate constraints to handle infeasibilities more effectively. Furthermore, multiple variants of the model and cost function demonstrate the flexibility and effectiveness of the proposed formulation. In addition, this work presents the first empirical robustness evaluation performed within the research groups.

The development of the DNN-based MPC in this work follows a structured, two-step approach, as illustrated in Fig. 5.1. The process builds upon the validated DNN-based engine model, as developed in previous chapter 4. First, a baseline control strategy is defined and evaluated to serve as a reference for subsequent developments. In the next step, an offline version of the MPC is developed and tested in an on-target model-in-the-loop (MiL) simulation. This includes defining the objective function and physical constraints, implementing the DNN model, and integrating the OCP using the open-source framework `acados`. The goal of this step is to tune the cost function and validate the controller performance under nominal conditions before deployment. In the second step, the controller is transitioned to a real-time environment: the H2DF engine using an embedded platform integrated into the test bench setup. Performance is evaluated under real-world conditions, and multiple controller variants, spanning both the DNN architecture and cost function design, are explored to identify configurations best suited for specific use cases. Finally, robustness investigations are conducted to assess sensitivity to noise and perturbations. This enables rapid development of a safe, high-performance prototype controller.

The chapter concludes with a summary and a final discussion that situates the presented MPC methodology within the wider landscape of the thesis.

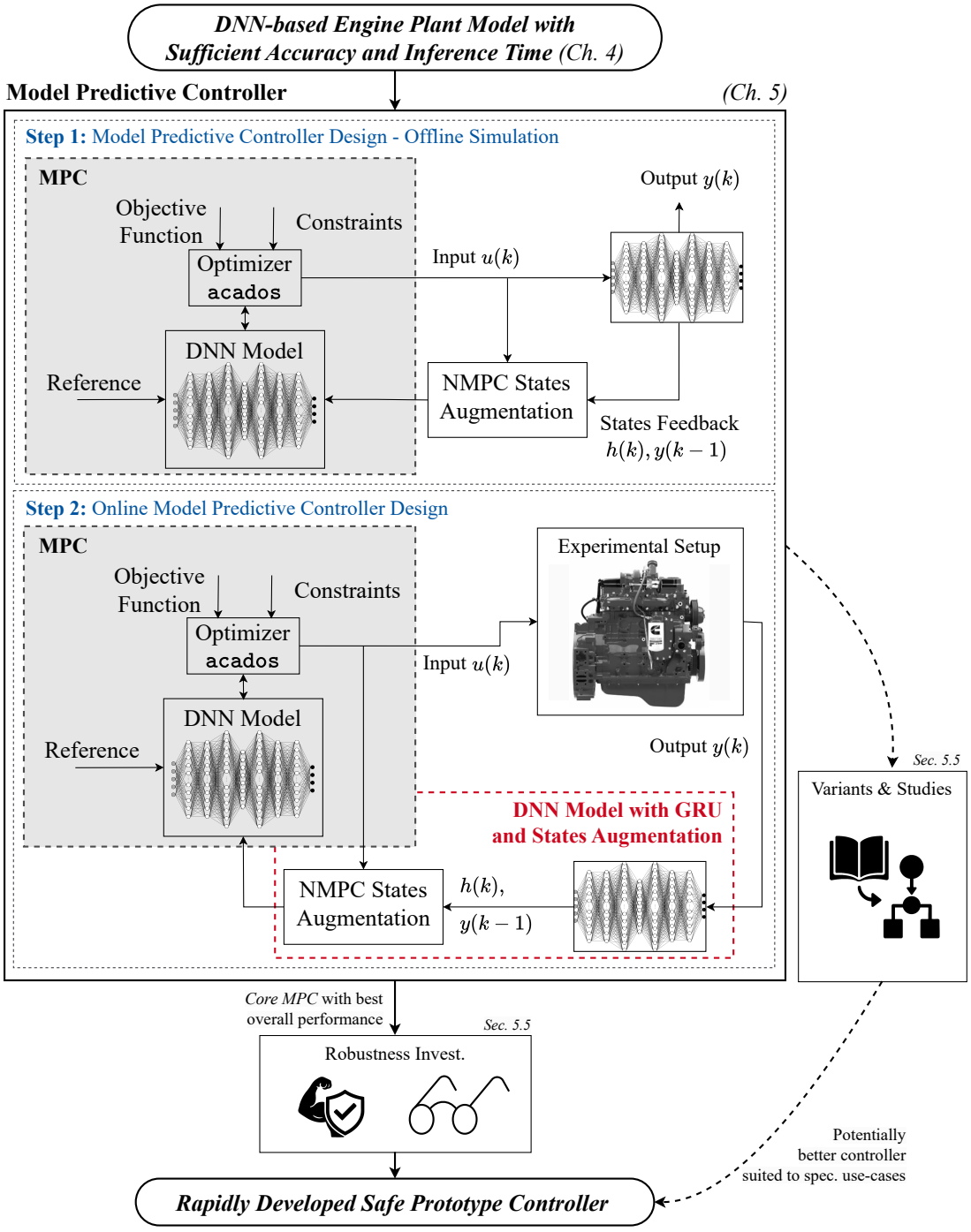


Figure 5.1: Development process of online MPC for H2DF.

5.2 Control Objective & Baseline Strategy

Following Chapter 3, the controls $u(k)$ for the MPC are defined as:

$$u(k) = \left[t_{\text{Main}}(k), t_{\text{P2M}}(k), \alpha_{\text{Main}}(k), t_{\text{H2}}(k)\right]^T \in \mathbb{R}^4. \tag{5.1}$$

With these controls, the MPC is able to control both the hydrogen and diesel fuel injections as explained in Section 3.3.

There are four engine output parameters $y(k)$ of interest used to capture the performance of the engine including: p_{mi} which is representative for the torque output, dp_{max} which is an indicator of combustion noise and engine durability, c_{NOx} and particulate matter c_{PM} cylinder-out emissions (see Eq. 5.2). All parameters are measured for cylinder 1.

$$y(k) = \left[p_{\text{mi}}(k), c_{\text{NOx}}(k), c_{\text{PM}}(k), dp_{\text{max}}(k)\right]^T \in \mathbb{R}^4. \tag{5.2}$$

The engine load p_{mi} is calculated in real-time on an field programmable gate array (FPGA) board over one full combustion cycle on the measured in-cylinder pressure trace, while dp_{max} is the filtered maximum value of the latter per crank angle degree (CAD). See Section 2.1 and Section 3.1 for more information on these variables.

Following this, the proposed main objective of the MPC is to track the reference engine output $p_{\text{mi,ref}}$ in terms of p_{mi}, while simultaneously minimizing engine-out c_{NOx} and c_{PM} emissions and limiting engine wear as indicated by dp_{max}.

The reference $p_{\text{mi,ref}}$ is given by a predefined trajectory of 4900 engine cycles, which runs for approximately 390 s (1500 min^{-1}). This trajectory includes rapid load steps at different levels, as well as short and long ramps, to test the controller's tracking performance. It also challenges the controller when handling elevated cylinder temperatures during long high-load ramps, which can lead to increased emissions, increased dp_{max}, and eventually knocking. The trajectory spans from a p_{mi} of 3 bar to 8 bar, corresponding to roughly 75% of the maximum p_{mi} identified for this engine under steady-state H2DF operation [113].

Fig. 5.2 shows the load reference trajectory tracking by the standard *Cummins* diesel-only baseline controller mappings, while Table 5.1 shows the performance metrics of the run. As performance metrics the mean and maximum values of all measured outputs defined in Eq. 5.2 are chosen, but p_{mi}. Here, due to the reference tracking control task of the controller, error metrics are selected. More details of the metrics can be found in Section 2.3. The NRMSE is normalized using the full range of the $p_{\text{mi,ref}}$ load reference trajectory (5 bar).

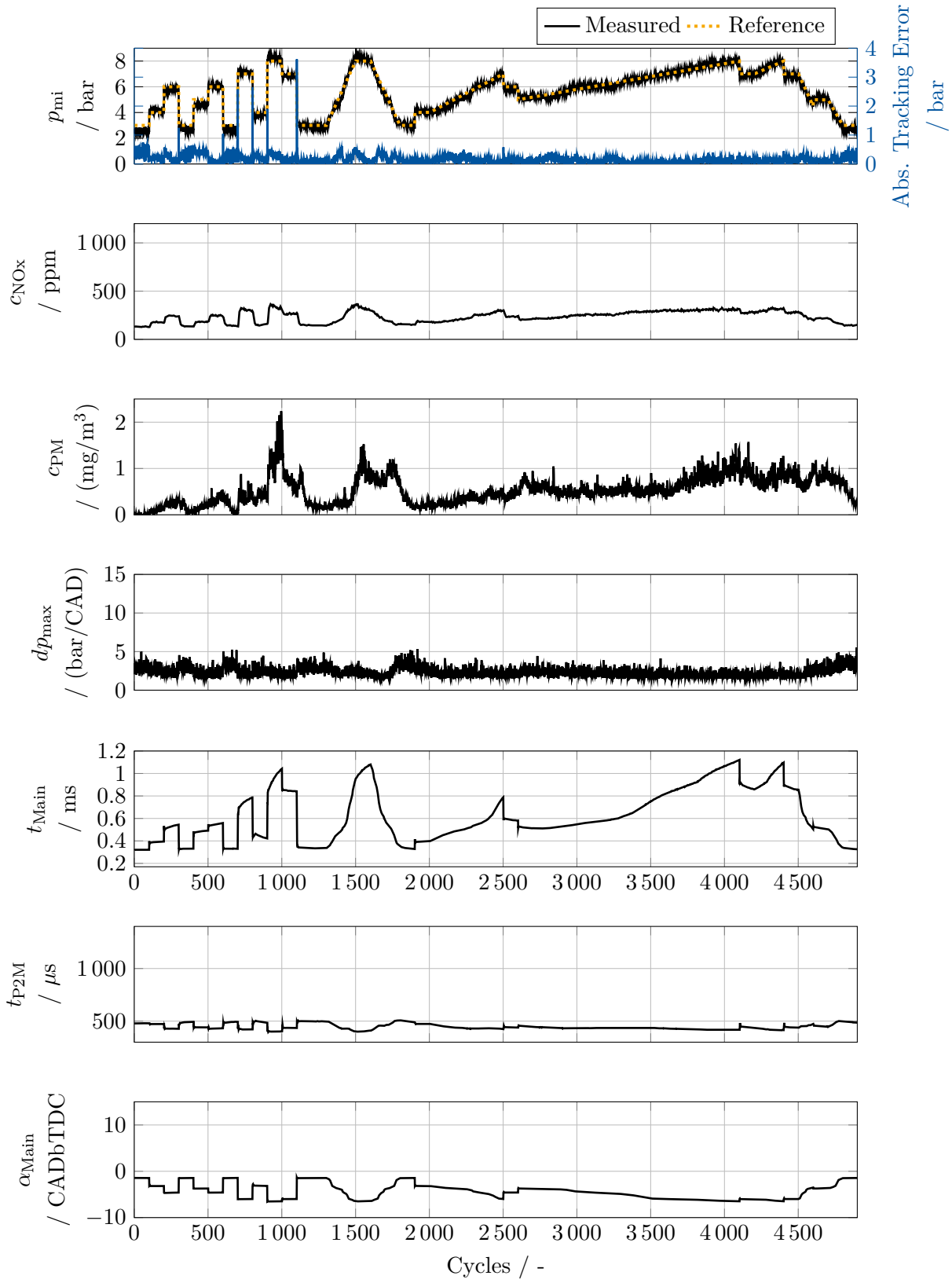


Figure 5.2: Diesel standard controller experimental results for tracking the standard load reference trajectory of 4900 engine cycles at 1500 min^{-1}.

Table 5.1: Diesel standard controller experimental results performance metrics for tracking the standard load reference trajectory of 4900 engine cycles at $1500\,\mathrm{min}^{-1}$.

Metric	**Result**
p_{mi} tracking	
MAE	0.162 bar
RMSE	0.222 bar
NRMSE	4.44%
c_{NOx}	
max	364 ppm
mean	235 ppm

Metric	**Result**
c_{PM}	
max	$2.24\,\mathrm{mg/m^3}$
mean	$0.54\,\mathrm{mg/m^3}$
dp_{max}	
max	5.53 bar/CAD
mean	2.34 bar/CAD

The data shows decent tracking performance on the provided trajectory with an NRMSE of 4.44%. The maximum cylinder pressure rise rate dp_{max} stays relatively low, as do NO_x emissions c_{NOx}, with mean values of 2.34 bar/CAD and 235 ppm, respectively. In contrast, the diesel baseline controller run exhibits relatively high soot emissions c_{PM} of $0.54\,\mathrm{mg/m^3}$ in average, which is expected for pure diesel operation.

In the following section, the development of the MPC including the underlying OCP and model formulation is explained in detail.

5.3 Nonlinear Model Predictive Control Setup

To integrate the trained DNN into MPC, the generic state-space model in Eq. 2.4a is refined according to the actual DNN's equations using forward propagation. Fig. 5.3 depicts the full computational graph of the DNN.

Following this, the forward propagation functions for the DNN can be defined: a GRU-layer (see Eq. 2.8) is enclosed by multiple fully connected (FC) layers using the ReLU activation function on each side (see Eq. 2.6):

$$z_{\mathrm{FC1}}(k) = \mathrm{ReLU}\left(w_{\mathrm{FC1}}^{T} u(k) + b_{\mathrm{FC1}}\right) \tag{5.3a}$$

$$z_{\mathrm{FC2}}(k) = \mathrm{ReLU}\left(w_{\mathrm{FC2}}^{T} z_{\mathrm{FC1}}(k) + b_{\mathrm{FC2}}\right) \tag{5.3b}$$

$$z_{\mathrm{FC3}}(k) = \mathrm{ReLU}\left(w_{\mathrm{FC3}}^{T} z_{\mathrm{FC2}}(k) + b_{\mathrm{FC3}}\right), \tag{5.3c}$$

$$z(k) = \sigma\left(w_{u,z}^{\top} z_{\mathrm{FC3}}(k) + w_{h,z}^{\top} h(k-1) + b_z\right) \tag{5.4a}$$

$$r(k) = \sigma\left(w_{u,r}^{\top} z_{\mathrm{FC3}}(k) + w_{h,r}^{\top} h(k-1) + b_r\right) \tag{5.4b}$$

$$g(k) = \tanh\left(w_{u,g}^{\top} z_{\mathrm{FC3}}(k) + w_{h,g}^{\top} (r(k) \otimes h(k-1)) + b_g\right) \tag{5.4c}$$

$$h(k) = (1 - z(k)) \otimes h(k-1) + z(k) \otimes g(k), \tag{5.4d}$$

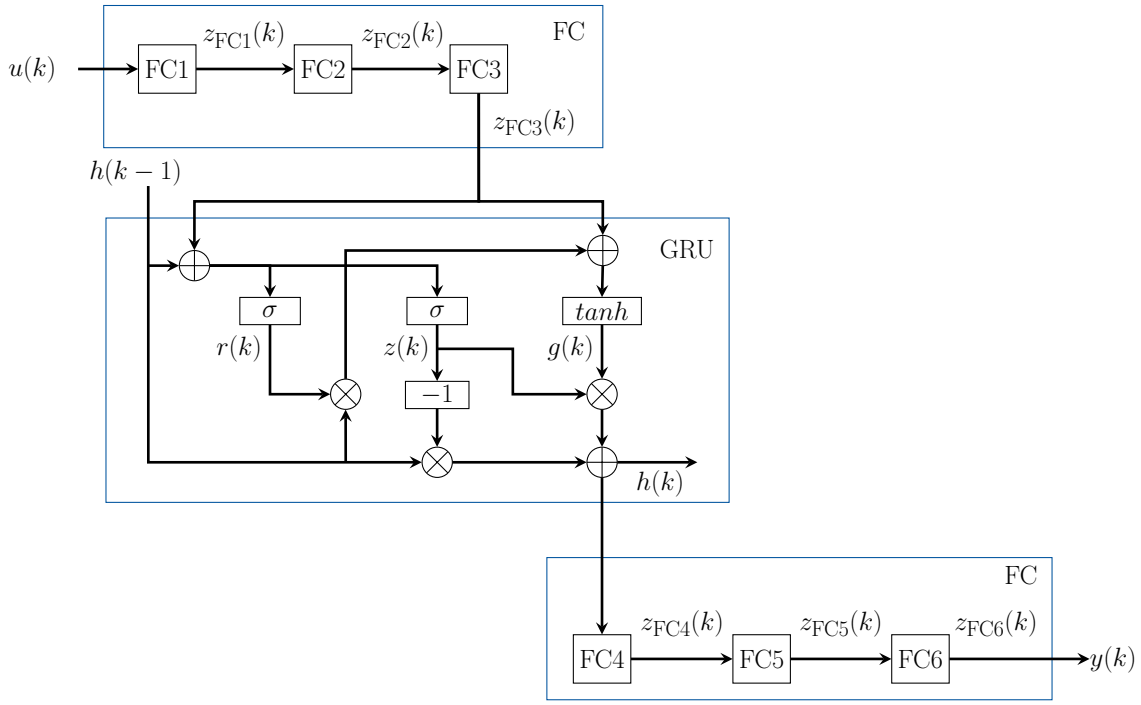


Figure 5.3: Computational graph of the DNN based engine model, using GRU and FC layer.

$$z_{\text{FC4}}(k) = \text{ReLU}\left(w_{\text{FC4}}^{\top} h(k) + b_{\text{FC4}}\right) \tag{5.5a}$$

$$z_{\text{FC5}}(k) = \text{ReLU}\left(w_{\text{FC5}}^{\top} z_{FC4}(k) + b_{\text{FC5}}\right) \tag{5.5b}$$

$$y(k) = w_{\text{FC6}}^{\top} z_{\text{FC5}}(k) + b_{\text{FC6}}, \tag{5.5c}$$

where $w_{\text{FC},i}$ and $b_{\text{FC},i}$ are the weights and biases of the FC layer with $i \in \{1,2,3,4,5,6\}$ and where $w_{u,(z,r,g)}$ are the weight matrices of the input vector $u(k)$ and $w_{h,(z,r,g)}$ are the weight matrices to the previous short-term states $h(k-1)$ (see Section 2.3 for more details). The activation functions used in Eq. (5.3, 5.4, 5.5) are given as:

$$\tanh(z) = \frac{e^{2z}-1}{e^{2z}+1} \tag{5.6a}$$

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{5.6b}$$

$$\text{ReLU} = \begin{cases} 0 & \text{if } z \leq 0 \\ z & \text{if } z > 0 \end{cases}. \tag{5.6c}$$

Summarizing this model per layer-bundle (FC-in, GRU, FC-out) leads to:

$$z_{\mathrm{FC3}}(k) = f_{\mathrm{FC,in}}\left(u(k)\right), \tag{5.7a}$$
$$h(k) = f_{\mathrm{GRU}}\left(h(k-1), z_{\mathrm{FC3}}(k)\right), \tag{5.7b}$$
$$y(k) = f_{\mathrm{FC,out}}\left(h(k)\right), \tag{5.7c}$$

where $f_{\mathrm{FC,in}}$, f_{GRU} and $f_{\mathrm{FC,out}}$ integrate and summarize the DNN equations of each of those layers from Eq. 5.3, Eq. 5.4, Eq. 5.5, respectively.

Further simplifying Eq. 5.7 by eliminating intermediate value $z_{\mathrm{FC3}}(k)$ and defining the model state vector $x(k)$ as the GRU's hidden states $h(k-1)$ of the previous timestep results in:

$$x(k+1) = f_{\mathrm{FC,in:GRU}}\left(x(k), u(k)\right), \tag{5.8a}$$
$$y(k) = f_{\mathrm{FC,out}}\left(x(k+1)\right), \tag{5.8b}$$

with $f_{\mathrm{FC,in:GRU}}$ combining the input FC layers and the GRU layer and $f_{\mathrm{FC,out}}$ representing the FC output layers.

To use Eq. 5.8 in a standard nonlinear state-space formulation (see Eq. 2.4a, where the output $y(k)$ depends only on the current state $x(k)$ and the current input $u(k)$), it is necessary to adjust the model's timestep definition. In particular, the definition of the engine cycle is selected so that each output explicitly captures the effect of the control input on the subsequent cycle (see Fig. 3.5). Finally, the nonlinear state-space representation suitable for MPC framework integration is defined as:

$$x(k+1) = f_{\mathrm{FC,in:GRU}}\left(x(k), u(k)\right), \tag{5.9a}$$
$$= f\left(x(k), u(k)\right), \tag{5.9b}$$
$$y(k) = f_{\mathrm{FC,out}}\left(f\left(x(k), u(k)\right)\right), \tag{5.9c}$$
$$= h\left(x(k), u(k)\right). \tag{5.9d}$$

Here, the model states $x(k)$, the model inputs $u(k)$ and the model outputs $y(k)$ are subsequently defined by the DNN input and output features definitions in Eq. 5.1 and Eq. 5.2, respectively:

$$x(k) = h(k-1) \in \mathbb{R}^8, \tag{5.10a}$$
$$u(k) = \left[t_{\mathrm{Main}}(k), t_{\mathrm{P2M}}(k), \alpha_{\mathrm{Main}}(k), t_{\mathrm{H2}}(k), p_{\mathrm{mi}}(k-1)\right]^{\mathsf{T}} \in \mathbb{R}^5, \tag{5.10b}$$
$$y(k) = \left[p_{\mathrm{mi}}(k), c_{\mathrm{NOx}}(k), c_{\mathrm{PM}}(k), dp_{\max}(k)\right]^{\mathsf{T}} \in \mathbb{R}^4. \tag{5.10c}$$

To improve control stability and reduce oscillations in the outputs, the gradients of the controls $\Delta u(k) = u(k) - u(k-1)$ are introduced as additional inputs [149]. This modification enables

the penalization of both absolute inputs and their rate of change within the cost function. Additionally, using the weights and constraints, the controller's aggressiveness can be adjusted in application.

Following this, using Eq. 5.9, the MPC state vector $\tilde{x}(k)$ and MPC output vector $\tilde{y}(k)$ are thus defined as:

$$\underbrace{\begin{bmatrix} x(k+1) \\ u(k) \end{bmatrix}}_{\tilde{x}(k+1)} = \underbrace{\begin{bmatrix} f\big(x(k),\, u(k-1) + \Delta u(k)\big) \\ u(k-1) + \Delta u(k) \end{bmatrix}}_{\tilde{f}\left(\tilde{x}(k),\, \Delta u(k)\right)}, \tag{5.11a}$$

$$\underbrace{\begin{bmatrix} y(k) \\ u(k-1) \end{bmatrix}}_{\tilde{y}(k)} = \underbrace{\begin{bmatrix} g(x(k)) \\ u(k-1) \end{bmatrix}}_{\tilde{g}} (\tilde{x}(k)), \tag{5.11b}$$

where $\tilde{f}$ and $\tilde{g}$ are the MPC's augmented nonlinear state transition function and output mapping function, respectively.

Subsequently, the physical controls are augmented into the MPC's state vector $\tilde{x}(k)$ and the gradients of those are defined as the MPC's manipulated variables vector $\tilde{u}(k)$:

$$\tilde{x}(k) = \begin{bmatrix} y_{\mathrm{p,mi}}(k), y_{\mathrm{c,NOx}}(k), y_{\mathrm{c,PM}}(k), y_{\mathrm{dp,Max}}(k), h(k-1), \\ u_{\mathrm{t,Main}}(k), u_{\mathrm{t,P2M}}(k), u_{\alpha,\mathrm{Main}}(k), u_{\mathrm{t,H2}}(k) \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{16}, \tag{5.12a}$$

$$\tilde{u}(k) = \Big[\Delta u_{\mathrm{t,Main}}(k), \Delta u_{\mathrm{t,P2M}}(k), \Delta u_{\alpha,\mathrm{Main}}(k), \Delta u_{\mathrm{t,H2}}(k)\Big]^{\mathsf{T}} \in \mathbb{R}^{4}. \tag{5.12b}$$

Thus, the discrete OCP-structured nonlinear program (NLP) is defined as follows (see Eq 2.5) for each engine cycle k, where i denotes the stage index within the prediction horizon:

$$\begin{aligned}
&\underset{\substack{\tilde{u}_0,\ldots,\tilde{u}_{N-1} \\ \tilde{x}_0,\ldots,\tilde{x}_N \\ \tilde{y}_0,\ldots,\tilde{y}_N \\ s_0,\ldots,s_N}}{\text{minimize}} \quad J \\
&\text{subject to} \qquad\qquad \tilde{x}_0 = \tilde{x}(k), \\
&\qquad\qquad\qquad\qquad \tilde{x}_{i+1} = \tilde{f}(\tilde{x}_i, \tilde{u}_i) \quad i = 0,\ldots,N-1, \\
&\qquad\qquad\qquad\qquad \tilde{y}_i = \tilde{g}(\tilde{x}_i) \quad i = 0,\ldots,N, \\
&\qquad\qquad\qquad\qquad \tilde{u}_{\min} \le F_{\tilde{u}}\, \tilde{u}_i \quad i = 0,\ldots,N-1, \\
&\qquad\qquad\qquad\qquad F_{\tilde{u}}\, \tilde{u}_i \le \tilde{u}_{\max} \quad i = 0,\ldots,N-1, \\
&\qquad\qquad F_{\tilde{x}}\, \tilde{x}_i + F_{s,\tilde{x}}\, s_{\min,\tilde{x},i} \ge \tilde{x}_{\min} \quad i = 0,\ldots,N, \\
&\qquad\qquad F_{\tilde{x}}\, \tilde{x}_i - F_{s,\tilde{x}}\, s_{\max,\tilde{x},i} \le \tilde{x}_{\max} \quad i = 0,\ldots,N, \\
&\qquad\qquad\qquad\qquad s_i \ge 0 \quad i = 0,\ldots,N,
\end{aligned} \tag{5.13}$$

where $s_i = (s_{\min,\tilde{x},i},\, s_{\max,\tilde{x},i}) \in \mathbb{R}^5$ are the non-negative slack variables of stage i and J is the finite-horizon cost defined in Eq. 5.14. The prediction horizon length N defines how many engine cycles the MPC predicts into the future. Soft constraints using slack variables s_i are introduced in Eq. 2.5 to enhance the MPC formulation. $F_{\tilde{u}}$, $F_{\tilde{x}}$ and $F_{s,\tilde{x}}$ in Eq. 5.13 are diagonal matrices which map the bounded outputs and inputs. They are incorporated into the constraints as additional non-negative variables that allow controlled constraint violations, which are penalized in the cost function. This mechanism ensures that a feasible solution can still be found even in situations where hard constraints might otherwise make the problem infeasible. Incorporating soft constraints in this way improves the controller's ability to handle tight or conflicting constraints, contributing to overall robustness in real-world applications [150]. For brevity, the detailed lagrange and mayer forms of the cost function (as referenced in Eq. 2.5) are omitted.

Following the control objective defined in Section 5.2, the finite-horizon cost J in Eq. 5.13 is defined in Eq. 5.14 using a linear least squares formulation:

$$
\begin{aligned}
J = \sum_{i=0}^{N} \Big[& \underbrace{\|y_{\mathrm{p,mi},i} - p_{\mathrm{mi,ref},i}\|_q^2}_{\text{Torque output tracking}} \\
& + \underbrace{\|y_{\mathrm{c,NOx},i}\|_q^2 + \|y_{\mathrm{c,PM},i}\|_q^2}_{\text{Emissions minimization}} \\
& + \underbrace{\|y_{\mathrm{dp,Max},i}\|_q^2}_{\text{Engine wear and noise reduction}} \\
& + \underbrace{\|u_{\mathrm{t,Main},i}\|_q^2 + \|u_{\mathrm{t,H2},i}\|_q^2}_{\text{Fuel consumption minimization and } \mathrm{H_2} \text{ share}} \\
& + \underbrace{\|s_i\|_q^1}_{\text{Constraint violation}} \Big] \\
& + \sum_{i=0}^{N-1} \underbrace{\|\Delta u_i\|_r^2}_{\text{Oscillation reduction}} ,
\end{aligned}
\tag{5.14}
$$

Q and R are positive semi-definite sparse weighting matrices, defined as:

$$Q = \mathrm{diag}(q_{\mathrm{p,mi}},\, q_{\mathrm{c,NOx}},\, q_{\mathrm{c,PM}},\, q_{\mathrm{dp,max}},\, 0, \ldots, 0,\, q_{\mathrm{t,Main}},\, q_{\mathrm{t,P2M}},\, q_{\alpha,\mathrm{Main}},\, q_{\mathrm{t,H2}}), \tag{5.15a}$$

$$R = \mathrm{diag}(r_{\Delta\mathrm{t,Main}},\, r_{\Delta\mathrm{t,P2M}},\, r_{\Delta\alpha,\mathrm{Main}},\, r_{\Delta\mathrm{t,H2}}). \tag{5.15b}$$

The matrices map the diagonal scalars q_i and r_i in Eq. 5.14 to the MPC's state and control vectors (see Eq. 5.12) in the ℓ_2-norm formulation. Adjusting these scalars tailors the cost to specific controller settings. Note that the weights on all hidden states $h(k-1)$ of the DNN are set to zero, so the DNN itself is not penalized.

The cost function in Eq. 5.14 is normalized identical to the DNN, utilizing the range of the variables from the training dataset.

Using $F_{\tilde{u}}$, $F_{\tilde{x}}$ and $F_{s,\tilde{x}}$, the control outputs are limited to match the hardware used while constraints imposed on the outputs are used to guarantee safe engine operation. Additionally, the rate of change Δ of the control outputs is constrained to allow for specific controller aggressiveness settings. All these constraints for the *Core MPC* are denoted in Table 5.2.

In the following section, the validation of the developed MPC in a MiL simulation is explained briefly.

Table 5.2: MPC constraint values (physical). Soft boundaries are marked with $*$.

Lower bound	**Variable**	**Upper bound**
0 bar$*$	$y_{\mathrm{p,mi}}$	9 bar$*$
0 ppm$*$	$y_{\mathrm{c,NOx}}$	1200 ppm$*$
0 mg/m^{3*}	$y_{\mathrm{c,PM}}$	1.5 mg/m^{3*}
0 bar/CAD$*$	$y_{\mathrm{dp,Max}}$	15 bar/CAD$*$
0.17 ms	$u_{\mathrm{t,Main}}$	0.50 ms
430 µs	$u_{\mathrm{t,P2M}}$	1400 µs
−10 CADbTDC	$u_{\alpha,\mathrm{Main}}$	10 CADbTDC
1.5 ms	$u_{\mathrm{t,H2}}$	4.0 ms
−0.005 ms/cycle	$\Delta u_{\mathrm{t,Main}}$	0.005 ms/cycle
−20 µs/cycle	$\Delta u_{\mathrm{t,P2M}}$	20 µs/cycle
−1.5 CAD/cycle	$\Delta u_{\alpha,\mathrm{Main}}$	1.5 CAD/cycle
−0.24 ms/cycle	$\Delta u_{\mathrm{t,H2}}$	0.24 ms/cycle

5.4 On-Target MiL Simulation and Embedded Deployment Feasibility

As a feasibility study and to identify potential formulation errors, the MPC is integrated and tested within a MiL simulation. The reference engine load profile $p_{\mathrm{mi,ref}}$ is provided to the controller as a target trajectory for p_{mi} (see Section 5.2 and Fig. 5.2). The tracking performance with respect to this reference is evaluated, and key controller characteristics such as responsiveness and aggressiveness are qualitatively analyzed. The MiL environment allows for early-stage validation of the control formulation and fine-tuning the cost function.

The simulation is set up in a `MATLAB/Simulink` environment which enables direct transition towards the actual rapid control prototyping (RCP) systems or embedded systems at the test bench. Fig. 5.4 shows a schematic of the structure of the MiL simulation.

The plant model used in the simulation employs the same DNN-based dynamics as those integrated into the MPC, as defined in Eq. 5.8. This leads to near-perfect control performance under ideal conditions. Fig. 5.4 further illustrates the simulation architecture, highlighting the state

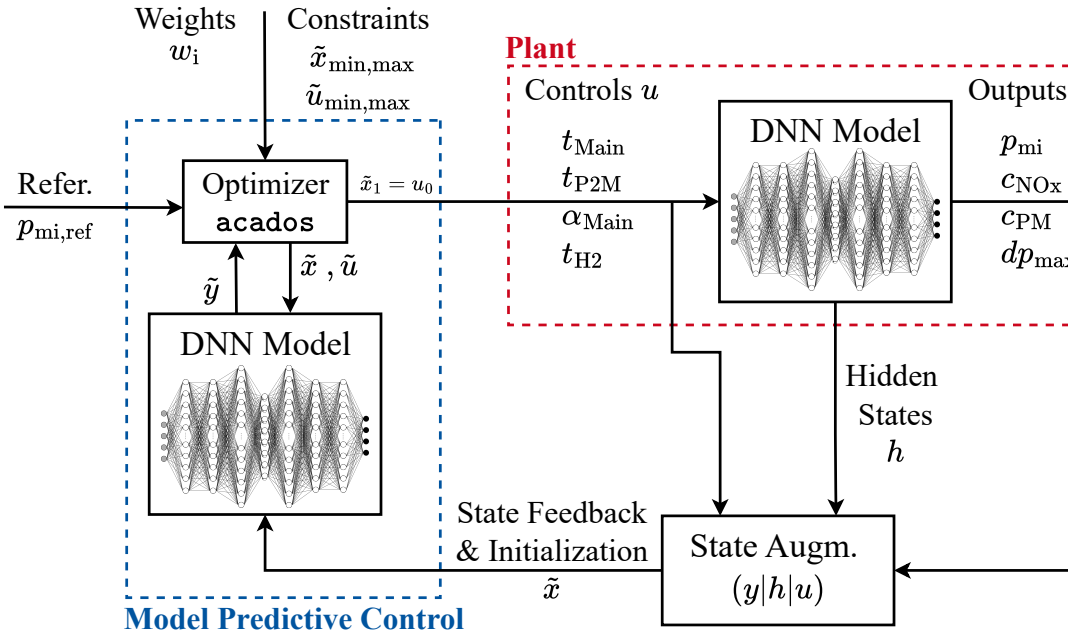


Figure 5.4: Block diagram of MiL simulation for early MPC validation.

augmentation $\tilde{x}(0) = (y \mid h \mid u)$, which initializes the full state vector of the MPC, including the GRU's hidden states, as described in Eq. 5.12.

