Machine Learning and Deep Learning for Modeling and Control of Internal Combustion Engines

by

Armin Norouzi Yengeje

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mechanical Engineering
University of Alberta

Abstract

Internal Combustion Engines (ICEs) are ubiquitous; they power a wide range of systems. The broad use of ICEs globally causes more than 20% of the total greenhouse gas emissions. In many countries, emission legislation is transitioning from certification using only traditional chassis dynomometer testing to now requiring the inclusion of Real Driving Emissions (RDE). Complying with this legislation has led to increased challenges to meet emissions levels under on-road use of the engine. The stringent legislation governing emissions and fuel economy, in combination with the complexity of the combustion process, have led to requirements for significantly more advanced engine controllers than are currently used. Reducing the emissions of diesel engines while simultaneously increasing their thermal efficiency through online control optimization and Machine Learning (ML) are the main objectives of this thesis. ML techniques offer powerful solutions that help to address the existing challenges in ICE modeling, control, and optimization. ML can also help to reduce the time, cost, and effort required for ICE calibration for both vehicular and stationary applications.

In this thesis, a four-cylinder medium-duty Cummins diesel engine and emission measurement system including an electrochemical fast Nitrogen Oxides (NO_x) sensor, Pegasor Particle Sensor (PPS-M), and MKS Fourier-Transform Infrared Spectroscopy (FTIR) are used for experimental implementation. A dSPACE MicroAutoBox II, which is a rapid prototyping system, is used for control implementation. In order to compare the proposed control method with the existing Cummins calibrated engine control unit (ECU), all the production calibration tables are imported to the MicroAutoBox. The simulation results presented in this thesis are developed using

a detailed physics-based model using the GT-power© software. A co-simulation of GT-power©/Matlab©/Simulink is used as an Engine Simulation Model (ESM).

The application of ML in engine control can be divided into three main categories: i) ML in emission prediction, ii) Integration of ML and Model Predictive Control (MPC), and iii) ML in the learning-based controller.

In the first category, a correlation-based order reduction algorithm is developed to model NO_x, resulting in a simple and accurate model. This algorithm utilizes Support Vector Machine (SVM) techniques to predict NO_x emission with high accuracy. In addition, a comprehensive study involving eight ML methods and five feature sets is done for Particulate Matter (PM) modeling using gray-box techniques. Then using the K-means clustering algorithm, a systematic way to select the best method for a specific application is proposed.

In the second category, two methods of combining ML and MPC were used: ML-based modeling and ML imitation control. First, ML is used to identify a model for implementation in MPC optimization problems. Additionally, ML can be used to replace MPC, where the ML controller learns the optimal control action by imitating the behavior of the MPC. Using the ESM to provide simulation data, SVM and deep recurrent neural networks, including long-short-term memory (LSTM) layers, are used to develop engine performance and emission models. Then based on these models, MPC is designed and compared to both a linear controller and the Cummins' calibrated ECU model in ESM. Then, a deep learning scheme is deployed to imitate the behavior of the developed controllers. These imitative controllers behave similarly to the online optimization of MPC but require significantly lower computational time. The LSTM-based MPC is then implemented on the real-time system using open-source software. Compared to the stock Cummins ECU, this controller has significant emission reduction, fuel economy improvement, and thermal efficiency.

Reinforcement Learning (RL) and Iterative Learning controller (ILC) are developed to investigate learning-based controllers. Using the ESM, a model-free off-policy

algorithm, Deep Deterministic Policy Gradient (DDPG), is developed. A safety filter is added to the deep RL to avoid damage to the engine. This filter guarantees output and input constraints for both RL and ILC. The developed safe RL is then compared with ILC and LSTM-NMPC.

Preface

This thesis is an original work by Armin Norouzi Yengeje. The research was conducted using available computing systems and the experimental setup of a diesel engine in the University of Alberta advanced combustion engine lab led by Dr. Koch.

Each chapter is fully or partially based on published or submitted papers in peer-reviewed journals and conference procedures [1–9]. The experimental setup, including engine controller reproduction and the emission measurement system setup was prepared in collaboration with David Gordon and Jakub McNally. Engine Simulation Model (ESM) using GT-power© software was developed by Saeid Shahpouri. For the real-time implementation of predictive model control, the embedded programming to compile the controller to `C` code using open-source software called `acados` was done in collaboration with RWTH Aachen University, Germany, and with the support from Alexander Winkler and Eugen Nuss.

I was the lead author of all of the peer-reviewed published papers used to write this thesis except Chapter 3. Chapter 3 entitled "Steady-state Particle Matter (soot) Gray-box Modeling" is based on a peer-review journal paper which Saeid Shahpouri was the lead author. In that work I was responsible for collecting raw data, analyzing and cleaning the data, and developing, optimizing, and deploying different ML methods. Saeid's contributions were developing a physics-based engine model, a physics-based PM model for comparison purpose with ML-based models, and providing gray-box and physical insight feature sets.

To victims of flight PS752...

Acknowledgments

First and foremost, I would like to express my genuine appreciation to Dr. Bob Koch for his overarching guidance, encouragement, and support throughout my study. Your advice always puts things in perspective, and I am deeply indebted to you.

I am also immensely grateful to Dr. Mahdi Shahbakhti for his valuable advice and support. I would like to thank Dr. Jakob Andert for his valuable feedback for our joint publications. I also thank Dr. Hoseinali Borhan, Dr. Lisa A Farrell and Cummins Research and Technology (R&T) for support and insightful discussions that contributed to the quality of the thesis.

Thanks to all of my friends and colleagues in both the University of Alberta and RWTH Aachen University labs that helped me in the last four years with their kind support. I would like to especially thank David Gordon and Masoud Aliramezani, two of my labmates who helped me conduct experiments. I also thank Saied Shahpouri, Alexander Winkler, Jakub McNally, and Hamed Heidarifar for their contributions to my projects.

To my family and friends who have supported me on my journey through my studies, you have my sincerest appreciation. To my mother, Sima, and my father, Reza, my sister Anita, thank you for everything you have done to support me in my life. To my aunt, Nasrin, and my grandmother, Fatemeh, for all their support to help me study abroad. I would like to thank my fiancée, Sara, for her unparalleled patience, consideration, and tremendous support.

Table of Contents

List of Tables

List of Figures

Abbreviations and Acronyms

aTDC	after Top Dead Center
AFR	Air-Fuel Ratio
AOC	Ammonia Oxidation Catalyst
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARX	AutoRegressive EXogenous
bTDC	before Top Dead Center
BM	Benchmark
BNN	Beysian Neural Network
BB	Black-Box
BMEP	Break Mean Effective Pressure
CO2	Carbon Dioxide
CO	Carbon Monoxide
CNG	Compressed Natural Gas
CI	Compression Ignition
CFD	Computational Fluid Dynamics

CAN Controller Area Network

COM Control-Oriented Model

DDPG Deep Deterministic Policy Gradient

Deep RL Deep Reinforcement Learning

DPG Deterministic Policy Gradient

DOC Diesel Oxidation Catalyst

DPF Diesel Particulate Filter

DOI Duration of Injection

ECU Engine Control Unit

ESM Engine Simulation Model

ERT Ensemble of Regression Trees

EGR Exhaust Gas Recirculation

EDA Exploratory Data Analysis

ELM Extreme Learning Machines

FS Feature Selection

FPGA Field Programmable Gate Arrays

FTIR Fourier-Transform Infrared Spectroscopy

FQ Fuel Quantity

FOM Full-Order Model

FC Fully Connected Layer

GPR Gaussian Process Regression

GA Genetic Algorithm

GB Gray-Box

HCCI Homogeneous Charge Compression Ignition

IMEP Indicated Mean Effective Pressure

ICE Internal Combustion Engines

ILC Iterative Learning Control

LASSO Least Absolute Shrinkage and Selection Operator

SVM-LPV Least-square Sup-port Vector Machine based Linear Parameter-Varying

LS-SVM Least-Square Support Vector Machine

LQR Linear Quadratic Regulator

LPV Linear Parameter Varying

LQG Linear Quadratic Gaussian

LQG Linear Quadratic Gaussian

LQR Linear Quadratic Regulator

LSTM Long-Short Term Memory

LOM Low-Order Model

ML Machine Learning

MPRR Maximum Pressure Rise Rate

MSE Mean Squared Error

MOR Model Order Reduction

MPC Model Predictive Control

NOx Nitrogen Oxides

NARX Nonlinear AutoRegressive EXogenous

NMPC Nonlinear Model Predictive Control

NMPC Nonlinear Model Predictive Control

NRMSE Normalized Root Mean Square of Error

PSO Particle Swarm Optimization

PM Particulate matter

PPS Pegasor Particle Sensor

PHY Physical insight features

PID Proportional Integral Derivative

RDE Real Driving Emissions

ReLU Rectified Linear Unit

RT Regression Tree

RL Reinforcement Learning

RVM Relevance Vector Machine

RMSE Root Mean Square of Error

SCR Selective Catalytic Reduction Catalyst

SMC Sliding Model Controller

SMC Sliding Model Controller

SI Spark Ignition

SOI Start of Injection

SVM Support Vector Machine

UHC Unburnt Hydrocarbons

VGT Variable Geometric Turbine

PART I: Introduction and Experimental setup

Chapter 1

Introduction and Background [1]

Global energy consumption is increasing due to rapid population growth and economic development caused by the industrial revolution [10, 11]. Figure 1.1 presents the energy consumption projection until 2040. As shown, energy utilization from fossil fuel in the transportation sector is rising [12]. Current transportation, such as mass public transportation and millions of personal vehicles, has enabled society to reach a high standard of living. Heavy-duty and medium-duty diesel engines are needed for both transportation and power generation. Diesel engines offer the critical advantages of high efficiency, fuel economy at full-load and part-load conditions, and a long lifetime [13]. Based on the latest transportation energy data, the proportion of diesel fuel usage has increased over the last 30 years on national highways in the United States (US) [14]. This trend shows the importance of heavy-duty applications of diesel engines. Natural Resources Canada reports that of the 69% of primary energy converted to secondary energy, 21% is devoted to the transportation sector. Of this amount, gasoline, diesel fuel oil, and aviation turbo fuels contribute the most: 58%, 28%, and 10% respectively [14].

Although diesel engines have many advantages and are used widely in transportation and power generation, they play a crucial role in contributing to environmental pollution problems worldwide. Diesel engine exhaust emissions are a main contributor to environmental pollution and several health problems [15]. Diesel engine emissions

[1] This chapter are partially based on [1, 2]

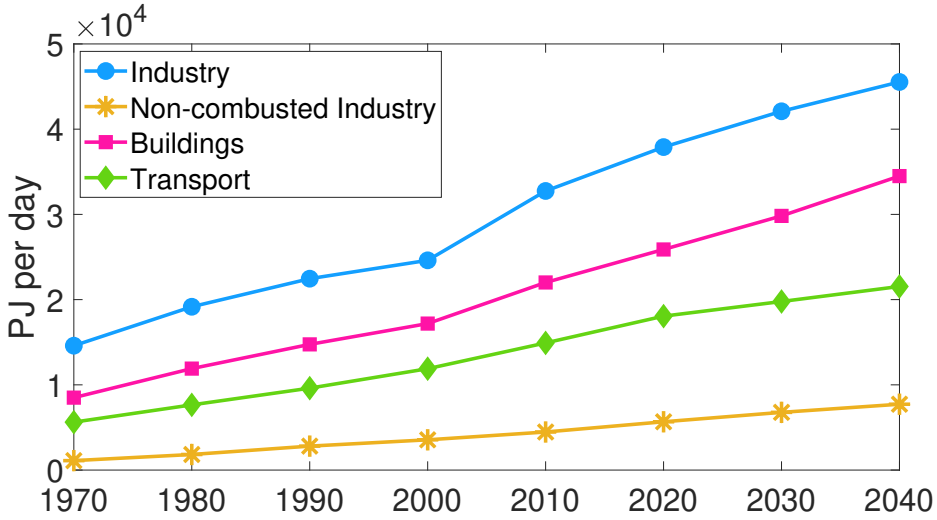


Figure 1.1: Primary energy consumption reshaped based on [12]

can be summarized as

- **Nitrogen oxides (NO_x):** The high combustion temperature and lean air-fuel mixture of diesel engines leads to relatively high NO_x emissions. The NO_x emissions in diesel engines mainly consist of Nitrogen monoxide (NO) and Nitrogen dioxide (NO_2). Typically, engine exhaust contains 70%-90% NO and 10%-30% NO_2 [16].

- **Particulate matter (PM) or soot:** PMs are complex structures formed by soot, hydrocarbons (resulting from fuel and lubrication), and other minor materials [17]. PM emissions and NO_x emissions usually vary inversely [18].

- **Carbon dioxide (CO_2):** CO_2 formation is proportional to fuel consumption for hydrocarbon fuel. Apart from the CO_2 emission regulations, CO_2 emission limits are also driven by user demand for fuel economy [19].

- **Carbon monoxide (CO):** CO is a colorless, odorless, non-irritating, but highly toxic gas which is a sub-product of combustion [20]. Diesel engines typically operate at lean conditions (higher air-fuel ratio than the stoichiometric air-fuel ratio). Therefore, CO emission is a less critical emission than NO_x and

PM, although it is still an important emission from a diesel engine [15].

- **Unburned Hydrocarbon (UHC):** UHCs are a product of incomplete combustion of the injected fuel due to low temperature or locally rich conditions inside the cylinder [21].

Air pollution can cause respiratory diseases and health complications. NO_x emissions are one of the main causes of edema, pneumonia, lung irritation, and bronchitis. They also affect asthma patients. CO emissions have adverse effects on fetal growth and on tissue development in young children. UHC contributes to circulatory or respiratory problems. In developing counties, PM has become the biggest contributor to poor air quality, particularly in large urban centers [22].

The widespread use of diesel engine in transportation and power generation has increased the global pollution level. It is predicted that by the end of 2035, the global CO_2 emission level will rise by 29% [22]. According to the International Energy Agency, the transportation sector's CO_2 emission grew by 92% from 1990 to 2020. The Agency predicts that between 2020 and 2035 [22] 8.6 billion metric tons will be emitted. Similarly, in Canada, from 2000 to 2017, the CO_2 emission level increased by 19.2% with freight trucks contributing 44% of the increase [14]. Strict emission regulations have led to a steady increase in the use of hybridization and electrification for passenger vehicles. However, the trend is not likely to carry over in near future to heavy-duty trucks due to the high battery costs and total-cost-ownership [23]. According to the recent studies [14, 23], in the best scenario, battery-electric commercial vehicles could reach 8 to 27 percent sales of medium duty e-trucks by 2030 [14, 23].

Real Driving Emissions (RDE) testing has been implemented by the New Euro 6d regulations as an additional requirement. RDE test results are significantly affected by ambient conditions, traffic, and driver behavior. According to RDE legislation, engines must operate cleanly under all conditions. This makes engine design and calibration much more challenging [24]. Complying with RDE legislation is such a

large shift from previous legislation. Therefore, with emission legislation becoming ever more stringent, intelligent engine emission control strategies that take advantage of Artificial Intelligence (AI) progress and using optimal control as well as advanced after-treatment systems are needed to meet the rigorous emission regulations. Taking advantage of AI approaches gives the system the ability to modify engine calibration during real driving by adapting controller based on the engine operating condition. This is crucial in RDE legislation compliance.

This chapter will examine systems and techniques for emission reduction as described in the literature. Then, the motivations and the organization of this thesis will be presented.

1.1 Emission Reduction Technologies

1.1.1 Diesel exhaust aftertreatment systems

A wide range of research has been conducted on diesel engine emission reduction using exhaust after-treatment systems, and by using alternative fuels with or without additives. The primary after-treatment systems which significantly decrease tail-pipe emissions are:

- **Diesel Particulate Filter (DPF):** DPFs are used to trap particulate matter (PM) from the exhaust gas to increase the reactivity of the trapped particles during DPF regeneration [25]. They are capable of removing more than 90% of the PMs [26]. DPF physically filters the PMs, which consequently increases the pressure drop over the filter. Increasing the pressure drop raises the engine backpressure and reduces the engine's thermal efficiency [27]. To compensate for this effect a periodic DPF regeneration process is done.

- **Diesel Oxidation Catalyst (DOC):** DOCs are used to oxidize CO and UHC. The DOC also regulates the NO/NO_2 ratio in the exhaust gas [28].

- **Selective Catalytic Reduction Catalyst (SCR):** Urea-based Selective Catalytic Reduction (SCR) is an effective technique to reduce NO_x emissions and may satisfy future emission regulations [29]. The urea $C(NH_2)_2O$ is injected into the SCR catalyst and converted to NH_3 and CO_2 in the SCR. The ammonia reacts with NO_x and produces N_2 and H_2O in the SCR.

Figure 1.2 shows the schematics of a typical diesel after-treatment system. Gases coming from the engine cylinders pass through the DOC and DPF where CO, unburned hydrocarbons, and soot are reduced. They then flow to the SCR subsystem, which decreases NO_x by using upstream injections of urea. This fluid breaks down to produce ammonia, which reduces or removes NO_x in the SCR. Then to oxidize excess ammonia before it leaves the tailpipe, the ammonia Oxidation Catalyst (AOC) is used [30].

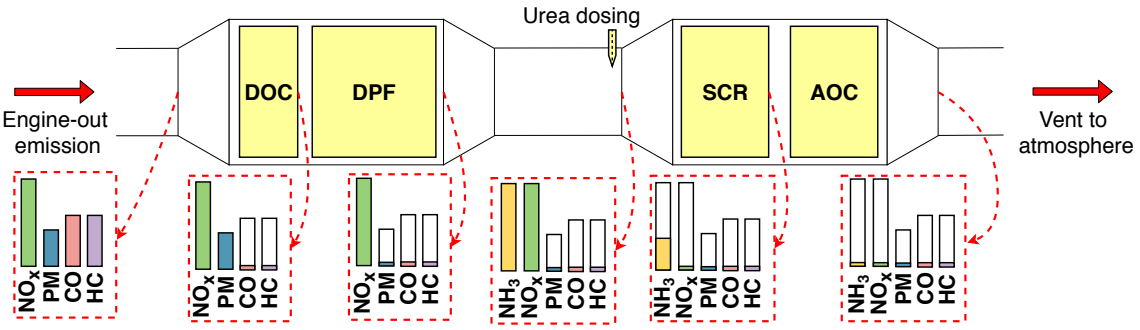


Figure 1.2: Schematics of a typical diesel after-treatment system- based on [30]

These after-treatment devices have significantly reduced tailpipe emissions. However, missing an intelligent control system based on the engine-out emission feedback makes the calibration process complex. RDE requirements mean that meeting increasingly stringent legislations is even more difficult [31]. Next, engine Diesel exhaust feedback control using optimal control strategies will be discussed.

1.1.2 Diesel exhaust feedback optimal control strategies

In automotive applications, especially in Internal Combustion Engines (ICEs) control, feedback control is often combined with feedforward control to deal with the influ-

ence of varying operating points. One of the most common techniques to design a feedforward controller is two-dimensional look-up tables—so-called calibration maps. The feedforward controller enables fast changes in operating points, while the feedback controller performs the error compensation. The most conventional feedback controller in ICEs is a Proportional Integral Derivative (PID) controller.

The gains of PID controllers are tuned using the procedure of parameter optimization and fine tuning using the trial-and-error method. The optimization process, also known as engine calibration, results in finding look-up table values and controller gains [32, 33]. Due to increasing requirements for low-fuel consumption and emissions, the number of control inputs have increased substantially, making manual test-bench calibration difficult and time-consuming.

An ideal solution is systematic optimization based on a simulation model developed and identified using experimental data. Several model-based controllers have been used in engine feedback control to address this, such as the Linear Quadratic Regulator (LQR) controller [34], Linear Quadratic Gaussian (LQG) controller [35], Sliding Model Controller (SMC) [36–38], Adaptive [38], and Model Predictive Control (MPC) controller [39, 40].

Among these model-based controllers, MPC is one of the most promising for dealing with the highly constrained nonlinear system of ICEs. MPC can provide an optimal real-time solution for meeting multi-objective goals while addressing system and operational constraints. New variants of MPC utilize optimization solvers and packages that are suitable for the real-time operation of time-critical systems [39, 40].

MPC, a control technique that has been increasingly used in industry during the past four decades, has the following five main advantages: (1) it implicitly considers constraints on state, input, and output variables, (2) it provides closed loop control performance and stability for the optimal problem with constraints, (3) it exploits the use of a future horizon while optimizing the current control law, (4) it offers the possibility of both offline and real-time implementations, and (5) it is capable

of handling uncertainty in the system's parameters, delays, and non-linearity in the model [41].

Model-based engine control techniques have been applied to ICEs for more than five decades [42]; however, conventional MPC techniques have been applied for ICE applications only over the past 23 years. Two examples of early MPCs on ICEs include: (i) the Air-Fuel Ratio (AFR) control of an SI gasoline engine using a linear AFR model by linear approximation of a neural network model [43] in 1998, and (ii) the idle speed control of an SI gasoline engine using a linear model by applying system identification techniques on GT-Power© engine model simulations [44] in 1999.

These early works were done in simulation environments, while recent work [45] includes the experimental implementation of a nonlinear multi-objective MPC on a real engine. MPC implementation for ICEs control is becoming increasingly common [39, 45–73]. The integration of ML and MPC is an emerging area that provides additional opportunities to control and optimize of ICEs.

1.2 Model Predictive Control (MPC) Background

The idea of using MPC began in the 1960s [74]; however, the first reported application of MPC in industry was in 1978 [75]. After initial application in the late 80s, MPC usage grew rapidly in several industries. In particular, the process industry was an early adopter of MPC as MPC was able to handle both input constraints and states constraints. Some of these processes were slow enough to allow MPC implementations with the processors of that time.

A survey in 1997 estimated 2233 applications of MPC from five different vendors [76]. A graphical depiction of MPC development and implementation is shown in Figure 1.3. Increased interest in MPC stability and robustness started in the early 1990s [76]. At the same time, multiple algorithms were developed to control the nonlinear systems using MPC [77]. Starting around the year 2000, new approaches began to be developed. Among these are the hybrid MPC, which considers both continuous

and discrete variables [78], and the explicit MPC [79].

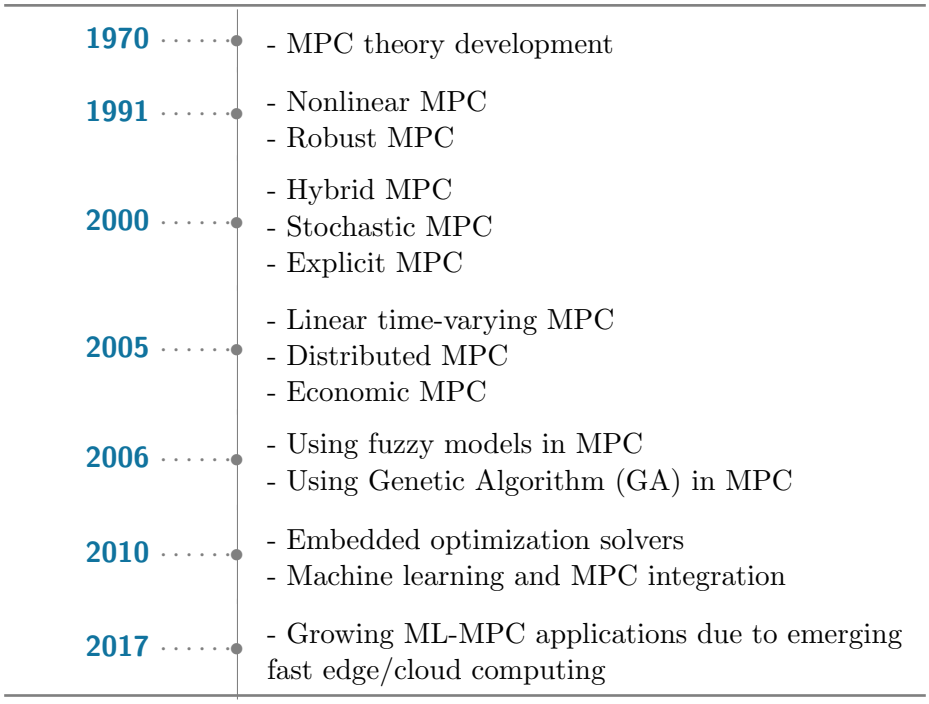


Figure 1.3: Timeline of major Theoretical MPC developments

MPC uses a receding horizon to minimize cost function to calculate optimal control inputs for a finite control horizon. Over the finite prediction horizon, the cost is minimized with respect to the system's dynamics, current states, and constraints. From these calculated control inputs typically, only the first step is applied to control the system output. Then, for the next time interval, MPC repeats the same process [45].

A schematic about the MPC operation in an ICE application is depicted graphically in Figure 1.4 where the Indicated Mean Effective Pressure (IMEP) is controlled using the injection fuel quantity as a control variable. In this figure, H_u is the control horizon, and H_p is the prediction horizon. The prediction horizon is longer than the control horizon with larger computational costs for longer horizons [80].

The control horizon depends on how fast the reference input and disturbance change. For example, in ICEs, the control horizon could be one cycle for an ICE

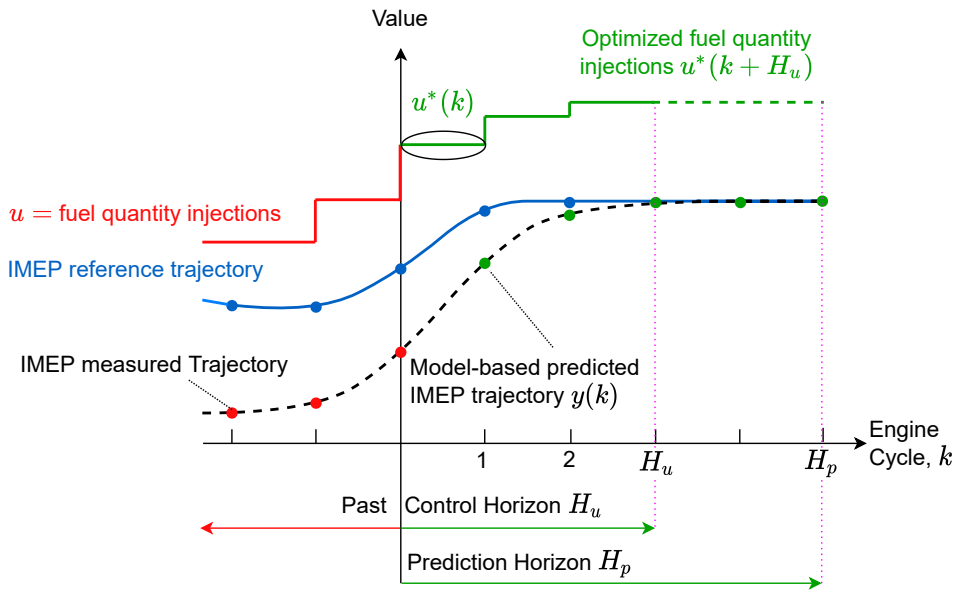


Figure 1.4: The MPC control concept and prediction receding horizon: illustrated for engine load control.

in a highly transient operation in a conventional vehicle and could be three or more engine cycles for operation in a hybrid electric vehicle for an operating when the ICE is decoupled from the road load conditions. The MPC formulation can be defined as:

$$\begin{aligned} \min_{u_0,\ldots,u_{N-1}} \quad & J_f\left(x_N\right) + \sum_{k=0}^{N-1} J(x_k,\ y_k,\ r_k,\ u_k,\ s_k) \\ \text{s.t.} \quad & x_{k+1} = f(x_k,\ u_k,\ d_k), \quad y_k = g(x_k,\ u_k,\ d_k) \quad k \in \mathbb{N}_0^{N-1} \\ & x_k \in \mathcal{X}, \quad u_k \in \mathcal{U} \quad k \in \mathbb{N}_0^{N-1} \\ & X_N \in \mathcal{X}_f \quad x_0 = x(t) \end{aligned} \tag{1.1}$$

where x, y, u, r, s, d, and N represent states, outputs, inputs, references, slack variables, disturbances, and the prediction horizon, respectively. In this equation, J, f, g, $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{X}_f$ represent the state function, output function, cost function, state constraint set, input constraint set, and terminal state constraint set, respectively. Next, the application of ML in ICE modeling and control will be discussed.

1.3 Application of Machine Learning in Internal Combustion Engines

In general, ML can be divided into three main categories: i) supervised learning, ii) unsupervised learning, and iii) Reinforcement Learning (RL). These methods are used widely in different strategies to model and control ICEs. Table 1.1 presents the categorization of the ML approaches specifically used in internal engine modeling and control.

Table 1.1: Machine learning in Diesel Engine Emission Modeling and Control– ANN: Artificial Neural Network; SVM: Support Vector Machine; RVM: Relevance Vector Machine; GPR: Gaussian Process Regression; ELM: Extreme Learning Machine; BNN: Bayesian Neural Network

Machine Learning	**Application**	**Methods**
Supervised learning	Steady-state models for estimation	ANN [81–89] ELM [90–92] SVM [89, 93–99] RVM [100] GPR [101]
	Dynamics modeling for control	ANN [43, 102–105] BNN [106] ELM [107, 108] LS-SVM [39, 40, 109, 110]
Unsupervised learning	Misfire/knock, and component fault detection	ANN [111] K-means [111, 112] Fuzzy C-means [113, 114]
Reinforcement learning	Pure learning	Actor-critic [115, 116] Q-learning [117–120]
	Tuning controller	RL for PID tuning [121]

1.3.1 Supervised Machine Learning

Supervised ML is usually used in diesel engine performance and emission modeling. The Artificial Neural Network (ANN) is a method that is widely applied to ICEs. ANN was used to develop exhaust emission and performance modeling of CNG-diesel engines [81], diesel engine coupled with Exhaust Gas Recirculation (EGR) [82], a hydrogen dual fuel diesel engine with biodiesel blends [83], a mixed biodiesel blends [84], a dual fuel diesel engine [85], hydrogen-enriched diesel engine [86], diethyl ether fueled single-cylinder diesel engine [87], and a diesel engine fueled with animal fat [88]. ANN uses the gradient descent algorithm for training, which increases the risk of converging to local minima. Additionally, the risk of overfitting is higher for ANN for the same size of training data with respect to the other methods, such as Support Vector Machines (SVM) [122]. Another ML method used in diesel engine modeling is Extreme Learning Machines (ELM). ELM is usually used as a single hidden layer feedforward network [123]. ELM has been used to predict engine performance and exhaust emissions [90], and the exergetic performance prediction of diesel engines [91], and to model and optimize biodiesel engine performance [92].

The Support Vector Machine (SVM) is another popular data-driven method that is now being increasingly used for modeling internal combustion engines mostly for the steady-state prediction of engine performance and emissions. The SVM is an ML approach that is capable of modeling complex and non-linear input-output relations based on a sufficiently large training data set [124–126]. This approach provides a black box model without directly involving a physical understanding of the system but can be accurately trained if the model's features are selected appropriately [11, 127, 128]. A NO_x prediction model was developed for a hydrogen-enriched compressed natural gas engine using an optimal SVM method where Particle Swarm Optimization (PSO) was used to find the regulatory parameters of SVM [94]. Studies have also examined the effect of SVM model parameters such as the penalty factor kernel,

insensitive band loss function, and the training sample size [94]. An optimal SVM for the diesel engine's NO_x prediction was developed in [93], in which the Genetic Algorithm (GA) was used to find the regulatory parameter of SVM. The SVM approach has been used for the exergetic modeling of diesel engines [96]; and to predict diesel engine performance, and emission [99], hydrogen-enriched compressed natural gas engine performance at specific conditions [98], and the performance and exhaust emission of marine diesel engines [97]. Also, the least-square version of the SVM was used to model and optimize engine performance fueled with biofuel [95].

Another approach used in modeling the performance and emission of diesel engines is the Relevance Vector Machine (RVM). Its functional form is identical to that of the SVM but it provides a probabilistic regression [129]. A diesel engine performance and exhaust emission model was developed using the RVM and compared with the conventional ANN model, which showed that the RVM is superior to the typical ANN approach [100].

Gaussian Process Regression (GPR) is another method in engine performance and emission modeling which is a nonparametric and Bayesian-based approach that has superior performance with small data sets and can provide an uncertainty measure on the predictions [130]. The main advantage of GPR is probabilistic prediction. Unlike other supervised ML methods, GPR infers a probability distribution over all possible ML model parameter values. A combination of a 1D-CFD model and a GPR ML method with a fixed input feature set was used in [101] for emission modeling including NO_x and soot emissions. The GPR was also used in black-box NO_x and soot [131–133], CO [132] modeling of diesel engines.

The requirement for accurate modeling to guarantee MPC performance while simultaneously having a simpler model has created an opportunity to utilize the ML method in developing required models in MPC platforms. In ML-based MPC, an ML-based model is used to develop a predictive dynamics model. This model is used directly to design MPC or implement optimization. Several ML-based data-

driven modeling techniques have been used to model dynamics of system, including ANN [102, 103], ELM [107, 108], Bayesian Neural Network (BNN) [106], and Least-Square SVM (LS-SVM) [39, 40] to provide a predictive dynamics model of sufficient accuracy for model-based control of ICEs.

Among all data-driven transient modeling using ML, ANN is the most common in ICEs. Adding shallow networks (low number of hidden layers) could be useful as an accurate function approximation for a classical time-series system identification technique, Nonlinear AutoRegressive eXogenous (NARX), to identify a dynamic system in the literature [43, 102]. This model, in general, is nonlinear and usually requires nonlinear programming for MPC implementation. Alternatively, by linearizing NARX, linear MPC can be used [43, 108]. ELM was combined with Model Predictive Control (MPC) to provide a model for both offline and online learning. An Homogeneous Charge Compression Ignition (HCCI) engine was modeled using ELM, and then nonlinear MPC was used to design Indicated Mean Effective Pressure (IMEP) and stability control [108].

1.3.2 Unsupervised Machine Learning

Unsupervised ML is mostly used for Fault Diagnosis (FD) of diesel engines. Different unsupervised clustering algorithms have been used for diesel engines FD, such as ANN clustering, K-means, and fuzzy C-means. The K-means algorithm specifies k number of centroids, and assigns each data point to the closest cluster while keeping the centroids as small as possible [134]. K-Means algorithm was used to organize potential engine faults of diesel marine engines [112]. Also, K-means and ANN clustering methods were compared for fault diagnosis on a main engine journal-bearing [111]. A fuzzy C-means clustering algorithm, which is more efficient than the K-means algorithm, has been used in diesel engines. In the K-means algorithm, each data point is allocated to one cluster. However, in the C-means algorithm, which is a fuzzy clustering technique, each data point is allocated to all of the clusters with a

membership degree. This algorithm was used in predictive modeling of marine engine performance [113] and fault diagnosis of diesel engine vibration signals [114].

A least-squares version of SVM was used to solve a set of linear equations to lower the computational cost for constrained optimization programming. Both regression SVM and least-squares SVM, so-called LS-SVM, have been used to provide ICE dynamics models for MPC. The Linear Parameter Varying (LPV) formulation of a Reactivity Controlled Compression Ignition (RCCI) model for CA50 and engine load control is driven based on the LS-SVM in [39].

1.3.3 Reinforcement Learning (RL)

Unlike the supervised and unsupervised learning that uses measurement data, RL works dynamically by interacting with system (environment) data. In RL, the goal is to neither cluster or label the data, but to generate the optimal outcome by finding the best sequence of actions. RL solves this problem by allowing a piece of software called an agent to explore, interact with, and learn from the environment. RL has a similar structure to the traditional control, and Figure 1.5 schematically presents this similarity. With both methods, we want to determine the correct inputs into a system that would generate the desired system's behavior. The controller is tuned using a tuning algorithm or adaptation law, while in RL policy updates are based on the RL algorithm [135]. Different kinds of RL algorithms have been developed. However, this ML approach has seldom been applied to a diesel engine control. One common algorithm used for model-free RL is Q-learning. In Q-learning, the value of an action for a particular state is learned and the optimal policy is found by maximizing the expected value (Q-value) of the total reward [136]. Q-learning has been used in the diesel engine control auxiliary power network of [117] a marine diesel engine. Q-learning RL has also been used as the idle speed control of a spark ignition engine by controlling the spark timing and intake throttle valve position [118]. Similar studies have been carried-out for diesel engine idle speed control by controlling fuel injection

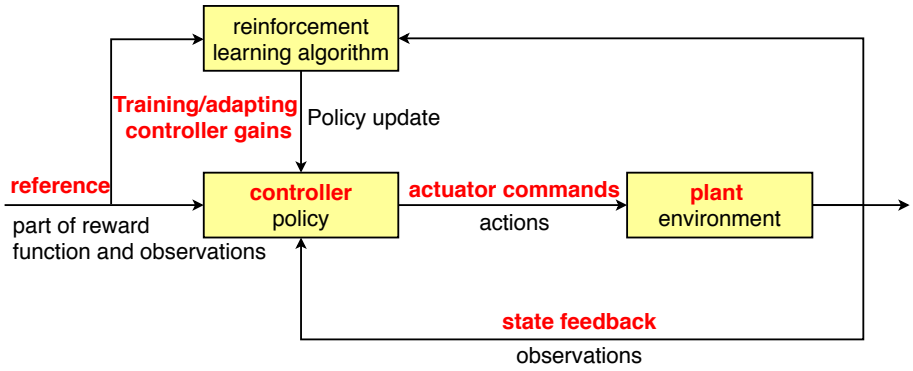


Figure 1.5: Reinforcement learning (RL) analogy to traditional controls based on [135] - Red is used to show traditional control and black is used to show RL

timing [119].

When an agent performs an action that has the highest reward without further exploring the environmental space it is considered a greedy policy. In continuous spaces, using a greedy policy to optimize the action at each time interval is extremely slow. Often, it is not possible to apply Q-learning easily to continuous action systems. However, an actor-critic method based on the Deterministic Policy Gradient (DPG) algorithm is a suitable choice for a system with a continuous space [137]. An actor-critic method using a neural network has also been used for the emission control of spark ignition engines [115, 116].

Although RL is now receiving attention from the control system community, a learning controller is not a new concept [138, 139]. One of the well-known learning-based controllers is Iterative learning control (ILC), which is used to improve the tracking performance of a system in the presence of repetitive input or disturbances [140, 141]. ILC was first introduced in 1984 [139] and since then has been used for various control problems. ILC has a simple structure and is computationally efficient for real-time applications and can have stability guarantees. Different types of ILC have been implemented for internal combustion engine control. ILC has been used in SI engine load control [142, 143], dual-fuel control of an HCCI engine [144], SI engine speed and air-to-fuel ratio [145], parameter optimization in Turbocharged

SI engine [146], variable injection rate control for CI engines [147], and EGR control in CI engine [148].

So far, existing technologies of diesel emission reduction have been introduced in Section 1.1 and the application of ML in ICE has been identified in Section 1.3. Next, using this knowledge, the main problems and gaps in the literature that motivated the research in this thesis along with proposed solutions will be discussed.

1.4 Problem Identification and Proposed Solutions

1.4.1 Emission Estimation Modelling challenges

Due to the complexity of combustion phenomena and the high number of subsystems in ICE, physical-based model development is time-consuming and may become non-linear and non-convex. ECUs are not capable of the computation required for detailed physical emission models; thus, these models cannot be used to control emissions in real-time. In addition, the accuracy of the physics-based method is typically compromised, mainly due to linearization or model-order reduction techniques. Data-driven or black-box models that use measurement data directly for training ML methods are an alternative approach for modeling. These models could be as accurate as 3D CFD physical models and require significantly less processing time for implementation of model-based controllers in ECUs [149–151].

Despite the many advantages of data-driven black-box models, there are two main drawbacks: i) they are complex and could run a high risk of overfitting, particularly when a large number of features are used [149–151], and ii) because they do not contain physical models, their accuracy will be decreased when physics change [89, 152]. The proposed method for the first challenge is to develop a Model Order Reduction (MOR) or Feature Selection (FS) algorithm. The propose method for the second challenge is to use gray-box or hybrid modeling techniques. The first challenge is

addressed by developing a SVM-based Model Order Reduction (MOR) (will be described in Chapter 3) and the second is addressed by developing a gray-box model using physical understanding of the engine (will be described in Chapter 4).

In Chapter 3, a correlation based MOR algorithm is developed using an SVM to model NO_x emissions and the Brake Mean Effective Pressure (BMEP) of a diesel engine. The SVM-based MOR algorithm is used to reduce the number of features of a 34-feature Full-Order Model (FOM) by evaluating the regression performance of the SVM-based model. Then, the SVM-based MOR algorithm is used to reduce the number of features of the FOM. Two models for NO_x emissions and BMEP are developed via MOR, one complex model with high-accuracy, called a High-Order Model (HOM), and the other with acceptable accuracy and a simple structure, called a Low-Order Model (LOM). The HOM has 29 features for NO_x and 20 features for BMEP, while the LOM has nine features for NO_x and six features for BMEP. The results illustrate that the developed SVM model has shorter training times (five to 14 times faster) and higher accuracy especially for test data compared to the ANN model.

In Chapter 4, a comprehensive analysis of diesel engine soot emissions modeling for control applications is presented by developing physical, black-box, and gray-box models for soot emissions prediction. Different feature sets based on the least absolute shrinkage and selection operator (LASSO) feature selection method and physical knowledge are examined to develop computationally efficient soot models with good precision. The physical model is a virtual engine modeled in GT-Power© software that is parameterized using a portion of experimental data. Different ML methods, including Regression Tree (RT), Ensemble of Regression Trees (ERT), svm, GPR, ANN, and BNN are used to develop the black-box models. The gray-box models include a combination of the physical and black-box models. A total of five feature sets and eight ML methods are tested. An analysis of the accuracy, training time and test time of the models is performed using the K-means clustering algorithm. This

provides a systematic way to categorize the feature sets and methods based on their performance and to select the best method for a specific application.

1.4.2 Model-based controller challenge

As mentioned in Section 1.1, MPC is one of the most promising controllers to deal with the highly constrained nonlinear system of ICEs. The use of a future horizon while optimizing the current control law is considered in MPC. Constraints on state, input and output variables are also structured in MPC to handle multi-variable systems. Constraint enforcement while performing optimization is a main advantage of MPC, and useful to address ICE requirements. Since MPC inherently enforces constraints this significantly reduces calibration and development time. Flexible handling of uncertainty, delays, and non-linearity in the model and the possibility of both offline and real-time implementation are other advantages of MPC that make it applicable in the automotive industry [41]. These MPC features have resulted in MPC being investigated widely in the automotive industry. However, MPC requires additional computing and memory resources compared with classic control and an accurate model is also needed.

To address these drawbacks of MPC (the need for fast optimization and an accurate model), ML techniques can be used to obtain the performance of MPC but at a lower computational cost. A variety of ML techniques can be combined with MPC to provide and update data-driven models and improve computational demand.

As discussed in Section 1.3, several ML-based data-driven modeling techniques have been used, including ANN [102, 103], ELM [107, 108], BNN [106], and LS-SVM LPV [39, 40, 110]. Among these models, the LS-SVM LPV model provides an accurate model. The traditional use of an LPV model requires the evaluation of a complex model in a grid of scheduling parameters; however, using ML provides a systematic way to create a state-space LPV model directly from measurement data. SVM-LPV [153] uses a LS-SVM framework to update state-space matrices. This

method has been applied to HCCI engine LPV modeling [39, 40, 110]. In this thesis SVM-LPV has been extended for CI engines emission prediction and a LPV MPC controller is designed accordingly.

A nonlinear model can improve the accuracy of emission modeling resulting in better control of MPC performance. One of the methods explored in this thesis that has not been explored in the ICE-related literature is Deep Recurrent Neural Network such as Long-Short Term Memory (LSTM) network. A Recurrent Neural Network (RNN) is structurally similar to a feedforward neural network with the exception of backward connections used to handle sequential time series. The advantage of the RNN compared to a conventional feedforward neural network for dynamic modeling is its computational efficiency which is the result of parameter sharing. However, conventional RNN cannot accurately capture long-term dependencies of the model. This can also be described as the "vanishing gradient," as the contribution of earlier steps becomes increasingly small. To solve this lack of long term memory of RNN, various types of cells with long-term memory have been introduced. The most popular and well-known of these long-term memory cells is the LSTM [154]. Therefore, in this thesis, a deep network with an LSTM layer which capable of predicting dynamics of a system with high accuracy due to long-term memory is used.

Even with ML-based modeling, MPC computational times (especially for deep networks) is high. Additionally, depending on the control complexity, convexity, and dynamics time scale it might not be feasible for real-time implementation. Since ICEs are complex nonlinear systems to control, high computational effort is often required for real-time implementation. One method to reduce MPC computation load is to replace MPC with an ML controller. The ML is used to mimic the MPC controller's behavior to significantly reduce computational time for real-time implementation. The optimization uses a powerful prototype ECU, and only an ML function is deployed for the MPC in real-time to significantly reduce the computational time required of a production ECU with limited computational capability. Here, to reduce the computa-

tional time of MPC, an ML-based imitation of the MPC controller is proposed where a deep network is trained using the MPC inputs and outputs. Imitation MPC has been previously explored in the control of heating, ventilation, and air conditioning (HVAC) systems [155, 156], vehicle dynamics control [157, 158], robotics [159], and power conversion [160] and has shown great success. There is little in the literature about applying imitation MPC to ICE control. Therefore, in this thesis, both modeling and imitation of MPC for ICEs has been investigated using both real-time and Engine Simulation Model (ESM) – a detailed physics-based model co-simulation with Matlab/Simulink©.

In Chapter 5, the SVM-LPV is used to model engine performance and the engine-out NO_x emissions to show the capability of the SVM-LPV technique. This is the first study to use this technique in emission modeling and control of a CI engine. LPV MPC is developed based on the developed SVM-LPV model. The online optimization of the MPC offers advantages in minimizing NO_x emissions and fuel consumption compared to the baseline feedforward production controller (ECU modeled in GT-power©). For imitation of MPC, the LPV-MPC is first implemented on ESM. Then, the input and output are recorded and a deep neural network, including a Long-Short-Term Memory (LSTM) layer, is used to mimic the behavior of the LPV MPC.

In Chapter 6, a deep neural network including LSTM layers is used to model the NO_x emission and engine performance of a diesel engine. Compared to the traditional ANN method to model system dynamics the LSTM can capture long-term dependencies in the data. A Nonlinear MPC (NMPC) is required for this process. As LSTM has a hidden and cell state, a new methodology of NMPC is developed to implement NMPC. Therefore, a novel approach to augment LSTM in the NMPC problem (LSTM-NMPC) by augmenting LSTM hidden and cell state into the nonlinear optimization problem has been developed. This LSTM-NMPC is used to develop imitation of NMPC. In the first step, the designed LSTM-NMPC are implemented on ESM. Next, the NMPC input and output are recorded, and a deep neural network

similar to the imitation LPV-MPC (Chapter 5), including an LSTM layer, is used to fit the controller data to mimic the behavior of the NMPC. The final step in the process involves replacing the online NMPC with an imitative controller to avoid the high computational time of NMPC. That is, instead of solving NMPC optimization online, the identified function, here a deep network is deployed with a much lower computation cost.

Comparisons between the LPV-MPC and LSTM-NMPC show the superior performance of the LSTM-NMPC in emission and fuel consumption reduction and load tracking performance. Therefore, the developed LSTM-NMPC is then implemented in the MicroAutoBox prototype system in Chapter 7. For adaptation, the LSTM model is modified by reducing the number of states while increasing the number of fully connected layers to make NMPC turnaround time feasible for a real-time system. Then, based on real-time data, the model is updated. Due to the accessibility of some new variables such as Particulate Matter (PM), Maximum Pressure Rise Rate (MPRR), and Indicated Mean Effective Pressure (IMEP) are added to the control problem. This makes real-time implemented controller more practical for an industry standard engine controller.

1.4.3 Model-free controller challenge

RL has been used in automotive powertrain control systems especially in energy management of hybrid electric vehicles [161, 162] and for internal combustion engines [115, 116, 119–121, 163]. Q-learning RL is used as idle speed control for a Spark Ignition (SI) engine by controlling the spark timing and intake throttle valve position [118]. Similar studies have been carried-out for diesel engine idle speed control by the control of fuel injection timing [119]. RL has also been used for emission control of spark ignition engines [115, 116]. A very limited number of studies have been carried out utilizing RL for internal combustion control, and most of the existing work has focused on spark-ignition engines. Deep RL algorithms have not been

implemented for diesel engine performance and emissions control. Safety concerns and constraints violations of pure learning controllers in highly complex systems such as internal combustion engines have hindered the development of these learning controllers. Fortunately, recent studies have addressed output constraints enforcement in the learning-based controller using a safe learning filter. This method enforces the output constraints and provides a method to implement safe-learning RL [164–167]. In this thesis, safe-learning RL has been extended to the CI engine.

Safe learning combined with a deep RL for control diesel engine emissions is not available in the literature. In Chapter 8, a deep RL with and without safety filters is designed and a comparison conducted to illustrate the potential of safe RL in engine and emission control. To compare RL to ILC, ILC and safe ILC are also designed. Additionally, RL is compared with the LSTM-NMPC that is developed in Chapter 6.

1.5 Contributions and Thesis outline

1.5.1 Thesis outline

This thesis is organized into five main parts in nine chapters. Figure 1.6 shows the three main core parts schematically. The main parts and chanters of this thesis are as follow:

- **PART I: Introduction and Experimental Setup**
  - **Chapter 1** provides background, motivation, and main contributions of this thesis.
  - **Chapter 2** presents the experimental setup details, Explanatory Data Analysis (EDA), and ESM details.
- **PART II: Machine Learning in Emission Prediction**

- **Chapter 3** describes developed ML-based NO_x steady-state model by developing MOR algorithm.
- **Chapter 4** describes modeling the PM (soot) steady-state model using gray-box and black-box techniques.

- **PART III: Integration of Machine Learning and Model Predictive Control**
  - **Chapter 5** describes ESM simulation-based implementation of MPC development using (SVM-LPV) and imitation MPC accordingly based on online MPC optimization.
  - **Chapter 6** shows ESM simulation based implementation of NMPC developed using an LSTM network and corresponding imitation NMPC based on online NMPC optimization.
  - **Chapter 7** describes real-time LSTM-NMPC implementation in engine and IN comparison with existing ECU.

- **PART IV: Machine Learning in Learning-based Controller**
  - **Chapter 8** provides developed safe learning algorithms for ILC and RL based on ESM simulation.

- **PART V: Conclusions**
  - **Chapter 9** provides conclusions.

1.5.2 Contributions

To summarize, the main contributions of this thesis are:

- **PART I: Introduction and Experimental Setup**

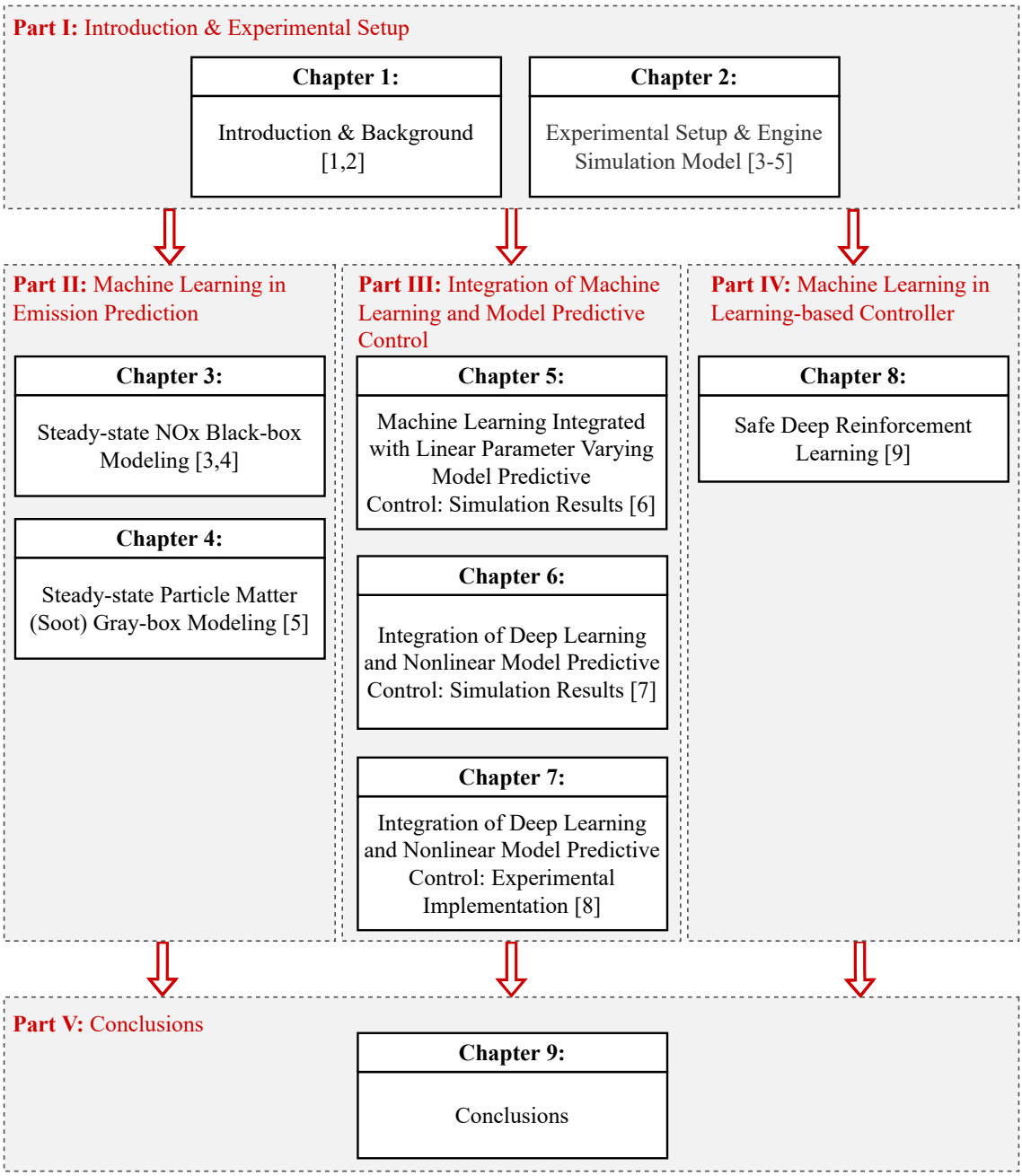


Figure 1.6: Schematic of the thesis organization

- Setting up a medium duty Diesel engine for experimental analysis and controller implementation,
- Exploratory analysis of data for "complete" speed-load maps from a medium-duty diesel compression ignition engines for use in steady-state NO_x and Soot models.

- **PART II: Machine Learning in Emission Prediction**
  - Developing a novel model order reduction algorithm for modeling NO_x and to trade-off model complexity and model accuracy,
  - Developing black-box and gray-box soot models using various ML techniques,
  - Developing a systematic clustering-based method to choose the best ML/feature set based on the model application.
- **PART III: Integration of Machine Learning and Model Predictive Control**
  - Adapting an SVM-LPV model to develop a linear parameter-varying model for engine-out NO_x emissions and engine performance metrics.
  - Designing an LPV-MPC based on an SVM-LPV model to minimize engine-out emissions and fuel consumption while maintaining the same output torque performance and comparing with a benchmark controller using simulation (ESM)
  - Designing an imitation based controller using deep neural network to clone the behavior of LPV-MPC to reduce the computational time of optimization.
  - Developing a transient engine performance and emission model based on LSTM capable of providing a high accuracy model for nonlinear model predictive control.
  - Developing a novel approach to augment LSTM in the NMPC problem (LSTM-NMPC) by augmenting LSTM hidden and cell state into nonlinear optimization problem.
  - Designing an NMPC based on an LSTM model to minimize engine-out emission and fuel consumption while maintaining same output torque

performance and is compared with a benchmark controller in simulation (ESM),

- Designing an imitation based controller using a deep neural network to clone the behavior of LSTM-NMPC to reduce the computational time of optimization while maintaining the NMPC performance.
- Adapting LSTM-based deep neural network based on real-time experimental data to develop transient engine performance and emission model. This model is capable of providing a high accuracy model for NMPC.
- Designing and real-time implementation of an NMPC based on the developed LSTM-based deep neural network to minimize engine-out emission and fuel consumption while maintaining the same output torque. This controller is then compared to the Cummins-calibrated ECU-based benchmark control is that replicated using a MicroAutoBox.

- **PART IV: Machine Learning in Learning-based Controller**
  - Designing a deep RL controller for diesel engine NO_x control to minimize NO_x and fuel consumption while maintaining the same output torque.
  - Designing a safe filter that provides safe RL and safe ILC for diesel engine emission control.

Chapter 2

Experimental Setup and Engine Simulation Model [1]

2.1 Experimental Setup

2.1.1 Engine and Engine Controller Setup

A 4.5-liter medium-duty Cummins diesel engine was used for control design and experimental testing in this thesis. Table 2.1 shows the relevant specifications of the Cummins QSB4.5 160 diesel engine that was used. The experimental setup and the schematics of experimental setup are shown in Figure 2.1 and 2.2.

Table 2.1: Engine specifications

Parameter	**Value**
Engine type	In-Line, 4-Cylinder
Displacement	4.5 L
Bore × Stroke	102 mm × 120 mm
Peak torque	624 N.m @ 1500 rpm
Peak power	123 kW @ 2000 rpm
Aspiration	Turbocharged
Certification Level	Tier 3 / Stage IIIA

[1] This chapter are partially based on [3–5]

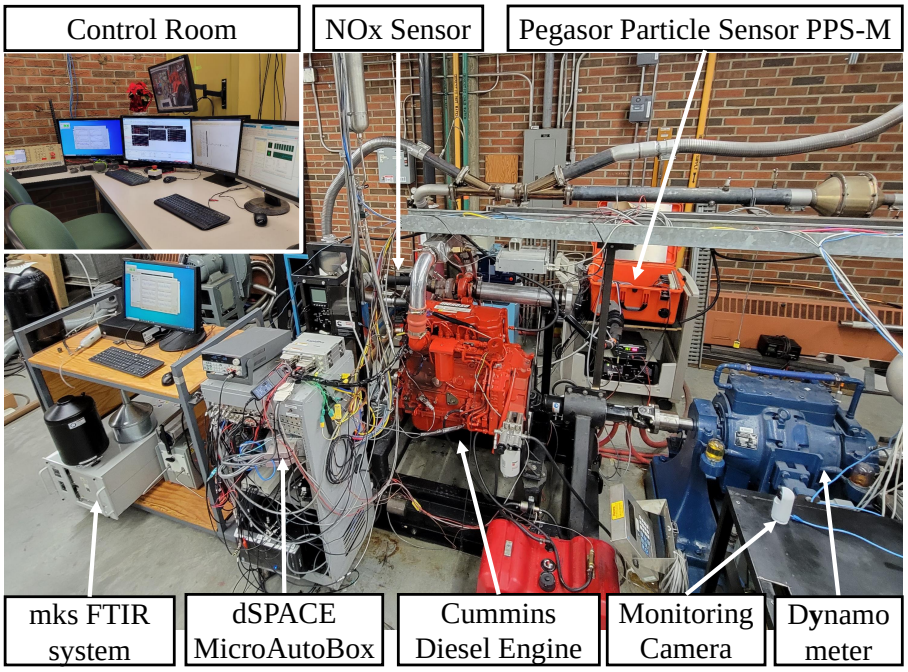


Figure 2.1: Diesel engine experimental setup

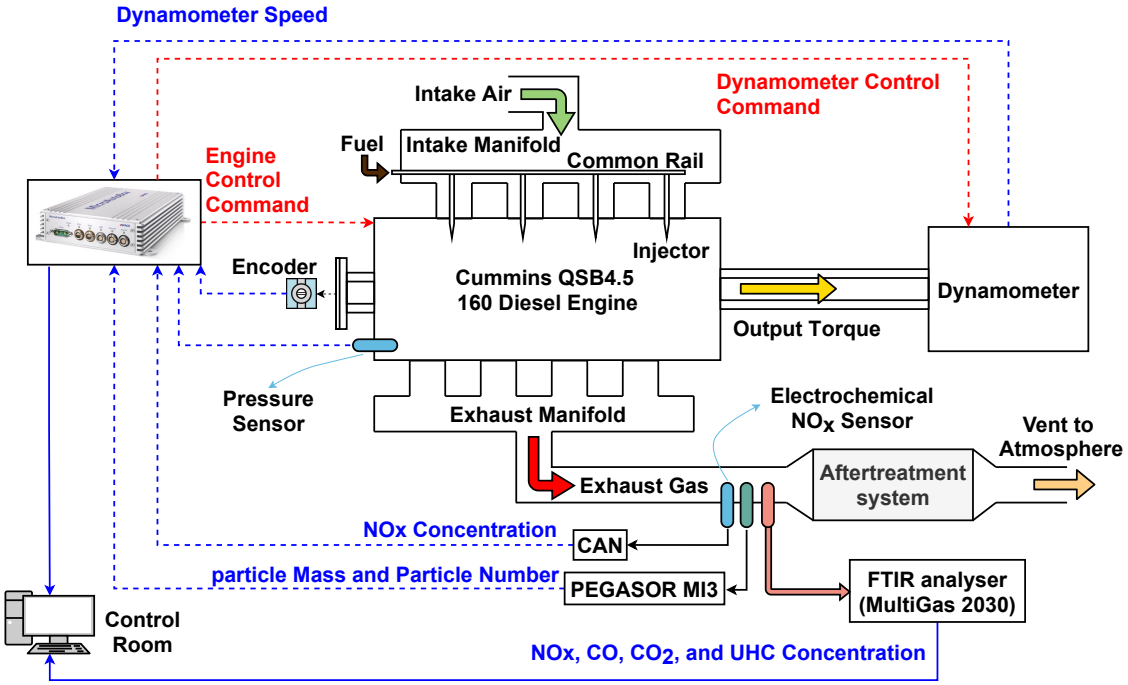


Figure 2.2: Schematic of diesel engine experimental setup

The Cummins production ECU was fully duplicated on an open fully flexible dSpace MictoAutoBox II (MAXB II). This was used to control various engine pa-

rameters including intake air pressure, engine speed, load, injected fuel amount, and fuel rail pressure. A Kistler piezoelectric pressure sensor was used to measure the in-cylinder pressure for all tests.

A National Instruments Data Acquisition System (DAQ) was used to record cylinder pressure at a 0.1° resolution for use in offline post-processing. The pressure signals were simultaneously input to the Field Programmable Gate Array (FPGA) board in the prototyping ECU. Details of MABX II prototyping ECU are provided in Table 2.2. The MABX II contains two main boards: a CPU and FPGA. The CPU (ds1401) was used to replicate the production Cummins ECU tables as well as to implement the controller developed in this work.

The Xilinx Kintex-7 FPGA contained within the MABX II was used to calculate various combustion metrics in real-time. These included IMEP and MPRR which were transferred from the FPGA to CPU to use as inputs to the NMPC. Details regarding the real-time calculation of these properties can be found in [168, 169].

To measure engine-out emissions, an electrochemical NO_x sensor, Pegasor Particle Sensor (PPS-M), and mks Fourier-Transform Infrared Spectroscopy (FTIR) were used. The mounted sensors to the exhaust pipe are shown in Figure 2.3.