As mentioned previously (see Section 3.4), with the engine operating at a constant speed of $n_{Eng} = 1500\,\text{min}^{-1}$, one complete engine cycle takes 80 ms, with only $t_{comp,max} = 18\,\text{ms}$ available for the MPC calculations before control actions must be defined. To meet these tight real-time requirements, the computationally efficient open-source package `acados` is used [30].

In the MiL simulation `acados` outperforms both `MATLAB`'s `MPC toolbox` [151] using `fmincon` as well as the `FORCES PRO` [152], [153] backends [32]. This difference in solver performance is attributed to the higher dimension of the state than the control input vector in addition to the short prediction horizon of three cycles needed to model the diesel ICE dynamics. This allows the utilized interior point-based quadratic programming (QP) solver `HPIPM` [154] to take full advantage of the condensation benefits. [32], [149], [155]

As no discretization is required for the model used, the plant model can be directly implemented using the discrete dynamics interface of `acados`. The Gauss-Newton approximation is used for the computation of the Hessian in the underlying sequential quadratic programming (SQP) algorithm. The OCP in Eq. 5.13 leads to a band diagonal structure in the matrices of the QPs, which are solved using `HPIPM` [154].

Initial MiL simulations on desktop-grade hardware revealed considerable computational demands due to the iterative nature of the MPC and the incorporated complex RNN model. It became evident that the utilized RCP system *dSPACE MicroAutoBox* (MABX) would not

be able to meet these performance requirements. To further address this, the MPC computations are offloaded to a *Raspberry Pi 400* (RPi) [156] equipped with an *ARM Cortex-A72* processor [157], overclocked to 2.2 GHz.

The MiL simulation was then embedded and executed directly on this target hardware. This enables a realistic assessment of the controller's computational feasibility and allows for tuning key parameters affecting computational load, specifically the number of SQP iterations and the prediction horizon length. These on-target MiL simulations showed that three SQP iterations were sufficient to achieve good tracking performance. Additionally, since the C2C dependency of a comparable combustion process (e.g. HCCI) typically spans approximately two engine cycles, a prediction horizon of three cycles was deemed adequate [158]. With this, a real-time capability was achieved with maximum computation times using the RPi of under 7 ms. The detailed MiL simulation results are omitted here, as they confirmed expectations: near-perfect reference tracking was achieved under these conditions.

This approach significantly reduces the required test bench time and engine runtime to develop the engine controller, thereby lowering fuel consumption and carbon dioxide emissions.

In the following section, the experimental validation of the developed H2DF MPC on the engine test bench is explained.

5.5 Experimental Results on the Test Bench

As outlined in the previous section, this work offloads the MPC calculations to the RPi when deployed on the test bench. The RPi communicates with the main engine controller running on the MABX over `user datagram protocol` (UDP) with *Ethernet* as physical layer as illustrated in Fig. 5.5.

The MPC computation on the RPi operates independently from the MABX, with the exception of receiving the current measured states and reference value. Both the optimizer and the DNN model are executed on the RPi, the resulting actuation values for the next cycle are transmitted to the MABX. This outsourcing enables the MPC to be set to more NLP and SQP iterations and deeper DNN model compared to a previous LSTM-based MPC integration on the MABX [35], [159]. An additional benefit of this approach is the ability to change the MPC or its DNN model independently from the engine control software on the MABX, which allows for rapid control development [33]. Finally, this modular controller design allows for the MPC to be executed on any external processor that can interface with the MABX over UDP.

The MPC, implemented with constraints from Table 5.2, is tuned for different configurations. The following subsections introduce these variants and present their performance results. A robustness investigation concludes the analysis.

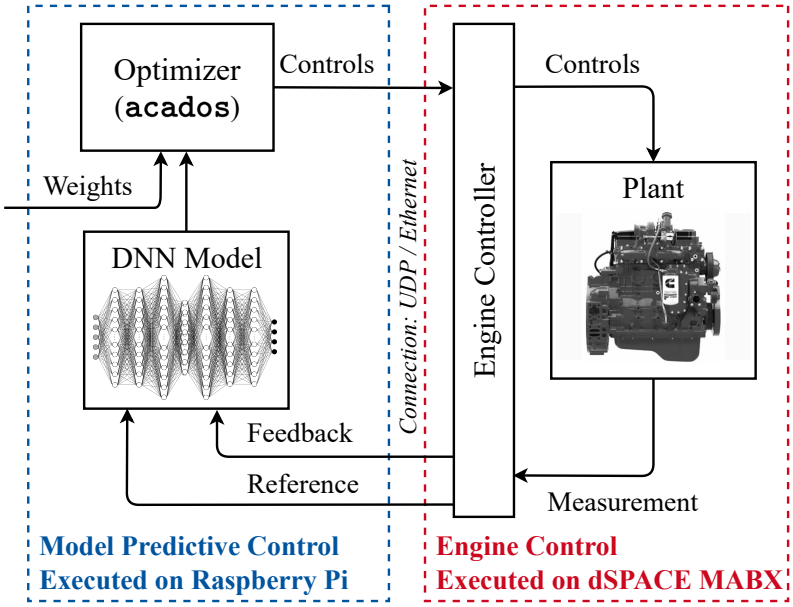


Figure 5.5: Block diagram of MPC and engine test bench communication setup.

5.5.1 Core MPC Variant

This subsection presents the experimental results of the *Core MPC* configuration implemented on the engine test bench. Selected based on extensive MiL simulations and few online tests, the *Core MPC* demonstrated robust overall performance in terms of reference tracking and constraint handling, while maintaining a balanced control aggressiveness. Its cost function was tuned to avoid excessive sensitivity to any specific term. The *Core MPC* configuration forms the basis for all alternative MPC variants explored in this work and is used as the main point of reference for the comparisons in the following subsections.

The experimental performance of the *Core MPC* over the reference load trajectory of 4900 cycles at a constant engine speed of $1500\,\mathrm{min}^{-1}$ is illustrated in Fig. 5.6.

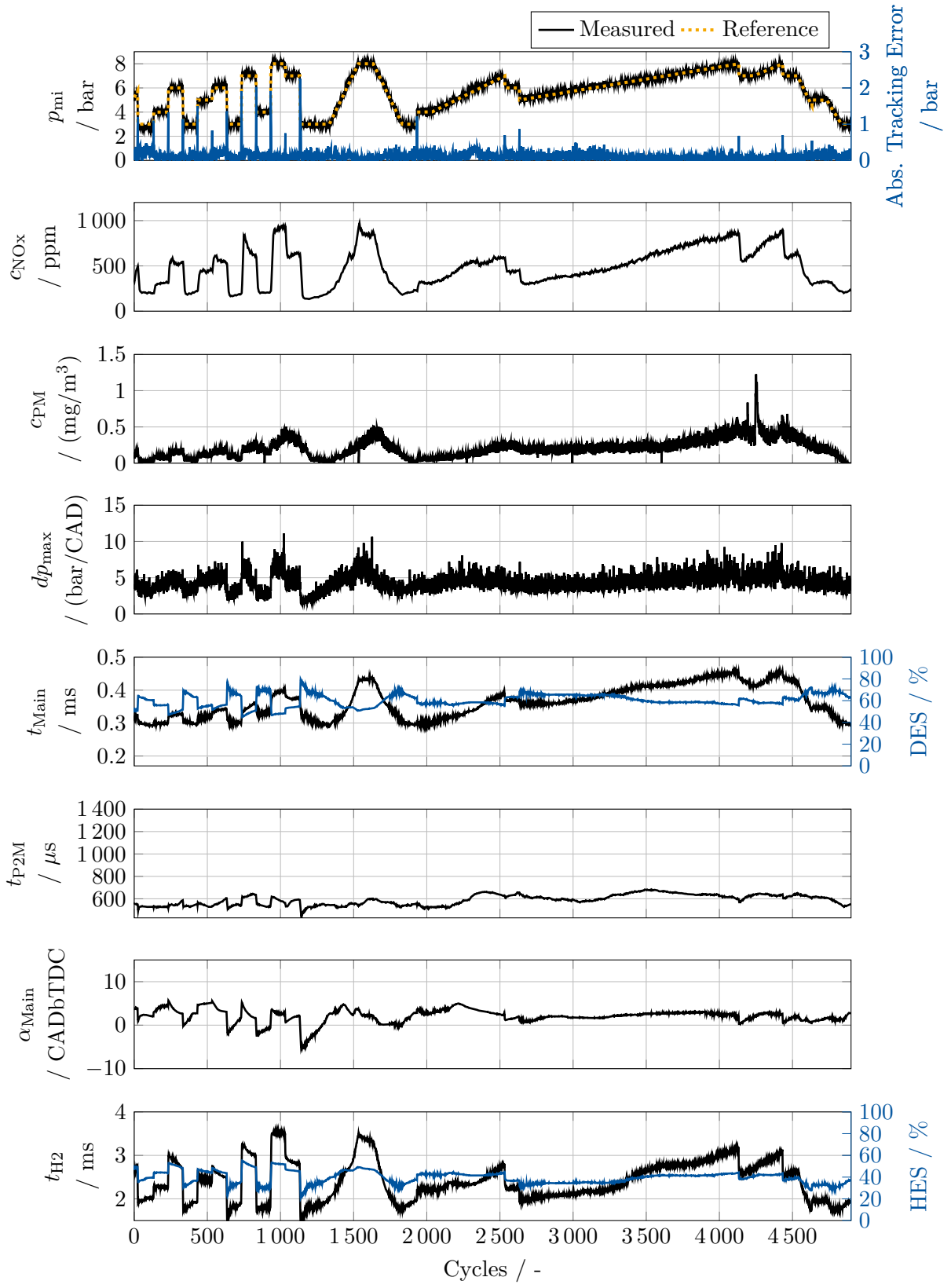


Figure 5.6: *Core MPC*. Experimental results. Tracking of the standard load reference trajectory of 4900 cycles at 1500 min^{-1}.

The measured p_{mi} closely follows the reference, while the mean absolute tracking error (MAE) remains generally under 0.12 bar, with occasional spikes during sudden load jumps, partly a consequence of the balanced controller aggressiveness setting. NO_x emissions c_{NOx} spike during transients and climbs to about 1000 ppm under high load, while the PM concentration c_{PM} remains below the constraint of 1.5 mg/m^3. As expected, soot emissions c_{PM} are significantly reduced compared to the diesel-only experimental results, while c_{NOx} increase. It is worth noting that this engine is not equipped with an EGR system, which could have helped suppress NO_x formation [55], [54], [160] and further improved emission results. $dp_{\max}$ stays under 5 bar/CAD except for brief peaks during rapid load changes and in the later stages of extended high-load operation. This demonstrates the controller's ability to maintain safe engine operation without excessive wear or noise. The pre-to-main t_{P2M} time and start of injection main α_{Main} vary within narrow bands to shape combustion. Finally, hydrogen duration of injection t_{H2} and hydrogen energy share (HES) surge during transients with the latter reaching around 40–60%, confirming flexible hydrogen/diesel blending. The diesel energy share (DES) is defined analogously to HES (see Eq. 2.3), using the energy content of the diesel injections, and is plotted together with t_{Main}. Overall, the MPC maintains precise load control, achieves comparably low soot emissions, tolerates a moderate c_{NOx} increase, seamlessly manages the fuel-blend ratio, and keeps engine wear and noise production within constraints.

Table 5.3 additionally summarizes these results in key performance metrics. Using the H2DF controller, the MAE for p_{mi} tracking falls to 0.117 bar (−27.8%) and the RMSE to 0.176 bar (−20.9%), corresponding to a 20.9% improvement in NRMSE. NO_x emissions c_{NOx}, however, increase substantially: the mean value rises by 105.1% to 482.8 ppm. In contrast, c_{PM} is greatly reduced, with the average dropping by 61.1% to 0.21 mg/m^3. The HES reaches as high as 55.1% and averages 39.7%, demonstrating significant hydrogen substitution without compromising load-tracking performance. Finally, this enables saving up to 55.1% total carbon dioxide emissions compared to diesel-only operation assuming identical combustion efficiency. Offloading

Table 5.3: *Core MPC.* Experimental results performance metrics. Tracking the standard load reference trajectory of 4900 engine cycles at 1500 min^{-1}. Results are compared to the diesel standard controller run.

Metric	**Result**	**Vs. diesel**
p_{mi} tracking		
MAE	0.117 bar	−27.8%
RMSE	0.176 bar	−20.9%
NRMSE	3.51%	−20.9%
c_{NOx}		
max	971 ppm	+166.4%
mean	483 ppm	+105.1%

Metric	**Result**	**Vs. diesel**
c_{PM}		
max	1.23 mg/m^3	−45.1%
mean	0.21 mg/m^3	−61.1%
$dp_{\max}$		
max	11.13 bar/CAD	+101.3%
mean	4.39 bar/CAD	+87.7%
HES		
max	55.1%	N/A
mean	39.7%	N/A

the most computationally intensive MPC optimization routines to the RPi reduces run-times to 3 ms to 7 ms (with $t_{\text{comp,max}}$) using up to three SQP iterations. This represents an approximately threefold speed-up compared to the 20 ms reported by Gordon et al. [35] on the MABX, which relied on a much shallower LSTM RNN and a simplified OCP formulation.

The *Core MPC* is further examined showing the MPC predictions in parallel with the measurements during a load step in Fig. 5.7. It can be observed that the MPC's internal DNN model predicts p_{mi} and c_{NOx} accurately during both steady-state and transient operation. The measured values (black) closely follow the predictions (blue), demonstrating a high degree of model accuracy. In contrast, c_{PM} exhibit a slight negative offset between 0.05 mg/m^3 and 0.15 mg/m^3. This deviation can likely be attributed to the limited accuracy of the sensor, which is known to be sensitive to ambient conditions and dependent on frequent maintenance (see Section 3.2). For both c_{PM} and dp_{max}, the DNN model does not fully capture all C2C scatter and variations. Instead, it smooths out higher-frequency dynamics and acts similarly to a low-pass filter. As discussed in Section 4, this behavior is expected due to the DNN's limited ability to model stochastic effects and short-term fluctuations. Focusing on the control task, and given the highly nonlinear dynamics of the engine, the prediction quality of the DNN and consequently the MPC can be considered effective.

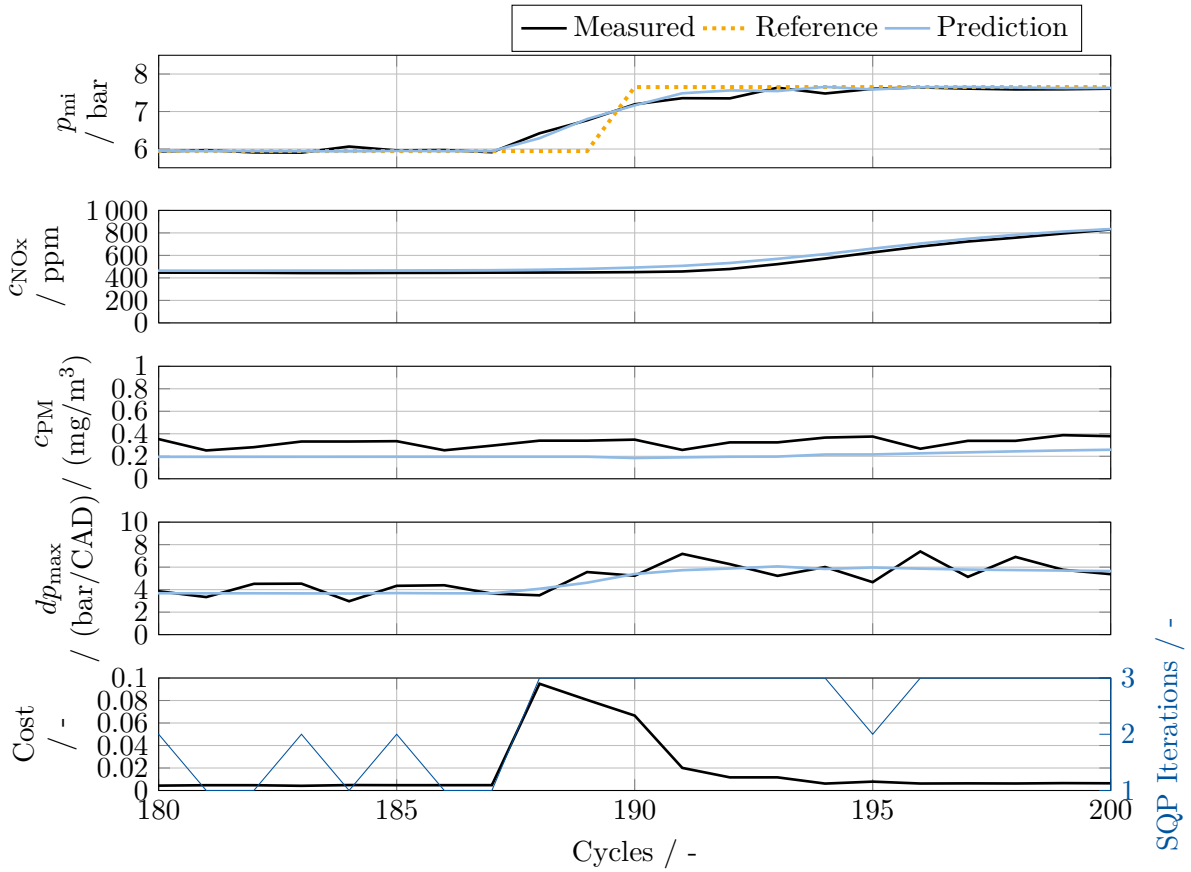


Figure 5.7: *Core MPC*. Experimental results. Extract on load step, highlighting MPC predictions and cost.

The bottom subplot visualizes the cost function value and the number of SQP iterations within the OCP framework `acados` per cycle. Both increase during the load step, with the SQP iterations reaching their maximum of three. This reflects the additional optimization effort required during transients and reaching higher loads to enforce constraints. In addition, the cost function remains low throughout the entire sequence, with a peak of only 0.1. This indicates that the model scaling and cost function are well-tuned, potentially contributing to numerically stable and balanced optimization.

To further illustrate the MPC's prediction behavior, Fig. A.10 shows the predicted outputs at one fixed time step k over the prediction horizon of $N = 3$. This further highlights the prediction quality discussed above, particularly in accurately capturing the system's short-term dynamics.

Fig. 5.8 illustrates the control behavior of the *Core MPC* during a representative engine load step, with a focus on the actuator trajectories and their respective change rates. The subsequent panels display the actuator trajectories for t_{Main} and t_{H2}, along with their respective change rates. It is evident that both t_{Main} and t_{H2} increase sharply during the step, particularly between cycles 730 and 735. This transient causes the change rates of both inputs to temporarily reach

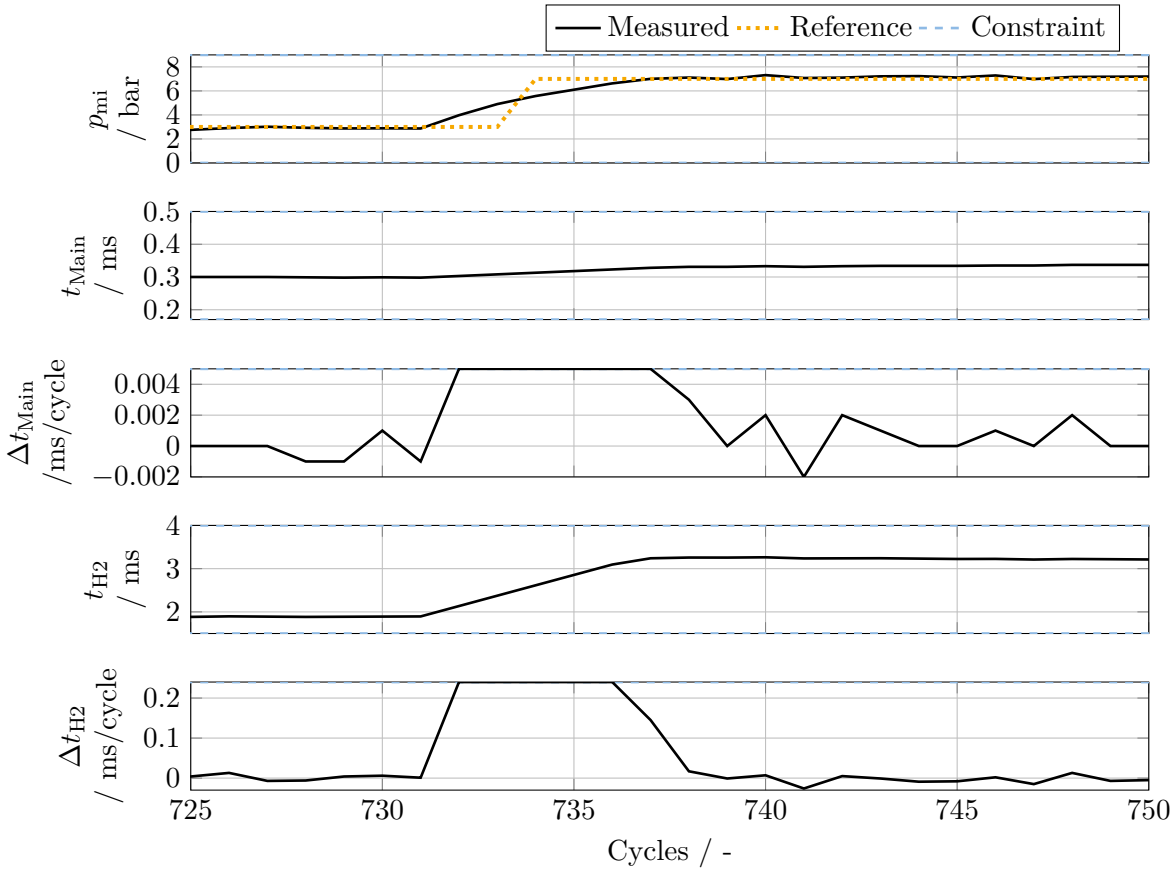


Figure 5.8: *Core MPC*. Experimental results. Extract on load step, highlighting controls change rates and constraints.

their upper constraints, as indicated by the red dashed constraint lines. Despite this, the MPC maintains constraint satisfaction while executing the necessary control actions. This highlights a key benefit of the chosen OCP formulation using the augmented state vector: the change rate constraints are handled directly within the optimization problem, allowing for aggressive yet constraint-compliant control. This makes it possible to limit how quickly the actuators are allowed to respond and thus change the overall controller aggressiveness. As a result, the MPC delivers a smooth response without violating constraints.

5.5.2 Variations of Cost Function & Model

In this subsection, variations of the MPC are presented, obtained by modifying either the cost function in the OCP or the underlying DNN model, to demonstrate the framework's flexibility and adaptability to different design objectives.

Cost Function Variation: Hydrogen Utilization

Firstly, the MPC cost function is varied in terms of increasing or reducing weights on the t_{H2} or t_{Main}, respectively. This leads to different tendencies of the controller to inject more or less hydrogen. As discussed in Section 3.4, this causes increased temperatures in the combustion chamber and subsequently higher $dp_{\max}$. Table 5.4 summarizes the results of three runs of the *Core MPC*.

Table 5.4: Experimental results comparison of varying H_2 utilization settings of the *Core MPC*. Tracking the standard load reference trajectory of 4900 cycles at $1500\,\mathrm{min}^{-1}$ with varying H2 utilization.

Metric	**Mid H_2 util.**	**Low H_2 util.**	**High H_2 util.**
p_{mi} tracking			
MAE / bar	0.122	0.205	0.132
RMSE / bar	0.184	0.276	0.201
NRMSE / −	3.68%	5.53%	4.02%
c_{NOx}			
max / ppm	971	829	1009
mean / ppm	474	417	487
c_{PM}			
max / mg/m^3	0.55	0.72	0.57
mean / mg/m^3	0.19	0.11	0.11
$dp_{\max}$			
max / bar/CAD	11.1	9.3	10.6
mean / bar/CAD	4.3	4.2	4.2
HES			
mean / −	40.17%	23.08%	44.89%
max / −	55.10%	42.66%	77.78%

The *mid H_2 util.* variants corresponds to the unchanged *Core MPC*, where as in the *low H_2 util.* and *high H_2 util.* variants weights are changed as discussed to achieve the lower or higher HES, respectively. Due to the changes in HES, c_{NOx} and c_{PM} vary between the three configurations, with *low H_2 util.* showing the lowest $\mathrm{NO_x}$ levels as expected. $dp_{\max}$ values remain within acceptable bounds, and no violation of physical constraints is observed. For the run with higher HES, it was quite critical to not hit the $dp_{\max}$ constraints, as in the later parts of the load reference trajectory with higher combustion chamber temperatures the cylinder tended to violate the $dp_{\max}$ thresholds due to heavy knocking. The HES reflects the primary difference between the scenarios: ranging from a modest 23% in *low H_2 util.* to a high of 45% in *high H_2 util.*, with corresponding maximum values nearing 78%. Fig. A.13 and Fig. A.12 visualize these experimental results graphically for the *low H_2 util.* and *high H_2 util.* variant, respectively.

Cost Function Variation: Control Change Rates

As mentioned previously, the control change rate constraints can be used to influence the controller behavior, particularly its aggressiveness. In two additional cost function configurations of the *Core MPC*, these constraints were respectively tightened and loosened. Fig. 5.9 shows an extract in the form of a step response from the standard load tracking trajectory for a *slow steps* variant.

The data shows that the MPC respects the configured constraints and subsequently exhibits either a more conservative or a more aggressive approach in reaching the requested p_{mi} setpoints. This demonstrates the high flexibility of the proposed MPC approach, potentially enabling easy switching between different engine driving modes, such as a "comfort mode" (*slow steps* variant) and a "sport mode" (*fast steps* variant). Table A.11 presents the performance metrics from the full runs on the standard load reference trajectory for the aforementioned variants, including a comparison to the *Core MPC*. The experimental results of the *fast steps* variant are depicted in Fig. A.11.

Model Variation: Feedforward

In addition to modifying the cost function to generate MPC variants with different behaviors, this work also explores a DNN model architecture that enables feedback-independent control. While it is expected that a controller without process feedback will generally perform worse, feedback-independent operation can be desirable in scenarios where sensor reliability is limited or sensor data is unavailable. Production engines generally do not provide direct in-cylinder pressure feedback such as p_{mi}, so the controller has to remain effective without this feedback signal.

To this end, a *no-feedback* model was developed using the same dataset and architecture as the main DNN introduced in Chapter 4, with one key difference: the removal of the feedback input feature $p_{\mathrm{mi}}(k-1)$. All other features and the model structure were kept identical. The resulting model achieved similar performance during training, with an MAE of 0.14 bar and an

$\text{NRMSE}_{\text{clipped}}$ of 3.0% on the unseen test dataset (see Table A.8 for full details). Additional insights into the training process and model behavior can be found in the loss-iteration plot (Fig. A.5), the corresponding parity plot (Fig. A.6) and the time-domain predictions on the test dataset (Fig. A.7).

The experimental evaluation of the *no-feedback* MPC variant is shown in Fig. 5.10. As anticipated, the absence of closed-loop correction led to slight static tracking offsets and periodic oscillations around the reference trajectory. These deviations highlight the expected limitations in prediction and adaptation without measurement-based feedback. Table 5.5 summarizes the performance metrics and compares them against the *Core MPC*. Despite degraded p_{mi} tracking performance with the NRMSE rising to 6.72%, a 91.4% increase over the *Core MPC*, the results remain within acceptable bounds. NO_x and soot emissions c_{NOx}, c_{PM} are slightly lower with the *no-feedback* variant, which can be associated with shifted operating conditions relative to the closed-loop *Core MPC* with the mean p_{mi} being about 4% lower. Interestingly, the mean HES remains virtually unchanged at 42.15%, showing that feedforward control alone can sufficiently maintain the dual-fuel approach. These results underscore the relevance of feedback integration

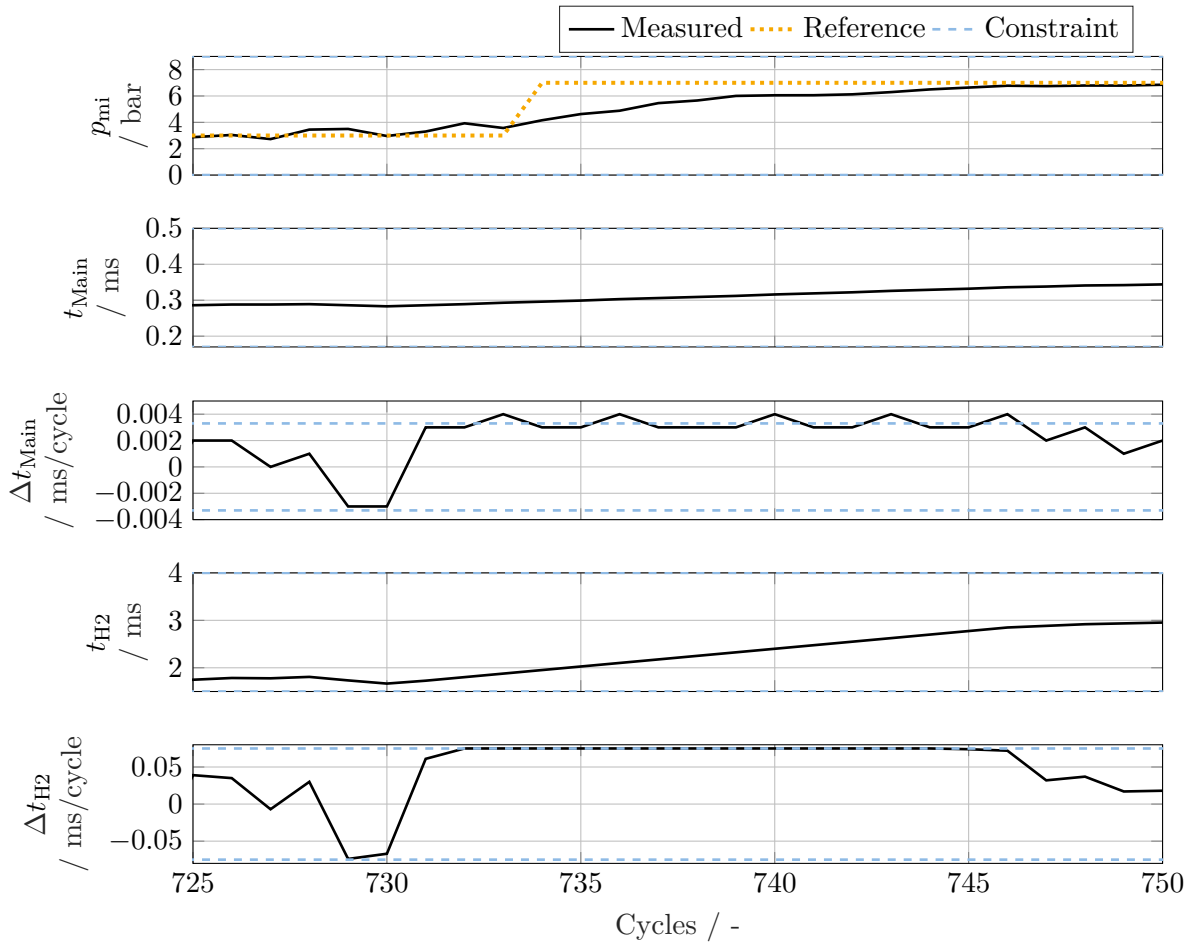


Figure 5.9: MPC *slow steps* variant. Experimental results load tracking. Controls change rates extract including constraints.

for accurate and responsive control under dynamic conditions. At the same time, they demonstrate the practical viability of feedback-free MPC configurations in constrained combustion systems, especially when robustness to sensor faults or absences is prioritized.

This naturally leads to the next subsection, which focuses on evaluating the robustness of the *Core MPC*.

Table 5.5: Experimental results comparison of the *no-feedback* variant with the *Core MPC*. Tracking the standard load reference trajectory of 4900 cycles at $1500\,\mathrm{min}^{-1}$. fb: feedback.

Metric	***No fb***	**Vs. *Core MPC***
p_{mi} tracking		
MAE	0.220 bar	+87.9%
RMSE	0.336 bar	+91.4%
NRMSE	6.72%	+91.4%
c_{NOx}		
max	959 ppm	−1.2%
mean	433 ppm	−10.4%
c_{PM}		
max	$0.86\,\mathrm{mg/m^3}$	−30.1%
mean	$0.15\,\mathrm{mg/m^3}$	−28.6%
dp_{max}		
max	9.14 bar/CAD	−17.9%
mean	3.71 bar/CAD	−15.5%
HES		
max	55.22%	+0.2%
mean	42.15%	+6.3%

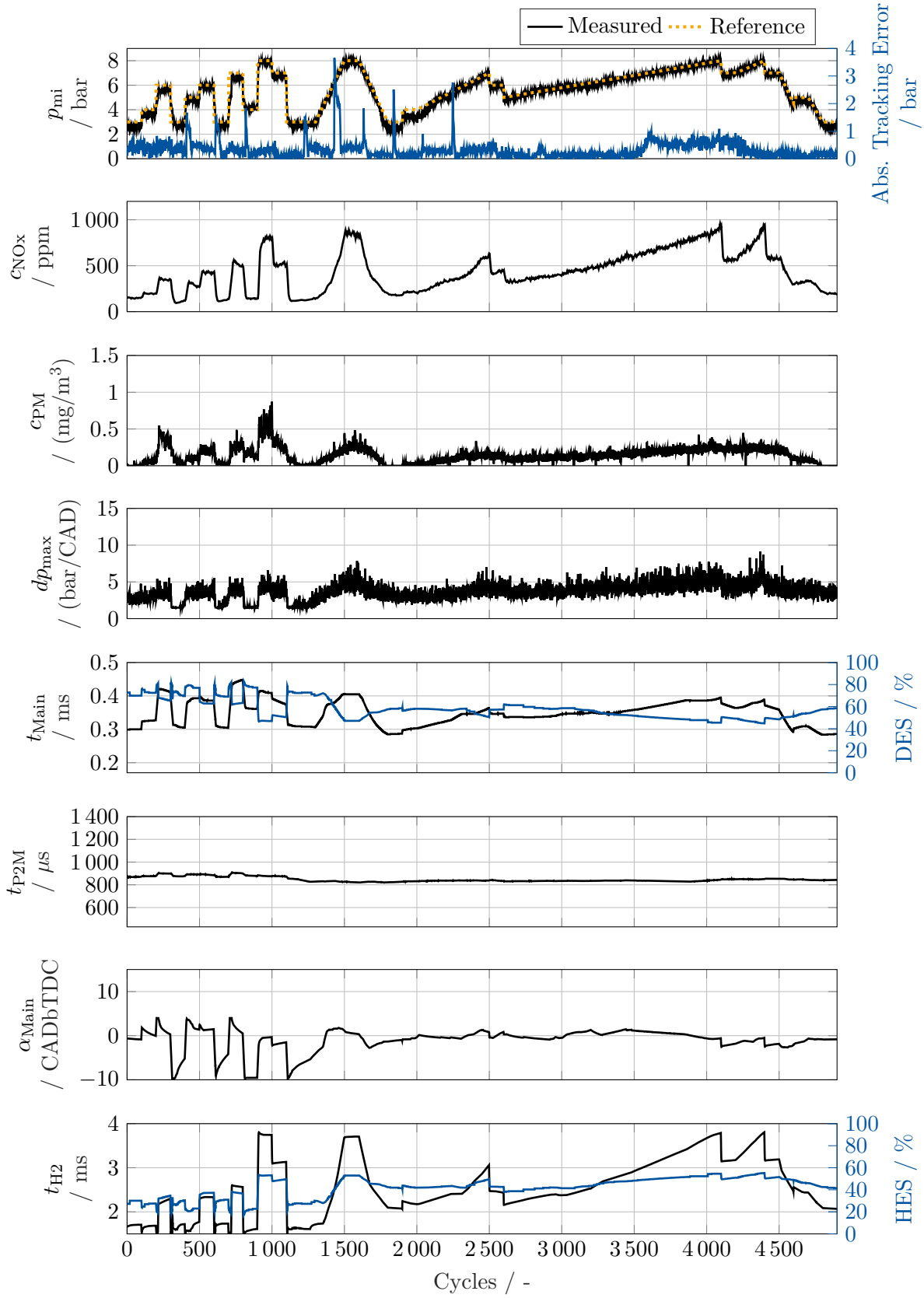


Figure 5.10: MPC *no feedback* variant. Experimental results load tracking.

5.5.3 Robustness Investigation

This section analyzes the controller's robustness, divided into two parts: modeled dynamics and unmodeled dynamics. Modeled dynamics cover perturbations considered during design including applying noise to the modeled inputs, while unmodeled dynamics address effects not captured by the model, such as the real-world deviations at the engine test bench.

Modeled Dynamics

To empirically assess the robustness of the DNN-based prediction model embedded in the MPC, a local sensitivity analysis in Monte-Carlo fashion is performed by perturbing the input space across a range of baseline conditions and GRU states. [119] The pseudo-code of the sensitivity analysis is shown in A.4. Due to the normalized nature of the input data, the perturbation range was set to $\Delta = [0, 1]$ during the sensitivity analysis.

Fig. 5.11 shows the sensitivity matrix visualized as a heat-map of the MPC DNN model (see Section 4.2). The rows represent the model outputs, while the columns correspond to the input variables. Each cell quantifies the local sensitivity of a specific output with respect to a particular input, with higher values indicating a stronger influence. The results show that p_{mi} is highly sensitive to t_{H2}, followed by t_{Main} and α_{Main}. In contrast, emissions outputs like c_{NOx} and c_{PM} display lower sensitivities across all inputs. The maximum pressure rise rate dp_{max} is moderately sensitive, particularly to α_{Main}.

This information supports constraint tuning for the MPC, suggesting tighter limits on sensitive inputs like injection timings to protect safety-critical outputs such as p_{mi}. In contrast, constraints on t_{P2M} could be relaxed due to its low overall influence. This result further highlights the successful training of the DNN, as it effectively maintains the primary control objective of tracking p_{mi} and additionally mirrors the expected combustion characteristics.

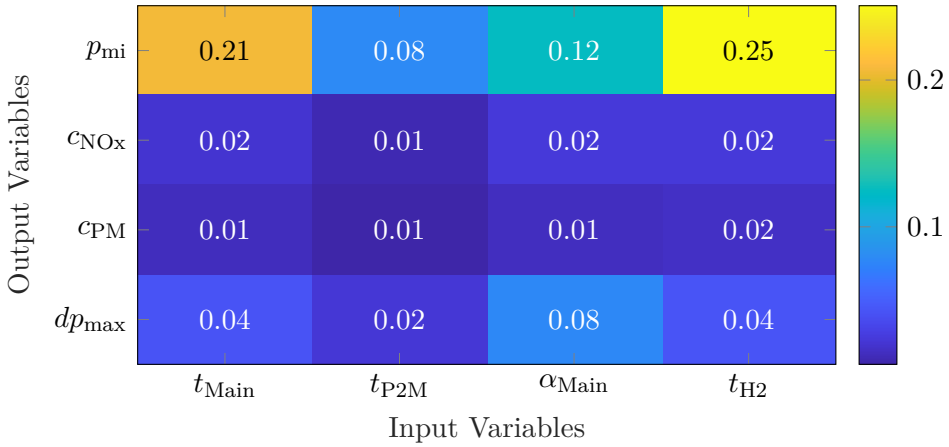


Figure 5.11: DNN-based engine model. Sensitivity matrix of model outputs with respect to input variables, visualized as a heat-map. Higher values indicate stronger local influence.

In addition to the sensitivity analysis, the robustness of the model was further investigated by applying various types of disturbances to the DNN in the MiL simulation presented in Section 5.4. The results are depicted in form of "robust" areas of the MPC in Table A.13, evaluated using physical limits like dp_{max} and `acados` solver failures. The MPC demonstrated robust performance under multiple artificial perturbations, including zero offsets, and additive white Gaussian noise injected into engine outputs, as well as control-signal and span-offset disturbances applied to the engine inputs. As expected, the MPC showed larger sensitivity to noise on p_{mi}, reflecting the DNN model's dependency on this feedback signal. When applying strong disturbances, such as a constant offset of up to 0.5 bar or additive white Gaussian noise with a noise power of 0.57 bar^2 (equivalent to a standard deviation of 0.75 bar), the state initialization could lead to constraint violations at the first prediction step, most noticeably for dp_{max}. Apart from these extreme cases, the MPC handled disturbances well. Although further visualizations are omitted here due to the high dimensionality of the results, a detailed analysis is provided in [161].

To complement the simulation results, these findings were also validated at representative operating points by introducing perturbations into the feedback or control signals on the RPi. For example, additional white Gaussian noise was injected into the p_{mi} feedback channel. Fig. 5.12 shows the response along the standard load trajectory when the noise power was artificially increased to 0.03 bar^2, resulting in an extreme amplitude of ± 0.5 bar, to assess the MPC's robustness limits. Even with apparent unrealistic large noise injected as seen here, the MPC kept load control on target. Only at higher load operating points, where the wildly fluctuating feedback pushed against the dp_{max} constraints, tracking slipped. Around engine cycle 950, the approach towards the dp_{max} constraint causes the MPC to reduce p_{mi} conservatively as a pre-

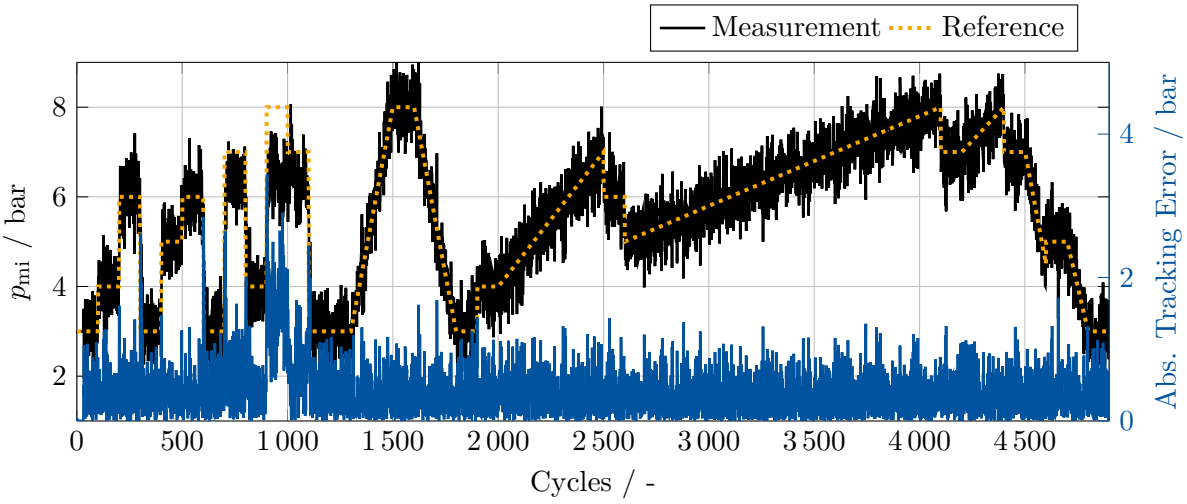


Figure 5.12: *Core MPC*. Experimental results, tracking under high noise injection on the p_{mi} feedback value. Tracking the standard load reference trajectory of 4900 cycles at 1500 min^{-1}. White-Gaussian noise with an amplitude of $\pm$ 0.5 bar, resulting from 0.03 bar^2 noise power.

ventive action. This resilient behavior underscores the MPC's robustness and its suitability for wider deployment.

Unmodeled Dynamics

To further evaluate the MPC's robustness, unmodeled dynamics were introduced by varying the plant, specifically the engine speed n_{Eng}. The MPC DNN model had been trained using data at a constant engine speed of $1500\,\mathrm{min}^{-1}$.

Fig. 5.13 shows an extract of a measurement where the *Core MPC* tracked a smooth reference trajectory varying between 3 bar and 8 bar. In parallel, the engine speed n_{Eng} was varied with smooth ramps between $1200\,\mathrm{min}^{-1}$ and $1800\,\mathrm{min}^{-1}$ to investigate the MPC's ability to maintain engine load tracking on p_{mi} under these significant plant dynamic changes. The data shows that with increasing engine speed, NO_x emissions slightly decrease due to reduced residence time of air at high temperatures, and conversely increase with reduced engine speed. It is also observable that soot formation increases at higher loads when the engine speed is raised, likely due to reduced time available for mixture formation without corresponding injection advancement – a behavior the MPC's DNN had not learned. Regarding p_{mi} load tracking, the MPC demonstrates robust behavior, maintaining close reference tracking with the absolute p_{mi} tracking error remaining below 0.5 bar.

Additionally, measurements of the tracking performance of the *Core MPC* were conducted on the standard 4900 cycle load profile at different constant engine speeds: $1200\,\mathrm{min}^{-1}$, and $1800\,\mathrm{min}^{-1}$ (see Fig. A.14 and Fig. A.15). The p_{mi} tracking performance worsened slightly compared to the *Core MPC* trained at $1500\,\mathrm{min}^{-1}$, with the NRMSE increasing from 3.51% to 5.54% ($1200\,\mathrm{min}^{-1}$) and 5.85% ($1800\,\mathrm{min}^{-1}$), while still maintaining acceptable tracking performance. The previously discussed observations regarding c_{NOx} and c_{PM} emissions and dp_{max} are confirmed here as well. Table A.12 provides a detailed overview of all performance metrics.

The significant changes to the plant (engine) combined with the maintained acceptable tracking performance of the MPC underline its robust control capability. This robustness paves the way for potential future applications, such as integration into production control systems or the generation of valuable fitting data for traditional control methods like LUT-based approaches.

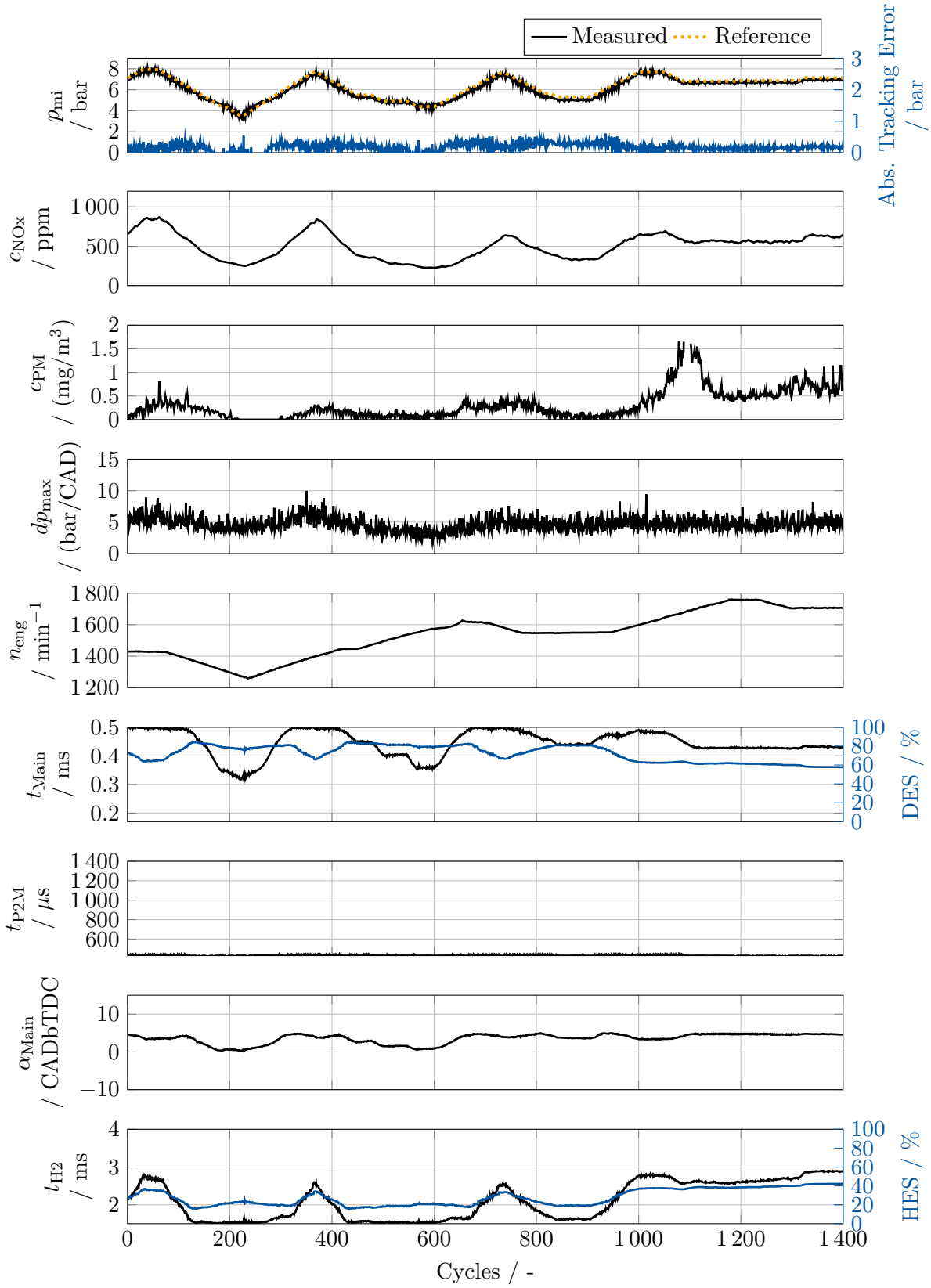


Figure 5.13: *Core MPC*. Experimental results, load tracking with model-plant mismatch (engine speed variation between 1200 min^{-1}, and 1800 min^{-1}).

5.6 Summary of the Chapter & Discussion

This chapter presents a methodical proof-of-concept for a nonlinear MPC framework that uses previously developed DNNs to safely control H2DF combustion while saving carbon dioxide emissions compared to the baseline diesel-only control. Building on the challenges identified in the introduction, the proposed framework delivers a powerful, flexible and real-time capable solution grounded in both simulation and experimental validation.

Using a GRU-based DNN model trained on experimental data, the MPC replaces conventional physical models with data-driven dynamics and integrates them into an OCP formulation with augmented controls and soft constraints solved with `acados`. The controller achieves robust tracking of a newly introduced and unseen dynamic engine load trajectory, balancing emissions, fuel flexibility, and engine wear in form of $dp_{\max}$. Several MPC variants are explored, each with specific trade-offs in hydrogen utilization and controller aggressiveness. Notably, both feedback and feedback-free controller designs are validated, underscoring adaptability to sensor availability. Additionally, robustness is empirically confirmed through a sensitivity analysis, MiL simulations and test-bench experiments, including high noise and engine speed variation tests.

The outsourcing optimization to a RPi enables real-time execution and modular deployment, allowing controller development from a scratch in under a day. Without altering the main engine code, the controllers can be changed on-the-fly. This accelerates development, testing, and adaptation and highlights the advantages of both the data-driven approach and the modular toolchain.

However, several limitations remain. The controller design and evaluation are empirical and prototypical in nature. No formal robustness guarantees, such as Lyapunov-based stability proofs, were established. The lack of a principle-based robustness framework limits the generalizability of the results and hinders formal certification. Future work should aim to combine empirical performance with formal guarantees. Additionally, even on an overclocked RPi with its 2.2 GHz clock-rate (a higher clock frequency than most typical automotive engine control units), the current MPC implementation remains too computationally intensive for broad deployment. Furthermore, because the RPi runs a full operating system, it can only guarantee soft real-time behavior and does not meet the deterministic timing or functional-safety standards required for production automotive controllers. These points raise the necessity to develop faster advanced control algorithms to run on slower automotive or low-cost embedded systems with significantly less computational power. While the automotive industry often relies on LUT, PI or linearized systems to reduce the computational demand, this work aims to integrate data-driven safe control algorithms able to handle nonlinear fast system dynamics on low-cost embedded systems. In the following chapter, the approach of utilizing behavior cloning to bridge this gap is presented.

6 Behavior Cloning of Expert Model Predictive Controller

The previous chapter highlighted the potential and limitations of advanced control strategies such as model predictive control (MPC) when deployed on embedded systems. While effective, MPC remains computationally demanding, often requiring powerful processors that exceed the capabilities of typical low-cost or automotive-grade hardware. These constraints emphasize the growing need for fast, efficient control algorithms that can operate on low-cost controllers without compromising safety or control quality.

Subsequently, this chapter presents the use of the purely data-driven method behavior cloning (BC) for controlling the H2DF engine, using demonstration data generated by the expert MPC developed in the previous chapter. The objective is to provide a lightweight alternative capable of handling the engine's nonlinear combustion dynamics on low-cost embedded hardware.

The next section provides an introduction including a brief literature review on imitation learning (IL) and BC as a subset of IL in the context of real-world systems and especially engine control, summarizing relevant applications of data-driven policy learning. The chapter's contribution is highlighted, and its structure is outlined.

This chapter is partially based on [34], [48].

6.1 Introduction to the Chapter

IL has received considerable attention in the context of autonomous driving. A landmark result by Bojarski et al. [162] (2016) showed that a deep convolutional network can be trained end-to-end to map front-camera images to steering commands, cloning human lane-following on a real vehicle. However, basic BC can struggle in complex scenarios, so later works introduced strategies to increase robustness. Codevilla et al. [163] (2018) proposed conditional IL, where the policy network takes a high-level navigation command as input and branches its output accordingly; this approach was tested in realistic simulators. Bansal et al. [164] (2019) augmented BC with synthesized off-policy data (perturbed expert trajectories) and auxiliary losses to penalize infractions; their policy was robust enough to drive a real car on a test track.

In robotics, IL has enabled robots to acquire complex skills from demonstrations. Zhang et al. [165] (2018) used virtual-reality-collected human demonstrations and BC to train a deep policy, enabling a robot to perform multi-step manipulation tasks from raw camera inputs. To improve reliability, Laskey et al. [166] (2017) injected noise into expert actions so the learned policy experiences and corrects off-track states; this strategy was validated on physical robot tasks. Recent work also addresses sub-optimal data: Zhang et al. [167] (2021) introduced a

confidence-weighted IL method that learns to down-weight low-quality demonstrations during training, yielding better performance (evaluated in simulation and on a real robot).

In summary, BC and IL have made substantial progress, particularly in autonomous driving and robotics, with real-world deployments confirming their feasibility beyond simulation. Despite this success, generalization and safety in open-ended environments remain key challenges. A central issue in IL is the compounding error caused by distribution shift, where the policy encounters states not represented in the expert data. This has motivated extensive research into robustness-enhancing strategies. Notably, to the best of the author's knowledge, no study to date has applied BC or IL to real-world engine control, marking a significant research gap that this chapter aims to address.

In response, this chapter introduces the BC concept as an alternative control technique to replace the MPC for H2DF engine control. First, the control objective and the overall BC-based approach are described. Then, the modeling process is briefly summarized. The implementation of the resulting controller is subsequently discussed. Finally, the chapter presents experimental results from engine test bench evaluations, as well as an additional benchmarking study that compares the runtime performance of the trained networks on embedded platforms such as the low-cost *ESP32*.

Fig. 6.1 further illustrates the development process for the online BC controller designed here. In the first step, the deep neural network (DNN)-based MPC developed in Chapter 5 generates optimal control inputs $u(k)$ in response to a reference trajectory. These are applied to the engine plant, and the resulting outputs $y(k)$ are measured and recorded to form the dataset $\{u(k), y(k)\}$ for training the BC controller. In the second step, this experimental dataset is used to train a DNN policy that learns to replicate the behavior of the MPC expert. The network is trained in a supervised manner, using the recorded data to minimize the difference between its predicted control actions and the expert's actions. In the final step, the trained BC controller is deployed and evaluated directly on the experimental setup – the H2DF engine; taking the reference and observed states as inputs and producing control actions in real-time. Additionally, a benchmarking of multiple DNNs is conducted to assess run-time performance and suitability on a low-cost embedded controller.

To conclude, the chapter offers a summary and a discussion that reflects on how the developed BC method addresses the overarching objectives of this thesis.

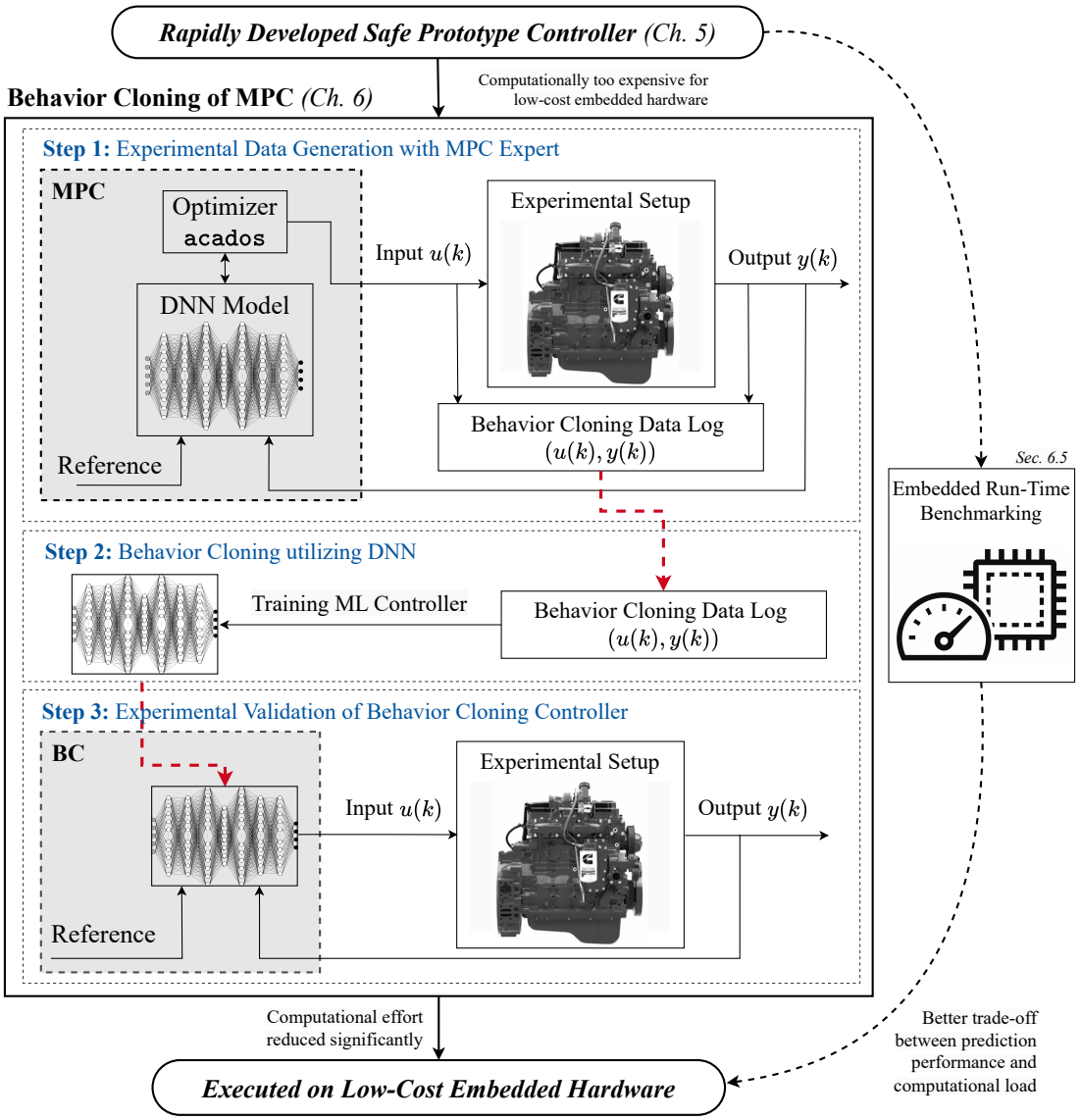


Figure 6.1: Development process of online BC controller for H2DF based on previously developed expert MPC.

6.2 Control Concept & Approach

The MPC control law developed in Chapter 5 determines optimal control actions by solving a constrained optimization problem at each time step. Formally, the MPC is defined as follows:

- **State space MPC:** $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, where the system state at time step k is $x(k) \in \mathcal{X}$.
- **Action space MPC:** $\mathcal{U} \subseteq \mathbb{R}^{n_u}$, where the control action at time step k is $u(k) \in \mathcal{U}$.

The MPC policy ($\pi_{\text{MPC}} \colon \mathcal{X} \to \mathcal{U}$) can be defined as the solution to the constrained optimization problem (simplification of the optimal control problem in Eq. 5.13):

$$\pi_{\text{MPC}}(x(k)) = \underset{\substack{u_{k-1},\dots,u_{k+N-1} \\ x_k,\dots,x_{k+N}}}{\arg\min} \quad J\,. \tag{6.1}$$

Since MPC optimizes a sequence of future control actions but only applies the first, the MPC control law follows a receding horizon approach:

$$u(k) = \pi_{\text{MPC}}(x(k)) \,. \tag{6.2}$$

Following this formal definition of the MPC expert control policy, the BC problem is defined. The goal is to approximate the MPC policy (π_{MPC}) by learning a policy using a DNN, where:

- **State space BC:** $\mathcal{S} \subseteq \mathbb{R}^{n_s}$, where the system state at time step k is $s(k) \in \mathcal{S}$, with $\mathcal{S} \subseteq \mathcal{X}^{n_x}$.
- **Action space BC:** $\mathcal{A} \subseteq \mathbb{R}^{n_a}$, where the control action at time step k is $a(k) \in \mathcal{A}$, with $\mathcal{A} \subseteq \mathcal{U}^{n_u}$.

The BC controller directly learns a DNN policy ($\pi_{\theta,\text{BC}}$), parameterized by network parameters θ, that maps states to actions:

$$\pi_{\theta,\text{BC}} \colon \mathcal{S} \to \mathcal{A} \,. \tag{6.3}$$

Given an input state $s(k)$, the BC policy $\pi_{\theta,\text{BC}}$ predicts the corresponding control action:

$$a(k) = \pi_{\theta,\text{BC}}(s(k)) \,. \tag{6.4}$$

The BC policy $\pi_{\theta,\text{BC}}$ is trained on a dataset of expert demonstrations collected from the MPC controller:

$$\mathcal{D} = \{(s_i,\ a_i)\}_{i=1}^{N}, \tag{6.5}$$

where $s_i \in \mathcal{S}$ denotes an observed system state, and $a_i \in \mathcal{A}$ is the corresponding expert action computed by MPC. Note that $\mathcal{S} \subseteq \mathcal{X}^{n_x}$ and $\mathcal{A} \subseteq \mathcal{U}^{n_u}$.

The objective is to find the policy parameters θ that minimize the discrepancy between the predicted actions $\pi_{\theta,\text{BC}}(s_i)$ and the expert actions a_i. This results in the supervised-learning problem:

$$\theta = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(\pi_{\theta,\text{BC}}(s_i),\ a_i) \,, \tag{6.6}$$

where $\mathcal{L}$ is a loss function (e.g. MSE) as defined in Section 2.3.

The input and output domains of the BC controller are specified as:

$$\mathcal{S} \subseteq \mathbb{R}^{n_s=4}, \quad \text{where} \quad s(k) = \Big[p_{\text{mi,ref,1}}(k),\ p_{\text{mi,ref,2}}(k),\ p_{\text{mi,ref,3}}(k),\ p_{\text{mi}}(k-1)\Big]^{\top}, \tag{6.7}$$

$$\mathcal{A} \subseteq \mathbb{R}^{n_a=4}, \quad \text{where} \quad a(k) = u(k) = \Big[t_{\text{Main}}(k),\ t_{\text{P2M}}(k),\ \alpha_{\text{Main}}(k),\ t_{\text{H2}}(k)\Big]^{\top}. \tag{6.8}$$

The BC controller is designed to imitate the control actions and structure of the MPC, including its inputs and outputs. To reduce complexity, the input space is restricted to the four most relevant features: the target reference trajectory of indicated mean effective pressure $p_{\text{mi,ref,i}}$ for $i \in \{1, 2, 3\}$, and the engine load feedback $p_{\text{mi}}(k-1)$. To enable a direct replacement of the

MPC with the BC controller, the same control outputs $u(k)$, as defined in Eq. 5.1, are used as the target outputs of the DNN during training.

In the following section, the BC DNN policy modeling and training as well as the underlying datasets are briefly explained.

6.3 Deep Neural Network Policy Modeling

To form the BC policy, a DNN is subsequently trained using the experimental data generated by the expert, the MPC. The network architecture consists of an input layer, several fully connected (FC) layers arranged in an encoder-decoder structure with rectified linear unit (ReLU) activation functions, and an output layer, similar to the one used in the MPC. Fig. 6.2 illustrates the relatively simple structure of the DNN. Due to the repetitive nature of this DNN training compared to the detailed procedure presented in Chapter 4, only a brief summary is provided.

The DNN comprises a total of $21,460$ learnable parameters, roughly threefold the size of the DNN utilized in the previously developed MPC. However, the simple forward propagation of the DNN inputs to generate outputs or control signals when used as a BC controller significantly reduces computation time.

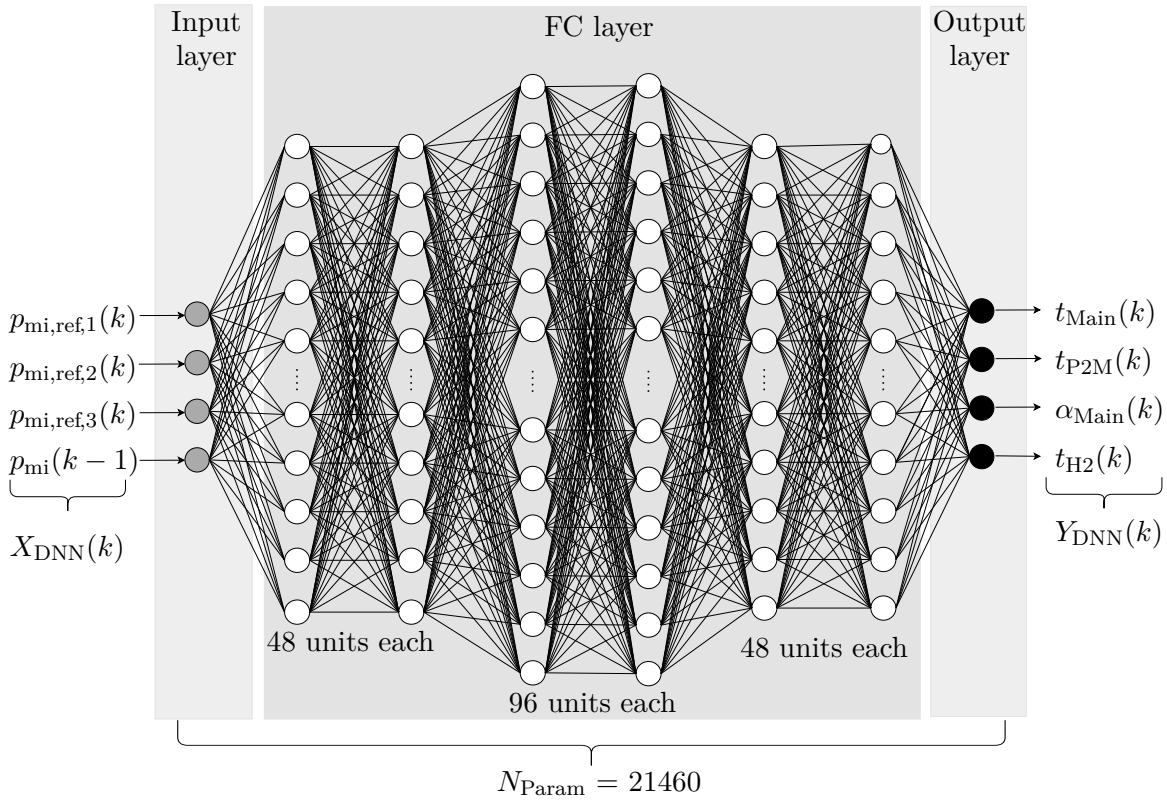


Figure 6.2: DNN-based BC controller. DNN architecture, FC layers with ReLU activation functions.

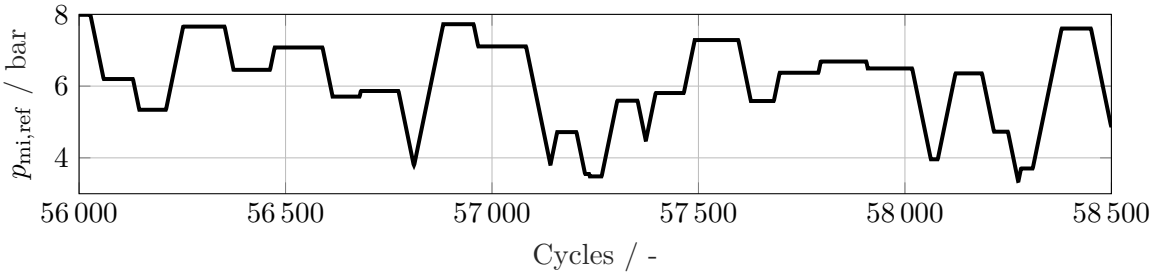


Figure 6.3: DNN-based BC controller. Data generation extract of ramp reference trajectory for p_{mi} load tracking. Randomly generated smoothened steps between 3 bar and 8 bar, utilized for data generation by MPC expert.