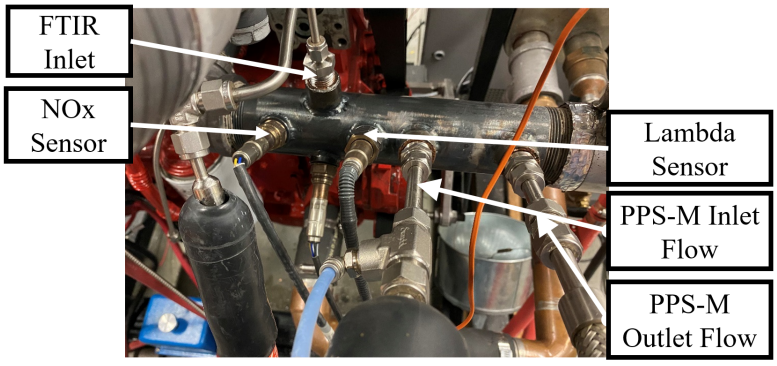


Figure 2.3: Diesel engine exhaust pipe– The FTIR, PPM, NO_x, and Lambda sensor are mounted in this pipe

Table 2.2: Rapid prototyping ECU Specifications

	Parameter	Specification
Processor	dSPACE® 1401	IBM PPC-750GL
	Speed	900 MHz
	Memory	16 MB main memory
I/O	dSPACE® 1511	
	Analog input	16 Parallel channels
	Resolution	16 bit
	Sampling frequency	1 Msps
	Analog output	4 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 RAM	445
	DSP	840
	I/O	478

2.1.2 Electrochemical NOx sensor

A production amperometric NO_x sensor (ECM-06-05) was used in the experiments. All the sensor working parameters were set using the sensor control module (*ECM-NOxCANt* P/N: 02-07). The sensor control module was connected to a computer via a Controller Area Network (CAN) interface (*Kvaser Light HS*) to monitor and log the measurements. The sensor along with sensor control module are shown in Figure 2.4.

Figure 2.4: electrochemical fast NO_x sensor and sensor control module

2.1.3 Fourier-Transform Infrared Spectroscopy (FTIR)

A Fourier-Transform Infrared Spectroscopy (FTIR) analyser (*MultiGas 2030*) was used to validate the ECM NO_x sensor measurement and to measure the concentration of other species in the exhaust gas. The FTIR spectrometer passes an infrared beam through a gas sample, obtains the interference pattern of the gas, and identifies the gas composition based on the absorption spectrum of the gas constituents. The FTIR analyser was connected to the diesel engine exhaust pipe to measure the engine raw emissions. The sample exhaust gas passed through two heated filters (*Flexotherm Flex*) connected by sample lines (*Flexotherm*) heated to 191^{o}C to avoid water vapor condensation in the sample gas as shown in Figure 2.5. The sample data was collected at a 5 Hz frequency using the mks Series 2000 MultiGas analyzer software version 10.1. In this setup liquid nitrogen was used to cool detector (which can maintain cryogenic temperatures for up to 12 hours), dry nitrogen gas for optics, and purge the spectrometer. More information about the FTIR setup is available in [170].

2.1.4 Pegasor Particle Sensor (PPS-M)

To measure soot emissions, a Pegasor Particle Sensor (PPS-M) was used. The schematic of the soot measurement setup is shown in Figure 2.6 where engine-out

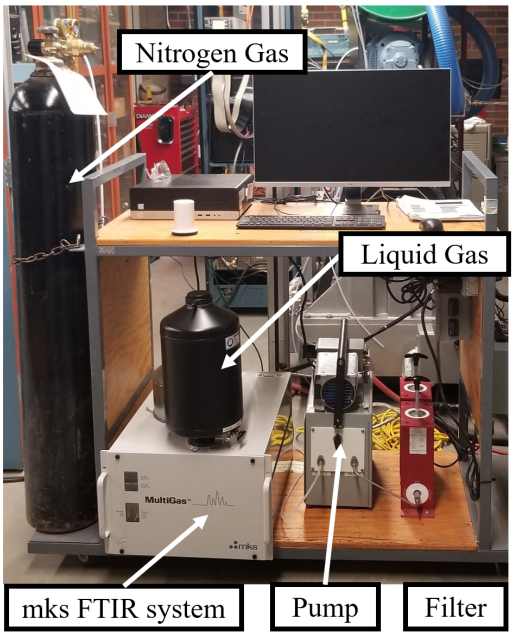


Figure 2.5: FTIR setup

exhaust gas flows through an inlet heater line to the pre-charger. The pre-charger was used to avoid any charge-related problem in soot measurement [171]. The pre-charger is essential to the accuracy of soot measurement because in recent emission technology, microscopic particles in the exhaust may be strongly charged. The Pegasor Pre-Charger is a self-heated, non-radioactive, negative diffusion charger. Using an integrated trap, Pegasor can eliminate ions and small charged particles from the sample line gas. It charges larger particles into a known negative charge state. The PPS-M sensor was cleaned for tests to allow good quality air at the right pressure inside the sensor pump unit. The sampling rate of PPS-M was 100 Hz with a 100 dB Sensor to Noise Ratio (SNR). This sensor detects particle sizes in the range of $[0.001, 290][\mathrm{mg/m^3}]$. The main PPS-M sensor's specifications are listed in Table 2.3.

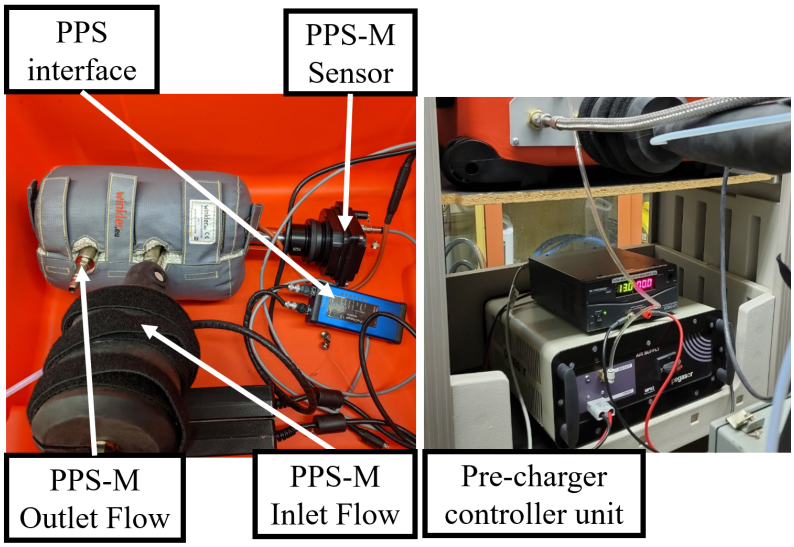


Figure 2.6: Pegasor Particle Sensor (PPS) setup with air pre-charger unit

Table 2.3: Pegasor Particle Sensor sensor specifications

Parameter	**Value**
Sensor temperature	200 °C
Extracted sample temperature	-40 up to 850 °C
Dilution	No need
Time response	0.2 s
Measured particle size range	10 nm and up
Particle number range	300 up to 10^9 1/cm^3
Particle mass range	10^{-3} up to 300 mg/m^3
Sample pressure	–20 kPa to +100 kPa
Clean air/Nitrogen supply	10 LPM @ 0.15 MPa
Operating voltage	24 V
Power consumption	6 W
Maximum theoretical error*	±24% PN and ±38% PM

* not engine specific

2.2 Exploratory Data Analysis (EDA)

2.2.1 Steady-state data analysis

The diesel engine was tested for 219 engine steady state operating conditions over the full range of engine speeds and loads in order to develop a steady-state model and understand the behaviour of the engine. Figures 2.7 and 2.8 shows the color map of raw soot and NO_x emissions data with respect to engine speed (x-axis) and load (y-axis), where black dots represent experimental points. Since this engine is designed for stationary applications, it has limited operating conditions. Therefore, 219 data points in Figures 2.7 and 2.8 cover most of the possible operating conditions.

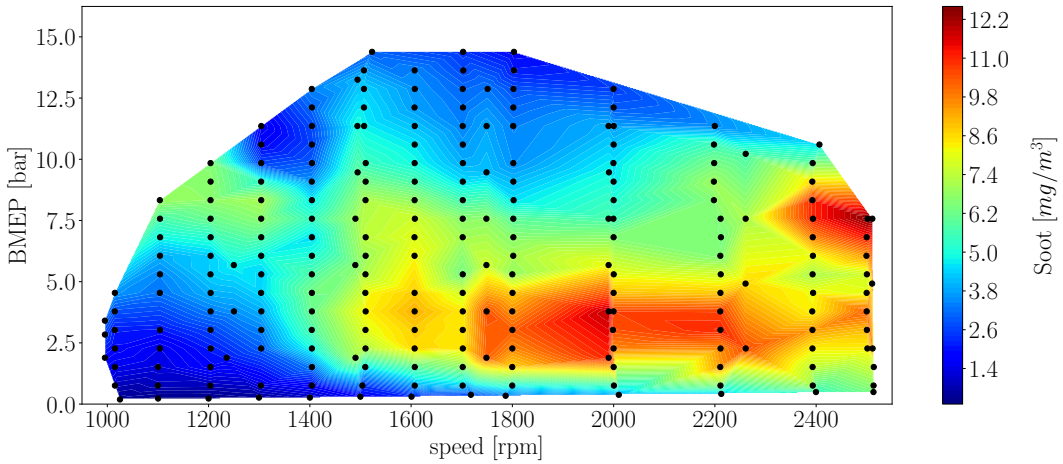


Figure 2.7: Engine-out soot measurements over speed and Break Mean Effective Pressure (BMEP)

To analyze the main features of the diesel engine that play an important role in engine emissions modeling, the histogram of them are plotted in Figure 2.9. This diesel engine has three injection pulse. The third injection was active in 39% of our experimentally collected data based on Figure 2.9b. Start and duration of all pulse of injections along with the total injected fuel in each cycle are shown in Figure 2.9a–d. Another main fuel path feature that affects soot emissions modeling is common rail pressure as shown in Figure 2.9e. The majority of data were collected in fuel

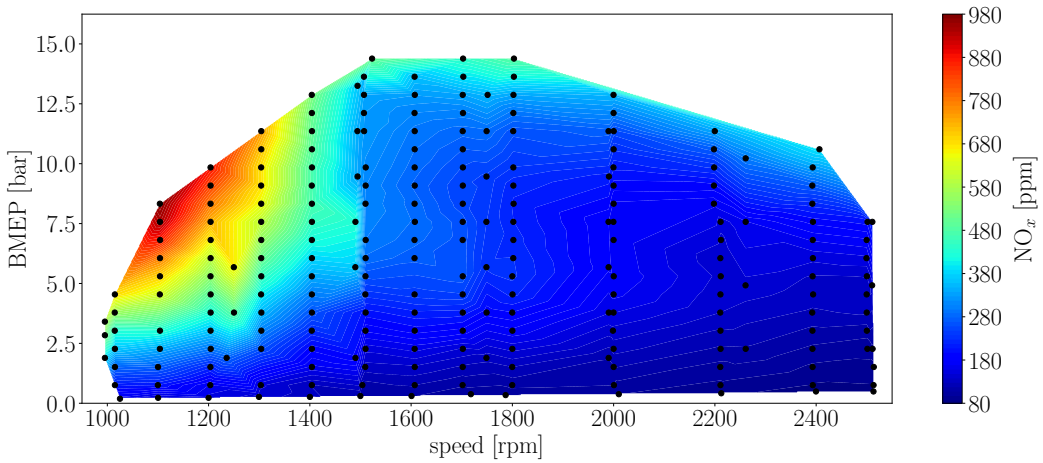


Figure 2.8: Engine-out NOx measurements over speed and Break Mean Effective Pressure (BMEP)

rail pressure from 700 to 1100 bar. The air path, intake manifold pressure and air-fuel equivalence ration (λ) are shown in Figure 2.9f–g. Output torque and engine speed are the other important features and are shown in Figure 2.9h–i. According to these histograms, the data collected from experiments covers most of the operating conditions of the engine successfully. Additionally, to conduct more in-depth analysis the start of injection for all pulses, all experimental points are plotted in Figure 2.10 where the x-axis is the Crank Angle Degree (CAD) and the y-axis is engine generated power. As shown, the above specific power post-injection is active in order to reduce unburned hydrocarbon and soot emission.

2.2.2 Transient data analysis

In order to develop transient models, transient data is collected by using Pseudo random binary generated inputs. For this reason, the main inputs of the system include the Duration of Injection (DOI) of the pilot (pre) injection, the DOI of the main injection, duration between the end of the pre-injection to the start of main injection (t_{P2M}), the Start of Injection (SOI) of the main injection, and common fuel rail pressure were changed randomly using a randomly generated sequence both for

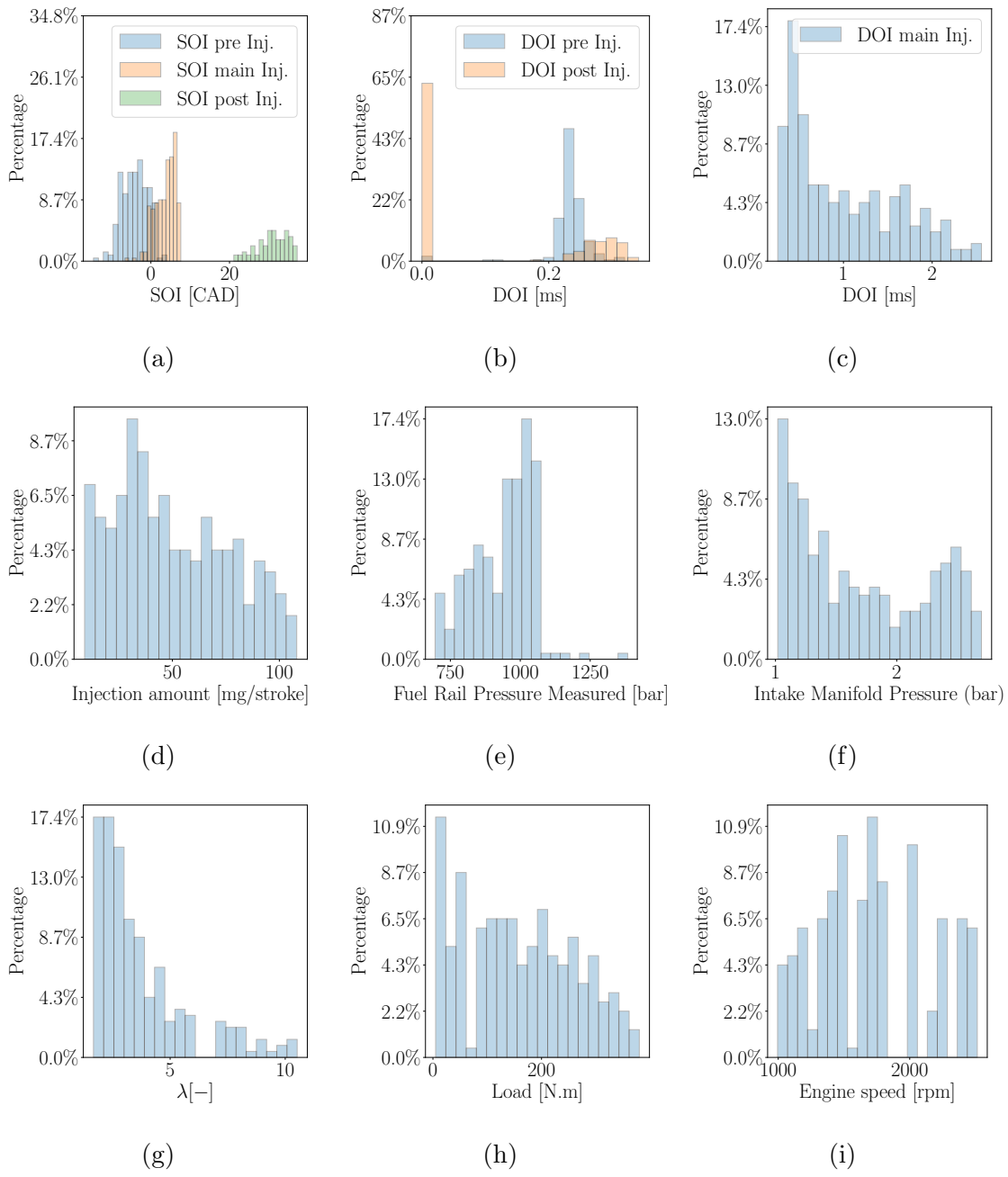


Figure 2.9: Diesel engine with soot measurement exploratory data analysis: a) Start of Injection (SOI), Duration of Injection (DOI), c) DOI of main injection, d) Injection amount, e) Fuel Rail Pressure Measured, f) Intake Manifold Pressure, g) fuel Equivalent ration (λ), h) load (engine output torque), i) Engine speed

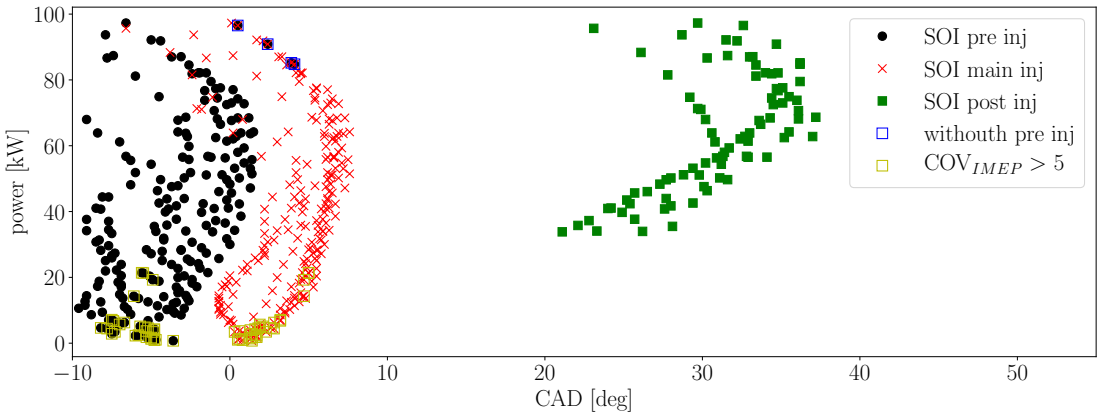


Figure 2.10: Engine produce power verses Start of pilot (pre), main, and post injection

frequency and amplitude. Then, the outputs of interest have been recorded. For emission, soot and NO_x are measured using a PPS-M sensor and an electrochemical fast response NO_x sensor. To calculate the Indicated Mean Effective Pressure (IMEP) and Maximum Pressure Rise Rate (MPRR), an online Field Programmable Gate Array (FPGA) calculation was used based on measured pressure from an in-cylinder pressure sensor.

2.3 Engine Simulation Model (ESM)

The first step toward developing an Engine Simulation Model (ESM) was to develop and parameterize the GT-Power physics-based model. GT-power© is a commercial software for modeling combustion engines. Physical modeling of the diesel engine is carried out using the GT power software, which contains several chemical and physical sub-models that simulate complex combustion processes. The DIPulse model is employed as the combustion model because it can deal with multi-injection combustion engines. An extended Zeldovich NO_x model is added to the combustion model in order to add a NO_x prediction for ESM. Approximately 15% of the raw experimental data is used to calibrate the combustion model using the Genetic Algorithm (GA) algorithm in GT-Suite© software. The calibration process uses NSGA-III [172] for multi-objective Pareto optimization as the search algorithm. GA is the optimal

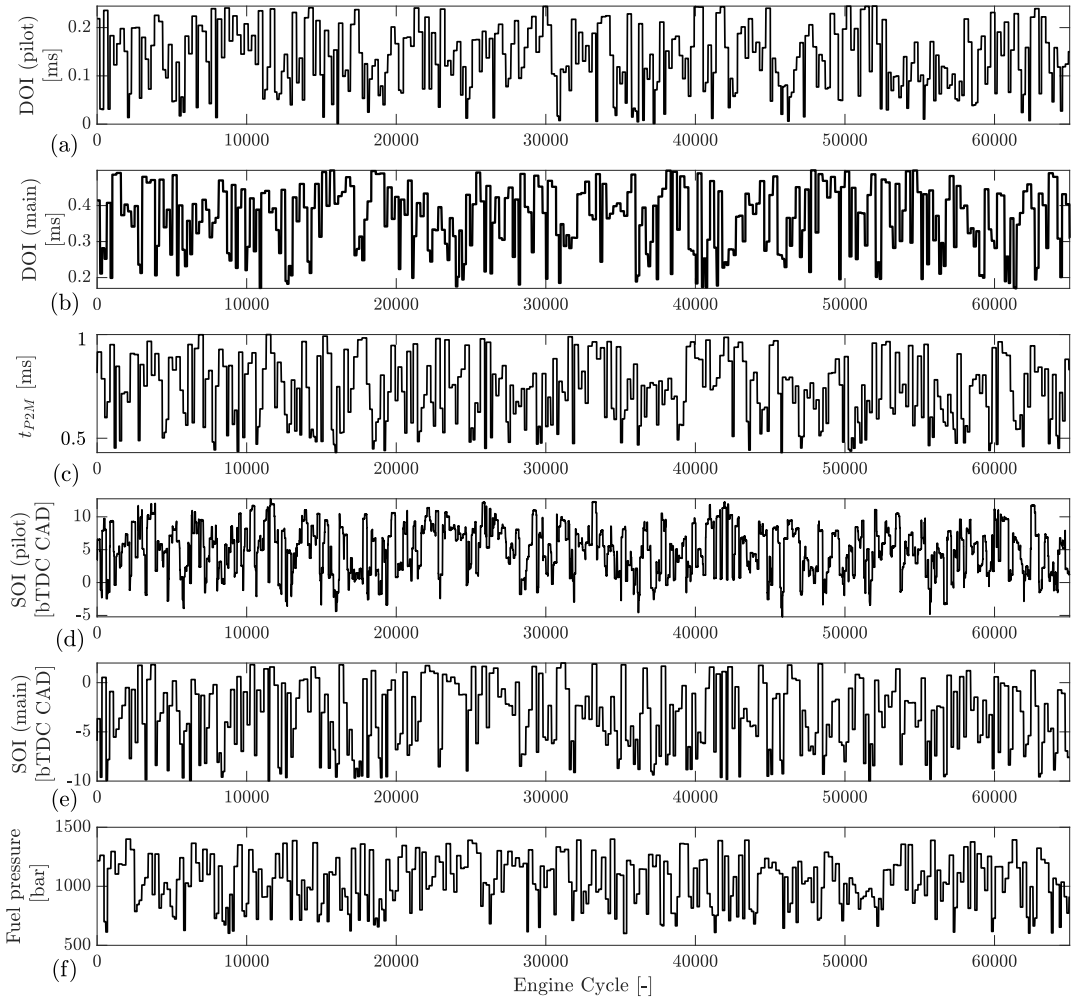


Figure 2.11: Experimental transient data– manipulated inputs: a) DOI of pilot injection, b) DOI of main injection, c) duration between end of pre-injection to start of main injection (t_{P2M}), d) SOI of pilot injection, e) SOI of main injection, f) fuel rail pressure

choice for problems with different levels of complexity, because of its ability to explore a broad design space [172].

Figure 2.13 schematically shows how a combustion model with NO_x physics-based model multipliers is calculated using the GA-based algorithm. The GAs, based on the results obtained, took into account experimental results of in-cylinder pressure traces for some optimization points. The multipliers for the combustion model are:

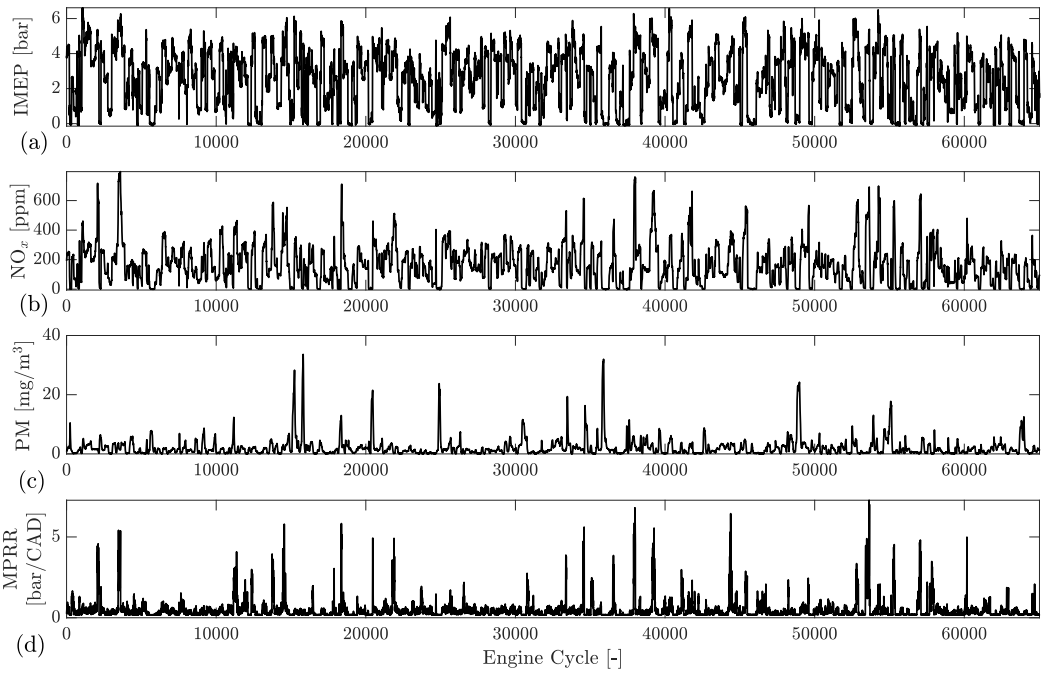


Figure 2.12: Experimental transient data– measured outputs: a) Nitrogen oxide (from fast response sensor), b) measured soot emission from PPM sensor, c) calculated indicated mean effective pressure (IMEP) based on online FPGA calculation, d) calculated maximum pressure rise rate (MPRR) based on online FPGA calculation

Entrainment Rate Multiplier, Ignition Delay Multiplier, Premixed Combustion Rate Multiplier, and Diffusion Combustion Rate Multiplier for DIPulse combustion model and NO_x calibration multiplier and end oxidation/activation energy multiplier for extended Zeldovich NO_x model. As DIPulse combustion multipliers are also affected by NO_x model accuracy, the DIPulse combustion model and extended Zeldovich NO_x model are calibrated using one multi-objective optimization. The GAs minimize the deviation between the experimental and simulation in-cylinder pressure trace and experimental NO_x value to calculate the optimal multipliers.

The validation result for crank angle position where 50% of the heat is released (CA50), NO_x, and maximum in-cylinder pressure is shown in Figure 2.14. The average error for CA50 and maximum in-cylinder pressure are about 2 CAD and 6% respectively, demonstrating the physical model's reliability.

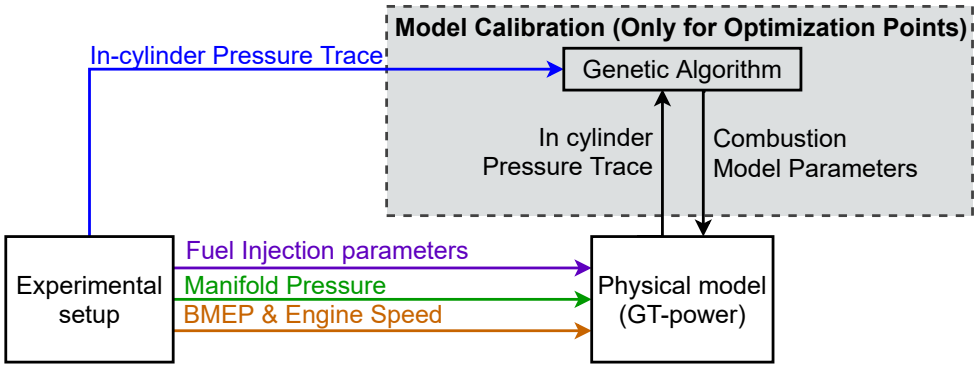


Figure 2.13: Engine Simulation Model (ESM) development procedure in GT-power© software

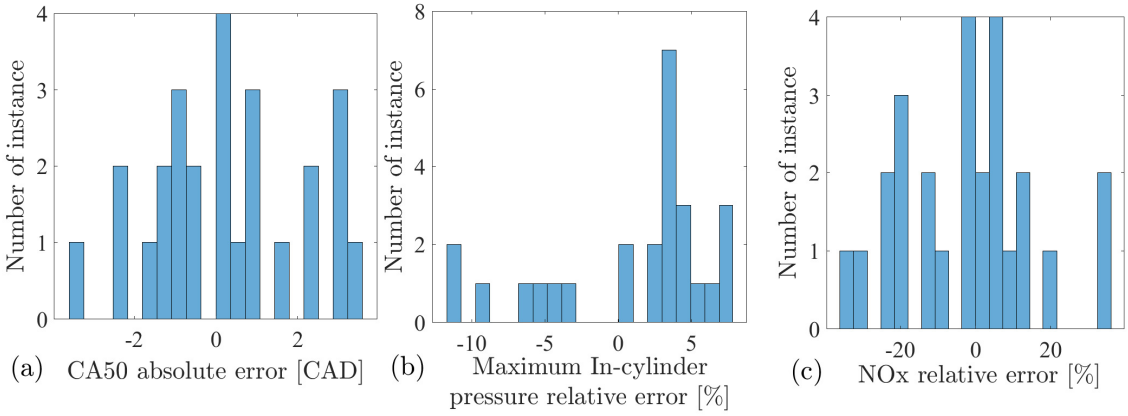


Figure 2.14: Histogram of error between physical-based model and experimental data

The in-cylinder pressure trace for different load and speed conditions are shown as a function of the crank angle (CAD) in Figure 2.15. Case I (136 [N.m] in 1200 [rpm]), case IV (271 [N.m] in 1800 [rpm]) and case VI (353 [N.m] in 2400 [rpm]) are selected from optimization points for model calibration (refer to Figure 2.13) while other cases are not used for calibration. This GT-power© based virtual combustion engine model is used in this thesis as a Engine Simulation Model (ESM) to develop data-driven models, and design controllers, and to evaluate the developed controllers.

2.4 Summary of chapter

This chapter provided the details of the experimental setup and experimental data analysis. Then, a physical-based combustion model using the DIPulse model and

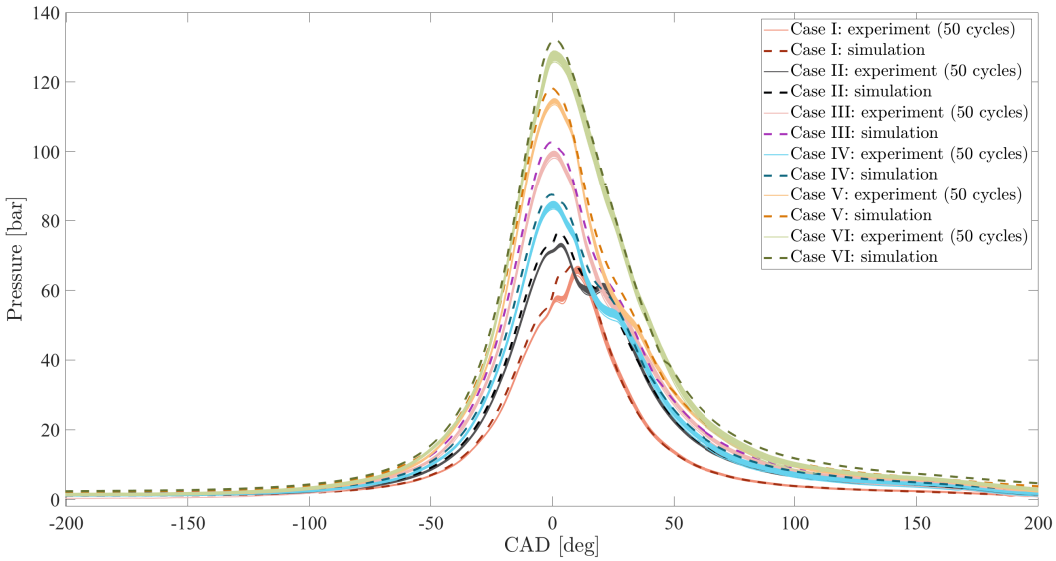


Figure 2.15: Engine Simulation Model (ESM) validation for six operating points. (Case I: 136 [N.m] in 1200 [rpm], Case II: 271 [N.m] in 1600 [rpm], Case III: 271 [N.m] in 1400 [rpm], Case IV: 271 [N.m] in 1800 [rpm], Case V: 271 [N.m] in 2000 [rpm], and Case VI: 353 [N.m] in 2400 [rpm]

an extended Zeldovic model for NO_x emission were developed using experimentally collected data. This model will be used in Chapter 4 for gray-box soot modeling. In addition, this model's co-simulation with Matlab/Simulink©, called ESM, will be used in Chapters 5 and 6 for developing transient model and model predictive control implementation. This model is also used in Chapter 8 to develop a learning-based RL and ILC controller for emission reduction.

PART II: Machine Learning in Emission Prediction

Chapter 3

Steady-state NOx Black-box Modeling [1]

A correlation-based Model Order Reduction (MOR) algorithm is developed using a Support Vector Machine (SVM) to model NO_x emission and the Break Mean Effective Pressure (BMEP) of a medium-duty diesel engine. The SVM-based MOR algorithm is used to reduce the number of features in a 34-feature Full-Order Model (FOM) by evaluating the regression performance of the SVM-based model. Then, the SVM-based MOR algorithm is used to reduce the number of features of the FOM. Two models for NO_x emission and BMEP are developed via MOR, one complex model with a high-accuracy, called the High-Order Model (HOM), and the other with an acceptable accuracy and simple structure, called the Low-Order Model (LOM). The HOM has 29 features for NO_x and 20 features for BMEP, while the LOM has nine features for NO_x and six features for BMEP. Then, the steady-state LOM and HOM are implemented in a Nonlinear Control-Oriented Model (NCOM). To verify the accuracy of the NCOM, a fast-response electrochemical NO_x sensor is used to experimentally study the engine transient NO_x emissions. The HOM and LOM SVM models of NO_x and BMEP are compared to a conventional Artificial Neural Network (ANN) with one hidden layer. The results illustrate that the developed SVM model has shorter training times (5 to 14 times faster) and higher accuracy, especially for test

[1] This chapter is based on [3, 4]

data compared to the ANN model. A control-oriented model (COM) is then developed to predict the system's dynamic behavior. Finally, the performance of the LOM and HOM are evaluated for different rising and falling input transients at four engine speeds. The transient test results validate the high accuracy of the HOM and the acceptable accuracy of the LOM for both NO_x and BMEP. The HOM is proposed as an accurate virtual plant while the LOM is suitable for model-based controller design.

3.1 Support Vector Machine

3.1.1 Convex Optimization Problem

The SVM, introduced by Vapnik [173, 174], is a supervised machine learning approach. SVM is typically used for classification of labeled data by creating a set of hyperplanes in an infinite-dimensional space [175]. SVM is also used for regression and function approximation, also called Support Vector Regression (SVR), which was introduced by Vapnik [176]. The main idea of SVM is to find an optimal hyperplane, $\mathbf{y}(\mathbf{u_i})$, to describe a set of labeled training data, $\{\mathbf{u_i}, \mathbf{z_i}\}$, where $\{\mathbf{u_i}\}$ is the feature (input) vector and $\{\mathbf{z_i}\}$ is the target (output) vector of training data. The function $\mathbf{y}(\mathbf{u_i})$ has two main characteristics:

1. $\mathbf{y}(\mathbf{u_i})$ must be as flat as possible,
2. $\mathbf{y}(\mathbf{u_i})$ has at most ϵ deviation for all training data.

In other words, the optimization problem is to find the flattest function for which the acceptable deviation from training data is at most ϵ. The optimal hyperplane which describes the training data, $\{\mathbf{u_i}, \mathbf{z_i}\}$, can be defined as:

$$\mathbf{y}(\mathbf{u_i}) = \mathbf{w}^T \mathbf{u_i} + \mathbf{b} \tag{3.1}$$

where $\mathbf{w}$ and $\mathbf{b}$ are found by solving the SVM algorithm for regression problems. The flatness of $\mathbf{y}(\mathbf{u_i})$ in the Eq. 3.1 is achieved by minimizing the second norm of

$\mathbf{w}$. Therefore, the main objective of the SVM algorithm is to find a function which minimizes $||\mathbf{w}||_2^2$ subject to the training error tolerance of ϵ. Then, the optimization problem to find the optimum $\mathbf{y}(\mathbf{u_i})$ is defined as:

$$\begin{aligned} &\textbf{Minimize:} \quad \frac{1}{2}||\mathbf{w}||_2^2 \\ &\textbf{Subject to:} \quad \begin{cases} \mathbf{z_i} - \mathbf{w}^T\mathbf{u_i} - \mathbf{b} \leq \epsilon \\ \mathbf{w}^T\mathbf{u_i} + \mathbf{b} - \mathbf{z_i} \leq -\epsilon \end{cases} \qquad i = 1, ..., m \end{aligned} \tag{3.2}$$

where m is the number of data points. The convex optimization problem [177], Eq. 3.2, is feasible when such a $\mathbf{y}(\mathbf{u_i})$ exists which is as flat as possible and approximates all training data with at most ϵ deviation. In other words, the convex optimization problem is feasible when:

$$-\epsilon \leq \mathbf{z_i} - \mathbf{y_i} \leq \epsilon \tag{3.3}$$

So, the ϵ-insensitive linear loss function is defined as [176]:

$$L_\epsilon(\mathbf{z_i}, \mathbf{y_i}) = \begin{cases} 0 & |\mathbf{z_i} - \mathbf{y_i}| \leq \epsilon \\ |\mathbf{z_i} - \mathbf{y_i}| - \epsilon & \text{otherwise} \end{cases} \tag{3.4}$$

where the loss function would be zero if the training error is less than ϵ. Also, the empirical risk function, R_{emp}, is defined based on the loss function as [178]:

$$R_{emp}(\mathbf{w}, \mathbf{b}) = \frac{1}{m}\sum_{i=1}^{m} L_\epsilon(\mathbf{z_i}, \mathbf{y_i}) \tag{3.5}$$

where $R_{emp}(\mathbf{w}, \mathbf{b})$ is used in the optimization problem to minimize the defined loss. If this function does not exist, the convex optimization problem is infeasible. In this case, slack variables are added to Eq. 3.3 to overcome the above optimization problem infeasibility as:

$$-\epsilon - \zeta_i^- \leq \mathbf{z_i} - \mathbf{y_i} \leq \epsilon + \zeta_i^+ \tag{3.6}$$

where the slack variables are introduced as penalty variables to overcome this infeasibility of the convex optimization problem. The empirical risk function can then be rewritten based on the slack variables using Eq. 3.6 as:

$$R_{emp}(\mathbf{w}, \mathbf{b}) = \frac{1}{m}\sum_{i=1}^{m} \left(\zeta_i^- + \zeta_i^+\right) \tag{3.7}$$

Then, the convex optimization problem is modified by adding the minimizing empirical risk function term to Eq. 3.2

$$\textbf{Minimize:} \quad \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m}(\zeta_\mathbf{i}^+ + \zeta_\mathbf{i}^-)$$

$$\textbf{Subject to:} \quad \begin{cases} \mathbf{z_i} - \mathbf{w}^T\mathbf{u_i} - \mathbf{b} \leq \epsilon + \zeta_\mathbf{i}^+ \\ \mathbf{w}^T\mathbf{u_i} + \mathbf{b} - \mathbf{z_i} \leq \epsilon + \zeta_\mathbf{i}^- \\ \zeta_\mathbf{i}^-, \zeta_\mathbf{i}^+ \geq 0 \end{cases} \tag{3.8}$$

where C is a positive regulatory parameter defined as a trade-off factor between the flatness of the model and minimizing the training error tolerance. A model with tolerated error and slack variables for a single feature-single target system is schematically depicted in Figure 3.1. The ϵ-insensitive linear loss function is schematically shown in Figure 3.2.

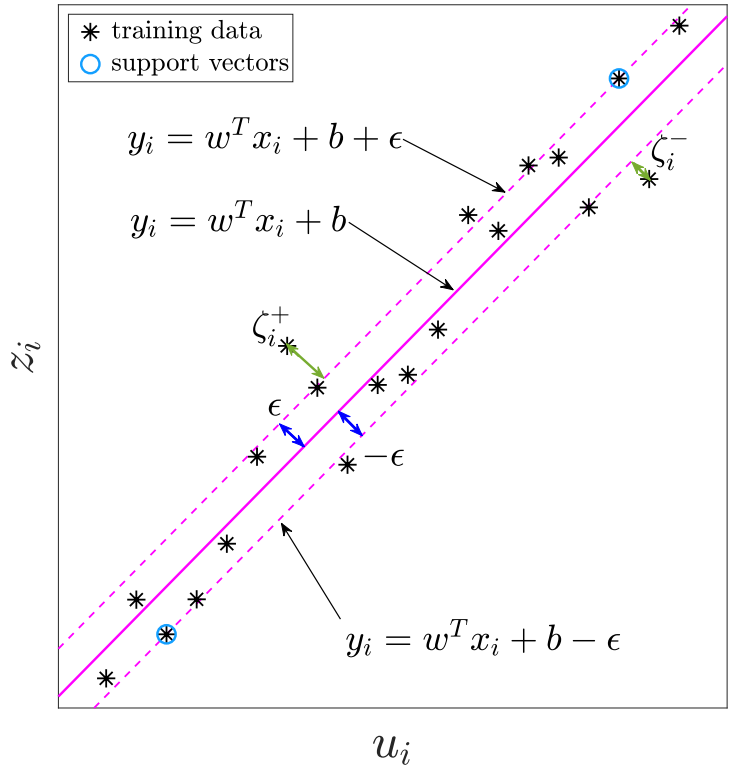


Figure 3.1: SVM regression and support vectors example

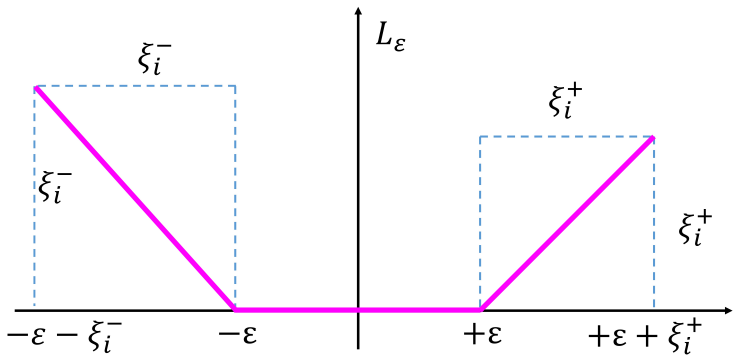


Figure 3.2: $\epsilon-$sensitive Loss function with slack variable based on [177]

3.1.2 Dual Optimization Problem and computing weights

To consider constraints of the convex optimization problem in Eq. 3.8, the Lagrangian function is calculated to change the convex optimization problem to a dual optimization problem as [177]:

$$\begin{aligned} L =& \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m}\left(\zeta_i^- + \zeta_i^+\right) - \sum_{i=1}^{m}\alpha_i^+(-\mathbf{z_i} + \mathbf{y_i} + \epsilon + \zeta_i^+) - \sum_{i=1}^{m}\mu_i^+\zeta_i^+ \\ & - \sum_{i=1}^{m}\alpha_i^-(\mathbf{z_i} - \mathbf{y_i} + \epsilon + \zeta_i^-) - \sum_{i=1}^{m}\mu_i^-\zeta_i^- \end{aligned} \tag{3.9}$$

where α_i^+, α_i^-, μ_i^+, and μ_i^- are Lagrangian Multipliers and $\alpha_i^+, \alpha_i^-, \mu_i^+, \mu_i^- \geq 0$. Based on the Saddle points condition, the partial differential of the Lagrangian function with respect to the optimization variables ($\mathbf{w}, \mathbf{b}, \zeta_i^+$, and ζ_i^-) must be equal to zero as [177]:

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \;\rightarrow\; w = \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-)\mathbf{u_i} \tag{3.10a}$$

$$\frac{\partial L}{\partial \mathbf{b}} = 0 \;\rightarrow\; \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-) = 0 \tag{3.10b}$$

$$\frac{\partial L}{\partial \zeta_i^+} = 0 \;\rightarrow\; \alpha_i^+ + \mu_i^+ = C \tag{3.10c}$$

$$\frac{\partial L}{\partial \zeta_i^-} = 0 \;\rightarrow\; \alpha_i^- + \mu_i^- = C \tag{3.10d}$$

where Eq. 3.10a is the support vector expansion, Eq. 3.10b is the bias constraints, and Eq. 3.10c and Eq. 3.10d are the box constraint. Based on the support vector

expansion, Eq. 3.1, the prediction function (model) can be rewritten using Eq. 3.10a as

$$\mathbf{y}(\mathbf{u}) = \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-)\mathbf{u_i}\mathbf{u} + \mathbf{b} \tag{3.11}$$

The dual optimization problem is obtained by substituting Eqs. 3.10a-3.10d into Eq. 3.9 as

$$\begin{aligned} \textbf{Minimize: } L =& \frac{1}{2}\sum_{i=1}^{m}\sum_{v=1}^{m}(\alpha_i^+ - \alpha_i^-)(\alpha_v^+ - \alpha_v^-)\mathbf{u_i}^T\mathbf{u_j} \\ & - \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-)\mathbf{z_i} + \epsilon\sum_{i=1}^{m}(\alpha_i^+ + \alpha_i^-) \\ \textbf{Subject to: } & \begin{cases} \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-) = 0 \\ 0 \le \alpha_i^+ \le C \\ 0 \le \alpha_i^- \le C \end{cases} \end{aligned} \tag{3.12}$$

Eq. 3.12 can be rewritten in a standard Quadratic Programming form (QP) [179]:

$$\begin{aligned} \textbf{Minimize: } & \frac{1}{2}\alpha^T\mathcal{H}\alpha + f^T\alpha \\ \textbf{Subject to: } & A_{eq}\alpha = B_{eq} \end{aligned} \tag{3.13}$$

where

$$\begin{gathered} \alpha = \begin{bmatrix} \alpha^+ \\ \alpha^- \end{bmatrix}, \quad \mathcal{H} = \begin{bmatrix} H & -H \\ -H & H \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} -\mathbf{z_i} + \epsilon \\ \mathbf{z_i} + \epsilon \end{bmatrix}, \\ H = \begin{bmatrix} \mathbf{u_i}^T\ \mathbf{u_j} \end{bmatrix}, \quad \mathbf{A_{eq}} = [1...1 \quad -1...-1], \quad \mathbf{B_{eq}} = [0] \end{gathered} \tag{3.14}$$

where $\mathbf{w}$ can be calculated by finding α (Solving Eq. 3.14) and substituting it into Eq. 3.10a. This fact shows that matrix $\mathbf{w}$ is calculated based on the linear combination of α and the training data.

3.1.3 Karush-Kuhn-Tucker (KKT) conditions and computing bias

Based on the KKT approach, the following equations must be fulfilled at the optimum point [180]:

$$\alpha_i^+(-\mathbf{z_i} + \mathbf{y_i} + \epsilon + \zeta_i^+) = 0 \tag{3.15a}$$

$$\alpha_i^-(\mathbf{z_i} - \mathbf{y_i} + \epsilon + \zeta_i^-) = 0 \tag{3.15b}$$

$$\mu_i^+\zeta_i^+ = (C - \alpha_i^+)\zeta_i^+ = 0 \tag{3.15c}$$

$$\mu_i^-\zeta_i^- = (C - \alpha_i^-)\zeta_i^- = 0 \tag{3.15d}$$

Considering Eq. 3.15, only the following five cases are possible:

$$\alpha_i^+ = \alpha_i^- = 0 \tag{3.16a}$$

$$0 < \alpha_i^+ < C,\ \alpha_i^- = 0 \tag{3.16b}$$

$$0 < \alpha_i^- < C,\ \alpha_i^+ = 0 \tag{3.16c}$$

$$\alpha_i^+ = C,\ \alpha_i^- = 0 \tag{3.16d}$$

$$\alpha_i^- = C,\ \alpha_i^+ = 0 \tag{3.16e}$$

For $|\mathbf{z_i} - \mathbf{y_i}|$ to be exactly equal to ϵ, only Eqs. 3.16b and 3.16c are necessary. So, the points of the training data which have $|\mathbf{z_i} - \mathbf{y_i}| = \epsilon$ are called support vectors (circled data points in Figure 3.1). Hence, the support vectors domain, S, is calculated as:

$$S = \{\ i \mid 0 < \alpha_i^-\ +\ \alpha_i^+ < C\} \tag{3.17}$$

where S is the index of the training data which form the SVM training algorithm support vectors. Accordingly, for the set of support vectors, $\mathbf{z_i}$ equals $\mathbf{y_i}\ +\ \text{sign}(\alpha_i^+\ -\ \alpha_i^-)\epsilon\ \ (i\ \in\ S)$. As a result, $\mathbf{b}$ is calculated as:

$$\mathbf{b} = \frac{1}{|S|}\sum_{i\in S}^{S}(\mathbf{z_i} - \mathbf{w}^T\mathbf{u_i} - \text{sign}(\alpha_i^+ - \alpha_i^-)\epsilon) \tag{3.18}$$

In summary, the convex problem (Eq. 3.8) is changed to the dual problem (Eq. 3.12). Then, by solving the quadratic programming, Eq. (3.13), and substituting it into the

support vector expansion, Eq. 3.10a, $\mathbf{w}$ is calculated. Then, vector $\mathbf{b}$ is calculated using Eq. 3.18 (KKT conditions). Finally, by substituting $\mathbf{w}$ and $\mathbf{b}$ into Eq. 3.1, the prediction model of a given data set ($\{\mathbf{u_i}, \mathbf{z_i}\}$) is found as:

$$\mathbf{y}(\mathbf{u}) = \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-)\mathbf{u_i}\mathbf{u} + \frac{1}{|S|}\sum_{i\in S}^{S}(\mathbf{z_i} - \mathbf{w}^T\mathbf{u_i} - \text{sign}(\alpha_i^+ - \alpha_i^-)\epsilon) \tag{3.19}$$

In this study, $\mathbf{y}(\mathbf{u})$ is used to predict steady-state diesel engine NO_x emission and BMEP. This function is used to predict the steady-state behaviour of the engine and will be denoted as $\mathbf{y_{ss}}(\mathbf{u})$ in subsequent sections.

3.2 Full-order Model (FOM)

The diesel engine model consists of three inputs and two outputs. The model inputs are the injected fuel amount m_f, engine speed n, and fuel rail pressure P_r. The model outputs are engine-out NO_x emission and BMEP. To provide the maximum model flexibility and minimize the model bias, the interactions of the primary features should also be considered. The number of resulting features depends on the highest order of interactions considered for the model. The number of total features is calculated based on the r-combination with repetitions formula $\frac{(l_o+r-1)!}{r!\ (l_o-1)!}$, where l_o is the number of original features (in our case $l_o = 3$), and r is the order of interactions [181]. So, the total number of features in a model of order r is equal to the sum of all the features with orders from 1 to r. The number of features for each interaction order is listed in Table 3.1.

The total number of experimental points used for training is 62 out of 84 data points. To simultaneously minimize the model bias and avoid overfitting, orders 1 to 4 of the original inputs and their interactions are considered as the base FOM model (34 features) to predict the steady-state values of NO_x and BMEP. The FOM features are listed in Table 3.2. The feature vector, $\mathbf{U_j}$, is defined using Table 3.2 as

$$\mathbf{U_l} = \{\mathbf{u_i}\}_\mathbf{j} \quad i = 1, ..., m, \quad l = 1, ..., 34 \tag{3.20}$$

Table 3.1: Number of features in each order from 1 to 6 using r-combination with repetitions formula

Order (r)	**r-combination with repetitions**	**Features number up to order r**
1	$\frac{(3+1-1)!}{1!\ 2!} = 3$	3
2	$\frac{(3+2-1)!}{2!\ 2!} = 6$	$3 + 6 = 9$
3	$\frac{(3+3-1)!}{3!\ 2!} = 10$	$10 + 9 = 19$
4	$\frac{(3+4-1)!}{4!\ 2!} = 15$	$15 + 19 = 34$
5	$\frac{(3+5-1)!}{5!\ 2!} = 21$	$34 + 21 = 55$
6	$\frac{(3+6-1)!}{6!\ 2!} = 28$	$55 + 28 = 83$

where m is the number of data points and l is the index number of the features.

As the dimensions and the range of features are quite different, all of the features must be normalized to improve the training performance [182]. Particularly, for SVMs, the training time can be significantly reduced by normalizing the features [183]. Here the rescaling or also called min-max normalization method is used to normalized feature for the SVM:

$$\bar{\mathbf{U}} = \frac{\mathbf{U} - \min(\mathbf{U})}{\max(\mathbf{U}) - \min(\mathbf{U})} \tag{3.21}$$

The system outputs vector is defined as

$$\mathbf{Z} = \{\mathbf{z_i}\} = [\mathbf{NO_{x,i}}\ \ \mathbf{BMEP_i}]^T \quad i = 1, ..., m \tag{3.22}$$

Then, the predicted steady-state NO_x and BMEP are:

$$\mathbf{y_{ss}} = [\mathbf{NO_{x,ss}}\ \ \mathbf{BMEP_{ss}}]^T \tag{3.23}$$

By solving the SVM algorithm for a given training data set, $\{\bar{\mathbf{U}}_l, \mathbf{Z}\}$, where $\bar{\mathbf{U}}_l$ and $\mathbf{Z}$ are calculated from Eq. 3.20 and Eq. 3.22, respectively, the approximate function, $\mathbf{y_{ss}}$ is obtained to predict the steady-state values of NO_x and BMEP. To cover a wide range of engine operating conditions, the diesel engine is run at 84 operating points,

Table 3.2: Features U_l of the Full-Order Model (FOM) of NO_x and BMEP

$U_1 = m_f$	$U_2 = n$	$U_3 = P_r$
$U_4 = m_f^2$	$U_5 = n^2$	$U_6 = P_r^2$
$U_7 = m_f n$	$U_8 = m_f P_r$	$U_9 = nP_r$
$U_{10} = m_f^3$	$U_{11} = n^3$	$U_{12} = P_r^3$
$U_{13} = m_f^2 n$	$U_{14} = m_f^2 P_r$	$U_{15} = (n^2)P_r$
$U_{16} = n^2 m_f$	$U_{17} = P_r^2 m_f$	$U_{18} = P_r^2 n$
$U_{19} = m_f n P_r$	$U_{20} = m_f^4$	$U_{21} = n^4$
$U_{22} = P_r^4$	$U_{23} = m_f^3 n$	$U_{24} = m_f^3 P_r$
$U_{25} = n^3 P_r$	$U_{26} = n^3 m_f$	$U_{27} = P_r^3 m_f$
$U_{28} = P_r^3 n$	$U_{29} = (m_f n)^2$	$U_{30} = (m_f P_r)^2$
$U_{31} = (nP_r)^2$	$U_{32} = P_r^2 n m_f$	$U_{33} = n^2 m_f P_r$
$U_{34} = m_f^2 P_r n$		

62 data points (74 %) are used as the training data, and 22 data points (26 %) are used to test the SVM learning algorithm. To find hyperparameters of SVM (C–trade-off between the model flatness and the tolerated error), 15% of the training data set (9 points of 62 training points) are selected randomly and used for cross-validation. To find the best regulatory parameter C of the FOM for both NO_x and BMEP, the effect of varying C on the squared correlation coefficient (R^2), the maximum error between prediction and actual data (E_{max}), and the cost function ($J(E_{max}, R^2)$) for both the training and test data are analysed. The proposed cost function to find C is defined as

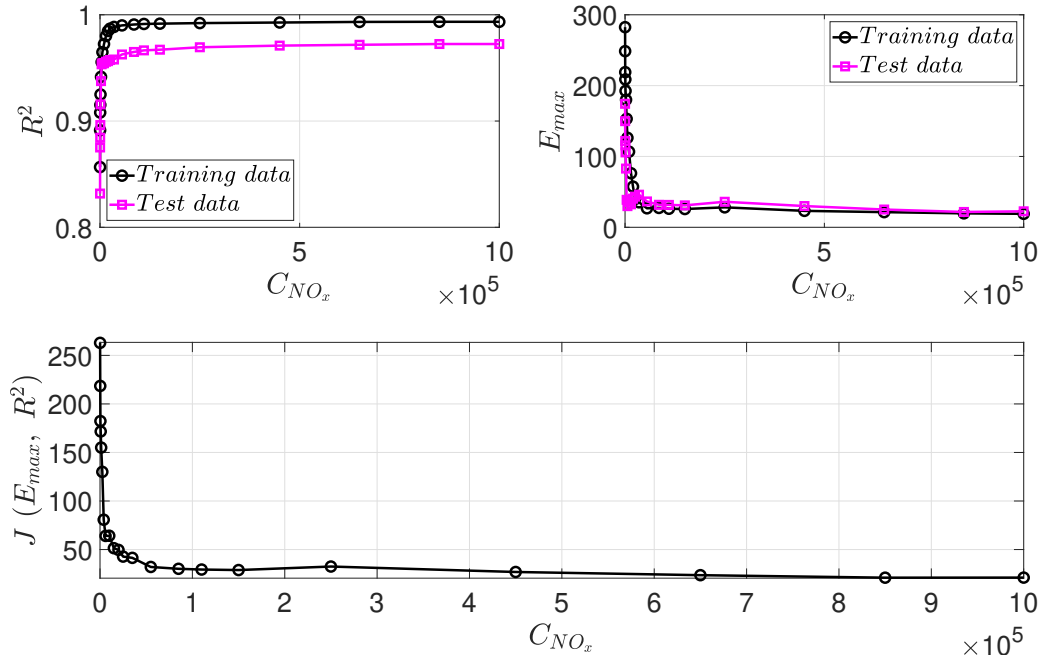
$$J(E_{max}, R^2) = \sqrt{\frac{E_{max,tr}\ E_{max,ts}}{R_{tr}^2\ R_{ts}^2}} \tag{3.24}$$

where $E_{max,tr}$ and $E_{max,ts}$ are the maximum errors between the prediction and the actual data for the training and test data sets respectively. Also, R_{tr}^2 and R_{ts}^2 are the squared correlation coefficients for the training and test data sets, respectively. The goal is to increase C and minimize the maximum error and maximize the squared

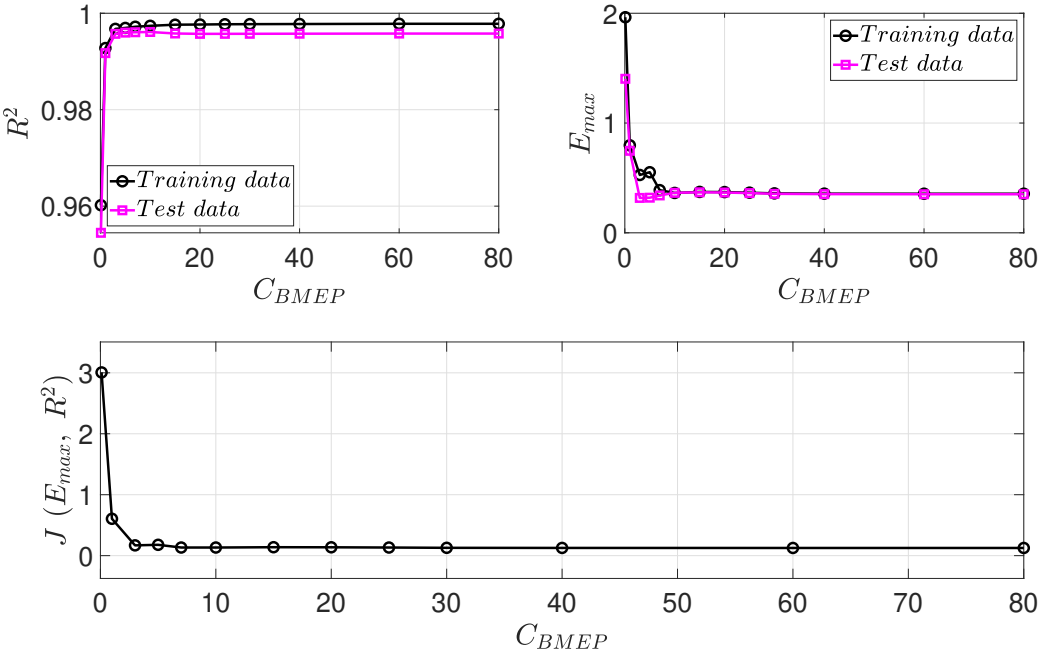
correlation coefficients for both the training and test data. Therefore, the best C for modeling is obtained by minimizing $J(E_{max}, R^2)$. In this section, cross-validation data are used in Eq. 3.24 to find the regulatory parameter C. The squared correlation coefficient R^2, the maximum error between the prediction and the actual data E_{max} and the cost function $J(E_{max}, R^2)$ with respect to the regulatory parameter C for training, cross-validation, and the test data set of the FOM NO_x and BMEP model are shown in Figure 3.3. The regulatory parameter, C, is a trade-off between the model flatness and the tolerated error. Based on the results shown in Figure 3.3, the prediction error increases by decreasing C.

The squared correlation coefficient (R^2) is used to quantify the model's accuracy. The maximum error between the prediction and the actual data for both the training and cross-validation data decrease as the regulatory parameter C increases, resulting in a decrease in $J(E_{max}, R^2)$. After C reaches a certain value of C_o, the model performance enhancement levels off since the squared correlation coefficient and the maximum error for all data are saturated. By increasing C to more than C_o, the model performance remains unchanged, but the model flatness decreases, i.e., the overfitting probability has increased. As a result, the model is less robust for new test data due to possible overfitting. By setting $C = C_o$, the model performance is maximized while overfitting constraints are fulfilled. To ensure that all the important features are considered when minimizing the slack variables in the optimization problem, a sufficiently large value of regulatory parameter C must be selected. Based on Figure 3.3, C_o for NO_x and BMEP are selected to be $C_{o,NO_x} = 85000$ and $C_{o,bmep} = 60$, respectively as increasing C has no effect in accuracy and these value is sufficiently large. The prediction versus the actual value for FOM NO_x and BMEP are shown in Figure 3.4. Here, the cross-validation portion of the training data are shown for both NO_x and BMEP; however, to reduce the complexity of figures, for the rest of the thesis, combined cross-validation and training data is illustrated as training data. It should be noted that the regulatory parameter remains constant throughout the

MOR process.

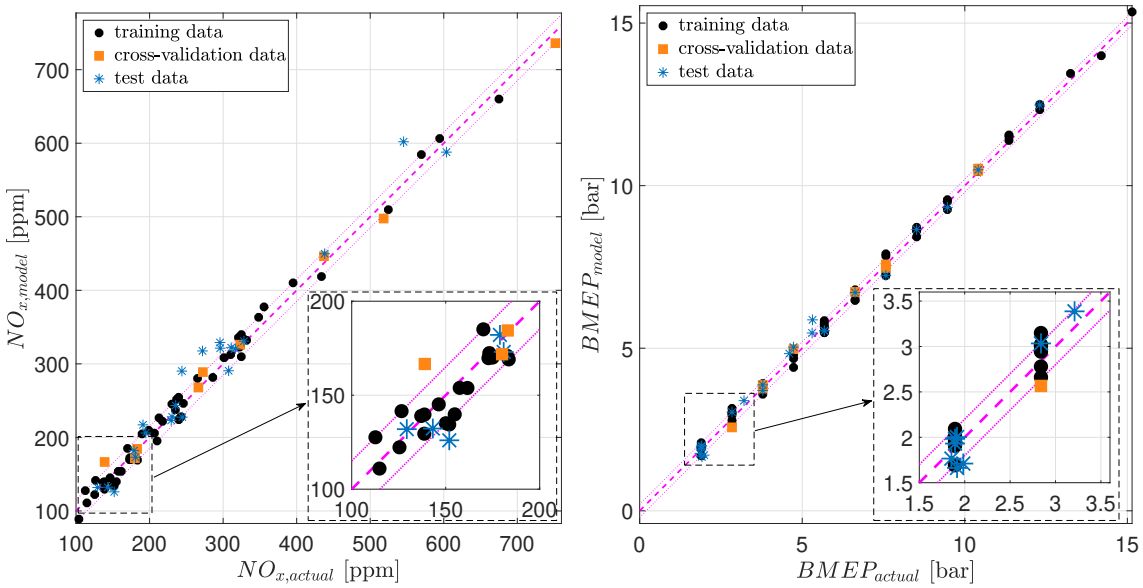


(a) NO_x



(b) BMEP

Figure 3.3: Maximum error (E_{max}), correlation coefficient (R^2), and cost function ($J(E_{max}, R^2)$) vs regulatory parameter C for a) NO_x b) BMEP



(a) Prediction vs Actual for NO_x FOM (b) Prediction vs Actual for BMEP FOM

Figure 3.4: Prediction vs actual data for NO_x and BMEP FOM

3.3 Model Order Reduction (MOR) Algorithm

Next, using the proposed FOM the regression matrices **w** and **b** are obtained using SVM. The best C value for the SVM problem, C_o, is found in the previous section. In this section, the MOR algorithm is proposed to reduce the order of NO_x and BMEP steady-state FOM. MOR helps to achieve an appropriate model by removing redundant features and selecting the important ones. The flow chart in Figure 3.5 shows the Nonlinear Reduced Control-Oriented Model (NRCOM). Starting from FOM with 34 features,**w** and **b** are calculated for a given data set of (m_f, n, P_r) as the inputs and $(BMEP, NO_x)$ as the targets. Then, the value of **w** is evaluated for each feature. After that, the feature for which the **w** array has the minimum value is removed. Then, the SVM algorithm for regression is solved for a new set of features. As a result of MOR algorithm for NO_x and BMEP, two types of models are proposed as:

1. High-Order Model (HOM): For the HOM, the priority is model accuracy rather than the number of features and the computation time. Therefore, only the

unnecessary features of FOM are removed. The HOM model can be used in applications that require high accuracy such as developing a NO_x sensor fault-detection algorithm or virtual plants to evaluate a controller in simulation.

2. Low-Order Mode (LOM): For the LOM, the number of features and the computation time must be balanced with model accuracy. The objective is to find a simple model with fewer features and an acceptable accuracy. As the LOM has a simple structure and acceptable accuracy, it is useful for designing a controller [184].

As shown in Figure 3.5, the features of HOM, m_{HOM}, are selected in a way that $J_l(R^2, E_{max})$ is minimized since the main objective of model order reduction for HOM is to maximize the model accuracy by removing the redundant features, with no concern for reducing the size of the model. However, reducing the model size while keeping the accuracy acceptable was the objective for model order reduction to the LOM. To avoid significant loss of the model accuracy, the least significant features are removed one by one until the relative difference between cost functions J_l and J_{l-1} exceeds the acceptable threshold. Then the corresponding feature number becomes the number of features of the LOM, m_{LOM}. The relative difference between J_l and J_{l-1} is defined as:

$$d_r(J_l, J_{l-1}) = \frac{|J_l - J_{l-1}|}{\max(J_l, J_{l-1})} \times 100 \tag{3.25}$$

In this study, the LOM is found by defining $d_r = 25\%$ threshold. In other words, starting MOR from the initial features, as soon as the relative difference between J_l and J_{l-1} exceeds 0.25, the corresponding l is considered as LOM features set, l_{LOM}.

3.3.1 NOx steady State Model

The squared correlation coefficient (R^2) and the maximum error between prediction and actual data (E_{max}) for both the training and test data and defined cost function ($J(R^2, E_{max})$) with respect to the number of features are shown in Figure 3.6.

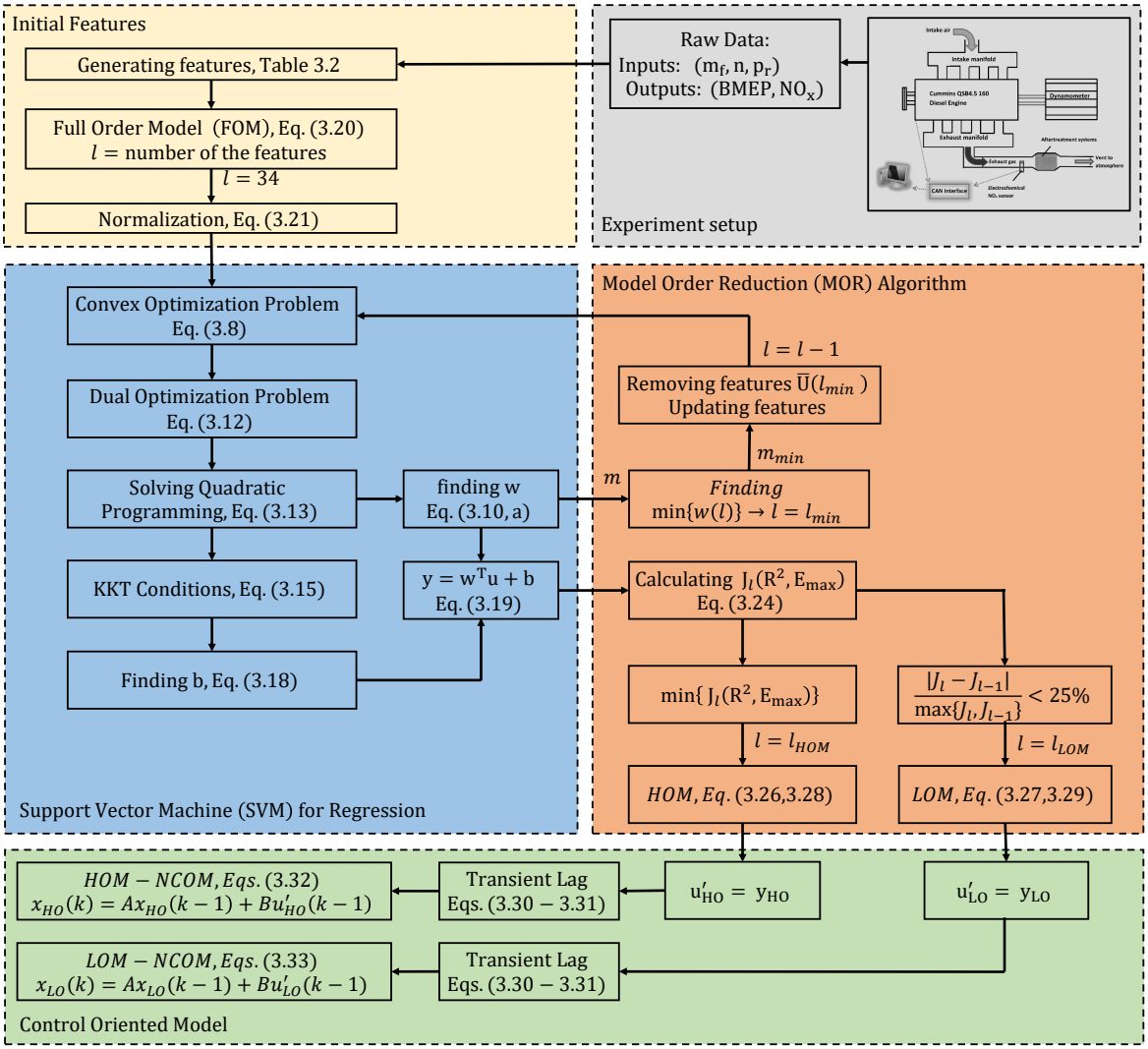


Figure 3.5: Control Oriented Model (COM) development and SVM-based MOR algorithm

Based on Figure 3.6, since $\min\{J(R^2, E_{\max})\}$ is achieved for a 29-feature model ($m_{HOM,NO_x} = 29$). These models with these 29 features is chosen as the HOM NO_x. In other words, the 29-feature model is chosen as the HOM because it has the highest accuracy among all the models studied. Tracking $J_l - J_{l-1}$ as a function of l in Figure 3.6 by starting from $l = 34$, the first relative difference larger than 25% occurs for a 9-feature model ($l_{LOM,NO_x} = 9$). The model with 9 features is chosen as the LOM for NO_x. The model with 9 features is chosen as LOM because by decreasing the model features to less than 9, there is a significant reduction in model performance (J_l).

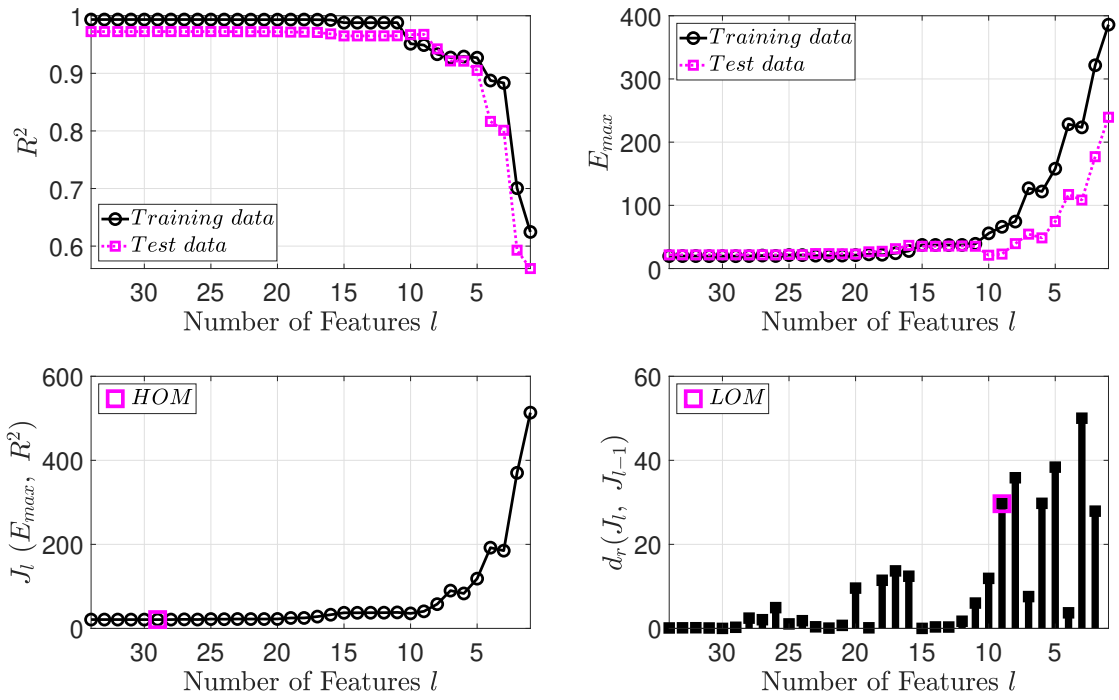


Figure 3.6: Maximum error (R^2), squared correlation coefficient (R^2), and cost function ($J(E_{max}, R^2)$) vs number of features of prediction function for steady-state NO_x prediction

As ANN is widely used for engine performance and emission modeling, the SVM model for all of the developed models (FOM, HOM, and LOM) are compared with an ANN using the same set of features. This provide a standard to compare these results to an ANN. Here, a two-layer (one hidden layer and one output layer) feed-forward backpropagation network with three neurons in the hidden layer is employed, and the Levenberg-Marquardt training method is used to train the model which has a relatively fast convergence [97]. The selection of hidden layer and neurons number was based on similar ANN-based studies in the literature. To make sure that the number of neurons are compatible with the size of the data set, three neurons are considered for the hidden layer as proposed by a similar study with a similar data size [97, 185]. The same training, cross validation, and test data set are used for the SVM and the ANN models. Both algorithms use 15% of the training data set to find the model hyperparameters.

The maximum error between the prediction and the actual training data set, squared

correlation coefficient, the defined cost function, and training time for both the SVM and ANN training methods are listed in Table 3.3. The results reveal that, the SVM model has a shorter training time and a more accurate model (larger squared correlation coefficient and smaller maximum error between actual and model), especially for the test data. This is partly because ANN uses a gradient descent algorithm for training which increases the risk of converging to local minima. Additionally, the risk of overfitting is higher for ANN for the same size of training data [122]. This problem is also shown in the results where the squared correlation coefficient of test data for ANN is less than the SVM model. Since the training time for SVM is significantly less than for ANN, it is more suitable for real-time online learning applications where the model is training while running and collecting data. Another benefit of using this SVM is that the model is far simpler to explain mathematically in the form of an equation, especially when a linear kernel is used. When using the linear kernel, the SVM model is defined based on the vector $\mathbf{w}$ with a bias $\mathbf{b}$.

It should be noted that the performance of HOM is even better than FOM as a result of removing unnecessary features that affect the flatness of the SVM algorithm. Based on Table 3.3, the accuracy of LOM is acceptable as the error is below the defined threshold.

Thus, the HOM and LOM features are (see Table 3.2)

$$\bar{\mathbf{U}}_{\mathbf{NO_x,HO}} = \bar{\mathbf{U}}_l \qquad l = 1-9, 11-15, 17, 19-27, 29, 31-34 \tag{3.26}$$

$$\bar{\mathbf{U}}_{\mathbf{NO_x,LO}} = \bar{\mathbf{U}}_l \qquad l = 2, 5, 8, 15, 21, 22, 27, 32, 33 \tag{3.27}$$

where l is the feature index. By solving the SVM algorithm for NO_x, the features of HOM and LOM are obtained. The predicted steady-state NO_x vs the actual value for both the high-order and low-order steady-state NO_x model is shown in Figure 3.7. Based on Figure 3.7-(a), most of the test and the training data are within the defined tolerance ϵ for the HOM of NO_x. However, as shown in Figure 3.7-(b), the accuracy

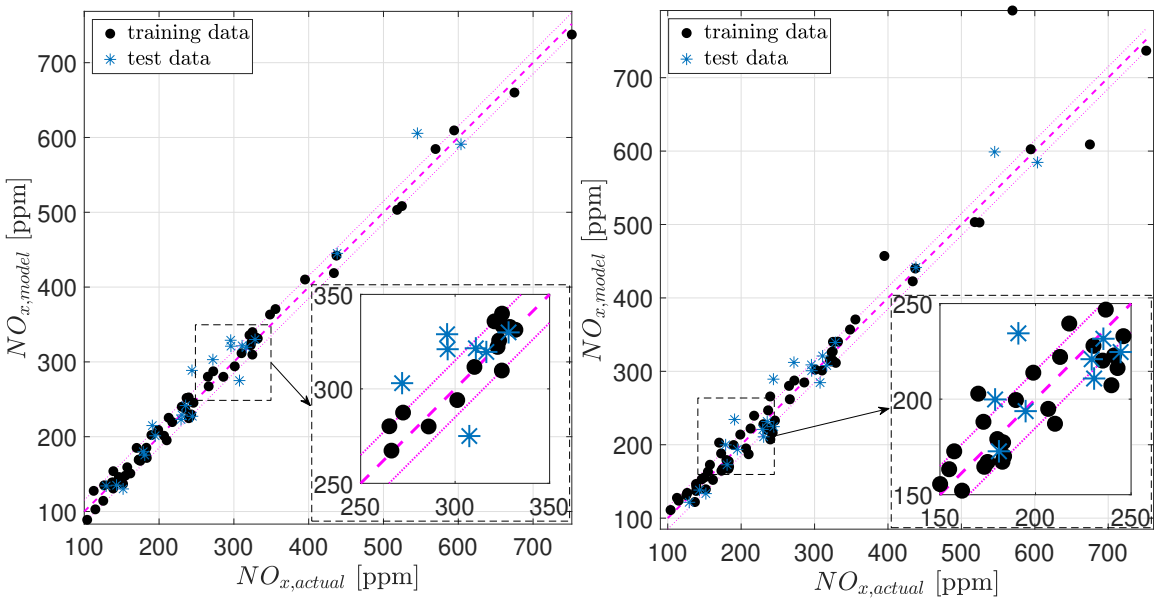
Table 3.3: Performance of the NO_x FOM, HOM, and LOM

Model Type	**FOM**		**HOM**		**LOM**	
No. of Features (l)	34		29		9	
Training Method	**SVM**	**ANN**	**SVM**	**ANN**	**SVM**	**ANN**
$E_{max,tr}$ $[ppm]$	19.7	25.6	19.6	27.3	66.0	57.9
$E_{max,ts}$ $[ppm]$	21.7	60.7	21.7	47.8	22.9	60.3
R^2_{tr}	0.993	0.997	0.993	0.984	0.949	0.989
R^2_{ts}	0.972	0.978	0.973	0967	0.968	0.976
$J(E_{max}, R^2)$ $[ppm]$	21.0	39.9	20.9	37.1	40.6	54.7
Training Time $[ms]$	9.5	240.6	11.1	202.0	13.1	194.5

of the LOM is not consistent throughout all data points for both training and test points and the number of outliers are greater than the HOM.

3.3.2 BMEP steady state Model

Similar to the NO_x steady-state model, the BMEP reduced steady-state model is obtained. The squared correlations coefficient (R^2) and maximum error between prediction and actual data (E_{max}) for both the training and test data and defined cost function ($J(R^2, E_{max})$) with respect to the number of features are shown in Figure 3.8. Based on Figure 3.8, a 20-feature model ($l_{HOM,BMEP} = 20$) and a 6-feature model ($l_{LOM,BMEP} = 6$) are chosen as the HOM and LOM of BMEP, respectively. The maximum error between the prediction and the actual data (E_{max}), the squared correlation coefficient (R^2), cost function ($J(E_{max}, R^2)$), and training time for ANN and SVM training methods are listed in Table 3.4. Similar to the NO_x model, for all the BMEP models, the SVM has faster training and more accurate response compared to the ANN especially for the test data. The general performance of HOM is acceptable with respect to the FOM, while it has a simpler structure. A 6-feature model is chosen as LOM using the same criteria as before (Eq. 3.25). As shown in Table 3.4, the accuracy of the model is acceptable, and by reducing the model further, the



(a) HOM of NO_x (with 29 features)

(b) LOM of NO_x (with 9 features)

Figure 3.7: Prediction vs actual data for the LOM and the HOM of NO_x

model accuracy is no longer acceptable.

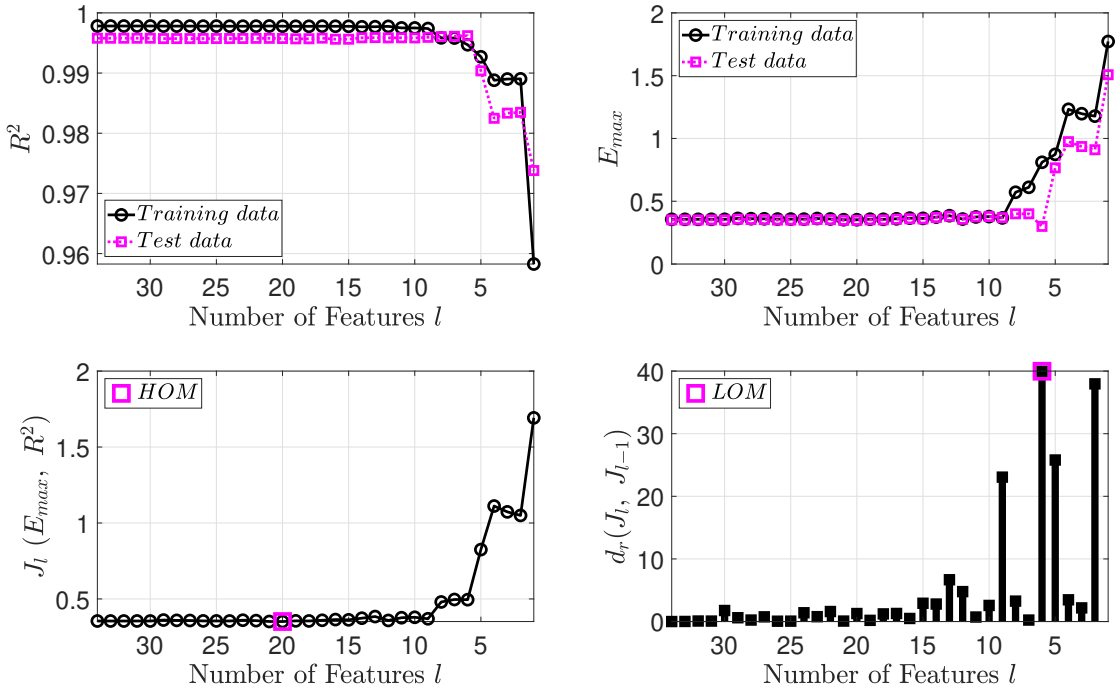


Figure 3.8: Maximum error (R^2), squared correlation coefficient (R^2), and cost function ($J(E_{max}, R^2)$) vs number of features of prediction function for steady-state BMEP prediction

Table 3.4: Performance of the BMEP Full-Order Model (FOM), High-Order Model (HOM), and Low-Order Model (LOM)

Model Type	**FOM**		**HOM**		**LOM**	
No. of Features	34		20		6	
Training Method	**SVM**	**ANN**	**SVM**	**ANN**	**SVM**	**ANN**
$E_{max,tr}$ $[ppm]$	0.36	0.40	0.35	0.39	0.81	0.54
$E_{max,ts}$ $[ppm]$	0.35	0.45	0.35	0.42	0.30	0.47
R^2_{tr}	0.998	0.999	0.998	0.995	0.995	0.996
R^2_{ts}	0.996	0.996	0.996	0.996	0.996	0.996
$J(E_{max}, R^2)$ $[ppm]$	0.3	0.4	0.3	0.4	0.5	0.5
Training Time $[ms]$	35.9	199.8	9.2	218.0	9.5	214.7

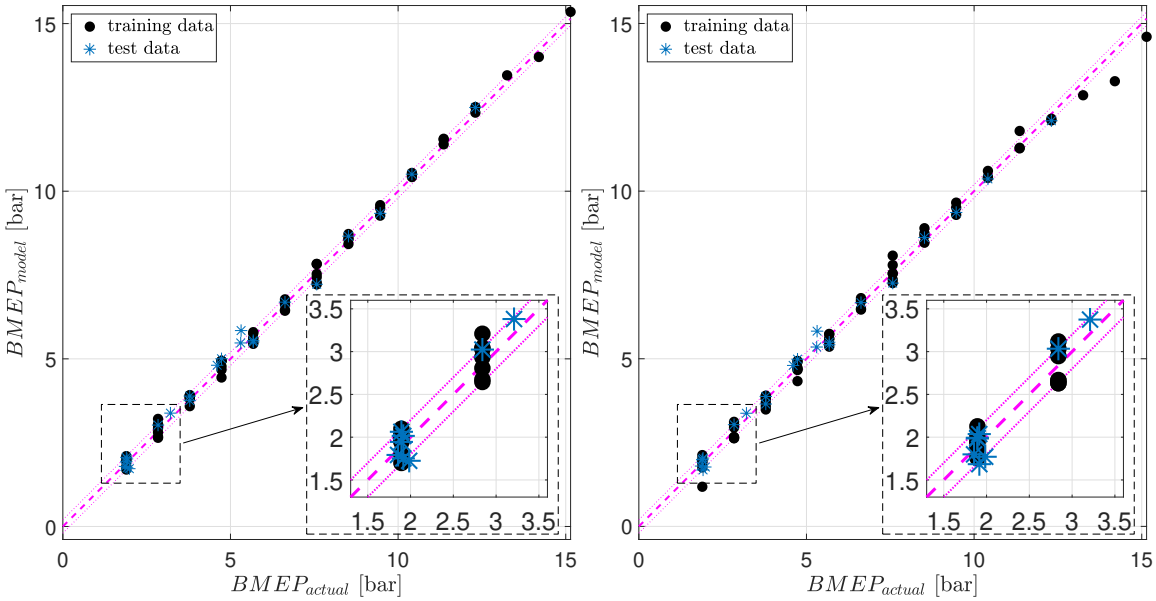
Thus, the HOM and LOM features are obtained as:

$$\bar{\mathbf{U}}_{\mathbf{BMEP,HO}} = \bar{\mathbf{U}}_l \qquad l = 1, 4, 7-10, 17-21, 24-27, 29-32, 34 \tag{3.28}$$

$$\bar{\mathbf{U}}_{\mathbf{BMEP,LO}} = \bar{\mathbf{U}}_l \qquad l = 1, 18, 21, 25, 27, 30 \tag{3.29}$$

where l is the feature index. By solving the SVM algorithm for BMEP, the HOM and the LOM are achieved. The predicted steady-state BMEP with respect to the actual value for both of the high-order and the low-order steady-state BMEP models are shown in Figure 3.9. As shown in Table 3.4, the HOM and LOM have an acceptable accuracy while the HOM has a higher accuracy than LOM. However, the LOM of BMEP has only 6 features, which makes it a simple model that requires a low computational effort. Most of the test and training data for both the HOM and LOM of BMEP are within the defined tolerance ϵ, as shown in Figure 3.9. This means that the MOR improves in the accuracy of the FOM by removing its unnecessary features.

One important observation from the training time (Table 3.3 and Table 3.4) is



(a) of BMEP (with 20 features) (b) LOM of BMEP (with 6 features)

Figure 3.9: Prediction vs actual data for HOM and LOM of BMEP

that by increasing the number of features in the ANN, the training time is increased. However, this trend inverses in the SVM such that by decreasing the number of the features, the training time increases. This behavior results in reducing the overall training time due to the time saved by reducing the number of training iterations [186]. This trend appears in the HOM and LOM models.

3.4 Control Oriented Model (COM)

The model described in the previous section is used to determine steady state NO_x and BMEP. Now a simple first order dynamic model for transient operation will be defined. To derive the discrete-time dynamic COM, the NO_x concentration at step k for a sampling interval of T, is calculated as follows:

$$NO_x(k) = (1 - \frac{T}{\tau_{NOx} + T})NO_x(k-1) + \frac{T}{\tau_{NOx} + T}NO_{x,ss}(k-1) \tag{3.30}$$

and the BMEP at step k is calculated using the following equation:

$$BMEP(k) = (1 - \frac{T}{\tau_{BMEP} + T})BMEP(k-1) + \frac{T}{\tau_{BMEP} + T} BMEP_{ss}(k-1) \tag{3.31}$$

where $NO_{x,ss}(k-1)$ and $BMEP_{ss}(k-1)$ are the steady state NO_x and BMEP. The sample interval is τ and k is the sample time and τ_{NOx} and τ_{BMEP} are the time constants for NO_x and BMEP respectively, which are estimated based on the experimental data and are found to be 1 and 0.2 seconds, for NO_x and BMEP respectively [187]. The state space of the COM for both high-order and low-order models can be defined as:

$$\mathbf{x_{HO}}(k) = \mathbf{A}\mathbf{x_{HO}}(k-1) + \mathbf{B}\hat{\mathbf{u}}_{\mathbf{HO}}(\mathbf{k-1}) \tag{3.32}$$

$$\mathbf{x_{LO}}(k) = \mathbf{A}\mathbf{x_{LO}}(k-1) + \mathbf{B}\hat{\mathbf{u}}_{\mathbf{LO}}(\mathbf{k-1}) \tag{3.33}$$

where vector $\mathbf{x}$(K) contains two model states:

$$\mathbf{x_{HO}}(\mathbf{k}) = \begin{bmatrix} NO_{x,HO}(k) & BMEP_{HO}(k) \end{bmatrix}^T \tag{3.34}$$

$$\mathbf{x_{LO}}(\mathbf{k}) = \begin{bmatrix} NO_{x,LO}(k) & BMEP_{LO}(k) \end{bmatrix}^T \tag{3.35}$$

and vector $\hat{\mathbf{u}}(\mathbf{k})$ is calculated as

$$\hat{\mathbf{u}}_{\mathbf{HO}}(\mathbf{k}) = \begin{bmatrix} NO_{x,HO,ss} \\ BMEP_{HO,ss} \end{bmatrix} \tag{3.36}$$

$$\hat{\mathbf{u}}_{\mathbf{LO}}(\mathbf{k}) = \begin{bmatrix} NO_{x,LO,ss} \\ BMEP_{LO,ss} \end{bmatrix} \tag{3.37}$$

where $NO_{x,HO,ss}$, $BMEP_{HO,ss}$, $NO_{x,LO,ss}$, and $BMEP_{LO,ss}$ are listed in Appendices A and B. The vector $\mathbf{y}$ contains two model outputs:

$$\mathbf{y}(\mathbf{k}) = \begin{bmatrix} x_1(k) & x_2(k) \end{bmatrix} \tag{3.38}$$

Matrices **A** and **B** are:

$$\mathbf{A} = \begin{bmatrix} 1 - \frac{T}{\tau_{NOx}+T} & 0 \\ 0 & 1 - \frac{T}{\tau_{BMEP}+T} \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} \frac{T}{\tau_{NOx}+T} & 0 \\ 0 & \frac{T}{\tau_{BMEP}+T} \end{bmatrix} \tag{3.39}$$

Therefore, the HOM and LOM Nonlinear Control Oriented Models (NCOM) for NO_x and BMEP are obtained as Eqs. 3.31 and 3.30. The open-loop response for the HOM-NCOM and LOM-NCOM for NO_x and BMEP at four different engine speeds of 1250, 1500, 1750, and 2000 rpm are shown in Figures 3.10, 3.11, 3.12, and 3.13. In all cases, P_r and m_f are the system inputs and are applied to both the HOM-NCOM and LOM-NCOM. In each transient test, the engine speeds remains constant (with a 10 rpm tolerance). In all of these plots the open-loop response of both the HOM-NCOM and LOM-NCOM based on the model vs actual measurements are shown. Figure 3.9 shows that, for BMEP both the HOM-NCOM and LOM-NCOM follow the experimental responds closely as expected. However, the NO_x response for LOM-NCOM has different accuracies at different engine speeds. For instance, in Figure 3.13 and Figure 3.11, the LOM-NCOM NO_x response is less accurate than in the HOM model. Nonetheless, the NO_x response for the HOM-NCOM is accurate at all of speeds studied. The HOM-NCOM model has an accurate response over a wide range of engine operating points and is an accurate model for possible use as a virtual plant to simulate the designed controller before implementation in a real-time system. Additionally, it can be used as an accurate model for an NO_x sensor fault-detecting algorithm. As the LOM-NCOM has a simple structure, it is quite suitable for designing a model-based robust controller and is also capable of predicting samples ahead based on the system's current states and inputs. A robust controller can be

used to overcome the model mismatch between the LOM-NCOM and HOM-NCOM.

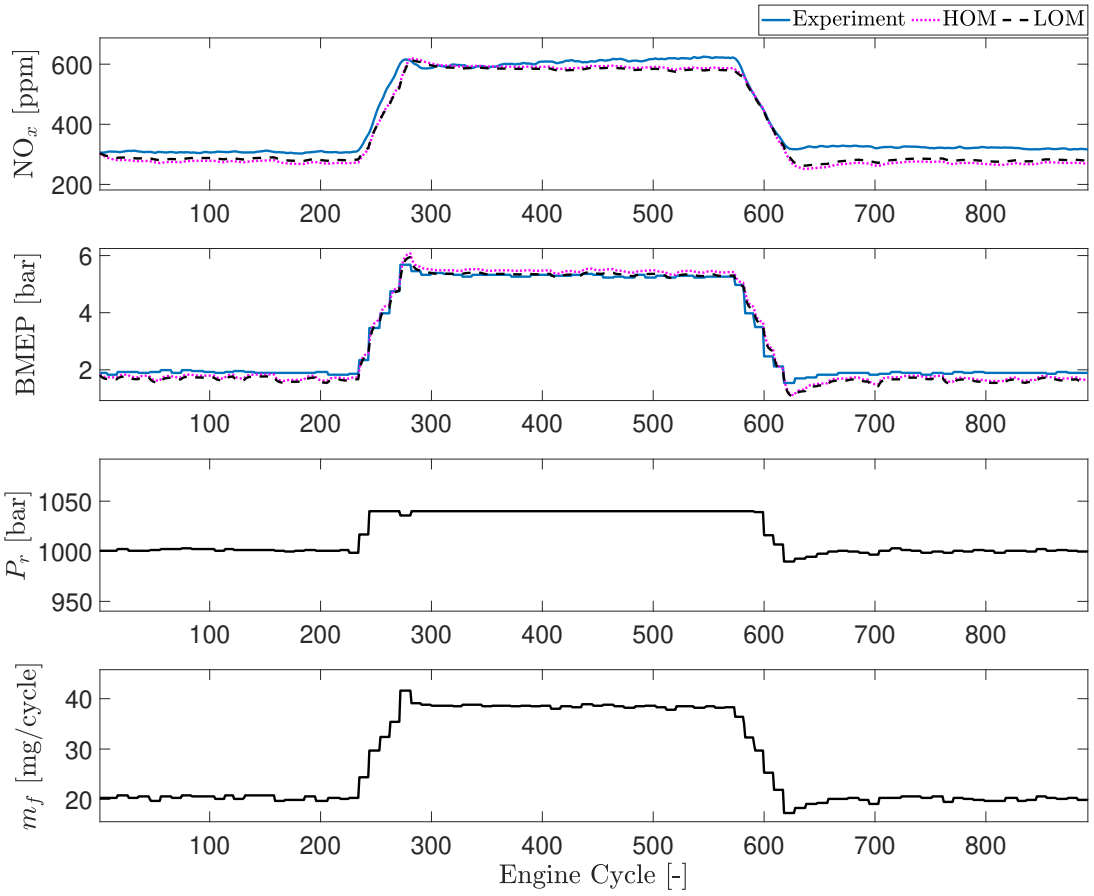


Figure 3.10: Transient response at engine speed = 1250 *rpm*

3.5 Summary of chapter

In this chapter, an MOR algorithm is developed using an SVM approach to predict the steady-state NO_x and BMEP of a medium-duty diesel engine. Based on the proposed SVM-based MOR algorithm and starting with a 34-feature FOM, an HOM and an LOM are developed to predict the steady-state NO_x emission and BMEP. The features of the models are calculated based on orders 1 to 4 of the main model inputs and their interactions. In this thesis 74% of experimental data is used to train the steady-state NO_x and BMEP , and 26% is used as test data. The model

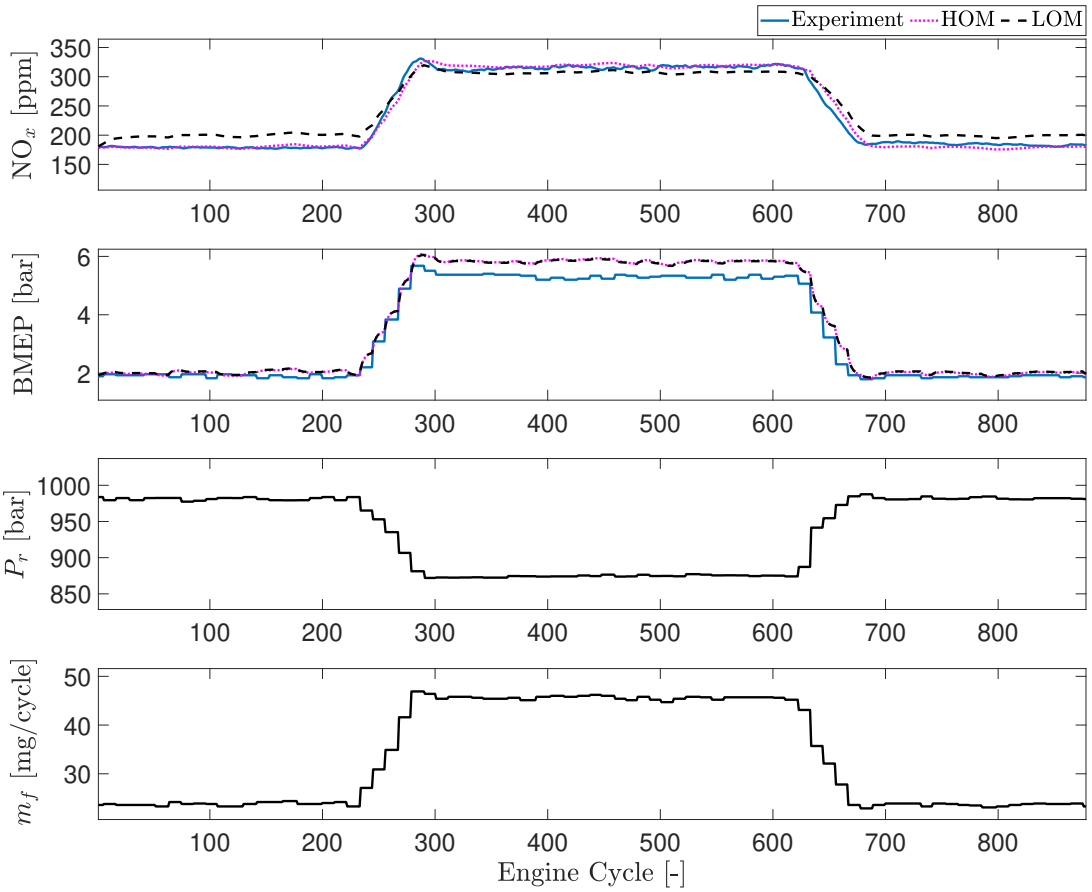


Figure 3.11: Transient response at engine speed = 1500 *rpm*

inputs are engine speed, injected fuel amount, and fuel rail pressure. The results of the steady-state model show that the HOM model has an accurate prediction but a more complex structure with 29 features for NO_x and 20 features for BMEP. For the steady-state NO_x model, the squared correlation coefficient of the test (R^2_{ts}) is equal to 0.9724, 0.9725, and 0.9677 for the FOM, HOM, and LOM, respectively. The R^2_{ts} value is equal to 0.9957, 0.9957, and 0.9962 for the FOM, HOM, and LOM, respectively for the BMEP steady-state model. Consequently, by removing unnecessary features based on the SVM-based MOR algorithm, the HOM performance for both NO_x and BMEP is enhanced while the HOM complexity decreases 27.9 % with respect to the FOM. The LOM model has an acceptable accuracy with a squared correlation

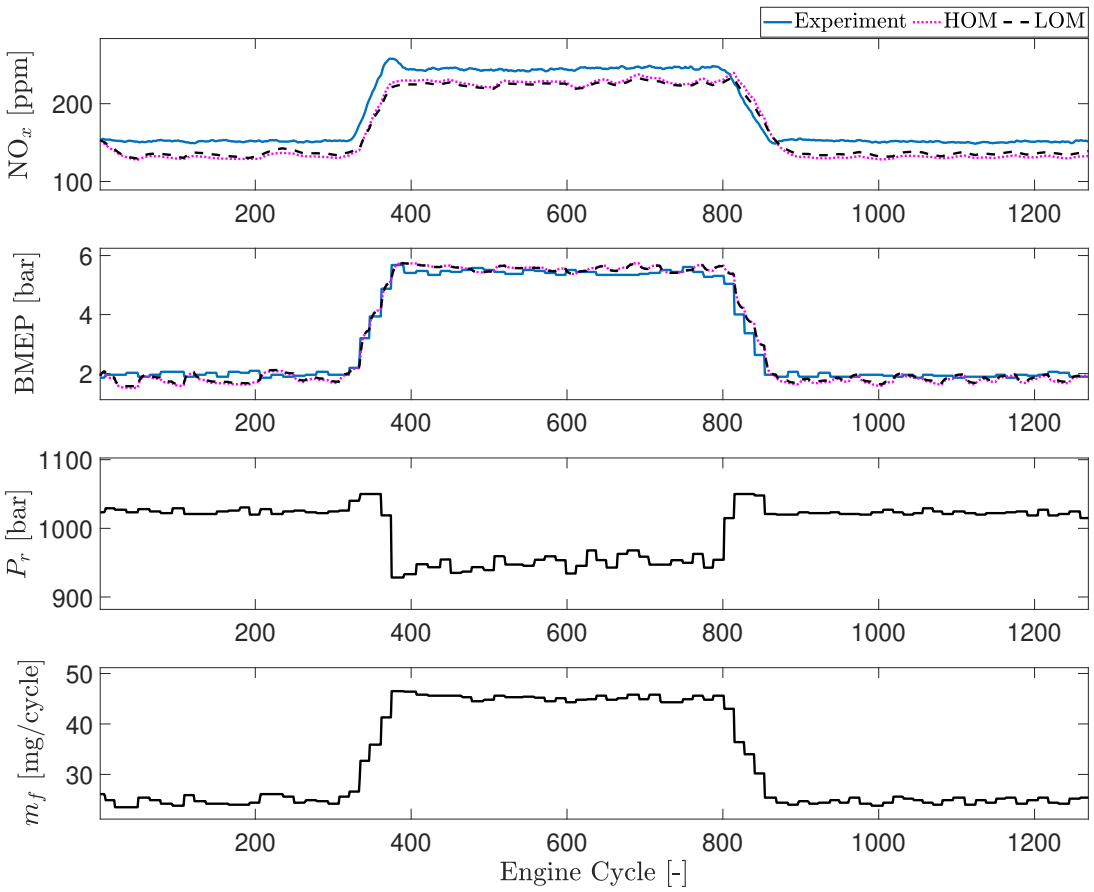


Figure 3.12: Transient response at engine speed = 1750 *rpm*

coefficient of 0.9393 for NO_x and 0.9961 for BMEP while it has 77.9 % and 69.4 % fewer features with respect to the FOM and HOM, respectively. All of the FOM, HOM, and LOM SVM models of NO_x and BMEP are compared with an ANN, and the results show shorter training time and more accurate results in the test data for the SVM models compared to the ANN. The SVM model training are at least 5 to 14 times faster than the corresponding ANN models with the same set of features for NO_x and BMEP respectively. In addition, the use of a linear kernel in the SVM make it more suitable for real-time applications and for COMs.

Then, a nonlinear control-oriented model (NCOM) is developed based on the developed SVM models to predict the transient behavior of the system. A fast response

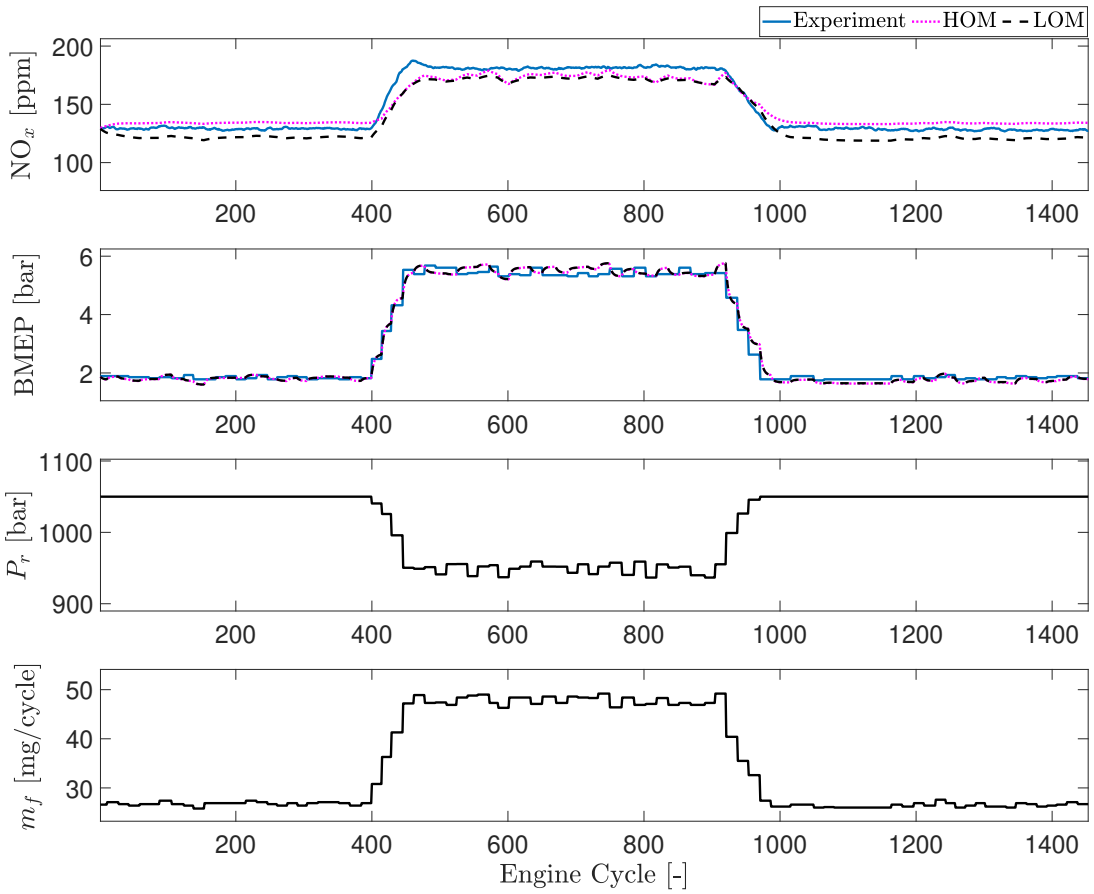


Figure 3.13: Transient response at engine speed = 2000 *rpm*

electrochemical NO_x sensor is used to verify the transient response of the NCOM. The transient results of HOM and LOM are compared to experimental data showing an accurate and robust prediction of engine BMEP at different engine speeds for rising and falling step changes of the fuel rail pressure and the injected fuel amount for HOM. Additionally, the LOM model has an accurate response at different speeds for BMEP; however, the NO_x prediction with LOM has varying accuracy at different engine speeds. It can be concluded that the HOM can predict NO_x and BMEP over a wide range of operating points, which makes it ideal to be used as a virtual plant for fault detection purposes. The LOM has a simpler structure, and an acceptable accuracy which makes it useful in designing a model-based robust controller such as

sliding mode controller.

Chapter 4

Steady-state Particle Matter (soot) Gray-box Modeling [1]

In this chapter, a detailed analysis of diesel engine soot emissions modeling for control applications is presented. Physical, black-box, and gray-box models are developed for soot emissions prediction. Additionally, different feature sets based on the least absolute shrinkage and selection operator (LASSO) feature selection method and physical knowledge are examined to develop computationally efficient soot models with good precision. The physical model is a virtual engine modeled in GT-Power© software that is parameterized using a portion of experimental data. Different ML methods, including Regression Tree (RT), Ensemble of Regression Trees (ERT), Support Vector Machines (SVM), Gaussian Process Regression (GPR), Artificial Neural Network (ANN), and Bayesian Neural Network (BNN) are used to develop the black-box models. The gray-box models include a combination of the physical and black-box models. A total of five feature sets and eight different ML methods are tested. An analysis of the accuracy, training time and test time of the models is performed using the K-means clustering algorithm. The analysis provides a systematic way to categorize the feature sets and methods based on their performance and to select the best method for a specific application.

[1] This chapter is based on [5]

4.1 Gray-Box, Black-Box, and White-Box modeling

The physical model, black-box, and gray-box are described in this section. The first step toward developing physical and gray-box models was to develop a physical model for combustion, which is described in Chapter 2. To compare with black-box and gray-box models, a physical-based model as a white-box model is developed using the Hiroyasu model [188]. The model is calibrated using 8% of the experimental data. The oxidation multiplier and formation multiplier are the two parameters of the model used in the calibration process. The calibration process follows the ESM development process, which uses the Genetic Algorithm (GA) NSGA-III [172] for multi-objective Pareto optimization as the search algorithm [172]. The two key inputs for GA are the population size and number of generations. A population size of 16 is chosen while the number of generations for soot model calibration are 10.

The process of selecting important features from a feature set is called feature selection (FS). FS reduces the size of the input feature set, which results in improving ML method performance. FS process is depicted schematically in Figure 4.1. A total of five feature sets are used in this study to simulate soot emissions. For FS in this work, a combination of physical insight and LASSO FS technique is used. For physical insight FS, the most significant features are selected based on expert prior knowledge while LASSO FS offers a more systematic form of feature selection regardless of prior knowledge of system.

The two black-box feature sets (containing only experimental data) used are: black-box without any feature selection method (BB) and a black-box with LASSO FS (BB+L). The gray-box features sets used are: gray-box with physical insight FS (GB+PHYS), gray-box with LASSO FS (GB+L) and ray-box with physical insight and LASSO FS (GB+PHYS+L). In GB+PHYS, data-driven features are chosen solely based on physical insight into soot oxidation and formation processes. With

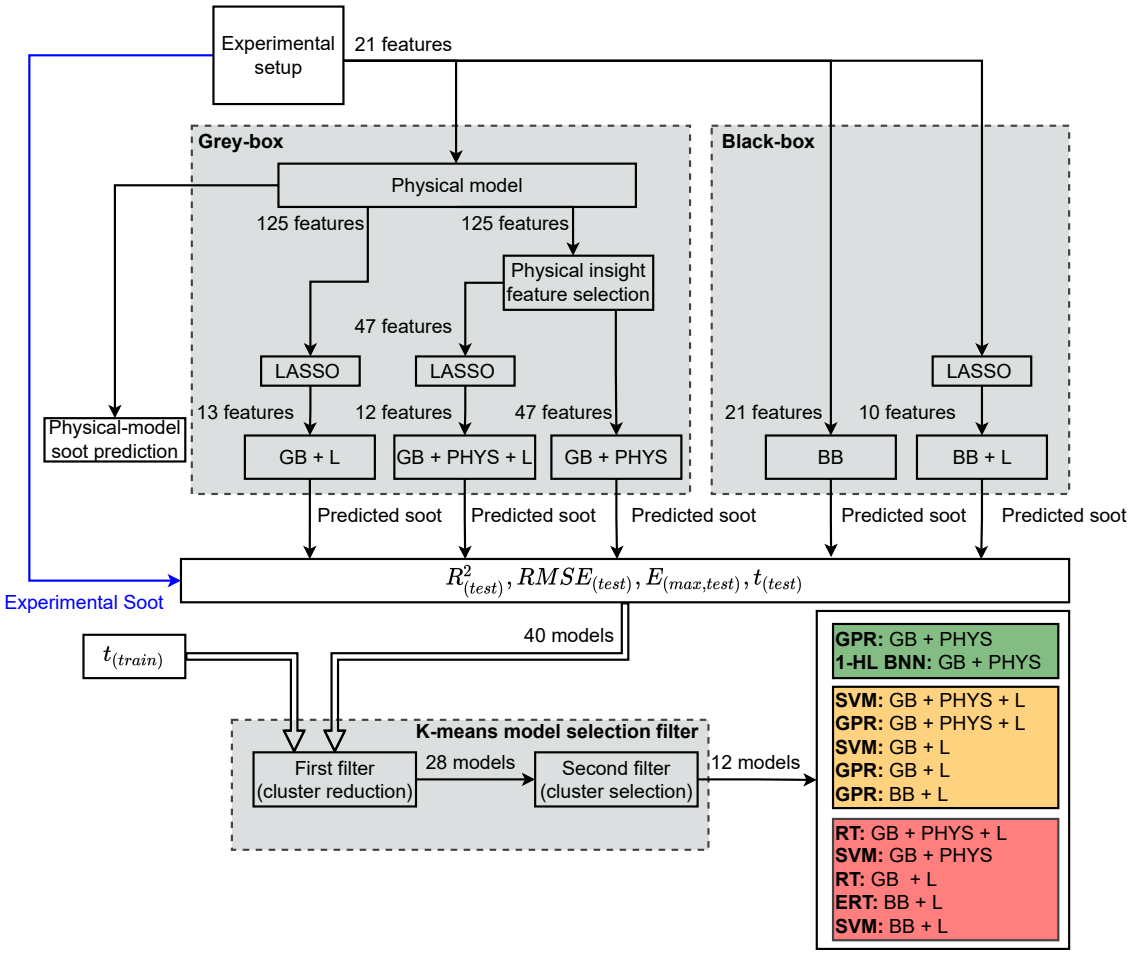


Figure 4.1: Overview of the GB and BB soot emissions model selection process by K-means clustering algorithm

GB+L, the LASSO feature selection method selects the parameters. GB+PHYS+L first uses physical insight to select the most important features, then the LASSO FS method is applied to select the final features. The number of features for the five different methods are summarised in Figure 4.1. As seen, the experimental data is used for the physical-based model. The GB and BB model inputs are similar, including injection properties (total mass of injected fuel, start of injection (SOI), fuel rail pressure), intake manifold pressure, BMEP, and engine speed. The K-means clustering algorithm is used to select the most suitable models and feature sets based on errors and timing (testing and training times). Two K-means clustering algorithms

are applied (the first filter and the second filter). The first filter eliminates feature sets and models with low accuracy and slow training and prediction times, whereas the second filter selects the best ML method along with feature sets in terms of accuracy, training, and prediction cost for different applications. Finally, 12 soot models are chosen in total, which will be explained further in Section 4.3.

4.2 Machine Learning Methods

ML algorithms are used in all three aspects of soot modeling including pre-processing, modeling, and post-processing.

4.2.1 Pre-Processing: Feature Selection

For finding the most effective soot prediction parameters, a LASSO feature selection algorithm is employed for both black-box and gray-box models. LASSO is a regression method that performs feature selection and regularization to improve a model's prediction accuracy. In LASSO regression, the predicted output is $\hat{y}_i = \theta^T x_i$ where θ is the model's coefficient that is calculated by minimizing the following cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{l} |\theta_i| \tag{4.1}$$

where m is the number of training data points and l is number of the parameters of the model, $\sum_{i=1}^{l} |\theta_i|$ is the L_1 regularization and λ is the regularization variable. Adding L_1 regularization leads to driving the weights down to exactly zero (produces sparsity in the solution) and results in performing a systematic feature selection [154]. This sparsity depends on λ, which is calculated in the cross-validation process in the current study.

4.2.2 Regression Models

The five well-known supervised learning regression algorithms are employed: Regression Trees (RT), Ensemble of the Regression Trees (ERT), Gaussian Process Regres-

sion (GPR), Support Vector Machine (SVM), and Neural Network (NN). These are used to train both the black and gray-box soot models.

A data-driven regression model can be generalized to fit a parameterized model, $\hat{y} = h_\theta(x_i)$, for a given training set $\mathcal{D}_{train} = (x_i, y_i)$ such that $\hat{y}$ converges to y_i subject to given constraints. In this problem, x_i is the input feature, y_i is the measured output, and θ is the parameters set. The parameters set can be calculated by solving the following optimization problem

$$\min_{\theta} \quad J(\theta) \tag{4.2}$$

where $J(\Theta)$ is a cost function defined as

$$J(\Theta) = \bar{J}(\Theta) + \lambda L(\Theta) \tag{4.3}$$

where $\bar{J}(\Theta)$ is defined based on an error $e_i(\Theta) = h_\theta(x_i) - y_i$ to minimize the prediction error while regularization term, $L(\Theta)$, is added to regulate parameters, Θ. In general, $L(\Theta)$ is L_1 or L_2 loss function for regularization purposes. For LASSO regression, the L_1 loss function is used while in other regression methods such as Ridge, SVM, and ANN the L_2 loss function is used. The L_2 loss function is defined as

$$L_2(\Theta) = \sum_{i=1}^{l} (\theta)^2 \tag{4.4}$$

The regulatory parameter or penalized variable, λ, produces a trade-off between the smoothness of the model and the training error tolerance minimization [154]. For some algorithms such as SVM as discussed in Chapter 3, Section 3.1, the optimization problem is constrained and the optimization of Eq. 4.2 are solved subject to a constraint function as $\phi(\theta)$.

K-Fold cross Validation

K-fold cross-validation algorithm is used to avoid overfitting of models during training. The K-fold cross-validation first rearranges the dataset randomly and then divides the dataset into k groups. In this study, 5-fold validation is used for all ML methods. In each iteration, the K-fold algorithm chooses one group as a fold, trains a model on the rest of the groups (out of the fold), and assess it on the fold set [189].

Hyperparameters Optimization

Hyperparameters of ML methods such as tolerated error, regularization parameter (λ), and optimization iteration stop criteria in the optimization problem of Equation (4.2) play an important role in decreasing modeling errors and increasing the model's reliability. If an ML algorithm such as A_Λ (where $\mathrm{A}_\Lambda \in \{\mathrm{RT}, \mathrm{ERT}, \mathrm{SVM}\}$ in this study) has N hyperparameters such as $\Lambda = \lambda_1, \lambda_2, ..., \lambda_N$, the optimum hyperparameters can be found by solving the following optimization problem [190]

$$\Lambda^* = \arg\min_{\Lambda} V(h_\theta(x_i), \mathcal{D}_{train}, \mathcal{D}_{valid}) \tag{4.5}$$

where $V(h_\theta(x_i), \mathcal{D}_{train}, \mathcal{D}_{valid})$ measures the performance of a model for a given training and validation set, $\mathcal{D}_{training}$ and $\mathcal{D}_{valid}$, based on algorithm A_Λ.

In this work, Bayesian optimization [191] is used for the RT, SVM, and ERT models' hyperparameters optimization, while a grid search [154], is used for NN-based models such as ANN and BNN.

For the Bayesian optimization to tune the hyperparameters, the evaluation used in Equation (4.5) is

$$V(\lambda) = \frac{1}{n}\sum_{i=1}^{m}(\hat{y_i} - y_i)^2 \tag{4.6}$$

where m is the size of the training set. The model is trained based on training $\mathcal{D}_{train}$ and cross-validated on $\mathcal{D}_{valid}$ in the inner loop of this optimization. Then, $V(\lambda)$ is calculated using both training and cross-validation sets.

To evaluate all possible hyperparameter combinations in NN-based methods, grid search is often used [89]. A search along the space of hyperparameters learning with high probability is tried in Bayesian optimization in a grid search, and all the possible hyperparameters combinations within a given range are tried. In this study, all combinations of layer $L \in \{1, 2\}$ (shallow network) and neurons $s_l \in (1, 40)$ are considered where L and s_l are the number of layers and number of neurons in the l^{th} layer. The layers and neuron's upper limit are set to 2 and 40, respectively, since the limited number of training data means that a deeper network should be avoided.

Regression Tree (RT)

RT is a modeling method with an iterative process of splitting the data into branches where the main algorithm to train RT is Classification and Regression Trees (CART) [192]. In a regression tree, the data are divided into different classes similar to the classification problem with the only difference that each class is assigned to a specific value. RT divides data to k classes based on a threshold (t_k) based on the following cost function

$$J(\theta) = \frac{m_l}{m}\text{MSE}_l + \frac{m_r}{m}\text{MSE}_r \tag{4.7}$$

where subscripts l and r denote left and right and the Mean Squared Error (MSE) is defined as

$$\text{MSE}(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{4.8}$$

where $\hat{y} = \frac{1}{m_{node}}\sum_{i\in\text{node}} y(i)$ and m_l and m_l are left and right branches of the tree. In this method, both k and t_k are considered as model weights and integrated in θ. To avoid overfitting, the minimum number of samples required at a leaf node (Minimum Samples Leaf (MSL)) is added to the CART algorithm as a regularization parameter. The maximum depth of the tree is another regularisation parameter [154].

Ensemble of Regression Trees (ERT)

The ERT is constructed using several decision trees. Three primary hyperparameters to tune the ERT are aggregation methods, number of learners, and MSL. In ensemble learning, bootstrap aggregation (Bagging) and hypothesis boosting (Boosting) are two standard aggregation methods. In bagging, the training algorithm is the same for every predictor, while the training set is a random subset of the training set, i.e., several RTs are trained based on different random subsets of the training set. A well-known example of using the bagging method is the Random Forest. In boosting, a sequential architecture of several weak learners is aggregated, i.e., a series of RTs is trained based on the same training data and layers of RT connected through a series architecture [154]. In this study, Bayesian optimization is used to tune the ERT hyperparameters including a number of learners (number of RT in ERT), MSL, and the aggregating method (boosting/bagging).

Support Vector Machine (SVM)

The SVM is an ML method to find a correlation between input and output by solving a convex quadratic programming problem. More details are provided in Section 3.1 and here SVM with nonlinear kernel is also considered. The cost function of SVM can be defined as

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\theta_i^2 + C\sum_{i=1}^{m}(\zeta_i^+ + \zeta_i^-) \tag{4.9}$$

where ζ_i^- and ζ_i^+, are so-called slack variables and perform as penalty variables to tackle a possible infeasibility of an optimization problem. C includes regulatory parameters (see also Eq. 3.8) . Eq. 4.9 follows the original cost function defined in [176] and equals $1/\lambda$ [154]. Thus, the SVM optimization equation can be rewritten as

$$J(\theta) = \frac{\lambda}{2}\sum_{i=1}^{m}\theta_i^2 + \sum_{i=1}^{m}(\zeta_i^+ + \zeta_i^-) \tag{4.10}$$

The constraint function, $\phi(\theta)$, of SVM in Equation (4.2) is

$$\phi(\theta) = \begin{cases} y_i - h_\theta(x_i) \leq \epsilon + \zeta_i^+ \\ h_\theta(x_i) - y_i \leq \epsilon + \zeta_i^- \\ \zeta_i^-, \zeta_i^+ \geq 0 \end{cases} \tag{4.11}$$

where ϵ is the maximum tolerable deviation for all training data– refer to Section 3.1. In SVM, instead of training data in $\hat{y} = h_\theta(x_i)$, a function of training data, a so-called kernel function, can be replaced by $\hat{y} = h_\theta(\Gamma(x_i))$. This method is called the SVM kernels trick and adding the kernel does not affect the cost function other than using higher dimension feature set instead of x_i in $\hat{y}$. Different kernels such as linear, polynomial, and Gaussian RBF kernels can be considered in optimization. These kernels are defined as

$$K(x_i, x_j) = \begin{cases} {x_i}^T x_j & \text{Linear} \\ ({x_i}^T x_j + c)^n & \text{Polynomial} \\ exp(-\gamma ||x_i - x_j||_2^2) & \text{Gaussian RBF} \end{cases} \tag{4.12}$$

where n and γ are degrees of polynomial and scales of RBF kernels, respectively, [193]. In this study, the optimal kernel type including kernel parameters, i.e., scale and degree of freedom, as well as λ and ϵ are found using Bayesian optimization.