To generate the training data, the MPC actuates the engine, thus generating a total of 86,000 cycles or data points while tracking a ramp reference trajectory on the engine load p_{mi}. The load trajectory randomly varied within the range of 3 bar to 8 bar p_{mi}, while keeping the rate of change of the reference limited to 0.05 bar per cycle. This ensured safe and consistent operation of the MPC, only limited by the hydrogen storage. An extract of the trajectory is shown in Fig. 6.3. The resulting dataset, comprising a total of 86,000 engine cycles or samples N from three different MPC-driven experimental measurement runs, is sequentially divided into training, validation, and test sets for each measurement run. The overall dataset is split into proportions of 80%, 15%, and 5%, respectively.

Fig. A.16 shows the measured engine inputs and engine outputs in time-domain while the data distribution of the input with the highest variance, the engine load feedback $p_{\mathrm{mi}}(k-1)$, is illustrated in Fig. 6.4. The data demonstrates consistency of p_{mi} distribution across the different datasets, ensuring comparability for training and evaluation. The data distribution for $p_{\mathrm{mi,ref,i}}$ is even more uniform and is therefore omitted here for brevity. All data is normalized using the total datasets minimum and maximum values.

The training environment is the same as the one used for the MPC's DNN training, utilizing `MATLAB/Simulink` and the `Deep Learning Toolbox`, with hyperparameters listed in Table A.14. While the DNN dynamics in the MPC had to be manually converted into mathematical equations, the chosen environment here enables a simple integration of the BC controller into the engine setup using the `Embedded Coder`. The resulting loss-epoch plot of the training is shown in Fig. A.18. The prediction results on the unseen test dataset are illustrated in Fig. A.19 (parity plot) and Fig. 6.5 (time-domain), demonstrating the solid performance of the identified DNN. Additional performance metrics are provided in Table 6.1.

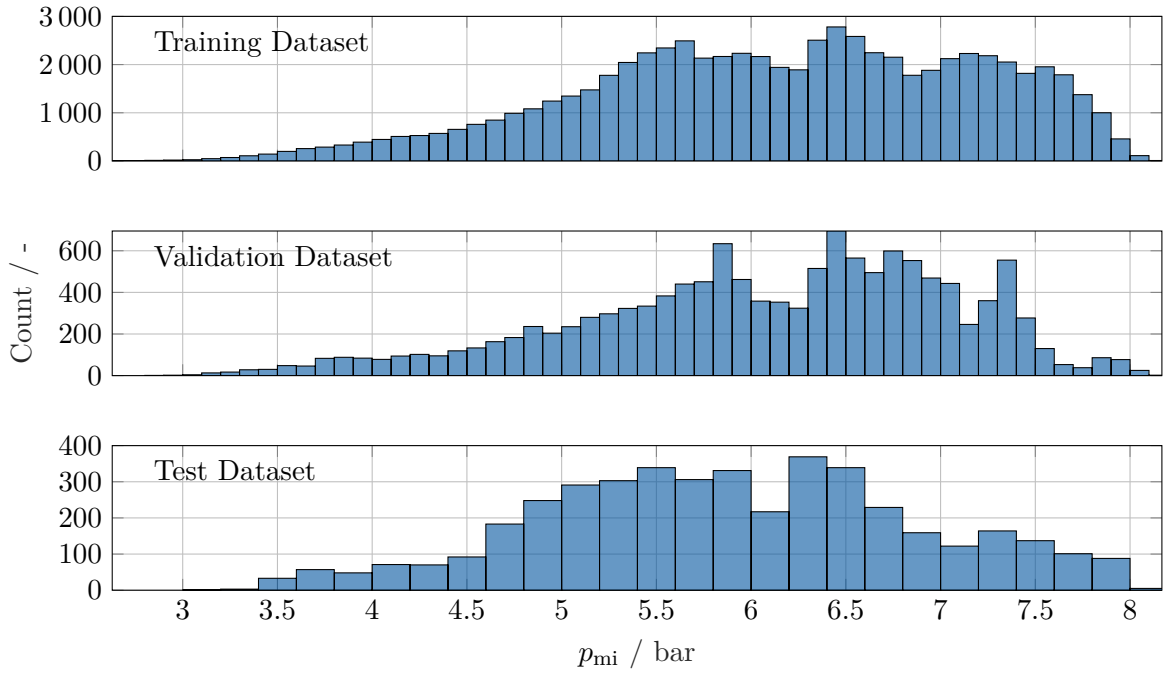


Figure 6.4: DNN-based BC controller. Experimental data, distribution to datasets for p_{mi}. Generated by MPC expert policy. Upper plot: training dataset (80%), mid plot: validation dataset (15%), lower plot: unseen test dataset (5%). Total data points: 86,000.

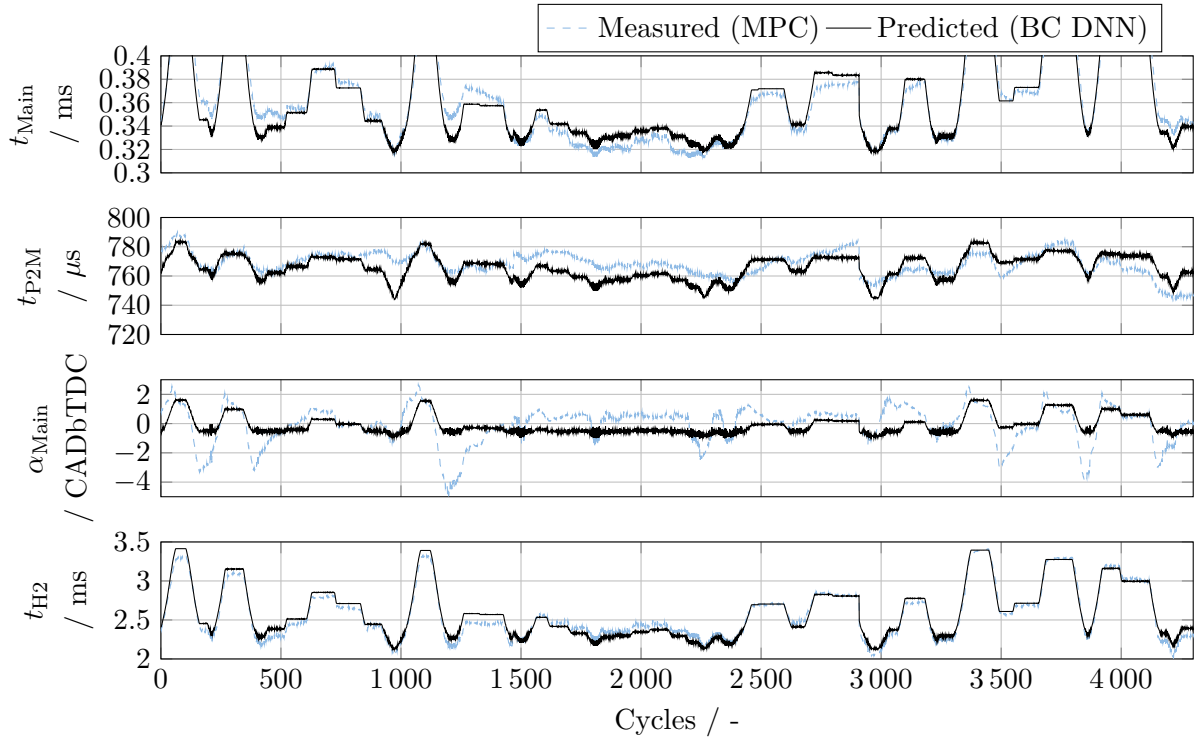


Figure 6.5: DNN-based BC controller. Predicted outputs in time domain.

Table 6.1: DNN-based BC controller. DNN prediction performance metrics on validation and unseen test datasets.

Metric	Validation	Test
t_{Main}		
MAE / ms	0.007	0.007
NMAE / −	3.70%	3.66%
RMSE / ms	0.009	0.008
NRMSE / −	4.77%	4.48%
t_{P2M}		
MAE / µs	10.52	6.31
NMAE / −	2.45%	2.40%
RMSE / µs	12.92	7.80
NRMSE / −	4.92%	2.97%

Metric	Validation	Test
α_{Main}		
MAE / CADbTDC	0.72	0.82
NMAE / −	7.67%	7.72%
RMSE / CADbTDC	0.96	1.10
NRMSE / −	8.93%	10.29%
t_{H2}		
MAE / ms	0.047	0.055
NMAE / −	3.20%	3.20%
RMSE / ms	0.060	0.066
NRMSE / −	3.47%	3.80%

The results show good correspondence with the MPC data, achieving an NRMSE of 4.5% and 3.8% for the tracking on unseen data of the two most important controls with highest correlation to the outputs, the duration of injection of both diesel and hydrogen, respectively. The DNN demonstrates limited accuracy in predicting the injection timing, the start of the main injection main α_{Main}. This behavior can be attributed to the high variance of the data to be predicted combined with the low variance of the input data, and the minor influence of the injection timing on the process outputs. Thus, this feature-selection combined with the data can be considered challenging, with many DNN trainings actually diverging.

Following these DNN developments, the engine test bench results are presented in the following.

6.4 Experimental Results on the Test Bench

To provide a basis for comparison with the BC controller, the tracking performance of the *Core MPC* is first evaluated on the newly introduced, unseen fast step load reference trajectory, as illustrated in Fig. 6.6, covering 2400 cycles at $1500\,\mathrm{min}^{-1}$. The trajectory is introduced to simplify the evaluation setup while emphasizing higher dynamics, thereby posing significant challenges to the controllers.

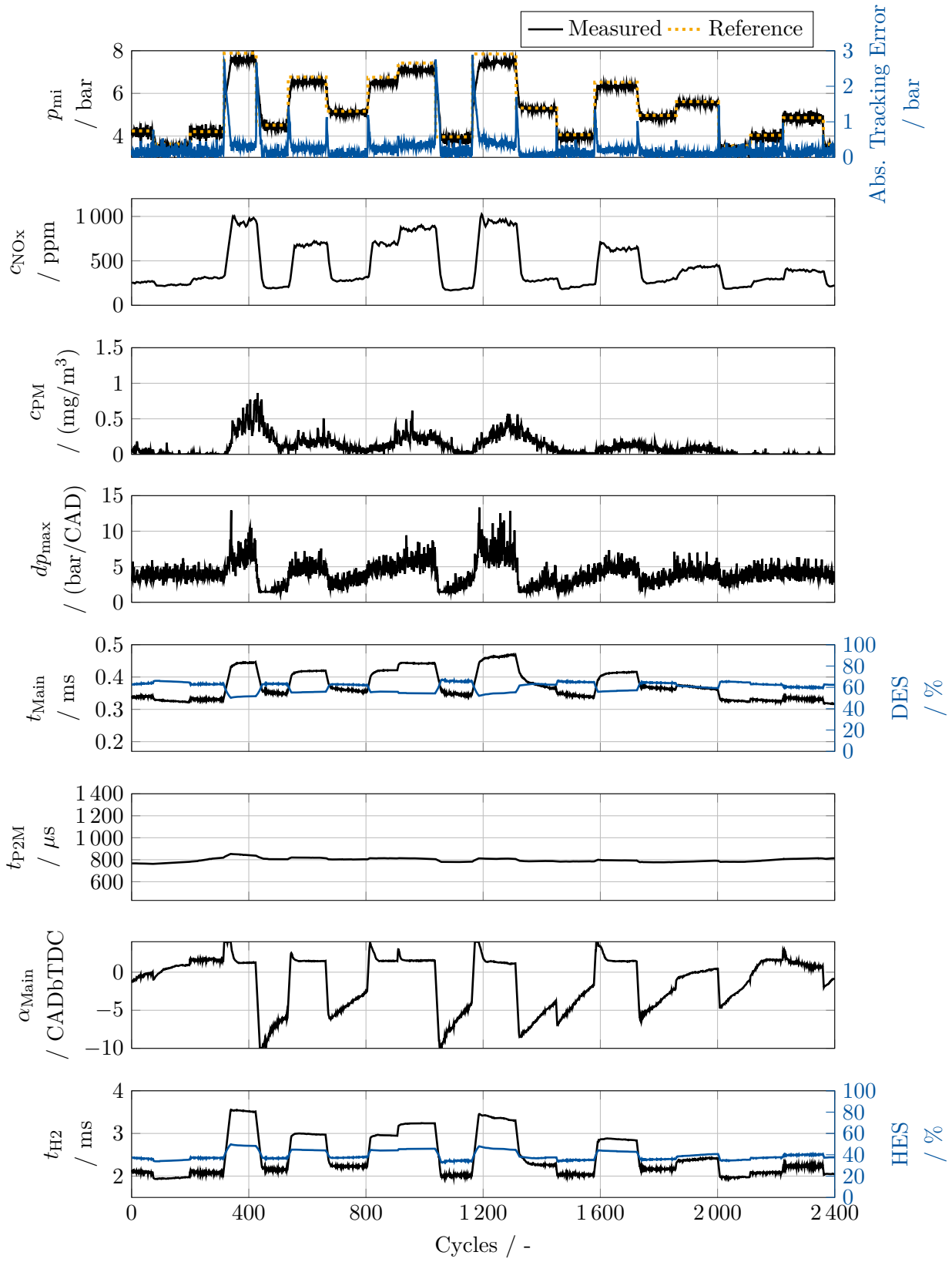


Figure 6.6: DNN-based BC controller. *Core* MPC benchmark experimental tracking results. Load reference trajectory with rapid steps, total length of 2400 cycles.

It is observable that the *Core MPC* is struggling to track the p_{mi} reference trajectory in higher loads while also producing minor oscillations in lower loads. These fluctuations are caused by incomplete opening of the hydrogen injector, a known issue when operating near its minimum activation threshold. The feedback-dependent nature of the MPC causes the control inputs to oscillate as it attempts to compensate for the inconsistent system response, which is reflected in the variation of the actuator signals. It can also be seen that the MPC reaches the engine load reference rather late after a jump, which is due to the conservative setting of the MPC following its constraints on the rate of change of the controls as explained in previous sections. Despite this, the MPC demonstrates robust tracking behavior with only minor deviations. It is important to note that these experiments were conducted approx. three months after the initial data used to train the MPC models was collected. This time gap underscores the challenges posed by model-plant mismatch and highlights the need for strategies such as adaptive control, observer-based correction, or online retraining of the deployed DNNs to maintain control accuracy over extended periods.

Subsequently, the BC controller is evaluated on the same 2400 cycle reference trajectory at $1500\,\mathrm{min}^{-1}$, and its tracking performance (Fig. 6.7) is compared to the *Core MPC*. Notably, the BC controller had not been trained with such fast transient profiles; its training data consisted exclusively of smoothed MPC trajectories as presented in Fig. 6.3.

Compared to the *Core MPC*, the BC controller demonstrates notably fast and accurate tracking of the p_{mi} reference trajectory, even responding to load jumps more quickly than the MPC expert policy itself. This behavior can be attributed to the absence of recurrent neural network (RNN) components, which reduces the network's ability to model temporal dependencies and instead leads to more reactive behavior. Despite its simpler DNN architecture, the controller effectively fulfills its imitation objective: it mimics the MPC policy with high fidelity, achieving comparable overall performance while maintaining high hydrogen energy share (HES). The BC policy successfully keeps emissions c_{NOx}, c_{PM} and dp_{max}, within acceptable limits, demonstrating its ability to maintain safe operation.

As in the *Core MPC* case, oscillations in p_{mi} occur at lower loads due to the injector operating near its lower activation limit, leading to missing hydrogen injections. This known system behavior also results in oscillatory control signals, as the feedback-driven controller attempts to counteract the missing actuator response. Additionally, a slight static offset is observed at high loads. It is dominated by a model–plant mismatch, caused mainly by accelerated wear of the hydrogen injector, which receives little lubrication from the pure hydrogen and therefore changed its characteristics after the training data had been recorded. Although the controller receives p_{mi} feedback, the cloned policy provides no offset-mitigation mechanism such as a disturbance observer, so this mismatch cannot be compensated and a static load offset remains. Due to the BC controller's imitating nature, this behavior was to be expected.

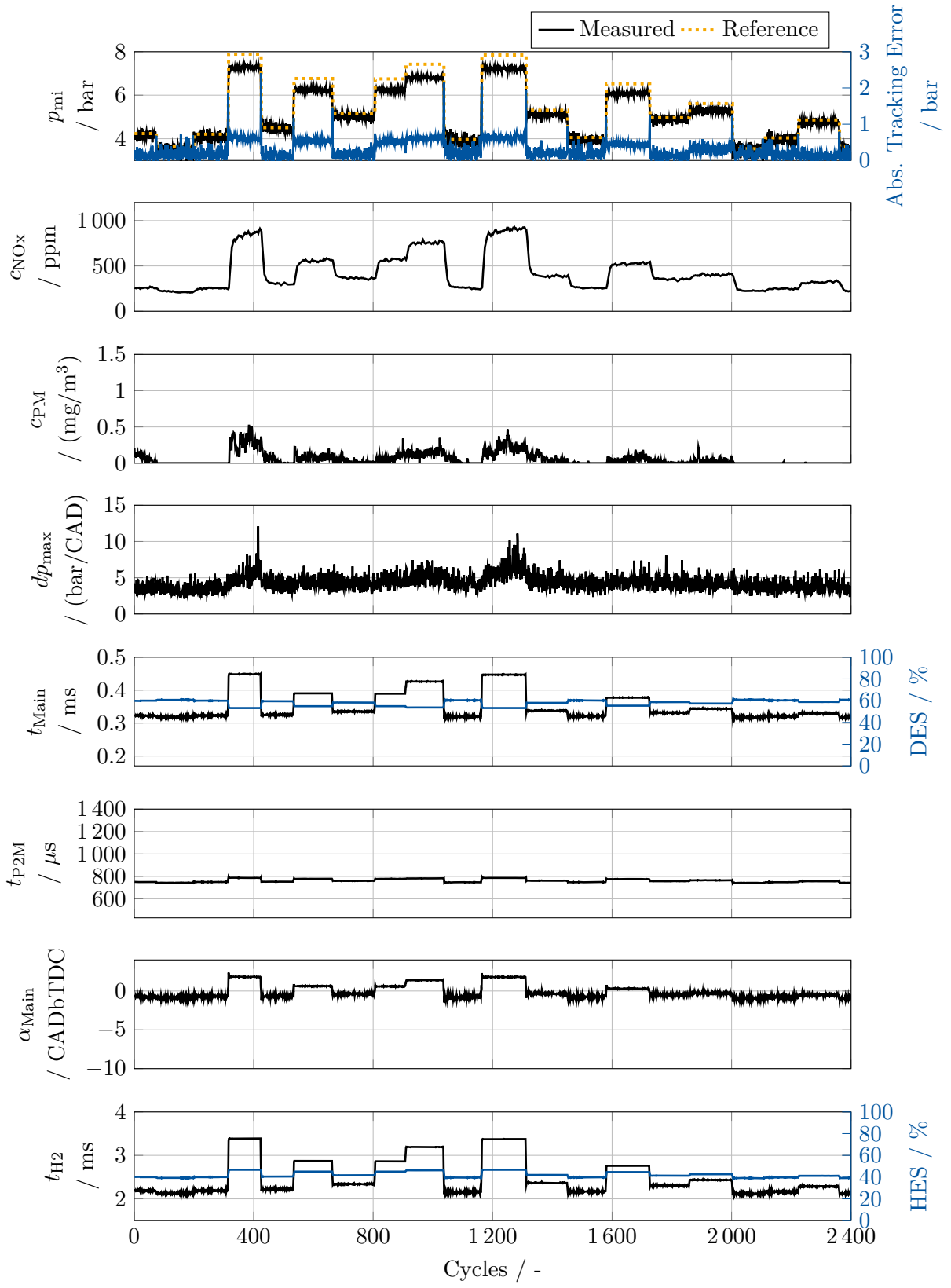


Figure 6.7: DNN-based BC controller. Experimental tracking results. Unseen load reference trajectory with rapid steps, total length of 2400 cycles.

Feedback-Free Policy Variant

Additionally, for similar investigation of the performance of the BC controller without sensor feedback, a second variant is introduced. The *no feedback* variant input and output domains for the BC controller are defined as the state space $\mathcal{S}_{\text{noFb}}$ and action space $\mathcal{A}_{\text{noFb}}$, with dimension $n_{s,\text{noFb}}$ and n_a, respectively:

$$\mathcal{S}_{\text{noFb}} \subseteq \mathbb{R}^{n_{s,\text{noFb}}=3}, \quad \text{where} \quad s(k) = p_{\text{mi,ref,i}}(k), \text{ with } i \in \{1,2,3\}, \tag{6.9}$$

$$\mathcal{A}_{\text{noFb}} = \mathcal{A} \subseteq \mathbb{R}^{n_{a,\text{noFb}}=n_a=4}. \tag{6.10}$$

The corresponding results for the training of the DNN of the *no feedback* variant are omitted here for brevity and depicted in Fig. A.20, Fig. A.21, Fig. A.22, Table A.15, respectively.

Fig. 6.8 illustrates the tracking behavior of the *no feedback* variant BC controller evaluated on the same dynamic load reference used previously. Compared to the standard BC controller, the absence of process feedback leads to slightly degraded tracking accuracy of the p_{mi} reference, most notably in the form of minor static offsets and increased oscillations. Despite this, the controller still produces smooth control commands. This behavior aligns with expectations, as the controller lacks temporal correction mechanisms typically enabled through RNN components. Emissions and dp_{max} stay within safe limits, demonstrating acceptable performance of the feedforward-only BC controller even under sensor faults or missing feedback.

Table 6.2 further quantifies the performance of both BC controller variants compared to the *Core MPC* in terms of performance metrics. Both BC variants achieved solid p_{mi} tracking, with NRMSE values of 8.97% (*feedback variant*) and 10.39% (*no feedback variant*). The *feedback variant* even reaches a slightly lower NRMSE than the MPC (9.21%), the *no feedback variant* 1.2 percentage points above it, whereas the MAE of the *feedback variant* is higher: the MPC's rate-limited controls settle slowly after the fast load steps, producing fewer small deviations but larger transient errors, which weigh more heavily in the RMSE. Despite this, the mean value of dp_{max} is notably higher for both BC controllers: up to +5.7% compared to the MPC for the *feedback variant*. High dp_{max} values are critical, as they impose mechanical stress and wear on engine components and are therefore explicitly constrained in the MPC formulation. The following section investigates the extrapolation capability and the behavior when approaching or violating physical constraints, with particular focus on dp_{max}.

Extrapolation Capability

Given the lack of formal safety guarantees in purely data-driven controllers, it is critical to assess how the BC policy behaves when faced with unseen, out-of-bounds inputs. Fig. 6.9 illustrates this extrapolation scenario, showing the *feedback variant* controller's response as $p_{\text{mi,ref}}$ is pushed to 9.5 bar, beyond the 8 bar boundaries seen during training.

Interestingly, the controller does not violate the implicitly learned bounds of its controls, the DNN's outputs. Instead, it attempts to maximize both diesel and hydrogen injection timings

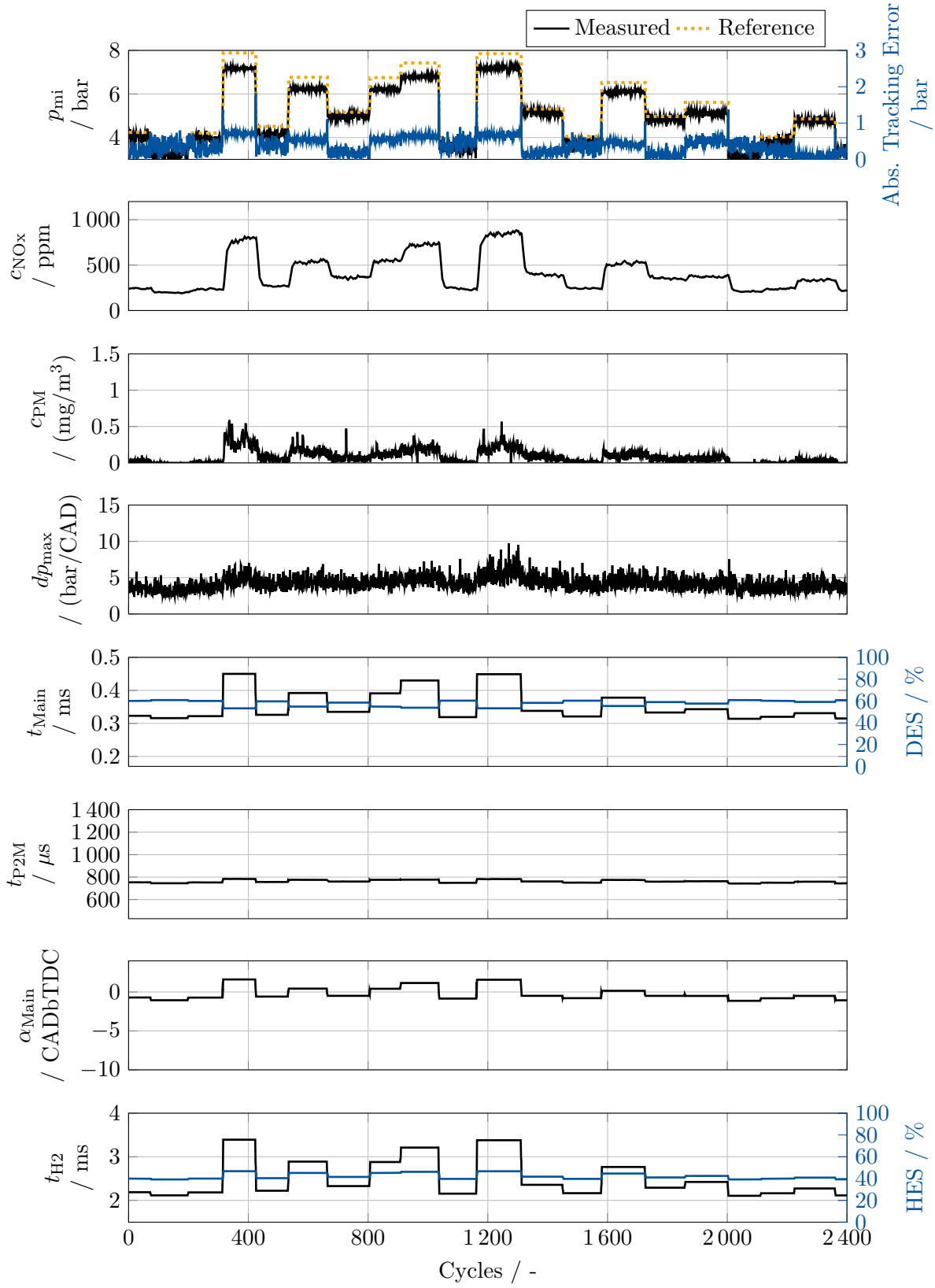


Figure 6.8: DNN-based BC controller *no feedback* variant. Experimental tracking results. Unseen load reference trajectory with rapid steps, total length of 2400 cycles.

Table 6.2: Comparison of experimental results for BC controllers (with and without feedback) and *Core MPC*. Unseen load reference trajectory with rapid steps, total length of 2400 cycles. The NRMSE is normalized with the range of the reference trajectory (4.35 bar, from 3.54 bar to 7.88 bar).

Metric	**BC** ***(feedback)***	**BC** ***(no feedback)***	**Vs.** ***core MPC***
p_{mi} tracking			
MAE / bar	0.311	0.391	fb: +28.7%, no fb: +61.7%
RMSE / bar	0.390	0.452	fb: -2.6%, no fb: +12.8%
NRMSE / −	8.97%	10.39%	fb: -2.6%, no fb: +12.8%
c_{NOx}			
max / ppm	927	884	fb: -9.5%, no fb: -13.7%
mean / ppm	424	407	fb: -3.6%, no fb: -7.5%
c_{PM}			
max / mg/m^3	0.52	0.59	fb: -40.0%, no fb: -31.7%
mean / mg/m^3	0.06	0.09	fb: -48.5%, no fb: -24.7%
dp_{max}			
max / bar/CAD	12.08	9.71	fb: -9.4%, no fb: -27.2%
mean / bar/CAD	4.33	4.24	fb: +5.7%, no fb: +3.5%
HES			
max / −	46.86%	46.78%	fb: -6.0%, no fb: -6.1%
mean / −	42.26%	42.18%	fb: +7.0%, no fb: +6.8%

to meet the requested load, while maintaining high HES. This behavior demonstrates stable performance of the learned policy, even in unseen operating regions. Nevertheless, the high fuel quantities and load level lead to elevated dp_{max} values. This indicates that the BC controller was not able to learn these constraints implicitly; a result that is expected, given the limited presence of such cases in the original training data from the MPC.

These results suggest that the BC approach offers promising extrapolation capabilities and potential for safe and reliable control, even when operating outside its trained distribution. Nevertheless, they also indicate that additional measures may be needed to ensure constraint satisfaction in safety-critical regions.

Embedded Performance

Finally, the embedded performance of the BC controller is evaluated. While the MPC required up to 7 ms per cycle, the BC controller consistently achieved inference times below 2 ms. Notably, this figure includes the latency introduced by the UDP communication, which alone accounts for approximately 1 ms of overhead. This marks a substantial improvement in computational efficiency. Compared to the MPC, the BC approach offers a roughly 250% speedup, enabling practical deployment on resource-constrained embedded platforms.

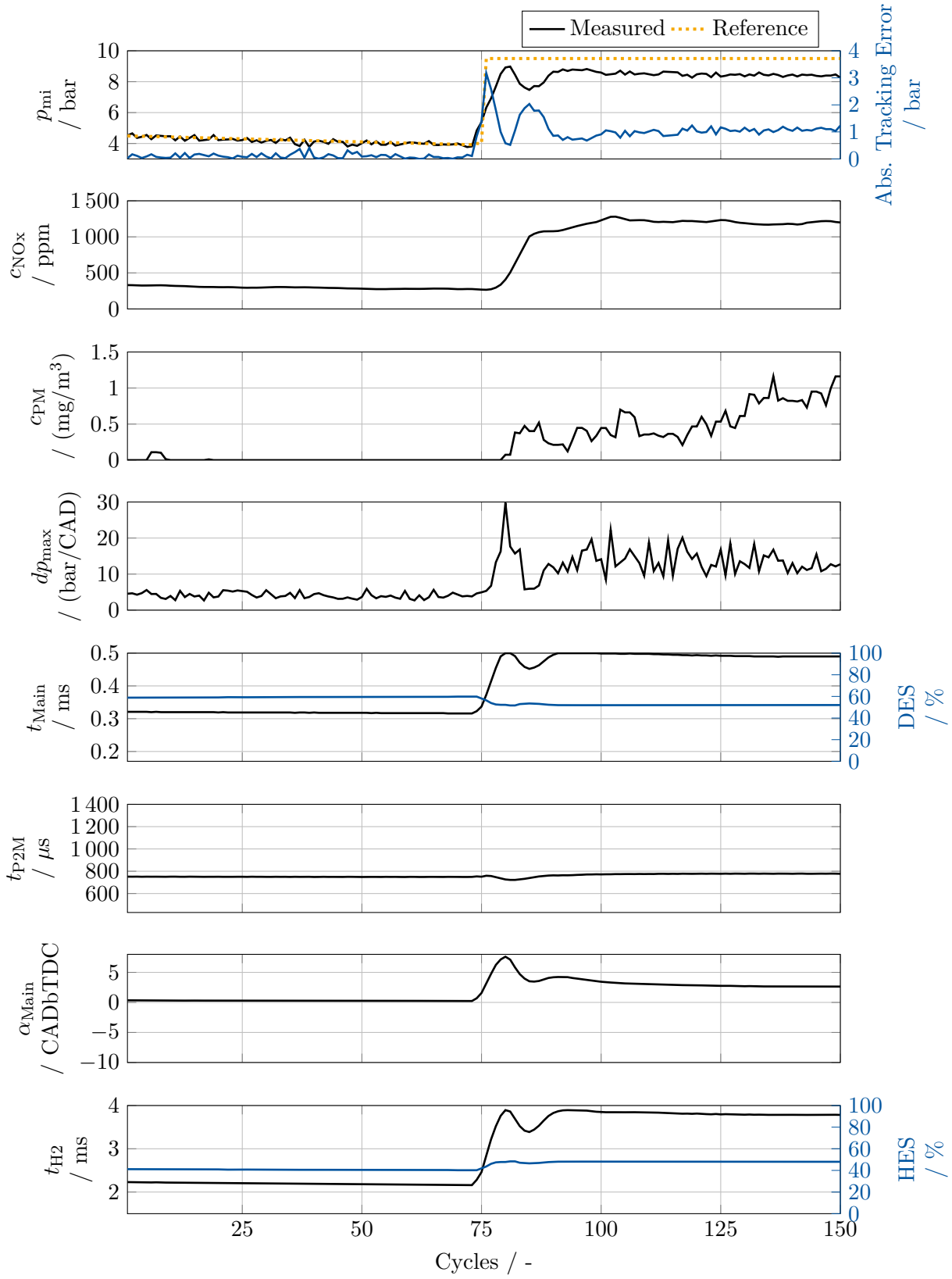


Figure 6.9: DNN-based BC controller. Extrapolation capability on out-of-bounds unseen input data, extract of experimental tracking results.

6.5 Embedded Runtime Benchmarking on Constrained Hardware

Following the successful experimental validation of the BC controllers on the engine test bench (achieving reliable control performance and real-time capability on the RPi) this section explores the next step towards practical deployment: run-time benchmarking on low-cost embedded hardware.

To this end, different DNN architectures and model sizes are evaluated on the well-known *ESP32* micro-controller. Integrated into an open-loop simplified simulative environment based on `MATLAB/Simulink` and its `Embedded Coder`, the net inference times are assessed. Specifications of the *ESP32* can be found in Table A.16 [168]. This device represents a cost-effective and widely available platform, often used in industrial and automotive prototyping. The goal is to assess the computational feasibility of running the trained BC controller on even more constrained embedded systems.

Fig. 6.10 presents the profiling results of different DNN architectures executed on a low-cost *ESP32* controller running at 180 MHz. The x-axis denotes the total number of learnable parameters per DNN, which serves as an approximate measure of the network size. As shown in Fig. A.23, this parameter exhibits a linear correlation of approx. factor 2 with the number of FLOPs previously introduced in Section 2.3 based on a dataset of over 27,000 DNN architectures generated. Consequently, the number of learnable parameters is used here as a practical metric for estimating computational load. The y-axis indicates the maximum execution time in ms observed during inference on the *ESP32*. Each data point represents the average of five independent profiling runs for a given DNN instance. All profiled models follow the encoder-decoder structure introduced earlier, and their computational complexity is varied by linearly scaling the number of neurons in each layer. Three network types were evaluated: feedforward neural networks (FFNNs, red), gated recurrent units (GRUs, green), and long short-term memory units (LSTMs, blue). As expected, FFNNs achieve the shortest execution times for a given number of parameters, followed by GRU and LSTM architectures. This performance ranking

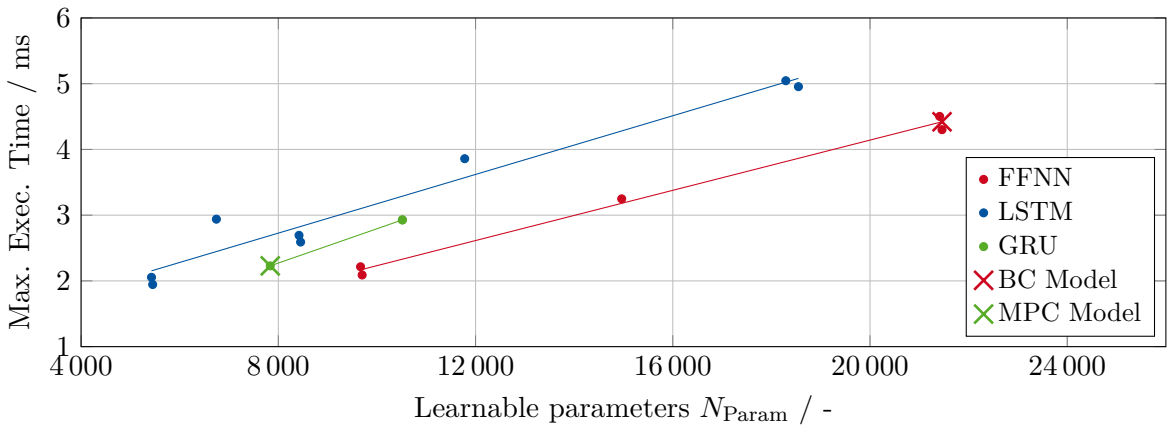


Figure 6.10: Profiling results (max. execution time) of different DNN architectures on low-cost embedded controller *ESP32* with a clock-rate of 180 MHz.

reflects their respective computational complexities: LSTMs, which rely on two memory states per unit, consistently require the most computation, whereas GRUs with a simplified structure of one memory state per unit are more efficient. FFNNs, lacking internal memory mechanisms, offer the lowest computational overhead, making them especially well-suited for real-time applications on resource-constrained embedded platforms. The two main DNNs developed in this thesis achieved computational times of 2.23 ms (15,164 FLOPs, 7900 learnables) for the MPC model and 4.42 ms (42,580 FLOPs, 21460 learnables) for the BC model. In Fig. 6.10, they are indicated by cross-markers. Note that this profiling covers network inference only, where the smaller MPC model is the faster one. In closed-loop operation, however, the MPC evaluates its network many times per engine cycle, whereas the BC controller requires a single forward pass.

All tested network variants remain well below the real-time threshold of $t_{\text{comp,max}} = 18\,\text{ms}$, ensuring real-time capability with a significant safety margin. With the largest networks achieving execution times on the *ESP32* of under 6 ms, this corresponds to a real-time margin factor of approximately 3×, demonstrating that the deployment of large DNN controllers on such resource-constrained embedded hardware is feasible.

Finally, this analysis closes the loop from advanced data-driven control design to embedded deployment, highlighting the real-world applicability of the BC approach for industrial use cases where resources are limited.

6.6 Summary of the Chapter & Discussion

This chapter demonstrated a methodical proof-of-concept for replacing a computationally expensive MPC with a lightweight, data-driven policy trained via BC. By leveraging supervised deep learning on expert-generated data, a DNN was trained to imitate the original controller and deployed on real-world engine hardware.

Two controller variants, one with and one without feedback, were evaluated against the MPC on dynamic unseen p_{mi} engine load trajectories. Both policies maintained safe and accurate load tracking, including compliance with emissions and dp_{max} constraints, while significantly reducing inference time. On the test bench, the cloned controller achieved a speedup of over 250% compared to the MPC, and embedded profiling confirmed real-time execution on low-cost hardware such as the *ESP32*, demonstrating the practical feasibility of deploying ML-based control on resource-constrained platforms.

However, despite the performance, several limitations must be acknowledged. As in the previous chapter, no formal robustness guarantees were provided, and the approach lacks deterministic analysis methods. This empirical and prototype-driven character, while suitable for early-stage development, currently could limit acceptance in safety-critical applications. Consequently, traditional solutions such as LUT or hand-crafted logic still dominate in regulated environments like automotive powertrain systems.

7 Conclusion

Summary

This thesis explored advanced data-driven control methodologies for a hydrogen diesel dual-fuel (H2DF) internal combustion engine (ICE), with a focus on control performance, real-time feasibility, and the safe, efficient operation of the engine to support CO_2 emission reduction.

The initial focus was on developing deep neural network (DNN) models capable of capturing the complex and nonlinear dynamics of H2DF combustion. Leveraging over 99,000 engine cycles of pseudo-random excitation data, a combination of feedforward neural networks (FFNNs) and gated recurrent unit (GRU) layers in an encoder-decoder structure provided accurate predictions for key engine metrics such as indicated mean effective pressure p_{mi}, NO_x emissions c_{NOx}, particulate matter c_{PM} emissions, and maximum cylinder pressure rise rate dp_{max}. The model showed particular strength in capturing system dynamics and output gradients – an essential requirement for control-oriented applications such as model predictive control (MPC). Automated machine learning (AutoML) techniques streamlined the multi-objective optimization of balancing prediction accuracy against computational demands, highlighting the potential of automated methods for these embedded control applications of DNN development.

Building upon this modeling foundation, this thesis presents a comprehensive nonlinear MPC framework that integrates the DNN models via the real-time optimal control software `acados`. The resulting MPC demonstrated robust performance, effectively balancing hydrogen utilization, emissions reduction and compliance, and engine safety under dynamic load conditions. Experimental validation confirmed real-time capability and adaptability across feedback and feedback-free scenarios, supported by outsourcing computational tasks to a *Raspberry Pi* for flexible, modular control deployment.

The proposed MPC outperformed the diesel baseline controller while enabling partial substitution of diesel fuel with hydrogen: p_{mi} tracking improves notably, achieving a mean absolute error of 0.117 bar (a reduction of 27.8%), and a normalized RMSE of 3.51%, corresponding to a 20.9% improvement. Soot emissions were significantly reduced, with the mean c_{PM} dropping by 61.1% to 0.21 $\mathrm{mg/m^3}$, while c_{NOx} increased by 105.1% to 483 ppm. All emission and dp_{max} constraints were satisfied by the MPC, ensuring safe and reliable operation of the H2DF engine. The hydrogen energy share (HES) reached up to 55.1% and averaged 39.7%, demonstrating substantial hydrogen substitution without compromising load-tracking performance. Assuming equivalent combustion efficiency, this enables a potential carbon dioxide emissions reduction of up to 55.1%.

Empirical robustness analyses demonstrate that the proposed MPC maintains stable and constraint-compliant operation when subjected to disturbances affecting both modeled dynamics (e.g., p_{mi} feedback) and unmodeled dynamics, such as model–plant mismatch induced by varying

engine speed, in both simulations and real-world experiments. In addition, multiple MPC variants were implemented and evaluated, ranging from configurations that maximize the HES to purely feedforward strategies, highlighting the flexibility and adaptability of the control framework.

The presented rapid control prototyping toolchain based on the integration of the *Raspberry Pi* (RPi) allows for the complete development cycle – from data acquisition to model training, controller formulation, and test bench deployment – to be executed in under one day. Compared to traditional control development approaches, which often require weeks of manual calibration and hardware integration, this represents a significant advancement in both speed and accessibility.

Finally, behavior cloning (BC) techniques were explored as a computationally efficient alternative to MPC. DNNs trained on expert trajectories generated by the MPC successfully replicated controller behavior on the test bench, achieving over 250% faster inference compared to the original MPC. Embedded benchmarking using an *ESP32* demonstrated that the cloned controller enables real-time deployment on low-cost, readily available hardware.

Consequently, the two research questions formulated in Chapter 1 were successfully addressed as follows:

1. **Advanced control systems can enable low-emission and safe operation of H2DF engines by effectively managing their complex and highly dynamic behavior under real-world conditions.** This thesis demonstrates that H2DF combustion offers a scalable near-term solution for significantly reducing CO_2 emissions. By retrofitting existing diesel engines with minimal hardware modifications and applying data-driven nonlinear control strategies, clean and safe engine operation is achieved. The proposed approach enables precise load tracking and constraint regulation, providing an immediate pathway toward cleaner mobility without relying on full electrification or new infrastructure.
2. **Furthermore, this thesis demonstrates the feasibility and necessity of accelerating control development.** The proposed data-driven methods enable significant reductions in development time and cost, which is particularly critical given the global shortage of skilled engineers. In addition, the ability to run advanced control algorithms on currently available low-cost embedded hardware enables immediate real-world deployment of the proposed technology, accelerating its impact and adoption.

The presented approach of integrating recurrent neural networks (RNNs) into MPC has already been successfully applied by the group to various processes, including diesel combustion [32], [35], homogeneous charge compression ignition (HCCI) [33], [118], and electric vehicle torque control of a permanent magnet synchronous motor [38]. Furthermore, the group has recently expanded the integration of RNNs to an advanced observation technique known as moving horizon estimation [36], which complements the full control loop by enabling robust state estimation.

These results highlight both the potential and the generalizability of the modular RNN-based concept for a broad range of applications in dynamic and nonlinear systems.

Discussion

While the presented methods demonstrate substantial potential, several important limitations must be acknowledged.

First, the experimental setup used in this thesis involved a relatively small engine with hydrogen port fuel injection (PFI) restricted to a single cylinder. While sufficient for initial prototyping and concept validation, this limits the generalizability of the findings to larger or fully hydrogen-fueled systems. Scaling the approach to multi-cylinder or full H2-ICE requires additional validation and investigation, which is already underway at the UofA.

Regarding the DNN-based modeling approach, its success relies on the availability of a robust and tolerant engine system for safe data generation using pseudo-random excitation. Such procedures are feasible with robust diesel-based setups but pose safety risks for more sensitive combustion processes, such as HCCI.

Both the MPC and BC controllers, though validated experimentally, share a critical limitation: the absence of formal guarantees for stability, robustness, or constraint satisfaction. No convex model structures or Lyapunov-based analysis were developed to back the observed empirical success. While this prototype-driven methodology accelerates development and enables early deployment, it risks sidelining classical control principles that remain crucial for certification and long-term reliability. As highlighted in the plenary talk by Borrelli at the *American Control Conference 2024* in Toronto, bridging empirical machine learning with theory-grounded control design is essential for sustainable progress in the future of control system theory and application [169].

In summary, while the demonstrated approaches significantly reduce development time and enable rapid prototyping, their broader adoption – particularly in safety-critical and regulated applications – requires integration of formal analysis and hybrid strategies that combine empirical learning with principle-based control design.

Outlook

To address these limitations and further advance the field, several avenues for future research and development emerge from this work. Suggested next steps include:

- Explore the validation in multi-cylinder H2DF setups and additionally extend the approach towards full hydrogen operation without diesel. These efforts are already underway at the UofA, where a 6.7L *Cummins* H2DF engine is planned to serve as a testbed for scaling the technology to larger, more representative engine platforms.
- The integration of reinforcement learning (RL) as a standalone control strategy replacing the MPC has already been successfully demonstrated during the course of this thesis, based on the foundational work by Sharma et al. [43], Sharma et al. [44], Sharma [119], and Bedei et al. [170].
- In parallel, RL is being integrated to augment MPC with self-adaptive capabilities, enabling improved adaptability under changing conditions. This advancement is being conducted by Bedei et al. [45].
- Developing multi-level MPC architectures with separate, cascaded networks managing slower emission targets distinct from torque control.
- Exploring online adaptive cloud-based models to dynamically update and refine the controller models.
- Establishing formal guarantees for the learned controllers by exploring Lyapunov-based stability analysis or convex model reformulations. [171], [172]
- Investigating hybrid strategies combining classical control theory and data-driven methodologies, such as neural ordinary differential equations [173] and deep Koopman operator theory [174], to bridge empirical and principle-based control approaches.
- Utilizing data-driven methods to enhance conventional control architectures, such as calibrating look-up-tables or proportional-integral controllers through RL or supervised learning as seen by Koch et al. [175] and Garg et al. [176].

Ultimately, the critical assumption underlying the thesis – the availability of genuinely sustainable hydrogen – remains pivotal. Achieving the envisioned green and net-zero carbon scenario by 2050 (see Chapter 1) necessitates not only technical innovations but significant infrastructural and industrial changes. Solutions presented in this thesis represent meaningful contributions towards this overarching goal, highlighting the role of advanced control and ML techniques in driving automotive systems and drivetrains towards sustainability.

References

[1] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017, ISBN: 9781107016880. [Online]. Available: https://books.google.de/books?id=Kyc8vgAACAAJ.

[2] Environment and C. C. Canada, *Greenhouse gas emissions*, https://www.canada.ca/en/environment-climate-change/services/environmental-indicators/greenhouse-gas-emissions.html, Accessed: 2024-05-14, 2021.

[3] I. P. on Climate Change (IPCC), *Climate change 2022: Mitigation of climate change. contribution of working group iii to the sixth assessment report of the intergovernmental panel on climate change*, P. R. S. et al., Ed., 2022. DOI: 10.1017/9781009157926. [Online]. Available: https://www.ipcc.ch/report/ar6/wg3/.

[4] I. E. Agency, *CO_2 emissions in 2022*, https://www.iea.org/reports/co2-emissions-in-2022, Accessed: 2024-05-14, 2023.

[5] I. T. Forum, *ITF Transport Outlook 2023*. OECD Publishing, 2023. DOI: 10.1787/b6cc9ad5-en. [Online]. Available: https://doi.org/10.1787/b6cc9ad5-en.

[6] H. Quak, N. Nesterova, and T. Van Rooijen, "Possibilities and Barriers for Using Electric-powered Vehicles in City Logistics Practice," en, *Transportation Research Procedia*, vol. 12, pp. 157–169, 2016, ISSN: 23521465. DOI: 10.1016/j.trpro.2016.02.055. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2352146516000569.

[7] A.-M. Anderson and R. J. Kish, "The costs and benefits of electric trucks: A synopsis of the u.s. trucking market," *Sustainability*, vol. 17, no. 5, p. 2097, 2025. DOI: 10.3390/su17052097.

[8] "World energy transitions outlook 2024: 1.5°C pathway," International Renewable Energy Agency (IRENA), Abu Dhabi, Tech. Rep., Nov. 2024. [Online]. Available: https://www.irena.org/Publications/2024/Nov/World-Energy-Transitions-Outlook-2024.

[9] McKinsey & Company, "Global Energy Perspective 2023: Hydrogen Outlook," Tech. Rep., Jan. 2024, Accessed: 2025-05-09. [Online]. Available: https://www.mckinsey.com/industries/oil-and-gas/our-insights/global-energy-perspective-2023-hydrogen-outlook.

[10] J. M. Marín Arcos and D. M. F. Santos, "The hydrogen color spectrum: Techno-economic analysis of the available technologies for hydrogen production," *Gases*, vol. 3, no. 1, pp. 25–46, 2023. DOI: 10.3390/gases3010002. [Online]. Available: https://doi.org/10.3390/gases3010002.

[11] J. Lof et al., *The future of freight b: Assessing zero emission diesel fuel alternatives for freight transportation in canada*, 2019. [Online]. Available: http://www.cesarnet.ca/publications/cesar-scenarios.

[12] Q. Hassan, I. D. J. Azzawi, A. Z. Sameen, and H. M. Salman, "Hydrogen fuel cell vehicles: Opportunities and challenges," *Sustainability*, vol. 15, no. 15, p. 11 501, 2023. DOI: 10.3390/su151511501.

[13] T. Tsujimura and Y. Suzuki, "The utilization of hydrogen in hydrogen/diesel dual fuel engine," *International Journal of Hydrogen Energy*, vol. 42, no. 19, pp. 14 019–14 029, 2017, Special Issue on The 21st World Hydrogen Energy Conference (WHEC 2016), 13-16 June 2016, Zaragoza, Spain, ISSN: 0360-3199. DOI: https://doi.org/10.1016/j.ijhydene.2017.01.152. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S036031991730318X.

[14] K. Wróbel, J. Wróbel, W. Tokarz, J. Lach, K. Podsadni, and A. Czerwiński, "Hydrogen internal combustion engine vehicles: A review," *Energies*, vol. 15, no. 23, p. 8937, 2022. DOI: 10.3390/en15238937.

[15] D. B. Layzell, D. Millington, and R. Lee, "Towards a fuel hydrogen economy in the calgary region: A feasibility study," The Transition Accelerator, Tech. Rep. 1, Feb. 2023, pp. 1–64. [Online]. Available: https://transitionaccelerator.ca/reports/towards-a-fuel-hydrogen-economy-in-the-calgary-region/.

[16] S. Verhelst and T. Wallner, "Hydrogen-fueled internal combustion engines," *Progress in Energy and Combustion Science*, vol. 35, no. 6, pp. 490–527, 2009, ISSN: 0360-1285. DOI: /10.1016/j.pecs.2009.08.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360128509000422.

[17] M. Guan et al., "Characterizing hydrogen-diesel dual-fuel performance and emissions in a commercial heavy-duty diesel truck," *International Journal of Hydrogen Energy*, vol. 86, pp. 1085–1096, 2024. DOI: 10.1016/j.ijhydene.2024.08.480.

[18] F. Scrignoli, A. M. Pisapia, T. Savioli, E. Mancaruso, E. Mattarelli, and C. A. Rinaldini, "Exploring hydrogen–diesel dual fuel combustion in a light-duty engine: A numerical investigation," *Energies*, vol. 17, no. 22, p. 5761, 2024. DOI: 10.3390/en17225761.

[19] M. El Hannach, L. Guzman, P. Ahmadi, S. Pickup, and E. Kjeang, "Life cycle assessment of hydrogen and diesel dual-fuel class 8 heavy duty trucks," *International Journal of Hydrogen Energy*, vol. 44, no. 16, pp. 8575–8584, 2019. DOI: 10.1016/j.ijhydene.2019.02.027.

[20] H. on Demand Technologies, *Diesel-hydrogen dual fuel: A path to cleaner, more efficient engines*, https://hodtec.com/blog/diesel-hydrogen-dual-fuel-a-path-to-cleaner-more-efficient-engines/, 2024.

[21] Caterpillar Inc., *Cat Mining Trucks Built to Rebuild – Together, for the Long Haul*, https://www.caterpillar.com/en/news/caterpillarNews/2021/together-for-the-long-haul.html, 2021.

[22] C. R. Koch, J. T. McNally, and et al., "Blue hydrogen–diesel dual-fuel engine technology development for freight decarbonization (project summary)," in *Univ. of Alberta Hydrogen Initiative Final Report*, Available via Alberta Innovates (Project No. CCUS000002), 2023. [Online]. Available: https://albertainnovates.ca/projects/blue-hydrogen-diesel-dual-fuel-engine-technology-development/.

[23] Y. Suzuki, T. Tsujimura, and T. Mita, "The performance of multi-cylinder hydrogen/diesel dual fuel engine," *SAE Int. J. Engines*, vol. 8, no. 5, pp. 2240–2252, 2015. DOI: 10.4271/2015-24-2458.

[24] S.-R. Jhang, K.-S. Chen, S.-L. Lin, Y.-C. Lin, and W.-L. Cheng, "Reducing pollutant emissions from a heavy-duty diesel engine by using hydrogen additions," *Fuel*, vol. 172, pp. 89–95, 2016. DOI: 10.1016/j.fuel.2016.01.032.

[25] P. Dimitriou, M. Kumar, T. Tsujimura, and Y. Suzuki, "Combustion and emission characteristics of a hydrogen-diesel dual-fuel engine," *International Journal of Hydrogen Energy*, vol. 43, no. 28, pp. 13 605–13 617, 2018. DOI: 10.1016/j.ijhydene.2018.05.062.

[26] M. Wang, A. Matsugi, Y. Kondo, Y. Sakamoto, and Y. Kajii, "Impact of hydrogen mixture on fuel consumption and exhaust gas emissions in a truck with direct-injection diesel engine," *Energies*, vol. 16, no. 11, p. 4466, 2023. DOI: 10.3390/en16114466.

[27] X. Liu, S. Liu, L. Shen, Y. Bi, and L. Duan, "Study on the effects of the hydrogen substitution rate on the performance of a hydrogen–diesel dual-fuel engine under different loads," *Energies*, vol. 16, no. 16, p. 5971, 2023. DOI: 10.3390/en16165971.

[28] A. Norouzi, H. Heidarifar, M. Shahbakhti, C. R. Koch, and H. Borhan, "Model predictive control of internal combustion engines: A review and future directions," *Energies*, vol. 14, no. 19, 2021, ISSN: 1996-1073. DOI: 10.3390/en14196251.

[29] T. Albin Rajasingham, *Nonlinear Model Predictive Control of Combustion Engines: From Fundamentals to Applications* (Advances in Industrial Control). Cham: Springer International Publishing, 2021. DOI: 10.1007/978-3-030-68010-7.

[30] R. Verschueren et al., "Acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021, ISSN: 1867-2957. DOI: 10.1007/s12532-021-00208-8.

[31] AutoML.org, *Automl.org – automatic machine learning*, Accessed: 31 October 2025, 2025. [Online]. Available: https://www.automl.org/.

[32] A. Norouzi et al., "Deep learning based model predictive control for compression ignition engines," *Control Engineering Practice*, vol. 127, p. 105 299, 2022, ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2022.105299. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066122001472.

[33] D. C. Gordon et al., "Introducing a deep neural network-based model predictive control framework for rapid controller implementation," in *2024 American Control Conference (ACC)*, 2024, pp. 5232–5237. DOI: 10.23919/ACC60939.2024.10644830.

[34] A. Norouzi et al., "Machine learning integrated with model predictive control for imitative optimal control of compression ignition engines," *IFAC-PapersOnLine*, vol. 55, no. 24, pp. 19–26, 2022, 10th IFAC Symposium on Advances in Automotive Control AAC 2022, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2022.10.256. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322022881.

[35] D. C. Gordon et al., "End-to-end deep neural network based nonlinear model predictive control: Experimental implementation on diesel engine emission control," *Energies*, vol. 15, no. 24, 2022, ISSN: 1996-1073. DOI: 10.3390/en15249335. [Online]. Available: https://www.mdpi.com/1996-1073/15/24/9335.

[36] A. Winkler, P. Shah, K. Baumgärtner, V. Sharma, D. Gordon, and J. Andert, "Incorporating a deep neural network into moving horizon estimation for embedded thermal torque derating of an electric machine," *Energies*, vol. 18, no. 14, 2025, ISSN: 1996-1073. DOI: 10.3390/en18143813. [Online]. Available: https://www.mdpi.com/1996-1073/18/14/3813.

[37] A. Winkler et al., "Embedded real-time nonlinear model predictive control for the thermal torque derating of an electric vehicle," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 359–364, 2021, 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.08.570. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896321013458.

[38] A. Winkler, W. Wang, A. Norouzi, D. Gordon, C. Koch, and J. Andert, "Integrating recurrent neural networks into model predictive control for thermal torque derating of electric machines," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 8254–8259, 2023, 22nd IFAC World Congress, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2023.10.1010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896323013939.

[39] A. Norouzi et al., "Deep learning and nonlinear model predictive control integration for compression ignition engine emission reduction," in *Proceedings of the Canadian Society for Mechanical Engineering International Congress*, Edmonton, Alberta, 2022. DOI: 10.7939/R3-MJW8-AF75. [Online]. Available: https://ualberta.scholaris.ca/server/api/core/bitstreams/bb75b8d4-7bc9-4dde-a9c4-f9a4ff744f26/content.

[40] D. Gordon, E. Sperling, A. Norouzi, A. Winkler, J. Andert, and C. R. Koch, "Mpc implementation for hcci combustion using a deep neural network based model," in *Proceedings of the Combustion Institute - Canadian Section Spring Technical Meeting*, Edmonton, Alberta, Canada, 2023.

[41] D. Gordon, E. Sperling, A. Norouzi, A. Winkler, J. Andert, and C. R. Koch, "Deep neural network based performance and emission modeling of an hcci engine," in *Proceedings of the Combustion Institute - Canadian Section Spring Technical Meeting*, Edmonton, Alberta, Canada, 2023.

[42] H. Mehnatkesh et al., "Systematic framework for deep learning-based predictive injection control with bayesian hyperparameter optimization for a hydrogen/diesel dual-fuel engine," *Control Engineering Practice*, vol. 164, p. 106 513, 2025, ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2025.106513. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066125002758.

[43] V. Sharma, A. Winkler, A. Norouzi, H. Guo, J. Andert, and D. Gordon, "Safe reinforcement learning-based control for hydrogen diesel dual-fuel engines," *IFAC-PapersOnLine*, vol. 59, no. 5, pp. 19–24, 2025, 11th IFAC Symposium on Advances in Automotive Control AAC 2025, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2025.07.075. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896325004288.

[44] V. Sharma, A. Winkler, A. Norouzi, H. Guo, J. Andert, and D. Gordon, "Reinforcement learning for hydrogen dual-fuel engines," 4. Aachen Hydrogen Colloquium, Aachen (Germany), 6 May 2025 - 7 May 2025, 2025. [Online]. Available: https://publications.rwth-aachen.de/record/1013437.

[45] J. Bedei, M. McBain, A. Winkler, C. R. Koch, J. Andert, and D. Gordon, "Hybrid reinforcement learning and model predictive control for adaptive control of hydrogen-diesel dual-fuel combustion," in *Proceedings of the Combustion Institute – Canadian Section Spring Technical Meeting*, Calgary, Alberta, Canada, 2025. DOI: 10.48550/arXiv.2504.16875. arXiv: 2504.16875 [cs.LG]. [Online]. Available: https://www.combustion-institute.ca/2025-meeting/.

[46] A. E. K. Winkler, L. Li, A. Kampmeier, T. Brinkmann, L. Koch, and J. L. Andert, "Automated Machine Learning (AutoML) for Data-Driven Modeling: A Study on Hyperparameter Optimization for Time-Series Prediction of Battery Cell Temperature," in *Proceedings of the 34th Aachen Colloquium Sustainable Mobility*, Poster presentation at the 34th Aachen Colloquium Sustainable Mobility, 6–8 October 2025. RWTH-2025-08344, Aachen, Germany, Oct. 2025. [Online]. Available: https://publications.rwth-aachen.de/record/1019338.

[47] A. Winkler et al., *Hydrogen–diesel dual-fuel engine operation using real-time GRU-based nonlinear model predictive control*, Preprint, submitted to International Journal of Hydrogen Energy, Sep. 2026. DOI: 10.48550/arXiv.2609.15819. arXiv: 2609.15819 [eess.SY]. Accessed: Sep. 15, 2026. [Online]. Available: https://arxiv.org/abs/2609.15819.

[48] A. Winkler, N. Naduvath Mana, D. Gordon, and J. Andert, *Deep behaviour cloning of model predictive control for real-time operation of a hydrogen–diesel dual-fuel engine*, Preprint, submitted to Control Engineering Practice, 2026. DOI: 10.48550/arXiv.2609.33795. arXiv: 2609.33795 [eess.SY]. [Online]. Available: https://doi.org/10.48550/arXiv.2609.33795.

[49] A. Winkler, *Deep neural network (dnn) training scripts for h2df engine models (alexwin9478/h2dfdnntrainexpdata: Zenodo)*, version v1.0, Aug. 2025. DOI: 10.5281/zenodo.16902580. [Online]. Available: https://doi.org/10.5281/zenodo.16902580.

[50] A. Winkler, *Deep mpc scripts using acados incl. closed-loop simulation model (alexwin9478/h2dfdeepmpc: Zenodo)*, version v1.0, Aug. 2025. DOI: 10.5281/zenodo.16902940. [Online]. Available: https://doi.org/10.5281/zenodo.16902940.

[51] A. Winkler, *Automl scripts for lstm-based dnns with optuna on hpc (alexwin9478/automlfordnntraining: Zenodo)*, version v1.0, Aug. 2025. DOI: 10.5281/zenodo.16903153. [Online]. Available: https://doi.org/10.5281/zenodo.16903153.

[52] A. Winkler, *Deep learning and behavior cloning for h2df engine control (alexwin9478/h2df-behaviorcloning: Zenodo)*, version v1.0, Aug. 2025. DOI: 10.5281/zenodo.16902948. [Online]. Available: https://doi.org/10.5281/zenodo.16902948.

[53] A. S. Mehr, A. D. Phillips, M. P. Brandon, M. T. Pryce, and J. G. Carton, "Recent challenges and development of technical and technoeconomic aspects for hydrogen storage, insights at different scales; a state of art review," *International Journal of Hydrogen Energy*, vol. 70, pp. 786–815, 2024, ISSN: 0360-3199. DOI: https://doi.org/10.1016/j.ijhydene.2024.05.182. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S036031992401869X.

[54] C. White, R. Steeper, and A. Lutz, "The hydrogen-fueled internal combustion engine: A technical review," *International Journal of Hydrogen Energy*, vol. 31, no. 10, pp. 1292–1305, 2006, ISSN: 0360-3199. DOI: https://doi.org/10.1016/j.ijhydene.2005.12.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360319905003721.

[55] J. B. Heywood, *Internal Combustion Engine Fundamentals*. McGraw-Hill Education, 1988.

[56] T. V. Johnson, "Review of diesel emissions and control," *International Journal of Engine Research*, vol. 10, no. 5, pp. 275–285, 2009. DOI: 10.1243/14680874JER04009. eprint: https://doi.org/10.1243/14680874JER04009. [Online]. Available: https://doi.org/10.1243/14680874JER04009.

[57] R. D. Reitz, "Directions in internal combustion engine research," *Combustion and Flame*, vol. 160, no. 1, pp. 1–8, 2013, ISSN: 0010-2180. DOI: https://doi.org/10.1016/j.combustflame.2012.11.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S001021801200315X.

[58] N. B. L. Thermodynamics Research Center, "Nist chemistry webbook: Thermophysical properties of fluid systems, nist standard reference database number 69," National Institute of Standards and Technology, U.S. Department of Commerce, Tech. Rep., 2023. DOI: https://doi.org/10.18434/T4D303.

[59] S. C. Davis and R. G. Boundy, "Transportation energy data book: Edition 39," Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States), Tech. Rep., Feb. 2021. DOI: 10.2172/1767864. [Online]. Available: https://www.osti.gov/biblio/1767864.

[60] K. Naseem et al., "Essential parts of hydrogen economy: Hydrogen production, storage, transportation and application," *Renewable and Sustainable Energy Reviews*, vol. 210, p. 115 196, 2025, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2024.115196. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032124009225.

[61] T. Stepan, T. Breitender, and J. Winklhofer, "Comparison of liquefied gas energy carriers and conventional fossil fuels with a focus on storage requirements," SAG Innovation GmbH, Tech. Rep., 2023. [Online]. Available: https://www.sag.at/wp-content/uploads/2023/05/sag_whitepaper_comparison_engl.pdf.

[62] A. Magliano, C. Perez Carrera, C. M. Pappalardo, D. Guida, and V. P. Berardi, "A comprehensive literature review on hydrogen tanks: Storage, safety, and structural integrity," *Applied Sciences*, vol. 14, no. 20, 2024, ISSN: 2076-3417. DOI: 10.3390/app14209348. [Online]. Available: https://www.mdpi.com/2076-3417/14/20/9348.

[63] M. M. H. Bhuiyan and Z. Siddique, "Hydrogen as an alternative fuel: A comprehensive review of challenges and opportunities in production, storage, and transportation," *International Journal of Hydrogen Energy*, vol. 102, pp. 1026–1044, 2025. DOI: 10.1016/j.ijhydene.2025.01.033.

[64] Ludwig-Bölkow-Systemtechnik GmbH, *Milestone reached: Over 1,000 hydrogen refuelling stations in operation worldwide in 2024*, https://www.h2stations.org/press-release-2025-milestone-reached-over-1000-hydrogen-refuelling-stations-in-op-eration-worldwide-in-2024/, Press release, 11 February 2025. Around 1,160 hydrogen refuelling stations worldwide at the end of 2024, 2025.

[65] T. Jamal et al., "Fuelling the future: An in-depth review of recent trends, challenges and opportunities of hydrogen fuel cell for a sustainable hydrogen economy," *Energy Reports*, vol. 10, pp. 2103–2127, 2023, ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2023.09.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484723012477.

[66] S. E. Hosseini and B. Butler, "An overview of development and challenges in hydrogen powered vehicles," *International Journal of Green Energy*, Oct. 2019. DOI: 10.1080/15435075.2019.1685999.

[67] R. Diesel, "Method of and apparatus for converting heat into work," pat. US542846A, Filed August 26, 1892; granted July 16, 1895, Jul. 1895. [Online]. Available: https://patents.google.com/patent/US542846A.

[68] C. Cunanan, M.-K. Tran, Y. Lee, S. Kwok, V. Leung, and M. Fowler, "A review of heavy-duty vehicle powertrain technologies: Diesel engine vehicles, battery electric vehicles, and hydrogen fuel cell electric vehicles," *Clean Technologies*, vol. 3, no. 2, pp. 474–489, 2021, ISSN: 2571-8797. DOI: 10.3390/cleantechnol3020028. [Online]. Available: https://www.mdpi.com/2571-8797/3/2/28.

[69] K. Reif, Ed., *Diesel Engine Management: Systems and Components*. Springer Vieweg, 2014, ISBN: 978-3-658-03981-3.

[70] F. Atzler, O. Kastner, R. Rotondi, and A. Weigand, "Multiple injection and rate shaping part 1: Emissions reduction in passenger car diesel engines," in *SAE 2009 International Conference on Engines and Vehicles*, ser. SAE Technical Paper, Consiglio Nazionale delle Ricerche, Sep. 2009. DOI: 10.4271/2009-24-0004.

[71] R. Stone, *Introduction to Internal Combustion Engines*, 4th. Basingstoke, UK: Palgrave Macmillan, 2012, ISBN: 978-0230576636.

[72] N Saravanan and G Nagarajan, "An experimental investigation of hydrogen-enriched air induction in a diesel engine system," *International Journal of Hydrogen Energy*, vol. 33, no. 6, pp. 1769–1775, 2008. DOI: 10.1016/j.ijhydene.2007.12.065. [Online]. Available: https://doi.org/10.1016/j.ijhydene.2007.12.065.