Gaussian Process Regression (GPR)

GPR is a nonparametric and Bayesian-based approach that has superior performance with small data sets and can provide an uncertainty measure on the predictions [130]. The main advantage of GPR is probabilistic prediction. Unlike other supervised ML methods, GPR infers a probability distribution over all possible ML model parameter values. The GPR cost function is defined based on negative log marginal likelihood as

$$J(\theta) = -\log(p(\theta|y, X)) \tag{4.13}$$

where $p(\theta|y, X)$ is posterior distribution (i.e., a likelihood function of θ given X and y) that is defined based on Bayes' Rule as

$$p(\theta|y, X) = \frac{p(y|X, \theta)p(\theta)}{p(y|X)} \tag{4.14}$$

$p(y|X,\theta)$ is a likelihood function of y given X and θ, and $P(y|X)$ is a marginal likelihood function of y given X [130]. Different covariance kernel functions are considered in this study, such as Exponential Kernel, Matern, and Quadratic Kernel with different options. Here, two standard kernels for GPR method including Rational Quadratic kernel function and Matérn kernel function are used. The Rational Quadratic kernel function is

$$K(x_i, x_j|\theta) = \sigma_l^2(1 + \frac{r^2}{2\alpha\sigma_l^2})^{-\alpha} \tag{4.15}$$

and the general Matérn kernel function defines as

$$K_{p+1/2}(x_i, x_j) = \sigma_f^2 \exp(-\frac{\sqrt{2p+1}r}{\sigma_l})\frac{p!}{(2p)!}\sum_{i=1}^{p}\frac{(p+i)!}{i!(p-i)!}(\frac{2\sqrt{2p+1}r}{\sigma_l})^{p-i} \tag{4.16}$$

where r is the Euclidean distance between x_i and x_j ($r = \sqrt{(x_i - x_j)^T(x_i - x_j)}$), σ_l is characteristic length scale, σ_f is signal standard deviation, and α is a positive-valued scale-mixture parameter [130]. In Eq. (4.16), usual value for p is $p = 0$ (Matérn 1/2 $K_{1/2}(x_i, x_j)$), $p = 1$ (Matérn 3/2 $K_{3/2}(x_i, x_j)$), and $p = 2$ (Matérn 5/2 $K_{5/2}(x_i, x_j)$). The Beysian optimization method in this study resulted in using Matérn 5/2 function as the optimum choice for two cases GB + L, GB + PHYS, and GB + PHYS + L which is defined as

$$K_{5/2}(x_i, x_j) = \sigma_f^2(1 + \frac{\sqrt{5}r}{\sigma_l} + \frac{5r^2}{3\sigma_l^2})\exp(-\frac{\sqrt{5}r}{\sigma_l}) \tag{4.17}$$

Neural Network (NN)

In general, an NN is a set of algorithms to model phenomena by mimicking the behavior of the human brain. NN contains three main layers: the input layer, hidden layer (HL), and output layer network [194]. Since only a small amount of data is available, only shallow neural networks with only 1 or 2 hidden layers are considered in this study which are denoted as ANN. Similar to previous ML methods, the cost function of an NN method can be written as

$$J(\theta) = \sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 + \frac{\lambda}{2}\sum_{k=1}^{K-1}\sum_{i=1}^{s_k}\sum_{j=1}^{s_{k+1}}(\theta_{j,i}^{(k)})^2 \tag{4.18}$$

where K is the number of total layers (input + output + hidden layer), s_k is the number of neurons in the k^{th} layer, and m is the size of the training set. The first term in this equation is used to minimize the modeling error while the L_2 loss function is used for regularization. As input neurons and output neurons are set by input and output layers, only the hidden layer number and neuron size are found by using the grid search, i.e., ($L_{HL} = K - 2$) and the number of neurons (s_2 and s_3) in the Hidden Layer (HL).

Bayesian-based NN, denoted as BNN, refers to extending the ANN with Bayesian inference. Unlike the ANN, in which the model's weights are assigned as a single value, in BNN, weights are considered to be a probability distribution. These probability distributions of network weights are used to estimate the uncertainty in weights and predictions [195]. All ANN and BNN configuration combinations are evaluated using this optimization method, and the best model is obtained based on cross-validation data.

A summary of developed models, along with hyperparameter optimization methods and optimized parameters, are listed in Table 4.1.

4.2.3 Post-Processing: Model Selection

The K-means clustering algorithm, an unsupervised ML method, is used to analyze the results and select the best feature sets and methods for different applications. K-means algorithm divides data into n clusters with equal variance. To do this the K-means algorithm tries to divide this data into M disjoint clusters, then minimizes the within-cluster sum-of-squares or inertia, which is the sum of squared Euclidean distance between cluster members and the cluster center

$$J(\theta_1, ..., \theta_M) = \sum_{i=1}^{N}\sum_{k=1}^{M} I(x_i \in C_k)||x_i - \theta_k||^2 \tag{4.19}$$

where $J(\theta)$ is a cost function of the K-means algorithm (also known as inertia) and θ_k is the center of cluster k. If $x_i \in C_k$, $I(x_i \in C_k)=1$; otherwise, $I(x_i \in C_k)=0$. The algorithm starts with random centers and updates the centers in each iteration until the centers remain unchanged, which is a local optimum point. In order to find out the optimum number of clusters for a data set, the elbow method could be used. In this method, inertia is plotted as a function of the number of clusters. The elbow of this curve shows the optimum number of clusters. All these models are evaluated for the test set in Section 4.3, and results will be discussed next.

4.3 Results and Discussion

The engine experimental data points are divided into 80% (175 points) for training $\mathcal{D}_{train}$, and 20% for testing $\mathcal{D}_{test}$ (44 points). Figure 4.2 shows the distribution of the test and training data. The K-fold validation method with five folds ($k = 5$) is also included in training $\mathcal{D}_{valid}$. Testing data $\mathcal{D}_{test}$ is used only for the final evaluation of the model. Figure 4.3 shows the color map of raw soot data with respect to engine speed (x-axis) and load (y-axis), where black dots represent experimental points (see Chapter 2 for more detail about soot data collection). Since this engine is designed for stationary applications, it has limited operating conditions. Therefore, the 219 data points in Figure 4.3 cover most of the possible operating conditions.

Table 4.1 and Table 4.2 show details about the data-driven methods used in this study and their performance for different feature sets. A total of 40 models are defined by five different feature sets and eight ML methods. Model performance is evaluated by considering the following criteria:

1. The coefficient of determination of test data R^2_{test};

2. Root Mean Square of Error of test data $RMSE_{test}$ [mg/m^3];

3. Maximum of absolute prediction error of test data $|E_{test,max}|$ [mg/m^3];

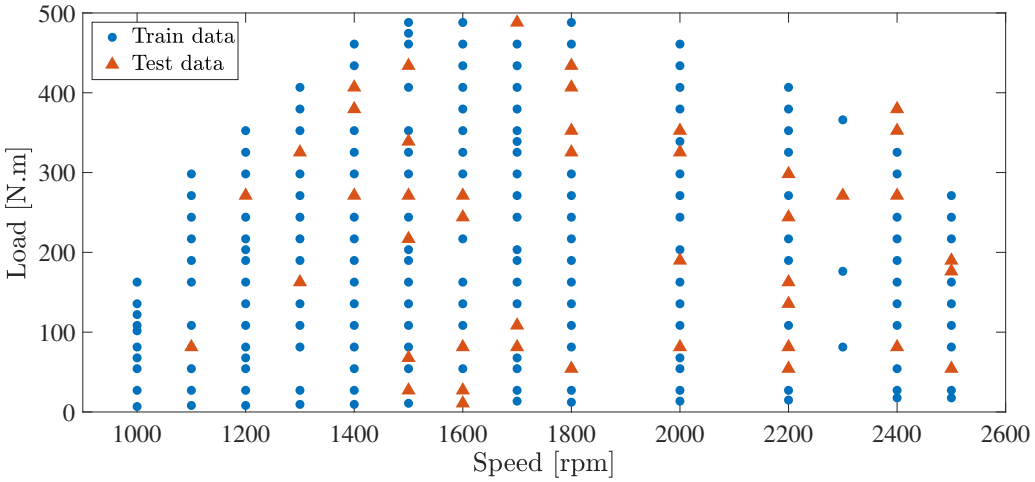


Figure 4.2: Training and test data for ML approaches– 175 data points are used as the training dataset (80%) and 44 data points are used as the testing dataset (20%)

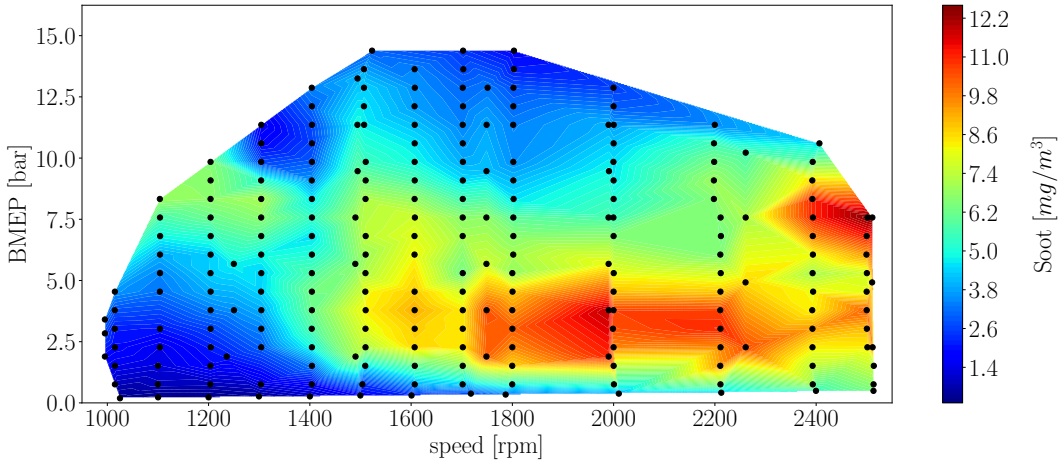


Figure 4.3: Engine-out soot measurements over speed and Break Mean Effective Pressure (BMEP)

4. Training time t_{train} [s];

5. Prediction time t_{test} [ms].

The accuracy of the model is based on the first three criteria. The third criterion is useful to assess the model reliability since outliers cause high maximum errors. High maximum error means that model will be inaccurate in some instances. A low maximum error is associated with less severe outliers and a more robust model. There

Table 4.1: Training and optimization of ML-based model hyperparameters.

Method	**Opt. Method**	**Opt. Hyper-parameters**	**Model Type**	**Opt. Model Configuration**
RT	Bayesian	Min samples leaf (MSL)	BB	MSL = 13
			BB + L	MSL = 1
			GB + L	MSL = 5
			GB + PHYS	MSL = 5
			GB + PHYS + L	MSL = 5
ERT	Bayesian	Ensemble method, min samples leaf, and number of learners	BB	Boosting, 75 Learners, and MSL = 2
			BB + L	Boosting, 28 Learners, and MSL = 4
			GB + L	Boosting, 35 Learners, and MSL = 5
			GB + PHYS	Boosting, 488 Learners, and MSL = 47
			GB + PHYS + L	Boosting, 487 Learners, and MSL = 2
SVM	Bayesian	Kernel function λ and ϵ	BB	Cubic, $\lambda = 0.96$, $\epsilon = 0.010$
			BB + L	Quadratic, $\lambda = 0.77$, $\epsilon = 0.330$
			GB + L	Gaussian, $\lambda = 9.59$, $\epsilon = 0.004$
			GB + PHYS	Quadratic, $\lambda = 3.49$, $\epsilon = 0.003$
			GB + PHYS + L	Cube, $\lambda = 5.79$, $\epsilon = 0.009$
GPR	Bayesian	Kernel function, initial value for the noise standard deviation (σ)	BB	Rational quadratic, $\sigma = 12.68$
			BB + L	Rational quadratic, $\sigma = 0.0005$
			GB + L	Matérn 5/2, $\sigma = 0.0001$
			GB + PHYS	Matérn 5/2, $\sigma = 0.0001$
			GB + PHYS + L	Matérn 5/2, $\sigma = 2.996$
1-HL ANN	Grid search	Number of neurons in each layer	BB	Network conf.: [25]
			BB + L	Network conf.: [19]
			GB + L	Network conf.: [4]
			GB + PHYS	Network conf.: [4]
			GB + PHYS + L	Network conf.: [19]
2-HL ANN	Grid search	Number of neurons in each layer	BB	Network conf.: [7,25]
			BB + L	Network conf.: [25, 31]
			GB + L	Network conf.: [4, 13]
			GB + PHYS	Network conf.: [7,13]
			GB + PHYS + L	Network conf.: [16, 19]
1-HL BNN	Grid search	Number of neurons in each layer	BB	Network conf.: [7]
			BB + L	Network conf.: [31]
			GB + L	Network conf.: [31]
			GB + PHYS	Network conf.: [13]
			GB + PHYS + L	Network conf.: [25]
2-HL BNN	Grid search	Number of neurons in each layer	BB	Network conf.: [7,28]
			BB + L	Network conf.: [16, 13]
			GB + L	Network conf.: [10, 22]
			GB + PHYS	Network conf.: [22, 22]
			GB + PHYS + L	Network conf.: [10, 19]

Table 4.2: ML-based data-driven soot models comparison– BB, L, GB, and PHYS stand for black-box, LASSO, gray-box, and physical insight, respectively

Model	**Criteria**	**RT**	**ERT**	**SVM**	**GPR**	**1-HL NN**	**2-HL NN**	**1-HL BNN**	**2-HL BNN**
BB	R^2_{train}	0.85	0.95	0.86	0.87	0.86	0.86	0.88	0.90
	R^2_{test}	0.41	0.51	0.50	0.27	0.52	0.54	0.51	0.52
	$RMSE_{train}[mg/m^3]$	1.41	0.90	1.39	1.35	1.44	1.38	1.27	1.21
	$RMSE_{test}[mg/m^3]$	2.52	2.38	2.53	2.35	2.41	2.32	2.39	2.43
	$\lvert E_{test,max}\rvert[mg/m^3]$	8.7	8.5	8.2	7.7	6.6	7.9	7.7	7.5
	$t_{test}[ms]$	2.23	16.73	2.08	3.11	8.66	9.53	6.47	6.93
	$t_{train}[s]$	0.74	3.50	0.40	1.56	3.77	1.11	2.07	14.31
BB + L	R^2_{train}	0.98	0.99	0.97	1	0.97	0.98	0.99	0.99
	R^2_{test}	0.87	0.91	0.93	0.96	0.90	0.92	0.95	0.94
	$RMSE_{train}[mg/m^3]$	0.48	0.52	0.66	0.28	0.66	0.63	0.22	0.20
	$RMSE_{test}[mg/m^3]$	1.33	1.07	0.98	0.51	1.19	1.10	0.83	0.93
	$\lvert E_{test,max}\rvert[mg/m^3]$	5.02	3.14	4.37	1.87	4.35	4.53	2.85	4.3
	$t_{test}[ms]$	1.94	5.26	2.27	2.73	7.49	8	14.7	10.4
	$t_{train}[s]$	0.75	1.57	0.44	1.32	2.80	2.33	4.57	15.13
GB + L	R^2_{train}	0.97	0.99	0.98	0.99	0.96	0.96	0.99	0.99
	R^2_{test}	0.92	0.93	0.95	0.94	0.90	0.92	0.95	0.95
	$RMSE_{train}[mg/m^3]$	0.62	0.06	0.48	0.38	0.73	0.72	0.34	0.09
	$RMSE_{test}[mg/m^3]$	1.09	1.00	0.81	0.67	1.2	0.88	0.88	0.97
	$\lvert E_{test,max}\rvert[mg/m^3]$	2.9	3.7	1.9	1.9	3.6	2.3	2.3	2.6
	$t_{test}[ms]$	2.21	47.16	2.05	3.59	7.24	12.42	7.39	6.86
	$t_{train}[s]$	0.79	8.57	0.37	6.1	2.97	1.04	12.10	14.66
GB + PHYS	R^2_{train}	0.98	0.99	0.98	0.99	0.97	0.98	0.99	0.99
	R^2_{test}	0.87	0.96	0.94	0.97	0.90	0.89	0.93	0.83
	$RMSE_{train}[mg/m^3]$	0.54	0.01	0.57	0.13	0.70	0.6	0.07	0.01
	$RMSE_{test}[mg/m^3]$	1.3	0.74	0.91	0.5	1.2	0.94	1.2	1.06
	$\lvert E_{test,max}\rvert[mg/m^3]$	5.88	1.8	3.3	1.58	4.35	4.76	2.67	5.52
	$t_{test}[ms]$	2.74	58.19	3.1	5.87	7.3	14.22	6.69	10.63
	$t_{train}[s]$	0.75	13.90	0.46	43.24	3.09	1.11	35.87	103.90
GB + PHYS + L	R^2_{train}	0.98	0.99	0.98	0.99	0.95	0.98	0.99	0.99
	R^2_{test}	0.89	0.95	0.97	0.96	0.91	0.94	0.90	0.93
	$RMSE_{train}[mg/m^3]$	0.60	0.01	0.57	0.31	0.87	0.49	0.13	0.08
	$RMSE_{test}[mg/m^3]$	1.24	0.83	0.71	0.52	1.2	0.94	1.19	1.06
	$\lvert E_{test,max}\rvert[mg/m^3]$	2.94	2.65	1.64	1.41	3.42	2.97	4.73	3.4
	$t_{test}[ms]$	2.06	56.31	2.28	3.08	9.13	10.4	6.32	7.06
	$t_{train}[s]$	0.79	10.65	0.52	3.77	2.70	1.22	8.59	8.22

is a direct relationship between the complexity of the model and the training time. Overfitting is more likely to occur in complex models, so typically less complex models are more likely to show the same performance for different applications [196]. The K-means clustering algorithm is employed to choose the most appropriate models and feature sets for a variety of applications including calibration, real-time control, and to study the effect of changes in different engine components. The above five separate parameters are used as the input feature set for the K-means algorithm. The appropriate number of clusters must be determined before using the K-means algorithm. This is accomplished with the elbow method, as previously mentioned. Based on the elbow method, the optimum number of clusters is 6.

Figure 4.4 shows the result of clustering of the models. The same color is assigned to models that are part of the same cluster. The first filter (the first K-means algorithm) aims to exclude data sets and methods with low accuracy and high training and testing times. The red and black clusters (the clusters where the members are shown in red and black in Figure 4.4) have a very low accuracy compare to other cluster members (low R^2, high RMSE and high $|E_{max}|$ in Figure 4.4(a), (b), and (c)). A higher t_{test} is the main characteristic of the green cluster members compared to other clusters based on Figure 4.4(d). Additionally, the pink clusters have a considerably larger $t_{training}$ than the others based on Figure 4.4(e). This analysis leads to the removal of the red, black, green, and pink clusters due to their low accuracy and long training and prediction (testing) times. As a result, 12 of the 40 models are removed by the first filter, leaving 28 models for the second K-means based filter.

A second K-means filter is applied to choose the best models out of the remaining models for the varied applications including real-time control and calibration. Figure 4.5 shows the result of the clustering by means of the second filter. Each cluster is assigned a number to simplify the subsequent discussion. The error values, training time, and test time for members of different clusters are shown in Figure 4.6. Members of clusters 1, 4 and 2 have higher accuracy than the other clusters. Members of

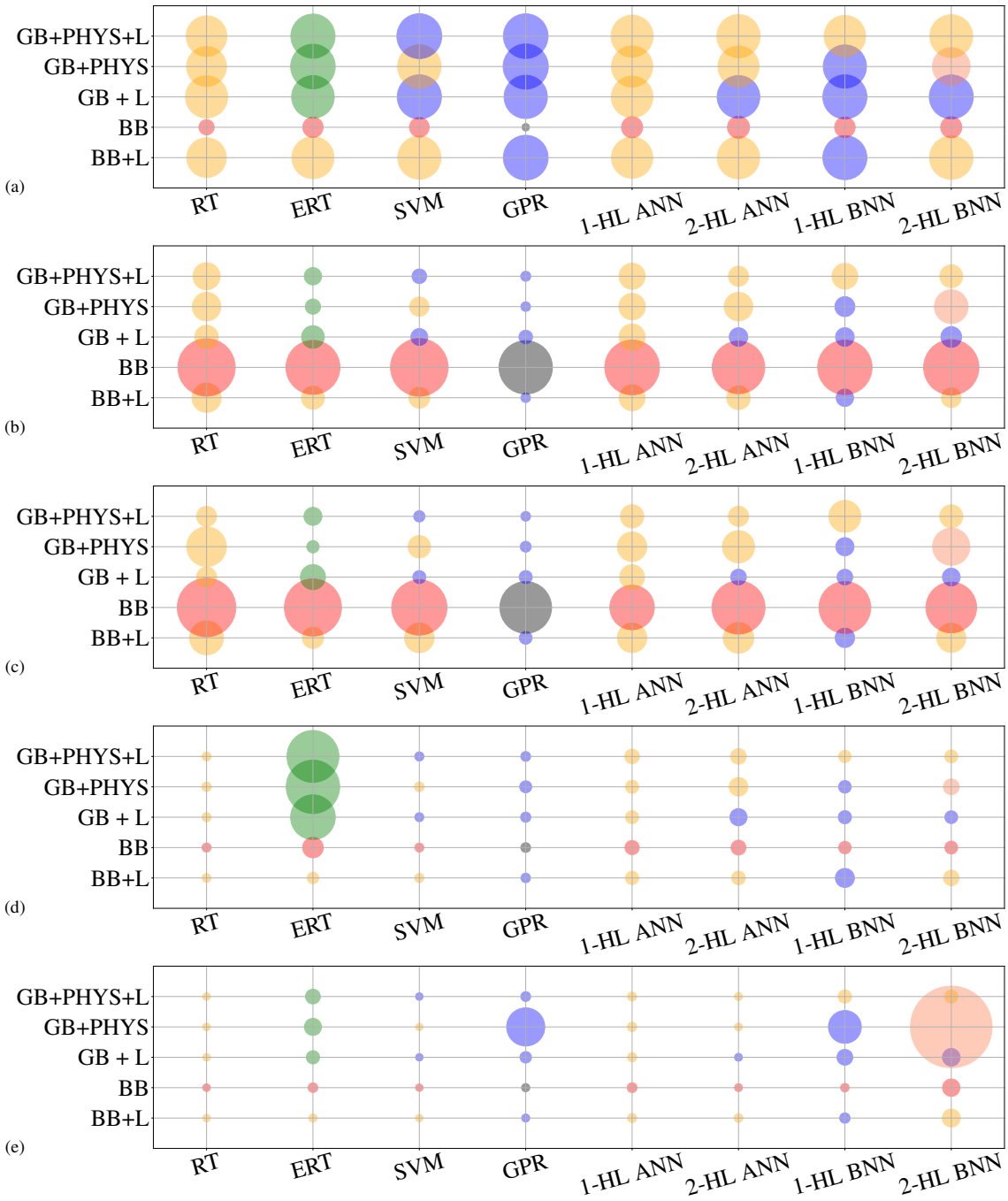


Figure 4.4: First filter clustering of models using K-means algorithm: 40 models divided into 6 clusters and sorted based on (a) R^2_{test}, (b) $RMSE_{test}$ [mg/m^3], (c) $|E_{test,max}|$ [mg/m^3], (d) t_{test} [ms] (test time), and (e) t_{train} [ms] (training time)

cluster 0 and 3 have the largest maximum error, lowest R^2 and highest RMSE with high testing time based on Figure 4.6(a), (b), and (c). As a result, these clusters can be removed as it is low in accuracy and high in deployment (test) time. Using

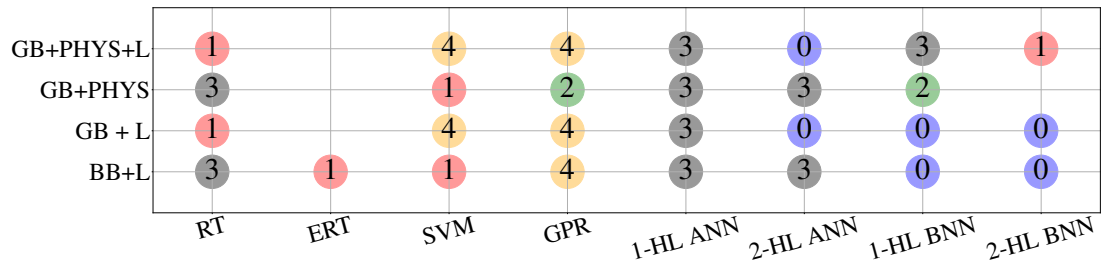


Figure 4.5: Second filter clustering of models using K-means algorithm. The assigned number for each color is shown.

Table 4.3: Selected models based on K-means filters

Cluster	Model	Accuracy	Reliability	Less complexity	Real-time control	Virtual test
2	**GPR:** GB + PHYS	×	×			×
2	**1-HL BNN:** GB + PHYS		×			
4	**SVM:** GB + PHYS + L	×	×	×		×
4	**GPR:** GB + PHYS + L	×	×	×		×
4	**SVM:** GB + L	×	×	×		×
4	**GPR:** GB + L		×	×		×
4	**GPR:** BB + L	×	×	×	×	
1	**RT:** GB + PHYS + L			×		
1	**SVM:** GB + PHYS			×		
1	**RT:** GB + L			×		
1	**ERT:** BB + L					
1	**SVM:** BB + L			×	×	

the remaining models, we could determine which feature sets and methods were best suited to the different applications. Table 4.3 shows the selected ML methods and feature sets for different applications.

For accuracy, R^2, RMSE and $|E_{max}|$ are important parameters. The reliability of a model depends heavily on its $|E_{max}|$. A high value of $|E_{max}|$ indicates severe outliers. As a result, there is a possibility of high error rates for some predictions in the model, making it unreliable. Training time is a deciding factor in choosing a model with a low degree of complexity. The selection of models is limited to experimental feature sets for real-time control and adaptive learning because only measurable features could be used as input in real-time control . So, the experimental feature sets (BB and BB+L) are acceptable. Unlike real-time control, virtual tests are based on feature

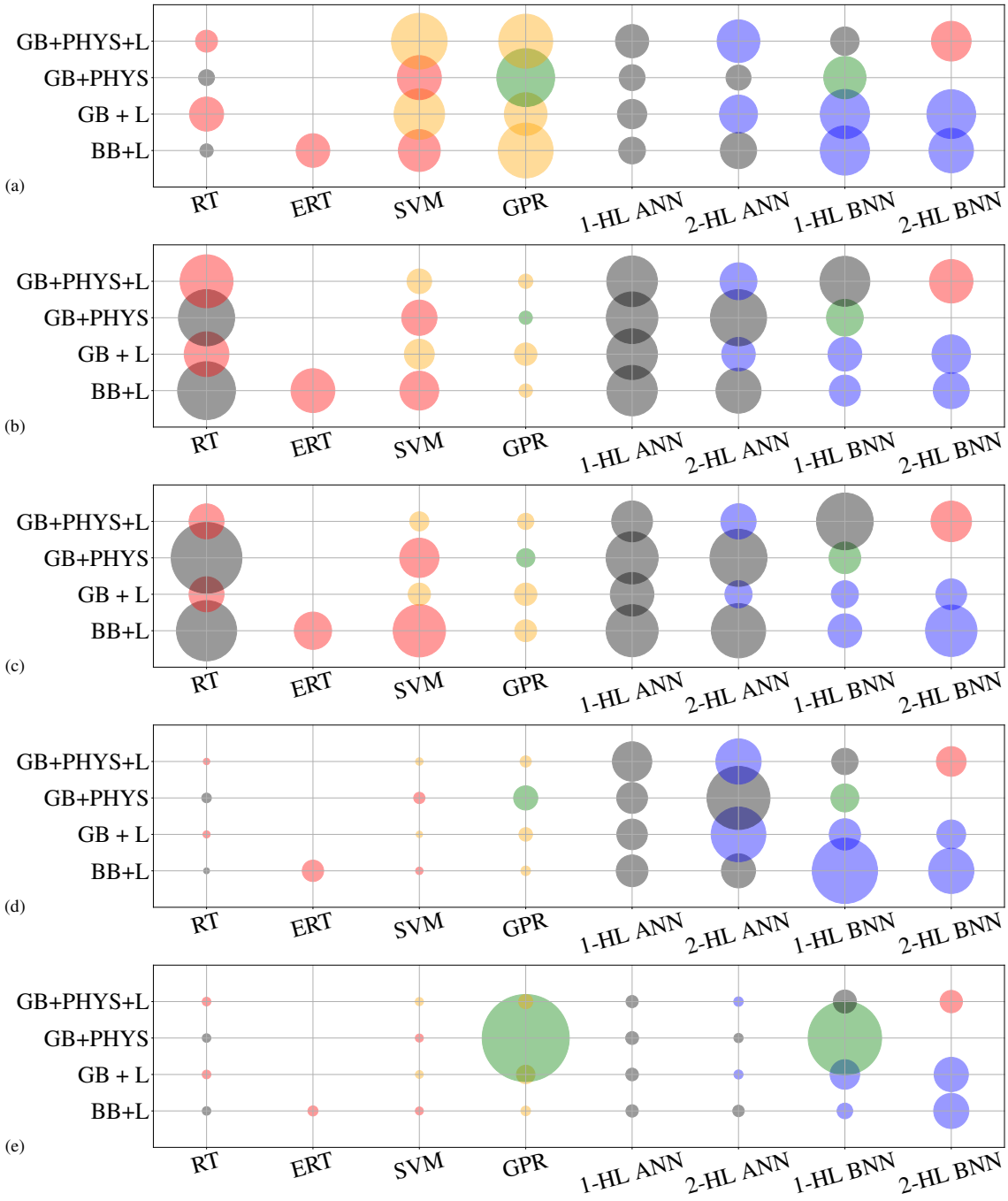


Figure 4.6: Second filter clustering of Models using K-means algorithm: 28 models divided into 5 clusters where three clusters, including 12 models, have been chosen as the final selection. (a) R^2_{test}, (b) $RMSE_{test}$ [mg/m^3], (c) $|E_{test,max}|$ [mg/m^3], (d) t_{test} [ms] (test time), and (e) t_{train} [ms] (training time)

sets generated by the engine model (GB, GB+L, and GB+PHYS+L). Clustering is used to choose the models with the highest possible accuracy for different applications. Based on Figure 4.6 (a), (b), and (c) clusters 2 and 4 have the highest accuracy and reliability, so the majority of their members were selected for these factors. Based on Figure 4.6 (e), cluster 2 is characterized by the high training time. So, its members are not selected based on the lower complexity criterion. Cluster 1 has acceptable accuracy for most of its cases, despite being not as accurate as cluster 4 and has a low training time. As a result, some of the members of cluster 1 are rated as less complex.

Table 4.3 shows the 12 selected models for different applications. Figure 4.7 shows the prediction vs experiment diagrams for the physical soot model. Figures 4.8 and 4.9 shows the prediction vs experiment diagrams for the test data for 12 selected models. By comparing the results in Figure 4.7 and Figures 4.8- 4.9, all the 12 models are much more accurate than the physical soot model. The complexity of soot formation and oxidation processes [197] makes it difficult for soot emissions formation and oxidation processes to be adequately represented by 1D physical soot models [197] which is reflected in Figure 4.7. Model-based studies for soot emissions prediction show the same trend [198], and have motivated the data-driven methods of soot emissions prediction.

According to Table 4.5, GPR and SVM are the most accurate methods for this data set. Further, the virtual engine model enhances the model's accuracy and 4 out of 5 models that are selected for high accuracy have used some forms of the GB feature set. In general, GPR: GB + PHYS, SVM: GB + PHYS + L (both has the same R^2_{test}), and GPR: BB+L are found to be the best models among the GB and BB models, respectively. Comparison between GPR: GB + PHYS and SVM: GB + PHYS + L results in almost the same accuracy with different criteria including R^2_{test}, $\text{RMSE}_{\text{test}}$ [mg/m^3], and $|\text{E}_{\text{test,max}}|$ [mg/m^3]; however, as SVM: GB + PHYS+ L takes less training and testing time, it has been chosen as the best GB model. Figure 4.10

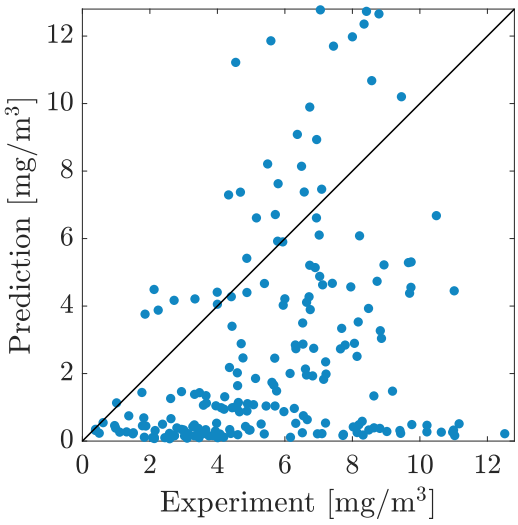


Figure 4.7: Comparison of the Physics-based GT-power soot model prediction against experimental data (when the data follows the diagonal line, the accuracy is acceptable)

shows the accuracy of soot prediction for SVM: GB + PHYS+ L and GPR: BB+L models for the training and the test data over engine speed-load diagram. For most of the engine's load and speed ranges, both models are quite accurate in soot prediction. In comparison to GPR: BB+L model (BB), the SVM: GB+PHYS+L model (GB) have fewer outliers. This is attributed to the use the combustion physical model in the gray-box model, which assists in reducing outliers in soot emissions prediction. Table 4.4 shows a comparison between state-of-the-art studies about soot emissions modeling using GB models.

Table 4.4: Comparison between studies about soot emissions modeling using GB models

Study	**Machine learning method**	**Soot modeling $\mathbf{R^2_{test}}$**
Lang et al. [101]	GPR	0.83
Mohammad et al. [89]	ANN	0.95
Shahpouri et al. [199]	SVM	0.95
Current study	SVM/GPR	0.97

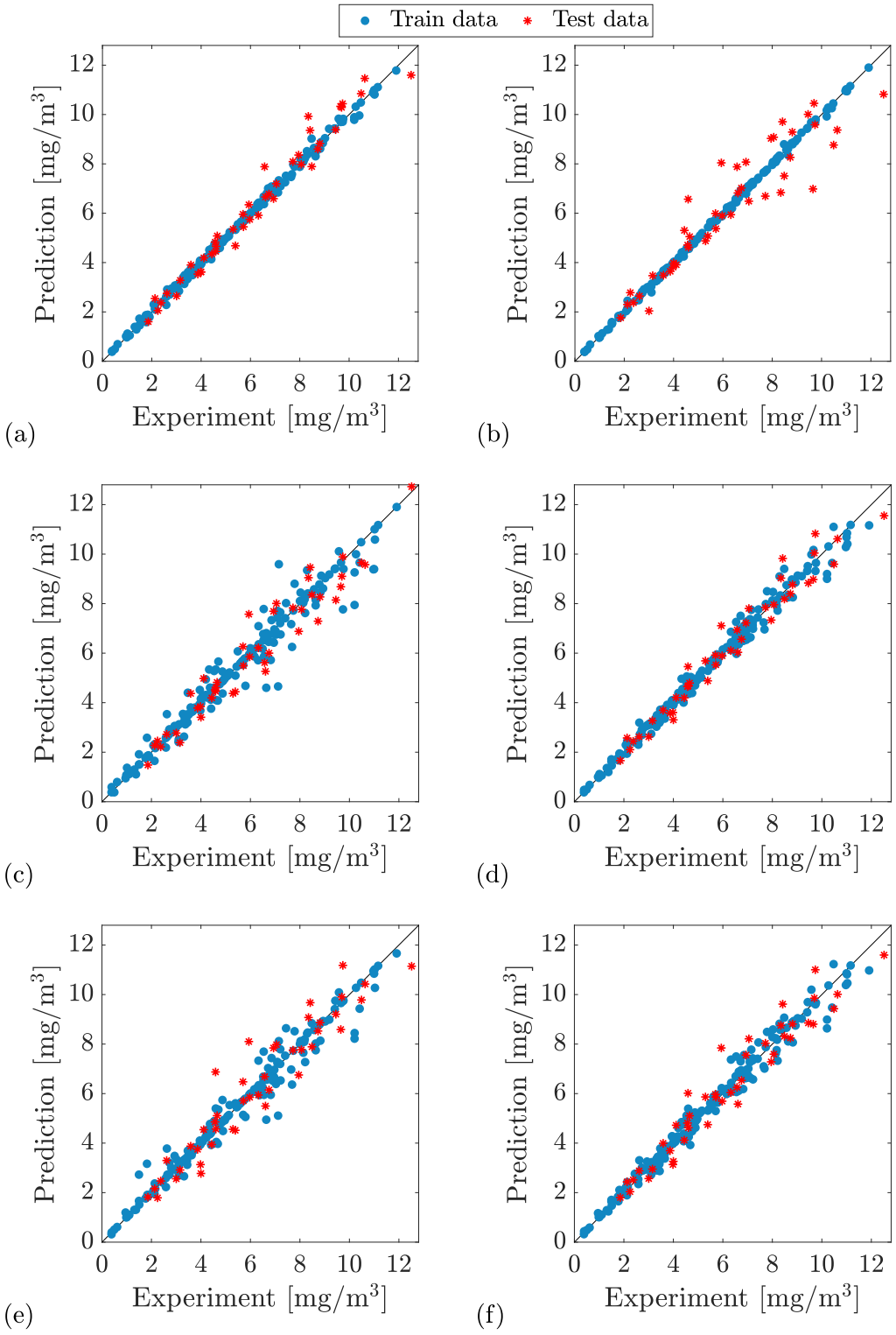


Figure 4.8: Comparison of model prediction versus experimental data for different models: (a) GPR: GB + PHYS, (b) 1-HL BNN: GB + PHYS, (c) SVM: GB + PHYS + L, (d) GPR: GB + PHYS + L, (e) SVM: GB, (f) GPR: GB (when the data follows the diagonal line, the accuracy is acceptable)

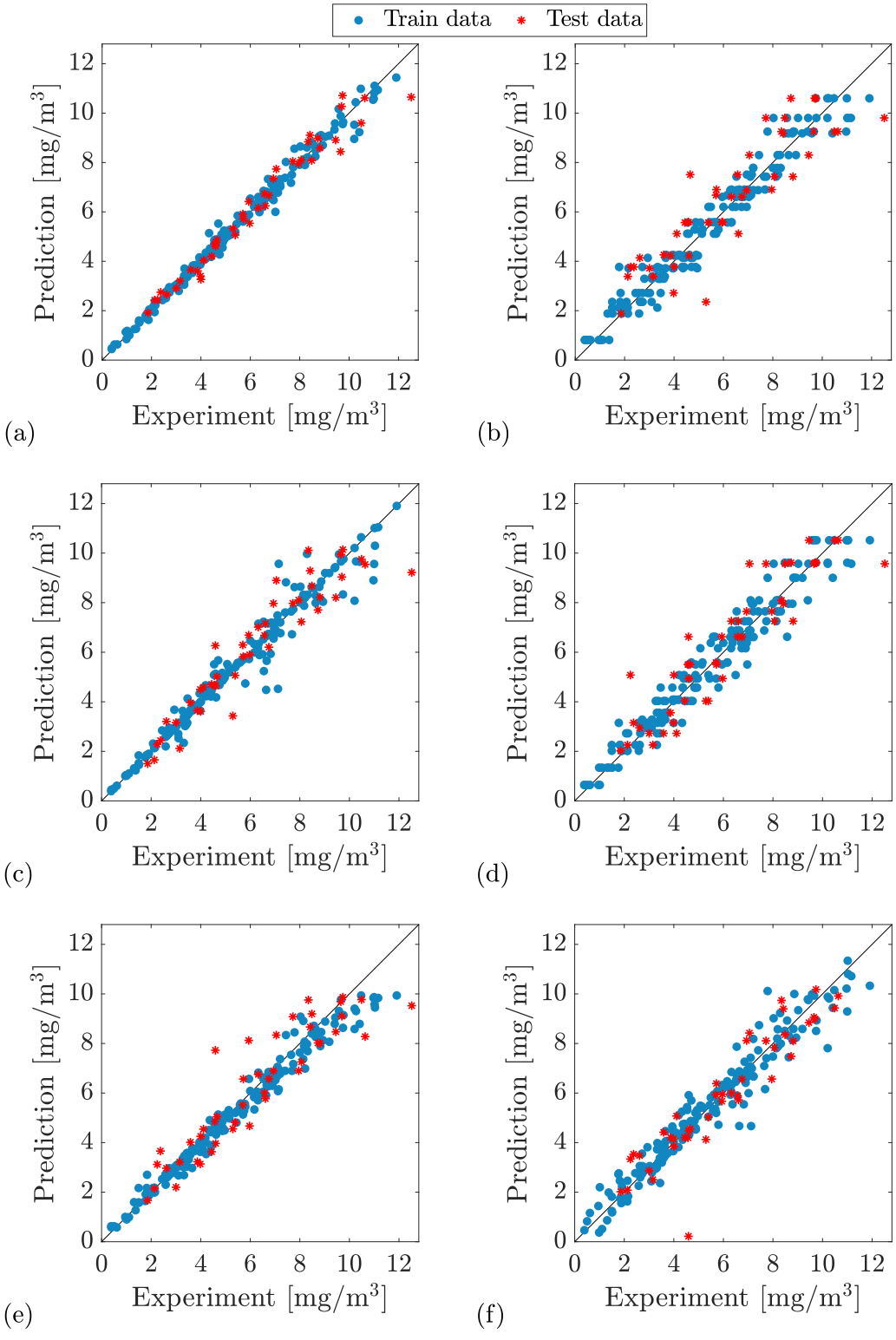


Figure 4.9: Comparison of model prediction versus experimental data for different models: (a) GPR: BB + L, (b) RT: GB + PHYS + L, (c) SVM: GB + PHYS, (d) RT: GB, (e) ERT: BB + L, (f) SVM: BB + L (good accuracy is when the data follows the diagonal line)

As seen, the best GB model developed in this study (SVM: GB+PHYS+L) with respect to test R^2_{test} outperforms the best models presented in previous studies.

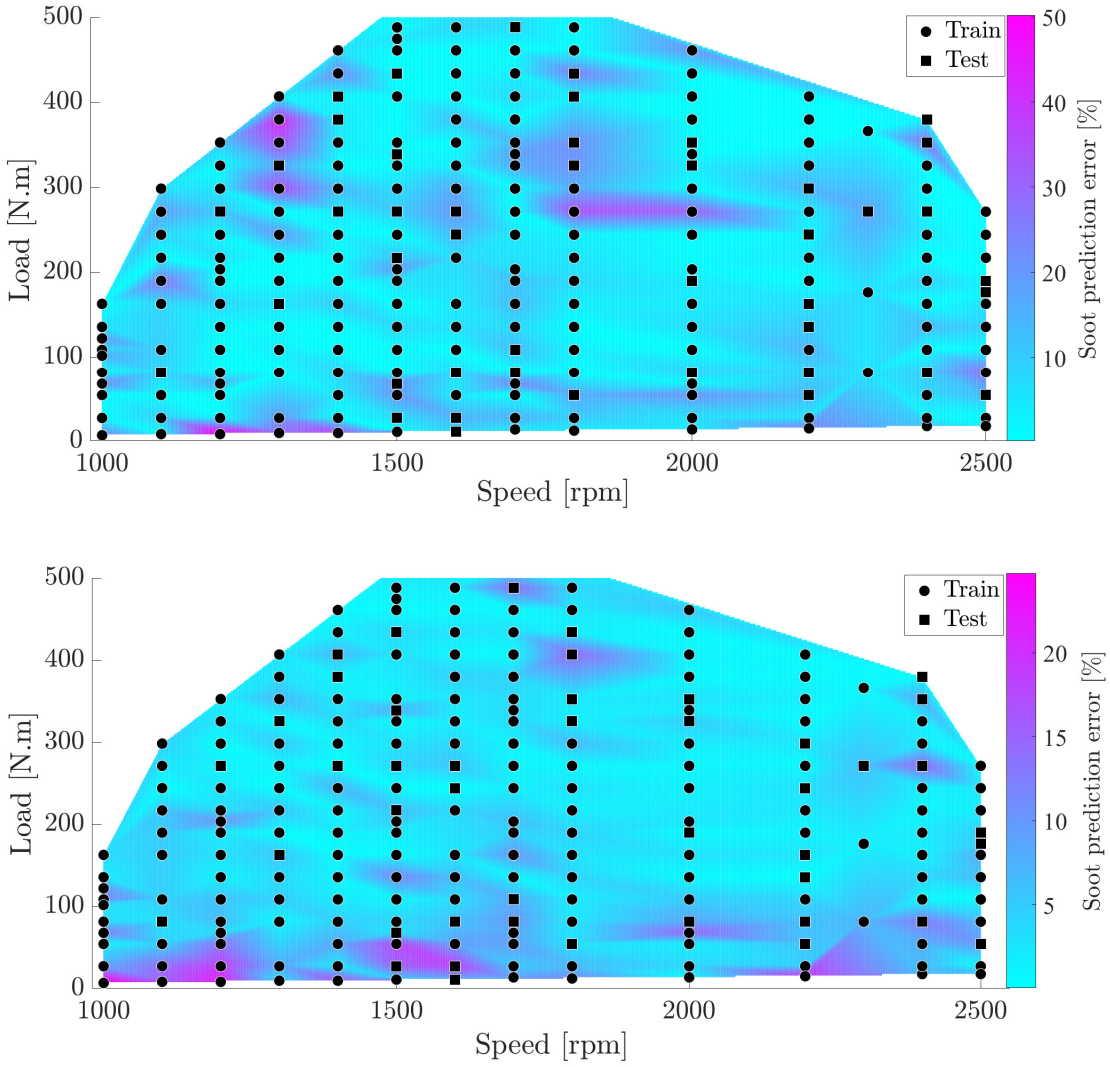


Figure 4.10: Prediction error [%] over engine speed and load for two models: (a) GPR: BB + L, (b) SVM: GB + PHYS + L

4.4 Summary of chapter

To predict soot emissions for a compression ignition engine, physical, BB, and GB modeling were used in this chapter. GB and BB soot emissions models were developed using eight different machine learning methods. Based on the LASSO feature selection

method and physical insight, five feature sets were tested for BB and GB models. To analyze the results, the K-means clustering algorithm was applied in two steps to categorize the models according to their performance. Different methods and feature sets were chosen for various applications. Real-time control is only feasible with BB methods since the physics-based model is too computationally expensive for a real-time Engine Control Unit (ECU). Based on the results, the GPR method with LASSO as the feature selection method is the most reliable ML method/feature set with R^2_{test} = 0.96, $RMSE_{test}$ [mg/m^3] = 0.51, $|E_{test,max}|$[mg/m^3] = 1.87 and t_{test} [ms] = 2.73. GB models can be used as a virtual engine to conduct simulation tests for development and calibration purposes, reducing the need for costly experiments. Among the GB models, an SVM-based ML method along with LASSO and physical insight for feature selection provides the best performance with R^2_{test} = 0.97, $RMSE_{test}$ [mg/m^3] = 0.71, $|E_{test,max}|$[mg/m^3] = 1.64 and t_{test} [ms] = 2.28. In most cases, GB models outperform their BB counterparts in terms of accuracy.

PART III: Integration of Machine Learning and Model Predictive Control

Chapter 5

Machine Learning Integrated with Linear Parameter Varying Model Predictive Control: Simulation Results [1]

In this chapter, two methods of combining ML and Model Predictive Control (MPC) are presented. The two methods presented are ML-based modeling and ML imitation of an online optimization of MPC controller. A model of the engine performance and emissions is developed using ML and is then used as the model implemented on a diesel engine for MPC. This online optimized MPC solution offers advantages in minimizing the NO_x emissions and fuel consumption compared to the baseline feedforward production controller. To reduce the computational cost of this MPC, a deep learning scheme is designed to mimic the behavior of the developed controller. An overview of the methodology is depicted in Figure 5.1. First, randomly generated inputs are fed into the ESM and the output engine performance is recorded for control modeling. A Least-square Support Vector Machine based Linear Parameter-Varying (SVM-LPV) model is developed using the input-output data. This model is used to design the LPV-MPC controller. Finally, this MPC controller is used to train the ML based imitation controllers. To assess, the LPV-MPC controller performance it is compared to a Linear Autoregressive with Extra Input (ARX) based linear MPC

[1] This chapter is based on [6]

using a GT-power©/Matlab/Ssmulink© co-simulation platform.

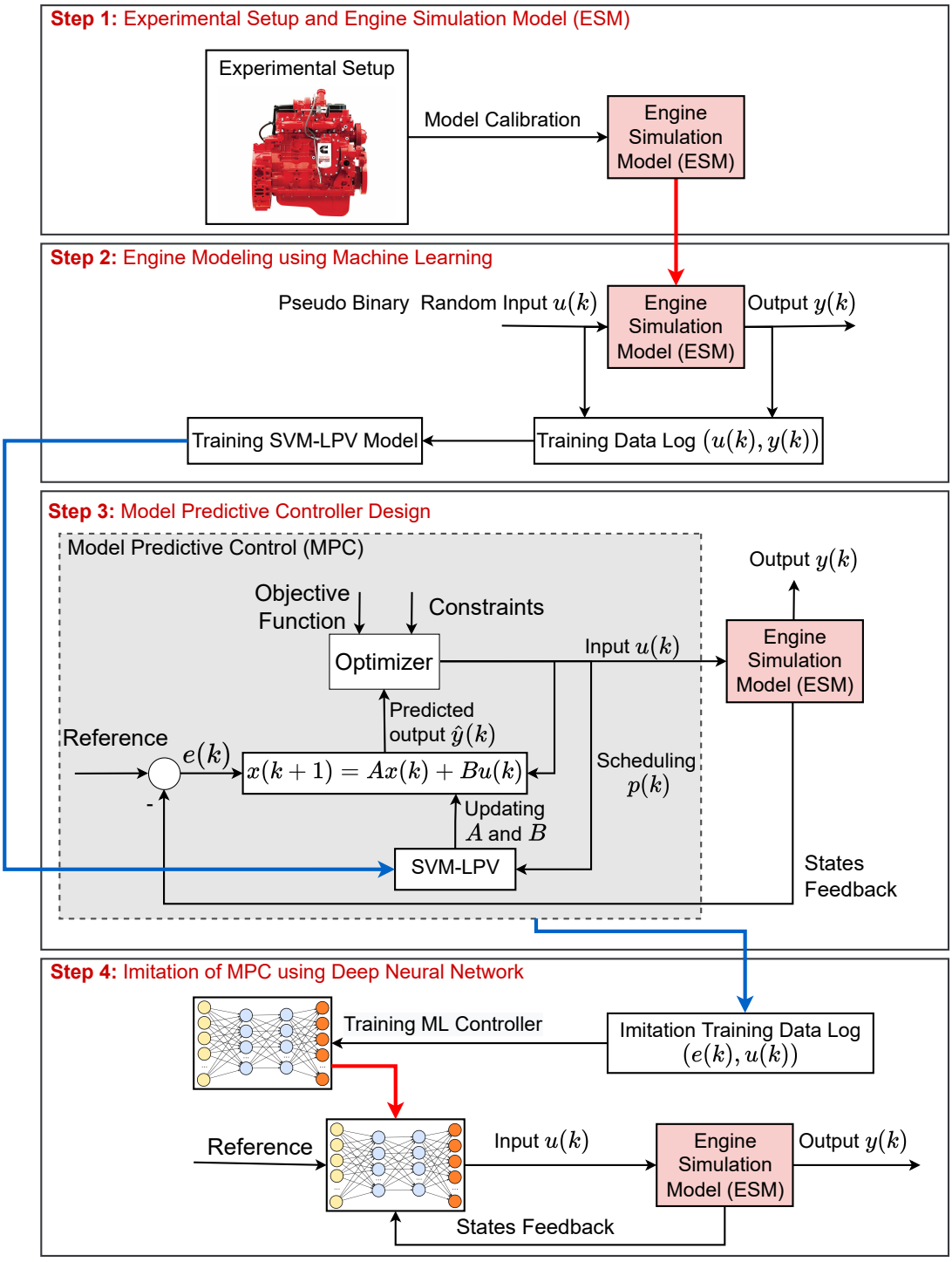


Figure 5.1: Modeling and controller design procedure based on ESM for SVM-LPV and corresponding imitation controller

5.1 Linear Parameter Varying Modeling

5.1.1 Support Vector Machine based Linear Parameter Varying (LPV) Model

First an LPV model is developed using the SVM framework. The LPV model is defined as:

$$\begin{aligned} \mathbf{x}(k+1) &= A\left(p(k)\right)\mathbf{x}(k) + B\left(p(k)\right)\mathbf{u}(k) \\ \mathbf{y}(k) &= C\left(p(k)\right)\mathbf{x}(p(k)) + D\left(p(k)\right)\mathbf{u}(k) \end{aligned} \tag{5.1}$$

Where the state matrices (A,B,C and D) are a function of scheduling parameters, $p(k)$. An SVM-based algorithm is used to update the state matrices. In this study $u(k)$. $x(k)$, and $y(k)$ are defined as

$$\begin{aligned} u(k) &= \begin{bmatrix} \mathrm{FQ}(k) & \mathrm{SOI}(k) & \mathrm{VGT}(k) \end{bmatrix}^T, \\ x(k) &= \begin{bmatrix} T_{\mathrm{out}}(k) & P_{\mathrm{man}}(k) & \mathrm{NO}_x(k) \end{bmatrix}^T, \\ y(k) &= \begin{bmatrix} T_{\mathrm{out}}(k) & \mathrm{NO}_x(k) \end{bmatrix}^T, \end{aligned} \tag{5.2}$$

where $\mathrm{FQ}(k)$ is Fuel Quantity (FQ), $\mathrm{SOI}(k)$ is the Start Of Injection (SOI) for the main injection, $\mathrm{VGT}(k)$ is the Variable Geometric Turbine (VGT) rate, (T_{out}) is the engine output torque, P_{man} is the intake manifold manifold pressure, and $\mathrm{NO}_x(k)$ is the engine-out Nitrogen Oxides ($\mathrm{NO_x}$) emissions. The SVM-LPV algorithm developed in [153] is then adapted for this specific problem. Additionally, to tune the hyperparameters of the SVM-LPV, a Bayesian optimization is implemented. It is assumed that the output of the model is equal or partially equal to states of the system and that the system states are measurable. Thus, the matrix C is not scheduled, and matrix D is identically zero. Then, the model can be simplified as

$$\mathbf{x}(k+1) = A\left(p(k)\right)\mathbf{x}(k) + B\left(p(k)\right)\mathbf{u}(k) \tag{5.3}$$

where $u(k) \in \mathbb{R}^{n_u}$, $x(k) \in \mathbb{R}^{n_x}$, and $p(k) \in \mathbb{R}^{n_{sp}}$ are the inputs, states, and scheduling parameters at k, and $A \in \mathbb{R}^{(n_x \times n_x)}$ and $B \in \mathbb{R}^{(n_x \times n_u)}$ are the state-space model matrices (n_x and n_u are the number of states and manipulated variable while n_{sp}

is the number of scheduling parameters). To formulate our problem in an SVM framework, $A(p(k))$ and $B(p(k))$ are written as

$$A\left(p(k)\right) = W_1\phi_1\left(p(k)\right), \qquad B\left(p(k)\right) = W_2\phi_2\left(p(k)\right) \tag{5.4}$$

where $\phi_1 \in \mathbb{R}^{(n_x \times n_h)}$ is the high dimension feature space where n_h is the dimension of a high dimensional feature space. Substituting Eq. 5.4 into Eq. 5.3 results in

$$\mathbf{x}(k+1) = \underbrace{[W_1 \quad W_2]}_{\mathbf{W}} \underbrace{\begin{bmatrix} \left(\phi_1\left(p(k)\right)\mathbf{x}(k)\right)^T \\ \left(\phi_2\left(p(k)\right)\mathbf{u}(k)\right)^T \end{bmatrix}}_{\Phi(k)^T} \tag{5.5}$$

The residual error of modeling, $e(k)$, is added to Eq. 5.5 as

$$\mathbf{x}(k+1) = W\Phi(k)^T + e(k) \tag{5.6}$$

The LS-SVM cost function is then defined as

$$\begin{aligned} \text{Minimize:} \quad & \frac{1}{2}||W||_2^2 + \frac{1}{2}\sum_{j=1}^{N} e(j)^T \gamma e(j) \\ \text{Subject to:} \quad & \mathbf{x}(j+1) = W\Phi(j)^T + e(j) \end{aligned} \tag{5.7}$$

Where N is the number of training samples used for modeling and j is the discrete time sample defined from 1 to N. In this LS-SVM formulation, γ is a diagonal matrix of size n_x that acts as the regularization parameter. The Lagrangian function could then be calculated based on Eq. 5.7 as

$$L(W) = \frac{1}{2}||W||_2^2 + \frac{1}{2}\sum_{j=1}^{N} e(j)^T \gamma e(j) - \sum_{j=1}^{N} \alpha_j^T \left(W\Phi(j)^T + e(j) - \mathbf{x}(j+1)\right) \tag{5.8}$$

where $\alpha_j^T \in \mathbb{R}^{n_x}$ are the discrete-time Lagrange multipliers. To find the optimum W, the derivatives of the Lagrangian, Eq. 5.8, with respect to optimization variables must be zero as

$$\frac{\partial L}{\partial W} = 0 \;\rightarrow\; W = \sum_{j=1}^{N} \alpha_j \Phi(j) \tag{5.9a}$$

$$\frac{\partial L}{\partial e} = 0 \;\rightarrow\; \alpha_j = \gamma e(j) \tag{5.9b}$$

$$\frac{\partial L}{\partial \alpha} = 0 \;\rightarrow\; x(j+1) = W\Phi(j)^T + e(j) \tag{5.9c}$$

Substituting Eqs. 5.9a and 5.9c into Eq. 5.6 results in

$$x(k+1) = \sum_{j=1}^{N} \alpha_j \underbrace{\Phi(j)\Phi(k)^T}_{[\Omega]} + \gamma^{-1}\alpha(k) \tag{5.10}$$

where $\Phi(j)\Phi(k)^T$ is the kernel matrix, $[\Omega]$, and could be defined as

$$[\Omega] = x(j)^T K\left(p(j), p(k)\right) x(k) + u(j)^T K\left(p(j), p(k)\right) u(k) \tag{5.11}$$

where $K(p(j), p(k))$ is a nonlinear kernel function. Usually, a Radial Basis Function (RBF) kernel, K_{RBF}, is used as the kernel function, which is defined as

$$K_{RBF}\left(p(j), p(k)\right) = \exp\left(-\frac{||p(j) - p(k)||^2}{2\sigma}\right) \tag{5.12}$$

where σ is a free parameter that is tuned during the hyperparameter optimization and $||p(j) - p(k)||^2$ is the L_2 norm between the two feature vectors. Writing Eq. 5.10 in a compact notation yields

$$X = \alpha\Omega + \gamma^{-1}\alpha \tag{5.13}$$

where $X = [x(1) \ \ ... \ \ x(N)]^T$. Solving this equation for α results in

$$\alpha = \left(I_N \odot \gamma^{-1} + \Omega^T \odot I_{n_x}\right)^{-1} X \tag{5.14}$$

where I_N and I_{n_x} indicate the identity matrix in the dimension of a training sample size and $\mathbf{x}$ size and $\odot$ is an element-wise or Kronecker product. By calculating α, the state-space model matrices can be calculated as

$$\begin{aligned} A\left(p(k)\right) &= \sum_{j=1}^{N} \alpha_j \mathbf{x}(j)^T K_{RBF}\left(p(j), p(k)\right) \\ B\left(p(k)\right) &= \sum_{j=1}^{N} \alpha_j \mathbf{u}(j)^T K_{RBF}\left(p(j), p(k)\right) \end{aligned} \tag{5.15}$$

where the j index shows the data used in the training set. Where the model is developed using the training set of $x(j) \quad j \in (1, 2, ..., N)$ and $u(j) \quad j \in (1, 2, ..., N)$. Additionally, the scheduling parameter, p is also given in the training set as $p(j) \quad j \in (1, 2, ..., N)$.

5.1.2 Bayesian Hyperparameters Optimization

The SVM-LPV model has two main hyperparameters: γ, the regularization coefficient, and σ, the kernel free parameters. The cost function of the hyperparameter optimization is defined as

$$J(\gamma, \sigma) = \frac{1}{N_{CV}} \sum_{i=1}^{N_{CV}} \left(\hat{X}(i) - X(i) \right)^2 \tag{5.16}$$

where $\hat{\mathbf{x}}(i)$ is the modeled output and $\mathbf{x}(i)$ is the measured states and N_{CV} is the validation dataset that is used for optimizing the parameters. Bayesian Optimization utilizes the Bayes Theorem to direct a search of a global optimization problem. The cost function versus iteration number for 100 iterations of the Bayesian optimization is shown in Figure 5.2. In this figure, the Bayesian optimization approaches to the global optimum after 76 iterations.

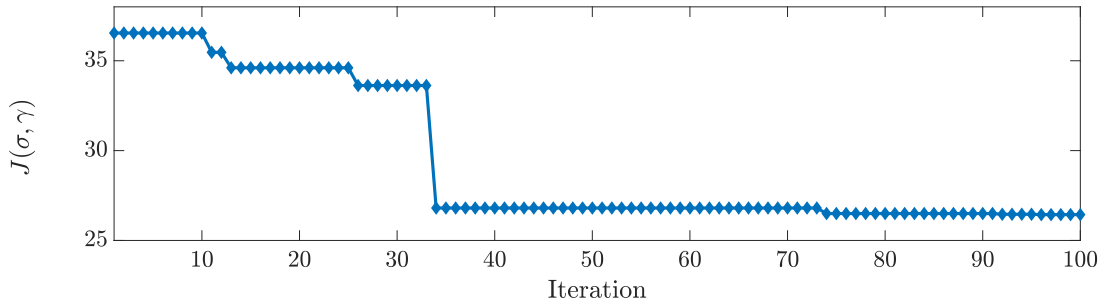


Figure 5.2: Bayesian optimization results for LPV-SVM model parameter optimization showing the cost function (J) values versus the integration number

The Bayesian-optimized SVM-LPV can capture all states with an accuracy of 7.3%, 1.1%, and 1.9% for NO_x, T_{out}, and P_{man} respectively when using the training data. As this model will be used for the MPC, its accuracy for a new data-set is critical. The SVM-LPV is compared to a standard model, a linear state-space model called Autoregressive with Extra Input (ARX). The ARX-based discrete-time state-space model of the diesel engine emissions and performance is trained using the same train-

ing dataset as the SVM-LPV resulting in:

$$
\begin{aligned}
A &= \begin{bmatrix} 0.73 & 7.13 & -0.002 \\ 0.02 \times 10^{-2} & 0.99 & 8.99 \times 10^{-6} \\ -0.61 & 33.94 & 0.91 \end{bmatrix} \\
B &= \begin{bmatrix} 1.27 & -1.09 & 1.01 \times 10^{-5} \\ -0.07 \times 10^{-2} & 0.14 \times 10^{-2} & -1.01 \times 10^{-5} \\ 2.94 & -8.24 & -0.02 \end{bmatrix} \\
C &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{5.17}
$$

The linear ARX and SVM-LVP model are run simultaneously in one simulation. The ML models are compared against the ESM. Both models are evaluated for the test data that is unseen for both models. Figure 5.3 shows the model comparison. Both models have a high accuracy within 5% normalized root mean square error (NRMSE) for estimating the output torque. However, the linear model fails to provide an accurate estimation for intake manifold pressure and NO_x emissions. The accuracy comparison for test data is presented in Table 5.1 For the intake manifold pressure, the SVM-LPV model has significantly better estimations than the linear model where NRMSE is 0.95% in comparison to 14.81% for the linear model. For NO_x, the SVM-LPV estimates with less than 7% error. The linear ARX model is unable to accurately capture the exact emission level, resulting in an NRMSE of 32.3%. Next, using the developed SVM-LPV model, an MPC combustion controller will be designed.

Table 5.1: Comparison of linear and LPV model error for new generated test data

	$\mathbf{NO}_x$	$T_{\mathbf{out}}$	$P_{\mathbf{man}}$
Linear	32.3%	4.04%	14.81%
SVM-LPV	6.95%	3.02%	0.96%

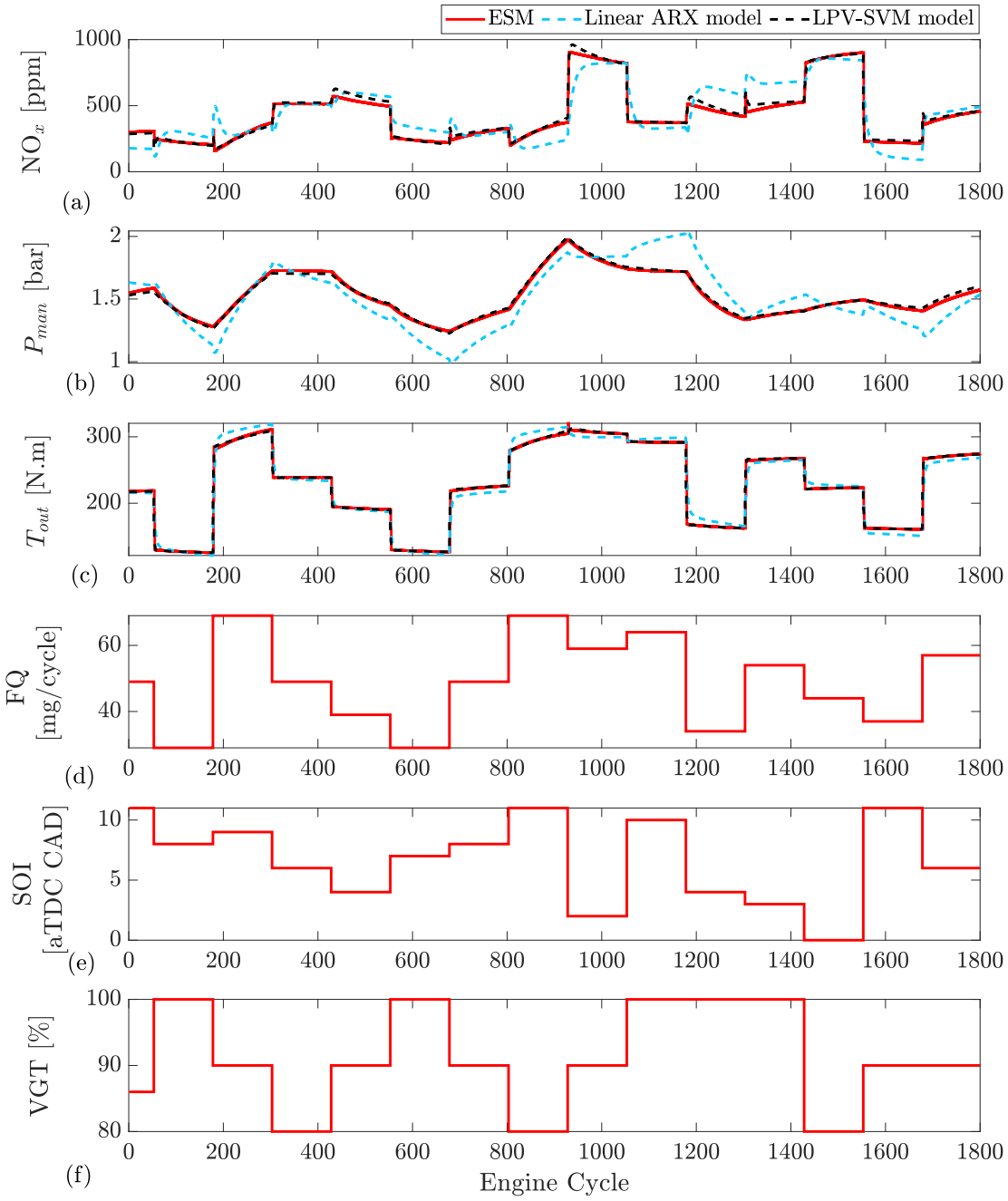


Figure 5.3: Linear ARX, LPV-SVM and ESM comparison for engine-out emissions and performance at engine speed of 1500 rpm: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) Fuel quantity (FQ), e) SOI, f) Variable Geometry Turbine (VGT) rate

5.2 Model Predictive Controller Design

5.2.1 Controller Design

In this section, an MPC controller is designed based on the developed LPV model. The objective of the controller is to minimize engine-out NO_x emissions and fuel consumption while maintaining the desired engine output torque. The cost function $J(\boldsymbol{u}(k), s(k))$ of the finite horizon optimal control problem (OCP) with a horizon length of N_p is defined as

$$\begin{aligned} J(\boldsymbol{u}(k), s(k)) = \sum_{i=0}^{N_p-1} \Big[& \underbrace{||T_{\text{out}}(k+i) - T_{\text{out, ref}}(k+i)||^2_{w_{T_{\text{out}}}}}_{\text{Torque output tracking}} + \underbrace{||\text{NO}_x(k+i)||^2_{w_{\text{NO}_x}}}_{\text{NO}_x \text{ minimizing}} \\ + & \underbrace{||FQ(k+i)||^2_{w_{FQ}}}_{\text{fuel consumption minimizing}} + \underbrace{||u(k+i) - u(k+i-1)||^2_{w_{\Delta u}}}_{\text{control effort penalty}} \\ + & \underbrace{w_s s(k)^2}_{\text{Constraint violation penalty}} \Big] \end{aligned} \tag{5.18}$$

where

$$||.||^2_w = [.]^T w[.] \tag{5.19}$$

where $s(k)$ is a slack variable that is added to the cost function to penalized possible violations of constraints. In this equation, w_v, $v \in [T_{\text{out}}, \text{NO}_x, \text{FQ}, \Delta u, s]$ are the MPC weights. The optimization decision, $\boldsymbol{u}(k)$, is defined as

$$\boldsymbol{u}(k) = [u(k)^T \;\; u(k+1)^T \;\; ... \;\; u(k+p-1)^T] \tag{5.20}$$

Based on the defined cost function, the OCP that is solved at each discrete-time instance as

$$\begin{aligned} \min_{\boldsymbol{u}(k), s(k)} \quad & J(\boldsymbol{u}(k), s(k)) \\ \text{s.t.} \quad & x(0) = \bar{x}(0) \\ & x(k+1) = f(x(k), u(k)) \quad k = 0, \ldots, N_p - 1 \\ & \underline{x} \leq x(k) \leq \overline{x} \qquad\qquad k = 0, \ldots, N_p \\ & \underline{u} \leq u(k) \leq \overline{u} \qquad\qquad k = 0, \ldots, N_p - 1 \end{aligned} \tag{5.21}$$

Where $f(x(k), u(k))$ are based on Eq. 5.3 and the SVM-LPV model of the system and x are the states of the model. A five-step prediction horizon N_p and a single step control horizon are used. For linear MPC, A and B are constant matrices.

The optimization of LPV-MPC is subject to the constraints listed in Table 5.2. The 500 ppm maximum NO_x is chosen as the upper limit on NO_x to keep the emissions below the maximum experimentally measured NO_x output from the Cummins calibrated engine controller. However, this constraint could be adjusted depending on emission legislation. The limit on FQ is used to prevent exceeding the fuel flammability limit. Limits in SOI are added to avoid early combustion (which could lead to combustion noise or knock) and late combustion (which can lead to low thermal efficiency and high exhaust gas temperatures). The turbocharger characteristic map is used to set the VGT limit.

Table 5.2: LPV-MPC constraint Values

Min Value $(\underline{x}, \underline{u})$	Variable (x, u)	Max Value $(\overline{x}, \overline{u})$
0 ppm	NO_x	500 ppm
10 mg/cycle	FQ	80 mg/cycle
-2 aTDC CAD	SOI	11 aTDC CAD
70 %	VGT	100 %

For the linear MPC, the model dynamics of the discrete-time state-space model given in Eq. 8.5 are utilized as $x_{k+1} = Ax_k + Bu_k$. The linear MPC formulation has been implemented in MATLAB© using the MATLAB© Linear MPC block, `mpc(sys)` and the corresponding linear MPC block in Simulink.

For the LPV model, the model matrices are changed based on scheduling parameters identified using the SVM techniques described above. Here the model dynamics, which are utilized as $x_{k+1} = f(x_k, u_k)$, are given in Eq. (5.3). For this, the MPC structure is defined in the MATLAB MPC toolbox©. Then using the MATLAB© Adaptive MPC block, the system matrices A and B are updated using the defined

scheduling parameters. For both the linear and LPV MPC, the `fmincon` Matlab© function has been used. For both LPV MPC and linear MPC, the interior-point (IP) algorithm is used in `fmincon` solver. In both of these models, the state vector is defined as

$$x(k) = \begin{bmatrix} T_{\text{out}}(k) & P_{\text{man}} & \text{NO}_x(k) \end{bmatrix}^T \tag{5.22}$$

The weights of the Linear and LPV MPC values are $w_{T_{\text{out}}} = 0.009$, $w_{\text{NO}_x} = 0.0004$, $w_{FQ} = 0.06$, $w_{\Delta u} = 0.1$. The constraint softening value was set as 0.1 which indicates hard constraints.

5.2.2 Controller Results

The Controllers are compared in this section. They are the developed MPC based on the SVM-LPV model, the linear ARX-based linear MPC (LMPC) and benchmark (BM) ESM calibrated ECU based on the Cummins production ECU. The results are compared in Figure 5.4 where the reference input is shown as well as the contrarians. Except for a slight violation in NO_x constraints (for example at engine cycle 600), both the LMPC and LPV-MPC controllers are able to keep NO_x emissions below the specified constraint. For the NO_x emissions, both the linear MPC and BM controller have higher overall emissions levels than the LPV-MPC controller. This is attributed to the linear model used in the linear MPC that is unable to capture the non-linear NO_x formation trends. The torque (T_{out}) tracking of the controllers are shown for a step up and a step down in Figure 5.5. Acceptable performance is achieved for both the LMPC and LPV-MPC. It is interesting to note that even though the BM uses the most fuel, it has a torque offset from the steady-state reference.

As shown, both the LPV-MPC and Linear MPC (LMPC) tend to generate late injection timings which reduces the peak combustion temperature, resulting in lower NO_x levels. However, this late combustion phasing can result in lower thermal efficiency and higher fuel consumption. For this reason an upper bound is added to the

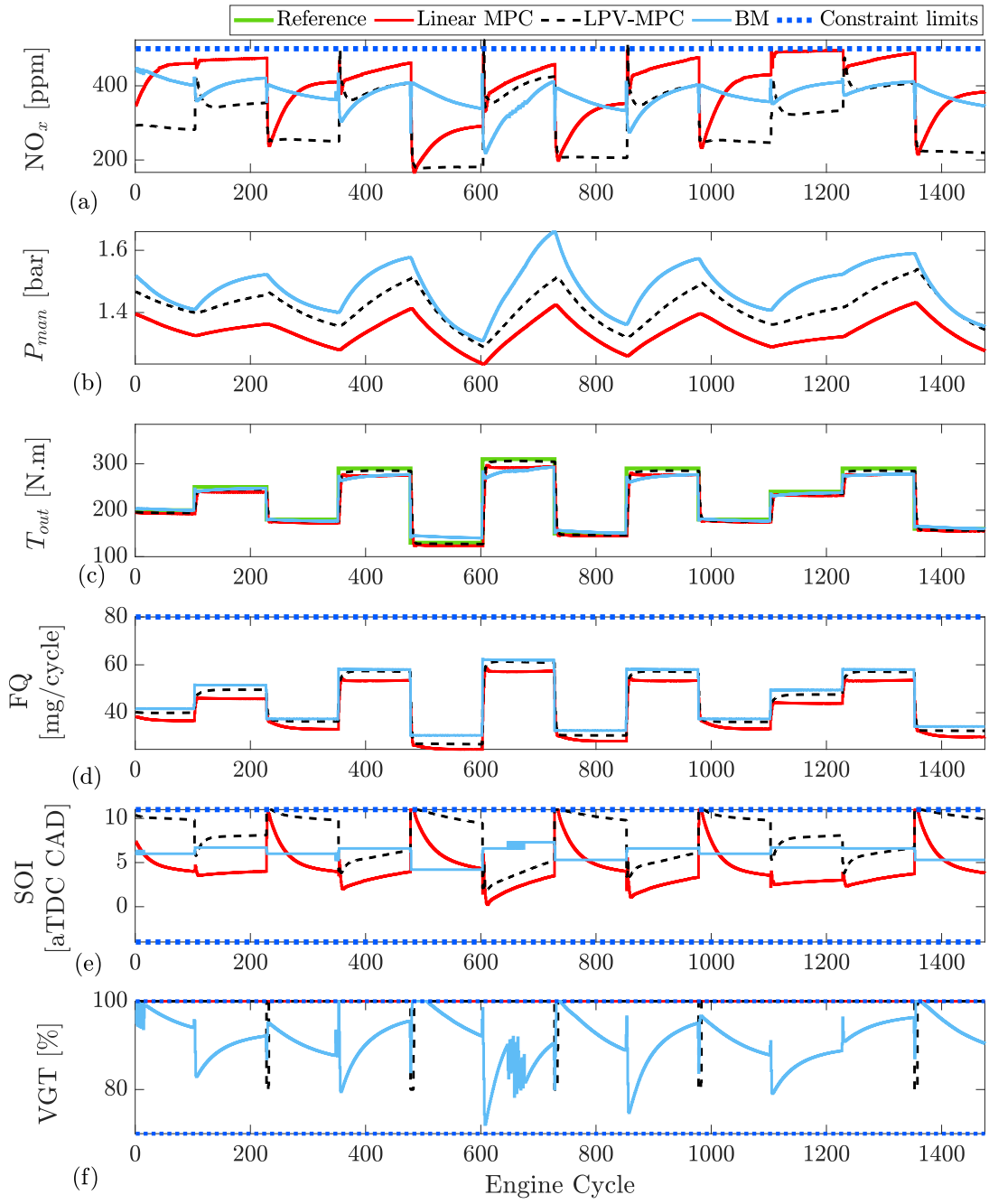


Figure 5.4: Linear MPC, LPV-MPC and Benchmark comparison in 1200 rpm: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate

SOI. The output torque (T_{out}) tracking performance is within 5% for all three controllers. The feed-forward based controller (refereed as BM) controller fails to reach

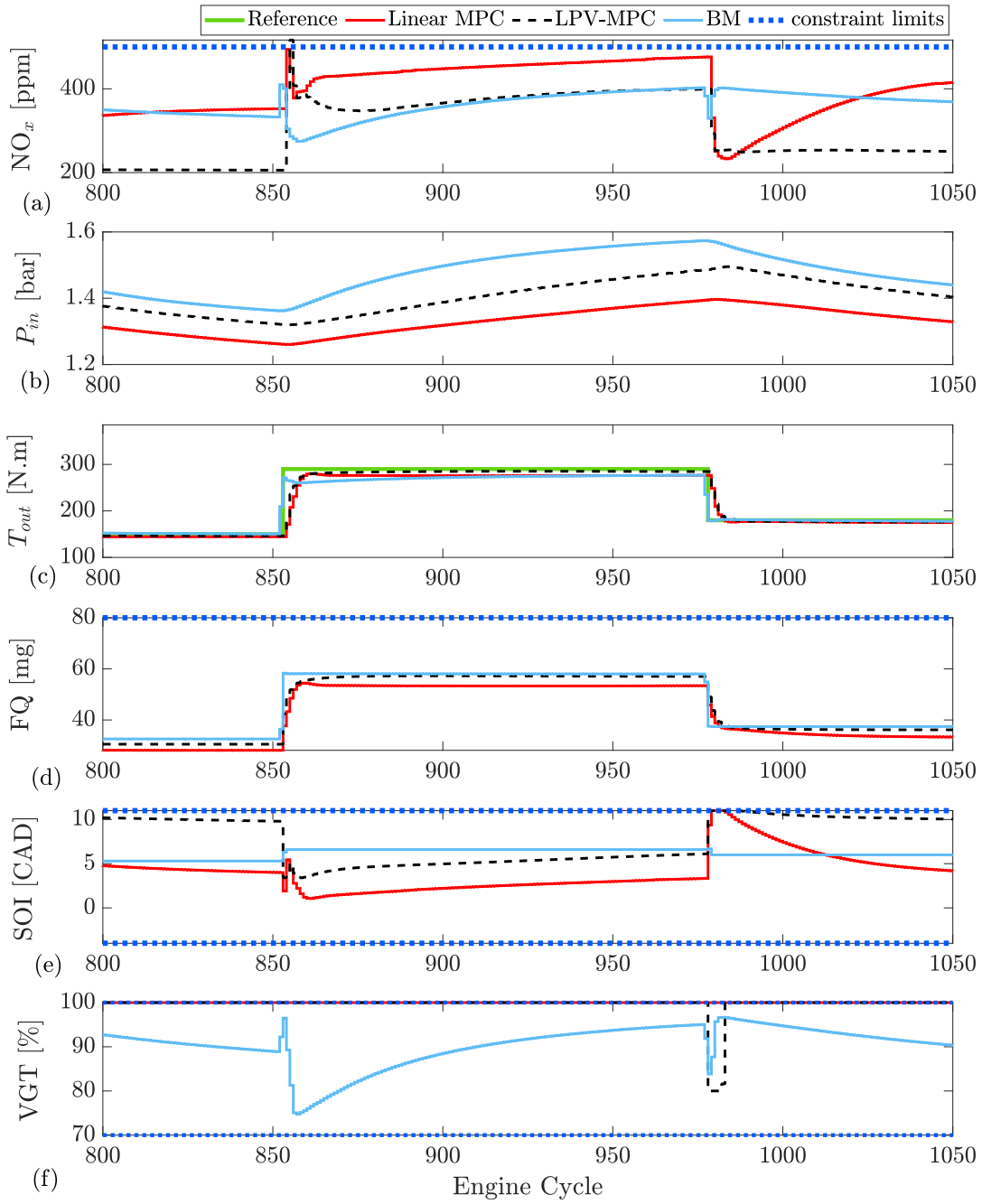


Figure 5.5: Linear MPC, LPV-MPC and Benchmark comparison in 1200 rpm zoomed from 800 to 1050 cycles: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate

the target torque and remains slightly below the reference on the step up and above the reference on the step down in Figures 5.4 and 5.5.

The LPV-SVM model contains gain-scheduling matrices A and B which are dependent on the inputs SOI and FQ. The scheduling parameters as a function of inputs for matrix A and B are shown in Figures 5.6-5.7 which shows that the relationship between the model inputs and the scheduling parameters are non-linear for the diesel combustion process. This non-linearity of the gain scheduling variables of the LPV-SVM model are an advantage of using the LPV model for combustion instead of using only a few points of linearization. The gain scheduling matrix B is similarly nonlinear.

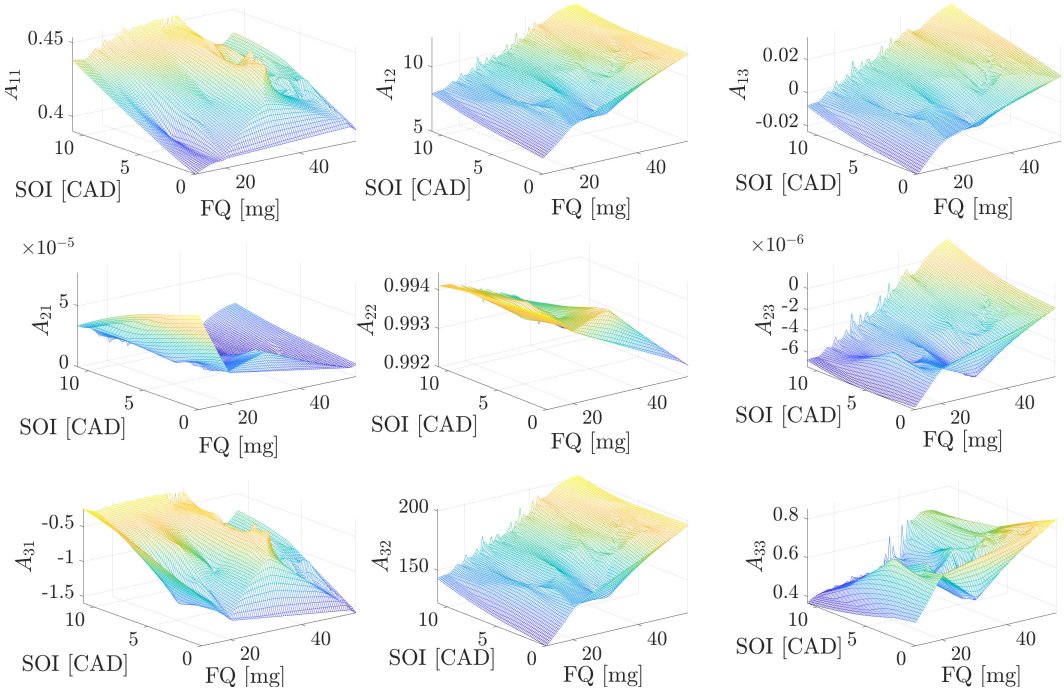


Figure 5.6: "A" matrix elements for the LPV-SVM model at an engine speed of 1500 rpm

The LPV model is developed at a constant speed of 1500 rpm. To evaluate the robustness of the controllers, each controller with the same constraints are tested at an engine speed of 1200 rpm. As shown in Figure 5.8 (zoomed version from 450 to 650 is shown in Figure 5.9), both the LMPC and LPV-MPC perform significantly better than the BM. Here the benchmark controller tends to advance injection timing at lower speeds which results in significant increases in NO_x emissions. Due the increased accuracy of the LPV model, the LPV-MPC performs slightly better. In the

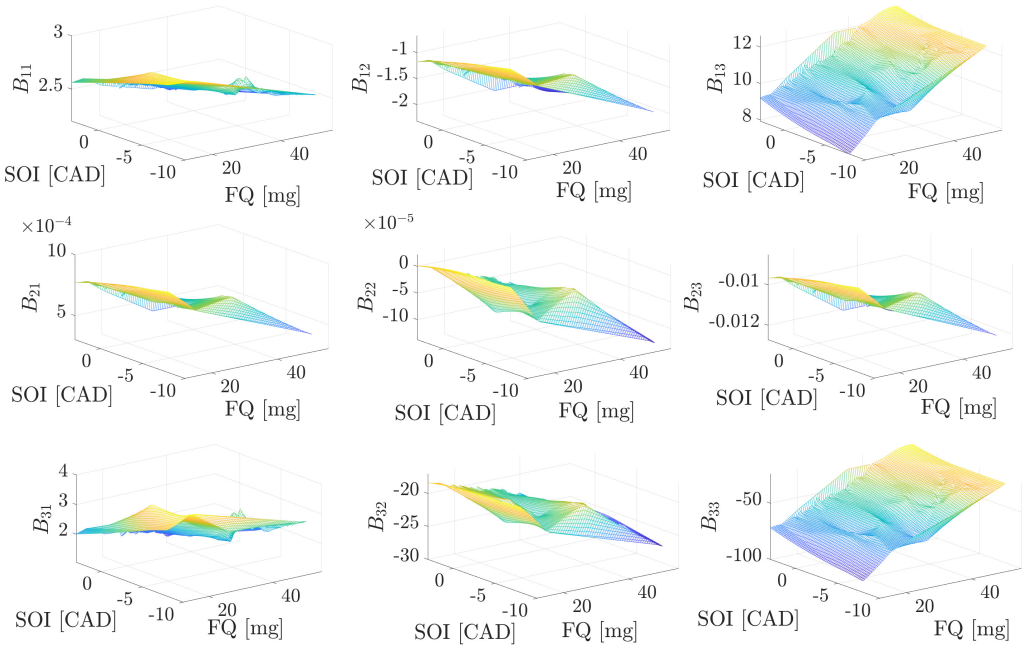


Figure 5.7: "B" matrix elements for the LPV-SVM model at an engine speed of 1500 rpm

next section MPC will be replaced by imitation ML to reduce computation.

5.3 Imitation of MPC using a Deep Neural Network

5.3.1 Imitation of MPC Concept

ML was used to model the system described in Section 5.2. In this section, ML is used to replace the MPC with a learning controller called imitative MPC. This is used to avoid the high computational time of MPC, that requires solving MPC optimization online. Instead, a function, in this case a deep network, is trained to approximate the MPC and deploy it with a much lower computational cost.

The schematic of imitative MPC was previously shown in Figure 5.1 (step 4). First, the LPV-MPC are implemented on ESM. Second, the input and output are recorded, and a deep neural network, including a Long-Short-Term Memory (LSTM) layer, are

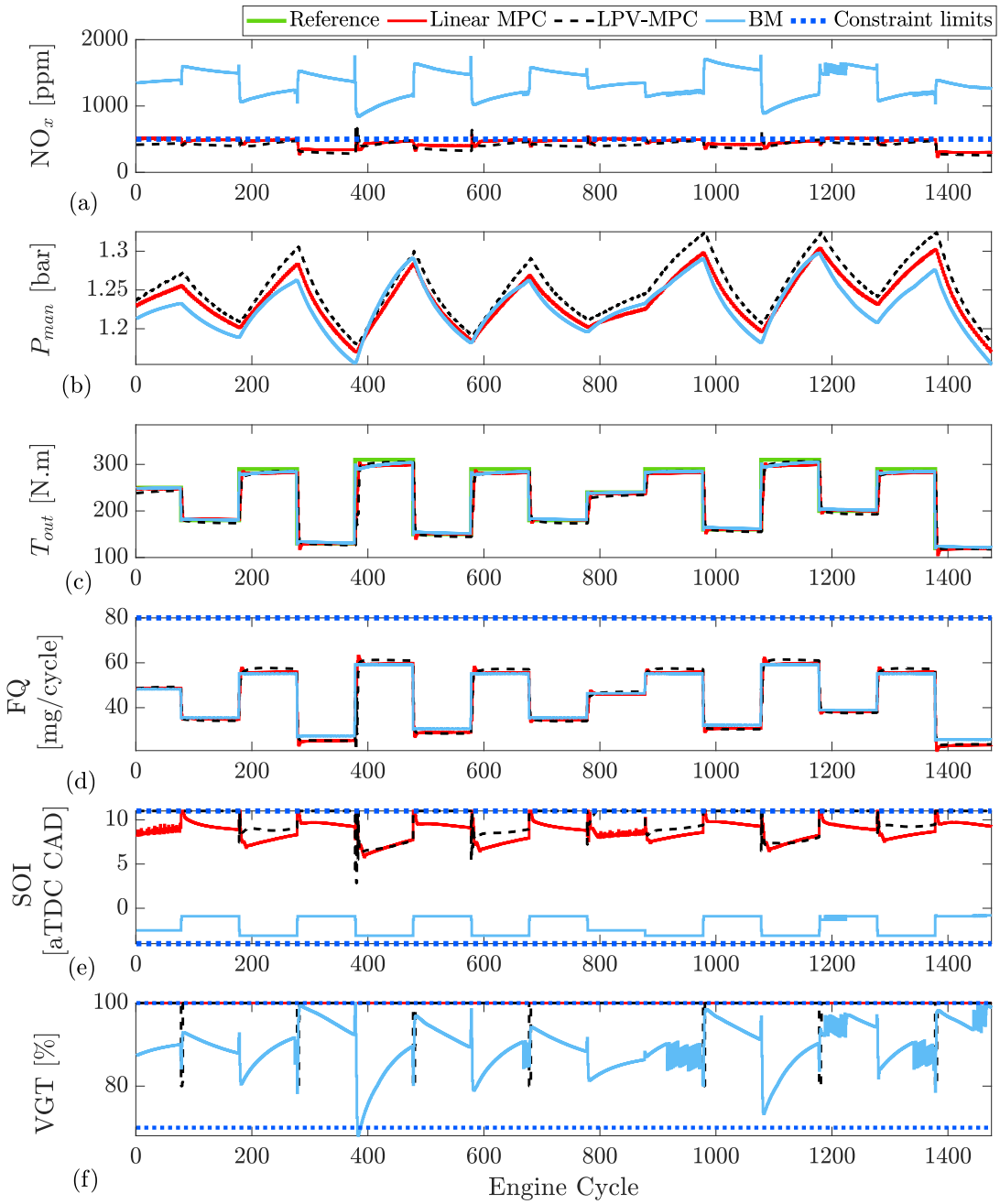


Figure 5.8: Linear MPC, LPV-MPC, and Benchmark comparison in 1200 rpm: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) FQ, e) SOI, f) VGT rate

used to mimic the behavior of the MPC as shown schematically in Figure 5.10. The inputs of this network are engine output torque, T_{out}, the error in output torque, $e_{T_{\text{out}}}$, engine-out NO_x, intake manifold pressure, P_{man}, and engine speed n_{rpm}. The outputs

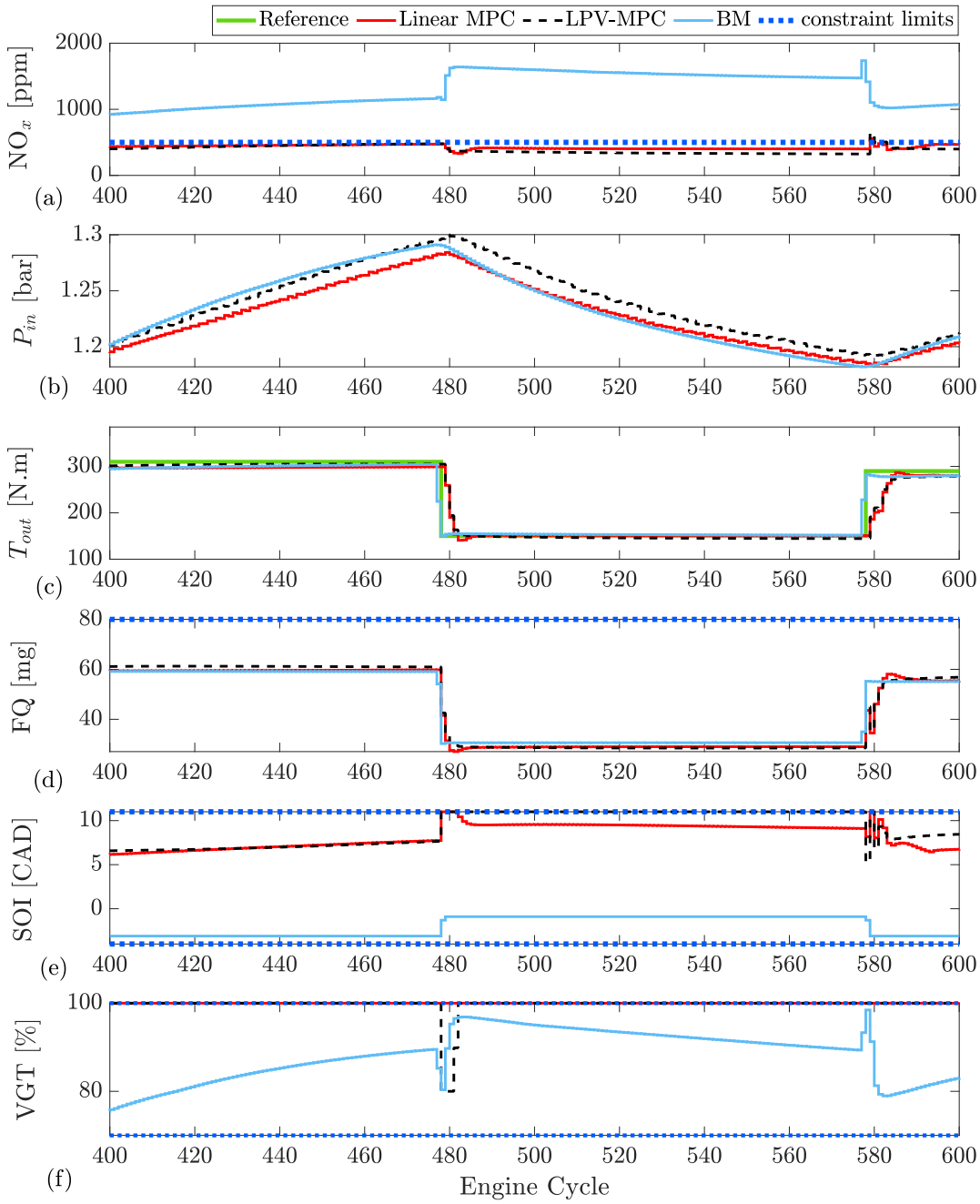


Figure 5.9: Linear MPC, LPV-MPC, and Benchmark comparison in 1200 rpm zoomed from 800 to 1050 cycles: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) FQ, e) SOI, f) VGT rate

are estimated (FQ), (SOI), and VGT. The outputs are denoted with a "hat", i.e., $\hat{\text{FQ}}(k)$, $\hat{\text{SOI}}(k)$, and $\hat{\text{VGT}}$. This network includes four main layers where the first, third and fourth layer are fully connected (FC) layers with a layer size (neuron) of 32.

The second layer is an LSTM layer with the same layer size. The reason for using an FC layer around the LSTM to create a deep network is to increase the complexity of the model without increasing the hidden and cell states of the entire network. Finally, the online MPC is replaced with the designed imitative MPC. More details on the LSTM are in the next section.

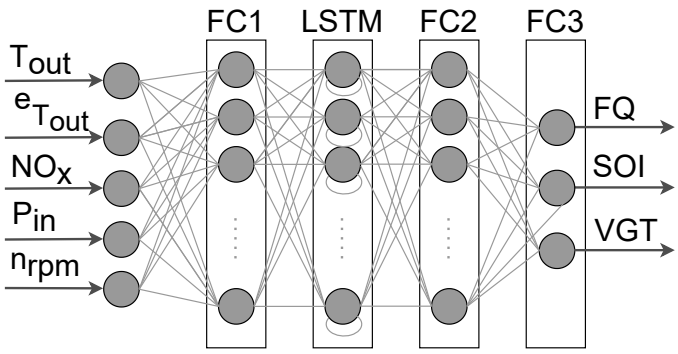


Figure 5.10: Structure of network for imitation of LPV-MPC

5.3.2 Forward Propagation of Imitative Controller

LSTM utilizes a hidden state that is split into two main parts as shown schematically in Figure 5.11: $h(k)$ the short-term state, and $c(k)$ the long-term state. The long-term state $c(k-1)$ travels though the network and first enters the forget gate $f(k)$ where past values are dropped. Then, additional values (memories) are added to the input gate $i(k)$ at each time step (k). Therefore at each time step some data is added, and some is dropped. Further, after adding new memory, the long-term state is replicated, passed into the hyperbolic tangent activation function (tanh), and the output gate filters the result to generate the short-term state $h(k)$ (equal to the cell's output $y(k)$ for this time step) [154].

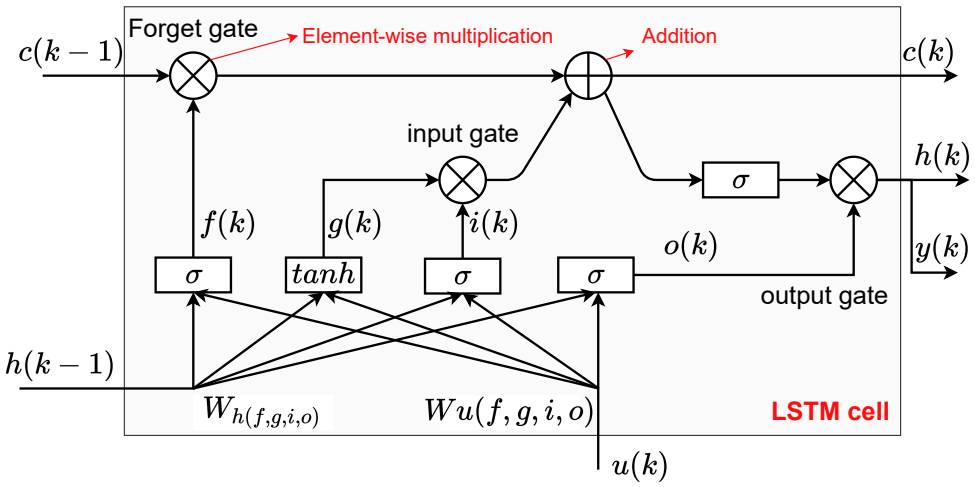


Figure 5.11: Long-Short-Term Memory (LSTM) cell structure

LSTM computations can be summarized as:

$$\begin{aligned}
i(k) &= \sigma\left(W_{ui}^T u(k) + W_{hi}^T h(k-1) + b_i\right) \\
f(k) &= \sigma\left(W_{uf}^T u(k) + W_{hf}^T h(k-1) + b_f\right) \\
g(k) &= \tanh\left(W_{ug}^T u(k) + W_{hg}^T h(k-1) + b_g\right) \\
o(k) &= \sigma\left(W_{uo}^T u(k) + W_{ho}^T h(k-1) + b_o\right) \\
c(k) &= f(k) \odot c(k-1) + i(k) \odot g(k) \\
h(k) &= y(k) = o(k) \odot \tanh\left(c(k)\right)
\end{aligned} \tag{5.23}$$

where $W_{u(f,g,i,o)}$ are the weight matrices applied to the input vector $u(k)$ and $W_{h(f,g,i,o)}$ are weight matrices of the previous short-term state $h(k)$. In this equation, $\odot$, is an element-wise multiplication and $b_{(f,g,i,o)}$ are the biases. In Eq. 5.23, $i(k)$ is the input gate, $f(k)$ is the forget gate, $g(k)$ is the cell candidate, $o(k)$ is the output gate, $c(k)$ is the cell state, and $h(k)$ is the hidden state. Two activation functions are used in Eq. 5.23 which are given as:

i) $\tanh(z)$ activation function:

$$\tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{5.24}$$

ii) $\sigma(z)$ activation function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{5.25}$$

The Fully Connected (FC) layers are added around the LSTM layer to increase the network capacity to better estimate the nonlinearity of the engine emissions and performance without increasing the number of hidden and cell states. An FC layer equation with a Rectified Linear Unit (ReLU) activation function is defined as

$$z_{FC}(k) = \text{ReLU}(W_{FC}^T u(k) + b_{FC}) \tag{5.26}$$

where ReLU is an activation function defined as

$$\text{ReLU} = \begin{cases} 0 & \text{if } z \leq 0 \\ z & \text{if } z > 0 \end{cases} \tag{5.27}$$

Therefore, the forward propagation of this imitative network is used to replace the online MPC optimization as

$$z_{FC1}(k) = \text{ReLU}\left(W_{FC1}^T u(k) + b_{FC1}\right) \tag{5.28a}$$

$$h_{LSTM}(k) = \underbrace{o_{LSTM}(k) \odot \tanh\left(c_{LSTM}(k)\right)}_{\text{based on Eq. 5.23}} \tag{5.28b}$$

$$z_{FC2}(k) = \text{ReLU}\left(W_{FC2}^T h_{LSTM}(k) + b_{FC2}\right) \tag{5.28c}$$

$$z_{FC3}(k) = \hat{u}(k) = W_{FC3}^T z_{FC2}(k) + b_{FC3} \tag{5.28d}$$

where $\hat{u}(k)$ is the approximated control variables defined as $[\hat{\text{FQ}}(k)\ \ \hat{\text{SOI}}(k)\ \ \hat{\text{VGT}}]^T$

5.3.3 Training Imitative MPC

The output of the imitative MPC network is the estimated control actions and are

$$\hat{u}(k) = \begin{bmatrix} \hat{\text{FQ}}(k) \\ \hat{\text{SOI}}(k) \\ \hat{\text{VGT}} \end{bmatrix} \tag{5.29}$$

The cost function for this network is as

$$J(W, b) = \frac{1}{m} \sum_{k=1}^{m} \mathcal{L}\left(\hat{u}(k), u(k)\right) + \frac{\lambda}{2m} \sum_{l=1}^{L} ||W^{[l]}||_2^2 \tag{5.30}$$

where $\mathcal{L}(\hat{u}(k), u(k))$ is the loss function, λ is the regularization coefficient, and $||W^{[l]}||_2^2$ is the Euclidean norm which is defined as

$$||W^{[l]}||_2^2 = \sum_{i=1}^{n^{[l]}} \sum_{j=1}^{n^{[l-1]}} (w_{ij}^{[l]})^2 \tag{5.31}$$

The Mean Squared Error (MSE) cost function is used and is defined as

$$\mathcal{L}(\hat{u}(k), u(k)) = \frac{1}{m} \sum_{k=1}^{m} (\hat{u}(k) - u(k))^2 \tag{5.32}$$

The training information along with the design values for the proposed network are summarized in Table 5.3. To train this model, MATLAB Deep Learning Toolbox © has been used with the Adam algorithm. In Figure 5.12, the loss function versus iteration for both the performance and emission networks are shown. Where within a defined number of epochs, the loss functions converges to a minimum. Epochs indicate the number of passes of the entire training dataset the ML algorithm has completed. The validation loss function also converges to match the training function, indicating that there is no overfitting or underfitting of the model. The training accuracy for FQ, SOI, and VGT is 4.3%, 6.3%, and 8.3% while for the validation data are 4.3%, 8.9%, 10.3%.

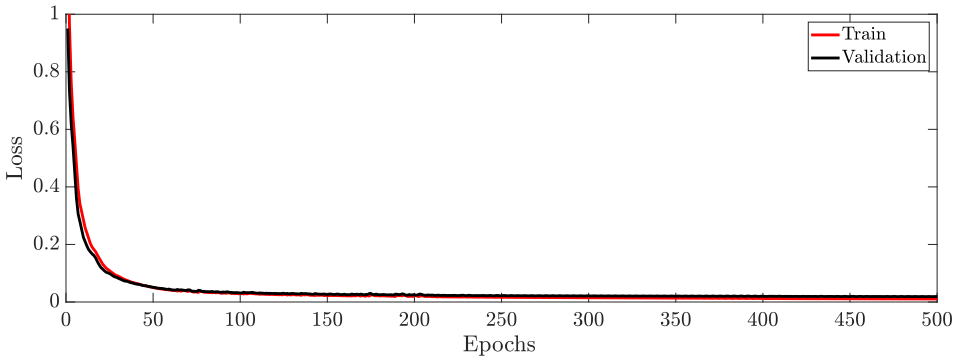


Figure 5.12: Loss vs. epochs for NO_x, torque and pressure model

To test the imitative LPV-MPC, the controller is tested on an unseen reference and compared to the LPV-MPC in Figure 5.13 at 1500 rpm and Figure 5.14 at 1200 rpm.

Table 5.3: Properties of 2-level engine performance and emission LSTM-based model

Name	**Value**
FC(1,2) size	32
FC3 size	3
LSTM size	32
Optimizer	Adam
Maximum Epochs	500
Mini batch size	512
Learn rate drop period	200 Epochs
Learn rate drop factor	0.5
L2 Regularization	0.8
Initial learning rate	0.01
Validation frequency	1
Momentum	0.9
Squared gradient decay	0.99

The results show that the imitative controller can successfully clone the behaviour of LPV-MPC and generate almost the same optimal control without performing the online optimization.

The performance of the controllers at engine speeds of 1500 and 1200 rpm are summarized in Table 5.4. In addition, the performance compassion of the controller to the baseline model are summarized in Table 5.5. As the LPV controller was designed only based on constant speed data at 1500 rpm, the LPV-MPC and imitative controller's performance show reasonable controller robustness to a different engine speed. Here significant NO_x emissions reduction can be seen for all the controllers over the baseline model except for the LMPC model at 1500 rpm which can be attributed to the use of a simplified linear model. In addition to the reduced emissions for all controllers, they maintain or improve fuel consumption compared to the baseline. This demonstrates the advantage of the optimized controllers over the calibration

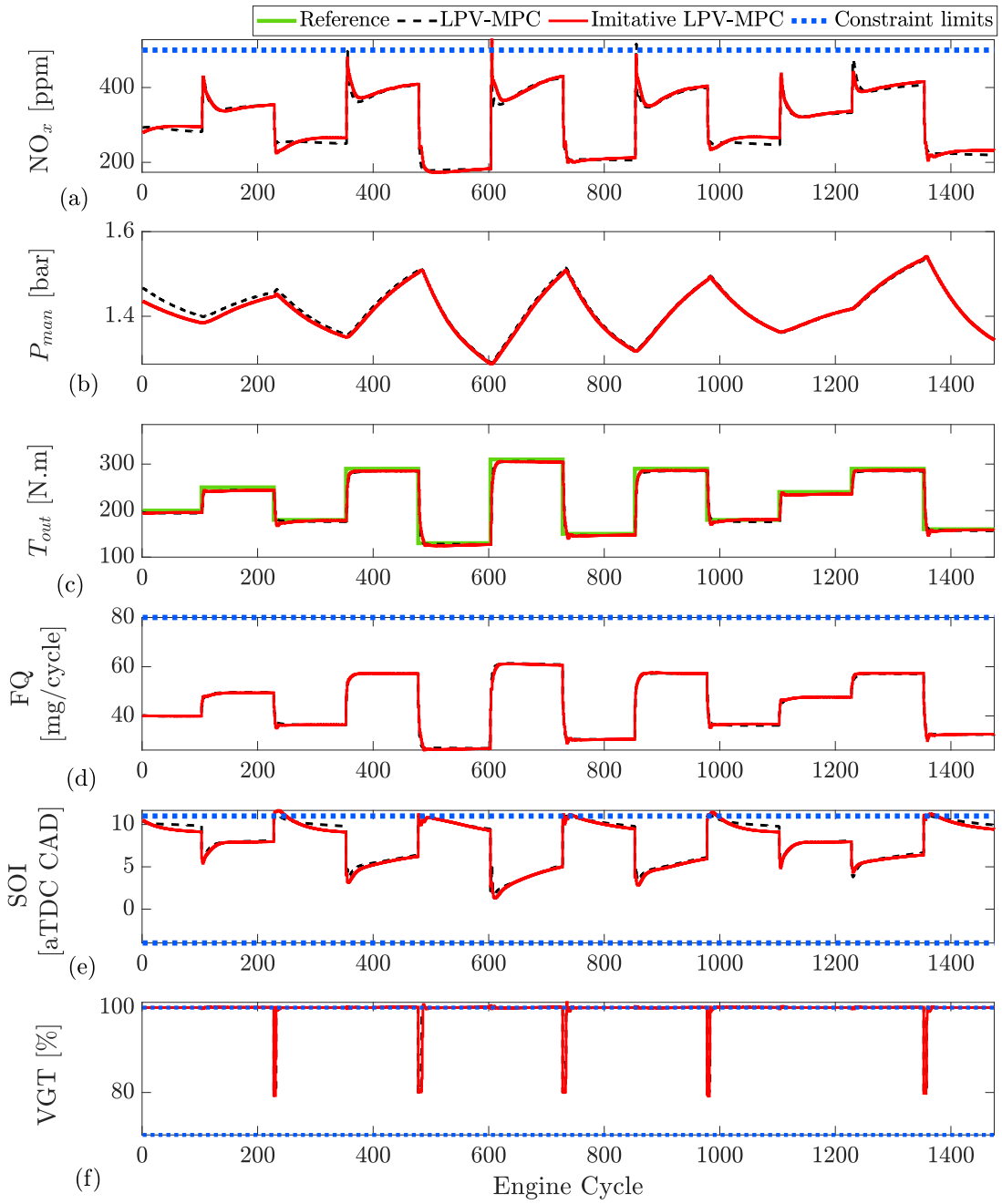


Figure 5.13: LPV-MPC and imitative LPV-MPC comparison in 1500 rpm: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) FQ, e) SOI, f) VGT rate

based baseline. One disadvantage of the developed models is an increase in load tracking error in comparison to the baseline model at 1500 rpm. However, this 2% discrepancy in load tracking results in significant emission reduction of 18-70% and

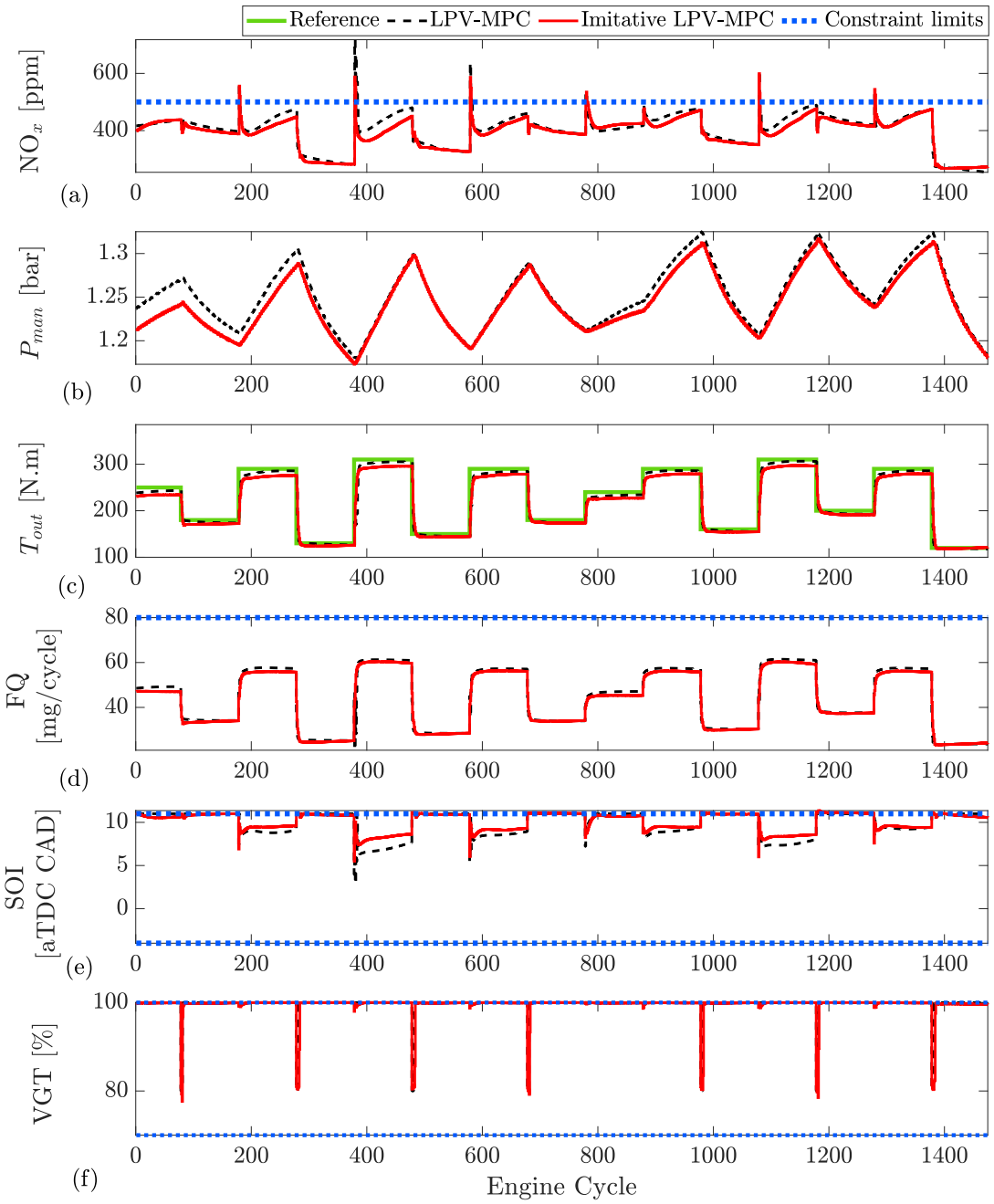


Figure 5.14: LPV-MPC and imitative LPV-MPC comparison in 1200 rpm: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) FQ, e) SOI, f) VGT rate

fuel consumption reduction of 1-10%.

The imitative MPC controllers provide similar improvements to the full MPC controllers compared to the baseline model, while providing significantly improved com-

Table 5.4: Proposed MPC and Imitative MPC results compared to Benchmark (BM) for engine speeds of 1500 and 1200 rpm

1500 rpm						
	Cumulative NOx [ppm]	Average NOx [ppm]	Load error [%]	Cumulative FQ [g]	Average FQ [mg]	Execution time [ms]*
Benchmark	556100.0	376.8	3.95	67.9	46.0	-
LMPC	593500.0	402.1	4.99	61.1	41.4	1.17
LPV-MPC	450570.0	305.3	2.96	65.5	44.4	1.69
Imitative LPV-MPC	455770.0	308.8	2.92	65.5	44.3	0.03
1200 rpm						
	Cumulative NOx [ppm]	Average NOx [ppm]	Load error [%]	Cumulative FQ [g]	Average FQ [mg]	Execution time [s]*
Benchmark	1961900.0	1329.2	2.18	64.8	43.9	-
LMPC	662970.0	449.2	3.16	64.0	43.4	1.93
LPV-MPC	593040.0	401.8	3.06	64.8	43.9	2.62
Imitative LPV-MPC	579220.0	392.4	5.74	63.6	43.1	0.03

* time per cycle

Table 5.5: Percentage of difference for proposed MPC and Imitative MPC with respect to the Benchmark for engine speeds of 1500 and 1200 rpm. Negative means that controller's performance is better than that of to BM

1500 rpm			
	NO_x [%]	FQ [%]	load tracking performance [%]
LMPC	+6.71	-10.00	-1.04
LPV-MPC	-18.98	-3.48	+0.99
Imitative LPV-MPV	-18.05	-3.70	+1.03
1200 rpm			
	NO_x [%]	FQ [%]	load tracking performance [%]
LMPC	-66.20	-1.23	-0.98
LPV-MPC	-69.77	0.00	-0.88
Imitative LPV-MPV	-70.48	-1.85	-3.56

putational times. As presented in Table 5.5, the imitative controllers are 50 and 77 times faster than online MPC optimization at 1500 and 1200 rpm, respectively. All these simulations carried on a computer equipped with an Intel Core i7-6700K

processor with 32.0 GB of RAM.

5.4 Summary of chapter

This chapter presents the integration of ML and MPC for both modeling and controller implementation for diesel engine applications. First, an SVM-based LPV model is developed to design an LPV-MPC. The LPV model showed better prediction accuracy for all engine outputs in test data compared to the linear baseline model. Using these models a linear MPC and an LPV-MPC are designed. Then, the LPV-MPC is implemented in an ESM simulation and the controller input and output data are collected from the MPC. This input-output data is used to train a deep neural network. After testing the imitative MPC controller at two different engine speeds, the imitative controller performs very closely to the full online optimized MPC performance but with a significant reduction in processing time. In addition, the MPC and imitative MPC showed significant improvements in NO_x emissions and a reduction in fuel consumption while providing similar load following capabilities as the feed-forward baseline production controller. Both the LPV-MPC and imitative controller are able to reduce NO_x emissions by 18-70% while reducing fuel consumption by 1-10% compared to the Cummins production controller and the imitative controller, which both require 1/50 computation time compared to online MPC optimization.

Chapter 6

Integration of Deep Learning and Nonlinear Model Predictive Control: Simulation Results [1]

Deep learning and nonlinear model predictive control (NMPC) are used in this chapter to minimize the emissions and fuel consumption of a compression ignition engine. In this chapter, similar to Chapter 5, deep learning is used for two applications: i) identify a model for MPC, ii) imitation of MPC. In Chapter 5, an SVM-based algorithm is used to model a system which results in a linear parameter varying parameter MPC. To identify the model for MPC in this chapter, a deep recurrent neural network including long-short-term memory (LSTM) layers to model the emission and performance of a compression ignition engine are used. This model is then used for model predictive controller implementation. As this model is a nonlinear model, a nonlinear version of MPC is then implemented. A novel scheme is used by augmenting hidden and cell states of the network in an NMPC optimization problem (LSTM-NMPC) that combined LSTM with MPC. For imitation of MPC, the developed LSTM-NMPC is used to train an imitation controller. A deep learning network is deployed to clone the behavior of the developed controller. The LSTM-NMPC and the imitative LSTM-NMPC are then compared with the calibrated Cummins ECU model in an engine simulation model (ESM). This chapter differs from Chapter 5 in

[1] This chapter is based on [7]

that a different method (deep learning) to identify the model for MPC is used.

Schematics of the procedure followed in this chapter and the main chapter sections are shown in Figure 6.1. The first stage was explained in Chapter 2, and a detailed physical-based model in GT-power has been developed and parameterized with experimental data. This results in the Engine Simulation Model (ESM) in GT-power. Then, randomly generated inputs are used to drive the ESM model and the ESM outputs are recorded for modeling. Next, using input-output pairs of data, an LSTM model is developed and used for the design of the NMPC controllers (the ESM is too complex for the MPC design). Finally, to reduce the computational time of the NMPC, the controllers output is used to train the ML-based imitation controllers. All of the controllers developed in this chapter are simulated using the ESM for co-simulation. The real-time implementation of the LSTM-NMPC on the engine is the subject of Chapter 7.

6.1 Long-Short Term Memory Network (LSTM) Model

The details and the LSTM equation have been described in Section 5.3. Here, the LSTM will be used for both modeling and imitation of the MPC. Through a design process, a 2-level deep network is proposed, including two series networks, one for predicting intake manifold pressure and output torque and another for NO_x emissions. Each network contains one LSTM layer and three fully connected (FC) layers. The structure of this network is shown schematically in Figure 6.2. The main reason for having two separate networks is to incorporate a physical understanding of the system. As NO_x is created during combustion and is usually measured through a sensor after each combustion cycle, it depends not only on $u(k)$ but also on intermediate states such as intake manifold pressure. For modeling NO_x, having output torque is also helpful and adds additional features to improve prediction accuracy. Based on physical understanding, NO_x depends on all five features of the NO_x prediction

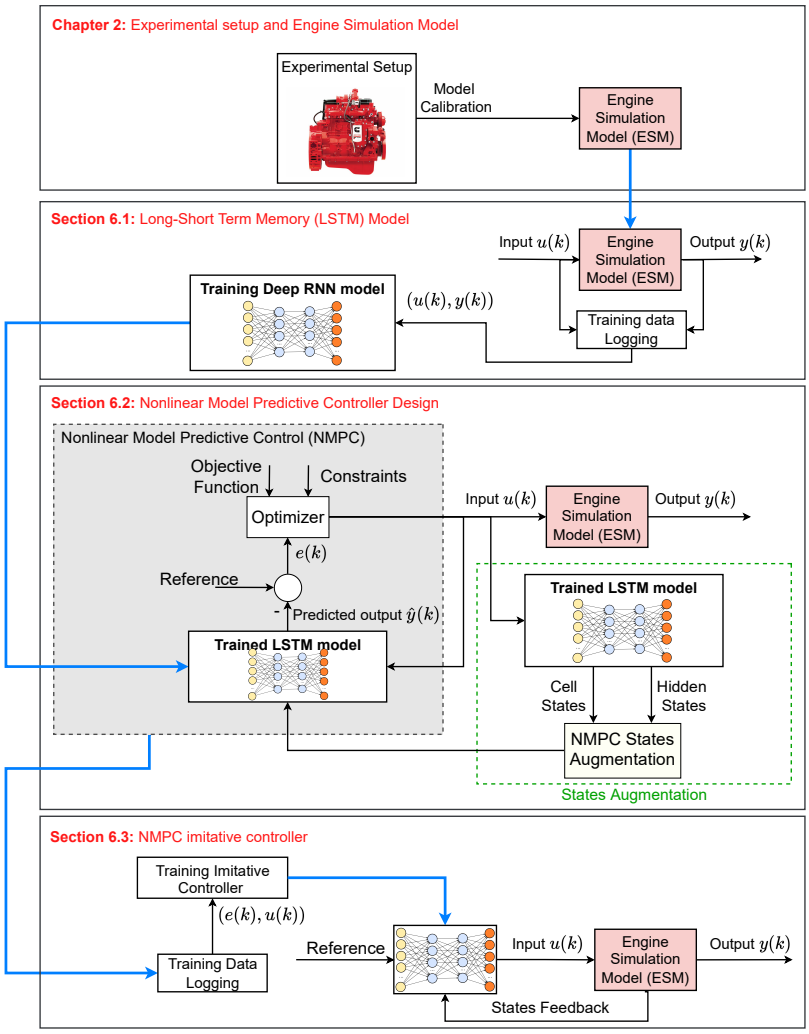


Figure 6.1: Modeling and controller procedure based on Engine Simulation Model (ESM) for LSTM model and corresponding LSTM imitation controller

network, but the other two outputs, including output torque and intake manifold pressure, are not dependent on NO_x. As LSTM is used in this architecture and has recurrent behavior, adding all outputs to a single network makes intake manifold pressure and output torque function of NO_x, which does not correspond to the physics. The FC layers are added around the LSTM layer to increase the network capacity to better estimate the nonlinearity of the engine emissions and performance without increasing the number of hidden and cell states (see Section 5.3).