[73] T. Gatts, S. Liu, C. Liew, B. Ralston, C. Bell, and H. Li, "An experimental investigation of incomplete combustion of gaseous fuels of a heavy-duty diesel engine supplemented with hydrogen and natural gas," *International Journal of Hydrogen Energy*, vol. 37, no. 9, pp. 7848–7859, 2012, 7th Petite Workshop on the Defect Chemical Nature of Energy Materials, 14-17 March 2011, Storaas, Kongsberg, Norway, ISSN: 0360-3199. DOI: https://doi.org/10.1016/j.ijhydene.2012.01.088. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S036031991200184X.

[74] E. W. Sperling, "Unburned hydrogen emissions of a hydrogen-diesel dual fuel engine," M.S. thesis, University of Alberta, Edmonton, Alberta, Canada, 2025. [Online]. Available: https://sites.ualberta.ca/~ckoch/thesis/sperling_thesis_final.pdf.

[75] R. Farzam et al., "Performance and emissions characteristics of hydrogen-diesel heavy-duty engines: The influence of engine control parameters," *International Journal of Hydrogen Energy*, vol. 109, pp. 1325–1340, 2025, ISSN: 0360-3199. DOI: https://doi.org/10.1016/j.ijhydene.2025.02.135. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360319925007001.

[76] D. Hauglustaine, F. Paulot, W. Collins, R. Derwent, M. Sand, and O. Boucher, "Climate benefit of a future hydrogen economy," *Communications Earth & Environment*, vol. 3, no. 1, p. 295, 2022. DOI: 10.1038/s43247-022-00626-z. [Online]. Available: https://doi.org/10.1038/s43247-022-00626-z.

[77] R. Dorf and R. Bishop, *Modern Control Systems Global Edition*. Pearson Deutschland, 2017, p. 1136, ISBN: 9781292152974. [Online]. Available: https://elibrary.pearson.de/book/99.150005/9781292152981.

[78] R. T. Stefani, *Design of Feedback Control Systems*. Oxford University Press, 2002.

[79] S. Zacher, "Basics of system dynamics and control theory," in *Closed Loop Control and Management: Introduction to Feedback Control Theory with Data Stream Managers*. Cham: Springer International Publishing, 2022, pp. 43–80, ISBN: 978-3-031-13483-8. DOI: 10.1007/978-3-031-13483-8_2.

[80] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, 2017, vol. 2, ISBN: 9780975937730. [Online]. Available: https://books.google.de/books?id=MrJctAEACAAJ.

[81] D. Abel, *Regelungstechnik. Institut für Regelungstechnik (RWTH Aachen University)*, 2022, Vorlesungsskript.

[82] S. Gros and M. Diehl, "Numerical optimal control," Draft textbook (manuscript), KU Leuven, 2024. [Online]. Available: https://www.syscop.de/files/2020ss/NOC/book-NOCSE.pdf.

[83] J. Nocedal and S. Wright, *Numerical optimization* (Springer series in operations research and financial engineering), 2. ed. New York, NY: Springer, 2006, XXII, 664, ISBN: 978-0-387-30303-1. DOI: 10.1007/978-0-387-40065-5.

[84] M. Diehl, H. Bock, and J. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control and Optimization*, vol. 43, pp. 1714–1736, Jan. 2005. DOI: 10.1137/S0363012902400713.

[85] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738. [Online]. Available: https://link.springer.com/book/9780387310732.

[86] E. Alpaydin, *Introduction to Machine Learning*. The MIT Press, 2014, ISBN: 0262028182, 9780262028189. [Online]. Available: https://mitpress.mit.edu/9780262043793/introduction-to-machine-learning/.

[87] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd. O'Reilly Media, Inc., 2019, ISBN: 1492032646. [Online]. Available: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

[88] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016, http://www.deeplearningbook.org, ISBN: 0262035618.

[89] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, pp. 386 –408, 1958. DOI: 10.1037/h0042519.

[90] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982. DOI: 10.1073/pnas.79.8.2554. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.79.8.2554.

[91] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[92] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, pp. 2451–2471, Oct. 2000. DOI: 10.1162/089976600300015015.

[93] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. [Online]. Available: https://aclanthology.org/D14-1179/.

[94] K. Zarzycki and M. Ławryńczuk, "LSTM and GRU neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors," *Sensors*, vol. 21, no. 16, p. 5625, 2021. DOI: 10.3390/s21165625.

[95] S. Yang, X. Yu, and Y. Zhou, "Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example," in *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, 2020, pp. 98–101. DOI: 10.1109/IWECAI50956.2020.00027.

[96] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. DOI: 10.48550/arXiv.1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980.

[97] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006, ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207006000239.

[98] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 437–478, ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_26. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_26.

[99] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, ser. NIPS'88, Cambridge, MA, USA: MIT Press, 1988, pp. 305–313. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf.

[100] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–35, 2017, ISSN: 0360-0300. DOI: 10.1145/3054912. [Online]. Available: https://doi.org/10.1145/3054912.

[101] S. Ross, G. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635. eprint: 1011.0686. [Online]. Available: https://arxiv.org/abs/1011.0686.

[102] J. Grigsby and Y. Qi, *A closer look at advantage-filtered behavioral cloning in high-noise datasets*, 2021. arXiv: 2110.04698 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2110.04698.

[103] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106 622, 2021, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2020.106622. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705120307516.

[104] A. Karras, C. Karras, N. Schizas, M. Avlonitis, and S. Sioutas, "AutoML with bayesian optimizations for big data management," *Information*, vol. 14, no. 4, p. 223, 2023, ISSN: 2078-2489. DOI: 10.3390/info14040223.

[105] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-598.html.

[106] F. Wilkinson, A. Cockrean, W.-C. Lin, S. McIntosh-Smith, and T. Deakin, "Assessing the gpu offload threshold of gemm and gemv kernels on modern heterogeneous hpc systems," in *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '24, Atlanta, GA, USA: IEEE Press, 2025, pp. 1481–1495, ISBN: 9798350355543. DOI: 10.1109/SCW63240.2024.00188. [Online]. Available: https://doi.org/10.1109/SCW63240.2024.00188.

[107] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1510.00149.

[108] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012, ISSN: 1532-4435. [Online]. Available: http://jmlr.org/papers/v13/bergstra12a.html.

[109] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in Neural Information Processing Systems*, vol. 24, pp. 2546–2554, 2011. [Online]. Available: http://papers.neurips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

[110] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[111] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017. DOI: 10.48550/arXiv.1603.06560.

[112] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, Association for Computational Linguistics, 2019, pp. 3645–3650. DOI: 10.18653/v1/P19-1355.

[113] J. McNally, "Hydrogen-diesel dual fuel combustion characterization for an internal combustion engine," M.S. thesis, University of Alberta, 2024. DOI: https://doi.org/10.7939/r3-z8jn-7s67.

[114] D. Gordon et al., "Development and experimental validation of a field programmable gate array–based in-cycle direct water injection control strategy for homogeneous charge compression ignition combustion stability," *International Journal of Engine Research*, vol. 20, no. 10, pp. 1101–1113, 2019. DOI: 10.1177/1468087419841744. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/1468087419841744.

[115] S. Amanatidis, M. M. Maricq, L. Ntziachristos, and Z. Samaras, "Measuring number, mass, and size of exhaust particles with diffusion chargers: The dual pegasor particle sensor," *Journal of Aerosol Science*, vol. 92, pp. 1–15, 2016, ISSN: 0021-8502. DOI: https://doi.org/10.1016/j.jaerosci.2015.10.005.

[116] E. GmbH, *Noxcant module datasheet*, Accessed: 2025-06-03, 2014. [Online]. Available: https://www.csm.de/en/downloads?catid=49&id=123%3Adatasheet-noxcant&task=download.send.

[117] D. Gordon et al., "Development and experimental validation of a real-time capable field programmable gate array–based gas exchange model for negative valve overlap," *International Journal of Engine Research*, vol. 21, no. 3, pp. 421–436, 2020, ISSN: 1468-0874. DOI: 10.1177/1468087418788491.

[118] D. C. Gordon, "Realtime machine learning based in-cycle control of homogeneous charge compression ignition," Ph.D. dissertation, University of Alberta, Edmonton, Canada, 2023. DOI: 10.7939/r3-3cgz-p614. [Online]. Available: https://era.library.ualberta.ca/items/f1560cd5-770e-4145-9ee3-cb437191c468.

[119] V. Sharma, "Safe reinforcement learning for hydrogen dual-fuel engine control," Veröffentlicht auf dem Publikationsserver der RWTH Aachen University; Masterarbeit, Rheinisch-Westfälische Technische Hochschule Aachen, 2025, Masterarbeit, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, 2025, viii, IX, 89 Seiten : Illustrationen. DOI: 10.18154/RWTH-2025-02451. [Online]. Available: https://publications.rwth-aachen.de/record/1006373.

[120] F. Zurbriggen, T. Ott, and C. H. Onder, "Fast and robust adaptation of lookup tables in internal combustion engines: Feedback and feedforward controllers designed independently," *Proceedings of the Institution of Mechanical Engineers, Part D*, vol. 230, no. 6, pp. 723–735, 2016. DOI: 10.1177/0954407015595336. [Online]. Available: https://doi.org/10.1177/0954407015595336.

[121] Gamma Technologies, LLC, *Gt-power engine simulation software*, Version 2025, Westmont, IL, USA: Gamma Technologies, LLC, 2025. [Online]. Available: https://www.gtisoft.com/gt-power/.

[122] ANSYS, Inc., *Ansys fluent user's guide*, Canonsburg, PA, USA: ANSYS, Inc., 2024. [Online]. Available: https://www.ansys.com/products/fluids/ansys-fluent.

[123] Convergent Science, Inc., *Converge cfd software*, Version 3.1, Madison, WI, USA: Convergent Science, Inc., 2024. [Online]. Available: https://convergecfd.com.

[124] E. Hendricks and S. C. Sorenson, "Mean value modelling of spark ignition engines," in *SAE Technical Paper Series*, SAE International, 1990. DOI: 10.4271/900616. [Online]. Available: https://doi.org/10.4271/900616.

[125] B. Saerens, M. Diehl, J. Swevers, and E. Van den Bulck, "Model predictive control of automotive powertrains - first experimental results," in *2008 47th IEEE Conference on Decision and Control*, 2008, pp. 5692–5697. DOI: 10.1109/CDC.2008.4738740.

[126] F. Tambon et al., "How to certify machine learning based safety-critical systems? a systematic literature review," *Automated Software Engineering*, vol. 29, no. 2, 2022. DOI: 10.1007/s10515-022-00337-x.

[127] J. Rezaei, M. Shahbakhti, B. Bahri, and A. A. Aziz, "Performance prediction of hcci engines with oxygenated fuels using artificial neural networks," *Applied Energy*, vol. 138, pp. 460–473, 2015, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2014.10.088. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261914011489.

[128] J. Syed, R. U. Baig, S. Algarni, Y. S. Murthy, M. Masood, and M. Inamurrahman, "Artificial neural network modeling of a hydrogen dual fueled diesel engine characteristics: An experiment approach," *International Journal of Hydrogen Energy*, vol. 42, no. 21, pp. 14 750–14 774, 2017, ISSN: 0360-3199. DOI: 10.1016/j.ijhydene.2017.04.096. [Online]. Available: https://www.researchgate.net/publication/316950147_Artificial_Neural_Network_modeling_of_a_hydrogen_dual_fueled_diesel_engine_characteristics_An_experiment_approach.

[129] S. Molina, R. Novella, J. Gomez-Soriano, and M. Olcina-Girona, "New combustion modelling approach for methane-hydrogen fueled engines using machine learning and engine virtualization," *Energies*, vol. 14, no. 20, p. 6732, 2021. DOI: 10.3390/en14206732.

[130] S. Shin, S. Lee, M. Kim, J. Park, and K. Min, "Deep learning procedure for knock, performance and emission prediction at steady-state condition of a gasoline engine," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 234, no. 14, pp. 3347–3361, 2020, ISSN: 0954-4070. DOI: 10.1177/0954407020932690.

[131] RWTH Aachen University, *Claix-2023 hpc cluster documentation*, https://help.itc.rwth-aachen.de/en/service/rhr4fjjutttf/article/fbd107191cf14c4b8307f44f545cf68a/, Accessed: 2025-04-28, 2023.

[132] G. Labs, *Grafana: The open and composable observability and data visualization platform*, https://github.com/grafana/grafana, Accessed: 2025-05-26, 2025.

[133] M. Abadi et al., *Tensorflow: Large-scale machine learning on heterogeneous systems*, https://www.tensorflow.org/about/bib, Accessed: 2025-05-26, 2015.

[134] F. Chollet et al., *Keras*, https://keras.io, 2015.

[135] G. van Rossum and the Python Development Team, *Python language reference, version 3.13.3*, Accessed: 2025-05-26, Python Software Foundation, 2025. [Online]. Available: https://docs.python.org/3/reference/index.html.

[136] S. Developers, *Singularity*, https://doi.org/10.5281/zenodo.1310023, Accessed: 2025-05-26, 2021.

[137] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, Accessed: 2025-05-26, Springer, 2003, pp. 44–60. DOI: 10.1007/10968987_3.

[138] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 2623–2631, ISBN: 9781450362016. DOI: 10.1145/3292500.3330701. [Online]. Available: https://doi.org/10.1145/3292500.3330701.

[139] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd. MIT Press, 2018, ISBN: 0262039400. [Online]. Available: https://mitpress.mit.edu/9780262039406/foundations-of-machine-learning/.

[140] J. Bedei et al., "Dynamic measurement with in-cycle process excitation of HCCI combustion: The key to handle complexity of data-driven control?" *International Journal of Engine Research*, vol. 24, no. 3, pp. 1155–1174, 2023. DOI: 10.1177/14680874221078264. eprint: https://doi.org/10.1177/14680874221078264. [Online]. Available: https://doi.org/10.1177/14680874221078264.

[141] J. Bengtsson, P. Strandh, R. Johansson, P. Tunestal, and B. Johansson, "Model predictive control of homogeneous charge compression ignition (hcci) engine dynamics," in *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, 2006, pp. 1675–1680. DOI: 10.1109/CACSD-CCA-ISIC.2006.4776893.

[142] S. Di Cairano, D. Yanakiev, A. Bemporad, I. V. Kolmanovsky, and D. Hrovat, "Model predictive idle speed control: Design, analysis, and experimental evaluation," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 84–97, 2012. DOI: 10.1109/TCST.2011.2112361.

[143] M. Iwadare and M. Ueno, "Multi-variable air-path management for a clean diesel engine using model predictive control," *SAE International Journal of Engines*, vol. 2, no. 1, pp. 764–773, Apr. 2009, ISSN: 1946-3936. DOI: 10.4271/2009-01-0733. [Online]. Available: https://doi.org/10.4271/2009-01-0733.

[144] A. Widd, H.-H. Liao, J. C. Gerdes, P. Tunestål, and R. Johansson, "Hybrid model predictive control of exhaust recompression hcci," *Asian Journal of Control*, vol. 16, no. 2, pp. 370–381, 2014. DOI: https://doi.org/10.1002/asjc.725. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/asjc.725.

[145] S. Wang, D. Yu, J. Gomm, G. Page, and S. Douglas, "Adaptive neural network model based predictive control for air–fuel ratio of SI engines," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 2, pp. 189–200, 2006. DOI: 10.1016/j.engappai.2005.08.005.

[146] V. M. Janakiraman, X. Nguyen, and D. Assanis, "Nonlinear model predictive control of A gasoline HCCI engine using extreme learning machines," *CoRR*, vol. abs/1501.03969, 2015, Extract of PhD Thesis. arXiv: 1501.03969. [Online]. Available: http://arxiv.org/abs/1501.03969.

[147] V. M. Janakiraman, X. Nguyen, and D. Assanis, "An ELM based predictive control method for HCCI engines," *Engineering Applications of Artificial Intelligence*, vol. 48, pp. 106–118, 2016. DOI: 10.1016/j.engappai.2015.10.007.

[148] B. K. Irdmousa, S. Z. Rizvi, J. M. Veini, J. D. Nabert, and M. Shahbakhti, "Data-driven modeling and predictive control of combustion phasing for rcci engines," in *2019 American Control Conference (ACC)*, 2019, pp. 1617–1622. DOI: 10.23919/ACC.2019.8815269.

[149] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," in *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Berlin, Heidelberg: Springer, 2009, pp. 391–417. DOI: 10.1007/978-3-642-01094-1_32.

[150] E. C. Kerrigan and J. M. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," in *Proc. UKACC International Conference (Control 2000)*, Institution of Engineering and Technology, 2000. [Online]. Available: http://hdl.handle.net/10044/1/10241/.

[151] *Model predictive control toolbox*, https://www.mathworks.com/products/mpc.html, The MathWorks, Inc., Natick, Massachusetts, United States, 2025.

[152] A. Domahidi and J. Jerez, *Forces professional*, https://www.embotech.com/forcespro, 2014–2019.

[153] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, pp. 1–17, 2017. DOI: 10.1080/00207179.2017.1316017.

[154] G. Frison and M. Diehl, "HPIPM: A high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020. DOI: 10.1016/j.ifacol.2020.12.073.

[155] G. Frison, D. Kouzoupis, J. Jørgensen, and M. Diehl, "An efficient implementation of partial condensing for nonlinear model predictive control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 4457–4462. DOI: 10.1109/CDC.2016.7798946.

[156] Raspberry Pi Ltd, *Raspberry pi 400 product brief*, Available at https://datasheets.raspberrypi.com/rpi400/raspberry-pi-400-product-brief.pdf, 2020.

[157] Arm Ltd, *Arm cortex-a72 mpcore processor technical reference manual*, Available at https://developer.arm.com/documentation/100095/latest/, 2016.

[158] C. Wouters et al., "Evaluation of the potential of direct water injection in HCCI combustion," in *2019 JSAE/SAE Powertrains, Fuels and Lubricants*, ser. SAE Technical Paper Series, SAE International, 2019. DOI: 10.4271/2019-01-2165.

[159] A. Norouzi, H. Heidarifar, H. Borhan, M. Shahbakhti, and C. R. Koch, "Integrating machine learning and model predictive control for automotive applications: A review and future directions," *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105 878, 2023. DOI: 10.1016/j.engappai.2023.105878.

[160] L. M. Das, "Hydrogen engines: A view of the past and a look into the future," *International Journal of Hydrogen Energy*, vol. 15, no. 6, pp. 425–443, 1990, ISSN: 0360-3199. DOI: 10.1016/0360-3199(90)90200-I. [Online]. Available: https://doi.org/10.1016/0360-3199(90)90200-I.

[161] S. Leichsenring, "Robustness analysis of a model predictive control for a hydrogen dual-fuel engine," Bachelor thesis, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, 2024.

[162] M. Bojarski et al., *End to end learning for self-driving cars*, 2016. arXiv: 1604.07316 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1604.07316.

[163] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia: IEEE Press, 2018, pp. 1–9. DOI: 10.1109/ICRA.2018.8460487. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8460487.

[164] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Proceedings of Robotics: Science and Systems (RSS)*, Freiburg, Germany, 2019. DOI: 10.15607/RSS.2019.XV.031. [Online]. Available: https://www.roboticsproceedings.org/rss15/p31.pdf.

[165] T. Zhang et al., "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–8. DOI: 10.1109/ICRA.2018.8461249.

[166] M. Laskey, J. Lee, R. Fox, A. D. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 78, 2017, pp. 143–156. DOI: 10.48550/arXiv.1703.09327. [Online]. Available: https://proceedings.mlr.press/v78/laskey17a.html.

[167] S. Zhang, Z. Cao, D. Sadigh, and Y. Sui, "Confidence-aware imitation learning from demonstrations with varying optimality," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21, Red Hook, NY, USA: Curran Associates Inc., 2021, ISBN: 9781713845393. DOI: 10.48550/arXiv.2110.14754.

[168] Espressif Systems, *Esp32 series datasheet*, https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf, Accessed: April 25, 2025, 2023.

[169] F. Borrelli, *Automatic control in the era of artificial intelligence*, Plenary Talk, American Control Conference (ACC), Plenary talk, not available online. See: https://acc2024.a2c2.org/about/plenary-speakers, Jul. 2024.

[170] J. Bedei, L. Koch, K. Badalian, A. Winkler, P. Schaber, and J. Andert, *Safe reinforcement learning for real-world engine control*, arXiv preprint, version 2, Feb. 2026. arXiv: 2501.16613 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2501.16613.

[171] P. K. Mishra, M. V. Gasparino, and G. Chowdhary, "Deep model predictive control with stability guarantees," *IEEE Transactions on Automatic Control*, vol. 70, no. 8, pp. 5460–5467, 2025. DOI: 10.1109/TAC.2025.3550072.

[172] S. Mukherjee, J. Drgoňa, A. Tuor, M. Halappanavar, and D. Vrabie, "Neural lyapunov differentiable predictive control," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 2097–2104. DOI: 10.1109/CDC51059.2022.9992386.

[173] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, Montréal, Canada: Curran Associates Inc., 2018, pp. 6572–6583. DOI: 10.48550/arXiv.1806.07366.

[174] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, p. 4950, 2018. DOI: 10.1038/s41467-018-07210-0.

[175] L. Koch, M. Picerno, K. Badalian, S.-Y. Lee, and J. Andert, "Automated function development for emission control with deep reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 117, p. 105 477, 2023, ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2022.105477. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622004675.

[176] P. Garg, E. Silvas, and F. Willems, "Safe and time-efficient exploration in Reinforcement Learning-based control of a vehicle thermal systems," *Control Engineering Practice*, vol. 164, p. 106 458, 2025. DOI: 10.1016/j.conengprac.2025.106458. [Online]. Available: https://doi.org/10.1016/j.conengprac.2025.106458.

[177] F. Mohr and M. Wever, *Naive automated machine learning*. Apr. 2023, vol. 112, pp. 1131–1170. DOI: 10.1007/s10994-022-06200-0. [Online]. Available: https://doi.org/10.1007/s10994-022-06200-0.

[178] Pegasor, *Pegasor pps-m brochure*, Pegasor, 2021. Accessed: Apr. 22, 2023. [Online]. Available: https://pegasor.fi/site/attachments/Pegasor_PPS_M_web_2.pdf.

Appendix

Fundamentals

$$\|x\|_M^2 = x^\top M x = \sum_{i=1}^{N} m_i x_i^2, \qquad x \in \mathbb{R}^N \tag{A.1a}$$

$$\begin{aligned} M &= diag\,(m_1,\ \ldots,\ m_N) \\ &= \begin{bmatrix} m_1 & & \\ & \ddots & \\ & & m_N \end{bmatrix} \end{aligned} \tag{A.1b}$$

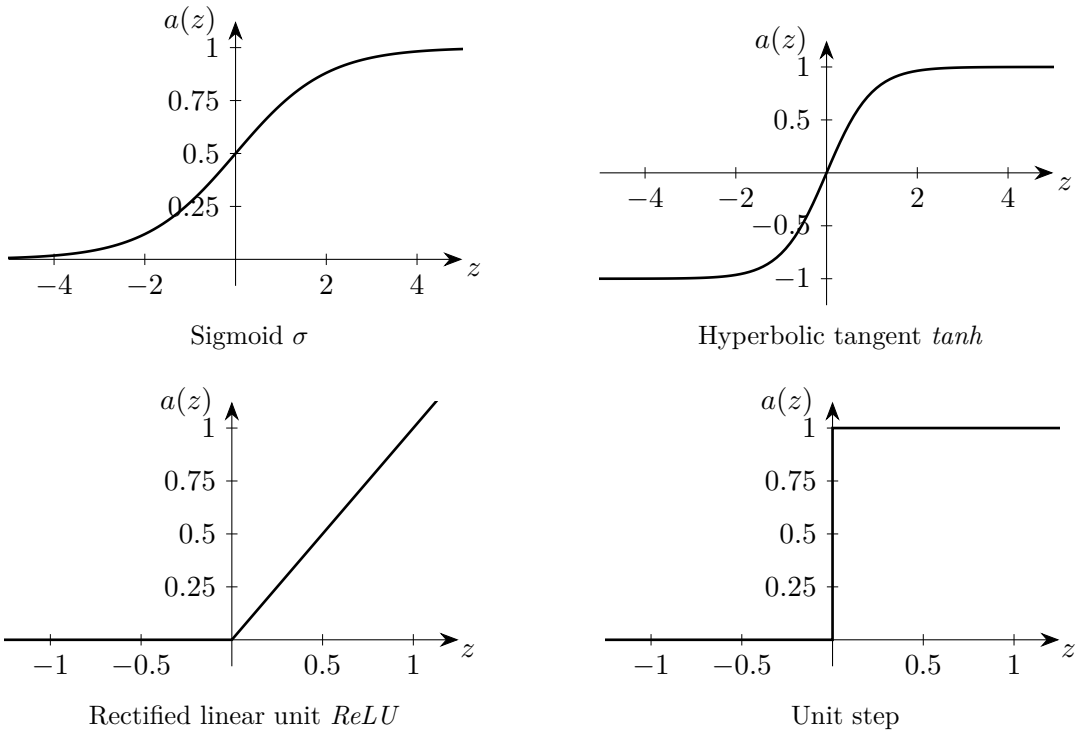


Figure A.1: Activation functions. $a(z)$: output, z: input.

Algorithm A.1 Pseudo-code. Training procedure for FFNN and RNN using SGD, based on [88].

Require: Training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, validation dataset $\mathcal{D}_{\text{val}}$, network type (`FFNN` or `RNN`)
Ensure: Trained network parameters θ^*

Initialize network parameters θ randomly
Set learning rate α, number of epochs T, batch size B
Define loss function $\mathcal{L}(y_i, \hat{y}_i)$ where $\hat{y}_i = f_\theta(x_i)$
Initialize record: $\mathcal{L}_{\text{val}}^{\text{best}} \leftarrow \infty$, $\theta^* \leftarrow \theta$
for epoch $= 1$ to T **do**
    if network type is `FFNN` **then**
        Shuffle dataset $\mathcal{D}$
        Randomly divide into mini-batches of size B
    else if network type is `RNN` **then**
        Divide dataset $\mathcal{D}$ into *sequential* mini-batches of size B
        Optionally reset hidden states between batches
    end if
    for each mini-batch $\mathcal{B} \subset \mathcal{D}$ of size B (sequential if RNN) **do**
        Compute predictions $\hat{y}_i = f_\theta(x_i)$ for all $(x_i, y_i) \in \mathcal{B}$
        Compute gradient $\nabla_\theta \mathcal{L}$ over the mini-batch
        Update parameters: $\theta \leftarrow \theta - \alpha \cdot \nabla_\theta \mathcal{L}$
    end for
    Evaluate validation loss $\mathcal{L}_{\text{val}} = \mathcal{L}(y, f_\theta(x))$ on $\mathcal{D}_{\text{val}}$
    if $\mathcal{L}_{\text{val}} < \mathcal{L}_{\text{val}}^{\text{best}}$ **then**
        Update best parameters: $\theta^* \leftarrow \theta$, $\mathcal{L}_{\text{val}}^{\text{best}} \leftarrow \mathcal{L}_{\text{val}}$
    end if
end for
return best-performing parameters θ^* based on lowest validation loss

Algorithm A.2 Pseudo-code. BC of an expert control policy, based on [99].

Require: Expert dataset $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N}$, where s_i are states and a_i are expert actions
Ensure: Cloned policy π_θ

Split $\mathcal{D}$ into a training set $\mathcal{D}_{\text{train}}$ and a validation set $\mathcal{D}_{\text{val}}$
Initialize policy model π_θ with parameters θ
Define a loss function $\mathcal{L}(\theta)$ to minimize the discrepancy between $\pi_\theta(s_i)$ and a_i
Set training hyperparameters and number of training iterations T
for $t = 1$ to T **do**
    Update policy parameters θ by minimizing $\mathcal{L}(\theta)$ on $\mathcal{D}_{\text{train}}$
    Evaluate $\mathcal{L}(\theta)$ on $\mathcal{D}_{\text{val}}$
end for
return trained policy π_θ

Algorithm A.3 Pseudo-code. Automated machine learning (AutoML) for DNNs. Own representation, based on [177].

Require: Dataset $\mathcal{D}$ (split into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$), Search Space $\mathcal{S}$, Objective Function $\mathcal{O}$ (to be minimized, e.g. the validation loss)
Ensure: Optimal Neural Network Architecture A^*
Initialize search space $\mathcal{S}$ with candidate architectures
Set maximum number of trials T
Initialize best objective value $P_{\text{best}} \leftarrow +\infty$, optimal architecture $A^* \leftarrow \emptyset$
for $t = 1$ to T **do**
    Select architecture $A_t \in \mathcal{S}$ using a search strategy
    Train A_t on training data $\mathcal{D}_{\text{train}}$ and validate on $\mathcal{D}_{\text{val}}$
    Evaluate objective value $P_t \leftarrow \mathcal{O}(A_t, \mathcal{D}_{\text{val}})$
    if $P_t < P_{\text{best}}$ **then**
        Update $P_{\text{best}} \leftarrow P_t$
        Update $A^* \leftarrow A_t$
    end if
    Update search strategy with feedback from P_t
end for
return A^*

Experimental Setup

Table A.1: *Cummins QSB* 4.5 Tier 3 engine parameter, as tested

Parameter	**Value**
Total displacement	4.46L
Number of cylinders	4
Bore	107 mm
Stroke	124 mm
Connecting rod length	192 mm
Compression ratio	17.2:1
Piston protrusion	0.43 mm
Headgasket thickness	1.6 mm
Valve train	Pushrod
No. of valves (In/Ex)	2/2
No. of camshafts	1
Max. valve lift (In/Ex)	8 mm/8 mm
Valve angle (In/Ex)	0°/0°
Valve diameter (In/Ex)	33 mm/33 mm
Combustion chamber type	Bowl in piston
Injection type	Direct
Injection pressure	300 bar to 1500 bar
Spray angle	124°
Injector holes	8 holes
Injection plane	Single
Injector actuation	Solenoid

Table A.2: Rapid prototyping ECU Specifications

	Parameter	**Specification**
Processor	*dSPACE 1401*	*IBM PPC-750GL*
	Speed	900 MHz
	Memory	16 MB main memory
Inputs/Outputs board	*dSPACE 1513*	
	Analog input	24 parallel channels
	Resolution	16 bit
	Sampling frequency	1 Msample/s
	Analog input	32 multiplexed channels
	Resolution	16 bit
	Sampling frequency	200 ksample/s
	Analog output	12 channels
	Digital input	40 channels
	Digital output	40 channels
FPGA	*dSPACE 1514*	*Xilinx Kintex-7*
	Flip-flops	407600
	Lookup table	203800
	Memory lookup table	64000
	Block Memory	445
	Digital signal processing	840
	Inputs/Outputs	478

Table A.3: Specifications of *PPS-M* emissions measurement system [178].

	Parameter	**Specification**
Pegasor PPS-M		
	Particle concentration range	1 µg/m^3 – 290 mg/m^3
	Minimum particle size	10 nm
	Sample temperature	200 °C
	Sample rate	100 Hz
	Signal-to-noise-ratio	100 dB

Table A.4: *ECM NOxCANt* module specifications [116].

Parameter	Specification
Measurement ranges	
NO_x	0 ppm to 5000 ppm
Lambda (λ)	0.40–25
Air-fuel ratio (AFR)	6.0–364
Oxygen concentration (O_2)	0% to 25%
Accuracy	
NO_x	±5 ppm (0–200 ppm), ±20 ppm (200–1000 ppm), ±2% (1000–5000 ppm)
Lambda (λ)	±0.008 at $\lambda = 1.0$; ±0.016 (0.8–1.2); ±0.018 elsewhere
AFR	±0.15 at AFR = 14.6; ±0.40 (12–18); ±1.0 elsewhere
O_2	±0.4% (0–2 %); ±0.8% elsewhere
Response Time	
Lambda, AFR, Φ, O_2	< 150 ms
NO_x	< 1 s

Table A.5: Properties of cylinder pressure sensors and amplifiers.

Cylinder	Model	Type	Sensitivity / (pC/bar)	ω_o / kHz	Calibration / (bar/V)
1	*Kistler 6124A*	Piezoelectric	-29.98	≥ 65	19.701
2	*Kistler 6125C*	Piezoelectric	-33	≥ 70	20.596
3	*Kistler 6125C*	Piezoelectric	-33	≥ 70	20.482
4	*Kistler 6125C*	Piezoelectric	-33	≥ 70	20.858

Table A.6: Boundaries for engine control input variables and their update interval and frequency used for PRBS excitation.

Variable	**Min. boundary**	**Max. boundary**
t_{Main} (DOI Main)	0.18 ms	0.5 ms
t_{P2M} (P2M)	430 µs	1400 µs
α_{Main} (SOI Main)	−10 CADbTDC	15 CADbTDC
t_{H2} (DOI H_2)	1.5 ms	4.5 ms
$t_{\text{Const}}/f_{\text{Change}}$	15 cycles (0.83 Hz)	35 cycles (0.35 Hz)

Table A.7: DNN-based engine plant model. Training hyperparameter settings.