Because the state and output functions have the form of $x(k+1) = F(x(k), u(k))$ and $y(k) = F(x(k), u(k))$ repetitively and must be imported into the NMPC frame-

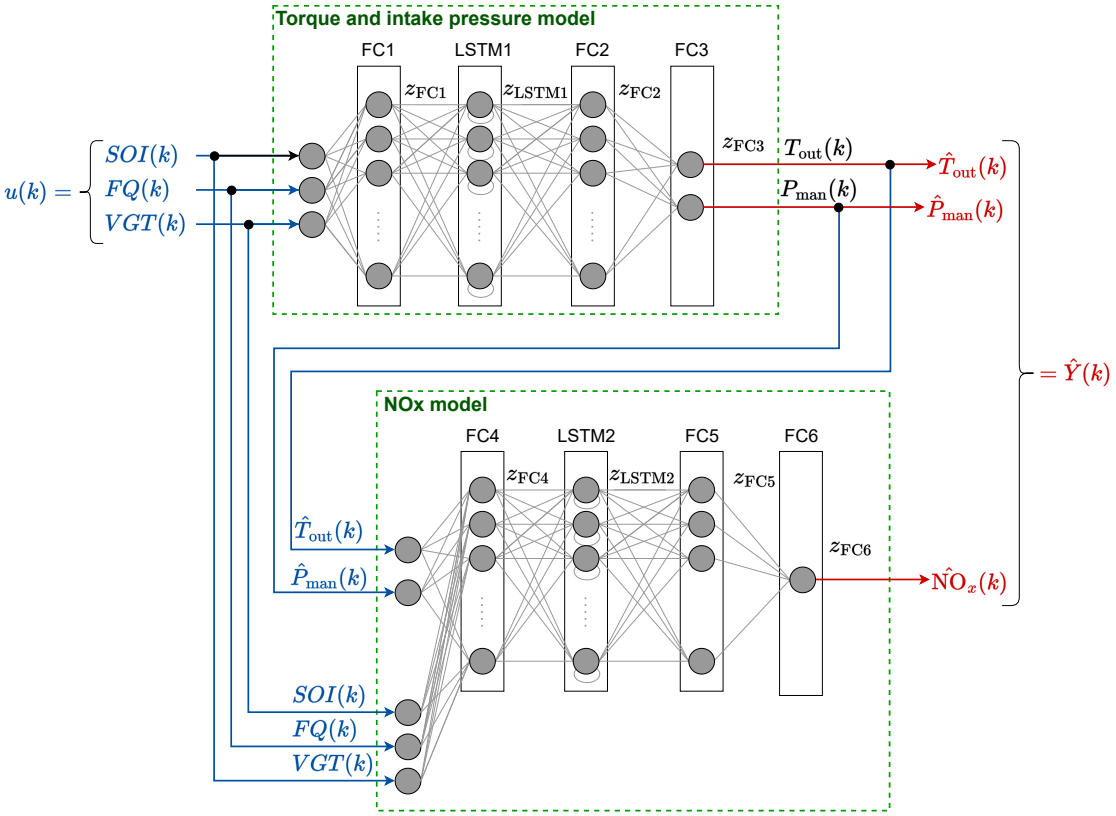


Figure 6.2: Structure of proposed deep neural network model for engine performance and emission modeling. FC: Fully connected layer, LSTM: Long-short term memory.

work the equations for a predictive model of MPC are presented here. To generate state and output functions, the forward propagation needs to be evaluated for the network shown in Figure 6.2. The forward propagation of the first part of the proposed network is used to estimate engine-out torque and intake manifold pressure and is calculated using the LSTM computation (Eq. 5.23) and the FC layer computation (Eq. 5.26) as

$$z_{FC1}(k) = \text{ReLU}\left(W_{FC1}^T u(k) + b_{FC1}\right) \tag{6.1a}$$

$$z_{LSTM1}(k) = h_{LSTM1}(k) = \underbrace{o_{LSTM1}(k) \odot \tanh\left(c_{LSTM1}(k)\right)}_{\text{based on Eq. 5.23}} \tag{6.1b}$$

$$z_{FC2}(k) = \text{ReLU}\left(W_{FC2}^T z_{LSTM1}(k) + b_{FC2}\right) \tag{6.1c}$$

$$\underbrace{z_{FC3}(k)}_{[T_{out}(k) \quad P_{man}(k)]^T} = W_{FC3}^T z_{FC2}(k) + b_{FC3} \tag{6.1d}$$

where each equation refers to the output of each layer in the top part of Figure 6.2

where $u(k)$ are the system inputs which are defined as

$$u(k) = [\ \mathrm{FQ}(k) \quad \mathrm{SOI}(k) \quad \mathrm{VGT}(k)\ ]^T \tag{6.2}$$

The estimated intake manifold pressure $\hat{P}_{\mathrm{man}}$ and output torque $\hat{T}_{\mathrm{out}}$ are then appended to the system inputs $u(k)$ to estimate the engine-out $\mathrm{NO_x}$ emissions in the lower part of Figure 6.2 as

$$z_{FC4}(k) = \mathrm{ReLU}(W_{FC4}^T \begin{bmatrix} u(k) & \underbrace{z_{FC3}(k)}_{[T_{\mathrm{out}}(k) \quad P_{\mathrm{man}}(k)]^T} \end{bmatrix} + b_{FC4}) \tag{6.3a}$$

$$z_{LSTM2}(k) = h_{LSTM2}(k) = \underbrace{o_{LSTM2}(k) \odot \tanh\left(c_{LSTM2}(k)\right)}_{\text{based on Eq. 5.23}} \tag{6.3b}$$

$$z_{FC5}(k) = \mathrm{ReLU}(W_{FC5}^T z_{LSTM2}(k) + b_{FC5}) \tag{6.3c}$$

$$\underbrace{z_{FC6}(k)}_{\mathrm{NO}_x(k)} = W_{FC6}^T z_{FC5}(k) + b_{FC6} \tag{6.3d}$$

The output of this network can be calculated as

$$\hat{Y}(k) = [\underbrace{\hat{T}_{\mathrm{out}}(k) \quad \hat{P}_{\mathrm{man}}(k)}_{z_{FC3}(k)} \quad \underbrace{\hat{\mathrm{NO}}_x(k)}_{z_{FC6}(k)}]^T = \begin{bmatrix} z_{FC3}(k) \\ z_{FC6}(k) \end{bmatrix} \tag{6.4}$$

The NMPC states of this model including hidden states and cell states, are

$$x(k) = \begin{bmatrix} h_{LSTM1}(k) \in \mathbb{R}^{26} \\ h_{LSTM2}(k) \in \mathbb{R}^{26} \\ c_{LSTM1}(k) \in \mathbb{R}^{26} \\ c_{LSTM2}(k) \in \mathbb{R}^{26} \end{bmatrix} \in \mathbb{R}^{104} \tag{6.5}$$

For training, the same cost and loss function described in Section 5.3.3 is used, but with different hyperparameters and numbers of hidden units. The training information along with the design values for the proposed network are summarized in Table 6.1. As described before, to train this model, MATLAB Deep Learning Toolbox© has been used with the Adam algorithm. In Figure 6.3, the loss function versus iteration

for both the performance (the load and pressure) and emission (NO_x) networks are shown. Within a defined number of epochs, the loss functions converges to a minimum. The validation loss function also converges to match the training function, indicating that no overfitting or underfitting of the model has occurred.

Table 6.1: Properties of 2-level engine performance and emission shown in Figure 6.2

Name	**Value**
FC(1,2,4,5) size	26
FC3 size	2
FC6 size	1
LSTM(1,2) size	26
Optimizer	Adam
Maximum Epochs	500
Mini batch size	512
Learn rate drop period	150 Epochs
Learn rate drop factor	0.5
L2 Regularization	0.1
Initial learning rate	0.001
Validation frequency	1
Momentum	0.9
Squared gradient decay	0.99

The training (first 80,000 cycles) and validation (80,000 to 100,000 cycles) results in the model shown in Figure 6.4. To develop this neural network based model, consisting of more than 11,000 learnable parameters, a larger data set is needed. The ESM was run for 100,000 engine cycles; cycles 1 to 80,000 are devoted to training and cycles 80,001 to 100,000 are used for validation, as shown in Figure 6.4. The results show that the LSTM model can estimate intake manifold pressure, output torque, and NO_x emission with high accuracy for both training and validation data sets. The accuracy of the training data is 2.35%, 1.98%, and 1.07% for NO_x, T_{out}, and P_{man},

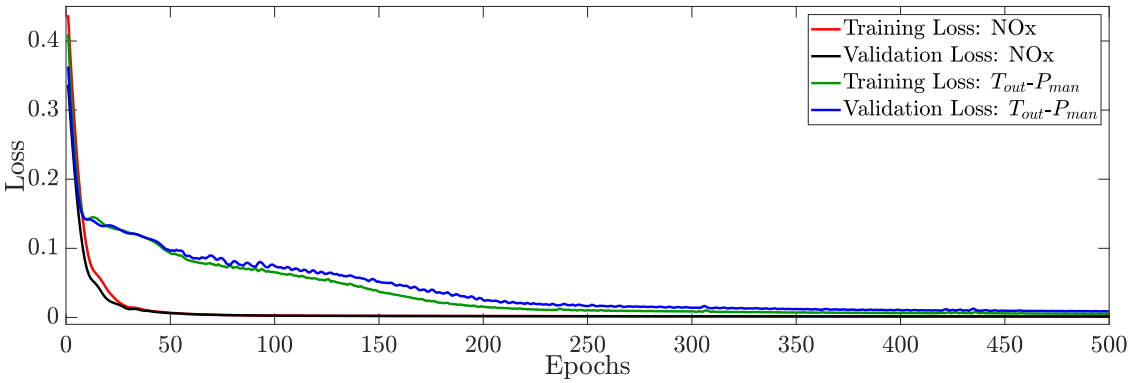


Figure 6.3: Loss vs. epochs for NO_x, torque and pressure model

respectively. For the validation data set, an accuracy of 2.86%, 2.27%, and 1.53% for NO_x, T_{out}, and P_{man} are found.

As this model will be used inside a Nonlinear Model Predictive Controller (NMPC) an accurate model is critical. To compare LSTM models, LSTM model is evaluated for test data (data never seen before for the models). In this manner, the LSTM model is evaluated using a test dataset that is new for the models. Figure 6.5 compares the ESM with the LSTM model; high accuracy is obtained on all of these outputs, shown by the curves lying almost on top of each other.

To summarize the accuracy of the model, the Normalized Root Means Square Error (NRMSE) between the ESM and LSTM model are calculated. The RMSE is normalized using the range (defined as the maximum value minus the minimum value) of the ESM logged data. The NRMSE for NO_x, P_{man}, and T_{out} are 4.04%, 2.21%, and 2.35%, respectively. As shown, this model is capable of predicting emission and output torque with an error of less than 5%. Comparing to the linear and LPV models developed in Chapter 5, the LSTM provides better accuracy, especially for the NO_x emission. For NO_x, it is 3% more accurate than the SVM-LPV, and for torque prediction is approximately 1% more accurate. However, for the manifold pressure, LPV provides 1.3% better accuracy compared to the LSTM. The LSTM is found to be generally more accurate than the SVM-LPV and the ARX linear model

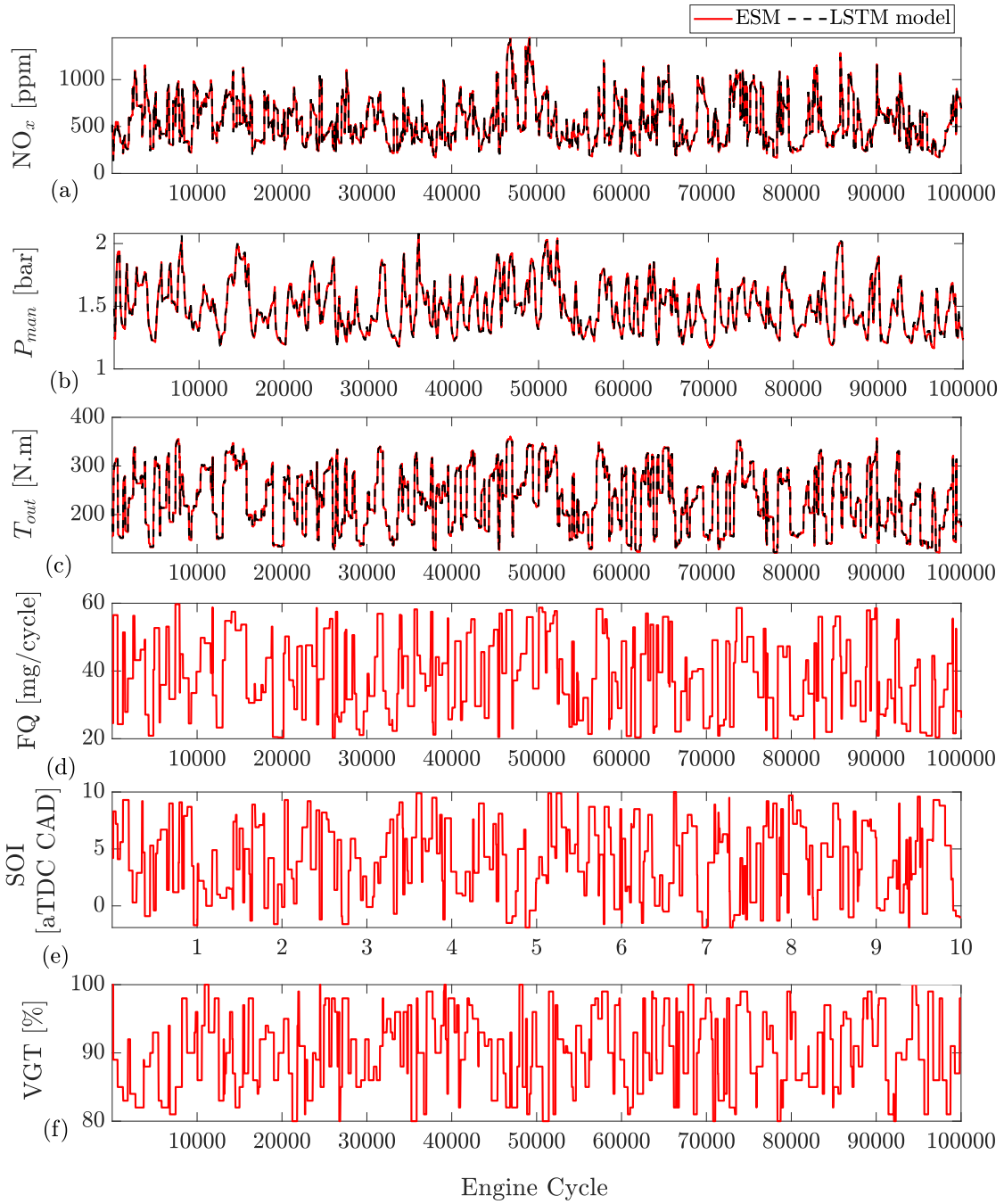


Figure 6.4: Training and validation results for the LSTM model vs. ESM: a) engine-out NO_x, b) intake manifold pressure P_{man}, c) engine-output torque T_{out}, d) FQ, e) SOI, f) VGT rate. Cycles 1 to 80,000 are devoted to training and cycles 80,001 to 100,000 used for validation

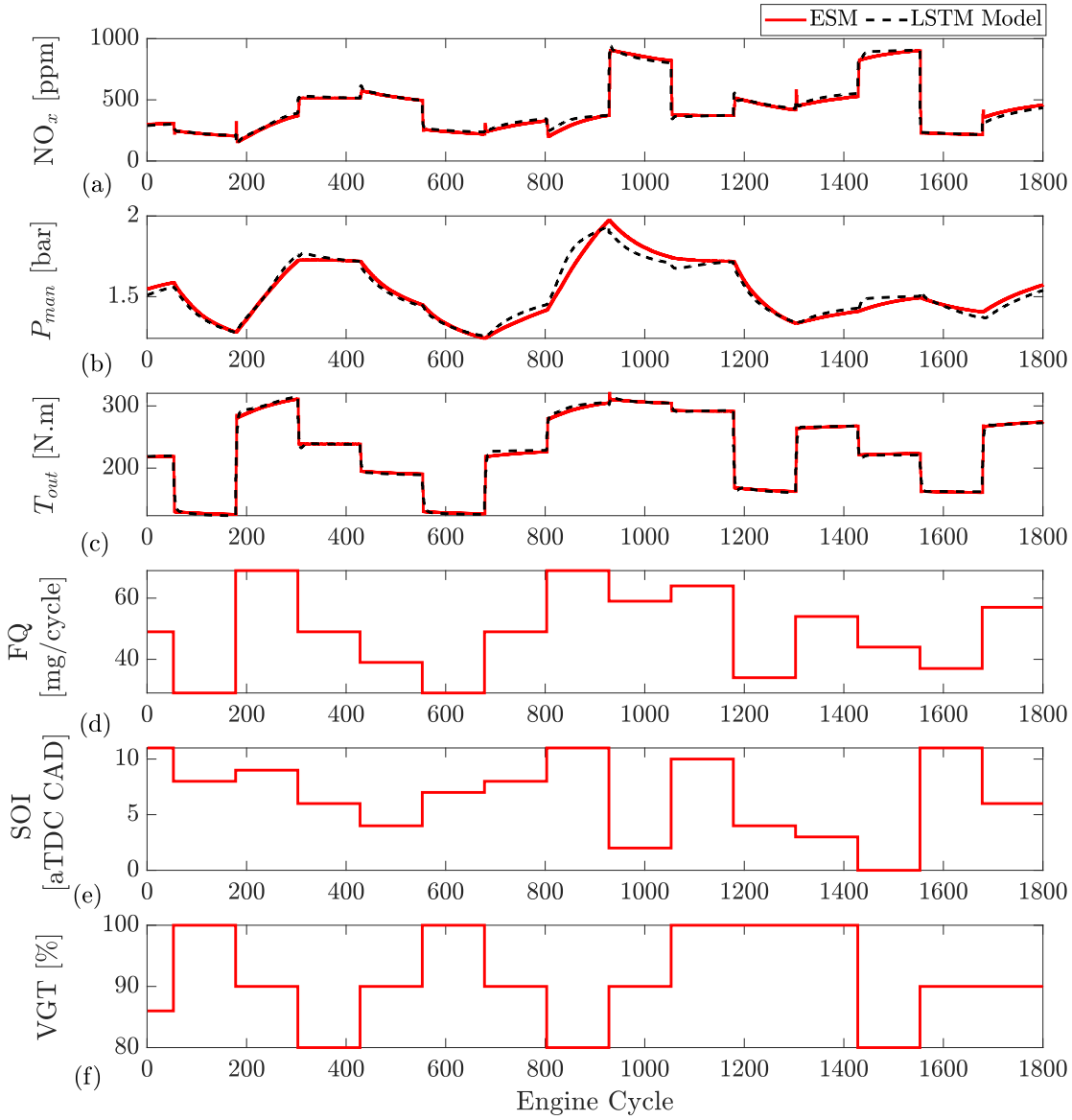


Figure 6.5: LSTM model comparison for engine-out emissions and performance: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine-output torque (T_{out}), d) FQ, e) SOI, f) VGT rate

developed in Chapter 5 for these outputs.

6.2 Nonlinear Model Predictive Controller Design

In this section, the control objective is the same as LPV-MPC controller (the same cost function Eq. 5.18 in Section 5.2 is used). But instead of an LPV model which

results in an LPV-MPC, a nonlinear function LSTM model is used which results in Nonlinear MPC (NMPC). Therefore in the optimal control problem (Eq. 5.21), $x_{k+1} = f(x(k), u(k))$ is a nonlinear model leading to nonlinear MPC. For the formulation of the LSTM-NMPC, the hidden and cell states of the two LSTM networks are augmented. Therefore, the prediction model of the LSTM-NMPC consists of the following: the state vector $x(k)$, the output vector $y(k)$, and the control vector $u(k)$ which are defined as

$$\begin{aligned} x &= [h_{\text{LSTM1}}, \quad h_{\text{LSTM2}}, \quad c_{\text{LSTM1}}, \quad c_{\text{LSTM2}}]^T, \\ y &= [T_{\text{out}}, \quad \text{NO}_x]^T, \\ u &= [\text{FQ}, \quad \text{SOI}, \quad \text{VGT}]. \end{aligned} \tag{6.6}$$

The added hidden and cell states (2 networks each have 26 cell states and 26 hidden states) lead to a total number of 104 states for the LSTM-MPC. A schematic of the LSTM-NMPC is shown in Figure 6.6. The recurrent network prediction and update functions, stated in Eq. 6.1 and Eq. 6.3, are defined by formulating the chain rule. The prediction horizon, N_p, has been chosen as $N_p = 5$ with a control horizon of one step the same as the LPV-MPC in Chapter 5.

The NMPC problem has 104 states, and all of these states must be estimated to update the NMPC block that is used in MATLAB/SIMULINK©. Therefore, the same dynamic model is used as an estimator for the hidden and cell states of the LSTM model to provide the 104 states of the NMPC problem. The lower and upper values of the state and control constraints are chosen to have the same value as the LPV-MPC and are listed in Table 5.2.

The weights of the LSTM-NMPC are tuned manually by giving the highest weight to the output torque and giving equal weight to the emission and fuel consumption reduction. The states, outputs, and control inputs are normalized using z-score normalization, which is defined as

$$z_{\text{normal}} = \frac{z - \mu}{\sigma} \tag{6.7}$$

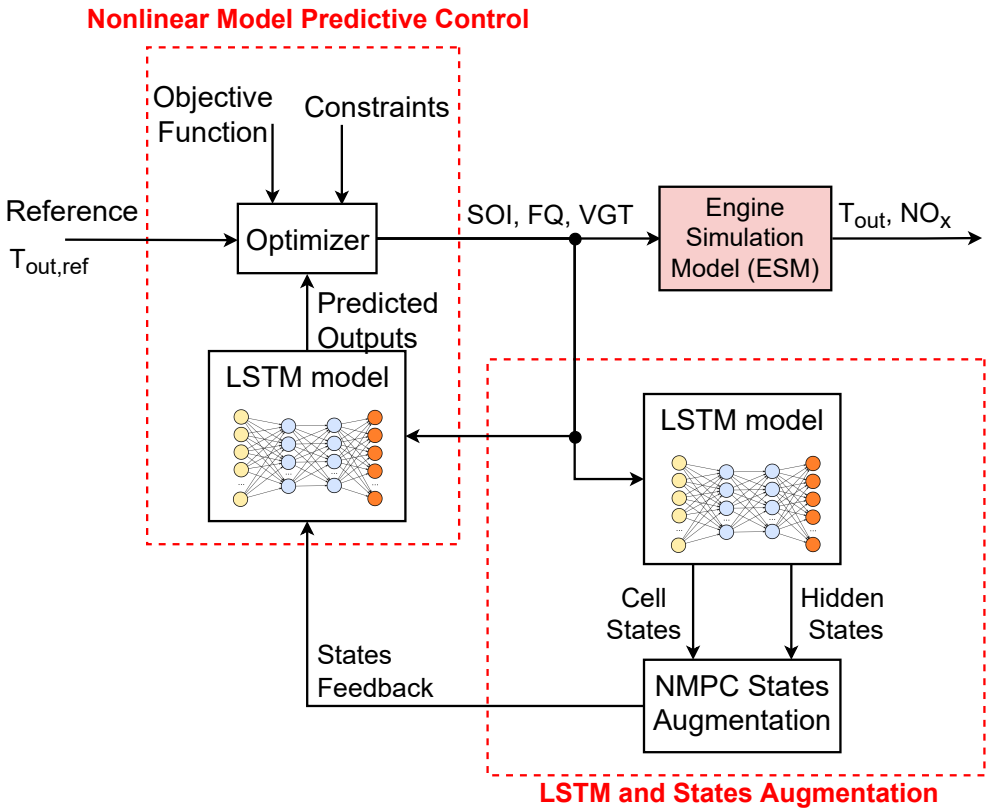


Figure 6.6: Block diagram of LSTM-NMPC structure

where μ is the mean and σ is the standard deviation of z. This can be used for both input and outputs.

In this chapter the NMPC is implemented in simulation on the ESM engine model. The MATLAB Toolbox© utilizing the `fmincon`© solver (SQP with QP solver of interior point method) is used for the NMPC simulation. Alternatives to solving the NMPC include state-of-the-art commercial solver `FORCES PRO`© by EMBOTECH [200, 201] (SQP solver) and open-source package `acados` [202, 203] with the QP solver `HPIPM` (High-Performance Interior-Point Method) [204]. In NMPC, the OCP structured Nonlinear Programming (NLP) is reformulated into a Sequential Quadratic Programming (SQP) problem by means of iterative quadratic approximations at the shooting nodes [205]. This results in sequential quadratic subproblems that are solved and contribute to the holistic solution of the NLP by means of reformulations of the initial problem. The number of SQP iterations are called SQP steps and have a

strong influence on the computational efficiency as well as the prediction quality of the NMPC. In Section 6.4, the computational timing of `fmincon`©, `FORCES PRO`©, and `acados` with `HPIPM` are compared. Computation cost is important and ideally needs to be reduced. Imitation NMPC described in the next section, could replace NMPC online optimization and provide significantly less computational effort.

6.3 NMPC Imitative Controller

The imitation of NMPC, using deep learning methods, is called imitative NMPC and is used to avoid the high computational time of the online optimization of the NMPC [158]. In Section 5.3, a deep neural network is used to imitate the LPV-MPC. Here, the same architecture with different hyperparameters is used to clone or imitate NMPC behavior.

The concept of imitative NMPC is shown schematically in Figure 6.7 which depicts the three main steps. In the first step, the previously designed LSTM-NMPC is implemented on ESM. Second, the NMPC input and output are recorded, and a deep neural network, including an LSTM layer is used to fit the controller data to mimic the behavior of the NMPC. Finally, the online NMPC is replaced with an imitative controller which greatly reduces the computational time of the NMPC. The result is that, instead of solving NMPC optimization online, the identified function –here a deep network– is deployed with a much lower computation cost.

In order to clone the behavior of the NMPC, a deep network structure using LSTM is proposed. The imitative NMPC structure for imitative LSTM-NMPC is schematically shown in Figure 6.8. The engine output torque T_{out}, error in output torque ($e_{T_{\text{out}}}$), engine-out NO_x, intake manifold pressure P_{man}, and engine speed n_{rpm} are the inputs of the imitative NMPC. The goal here is to generate control action of fuel quantity (FQ), start of injection (SOI), and VGT by mimicking the previously designed NMPC controllers.

The forward propagation of this imitative network, used to replace the online

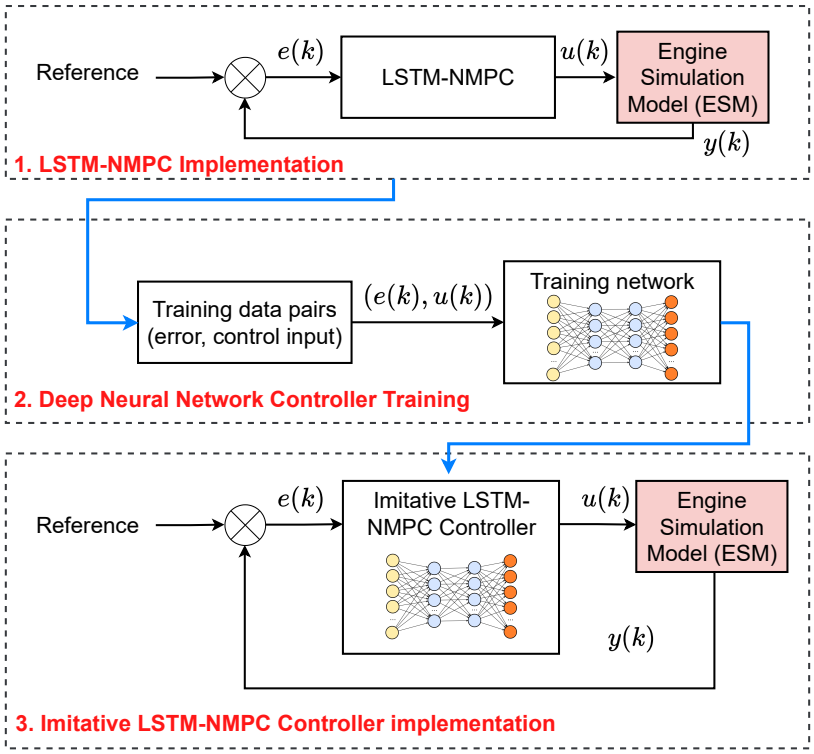


Figure 6.7: Concept of Imitative NMPC– 1) implementation of original LSTM-NMPC in ESM co-simulation, 2) training deep neural network based on LSTM-NMPC collected data, 3) replacing of trained deep network (imitative controller) with LSTM-NMPC in ESM co-simulation (NMPC: Nonlinear Model Predictive Control; LSTM: Long-Short-Term Memory; ESM: Engine Simulation Model).

optimization of MPC is

$$z_{FC1}(k) = \text{ReLU}\left(W_{FC1}^T u(k) + b_{FC1}\right) \tag{6.8a}$$

$$z_{LSTM}(k) = h_{LSTM}(k) = \underbrace{o_{LSTM}(k) \odot \tanh\left(c_{LSTM}(k)\right)}_{\text{based on Eq. 5.23}} \tag{6.8b}$$

$$z_{FC2}(k) = \text{ReLU}\left(W_{FC2}^T z_{LSTM}(k) + b_{FC2}\right) \tag{6.8c}$$

$$\underbrace{z_{FC3}(k)}_{\hat{u}(k)=[\hat{u}(k) \quad \hat{SOI}(k) \quad \hat{VGT}]^T} = W_{FC3}^T z_{FC2}(k) + b_{FC3} \tag{6.8d}$$

The control output of the imitative LSTM-NMPC network is

$$\hat{u}(k) = \begin{bmatrix} \hat{FQ}(k) \\ \hat{SOI}(k) \\ \hat{VGT} \end{bmatrix} \tag{6.9}$$

The structure and training of the imitative controller is summarized in Table 6.2.

As with all system identification, the performance of the imitative controller depends

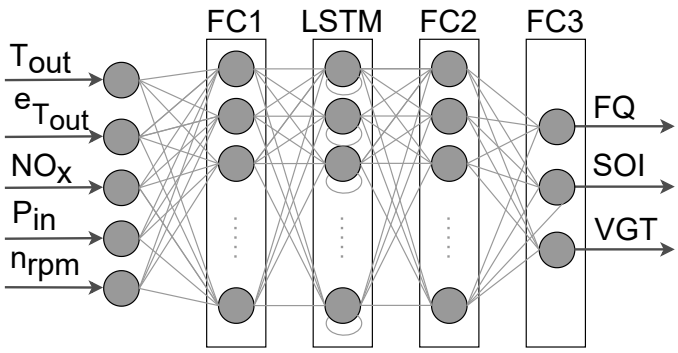


Figure 6.8: Structure of proposed network for imitation of NMPC

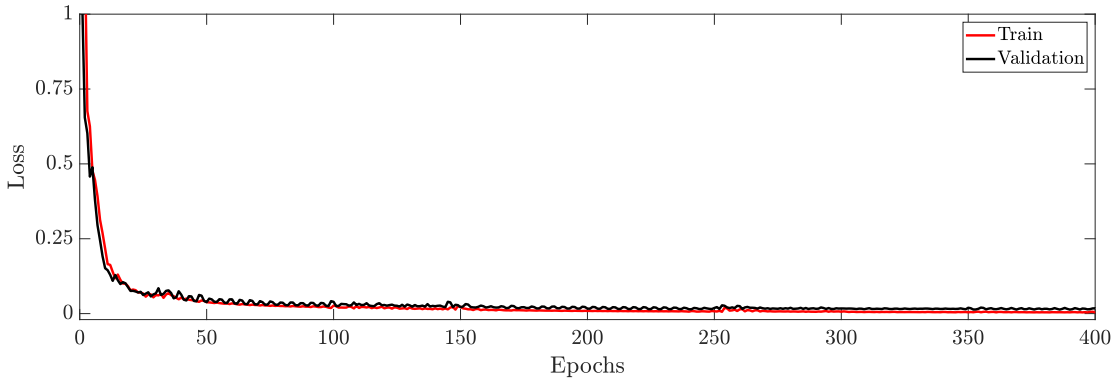


Figure 6.9: Imitative LSTM-NMPC loss function vs. epochs

on collected data. Widely varying operating conditions were simulated and used as inputs to the network. To do this, the NMPC controller was evaluated for randomly changing engine speed from 1200 rpm to 1800 rpm and a requested load (T_{out} reference) from 120 N.m to 320 N.m. This was done to make the imitative controller robust to a range of operating conditions. The loss function versus iteration is shown in Figure 6.9. This loss function indicates that the training process has been completed and the overfitting-underfitting has been avoided as the validation loss converges to the training loss.

To train the imitative controller, the NMPC is evaluated for 2000 seconds of simulation, in which 1600 seconds are reserved for training data and 400 seconds are reserved for validation. The RMSE of the training and validation set are listed in Table. 6.3 which shows that the DNN network can clone the NMPC behavior with an

Table 6.2: Properties of imitative controller based on LSTM-NMPC

Name	**LSTM-NMPC**
FC(1, 2) size	32
FC3 size	3
LSTM size	32
Optimizer	Adam
Maximum epochs	400
Mini batch size	512
Learn rate drop period	150 epochs
Learn rate drop factor	0.5
L2 Regularization	1
Initial learning rate	0.02
Validation frequency	1
Momentum	0.9
Squared gradient decay	0.99

average accuracy of 3.9% on training and 8.3% on validation compared to the previous LSTM-NMPC. These imitative controllers are tested for newly generated references, and their performance against NMPC online optimization will be presented in the next section.

Table 6.3: Imitative LSTM-NMPC controller train and validation error compared to LSTM-NMPC online optimization

	FQ	**SOI**	**VGT**
Train	2.12%	4.55%	4.90%
Validation	2.47%	9.98%	12.46%

6.4 Results and Discussion

The objective of the controller is to track a target engine load while minimizing both NO_x and the fuel used in the diesel engine. The results of the NMPC controller with an LSTM-based data driven model and imitative LSTM-NMPC controller are compared to the BM using the ESM. The BM model is the calibrated ECU tables based on the Cummins production ECU that is then embedded in GT-power© (further details of BM and ESM are available in Chapter 2). The controller inputs and outputs for all three controllers tested are shown in Figure 6.10.

The controllers are subject to constraints on all of the inputs (i.e. NO_x, FQ, SOI and VGT), which is a main advantage of MPC. These limits are listed in Table 5.2. Small constraints violation for NO_x at engine cycle 600 are seen for LSTM-NMPC. This is attributed to the constraint softening that is implemented in the NMPC. However, overall, both the LSTM-NMPC and the imitative LSTM-NMPC controllers are able to keep the NO_x levels below the specified constraint. The BM is significantly worse for NO_x than the nonlinear MPC model which is likely due to it operating away from the optimal calibration point in the feedforward tables for this stationary engine.

The upper limit of the SOI constraint is often active. Here the controllers would like to implement later injection timing than are currently allowed. These late injection timings reduce the peak combustion temperature leading to lower NO_x levels but also result in reduced combustion efficiency and increased fuel consumption.

The torque (T_{out}) tracking of both LSTM-NMPC and BM for a step input in torque are shown in Figure 6.11. Acceptable performance is achieved. An interesting trend is that even though the BM uses the most fuel, it has an undesirable offset from the steady-state torque reference.

To check the controller's robustness to engine speed, it is changed from 1500 rpm (the level at which the MPC models were designed and validated) to 1200 rpm. The

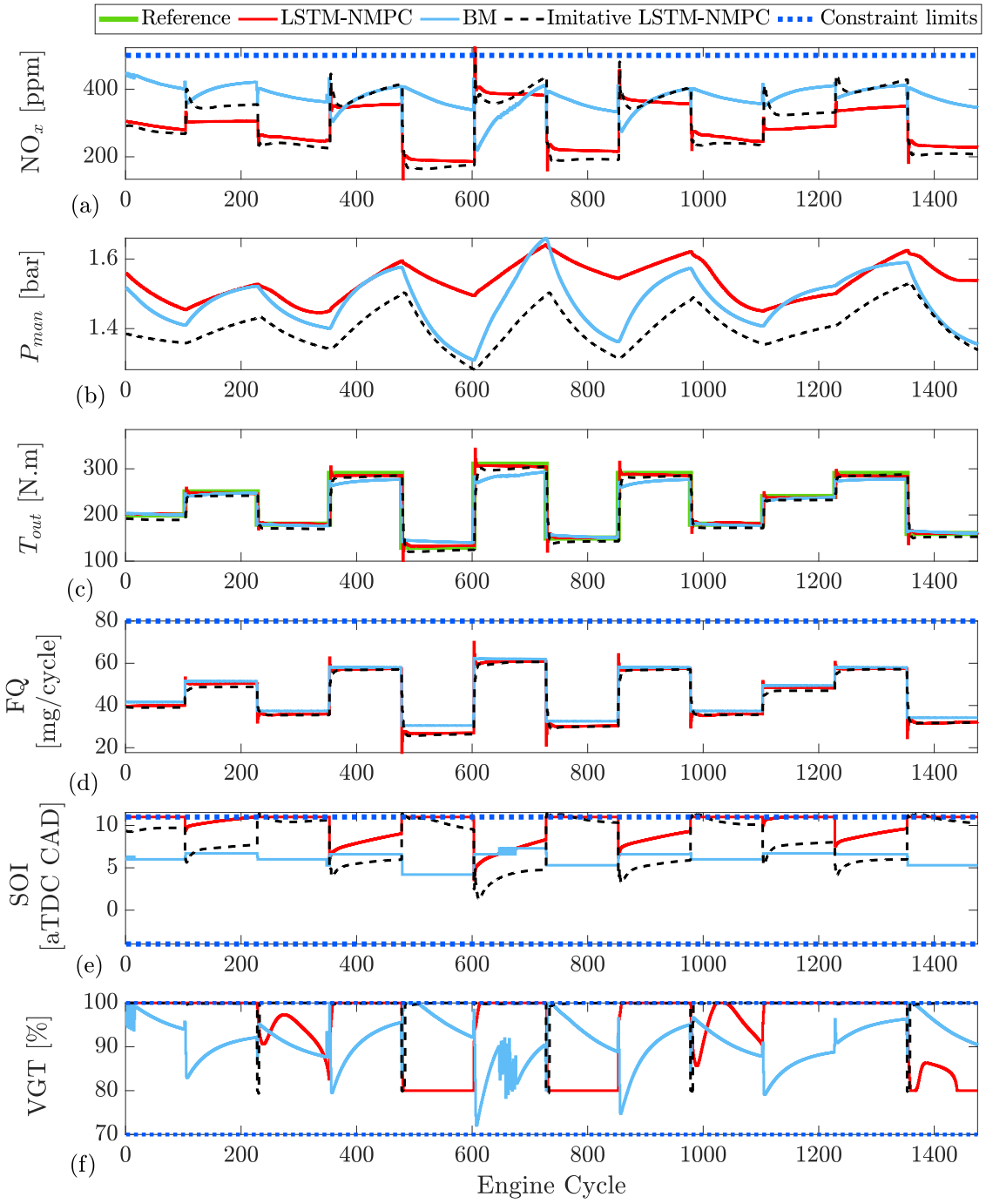


Figure 6.10: Controller comparison at $n_{\text{rpm}} = 1500$: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate

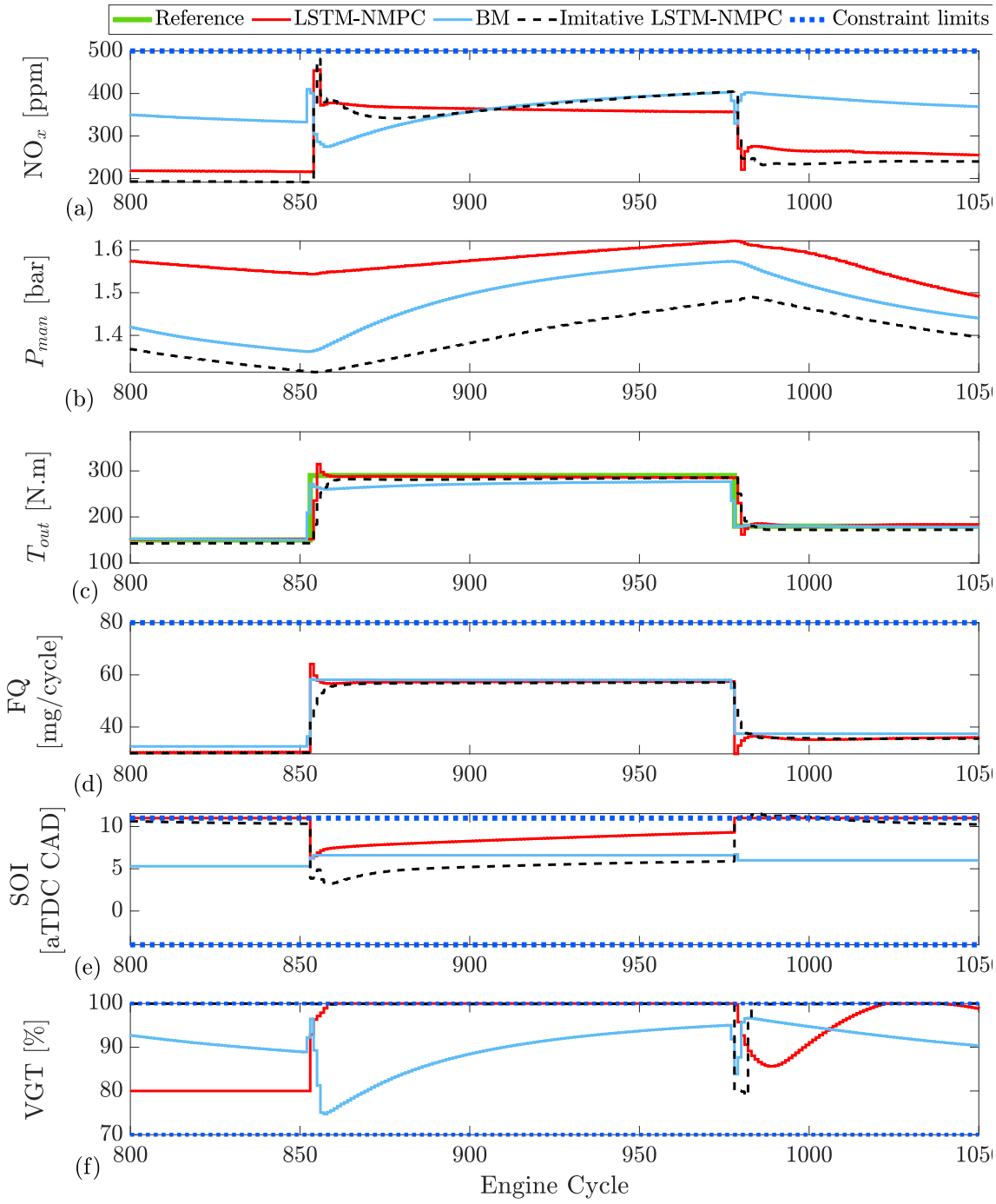


Figure 6.11: Controller comparison at $n_{\text{rpm}} = 1500$ zoomed from 800 to 1050 cycles: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate

Table 6.4: Turnaround time comparison between Matlab `fmincon`©, EMBOTECH `FORCES PRO`©, and `acados` solvers– PIL: Processor in the loop (performance of controller will be discussed in Chapter 7)

Solver	**Average Turnaround time [ms]**	**Real-time verification in PIL setup**
Matlab `fmincon`©	786.02	x
EMBOTECH `FORCES PRO`©	786.02 > t >> 12.20	x
`acados`	12.20	✓

NMPC performance at 1200 rpm can be seen in Figure 6.12. There, the BM has high NO_x values which are attributed to a significant advance in injection timing.

The performance of the controllers at both engine speeds are summarized by listing the cumulative and average NO_x emissions, load error, FQ and execution times are shown in Table 6.5. For the design speed of 1500 rpm, both the LSTM-NMPC and the imitative NMPC outperform the BM in terms of NO_x emissions with advantages in fuel consumption reduction.

To investigate the NMPC execution time, `acados` (SQP with QP solver `HPIPM`), `FORCES PRO` with SQP algorithm, and `fmincon` with SQP algorithm (QP solver interior point) are each evaluated and their computational time are compared and summarized in Table 6.4. `acados` with `HPIPM` provides the fastest solve time among the solvers tested. In the `acados` implementation, a maximum QP iteration of 50 and maximum NLP iteration / SQP steps of 5 are used, which showed an average runtime of 12.20 ms and a maximum runtime of 31.56 ms at 1500 rpm. This value is much faster than the average runtime of `fmincon` that required 786.02 ms. `FORCES PRO` was also tested and showed an improvement in runtime over the `fmincon` but was slower than the `acados` implementation. Due to the academic license agreement, the exact computational timing of `FORCES PRO` cannot be disclosed.

This difference in runtime can be attributed to the fully condensed problem formulation used in the `acados` implementation when compared with the sparse formulation

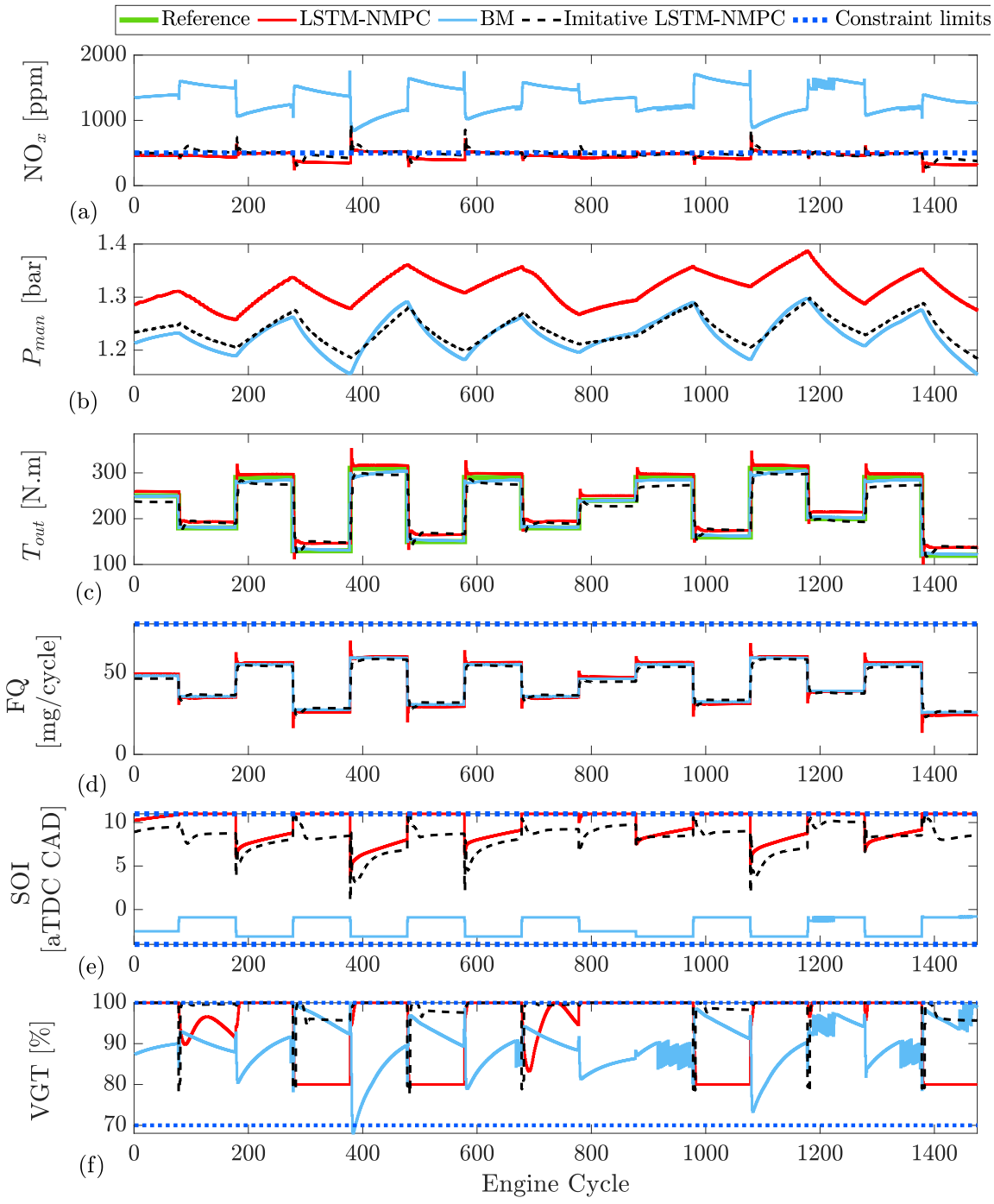


Figure 6.12: Controller comparison at $n_{\text{rpm}} = 1200$: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate

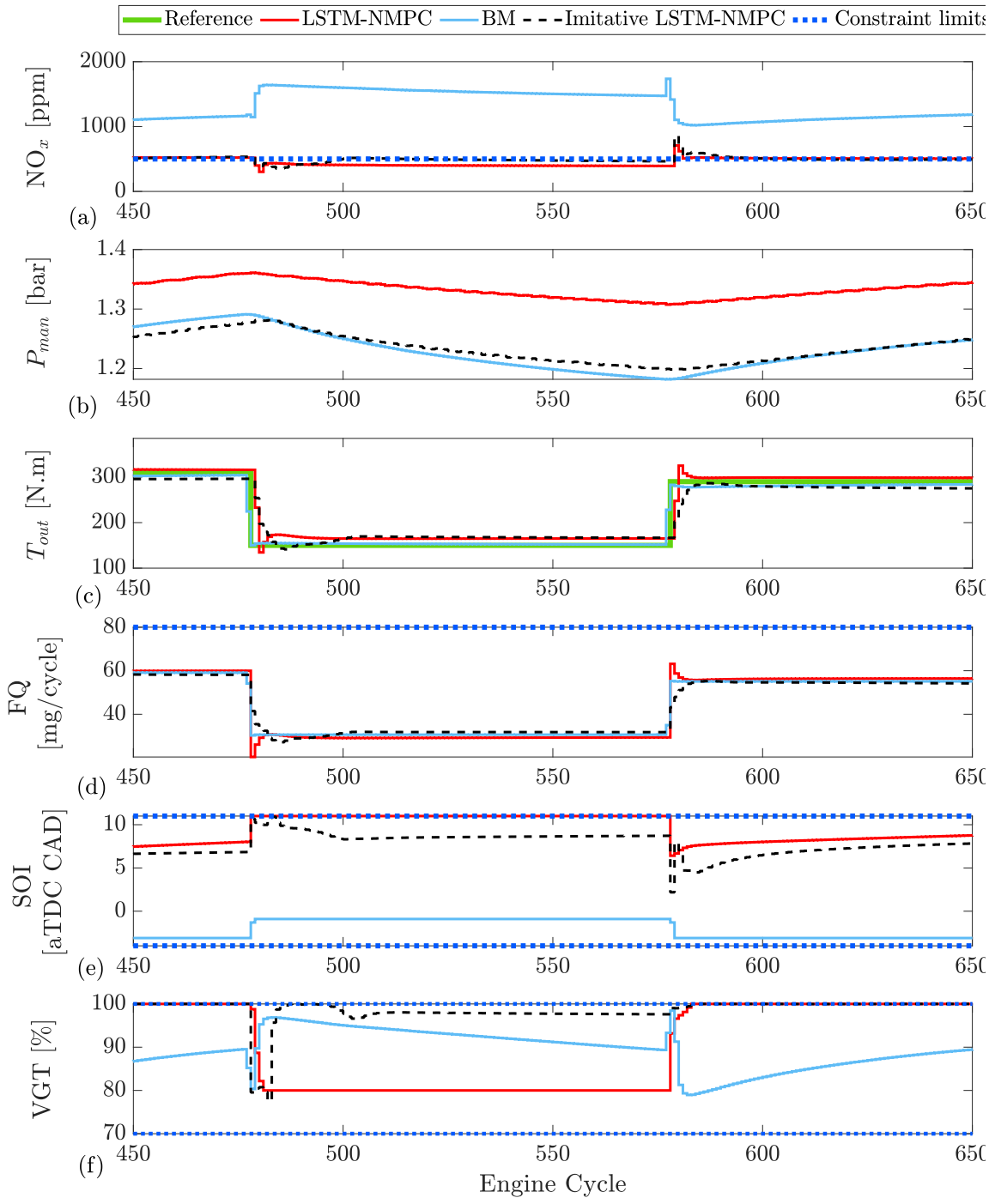


Figure 6.13: Controller comparison at $n_{\text{rpm}} = 1200$ zoomed from 450 to 650 cycles: a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate

of `fmincon` and `FORCES PRO`. The presented OCP consists of a large state vector with 104 states, a relatively small prediction horizon with 5 steps and a relatively small control vector with 3 control outputs. This allows the condensed problem formulation used in `acados` to take full advantage of the condensation benefits [206, 207]. Another reason for the difference in runtime is the use of the underlying algorithmic differentiation framework `CasADi` MX (Matrix expression) symbolic variables in `acados` instead of SX (scalar symbolic) variables in `FORCES PRO`. The general matrix expression type MX tends to be more economical when working with larger matrices [208].

Although the `acados` computational time of the online NPMC is orders of magnitude slower than imitation NMPC, the ML-based imitation controllers rely on MPC results, and thus an MPC must first be designed and then used to generate input-output data for imitative MPC training.

Table 6.5: Proposed MPC and Imitative MPC results compared to the benchmark for engine speeds of 1500 and 1200 rpm

1500 rpm						
	Cumulative NO_x [ppm]	Average NO_x [ppm]	Load error [%]	Cumulative FQ [g]	Average FQ [mg]	Execution time [ms]*
Benchmark	556100.0	376.8	3.95	67.9	46.0	-
LSTM-NMPC	428420.0	290.2	1.90	65.6	44.4	12.20**
Imitative NMPC	438850.0	297.3	3.74	64.6	43.8	0.04
1200 rpm						
	Cumulative NO_x [ppm]	Average NO_x [ppm]	Load error [%]	Cumulative FQ [g]	Average FQ [mg]	Execution time [s]*
Benchmark	1961900.0	1329.2	2.18	64.8	43.9	-
LSTM-NMPC	671650.0	455.0	6.95	64.9	43.9	11.50**
Imitative NMPC	718380.0	486.7	7.61	64.0	43.4	0.03

* per engine cycle of simulation

** `acados` execution time

The performance of the controllers compared to the BM can be seen in Table 6.6. A significant NO_x emissions reduction for the controllers over the BM is shown. In

addition to these improvements in emissions, the developed controllers use the same amount or less fuel than the baseline model. This demonstrates the advantage of the optimized controllers over the BM. The developed models do experience a slight increased load error compared to the BM at 1500 rpm. However, the 2% worse load tracking worth the significant emissions and fuel consumption benefits.

Table 6.6: Percentage of improvement for proposed MPC and Imitative MPC with respect to the benchmark for engine speeds of 1500 and 1200 rpm

1500 rpm			
	NO_x [%]	FQ [%]	load error [%]
LSTM-NMPC	-22.98	-3.48	+2.05
Imitative NMPC	-21.10	-4.78	+0.21
1200 rpm			
	NO_x [%]	FQ [%]	load error [%]
LSTM-NMPC	-65.77	+0.15	-4.77
Imitative NMPC	-63.38	-1.23	-5.43

Overall, the imitative NMPC controller provides similar improvements to the NMPC controller over the BM but the computational time is orders of magnitude smaller. This makes computational requirements for a real-time implementation for a production implementation much lower than the MPC online NMPC optimization.

In comparison with the LPV-MPC and linear MPC developed in Chapter 5, the LSTM-NMPC reduces more $NO_x$and FQ at 1500 rpm. At a lower speed of1200 rpm, the LSTM-NMPC could track loads better than the LPV-MPC with 3% better NO_x reduction (the LPV-MPC reduced NO_x by 67% while the LSTM-NMPc reduced it by 70%). For imitation MPC, both imitative LSTM-NMPC and LPV-MPC (Chapter 5) are comparable.

6.5 Summary of chapter

The integration of deep learning and MPC for both modeling and controller implementation is discussed in this chapter. First, using a DNN network, a LSTM network

is designed. An NMPC is then designed based on this network and by augmenting hidden and cell states. The accuracy of this model for test data has better than the LPV and linear model developed in Chapter 5. Higher accuracy is expected as the LSTM is capable of a more generalizable prediction since it uses hidden and cell states. In addition to using ML for modeling ML can also be implemented based on a implemented NMPC. The input-output data is then collected from the NMPC and used to train a deep neural network. Replacing the online NMPC with an imitative controller reduces the computational time of the NMPC and is compared with a baseline Cummins calibrated ECU model in simulation using the ESM.

Minimizing NO_x emissions and reducing the injected fuel amount while maintaining the same load is goal of the control. Furthermore, the controllers are constrained to meet NO_x limits and all inputs to ensure system safety. To evaluate the robustness of the controllers, the engine speed is changed from 1500 rpm where the NMPC model was validated to 1200 rpm. All of the controllers produce significant NO_x reduction compared to the BM feedforward controller especially at lower engine speeds. The NO_x level for 1500 and 1200 rpm for the NMPC are reduced by 23.0% and 65.8% when compared to the BM. The imitative controller successfully clones the NMPC behavior with a NO_x reduction of 21.1% at 1500 rpm and a 63.4% reduction at 1200 rpm when compared to the BM. The imitative controller performs similarly to the online MPC by learning from the MPC experiment but requires much lower computational time. The computation time for the imitative controller is a factor of 100 lower than the online optimized MPC.

A comparison of LSTM-NMPC and LPV-MPC (Chapter 5) shows that the LSTM-NMPC has better load tracking and emission reduction performance. Fuel consumption reduction is consistent at different speeds, especially at 1500 rpm (the speed at which this engine usually runs). Based on this comparison, the next chapter will describe the real-time implementation of the LSTM-NMPC controller on the engine with some modifications for the experimental setup, including the addition of a par-

ticle matter sensor feedback.

Chapter 7

Integration of Deep Learning and Nonlinear Model Predictive Control: Experimental Implementation [1]

In this chapter, a Long-Short Term Memory-based Nonlinear Model Predictive Control (LSTM-NMPC) is experimentally implemented in real-time to reduce the engine-out emissions and fuel consumption of a 4.5 liter 4-cylinder Cummins CI engine while constraints are simultaneously implemented on engine inputs and outputs. Here, the LSTM-NMPC developed in Chapter 6 is expanded and implemented on a real-time system using `acados` embedded programming. To make the previously developed controller more general for CI implementation, Particle Matter (PM) soot emission reduction is also augmented to optimize the NO_x and PM trade-off. In addition, multi-pulse injection timing and duration along with fuel pressure control are added and compared to LSTM-NMPC developed in Chapter 6 to have more degrees of freedom to optimally control the CI engine-out emissions.

In this chapter, the emissions and performance of the engine are modeled using a deep neural network with seven hidden layers and 24,148 learnable parameters created by stacking FC layers with an LSTM layer in a manner similar to Chapter 6. This model is then utilized to implement NMPC experimentally. In order to

[1] This chapter is based on [8]

implement this LSTM-NMPC, an open-source software package –`acados` with the quadratic programming solver `HPIPM` (High-Performance Interior-Point Method)– is employed. This `acados` embedded programming scheme is used for LSTM-NMPC real-time implementation. For real-time controller prototyping, a dSPACE MicroAutoBox II (MABX II) rapid prototyping system is used. A Field Programmable Gate Array (FPGA) is also employed to calculate in-cylinder pressure based combustion metrics online at every 0.1° crank angle [209]. The FPGA calculates Maximum Pressure Rise Rate (MPRR) and Indicated Mean Effective Pressure (IMEP).

7.1 Deep Neural Network Modeling

In Chapters 5 and 6, the application of a deep neural network using FC and LSTM layers has been demonstrated and its benefits over conventional RNN and the feed-forward network has been discussed. Here, this idea has been expanded to model the engine and the emissions using real-time collected data. The engine and emissions are modeled using a deep neural network with seven hidden layers, including 6 FC layers and one LSTM layer, as shown in Figure 7.1. The inputs to this model are the SOI for the main injection, DOI for both the main and pilot injections, duration between the end of the pilot injection, start of the main injection pre-2-main time (t_{P2M}), and the fuel rail pressure $\mathrm{P_{fuel}}$. The P2M time, t_{P2M}, is used instead of SOI_{pilot} to allow for hardware constraints which limit the minimal time between injections to be implemented in the controller. This is necessary to prevent unintended overlapping injections where the injector has not fully closed. Figure 7.2 shows the relationship between the SOI and DOI of both injections as well as the t_{P2M}. The model outputs are Nitrogen Oxide ($\mathrm{NO_x}$), Particle Matter (PM), Maximum Pressure Rise Rate (MPRR), and Indicated Mean Effective Pressure (IMEP). The FC layers are added before and after the LSTM layer to boost the network's capacity for estimating the engine's nonlinearity without increasing the number of hidden and cell states. In Chapter 6 it was shown that adding 26 hidden units adds 26 cell and 26 hidden states

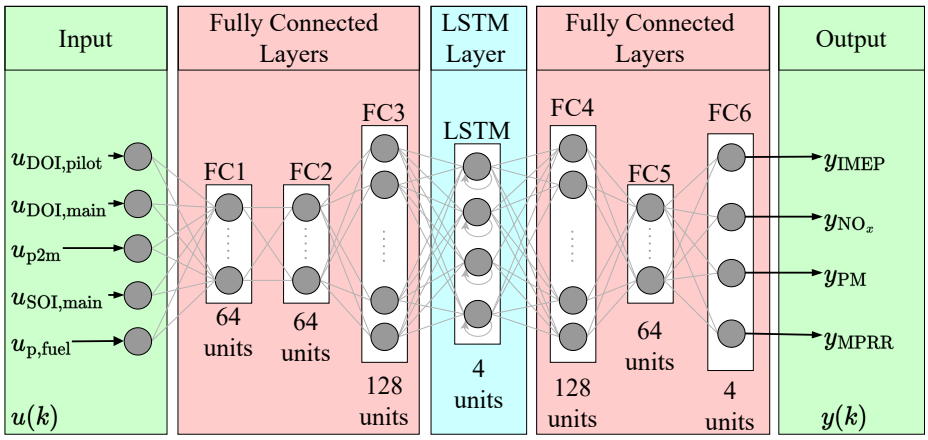


Figure 7.1: Structure of proposed deep neural network model for engine performance and emission modeling. LSTM: Long-short term memory, SOI: start of injection, DOI: duration of injection, P_{fuel}: fuel rail pressure, IMEP: indicated mean effective pressure, MPRR: maximum pressure rise rate, PM: particle matter, t_{P2M}: duration between end of pilot injection and start of main injection

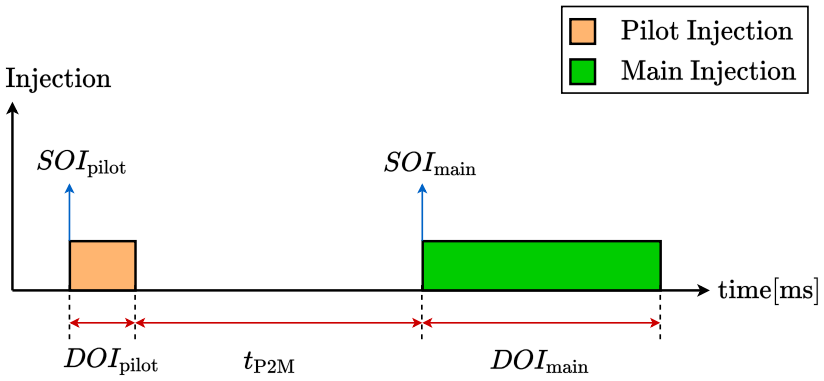


Figure 7.2: Diesel engine multiple injection. SOI: start of injection, DOI: duration of injection, t_{P2M}: duration between end of pilot injection and start of main injection

to a number of NMPC states. To make computational turnaround time as short as possible, keeping the number of hidden units for LSTM as small as possible is crucial. Instead, to capture the nonlinearity, more hidden units are added to the FC which resulted in a high number of learnable parameters to retain the complexity of the model but with fewer states.

To implement this network inside an NMPC, the previous description in Section 6.1 is followed and a forward propagation model is used. Then, by augmenting hidden states and cell states to the actual states of the system, this model can be used inside

the NMPC. The LSTM and FC computations are presented in Eq. 5.23 and Eq. 5.26. A schematic to compute the network is shown in Fig, 7.3. This computation includes employing an LSTM computation (Eq. 5.23) and an FC computation (Eq. 5.26), so the model is calculated as

$$
\begin{aligned}
z_{FC1}(k) &= \text{ReLU}\left(W_{FC1}^T u(k) + b_{FC1}\right) \\
z_{FC2}(k) &= \text{ReLU}\left(W_{FC2}^T z_{FC1}(k) + b_{FC2}\right) \\
z_{FC3}(k) &= \text{ReLU}\left(W_{FC3}^T z_{FC2}(k) + b_{FC3}\right) \\
i(k) &= \sigma\left(W_{ui}^T z_{FC3}(k) + W_{hi}^T h(k-1) + b_i\right) \\
f(k) &= \sigma\left(W_{uf}^T z_{FC3}(k) + W_{hf}^T h(k-1) + b_f\right) \\
g(k) &= \tanh\left(W_{ug}^T z_{FC3}(k) + W_{hg}^T h(k-1) + b_g\right) \\
o(k) &= \sigma\left(W_{uo}^T z_{FC3}(k) + W_{ho}^T h(k-1) + b_o\right) \\
c(k) &= f(k) \odot c(k-1) + i(k) \odot g(k) \\
h(k) &= o(k) \odot \tanh\left(c(k)\right) \\
z_{FC4}(k) &= \text{ReLU}\left(W_{FC4}^T h(k) + b_{FC4}\right) \\
z_{FC5}(k) &= \text{ReLU}\left(W_{FC5}^T z_{FC4}(k) + b_{FC5}\right) \\
\underbrace{z_{FC6}(k)}_{\hat{Y}(k)} &= W_{FC6}^T z_{FC5}(k) + b_{FC6}
\end{aligned}
\tag{7.1}
$$

where W_{FCi} and b_{FCi} are the weights and biases of the fully connected layer where $i \in 1, 2, 3, 4, 5, 6$ and $W_{u(f,g,i,o)}$ are the weight matrices of the input vector $u(k)$ and $W_{h(f,g,i,o)}$ are the weight matrices to the previous short-term states $h(k)$. In Eq. 7.1, $\tanh(z)$ in hyperbolic tangent, $\sigma(z)$ is sigmoid, and ReLU is the Rectified Linear Unit (ReLU) activation function. The functions $\tanh(z)$ and $\sigma(z)$ are used in the LSTM gates as described in LSTM computation in Eq. 5.23 and ReLU is a common choice for the FC layer in a DNN structure. These activation functions are defined in Eq 5.24, 5.25, and 5.27.

The augmented NMPC model (Eq. 7.1) states that include hidden states and cell

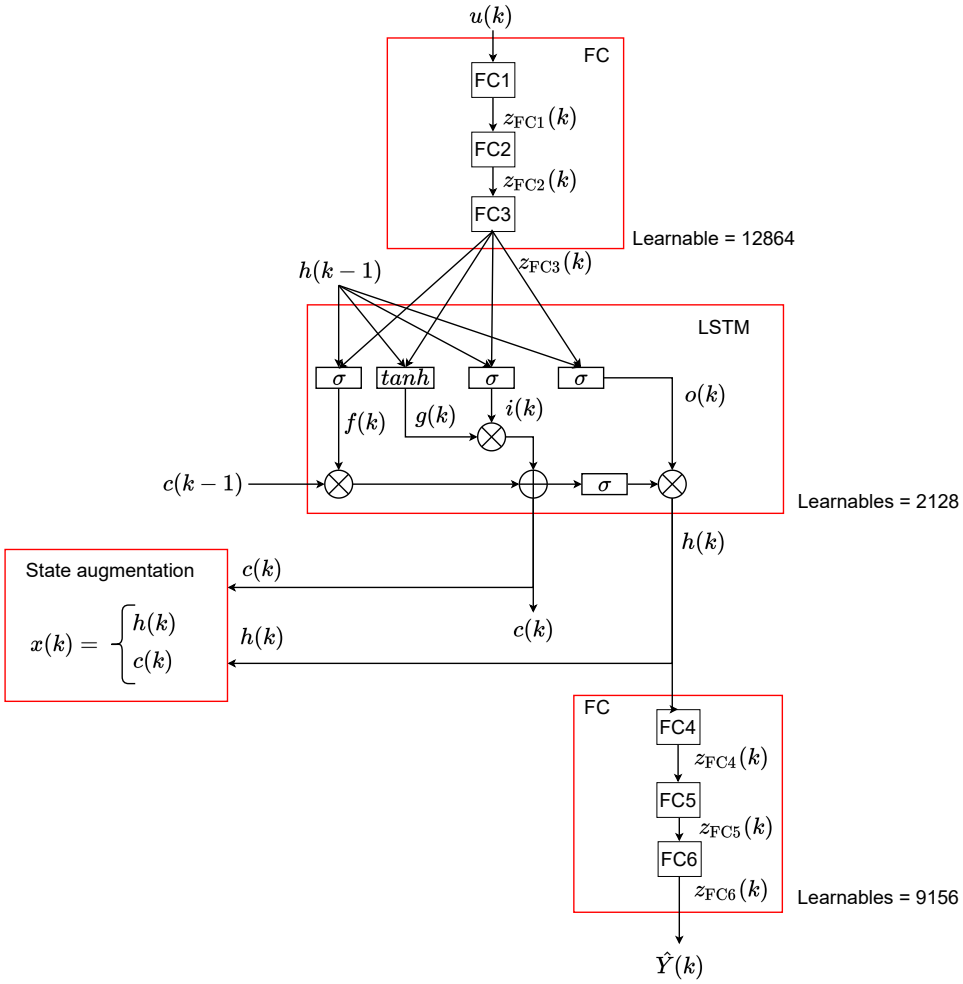


Figure 7.3: Computational graph of proposed deep network. FC: Fully Connected, LSTM: Long-Short Term Memory

states, are calculated based on the computational graph shown in Figure 7.3.

$$x(k) = \begin{bmatrix} h_{LSTM}(k) \in \mathbb{R}^4 \\ c_{LSTM}(k) \in \mathbb{R}^4 \end{bmatrix} \in \mathbb{R}^8 \tag{7.2}$$

By adding 4 hidden states and 4 cells, $x(k)$ has a total of 8 states. In the `acados` implementation, in order to add derivatives of inputs, 5 system inputs are also added to the states to make the augmented states dimension 13 states in total (more details are provided in Section 7.2). For training with experimental data, the cost function of this network is (see also Eq. 5.30, duplicated here)

$$J(W,b) = \frac{1}{m}\sum_{k=1}^{m}\mathcal{L}\left(\hat{Y}(k), Y(k)\right) + \frac{\lambda}{2m}\sum_{l=1}^{L}||W^{[l]}||_2^2 \tag{7.3}$$

where $\mathcal{L}\left(\hat{Y}(k), Y(k)\right)$ is the loss function. In this work, Mean Squared Error (MSE) loss function is used and is (see also Eq. 5.32, duplicated here)

$$\mathcal{L}\left(\hat{Y}(k), Y(k)\right) = \frac{1}{m}\sum_{k=1}^{m}(\hat{Y}(k) - Y(k))^2 \tag{7.4}$$

In Eq. 7.3, λ is the regularization coefficients, m is size of training data, and $||W^{[l]}||_2^2$ is the Euclidean norm which is (see also Eq. 5.31, duplicated here)

$$||W^{[l]}||_2^2 = \sum_{i=1}^{n^{[l]}}\sum_{j=1}^{n^{[l-1]}}(w_{ij}^{[l]})^2 \tag{7.5}$$

Table 7.1 summarizes the training data and design parameters for the proposed network. To train this model, the Adam algorithm was used in the MATLAB Deep Learning Toolbox©. The loss function vs iteration for the proposed deep network is given in Figure 7.4, in which the loss functions converge to a minimal value. Additionally, the validation loss function converges to the training loss function, suggesting that neither overfitting nor underfitting has occurred.

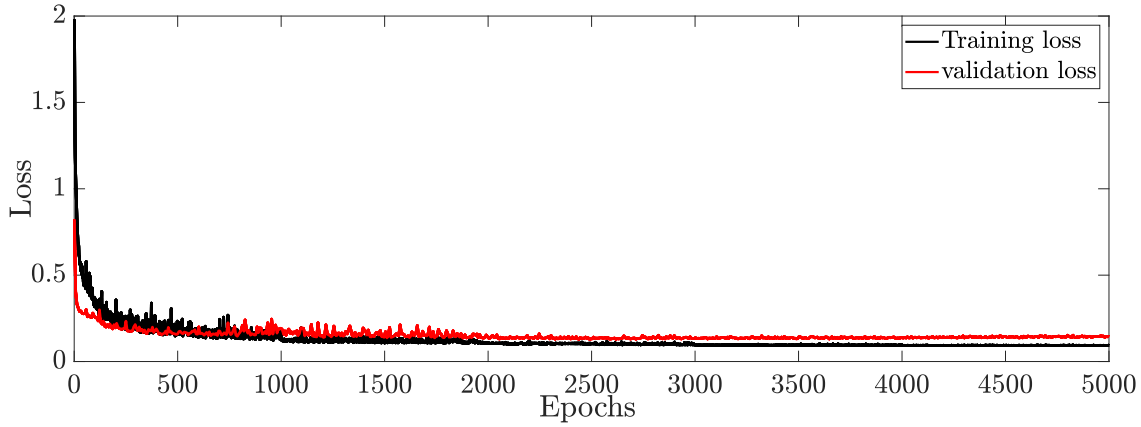


Figure 7.4: Loss vs. epochs for proposed deep neural network model

To develop this deep network, which has more than 24,148 learnable parameters, a large data set including 65,000 consecutive engine cycles are used. Therefore, the

Table 7.1: Specification of training proposed deep network to predict performance and emission

Name	**Value**
Optimizer	Adam
Maximum epochs	5000
Mini batch size	512
Learn rate drop period	1000 epochs
Learn rate drop factor	0.5
L2 Regularization	10
Initial learning rate	0.001
Validation frequency	64 iteration
Momentum	0.9
Squared gradient decay	0.99

diesel engine was run for 65,000 cycles, and all five inputs– SOI of main, DOI of main, DOI of pilot, P2M time (t_{P2M}), and fuel rail pressure– are changed randomly. A random signal is used to change both the amplitude and frequency of these five inputs. The training and validation results of the proposed model are compared to experimental values in Figures 7.5 and 7.6. Cycles 1 to 40,000 are utilized for training, cycles 40,001 to 52,000 are used for validation, and cycles 52,001 to 65,000 for testing. The SOI of the pilot is calculated based on P2M time and is illustrated in Figure 7.6 as the control behavior is easier to understand based on it.

The accuracy of these models for each output are summarized in Table 7.2. For accuracy, the Root Mean Square Error (RMSE) and Normalized Root Mean Square Error (NRMSE) are used which are

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{Y}(i) - Y(i))}{N}} \tag{7.6}$$

$$\mathrm{NRMSE} = \frac{RMSE}{Y_{\max} - Y_{\min,}} \times 100 \tag{7.7}$$

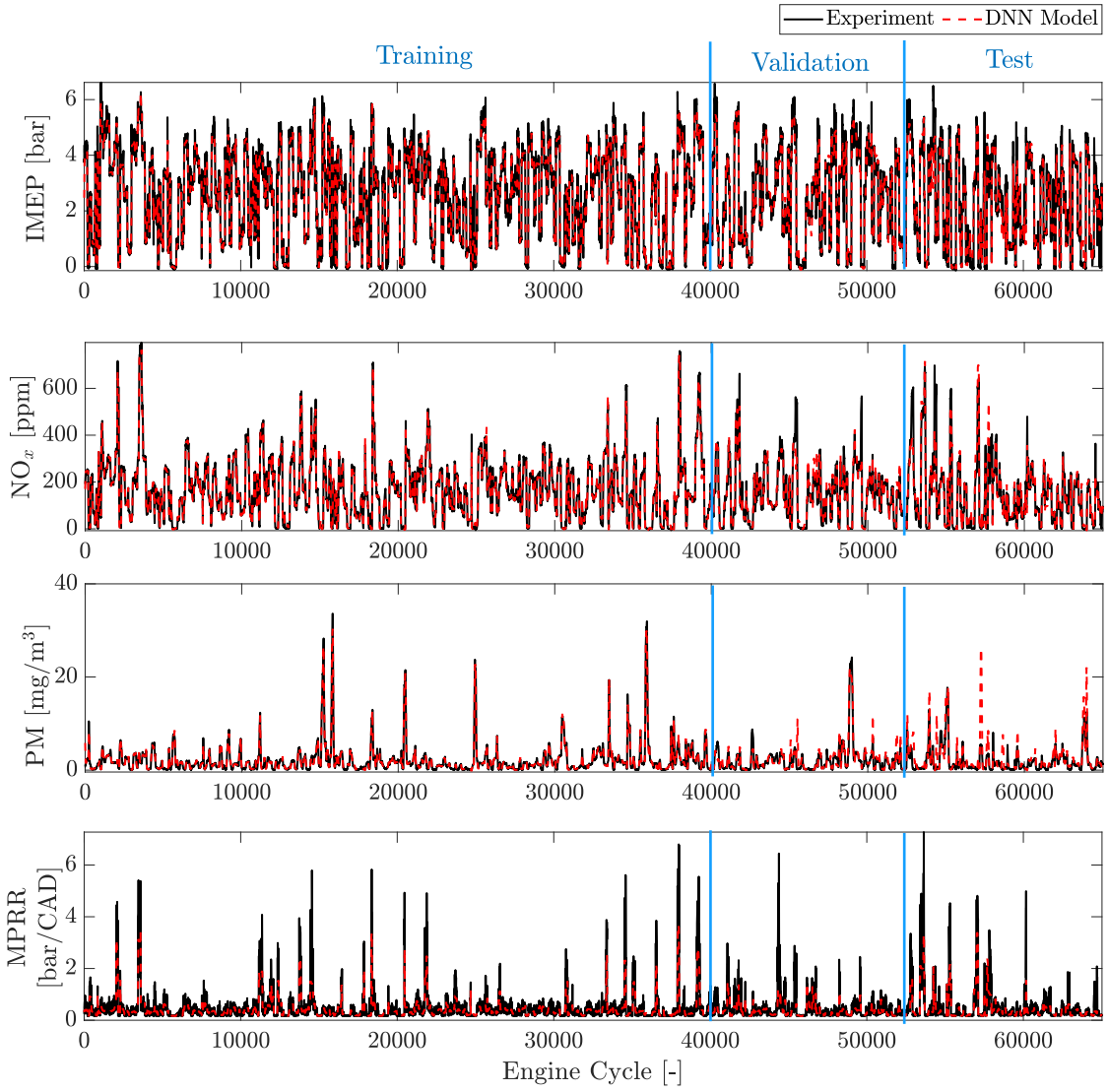


Figure 7.5: Training, validation, and testing results for LSTM-based DNN model vs. experimental data: a) IMEP, b) NO_x, c) PM, and d) MPRR

As presented in Table 7.2, IMEP is the most difficult parameter for the model to predict since it has a 7% error in training while other outputs are predicted with less than a 3% error. The same trend can be seen for the testing data, where IMEP has a 10% error while both emissions have an error of less than 8%. The MPRR prediction is more accurate than others for test data, with a 2.72% error. The model could be further tuned to improve prediction accuracy by adding more hidden and

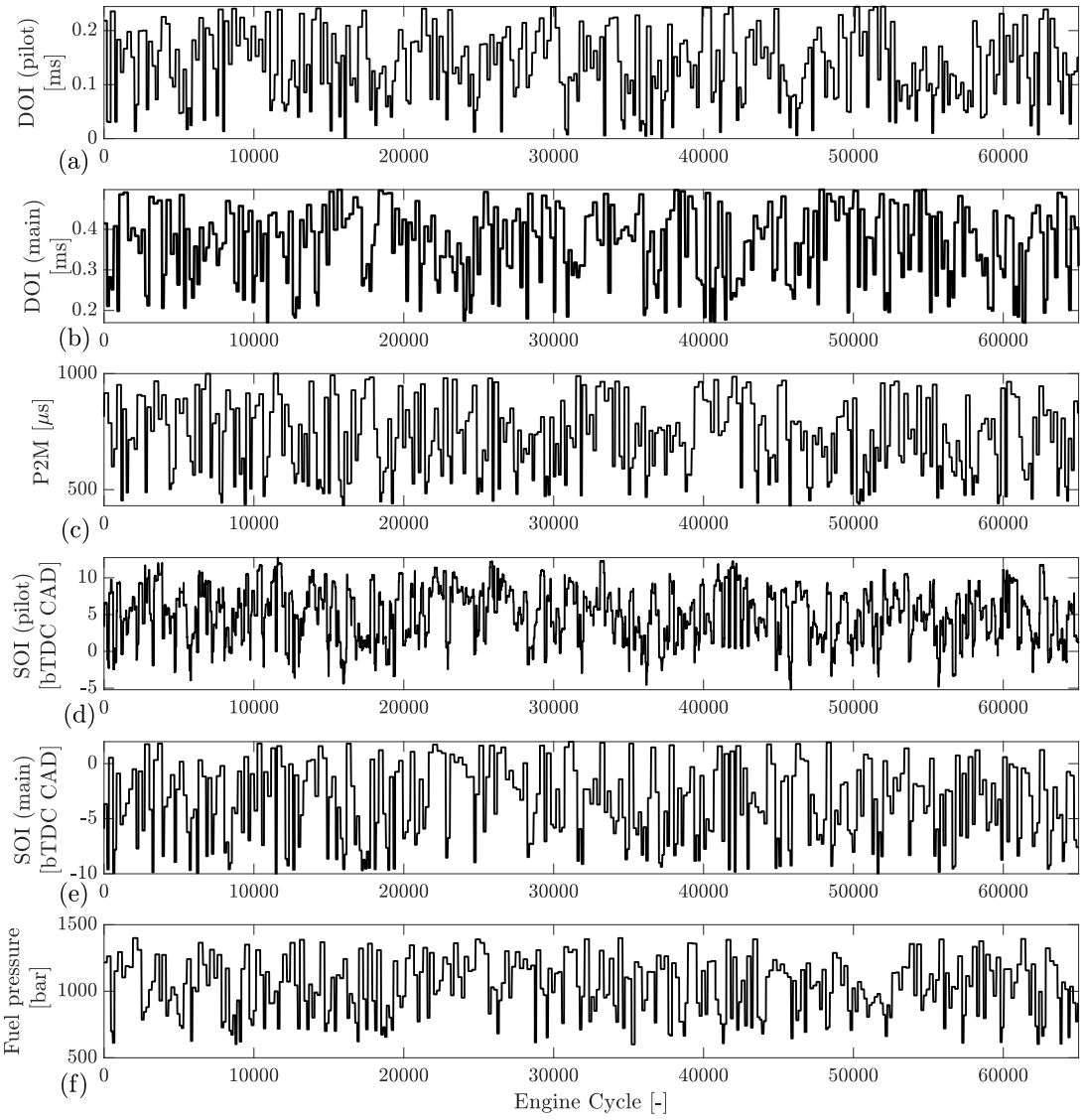


Figure 7.6: Experimental data inputs for training, validation, and testing data that used for the LSTM-based DNN model: a) DOI of pilot injection, b) DOI of main injection, c) duration between end of pilot injection and start of main injection, d) SOI of pilot injection, e) SOI of main injection, and f) fuel rail pressure

cell states to the LSTM layer; however, adding more states would clearly increase the computational time of the model on the real-time hardware. Therefore, this model was improved only by adjusting the number of hidden units of the fully connected layers. This model will be used for the NMPC design in the subsequent section.

Table 7.2: Error of DNN model vs experimental using RMSE and normalized RMSE: IMEP, FQ. PM, and MPRR.

	Unit	Training	Validation	Testing
IMEP	[bar]	0.29	0.36	0.41
	[%]	7.04	8.77	9.75
NOx	[ppm]	18.41	39.33	46.98
	[%]	2.90	6.19	7.40
PM	[mg/m^3]	0.37	1.27	2.38
	[%]	1.16	4.03	7.54
MPRR	[bar/CAD]	0.17	0.24	0.25
	[%]	2.45	2.63	2.72

7.2 Nonlinear Model Predictive Control

The details of the design and structure of the proposed NMPC is described in this section. For the NMPC, an optimization problem is solved at each sample instance for a receding horizon to determine the control inputs. In this case, the goal is to minimize engine-out emissions (NO_x and PM emissions) while simultaneously reducing fuel consumption and maintaining the requested output torque. Additionally, the NMPC is required to meet constraints on the control output and engine combustion metrics. Finally, the developed NMPC is implemented using dSPACE MABX II for diesel engine control. Compared to the simulation study in Chapter 6, the PM soot emission reduction has been added so the NO_x and PM trade-off of the diesel engine can be considered. In addition, multi-pulse injection timing and duration (see Figure 7.2) along with the fuel rail pressure control are added to the model in Chapter 7 to have more degree of freedom to control the CI engine-out emissions.

7.2.1 Controller Design

For the NMPC design, the Optimum Control Problem (OCP) with a finite horizon of length N_p which has the cost function, $J(\boldsymbol{u}(k))$, is now

$$\begin{aligned} J(\boldsymbol{u}(k)) = \sum_{i=0}^{N_p-1} \Big[& \underbrace{||\text{IMEP}(k+i) - \text{IMEP}_{\text{ref}}(k+i)||^2_{w_{\text{IMEP}}}}_{\text{IMEP tracking}} \\ & + \underbrace{||\text{NO}_x(k+i)||^2_{w_{\text{NO}_x}} + ||\text{PM}(k+i)||^2_{w_{\text{PM}_x}}}_{\text{NO}_x \text{ and PM minimization}} \\ & + \underbrace{||\text{DOI}_{pilot}(k+i)||^2_{w_{\text{DOI}_{pilot}}} + ||\text{DOI}_{main}(k+i)||^2_{w_{\text{DOI}_{main}}}}_{\text{fuel consumption minimization}} \\ & + \underbrace{||u(k+i) - u(k+i-1)||^2_{w_{\Delta u}}}_{\text{control effort penalty}} \Big] \end{aligned} \tag{7.8}$$

where w_j, $j \in [\text{IMEP}, \text{NO}_x, \text{PM}, \text{DOI}_{\text{pilot}}, \text{DOI}_{\text{main}}, \Delta u]$, are the MPC weights, and $\boldsymbol{u}(k)$ is the optimization decision defined as

$$\boldsymbol{u}(k) = [u(k)^T \;\; u(k+1)^T \;\; \ldots \;\; u(k+N_p-1)^T] \tag{7.9}$$

The outputs and manipulated variables of the model are defined as:

$$\begin{aligned} y(k) &= \begin{bmatrix} \text{IMEP} & \text{NO}_x & \text{PM} \end{bmatrix}^T \\ u(k) &= \begin{bmatrix} \text{DOI}_{\text{pilot}} & \text{DOI}_{\text{main}} & t_{\text{P2M}} & \text{SOI}_{\text{main}} & P_{\text{fuel}} \end{bmatrix}^T. \end{aligned} \tag{7.10}$$

The difference between the current IMEP and the required IMEP (IMEP error) is penalized in the cost function (Eq. 7.8), along with the NO_x, PM, and fuel injection duration minimization. This formulation results in the following OCP

$$\begin{aligned} \min_{\boldsymbol{u}(k)} \quad & J(\boldsymbol{u}(k)) \\ \text{s.t.} \quad & x(0) = \bar{x}(0) \\ & x(k+1) = f(x(k), u(k)) \quad k = 0, \ldots, N_p - 1 \\ & \underline{x} \le x(k) \le \overline{x} \qquad\qquad k = 0, \ldots, N_p \\ & \underline{u} \le u(k) \le \overline{u} \qquad\qquad k = 0, \ldots, N_p - 1 \end{aligned} \tag{7.11}$$

which is solved at each discrete-time instant. The states $x_{k+1} = f(x(k), u(k))$ are the developed dynamic model in Eq. 7.1 and $x(k)$ are the augmented states that are defined by adding inputs in Eq. 6.5 as

$$x(k) = \begin{bmatrix} h_{LSTM}(k) \in \mathbb{R}^4 \\ c_{LSTM}(k) \in \mathbb{R}^4 \\ u(k) \in \mathbb{R}^5 \end{bmatrix} \in \mathbb{R}^{13} \tag{7.12}$$

where u is manipulated variables defined in Eq. 7.10. The manipulated variables are added to states in order to create a δu term in cost function [206, 207].

Figure 7.7 illustrates the LSTM-NMPC schematic. For the measured states, MPRR and IMEP are provided by the 0.1° FPGA calculation while PM and NO_x are measured in real-time. They are then combined with the cell and hidden states that are estimated by the DNN model. The prediction horizon, p, is set to five steps with a one-step control horizon, which was used in the simulation in Chapter 6.

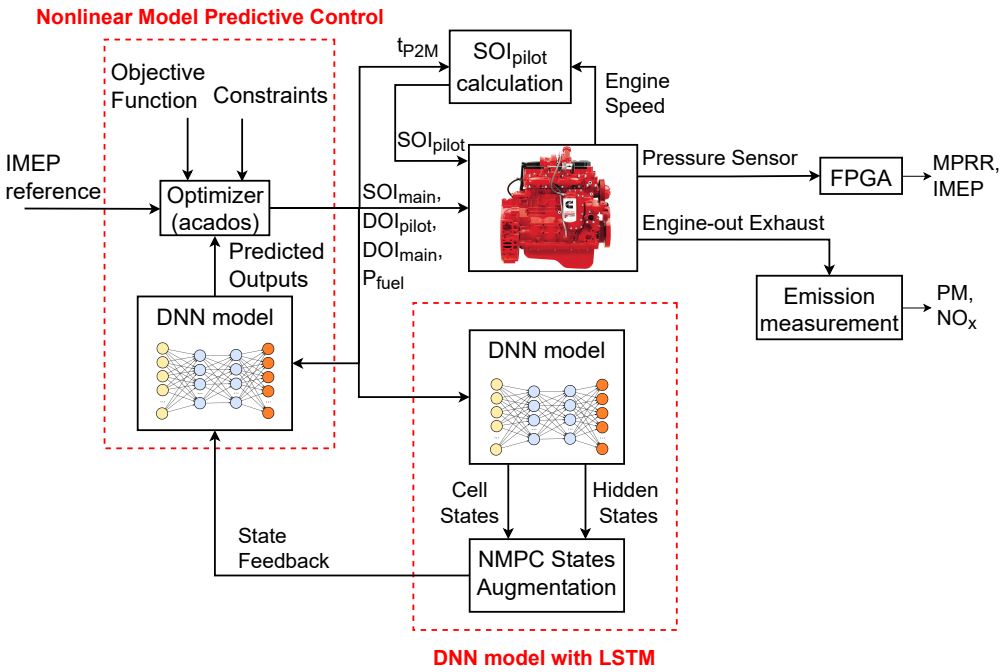


Figure 7.7: Block diagram of LSTM-NMPC structure

7.2.2 Constraint definition

One advantage of the NMPC is the ability to impose constraints on the control output and states. Here, constraints on control outputs are used to match hardware limitations, and constraints on measured states are used to ensure safe engine operation. Table 7.3 lists the lower and upper bounds for the state and control constraints that are implemented.

Table 7.3: Constraint Values

Min Value $(\underline{x}, \underline{u})$	Variable (x, u)	Max Value $(\overline{x}, \overline{u})$
0 bar	IMEP	7 bar
0 ppm	NO_x	500 ppm
0 mg/m^3	PM	10 mg/m^3
0 bar/CAD	MPRR	5 bar/CAD
0.17 ms	DOI_{pilot}	0.24 ms
0.17 ms	DOI_{main}	0.55 ms
0.43 ms	t_{P2M}	1 ms
-10 bTDC CAD	SOI_{main}	2 bTDC CAD
600 bar	P_{fuel}	1400 bar

The limits are imposed on IMEP to limit the engine to low-mid load operation. The upper IMEP constraint is below the engine maximum load but is imposed to keep the engine operating near the model calibration range for the initial NMPC real-time implementation.

The upper limit for NO_x and PM is used to constrain peak emission levels, and can be modified for different emission requirements. A limit of 500 ppm for NO_x, and 10 mg/m^3 for PM was selected for this work based on the maximum engine-out emissions recorded when operating the engine using the production ECU at engine speed of 1500 rpm.

Controlling the maximum pressure rise rate (MPRR) is crucial in combustion

engines to ensure quiet and safe/durable engine operation at various engine loads. MPRR is the rate at which the pressure increases in the cylinder, and the maximum permissible MPRR is engine and application-dependent. Here, a typical 5 bar/CAD constraint is implemented to ensure there is no engine damage [40].

Constraints are also imposed on the DOI for both the pilot and main injections. The minimum DOI is limited to keep the injector within its calibration range. The upper limit is defined as 25% higher than the maximum observed injection with the production ECU.

The $\mathrm{SOI}_{\mathrm{main}}$ is constrained on both the upper and lower ends. Early SOI is restricted to avoid early combustion phasing, which can result in high engine noise and low thermal efficiency or engine damage for extreme values. Additionally, late SOI is limited to avoid low thermal efficiency and elevated exhaust gas temperatures. The P2M time is constrained to short durations based on hardware limitations to ensure the injector has fully closed before opening for the main injection. The upper limit is to restrict excessively early pilot injections. Finally, a limit for the fuel pressure is imposed to keep the commanded fuel pressure within the injectors' normal operating range.

7.2.3 Real-time implementation techniques

Real-time implementation of NMPC is done on a dSPACE MABX II. The previous work comparing NMPC execution times of various solvers is was found that `acados` (High-Performance Interior-Point Method (HPIPM) QP solver) outperforms `FORCES PRO` and `fmincon` as detailed in Chapter 6.

In the implementation of the `acados` solver, a maximum QP iteration of 50 and a maximum Nonlinear Programming (NLP) iteration / SQP steps of 5 are used. From the simulation results in Chapter 6, an average runtime of 12.20 ms and a maximum runtime of 31.56 ms for `acados` has been observed while the average runtime of `fmincon` was 786.02 ms. `FORCES PRO` [200, 201] was also tested in the simulation

and showed an improvement in runtime over the `fmincon` but was slower than the `acados` implementation. The difference in runtime between solvers is attributed to two possible causes and is detailed in Chapter 6: Section 6.4.

Because `acados` utilizing `HPIPM` provides the fastest solution time of the solvers tested in the simulation it was used for the real-time implementation. The simulation time of the NMPC is on a modern PC while the real-time implementation is solved on an embedded processor located within the prototype ECU. The control is driven with the cycle time dependent on the speed of the engine. A turnaround time that varies from 100 ms at 1200 rpm to 66.7 ms at 1800 rpm is needed. The real-time implementation time based on `acados` has a maximum of 63.0 ms and an average of 62.3 ms which is feasible for real-time implementation of these engine speeds.

A difficulty in implementing the NMPC is related to compiling the necessary libraries for real-time implementation on the MABX. In this work `acados`, `blasfeo` and `HPIPM` had to be cross-compiled to run `acados` on the embedded system. Helpful instructions for this process can be found on the `acados` documentation website ([210] "Embedded Workflow"). With this overview, it is possible to use the `acados` interface of MATLAB/Simulink© on the MABX by using a code-generated s-function in Matlab. The latter refers to the target system specific libraries compiled in the above-mentioned "Embedded Workflow" section, and the code-generated problem-specific source files of the `acados` OCP instance and solver. The next section provides the real-time implementation results of the developed NMPC along with a comparison with the production of Cummin's ECU.

7.3 Experimental Results

The developed NMPC is experimentally tested for IMEP tracking performance, minimization of emissions (NO_x and soot), and fuel consumption while also meeting constraints on MPRR and SOI. The controller is subjected to a step and smooth change with a bandwidth of approximately 1 Hz in target IMEP. Then to test the controller's

robustness the engine speed is changed while maintaining a constant IMEP. Finally the controller is compared to the production ECU or BM for comparison to the NMPC.

7.3.1 Experimental results in changing IMEP

The deep neural network based NMPC is first evaluated for its load tracking performance by following a step reference between 2 and 6 bar IMEP. This load range is selected to match the lower and upper bounds of the training data that is used for the model's development. Figure 7.8 shows the multiple steps used to understand the performance of the controller on engine inputs and outputs.

The NMPC is capable of achieving the target IMEP within an engine cycle. A slight overshoot and some oscillation is seen after both the increase and decrease in target value. The oscillation in IMEP after the step change is attributed to the relatively slow dynamics of the fuel pump and resulting oscillation in fuel pressure as seen in Figure 7.8(h). The delay in the fuel system to a pressure change was not modeled, and thus the NMPC expects instantaneous changes in pressure which is not possible with the given common rail fuel system. Overall, the controller is found to be capable of achieving the reference setpoint by 0.26 bar average error and RMSE of 0.61 bar.

The changing NO_x and PM emissions can be seen in Figure 7.8(b) and (c), respectively. As expected an increase in IMEP results in an increase in emissions. However, to better quantify the improvement of this controller it will be compared to the production ECU (BM) later in this section.

The engine speed is controlled by a dynamometer and a variation of ± 50 RPM is observed by the dynamometer's PI speed controller. Finally, no constraint violations in MPRR, DOI, SOI or fuel pressure or other outputs are observed.

The NMPC is then tested by providing a smooth IMEP reference with a bandwidth of approximately 1 Hz. The controller's performance can be seen in Figure 7.9 where

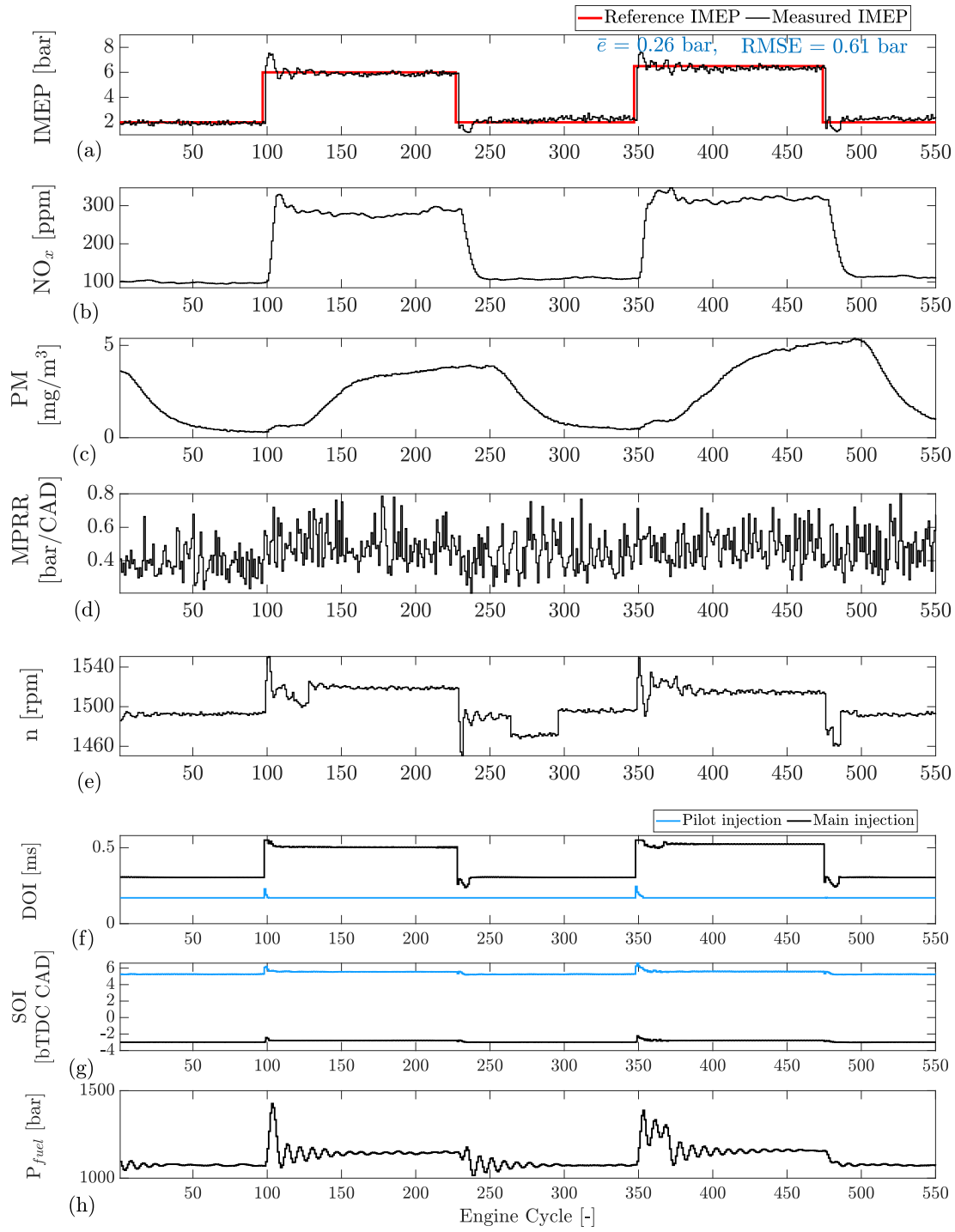


Figure 7.8: Experimental results for step changes of IMEP: a) IMEP, b) NO_x, c) PM, d) MPRR, e) engine speed, f) DOI, g)SOI, h) fuel rail pressure

the NMPC is again able to successfully track the target load. The controller is able to track the reference with a 0.16 bar average error and an RMSE of 0.20. Again, the

NMPC does not violate any input or output constraints. As shown in Figure 7.9(h) there is oscillation in the fuel rail pressure. The current IMEP tracking is acceptable; however, to further improve the controller a more accurate fuel pressure controller may be required.

Overall, the developed NMPC performs extremely well at 1500 rpm (the speed at which the deep neural network model is trained) for tracks both step and smooth IMEP reference with a bandwidth of approximately 1 Hz. In the next section, the robustness of the NMPC to a changing engine speed is experimentally tested. Since the model was developed at constant speed, variations in engine speed result in a model mismatch and are seen as an unmodeled disturbance.

7.3.2 Experimental results in changing engine speed

To further evaluate the NMPC engine, the speed is changing between 1200 to 1800 rpm while holding IMEP constant at 5 bar. This test will evaluate the control beyond the single speed at which the embedded model in NMPC was trained. The controller's performance in tracking step changes in load is shown in Figure 7.10. Steps of 100 rpm are implemented for the first 1500 engine cycles and then for the remaining cycles larger steps of up to 500 rpm are tested. Overall, the controller is able to maintain the IMEP setpoint over changing speeds with an average error of 0.27 bar. Once again, the NMPC is able to maintain all set constraints over the range of speeds tested.

A similar result can be seen with smooth engine speed change with a bandwidth of approximately 1 Hz, as shown in Figure 7.11. Again, no constraint violation occurs. The NMPC is able to maintain commanded engine load over the step and smooth engine speed change with a bandwidth of approximately 1 Hz on the engine.

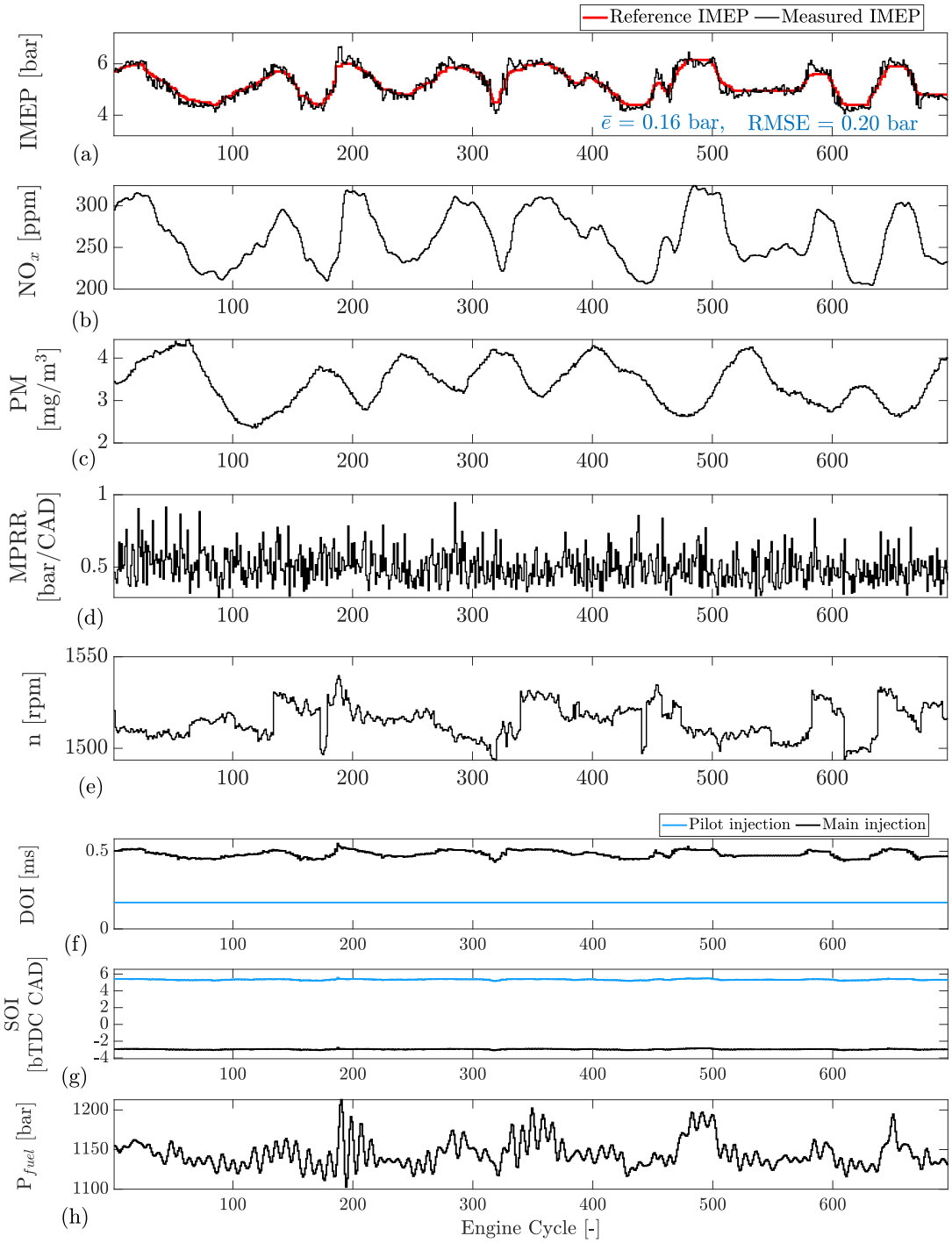


Figure 7.9: Experimental results of smooth IMEP reference with a bandwidth of approximately 1 Hz: a) IMEP, b) NO_x, c) PM, d) MPRR, e) Engine speed, f) DOI, g)SOI, h) fuel rail pressure

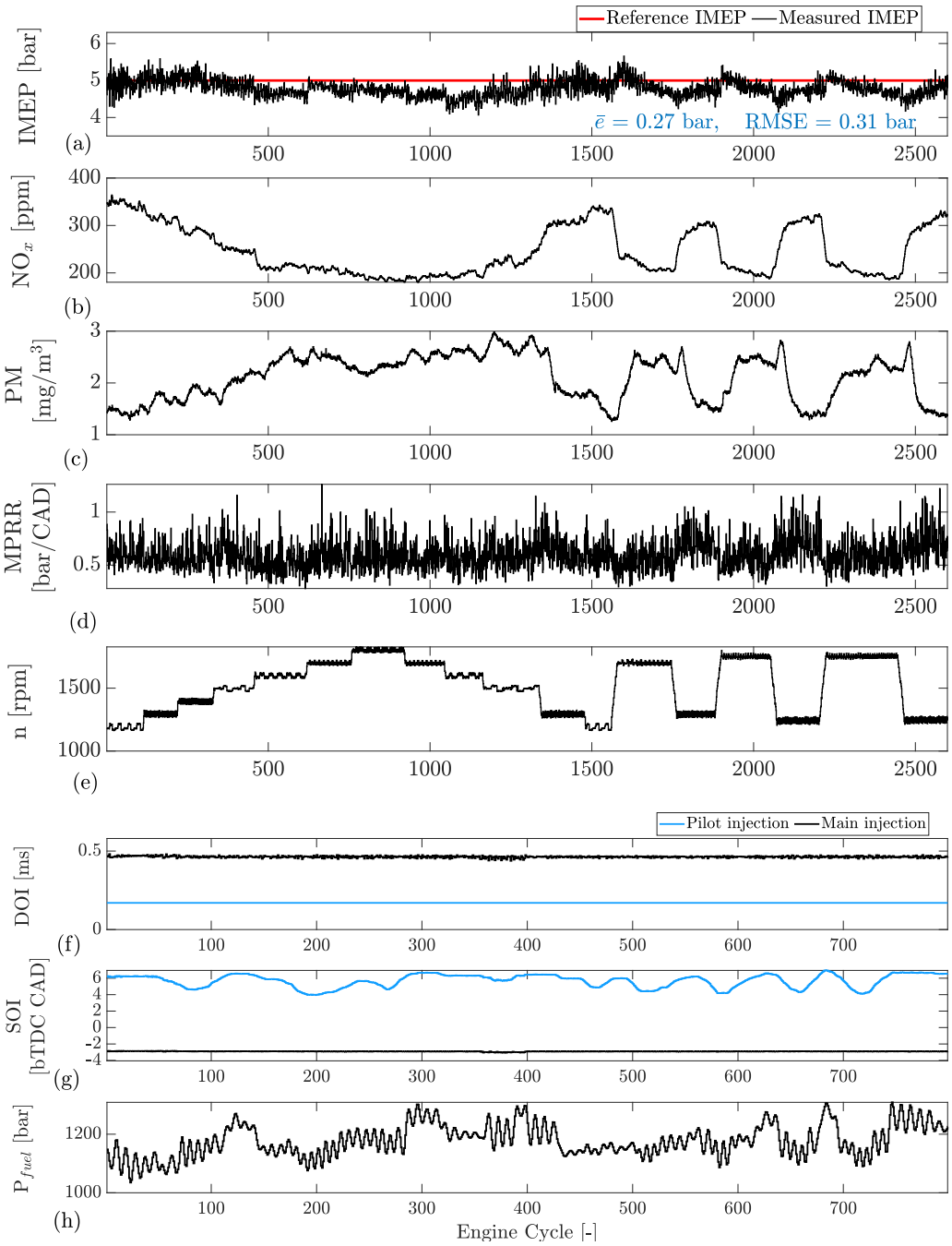


Figure 7.10: Experimental results of step changes of engine speed: a) IMEP, b) NO_x, c) PM, d) MPRR, e) Engine speed, f) DOI, g) SOI, h) fuel rail pressure

7.3.3 LSTM-NMPC vs Cummins calibrated ECU

In this section, the NMPC is compared to a BM engine controller. Here, the BM is taken as the replicated Cummins production ECU in the MicroAutoBox. Table 7.4

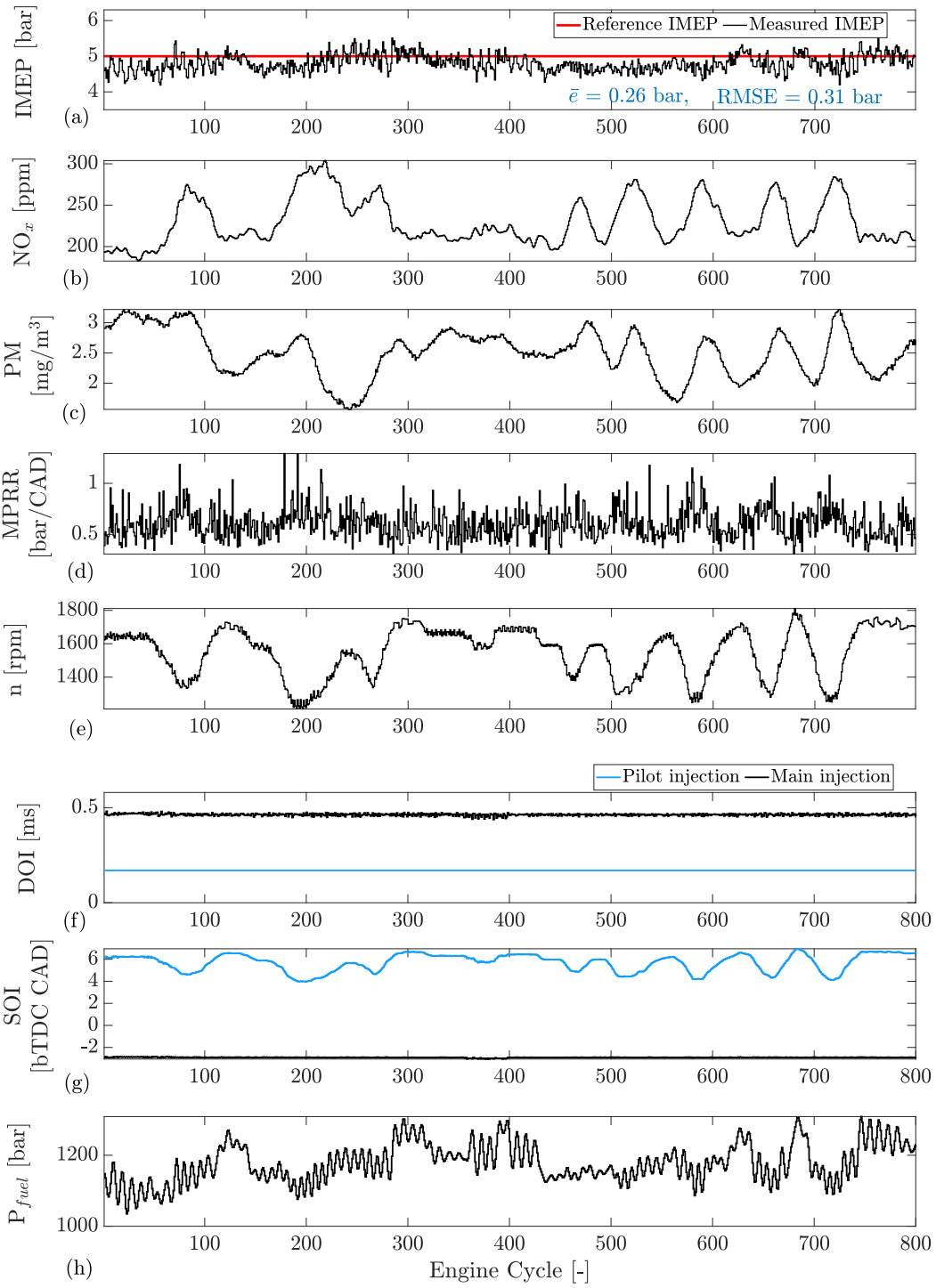


Figure 7.11: Experimental results of smooth engine speed change with a bandwidth of approximately 1 Hz: a) IMEP, b) NO_x, c) PM, d) MPRR, e) Engine speed, f) DOI, g) SOI, h) fuel rail pressure

presents 9 different load/speed cases varying from 2-6 bar IMEP and 1200-1800 rpm. Each row in the table represents the average of 200 cycles. It should be noted that the average IMEP may not necessarily perfectly match the reference value for either the BM or the NMPC. Generally the NMPC achieves closer to the reference value as the IMEP is input to the NMPC controller where the BM utilizes a feed-forward table. To compare normalized to load values, especially for emission comparisons, both NO_x and PM are converted to g/kWh unit, which represents the mass of emission produced per generated power. For the same reason, thermal efficiency is also compared.

Table 7.4: Proposed NMPC results compared to the BM, Cummins calibrated ECU, for different engine operating conditions (averaged over 400 cycles). Negative value represents that the LSTM-NMPC value is lower than BM. Δ : LSTM-NMPC-BM, IMEP: Indicated mean effective pressure, FQ: Fuel Quantity, η_{th} thermal efficiency. PM: Particle Matter

Case Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Reference IMEP [bar]	5.0	5.0	5.0	5.0	2.0	3.0	4.0	5.0	6.0
BM IMEP [bar]	4.83	5.22	5.04	5.03	2.29	3.09	3.91	4.94	6.02
NMPC IMEP [bar]	5.08	4.88	4.91	4.83	1.96	2.98	3.98	4.92	6.05
Engine Speed [rpm]	1190	1296	1701	1801	1509	1504	1504	1503	1504
ΔFQ [%]	-7.9	-11.0	-10.4	-9.6	-14.9	-8.3	-7.9	-8.5	-7.3
$\Delta\eta_{th}$ [%] difference	+4.7	+1.8	+3.0	+2.1	+0.1	+1.4	+3.1	+3.0	+3.2
ΔNO_x [%]*	-18.9	-11.2	17.0	3.4	-22.4	-8.7	6.7	9.1	20.7
ΔPM [%]*	-40.8	-35.3	-14.3	-15.4	-8.0	-36.4	-37.5	-43.6	-34.2

* Relative difference Calculated based on [g/kWh] units

Table 7.4 presents the average NO_x, particulate matter (PM), fuel quantity (FQ) and thermal efficiency at the given operating point. The percentage difference of the NMPC compared to the BM is shown. A negative value represents that the NMPC is below the BM. The main outputs that the NMPC is designed to improve are shown in Table 7.4. In all the cases tested, the NMPC is able to reduce the fuel used by 9.5% while also increasing the thermal efficiency by an average of 2.5%.

For the PM emissions, the NMPC shows a significant reduction compared to the BM at every operating point. However, when looking at NO_x there is not a clear

trend. At some operating points there is an increase in NO_x emissions while at others there is a decrease. Overall, on average there is a slight decrease of 0.5% in NO_x emissions. A NO_x and PM emission comparison between the NMPC and BM is shown in Figure 7.12. This shows the well known NO_x-PM trade-off. The cost function in the NMPC contains both NO_x and PM so a reduction in both emissions at the upper end of their range can be seen. When significant PM is present, the NMPC focuses more on reducing PM and may allow a slight increase in NO_x particularly if the NO_x value is fairly low, for example in cases 3 and 4. However, if both the PM and NO_x are high, as in case 1 or 2, the NMPC reduces both. This is a significant advantage as it shows that the NMPC is able to reduce both emissions when they are high, perhaps close to the regulation boundary. In addition, if one emission is comparable to the regulation boundary as in case 9 for PM, the NMPC reduces it significantly by slightly increasing the NO_x value, which is lower than the regulated values for this engine. In this case, the NMPC automatically handles the well-known PM and NO_x trade-off.

7.4 Summary of chapter

It has been demonstrated that deep learning and the NMPC can be successfully implemented for real-time minimization of compression ignition engine-out emissions and fuel consumption while imposing constraints on engine inputs and outputs. The emissions and performance characteristics of a 4.5 litre 4-cylinder Cummins compression ignition engine are modelled using a deep network with seven hidden layers and 24,148 learnable parameters constructed by stacking FC layers with an LSTM layer. This model is then used to design and implement an NMPC in real-time.

To develop this LSTM-NMPC, the open-source software `acados` is used in combination with the quadratic programming solution `HPIPM`. This `acados` embedded programming approach enables real-time operation of the LSTM-NMPC with an average turnaround time of 62.3 milliseconds on dSPACE MicroAutoBox.

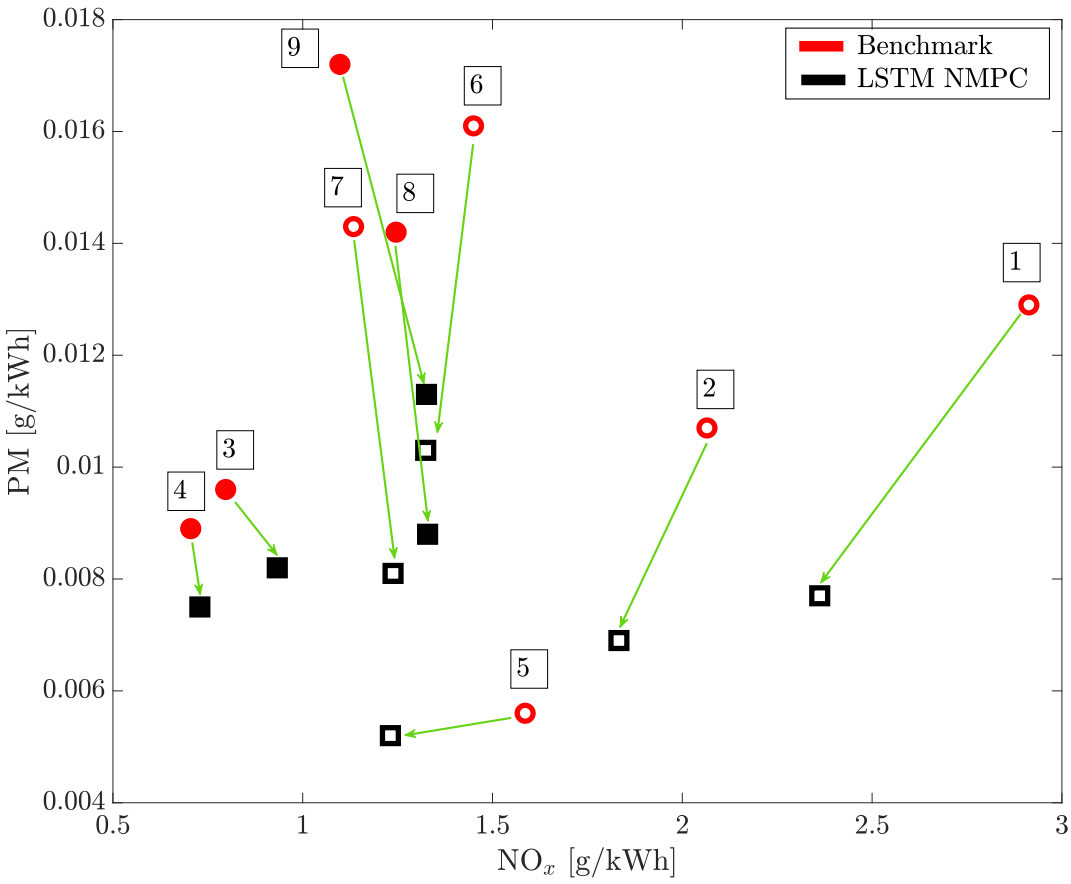


Figure 7.12: Experimental results of PM vs NOx trade-off improvement: in filled shapes ■, NO_x is slightly increased (cases 3, 4, 8, and 9), while in the remaining cases ◘, both PM and NO_x are decreased

A dSPACE MicroAutoBox II rapid prototyping engine controller is used to implement the developed LSTM-NMPC. The MABX II allows for not only implementation of the developed controller but also contains a FPGA that enables the real-time calculation of pressure rise rates and indicated mean effective pressure from measured in-cylinder pressure.

When compared to the Cummins calibrated production controller for mid-load points, the proposed LSTM-NMPC reduces fuel usage by 7.3–14.9% while boosting thermal efficiency by 0.1–4.7% depending on the operating point. This controller is capable of reducing NO_x and PM concentrations by up to 22.4% and 43.6%, respectively. In order to evaluate emission reduction potential, the well known trade-off

between NO_x and particulate emissions is analyzed. It was observed that when large amounts of PM are present, the NMPC prioritizes PM reduction while allowing a slight rise in NO_x if the NO_x amount is relatively low. However, if both PM and NO_x levels are high, the NMPC effectively reduces both. This is a significant benefit since it demonstrates the NMPC's ability to reduce emissions when they are near the imposed constraints or regulatory limit.

The developed controller has been evaluated for both step and smooth IMEP reference with a bandwidth of approximately 1 Hz where it has demonstrated acceptable tracking performance without violating input and output constraints. The average tracking error for a step reference is 0.26 bar with an RMSE of 0.61, while the average tracking error for a smooth IMEP reference with a bandwidth of approximately 1 Hz is 0.16 bar with an RMSE of 0.20.

The controller was evaluated at speeds ranging from 1200-1800 rpm to determine the its robustness for operating outside the training range. The experimental findings demonstrate that tracking and disturbance rejection are appropriate. The controller is able to maintain IMEP set-point with an average error of 0.16 for step and 0.27 for smooth speed change with a bandwidth of approximately 1 Hz. No constraint violation has been observed in any tested cases for the state, outputs, and inputs constants.

PART IV: Machine Learning in Learning-based Controller

Chapter 8

Safe Deep Reinforcement Learning [1]

A deep Reinforcement Learning (deep RL) application is investigated to control the emissions of a Compression Ignition diesel engine in this chapter. The main purpose, similar Chapter 5–7, is to reduce the engine out NO_x emissions while minimizing fuel consumption and reference tracking load error. This case differs from the previous chapters in that prior knowledge of a system model is not used to generate optimal control. Using the Engine Simulation Model (ESM) as a virtual engine in simulation, a Deep Deterministic Policy Gradient (DDPG) is developed–see Chapter 2 for ESM details. To reduce the risk of an unwanted output, a safety filter is added to the deep RL. The developed safe RL is then compared with the LSTM-based NMPC developed in Chapter 6. A well-known learning-based controller is Iterative learning control (ILC) which is used to improve the tracking performance of a system in the presence of repetitive input or disturbances. In this chapter, ILC has been developed for this problem and compared with the developed safe RL.

[1] This chapter is based on [9]

8.1 Deep Reinforcement Learning (Deep RL)

8.1.1 Reinforcement Learning vs. Deep Reinforcement Learning

The main goal of RL is to generate an optimal outcome by finding the best sequence of actions (see Chapter 1, Section 1.1 for more details). Unlike classical machine learning, RL uses an agent to explore, interact with, and learn from the defined system environment. The RL agent learns by receiving the environment observations and rewards and then generates a sequence of actions to reach a specific goal. The RL algorithm can be either model-free or model-based; due to the model requirement, the model-free algorithm has been the main focus in engineering applications [135, 211]. One common algorithm used for model-free RL is Q-learning. In Q-learning, the value of an action for a particular state is learned and the optimal policy is found by maximizing the expected value (Q-value) of the total reward [136].

Using Deep Neural Network (DNN) in RL, is referred to as deep RL, and has solved a wide variety of complicated decision-making tasks that were previously unfeasible to be solved. Earlier versions of RL algorithms had challenges in the design of the features selection. In contrast, deep RL has been able to successfully overcome complicated tasks often despite a limited amount of previous available information.

When an agent performs an action which has the highest reward without further exploring the environmental space it is considered a greedy policy. In continuous spaces, it takes an extremely long time to obtain a greedy policy to optimize the action at each time interval. Therefore, it is not often possible to apply Q-learning easily to continuous action systems. However, an actor-critic method based on the Deterministic Policy Gradient (DPG) algorithm is often a suitable choice for a system with a continuous space [137]. The DPG learning procedure is robust and stable because the off-policy network training takes samples from the replay buffer (which is a finite sized cache used to store previous samples from the environment). This

allows for the reduction of the correlation between samples [212]. Off-policy learning is independent of the agent's actions and it determines the optimal policy regardless of the agent's motivation. So in contrast with on-policy learning where the agent learns about the policy to generate the data, the off-policy estimates the reward for future actions and adds value (an estimated reward) to the new state without following any greedy policy [136].

8.1.2 Deep Deterministic Policy Gradient Agents (DDPG) Algorithm

The Deep Deterministic Policy Gradient Agents (DDPG) is a model-free and off-policy RL algorithm where an actor-critic RL agent calculates an optimal policy by maximizing the long-term reward. DDPG differs from DPG in that DDPG uses a Deep Neural Network (DNN) as an approximator in DDPG to learn for large state and action pairs [212]. In this chapter, a DDPG algorithm is used to minimize the engine-out emissions and fuel consumption while maintaining the same load. As presented in Algorithm 8.1 [212], during training, the actor and critic are updated by the DDPG algorithm at each sample time, and the agent stores past experiences using an experience buffer. The actor and critic are then updated using a mini-batch of those experiences randomly sampled from the buffer. The selected policy action is also perturbed using a stochastic noise model at each training step [135].

In the DDPG algorithm presented in Algorithm 8.1, a copy of the critic $Q'(x, u|\theta^{Q'})$ and actor network $\mu'(x|\theta^{\mu'})$ is initially created. Then, these target network weights are updated "gently" to follow the learned networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ with $\tau \ll 1$. The target value is constrained to change at a slow rate to improve the stability of learning. Exploration is a significant challenge of learning when the action spaces are continuous. Since exploration is an off-policy algorithm, in DDPG, the learning algorithm and exploration policy μ can be formed by combining a noise process $\mathcal{N}$ with the actor policy. In the DDPG algorithm, the Ornstein-Uhlenbeck process noise

model is used to create a noise process for agent exploration [135, 213, 214].