Parameter	**Value**
Optimizer	Adam
Validation metric	MSE
Max. epochs	5000
Mini-batch size	512
Verbose frequency	1
Initial learning rate	0.001
Learn rate schedule	Piece-wise drop by 25% every 250 epochs
L2 regularization	0.1

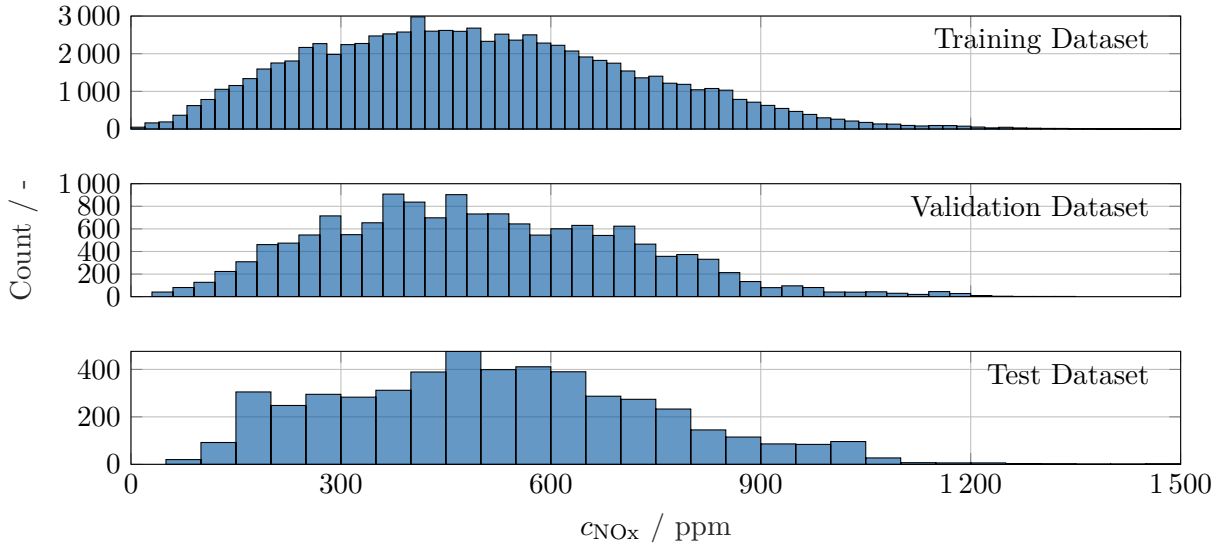


Figure A.2: Experimental pseudo-random binary sequence generated engine data distribution for NO_x (c_{NOx}). Upper plot: training dataset (80%), mid plot: validation dataset (15%), lower plot: unseen test dataset (5%). Total data points: 99,800.

$$\text{Total Configurations} = \sum_{n_{\text{before}}=1}^{4} \sum_{n_{\text{after}}=1}^{4} 4^{n_{\text{before}}} \times 4^{n_{\text{after}}} \times 4 = 462{,}400. \tag{A.2}$$

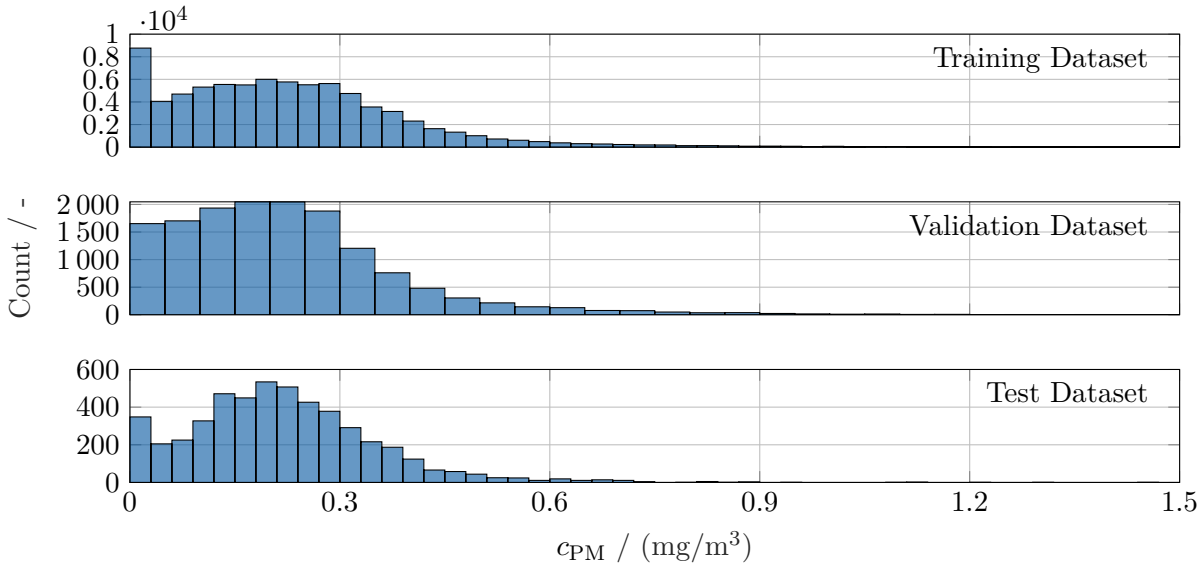


Figure A.3: Experimental pseudo-random binary sequence generated engine data distribution for soot (PM, c_{PM}). Upper plot: training dataset (80%), mid plot: validation dataset (15%), lower plot: unseen test dataset (5%). Total data points: 99,800.

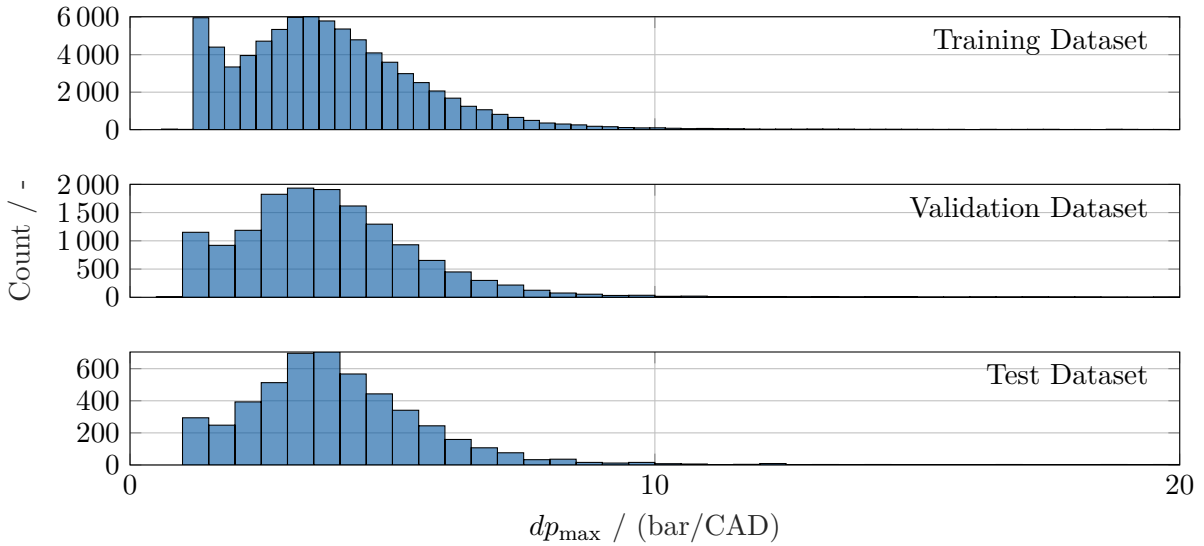


Figure A.4: Experimental pseudo-random binary sequence generated engine data distribution for dp_{max}. Upper plot: training dataset (80%), mid plot: validation dataset (15%), lower plot: unseen test dataset (5%). Total data points: 99,800.

Table A.8: DNN (*no feedback* variant) prediction performance on validation and unseen test datasets. Clipped values are normalized by a range where the first and 99th percentiles have been removed to avoid outliers.

Metric	Validation	Test
p_{mi}		
MAE / bar	0.17	0.14
$\mathrm{NMAE}_{\mathrm{clipped}}$ / –	2.2%	1.9%
RMSE / bar	0.36	0.23
$\mathrm{NRMSE}_{\mathrm{clipped}}$ / –	4.7%	3.0%
c_{NOx}		
MAE / ppm	31	34
$\mathrm{NMAE}_{\mathrm{clipped}}$ / –	3.1%	3.4%
RMSE / ppm	42	51
$\mathrm{NRMSE}_{\mathrm{clipped}}$ / –	4.2%	5.1%

Metric	Validation	Test
c_{PM}		
MAE / $\mathrm{mg/m^3}$	0.12	0.10
$\mathrm{NMAE}_{\mathrm{clipped}}$ / –	8.2%	6.4%
RMSE / $\mathrm{mg/m^3}$	0.34	0.13
$\mathrm{NRMSE}_{\mathrm{clipped}}$ / –	22.6%	8.8%
dp_{max}		
MAE / bar/CAD	0.87	0.91
$\mathrm{NMAE}_{\mathrm{clipped}}$ / –	6.8%	7.1%
RMSE / bar/CAD	1.9	1.6
$\mathrm{NRMSE}_{\mathrm{clipped}}$ / –	15.0%	12.2%

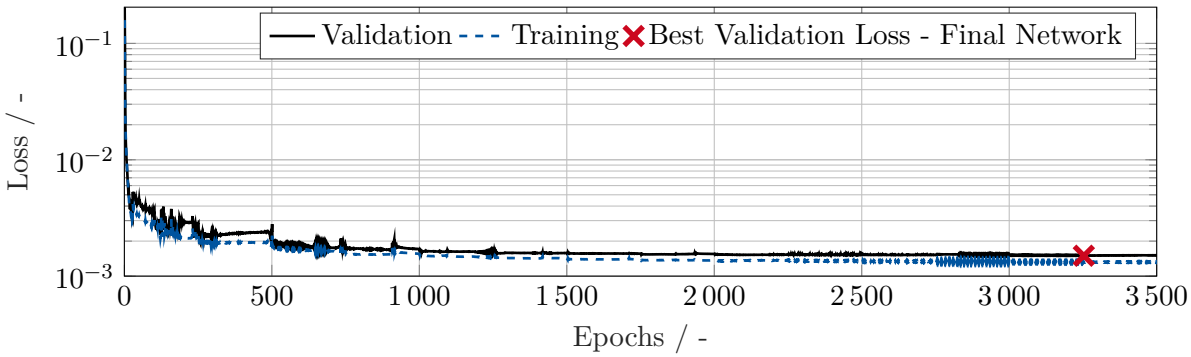


Figure A.5: MPC model without feedback. Loss-epoch plot for the DNN training. The final network is chosen according to the best validation loss.

Table A.9: Comparison of selected DNN architectures based on FLOPs, GE, and network structure. All models were trained with an initial learning rate of approximately 0.0008 and a batch size of 512. NL before: number of layers before GRU; NL after: number of layers after GRU, units before/after: number of neurons per layer.

Trial number	FLOPs	GE	NL before	units before	GRU size	NL after	Units after
141 (Best Pareto overall)	4024	0.00375	1	32	6	4	32, 8, 32, 4
1028 (Best FLOPs on Pareto)	328	0.00420	1	8	2	1	4
13077 (Best GE on Pareto)	14980	0.00365	1	32	8	4	64, 32, 32, 64

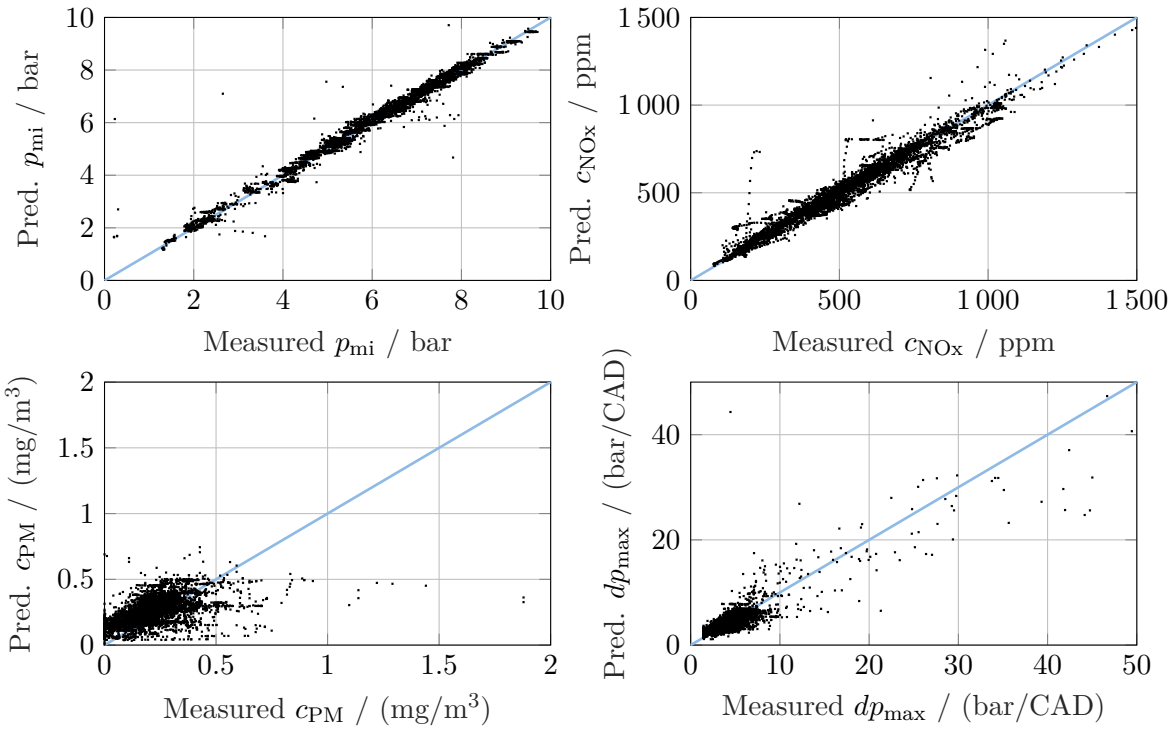


Figure A.6: MPC model without feedback. Predicted DNN outputs vs. experimental unseen test dataset.

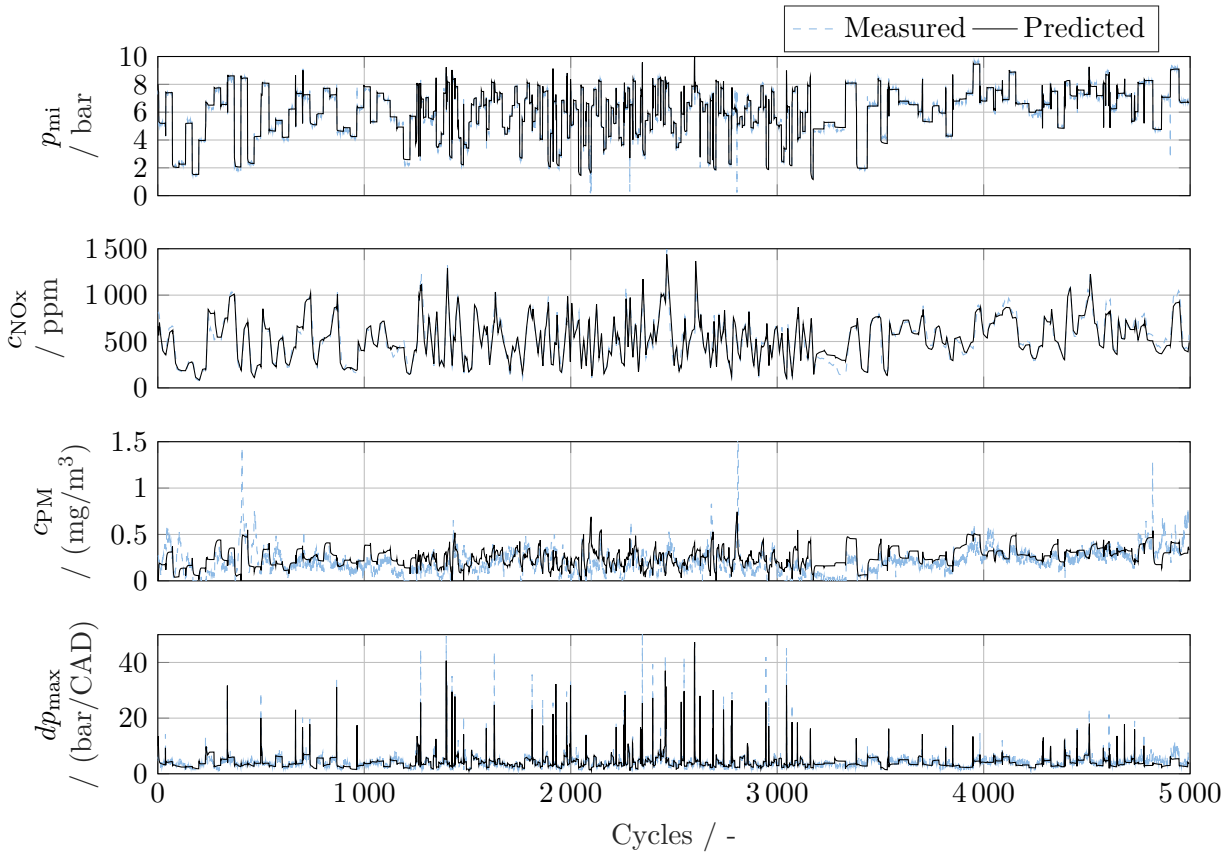


Figure A.7: MPC model without feedback. Predicted DNN outputs on the experimental unseen test dataset in time domain.

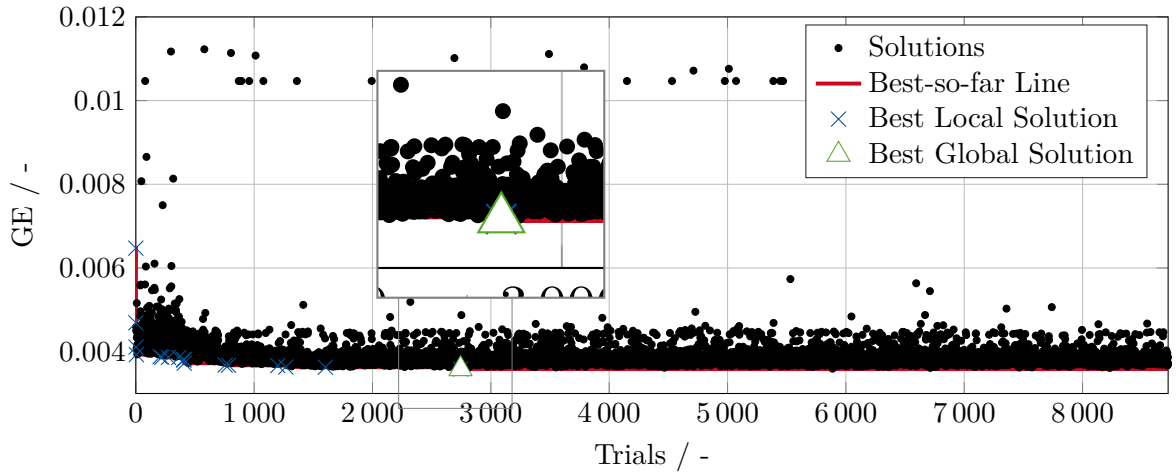


Figure A.8: `Optuna` AutoML single-objective optimization results, prediction performance (GE: generalization error), conducted on HPC using 384 cores in parallel, 8,700 trials total, 44 hours total runtime, energy consumption approx. 121 kWh.

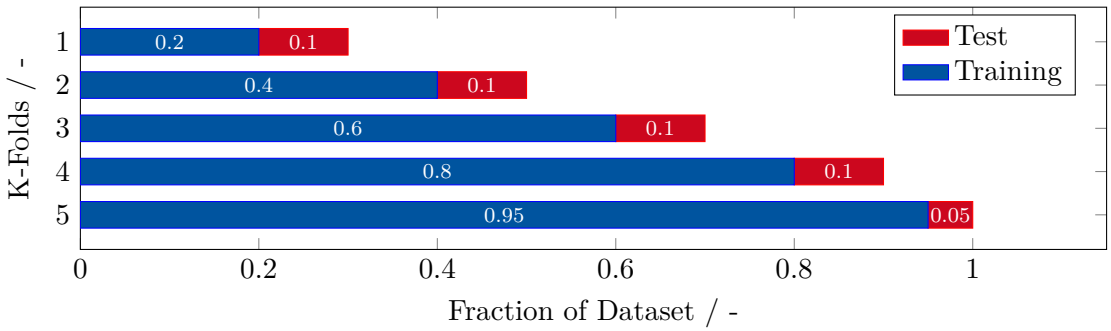


Figure A.9: Time series split method of cross-validation approach for $k = 5$ folds, normalized over the dataset.

Table A.10: *Core MPC*. `acados` solver settings.

Parameter	Value
Horizon length N	3
NLP solver max. iterations	3
NLP solver tolerance	0.001
QP solver max. iterations	50

Table A.11: Experimental results comparison of varying controller aggressiveness settings of the *Core MPC*. Tracking the shortened standard load reference trajectory of 4100 cycles at 1500 min^{-1}.

Performance metric	Mid steps (*Core MPC*)	Slow steps	Fast steps
p_{mi} tracking			
MAE / bar	0.117	0.133	0.127
RMSE / bar	0.176	0.267	0.215
NRMSE / -	3.51%	5.34%	4.30%
c_{NOx}			
max / ppm	971	904	1005
mean / ppm	483	443	425
c_{PM}			
max / $\mathrm{mg/m^3}$	1.23	0.57	1.10
mean / $\mathrm{mg/m^3}$	0.21	0.15	0.20
dp_{max}			
max / bar/CAD	11.13	9.99	8.30
mean / bar/CAD	4.39	3.99	3.57
HES			
mean / -	39.67%	40.06%	37.52%
max / -	55.1%	53.49%	51.14%

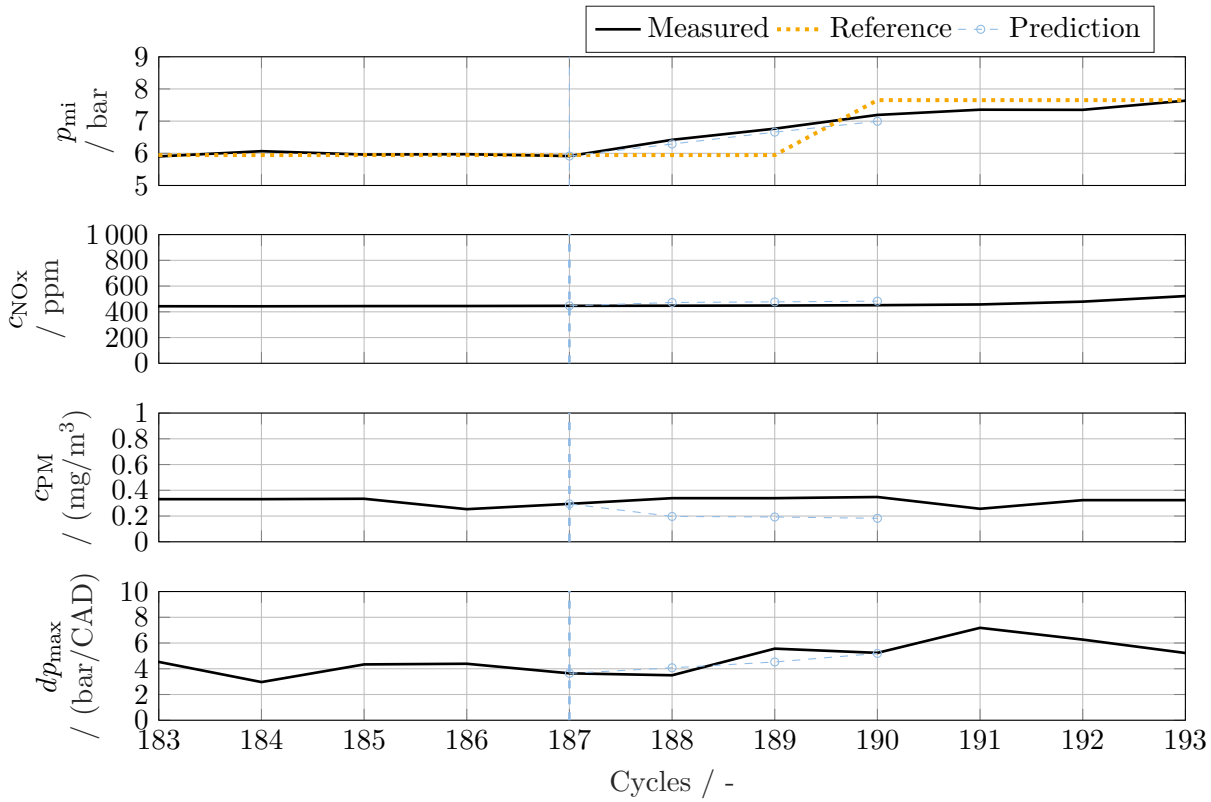


Figure A.10: *Core MPC*. Experimental tracking results. Predictions extract including constraints, zoom on load step, highlighting the MPC's predictions quality and constraint satisfaction.

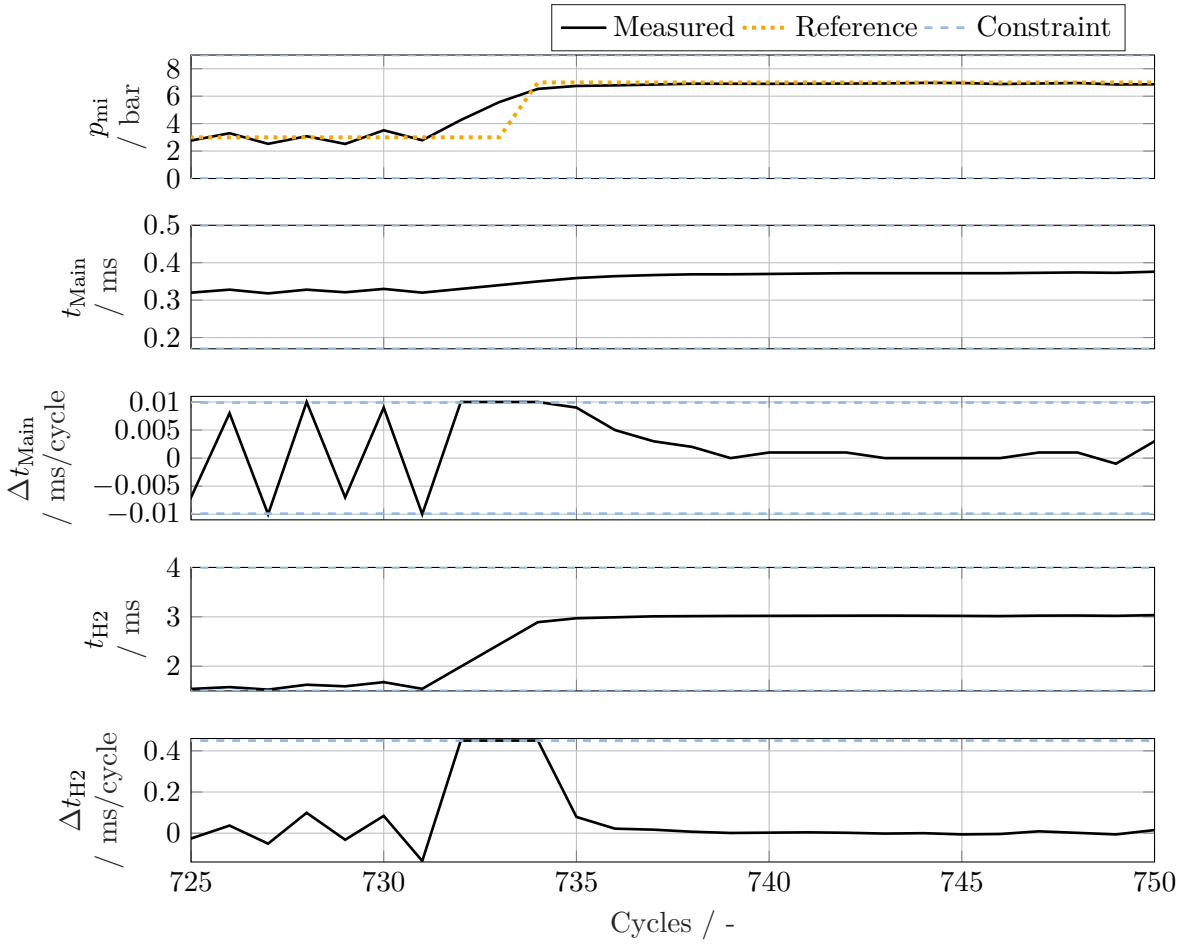


Figure A.11: MPC *fast steps* variant. Experimental results load tracking. Controls change rates extract including constraints.

Algorithm A.4 Local sensitivity analysis of a DNN-based engine model. [119]

Require: Trained model f_θ, system parameters $\mathcal{P}$, number of baseline samples N_b, number of hidden state samples N_l, number of inputs n_u, number of outputs n_y, input bounds $\mathcal{B}$, perturbation range $\Delta = [\Delta_{\min}, \Delta_{\max}]$, number of perturbation points N_p

Ensure: Averaged sensitivity matrix $S \in \mathbb{R}^{n_y \times n_u}$

Initialize sensitivity matrix $S \leftarrow \mathbf{0}_{n_y \times n_u}$
Sample random initial hidden states $h^{(l)} \sim \mathcal{U}(-1, 1)^{n_{\text{hidden}}}$ for $l = 1, \ldots, N_l$
for $b = 1$ to N_b **do**
  Sample baseline input $u_0 \in \mathcal{B}$
  Initialize per-sample sensitivity matrix $S_b \leftarrow \mathbf{0}_{n_y \times n_u}$
  for $l = 1$ to N_l **do**
    Estimate previous output $y_{t-1} \leftarrow f_{\text{state}}(h^{(l)}, \mathcal{P})$
    Form augmented input $u \leftarrow [\, u_0;\, y_{t-1} \,]$
    Predict baseline output $y_0 \leftarrow f_\theta(h^{(l)}, u, \mathcal{P})$
    for $i = 1$ to n_u **do**
      Define perturbation steps $\{\delta_j\}_{j=1}^{N_p} \leftarrow \text{linspace}(\Delta_{\min}, \Delta_{\max}, N_p)$
      Initialize sensitivity accumulator sensSum $\leftarrow \mathbf{0}_{n_y}$
      for $j = 1$ to N_p **do**
        Perturb input $\tilde{u} \leftarrow u$; $\tilde{u}_i \leftarrow u_i + \delta_j$
        Predict perturbed output $y_j \leftarrow f_\theta(h^{(l)}, \tilde{u}, \mathcal{P})$
        Update sensSum $\leftarrow$ sensSum $+ |(y_j - y_0)/\delta_j|$
      end for
      Compute average sensitivity $s_i \leftarrow \text{sensSum}/N_p$
      Update per-sample sensitivity $S_b(:, i) \leftarrow S_b(:, i) + s_i$
    end for
  end for
  Update global sensitivity matrix $S \leftarrow S + S_b/N_l$
end for
Average over all baseline samples $S \leftarrow S/N_b$
return Sensitivity matrix S

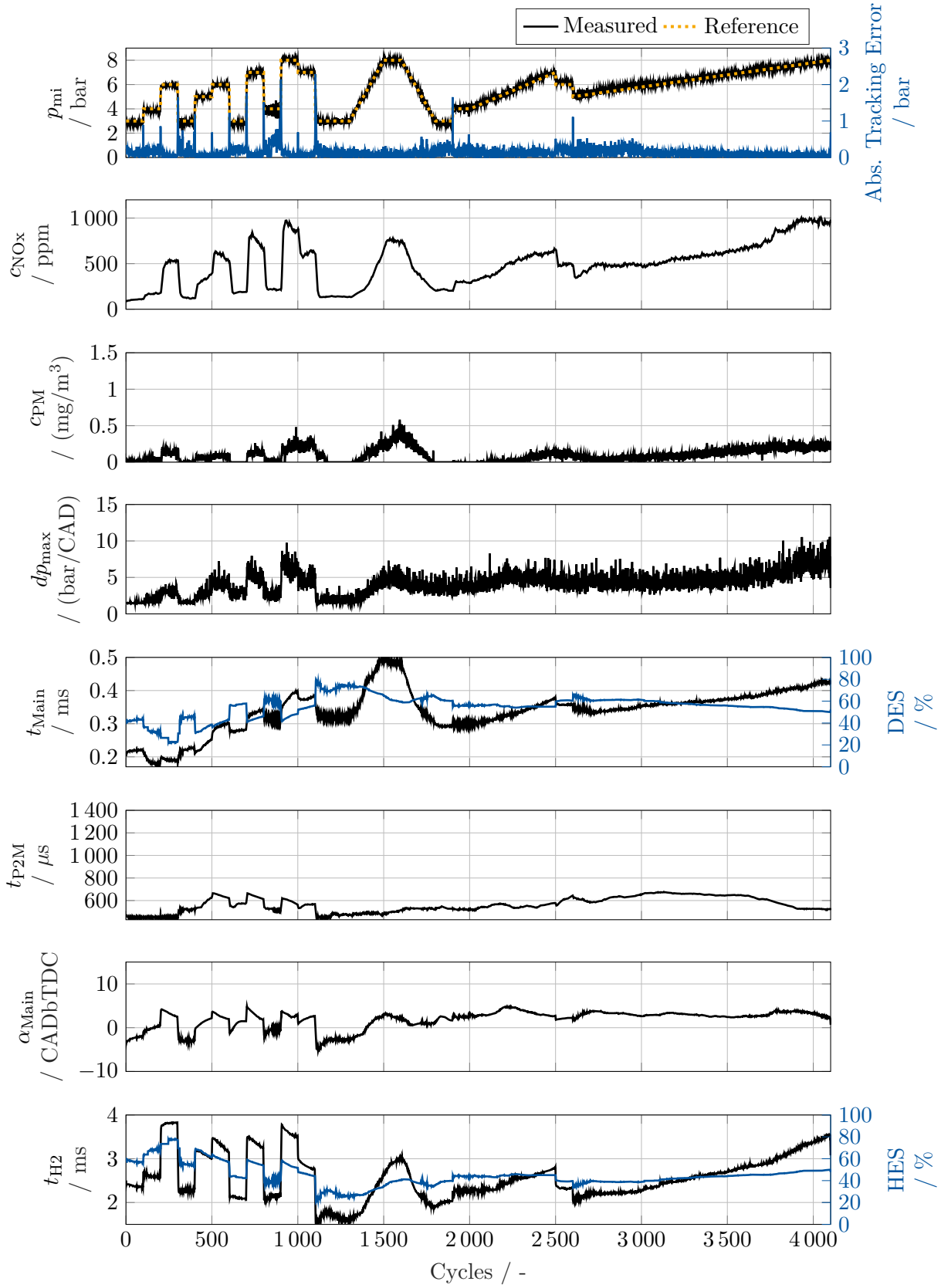


Figure A.12: MPC *high H_2 utilization* variant. Experimental results load tracking.

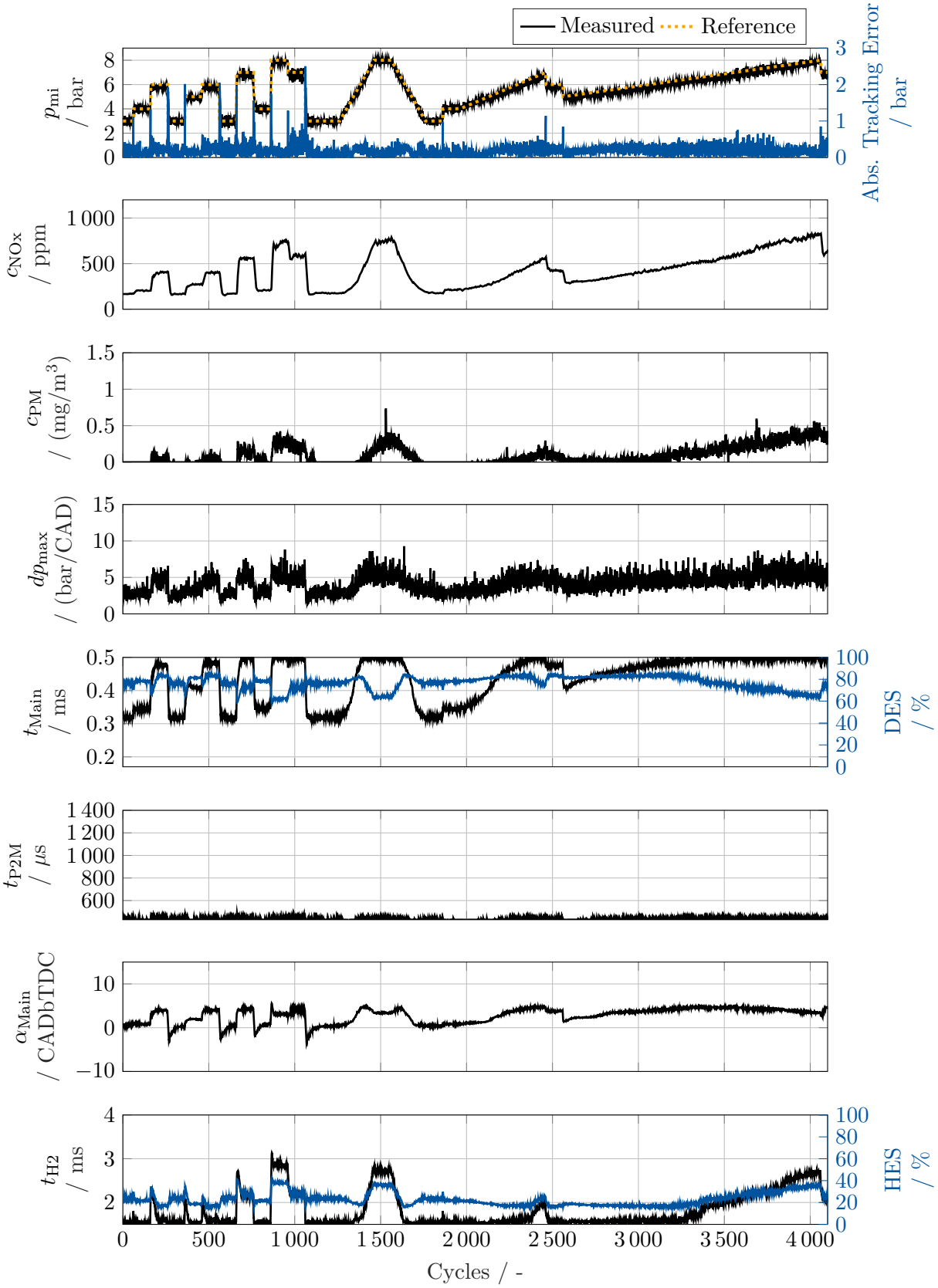


Figure A.13: MPC *low H_2 utilization* variant. Experimental results load tracking.

Table A.12: Experimental results comparison of *Core MPC* for varying engine speeds. Tracking the standard load reference trajectory of 4900 cycles at varying engine speeds of $1200\,\mathrm{min}^{-1}$, $1500\,\mathrm{min}^{-1}$, and $1800\,\mathrm{min}^{-1}$, demonstrating model-plant mismatch.

Performance metric	***Core MPC* 1500 min^{-1}**	**Low speed 1200 min^{-1}**	**High speed 1800 min^{-1}**
p_{mi} tracking			
MAE / bar	0.117	0.1955	0.228
RMSE / bar	0.176	0.277	0.293
NRMSE / -	3.51%	5.54%	5.85%
c_{NOx}			
max / ppm	970.69	1406.26	648.05
mean / ppm	482.8	689.38	326.01
c_{PM}			
max / mg/m^3	1.23	1.50	3.63
mean / mg/m^3	0.21	0.26	0.89
dp_{max}			
max / bar/CAD	11.13	14.58	7.10
mean / bar/CAD	4.39	6.13	3.16
HES			
mean / -	39.67%	27.61%	24.49%
max / -	55.10%	74.36%	42.88%

Table A.13: Robust bounds of *Core MPC* to various disturbances in MiL simulation. Evaluated using physical limits like dp_{max} and `acados` solver failures.

Variable	**Lower bound**	**Upper bound**
Noise and cyclic variation		
p_{mi} / bar^2	0	0.57
dp_{max} / bar^2/CAD^2	0	0.57
Zero offset		
p_{mi} / bar	-8	20
dp_{max} / bar/CAD	-20	20
t_{Main} / ms	-0.04	0.02
α_{Main} / CADbTDC	-5	3
t_{P2M} / µs	-0.5	0.5
t_{H2} / ms	-5.5	0.7
Span offset (output = value × input)		
t_{Main} / −	0.8	1.2
α_{Main} / −	0.1	1.4
t_{P2M} / −	0.3	1.4
t_{H2} / −	0.8	1.25

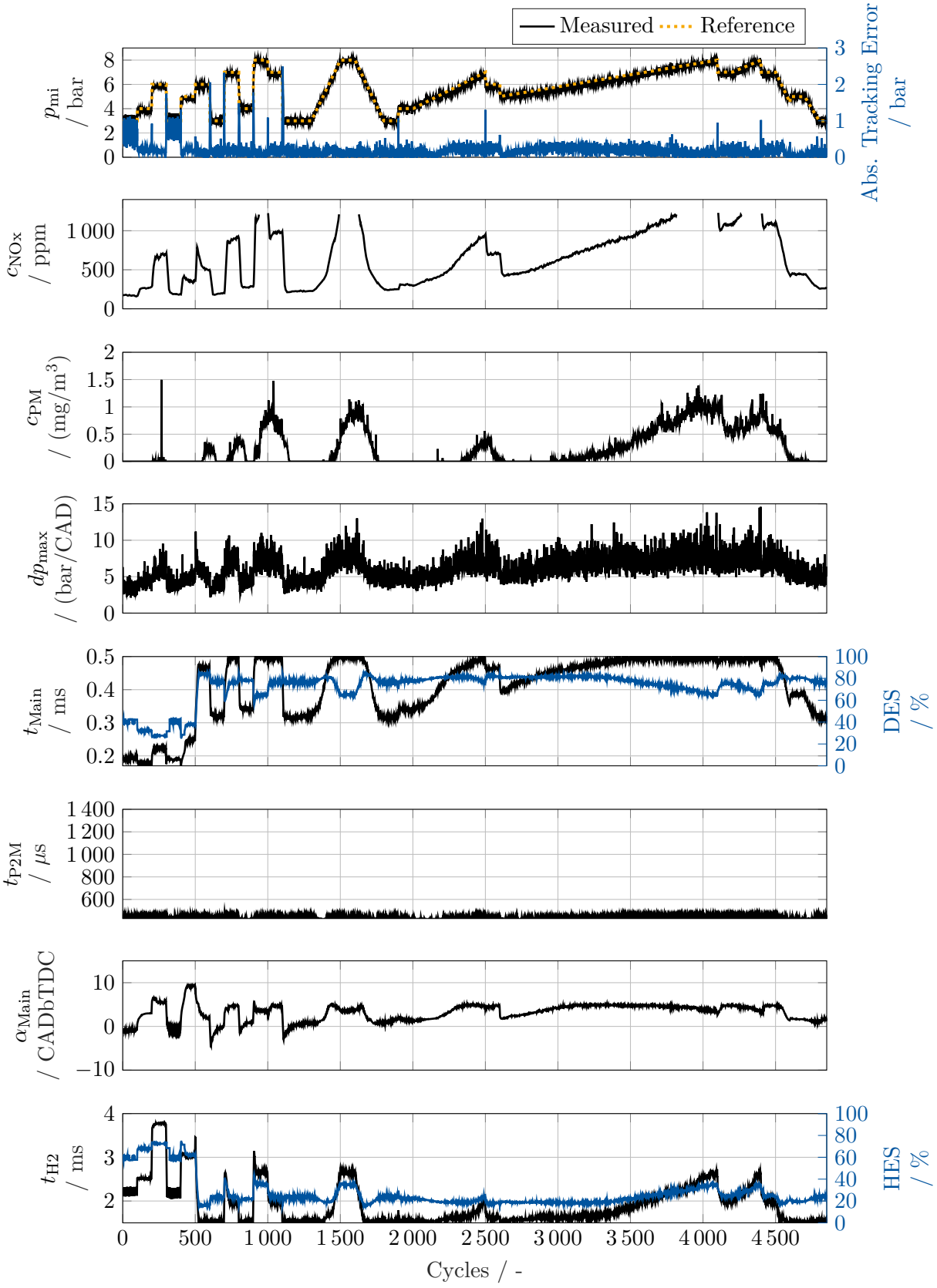


Figure A.14: *Core MPC*. Experimental results, load tracking under model-plant mismatch – low engine speed (1200 min^{-1}).

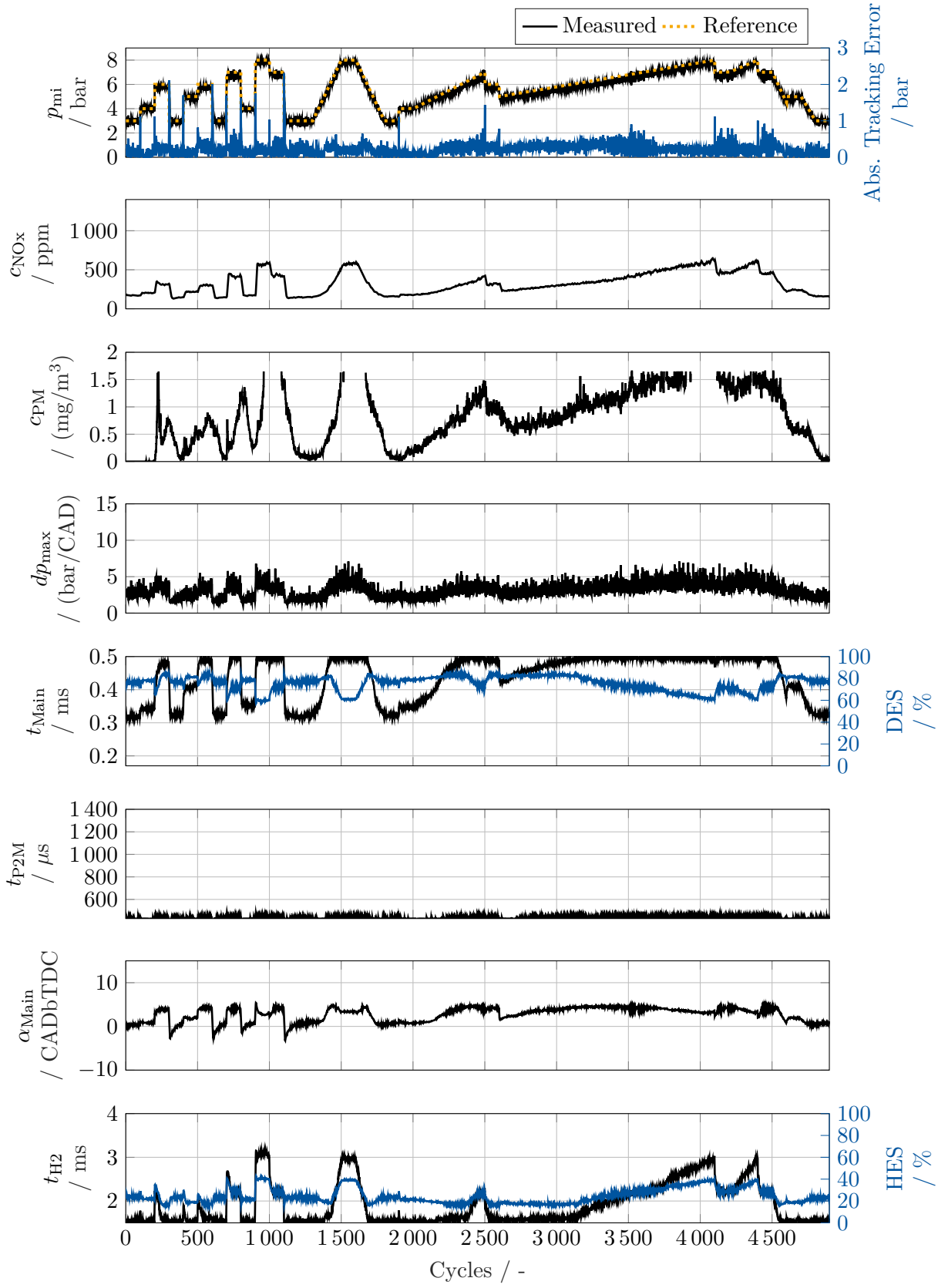


Figure A.15: *Core MPC*. Experimental results, load tracking under model-plant mismatch – high engine speed (1800 min^{-1}).

Behavior Cloning

Table A.14: DNN-based BC controller. DNN training hyperparameter settings.

Hyperparameter	**Value**
Max epoch	5000
Optimizer	Adam
Metric	MSE
Mini-batch size	512
Initial learning rate	0.0005
Learn rate schedule	Piecewise drop by 25% every 250 epochs
L2 regularization	0.1
Validation frequency	10

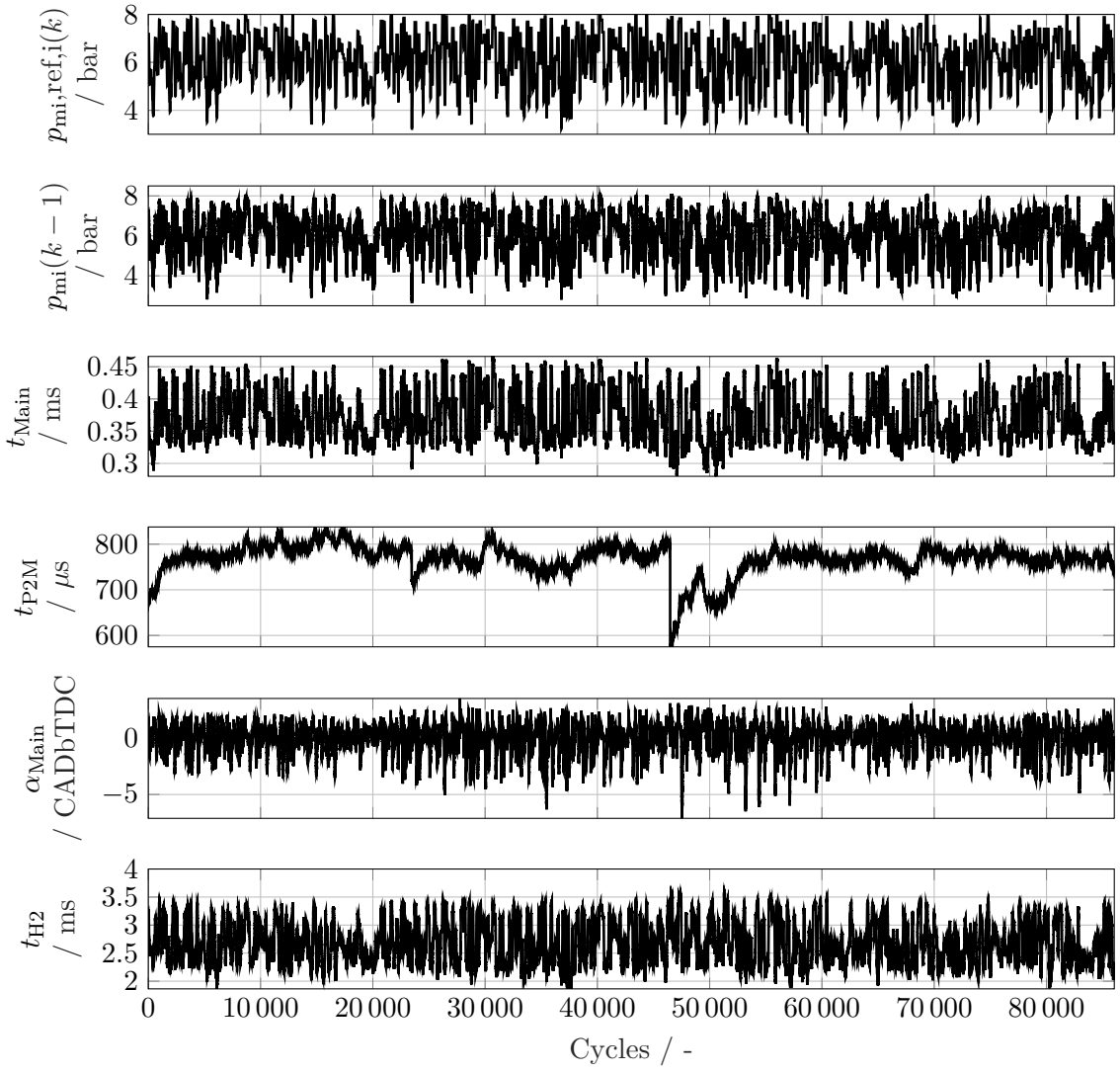


Figure A.16: DNN-based BC controller. Experimental data, engine inputs and outputs in time-domain. Generated by MPC expert policy. 86,000 consecutive engine cycles.

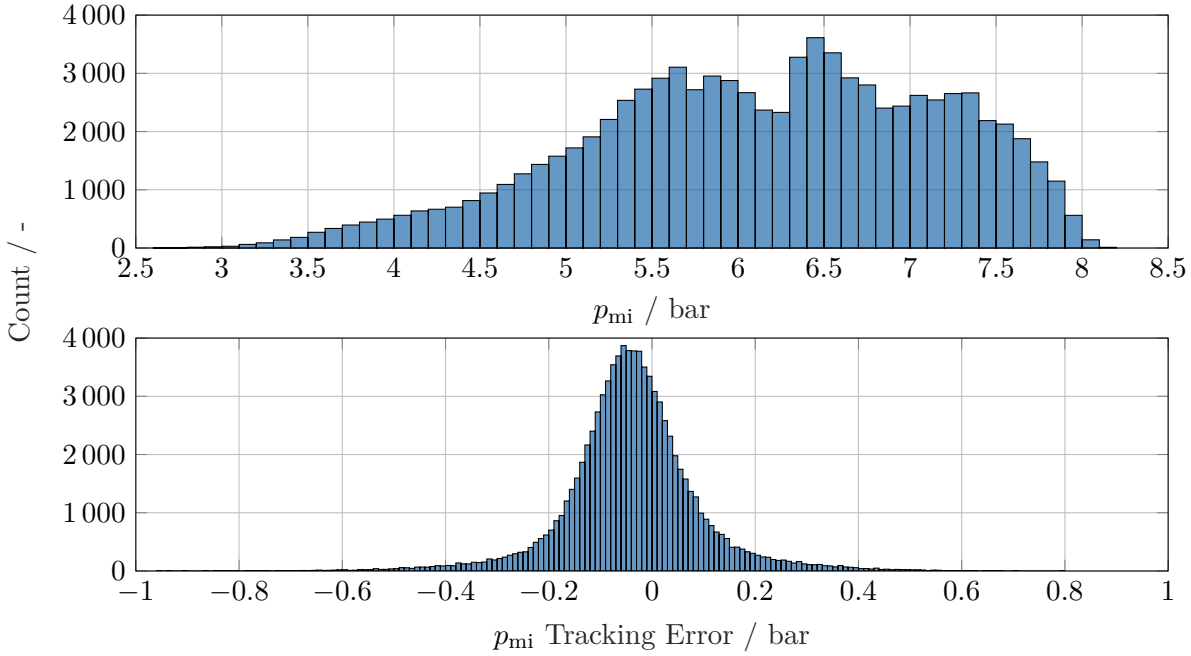


Figure A.17: DNN-based BC controller. Experimental data, total data distribution for DNN outputs (p_{mi} and its tracking error). Generated by MPC expert policy. Total data points: 86,000.

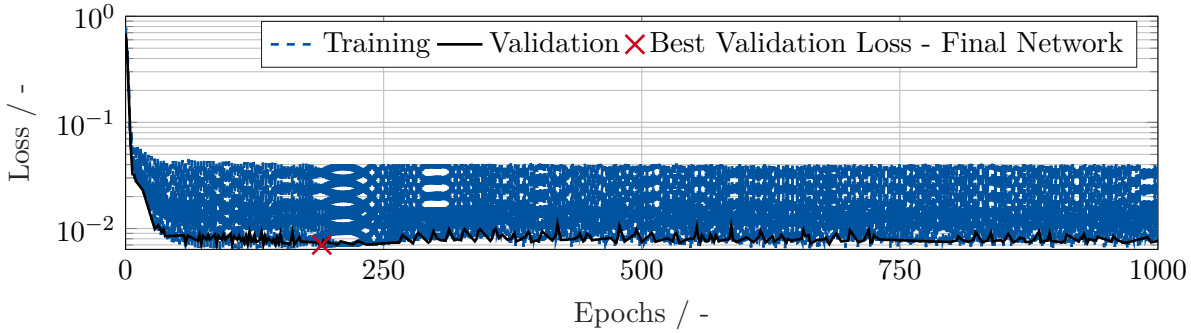


Figure A.18: DNN-based BC controller. Loss-epoch plot of DNN training. The final network is chosen according to the best validation loss.

Table A.15: DNN-based BC controller *no feedback* variant. DNN prediction performance on validation and test datasets.

Metric	**Validation**	**Test**
t_{Main}		
MAE / ms	0.00716	0.00688
RMSE / ms	0.00879	0.00839
NRMSE / –	4.73%	4.51%
t_{P2M}		
MAE / µs	10.52	6.31
RMSE / µs	12.92	7.80
NRMSE / –	4.92%	2.97%

Metric	**Validation**	**Test**
α_{Main}		
MAE / CADbTDC	0.716	0.819
RMSE / CADbTDC	0.959	1.099
NRMSE / –	8.98%	10.29%
t_{H2}		
MAE / ms	0.0466	0.0552
RMSE / ms	0.0598	0.0657
NRMSE / –	3.47%	3.80%

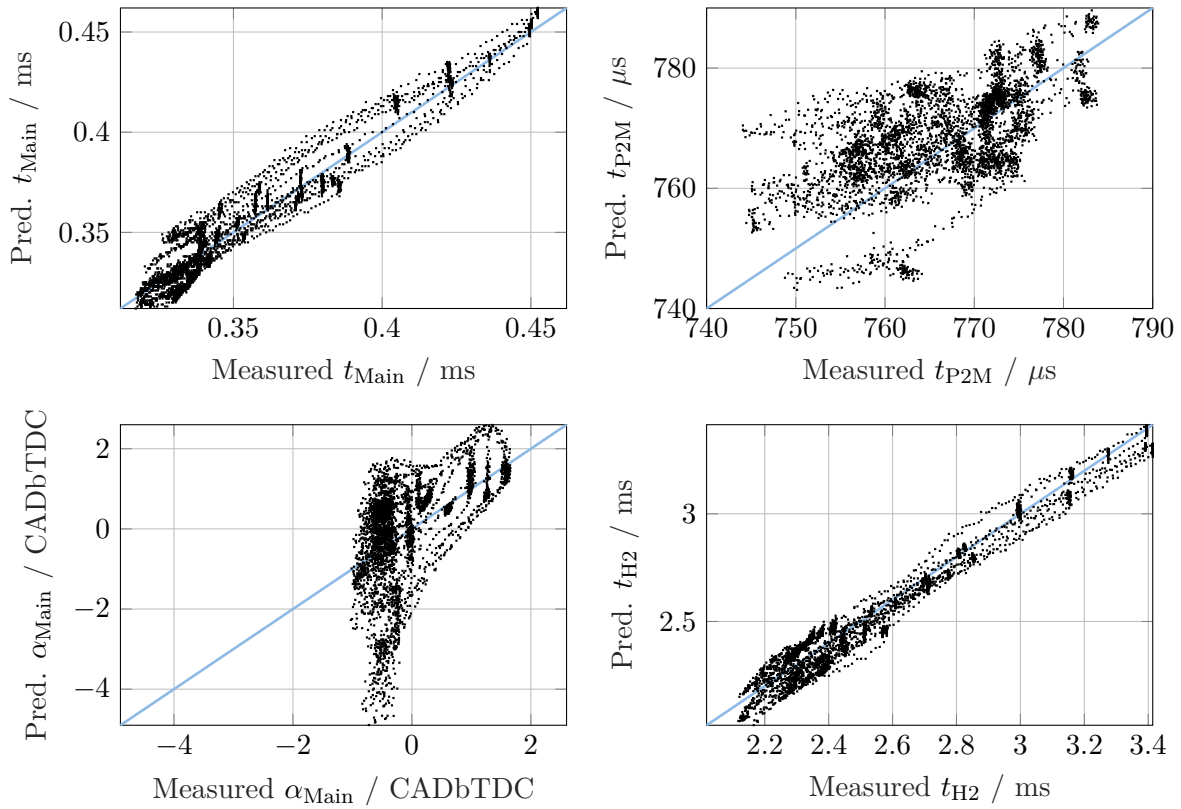


Figure A.19: DNN-based BC controller. DNN predicted outputs vs. true experimental unseen test dataset.

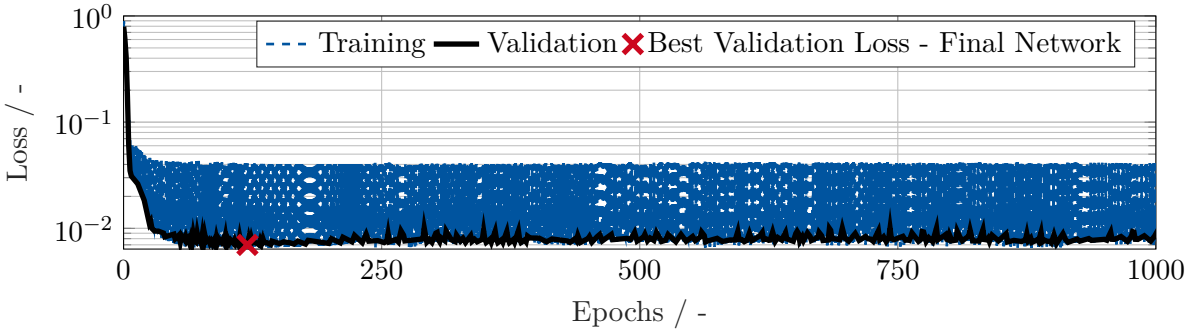


Figure A.20: DNN-based BC controller *no feedback* variant. Loss-epoch plot of DNN training. The final network is chosen according to the best validation loss.

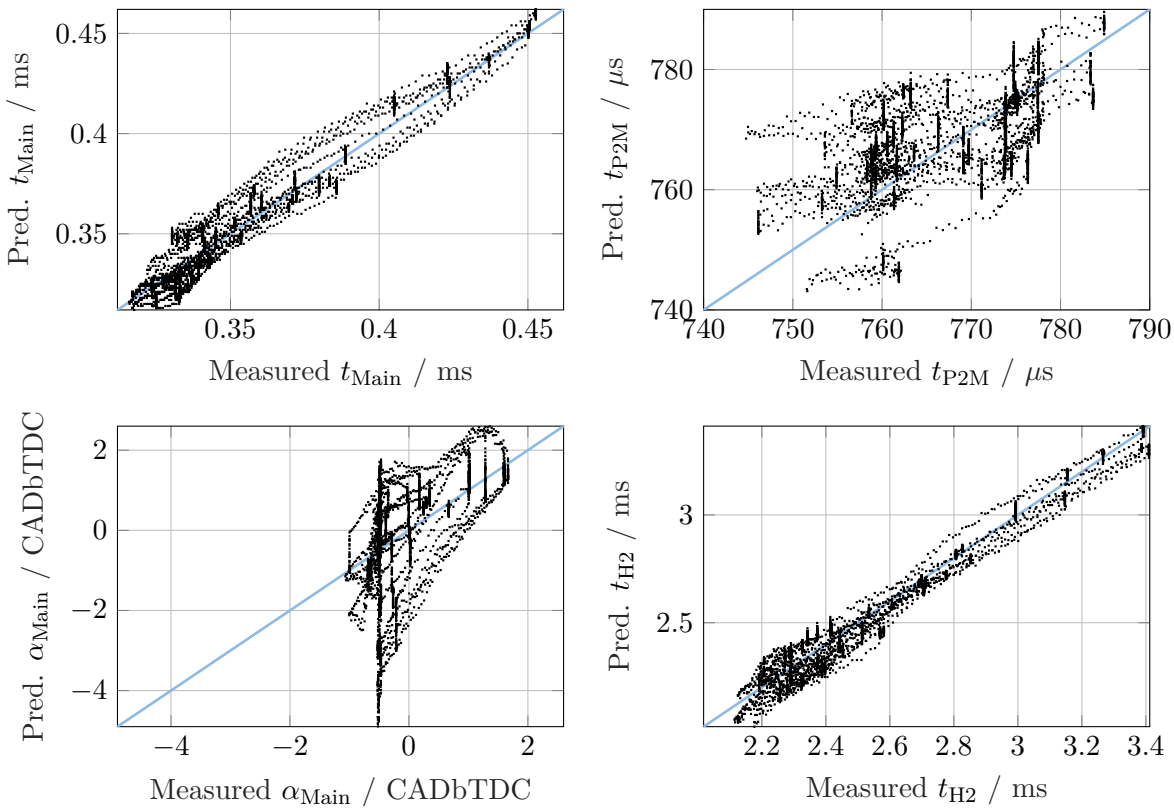


Figure A.21: DNN-based BC controller *no feedback* variant. DNN predicted outputs vs. true experimental unseen test dataset.

Table A.16: *ESP32* (Espressif Systems' 32-bit microcontroller) microcontroller specifications. (CPU: Central Processing Unit; SRAM: Static Random-Access Memory; ROM: Read-Only Memory; RTC: Real-Time Clock; QSPI: Quad Serial Peripheral Interface. From [168]).

Feature	**Specification**
CPU	Dual-core *Xtensa 32 bit LX6* microprocessor
Clock speed	Up to 240 MHz – in this work: 180 MHz
SRAM	520 kB
ROM	448 kB
RTC fast SRAM	8 kB
RTC slow SRAM	8 kB
Flash memory	External, via QSPI interface
Operating voltage	3.3 V

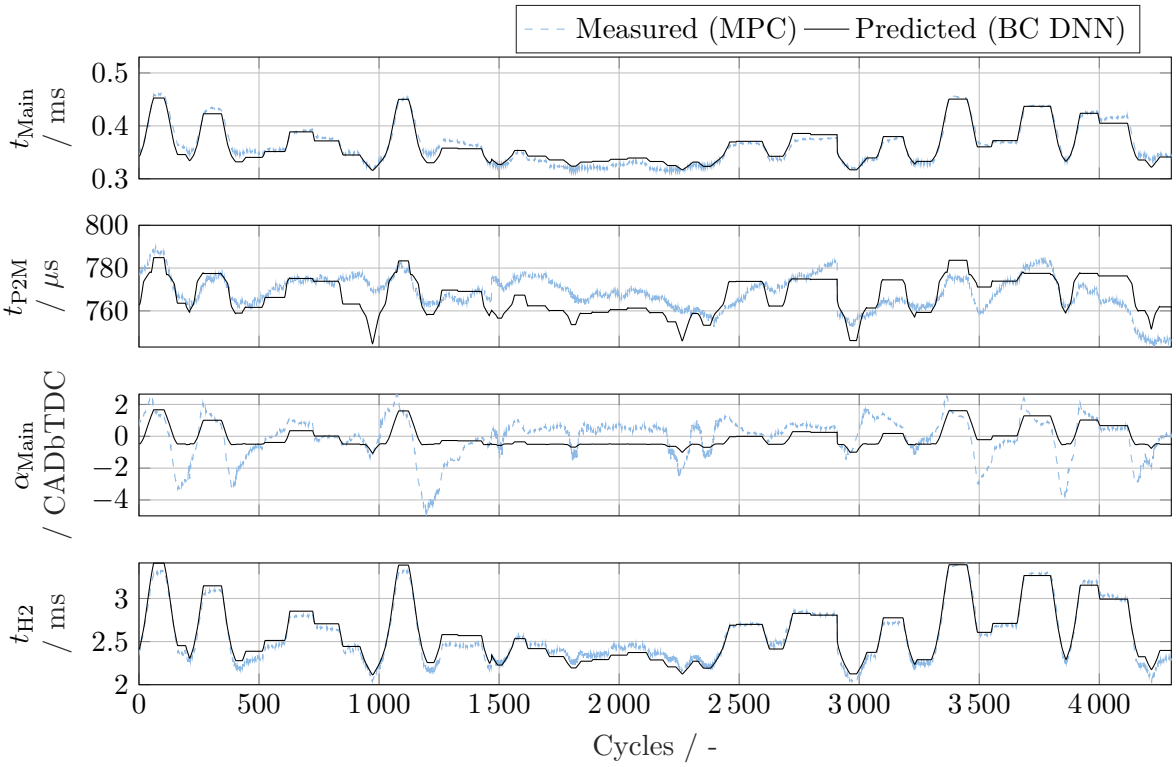


Figure A.22: DNN-based BC controller *no feedback* variant. DNN predicted outputs on the experimental unseen test dataset in time domain.

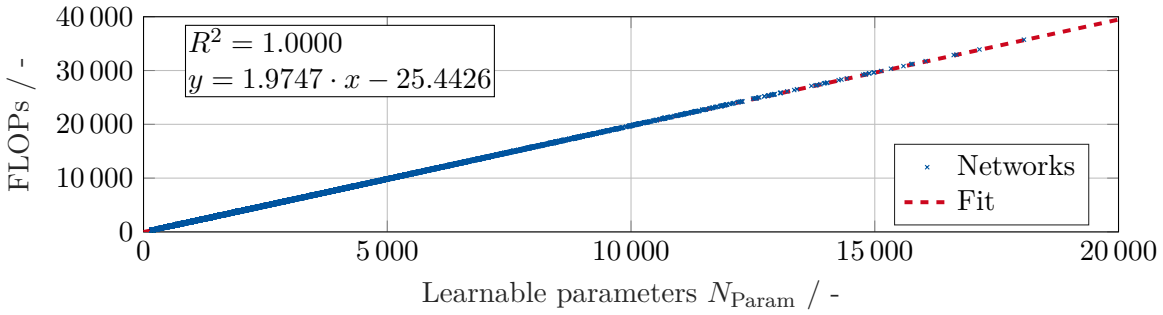


Figure A.23: Linear correlation of FLOPs and learnable parameters of multiple DNNs. All networks compromise of a similar encoder-decoder architecture, varying in number of layers and number of units per layer. FLOPs and learnable parameters are calculated with `TensorFlow Keras` functions.