Algorithm 8.1: Deep Deterministic Policy Gradient Agents (DDPG) algorithm [212]

Initialize critic network randomly $Q(x, u|\theta^Q)$ with weights θ^Q

Initialize actor network randomly $\mu(x|\theta^\mu)$ with weights θ^μ

Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$

Initialize replay buffer R

For episode = 1, E_f

Initialize a random noise process $\mathcal{N}$ to add action exploration

Receive initial observation state $x(1)$ For $k = 1$, k_f

Select action $a_t = \mu(x(k)|\theta^\mu) + \mathcal{N}(k)$

Execute action $u(k)$ and observe reward $r(k)$ and observe new state $x(k+1)$

Store $(x(k), u(k), r(k), x(k+1))$ in R

Sample a random minibatch of N transition $(x(k), u(k), r(k), x(k+1))$ from R

Set $\hat{r}(k) = r(k) + \gamma Q'\big(x(k+1), \mu'(x(k+1)|\theta^{\mu'})|\theta^{Q'}\big)$

Update critic by minimizing the loss:

$L = \frac{1}{N}\Sigma_i(\hat{r}(k) - Q(x(k), u(k)|\theta^{\theta^Q}))^2$

Update actor based on the sampled policy gradient:

$\nabla_{\theta^\mu} = \frac{1}{M}\sum_{i=1}^{M} G_u G_\mu$

$G_u = \nabla_u Q(x(k), u(k)|\theta^Q)$ where $u = \mu(x(k)|\theta^\mu)$

$G_\mu = \nabla_{\theta^\mu}\mu(x(k)|\theta^\mu)$

Update the target network:

$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$

$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$

8.1.3 Safe Deep Deterministic Policy Gradient

Despite all the advantages of deep RL, it relies on experience and interaction with the environment (here ESM). To enforce output constraints, the following optimization-based filter is added to the DDPG algorithm

$$\begin{aligned} \underset{x}{\text{Minimize:}}\ & ||u(k) - u_{RL}(k)||_2^2 \\ \text{subject to:}\ & y(k) < y_{\max} \\ & u_{\min} < u(k) < u_{\max} \end{aligned} \tag{8.1}$$

where $u(t)$ is a safe action and $u_{RL}(k)$ is the DDPG generated action. The goal of this optimization is to enforce that the output is not to exceed the defined output maximum value $y_{\max}$ given lower ($u_{\min}$) and upper bound ($u_{\max}$) of actions while minimizing the difference between the DDPG generated action and the safe action.

For the ESM plant model, the constrained output and control actions are defined as

$$\begin{aligned} y(k) &= [\mathrm{NO}_x(k) \;\; T_{\mathrm{out}}(k)]^T \\ u(k) &= [\mathrm{FQ}(k) \;\; \mathrm{SOI}(k) \;\; \mathrm{VGT}(k)]^T \end{aligned} \tag{8.2}$$

where FQ(k), SOI(k), and VGT(k) are injected fuel quantity, start of main injection, and Variable-geometry turbocharger (VGT) valve rate, respectively. The outputs are defined as engine-out $\mathrm{NO_x}$ emission $\mathrm{NO}_x(k)$ and output torque $T_{\mathrm{out}}(k)$. To simplify the control problem similar to that described in Chapters 5 and 6, the pilot-injection is kept constant at 9 mg and is injected 8 Crank Angle Degree (CAD) before the main injection.

The optimization of, Eq. 8.1, uses quadratic programming (QP) to find the control action $u(k)$ that minimizes the function $||u(k) - u_{RL}(k)||_2^2$. The QP solver applied the following constraints to the optimization:

$$\begin{aligned} f\big(x(k)\big) + g\big(x(k)\big)u(k) &< y_{\max} \\ u_{\min} < u(k) &< u_{\max} \end{aligned} \tag{8.3}$$

Where $f\big(x(k)\big)$ and $g\big(x(k)\big)$ are coefficients of the constraint function which depend on the modeled plant states $x(k)$. Linear plant dynamics developed in Chapter 5 is used here:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{8.4}$$

where A and B are state-space matrices developed using a Autoregressive with Extra

Input (ARX) model as:

$$
\begin{aligned}
A &= \begin{bmatrix} 0.7286 & 7.1252 & -0.0019 \\ 0.0002 & 0.9859 & 8.9878 \times 10^{-6} \\ -0.6105 & 33.94287 & 0.9076 \end{bmatrix} \\
B &= \begin{bmatrix} 1.2639 & -1.0899 & 1.0084 \times 10^{-5} \\ -0.0007 & 0.0014 & -1.01397 \times 10^{-5} \\ 2.9360 & -8.2453 & -0.0106 \end{bmatrix} \\
C &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{8.5}
$$

where the constrained output $y(k)$, states $x(k)$ and control actions $u(k)$ are defined as

$$
\begin{aligned}
y(k) &= [\mathrm{NO}_x(k) \;\; P_{\mathrm{man}}(k) \;\; T_{\mathrm{out}}(k)]^T \\
y(k) &= [\mathrm{NO}_x(k) \;\; T_{\mathrm{out}}(k)]^T \\
u(k) &= [\mathrm{FQ}(k) \;\; \mathrm{SOI}(k) \;\; \mathrm{VGT}(k)]^T
\end{aligned}
\tag{8.6}
$$

where $\mathrm{FQ}(k)$, $\mathrm{SOI}(k)$, and $\mathrm{VGT}(k)$ are injected fuel quantity, start of main injection, and Variable-geometry turbocharger (VGT) valve rate (percentage opening), respectively. The states are defined as engine-out $\mathrm{NO}_x(k)$ emission, intake manifold pressure $P_{\mathrm{man}}(k)$, and output torque $T_{\mathrm{out}}(k)$. By substituting Eq. 8.4 in Eq. 8.3, $f\big(x(k)\big)$ and $g\big(x(k)\big)$ can be found as

$$
\begin{aligned}
f\big(x(k)\big) &= CAx(k) \\
g\big(x(k)\big) &= CB
\end{aligned}
\tag{8.7}
$$

Substituting system matrices (Eq. 8.5) in Eq. 8.7 results in

$$
\begin{aligned}
f\big(x(k)\big) &= \begin{bmatrix} 0.7286 & 7.1252 & -0.0019 \\ -0.6105 & 33.94287 & 0.9076 \end{bmatrix} x(k) \\
g\big(x(k)\big) &= \begin{bmatrix} 1.2639 & -1.0899 & 1.0084 \times 10^{-5} \\ 2.9360 & -8.2453 & -0.0106 \end{bmatrix}
\end{aligned}
\tag{8.8}
$$

To simplify the control problem, the pre-injection is kept constant at 9 mg that is injected 8 Crank Angle Degree (CAD) before the main injection.

The upper bound of NO_x is used to regulate peak NO_x exhaust emissions levels. This value depends on government legislation limits. Here, the experimental maximum NO_x level of 500 ppm is observed for the production TIER 3 engine during standard operation load range and this value is used as the upper bound of NO_x. A 500 N.m torque is used as the upper bound for load to avoid high loads beyond the defined operating range. To regulate the amount of injected fuel and avoid large fuel injections, a constraint is imposed for injected fuel amount of 10 mg/cycle to 90 mg/cycle. To avoid late injections that cause combustion inefficiency and high exhaust gas temperatures, a lower limit of SOI is also imposed. Due to the physical limitations, the VGT is limited between 70% to 100%. To avoid increased combustion noise and cause low combustion efficiency, SOI is also limited by an upper bound. Therefore, the constraints can be summarized as:

$$\begin{aligned} y_{\min} &= [\mathrm{NO}_{x,\min}(k) \;\; T_{\text{out, min}}(k)]^T = [0 \;\; 0]^T \\ y_{\max} &= [\mathrm{NO}_{x,\max}(k) \;\; T_{\text{out, max}}(k)]^T = [500 \;\; 500]^T \\ u_{\min} &= [\mathrm{FQ}_{\min}(k) \;\; \mathrm{SOI}_{\min}(k) \;\; \mathrm{VGT}_{\min}(k)]^T = [10 \;\; -2 \;\; 70]^T \\ u_{\max} &= [\mathrm{FQ}_{\max}(k) \;\; \mathrm{SOI}_{\max}(k) \;\; \mathrm{VGT}_{\max}(k)]^T = [90 \;\; 11 \;\; 100]^T \end{aligned} \tag{8.9}$$

A schematic of safe DDPG for minimizing diesel engine emissions and fuel consumption while maintaining load is shown in Figure 8.1. The states of the system for the DDPG algorithm are defined as

$$x(k) = [\mathrm{NO}_x(k) \;\; e_{T_{\text{out}}}(k) \;\; T_{\text{out}}(k) \;\; P_{\text{man}}(k)]^T \tag{8.10}$$

where $P_{\text{man}}(k)$ is intake manifold pressure and $e_{T_{\text{out}}}(k)$ is output torque tracking error defined as

$$e_{T_{\text{out}}}(k) = T_{\text{out, r}}(k) - T_{\text{out}}(k) \tag{8.11}$$

where $T_{\text{out, r}}(k)$ is the requested load reference.

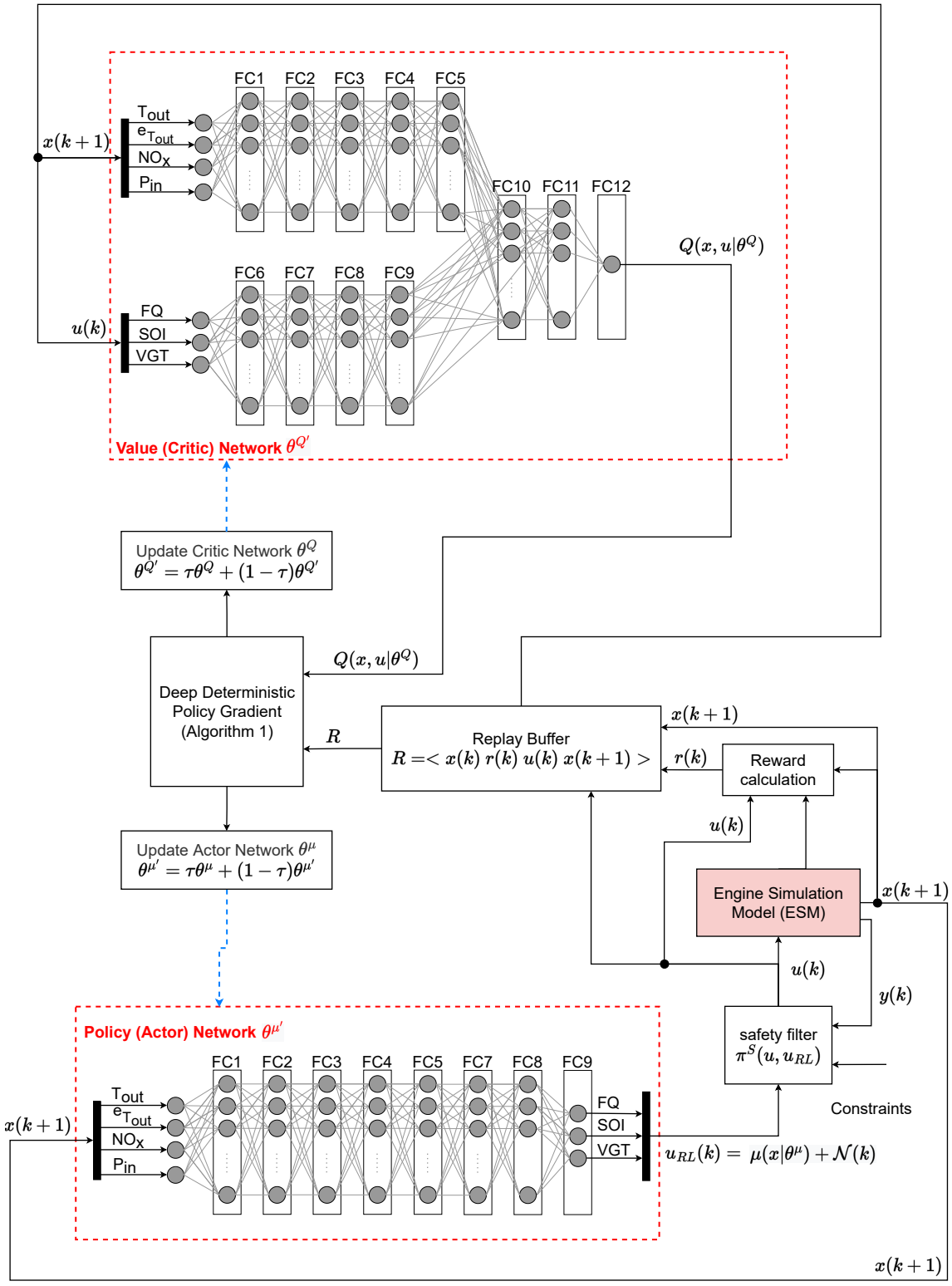


Figure 8.1: Safe Deep Deterministic Policy Gradient schematics to minimize diesel engine fuel consumption and NO_x reduction while maintaining output torque

To achieve the control objective, and output torque error, its derivative the fuel quantity and NO_x values are added to the reward function, $r(k)$ as

$$r(k+1) = -\Bigg(r_1 e_{T_{\text{out}}}(k) + r_2 \frac{e_{T_{\text{out}}}(k) - e_{T_{\text{out}}}(k-1)}{T_s} + r_3 FQ(k-1) + r_4 \text{NO}_x(k) + r_5(\text{NO}_x(k) > 500) \Bigg) \tag{8.12}$$

where $r(k)$ is reward and T_s is the sampling time, in this application it is each cycle or 0.08 s at a constant engine speed. r_1 to r_5 is a positive value representing reward weights. Here the agent is penalized when the system produces more than 500 ppm NO_x. The DDPG is designed to maximize the reward function for minimization of the constraints used in this work they are multiplied by a negative value.

As shown in Figure 8.1, the actor has 9 FC layers with a layer size chosen to be 64 (except output layer, FC9, which has 3 units). The critic is set to 12 FC layers with the same layer size (64) as the actor in each layer (except output layer, FC9, which has 1 unit). To train both the DDPG and safe DDPG a minibatch size of 64 and smoothing factor of 0.001 are used. A noise model has been implemented with a variance of 5.66, 0.42, and 0.01 for $FQ(k)$, $SOI(k)$, and $VGT(k)$, respectively. It is common to have this variable multiply to root square of sampling time ($\sigma^2 \times \sqrt{T_s}$) and be between 1% and 10% of the action range. To force the RL to explore more, the variance decay rate is selected as a small value (10^{-6}). All of these values including number of layers, layer size, and noise model parameter are designed based on trial and error and monitoring reward value vs iteration (episode). Other hyperparameters such as minibatch size, smoothing factor, variance decay rate, and learning rate are set based on suggested values in the literature [135]. This schematic also shows the configuration of the ESM and the implementation of a safety filter to enforce the provided constraints. RL agents in this study are trained using the Matlab Reinforcement Learning Toolbox©.

8.1.4 Safe RL versus RL

In this study, two agents are developed: "RL" which is a traditional DDPG implementation and "safe RL" which is a DDPG with a safety filter to constrain the output. In both agents, the structure of actor and critic are kept the same. The episodic reward that the agent receives vs the episode number is shown in Figure 8.2. A 40 second simulation (500 engine cycles) with a randomly requested load, $T_{\text{ref}}(k)$, is provided to the agent and the load reference is changed for each episode. On an Intel Core i7-6700K based PC with 32.0 GB RAM running each episode takes an average of 346.84 s for the total ESM simulation and RL algorithm to update the networks. For the training of both agents the simulation is set to run for a maximum of 5000 episodes. The episodic reward versus episode is presented until the agents reaches to its maximum reward in Figure 8.2. At this point, the agent is considered an acceptable agent and saved. As shown in Figure 8.2, safe RL takes almost twice as long to reach the maximum reward compared to regular RL. Safe RL has more space which needs to be explored so it takes longer. Additionally, due to the use of a safety filter in safe RL, it reaches a larger reward which can be seen by comparing the agent at episode 1572 of RL and the agent at episode 3189 of safe RL.

The comparison between the selected agents for both safe RL and RL is presented in Figure 8.3. As shown, regardless of the training process, both agents are capable of maintaining load and minimizing NO_x emissions and fuel quantity. The RL also tries to obey the constraints as they are included in the reward function. According to the results presented, the safety filter does not provide any benefits since even without the safety filter, RL can learn the constraints as well as minimize the tracking error and NO_x.

Despite the fact that both the final selected agents perform well, it is interesting to compare them during the training of the agents. Figure 8.4 shows two agents of the RL at various stages of training. These agents are also presented in Figure 8.2.

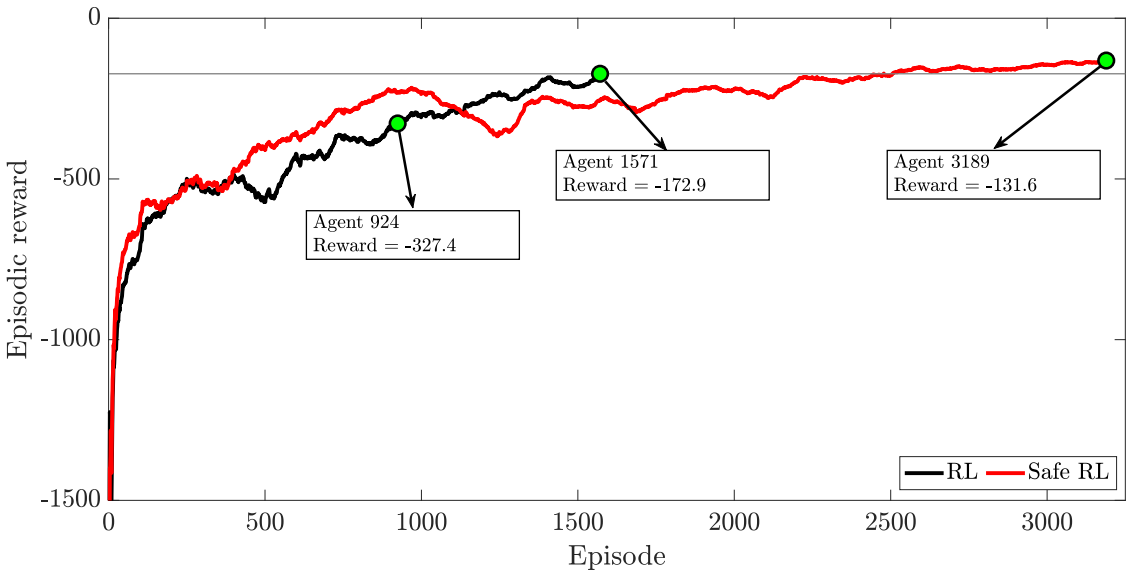


Figure 8.2: Episodic reward vs episode for safe RL and RL

One agent is in the middle of the training process at episode 924 and the other is the final agent that has reached the desired maximum reward of -150 at episode 1571. The oscillation observed from the controller during the early stages of training (episode 924) is due to the noise used to excite the system to allow for increased learning. Although agent 1571 is able to obey all constraints, there is a significant constraints violation in the NO_x output during training in earlier episodes. For online training the presence of safety filter is crucial in observing the constraints throughout training. However, if training is carried out in simulation, the use of a safety filter in not necessary, as the final agent is able to meet constraints while providing a stable output without the increased training time of using a safety filter.

8.2 Iterative Learning Controller (ILC)

A learning based controllers that has common elements with RL is an Iterative learning control (ILC). ILC has a simpler structure than RL as its control law update only includes two main filters and can be defined as

$$u_t(k) = Q(u_j(k-1)) + L(e_j(k-1)) \tag{8.13}$$

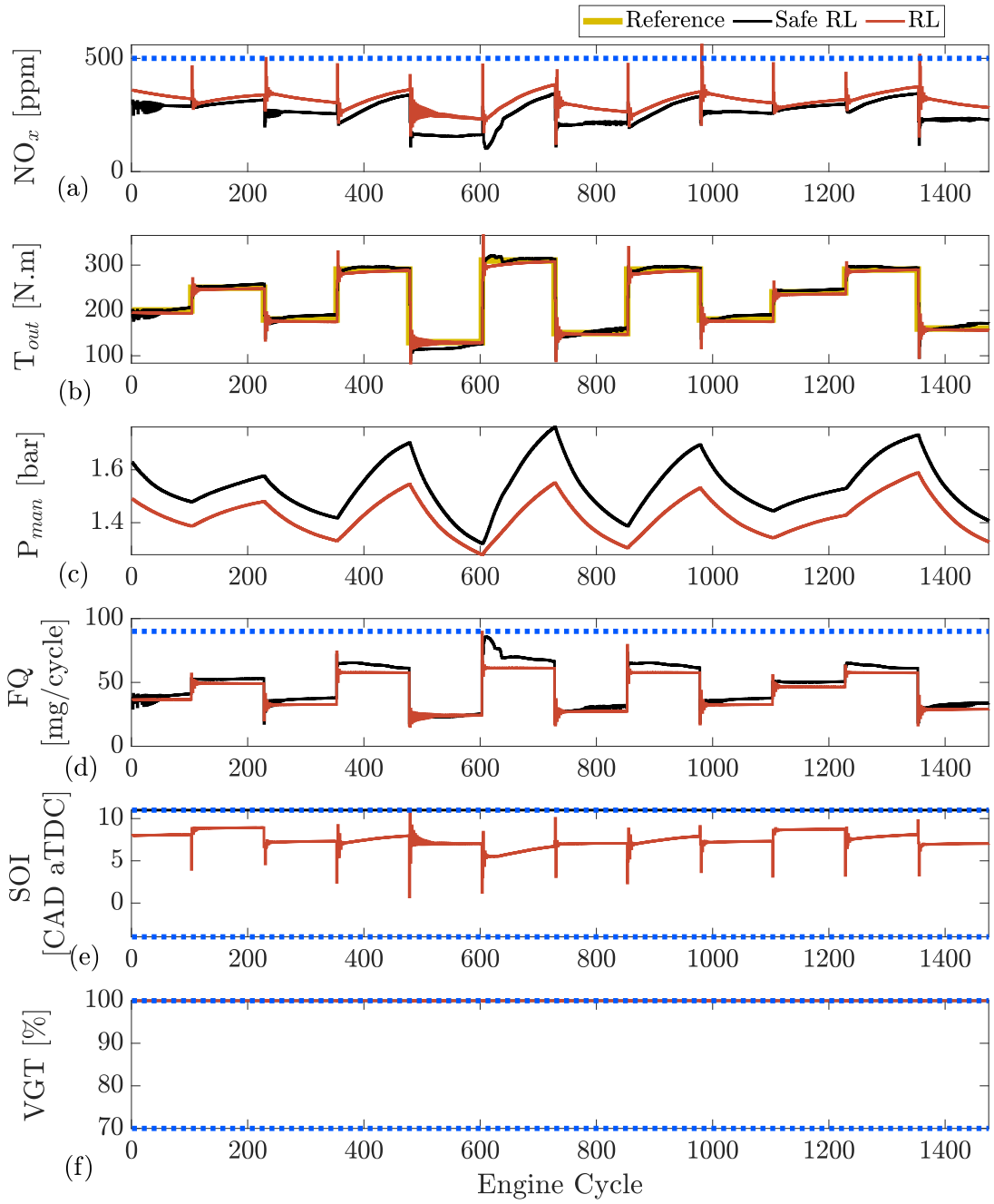


Figure 8.3: Safe RL vs RL: Comparison between two agent that reaches to maximum reward for safe RL (agent 3189) and RL (agent 1571)– a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate– The blue line represents the constraints boundary

where $L(e_j(k))$ is the L-filter or learning filter and $Q(u_j(k))$ is the Q-filter. In this equation, k represents the time interval. One of the simplest types of ILC is P-type

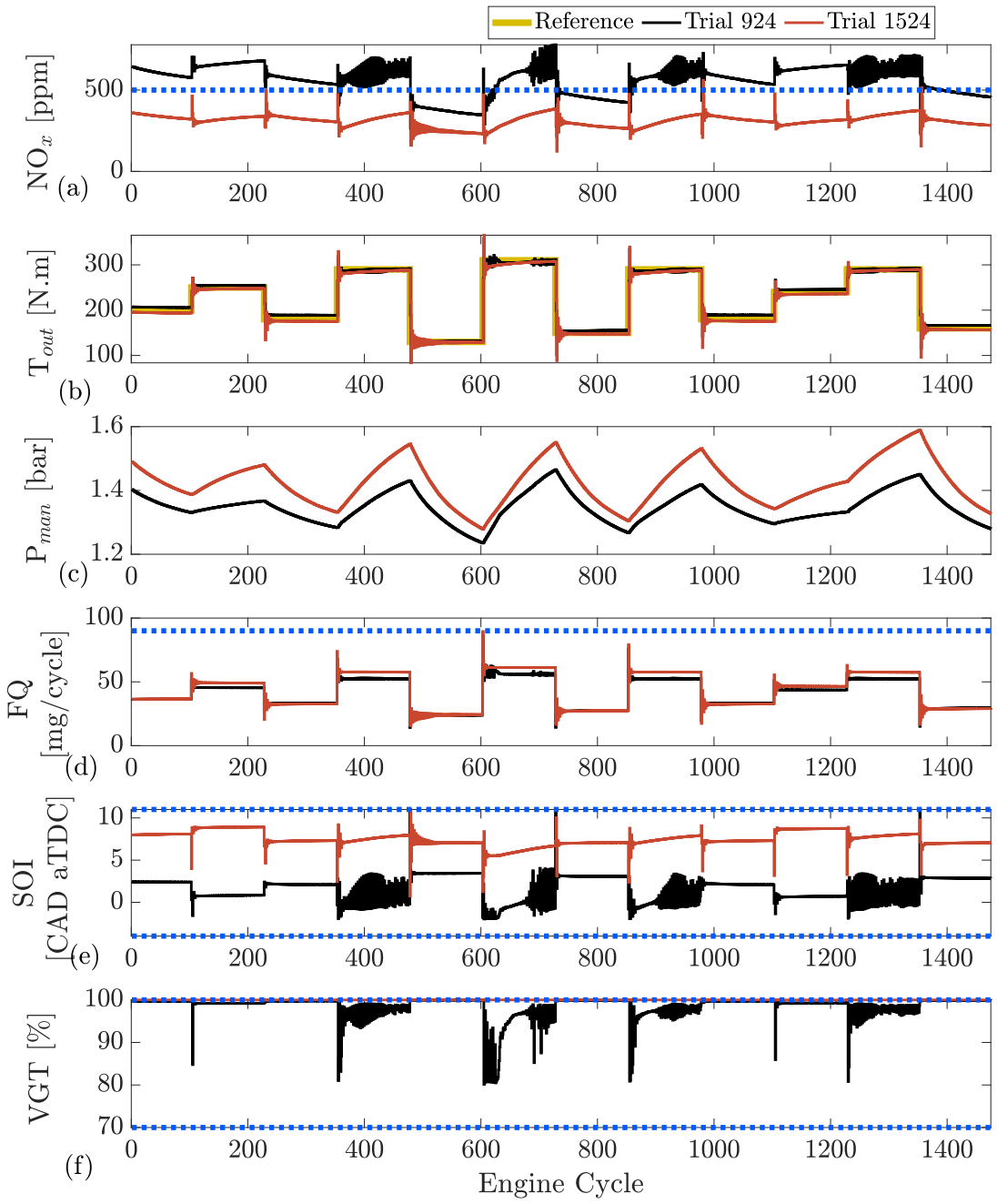


Figure 8.4: RL during training: comparison between agent in middle of training (agent 947) and agent that reaches to maximum reward (agent 1571)– a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate– The blue line represents the constraints boundary

ILC where the learning filter is $Pe_j(k)$ and P is a proportional gain and Q-filter is taken as unity. In a manner similar to the way safe RL enforces the output constraints,

a safety filter is added to ILC. Figure 8.5 shows a block diagram of the safe ILC. As shown, ILC learns from the previous error and control input to generate the current control action. The states, outputs, and inputs of ESM for ILC are

$$\begin{aligned} x(k) &= [\mathrm{NO}_x(k) \;\; P_{\mathrm{man}}(k) \;\; T_{\mathrm{out}}(k)]^T \\ y(k) &= [\mathrm{NO}_x(k) \;\; T_{\mathrm{out}}(k)]^T \\ u(k) &= [\mathrm{FQ}(k) \;\; \mathrm{SOI}(k) \;\; \mathrm{VGT}(k)]^T \end{aligned} \tag{8.14}$$

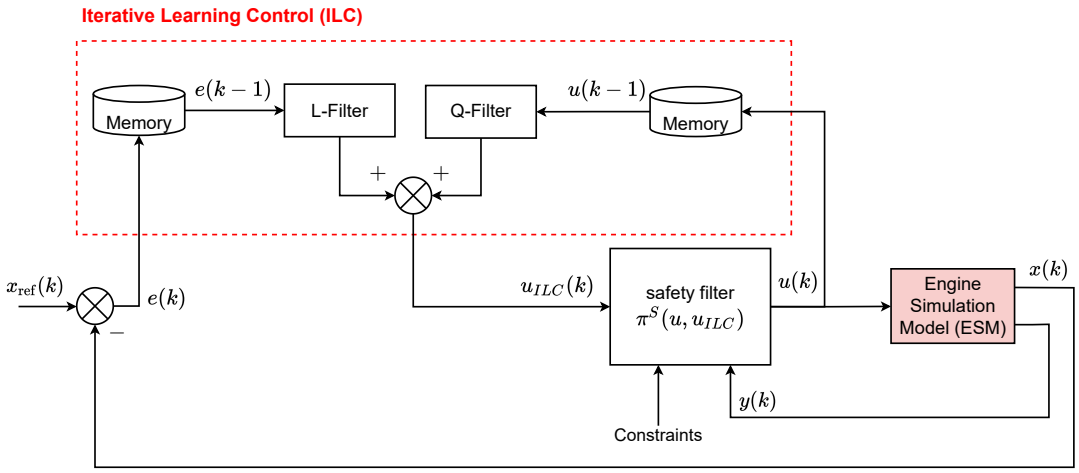


Figure 8.5: Safe iterative learning control block diagram

The implementation is slightly different compared to RL. Because of the nature of repetitive input requirements, a repetitive reference was implemented and the error between the actual state and the reference was provided to the ILC. The error can be defined as

$$e(k) = \begin{bmatrix} \mathrm{NO}_{x,\mathrm{ref}}(k) - \mathrm{NO}_x(k) \\ T_{\mathrm{out,\ ref}}(k) - T_{\mathrm{out}}(k) \\ P_{\mathrm{in,\ ref}}(k) - P_{\mathrm{ref}}(k) \end{bmatrix} \tag{8.15}$$

where $\mathrm{NO}_{x,\mathrm{ref}}(k)$, $T_{\mathrm{out,\ ref}}(k)$, and $P_{\mathrm{in,\ ref}}(k)$ are the respective reference values where each reference is repetitive with the same frequency. As the only tracking problem is the load output from the engine, this reference is the actual reference and the other two are implemented to satisfy the repetition requirements. For $\mathrm{NO_x}$, the reference value changes from 20 to 40 ppm for minimizing it. This variation is required to

artificially create a NO_x reference to minimize NO_x. Similarly, a reference for intake manifold pressure the set point is changed from 2 to 2.1 bar. All of the references are repeated every 300 cycles, i.e., for NO_x, the set point is 20 ppm for 150 cycles, then it changes to 40 ppm for 150 cycles, then it repeated. ILC and safe ILC training are shown in Figure 8.6. This figure presents 46 ILC iterations (a total of 13800 engine cycles). As shown after cycle 33, (9900 engine cycle), both the safe ILC and ILC learn to track desired references. As shown, the safe ILC is able to observe the output constraints; however, the ILC fails to remain within the constraints. Here, unlike the RL implementation, the presence of a safety filter for both the final stage and during training is necessary. As shown, the safe ILC tends to require late injections as SOI remains saturated at the upper limit. The existence of upper limit is necessary to avoid very late injection timing.

8.3 Results and Discussion

In this section the safe RL controller described in this section will be compared with the safe ILC and LSTM-NMPC (see Chapter 6). All these controllers are compared to a Cummins calibrated ECU modeled as ESM and denoted "benchmark (BM)".

First a comparison between the RL, LSTM-NMPC and BM controllers is presented in Fig 8.7. As shown, the safe RL is capable of accurately tracking the output torque with similar performance to the LSTM-NMPC. Both controllers outperform the BM feedforward production controller. Here the safe RL controller suffers from slightly increased overshoot when compared to the model based NMPC.

The controllers maintain NO_x emissions levels below the defined 500 ppm NO_x constraint. One clear trend in both the NMPC and RL is that the average NO_x value is significantly lower than the BM. This is expected as both controllers minimize NO_x and fuel consumption. One interesting trend is that the NO_x emissions of the safe RL model follow a similar trend to the BM but at a lower level. When comparing the RL to the LSTM-NMPC, overall the NO_x emissions are generally below the

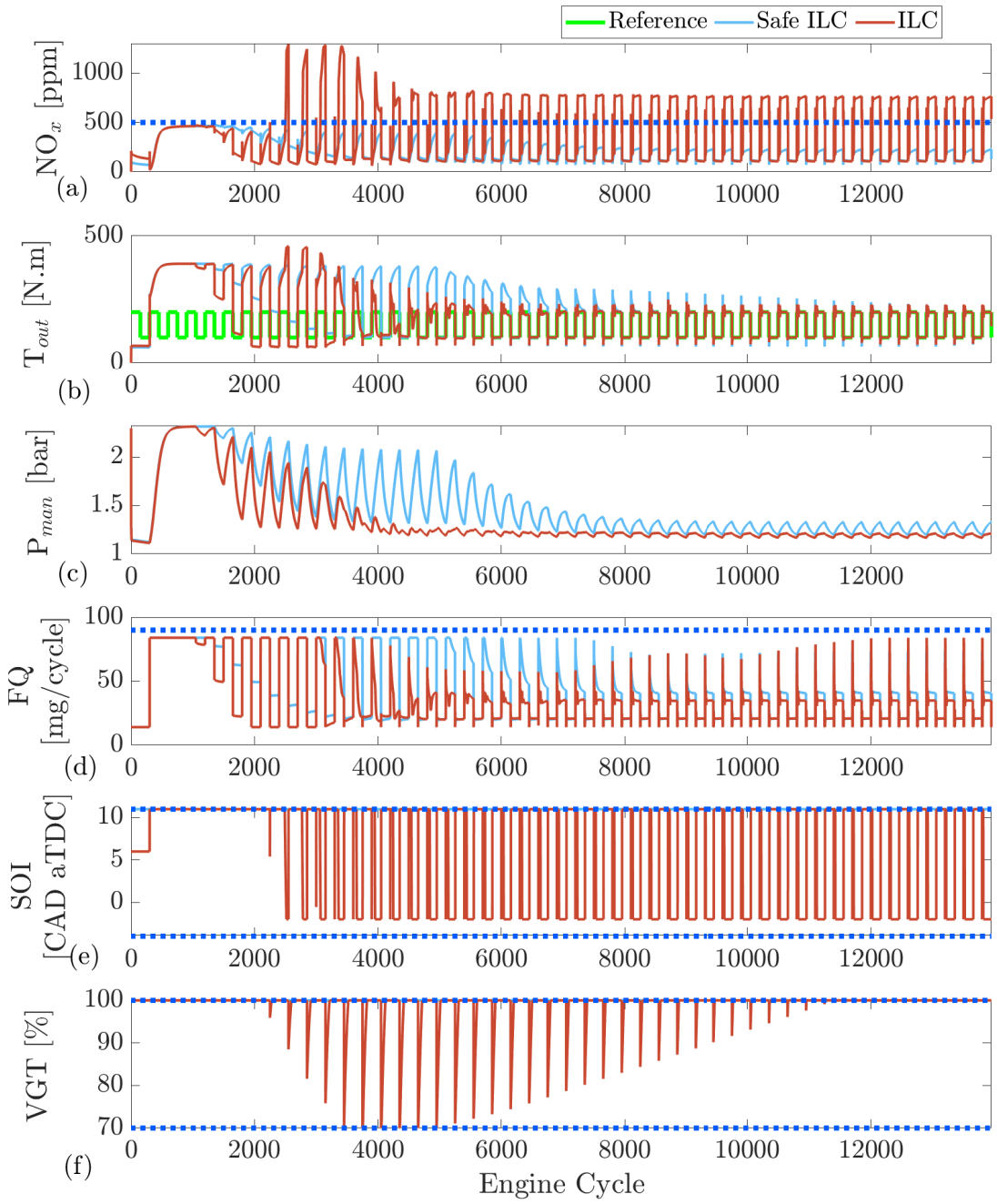


Figure 8.6: Simulation training ILC and safe ILC: The reference is repeated every 300 cycles and 46 ILC iterations are shown in this figure– a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate– The blue line represents the constraints boundary

LSTM-NMPC values and a significant reduction can be seen during the two cycles after a change in load where the LSTM-NMPC controller focuses on the load change

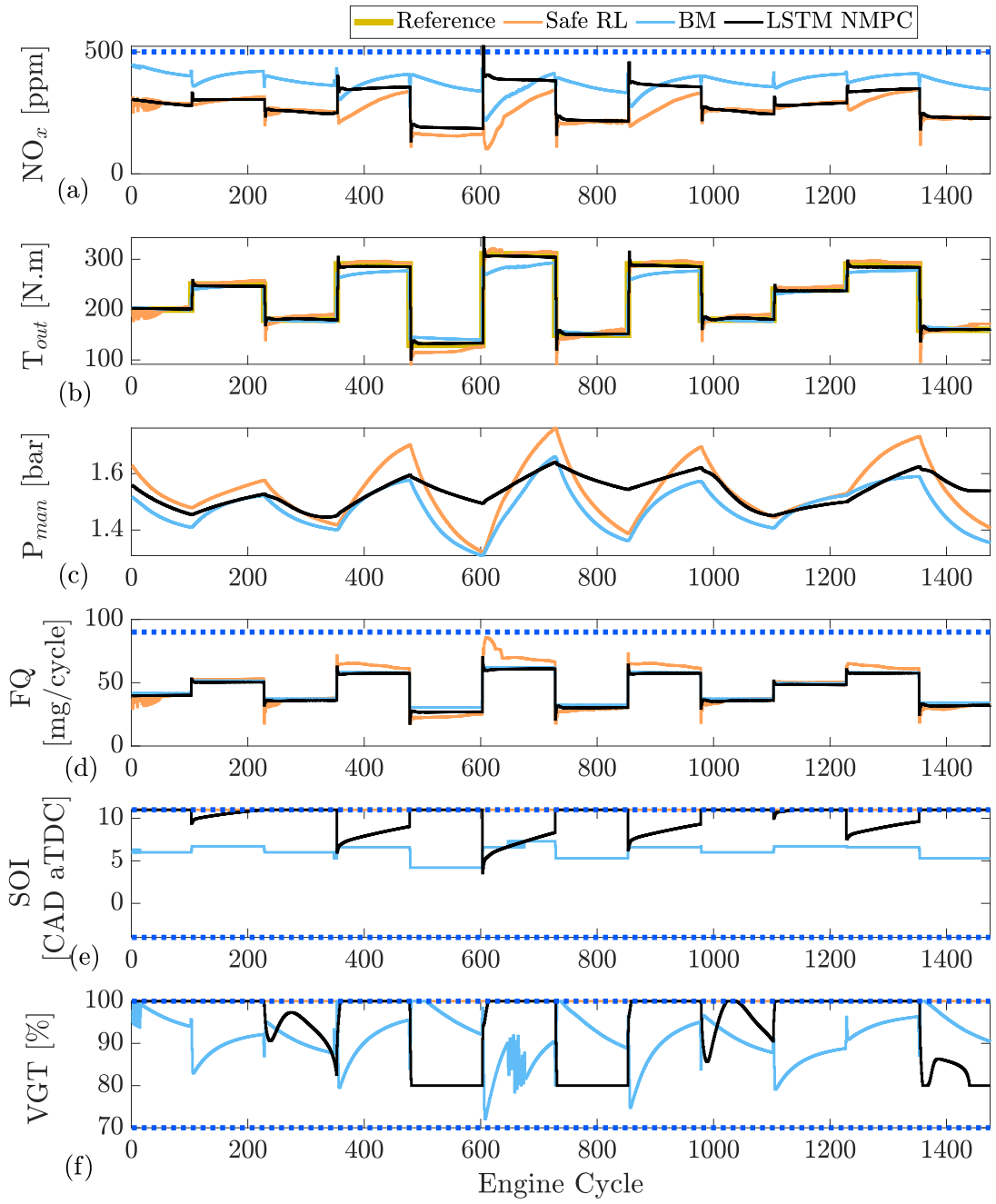


Figure 8.7: Safe Reinforcement Learning compared with LSTM-NMPC developed in Chapter 6 and the Cummins calibrated ECU which modeled in GT-power©– a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate– The blue line represents the constraints boundary

resulting in a spike of NO_x emissions.

The values of cumulative NO_x, fuel quantity, and execution time are compared in

Table 8.1: Comparison between Deep RL, Benchmark (BM), and Nonlinear Model Predictive Control developed in Chapter 6

	Cumulative NO_x [ppm]	Average NO_x [ppm]	Load error [%]	Cumulative FQ [g]	Average FQ [mg]	Execution time [ms]*
BM	5.6×10^5	376.8	3.95	67.9	46.0	-
Safe RL	3.8×10^5	260.4	3.85	69.1	46.8	4.5
LSTM-NMPC	4.3×10^5	290.2	1.90	65.6	44.4	12.20**

* per engine cycle of simulation

** average acados execution time

Table 8.1. To determine the execution time of the NMPC, an open-source package acados [202, 203] is used for the implementation. For the execution time, the idea is to examine the feasibility for real-time implementation and thus the deployment time of RL is considered only and the training time is excluded. In this study, RL has almost 3 times faster execution time than the online NMPC optimization.

As shown, RL has significantly lower NO_x in comparison to both the BM and the NMPC. The drawback of the RL controllers is slightly increased load error and fuel quantity. However, the improvement in NO_x reduction using a safe RL controller is more significant than the loss in load error and fuel quantity.

The deep RL controller performs comparably to the model-based NMPC. However, it is also of interest to compare with another learning-based control strategy such as the ILC. The developed RL controller is compared to ILC and the BM in Figure 8.8. As shown, both controllers are capable of tracking the desired output torque with similar performance to the BM. The ILC tracks the reference more closely than the deep RL control. The ILC tracking performance is almost perfect with very little overshoot which is one of its benefits of the ILC since the repetitive input requirements allow it to ILC learn by repetition. The RL controller suffers from slight torque overshoot but its performance is still acceptable.

All of the controllers tested were able to remain below the defined 500 ppm NO_x constraint. The NO_x reduction using the ILC is slightly better than the RL

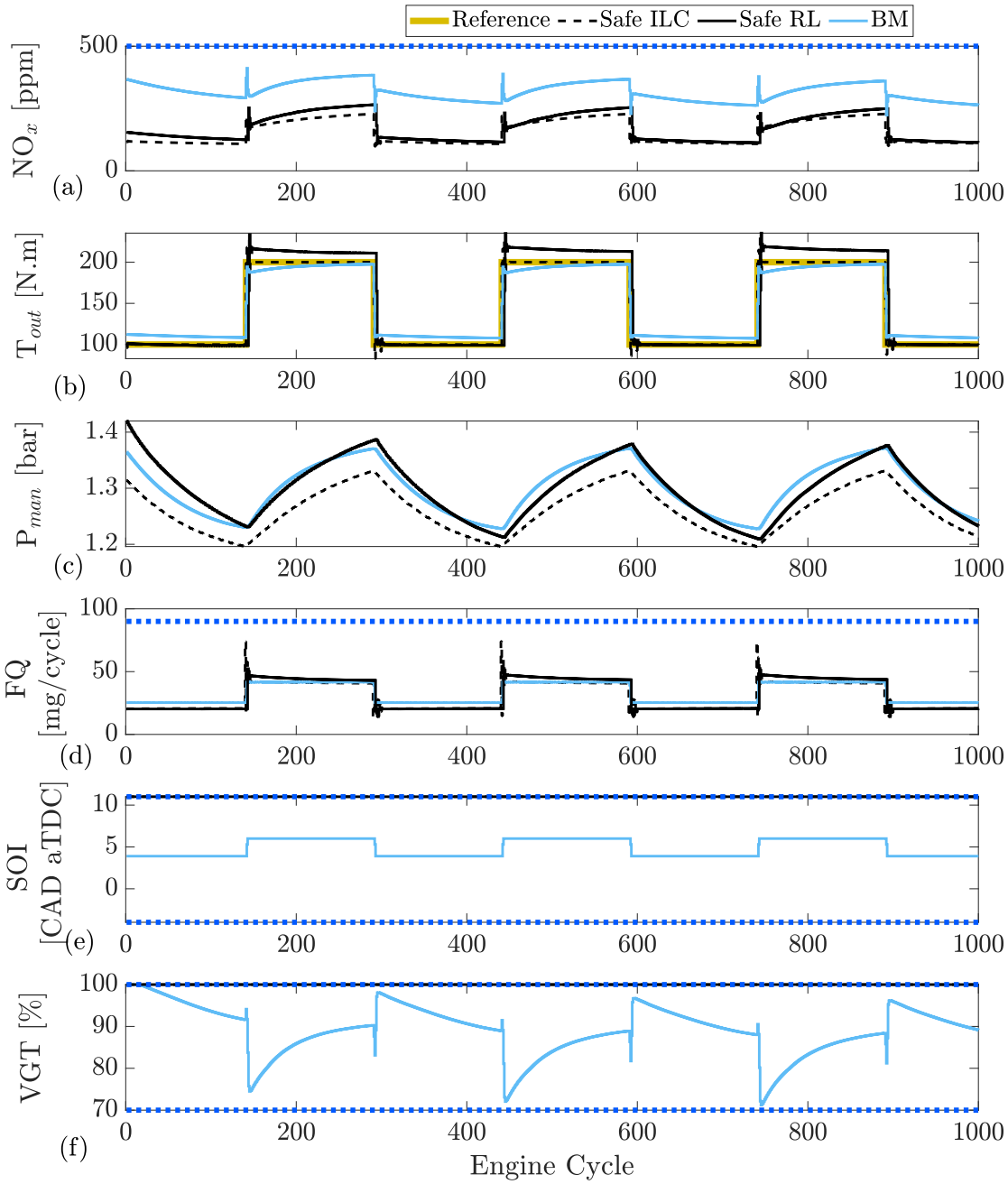


Figure 8.8: Safe Reinforcement Learning compared with safe ILC and Cummins calibrated ECU which modeled in GT-power©– a) engine-out NO_x, b) intake manifold pressure (P_{man}), c) engine output torque (T_{out}), d) fuel quantity (FQ), e) Start of injection (SOI), f) Variable Geometry Turbine (VGT) rate– Blue line represent constraints boundary

controller and both controllers significantly outperform the BM. When comparing the models in terms of fuel quantity used both the ILC and RL models use slightly

Table 8.2: Comparison between Deep RL, Bechmark, and ILC

	Cumulative NO_x [ppm]	Average NO_x [ppm]	Load error [%]	Cumulative FQ [g]	Average FQ [mg]	Execution time [ms]*
BM	3.17×10^5	317.0	6.65	32.6	32.6	-
Safe RL	1.71×10^5	171.0	5.23	31.4	31.4	4.5
Safe ILC	1.55×10^5	155.4	0.51	29.9	29.9	0.08

* per engine cycle of simulation

less fuel than the BM model. However, there is very little difference in the results between the RL and ILC, as can be seen in Figure 8.8.

The controller performance results, the value of cumulative NO_x, fuel quantity, and execution time are summarized in Table 8.2. As shown, both RL and ILC are able to reduce NO_x emissions significantly compared to the BM. Although a comparison between the RL and ILC fuel quantity (FQ) showed a better FQ for ILC. The execution time of ILC is two orders of magnitude faster than the RL with significantly better load tracking performance. The fast learning time of the ILC indicates that it could be used for real-time online training. However, its the main draw-back is that it requires a repetitive reference or disturbance. This condition may be possible for stationary engines; however, it is not feasible in most of the ICE applications especially for on-road engines. Therefore the slight performance loss of the RL compared to the ILC provides the flexibility to remove the requirement of a repetitive reference or disturbance.

The summary of these comparisons is presented in Table 8.3 and Table 8.4.

8.4 Summary of chapter

In this chapter, a deep RL-based controller was developed to minimize NO_x emissions and fuel consumption of a diesel engine while maintaining a constant output torque. Using the ESM two learning-based controllers, DDPG and ILC, are investigated in simulation. A safety filter was applied to DDPG and ILC to enforce the output

Table 8.3: Summary of comparison for developed controllers

Method	Constraints Enforcement	Execution Time	Model Requirement	Limitation
RL	-	50x	-	Time-consuming in training
ILC	-	1x	-	Repetitive reference requirement
Safe RL	✓	50x	✓	Time-consuming in training
Safe ILC	✓	1x	✓	Repetitive reference requirement
LSTM-NMPC	✓	150x	✓	High accuracy model requirement

Table 8.4: Summary of comparison for developed controllers– controller performance compared to benchmark. Range is used in safe RL as it is compared with BM using both repetitive and random reference twice with different reference.

Method	Load Tracking Error	Average Fuel Consumption Reduction*	Average NOx Reduction*
Safe RL	$3-5\%$	$0-4\%$	$30-45\%$
Safe ILC	$\leq 1\%$	$\geq 8\%$	$\geq 50\%$
LSTM-NMPC	$\leq 2\%$	$\geq 3\%$	$\geq 22\%$

* Reduction calculated relative to BM

constraints.

Comparison between safe RL and RL showed that both with similar performance once training was completed. The RL is able to learn to enforce the output constraints since they are part of the reward function. However, during training, there is a large violation of the constraints suggesting that using safe learning is crucial when working with real engineering system in real-time learning. For the ILC, the safety filter implementation shows a significant effect during both training and final controller performance. This suggests that ILC requires a safety filter to enforce output constraints.

The safe RL was then compared to safe ILC and LSTM-NMPC to evaluate which controller had better performance. Although ILC has a 4 percent better torque tracking and 16 ppm lower average NO_x emissions, than the RL based controller, it is require repetitive references and disturbances. Comparison with LSTM-NMPC

shows that the deep RL is capable of further reducing the average NO_x emissions by 30 ppm compared to LSTM-NMPC at a cost of 2% higher load error and 4.5% average fuel consumption increase. These performance differences between the models are very small. However, the LSTM-NMPC is a model-based controller which requires a relatively accurate model (black-box or white-box (physical) model) for the online MPC optimization. In contrast, the Safe RL learns directly from experiments and required a simple linear model for safety filter.

PART V: Conclusions

Chapter 9

Conclusions

This thesis studied the application of Machine Learning (ML) to engine modeling, control, and emission reduction. First the application of ML to emission and performance prediction was presented in Chapters 3 and 4 by modeling engine-out NO_x, PM, and load. Chapters 5, 6, and 7 presented integration of ML and Model Predictive Control (MPC) by using ML in the transient modeling needed in MPC and imitation of MPC. Finally, Reinforcement Learning (RL) and Iterative Learning Control (ILC) were presented in Chapter 8 as pure learning-based controls for emission and fuel consumption minimization. The main conclusions of this study and future related work are presented next.

9.1 Machine Learning in Emission Prediction

This thesis addressed two main challenges identified for the literature in emission prediction. First, to address the complexity of black-box models, a Model Order Reduction (MOR) algorithm was developed using a support vector machine (SVM) approach to predict the steady-state NO_x and Break Mean Effective Pressure (BMEP) of a medium-duty diesel engine. The MOR algorithm was used to produce two models: a complex model with high-accuracy called a High-Order Model (HOM), and a Low-Order Model (LOM) with acceptable accuracy. Unnecessary features were removed based on the SVM-based MOR algorithm resulting in an enhanced performance of

the HOM for both NO_x and BMEP while the HOM complexity decreased by 27.9% with respect to the Full-Order Model (FOM). The LOM model had an acceptable accuracy with a squared correlation coefficient of 0.94 for NO_x and 0.996 for BMEP while having 77.9% and 69.4% fewer features with respect to the FOM and HOM, respectively. This algorithm successfully reduced the order of the ML-based data-driven model without significant loss in the prediction accuracy of prediction. These results were published in [3, 4].

Secondly, black-box and gray-box modeling for engine PM or soot emissions was developed and compared to physics-based models which struggle with their prediction accuracy due to the complexity in Particle Matter (PM) emissions formation. Gray-box and black-box soot emissions models were developed using eight different machine learning methods. Based on the Least Absolute Shrinkage and Selection Operator (LASSO) feature selection method and physical insight, five different feature sets were tested for black-box and gray-box models. To analyze the results, the K-means clustering algorithm was applied in two steps to categorize the models according to their performance. Real-time control is only feasible with black-box methods since the physics-based model is too computationally expensive for use in current ECUs. Based on the results, the Gaussian Process Regression (GPR) method with LASSO as the feature selection method was the most reliable ML method/feature set. Gray-box models, although more complex than black-box models, are useful as a virtual engine to conduct simulation tests for development and calibration purposes, reducing the need for costly experiments. Among gray-box models, the SVM-based ML method combined with the use of LASSO and physical insight for feature selection provided the best performance. In most cases, gray-box models outperform their black-box counterparts in terms of accuracy. This work was published in [5].

9.2 Integration of Machine Learning and Model Predictive Control

The two main challenges of MPC that have been identified are: 1) an accurate model is required and 2) computation requirement time must be reduced to allow for real-time implementation. Two main methods using ML have been presented to address these two challenges. In the first method, ML was used to identify a model for implementation in model predictive control optimization problems. In the second method, ML was used as a replacement for the controller, where the ML controller learned the optimal control action by imitating or mimicking the behavior of the model predictive controller. The ML replaced the MPC controller.

In Chapters 5 and 6 two MPC controllers using an SVM-based Linear Parameter Varying (SVM-LPV) method and Long-Short Term Memory (LSTM) models were developed and compared with linear MPC and the Cummins calibrated benchmark ECU. The controllers in Chapters 5 and 6 were compared in simulation using the Engine Simulation Model (ESM) developed in Chapter 2.

To develop the LPV-MPC, an SVM-LPV model was first developed to design an LPV-MPC. Then, the LPV-MPC was implemented and the controller input and output data were collected from the MPC and used to train a deep neural network. Replacing the full online MPC with a deep network reduced the computational time of the MPC. After testing the imitative LPV-MPC controller at two different engine speeds, the imitative controller performed similarly to the full online optimization of LPV-MPC performance but with a significant reduction in the processing time. In addition, the MPC and imitative models showed significant improvements in NO_x emissions and a reduction in fuel consumption while providing similar load following capabilities as the feed-forward production controller. Both the LPV-MPC and imitative controller were able to reduce NO_x emissions by 18-70% while reducing fuel consumption by 1-10% compared to the Cummins production controller. The im-

itative controller required 1/50 of the computational time compared to online MPC optimization. This work was published in [6].

For the LSTM-NMPC, a deep neural network with an LSTM layer was designed to predict diesel engine performance and emissions. The NMPC was designed based on this network by augmenting hidden and cell states. This model has an acceptable prediction accuracy for inputs not yet seen (test data). This accuracy is expected as the LSTM is capable of a more generalizable prediction since it uses hidden and cell states. Similar to the imitative LPV-MPC, an imitative LSTM-NMPC controller has been developed. All of the controllers produce significant NO_x reduction, especially at lower engine speeds with respect to the benchmark feedforward production controller. The NO_x reduction for 1500 and 1200 rpm for the NMPC was 23.0% and 65.8%. The imitative controller successfully cloned the NMPC behavior resulting in a NO_x reduction of 21.1% at 1500 rpm. At 1200 rpm the reduction was 63.4% when compared to the BM. The imitative controller performed similarly to the online MPC by learning from the MPC experiment but it requires much lower computational time. The computational time for the imitative controller was a factor of 100 lower than the online optimized LSTM-NMPC. In addition, a benchmark comparison of NMPC execution time was performed using an open-source `acados` (QP solver `HPIPM`) package, state-of-the-art commercial `FORCES PRO` solver, and standard Matlab© `fmincon` solver. The `acados` program with `HPIPM` provided the fastest solve time among the solvers tested with an average runtime of 12.20 ms and a maximum runtime of 31.56 ms for 1500 rpm. This value is much faster than the average runtimes of `fmincon` that required 786.02 ms. `FORCES PRO` was also tested and while it showed an improvement in runtime over the `fmincon`, but it was significantly slower than the `acados` implementation. This work was submitted in [7].

The comparison between the LPV-MPC and LSTM-NMPC showed that the LSTM-NMPC in simulation has more emission reduction with better load tracking. This knowledge was used to choose the LSTM-NMPC for experimental implementa-

tion. The experimental engine control was implemented in a real-time system using dSPACE MictroAutoBox II rapid prototyping as described in Chapter 7. To do this, a new deep neural network was developed based on real-time data. Due to the accessibility to fast PM sensor and utilizing an FPGA, PM emissions, as well as the Indicated Mean Effective Pressure (IMEP) tracking and Maximum Pressure Rise Rate (MPRR) constraints were added to the model and controller. The main goal was minimized emissions and fuel consumption while considering constrains and tracking of IMEP. The constraint for MPRR was to ensure combustion stability and other inputs are constrained to ensure engine safety. In the real-time implementation, the `acados C` package was cross compiled for the MictroAutoBox. For this embedded program, the average run time of NMPC is 62.5 ms, which was feasible for real-time implementation with an engine speed of less than 1850 rpm. The controller achieved both reductions in PM and NO_x demonstrating the advantage of using systematic optimization to solve the NO_x-PM trade-off. The proposed controller can reduce average NO_x , PM, and fuel consumption up to 22.4%, 43.6%, and 14.9% while improving thermal efficiency up to 4.7%. The controller was also tested for transient changes in load. The average tracking error to a step reference of load was 0.26 bar with an RMSE of 0.61, while the average tracking error for smooth ($\sim$ 1 Hz bandwidth) load reference changes was 0.16 bar with an RMSE of 0.20. To determine the controller's robustness for operation outside the training range of the model, the controller was evaluated at speeds ranging from 1200 to 1800 rpm. The experimental findings demonstrate that good tracking and disturbance rejection were achieved for the LSTM-NMPC. Further, a systematic way to develop the NMPC for a complex nonlinear system with a fast sampling frequency was demonstrated by using ML with MPC. This work was submitted in [8].

9.3 Machine Learning in Learning-based Controller

A deep Reinforcement Learning (RL) based controller was developed in simulation to minimize NO_x emissions and fuel consumption of a diesel engine while maintaining a constant output torque. Using the ESM two learning-based controllers were investigated in simulation. The first was the RL controller utilizing a Deep Deterministic Policy Gradient (DDPG) which is implemented using a deep network for both actor and critic. This was extended with the addition of a safety filter. This safety filter was added to the manipulated control action and used to enforce output constraints. The second learning-based controller is an Iterative Learning Control (ILC) which is another well-known control strategy. The same safety filter was applied to ILC to enforce the output constraints.

Each learning-based controller with a safety filter was compared with its standard version to better understand the effect of adding a safety filter. It was found that for deep RL, both the safe and standard controller resulted in almost the same controller performance once training was completed. Even the standard RL was able to learn to enforce the output constraints. However, during training, there was a large violation of the constraints suggesting that the use of safe learning is crucial when working with engineering systems in real-time learning. For the ILC, the safety filter implementation showed a significant effect during both training and final controller performance. This suggested that the ILC requires a safety filter to enforce output constraints.

The safe RL was then compared to the safe ILC to evaluate which controller has better performance, as they both share a similar learning based controller approach. This comparison showed that the real-time turnaround time of the ILC was two orders of magnitude faster than the RL and the ILC had the ability to take advantage of online learning. Although ILC had a 4 percent better torque tracking and 16 ppm

lower average NO_x emissions than the RL based controller, it does have the limitation of requiring repetitive references and disturbances. This makes the ILC applicable only to certain ICE applications such as power generation which utilize a repetitive set-point. However, since many ICE applications are non-repetitive, for example on-road vehicle applications, the ILC is not a feasible option and the safe RL is a better choice.

To compare the safe RL to a state-of-the-art controller a comparison was made to the LSTM-NMPC. This comparison showed that the deep RL is capable of reducing the average NO_x emissions by 30 ppm more than the LSTM-NMPC at a cost of a 2% higher load error and 4.5% average fuel consumption increase. These performance differences between the models are very small. However, the LSTM-NMPC is a model-based controller which requires a relatively accurate model for the online MPC optimization. In contrast, the RL learns directly from experimental data.

9.4 Future Work

Possible ways to extend this work in the future include:

- Real-time implementation and online training of the RL. This work was the first to propose a method to investigate the possibility of RL implementation in engine control and emission reduction. Implementing these methods, perhaps using TensorFlow API called from MicroAutoBox, is one possible method to achieve RL in real-time.

- Although imitation learning has been studied in simulation, it seems promising for real-time implementation. There is no guarantee of safety and enforcing constraints, so other add-on methods such as using RL as an add-on controller, or using a safe filter with imitation or online updating an imitating controller require future investigation.

- One limitation of this work is related to the experimental setup. This engine

has no Exhaust Gas Recirculation (EGR), and the turbine controller is mechanical. Adding variable boost pressure using either a supercharger or a Variable Geometric Turbine (VGT) as well as adding an EGR system will allow more flexibility for control and make the work applicable to modern diesel engines. This resulting controller could further reduce emission and fuel consumption while maintaining constraints and safety.

Bibliography

[1] 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.

[2] A. Norouzi, H. Heidarifar, M. Shahbakhti, C. R. Koch, and H. Borhan, "Machine learning and model predictive control integration in automotive control system applications: A review and future directions," *Engineering Applications of Artificial Intelligence (Submitted on June 24, 2022)*, 2022.

[3] A. Norouzi, M. Aliramezani, and C. R. Koch, "A correlation-based model order reduction approach for a diesel engine nox and brake mean effective pressure dynamic model using machine learning," *International Journal of Engine Research*, vol. 22, no. 8, pp. 2654–2672, 2021. eprint: https://doi.org/10.1177/1468087420936949.

[4] A. Norouzi, D. Gordon, M. Aliramezani, and C. R. Koch, "Machine Learning-based Diesel Engine-Out NOx Reduction Using a plug-in PD-type Iterative Learning Control," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*, 2020, pp. 450–455.

[5] S. Shahpouri, A. Norouzi, C. Hayduk, R. Rezaei, M. Shahbakhti, and C. R. Koch, "Hybrid machine learning approaches and a systematic model selection process for predicting soot emissions in compression ignition engines," *Energies*, vol. 14, no. 23, 2021.

[6] A. Norouzi, S. Shahpouri, D. Gordon, A. Winkler, E. Nuss, D. Abel, J. Andert, M. Shahbakhti, and C. R. Koch, "Machine learning integrated with model predictive control for imitative optimal control of compression ignition engines," *IFAC-PapersOnLine*, 2022, 10th IFAC Symposium on Advances in Automotive Control AAC 2022 (In Press).

[7] A. Norouzi, S. Shahpouri, D. Gordon, A. Winkler, E. Nuss, D. Abel, J. Andert, M. Shahbakhti, and C. R. Koch, "Integration of Deep Learning and Nonlinear Model Predictive Control in Emission reduction of Compression Ignition Combustion Engines: A Simulation Study," *Control Engineering Practice (In Press)*, 2022.

[8] A. Norouzi, D. Gordon, A. Winkler, J. McNally, E. Nuss, D. Abel, J. Andert, M. Shahbakhti, and C. R. Koch, "Experimental implementation of deep neural network-based nonlinear model predictive control in diesel engine emission control," *Transactions on Control Systems Technology (Submitted on May 26, 2022)*, 2022.

[9] A. Norouzi, S. Shahpouri, D. Gordon, M. Shahbakhti, and C. R. Koch, "Safe Deep Reinforcement Learning in Diesel Engine Emission Control," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering (Submitted on April 3, 2022)*, 2022.

[10] D. G. Kessel, "Global warming—facts, assessment, countermeasures," *Journal of Petroleum Science and Engineering*, vol. 26, no. 1-4, pp. 157–168, 2000.

[11] I. Yusri, A. A. Majeed, R. Mamat, M. Ghazali, O. I. Awad, and W. Azmi, "A review on the application of response surface method and artificial neural network in engine performance and exhaust emissions characteristics in alternative fuel," *Renewable and Sustainable Energy Reviews*, vol. 90, pp. 665 –686, 2018.

[12] B. E. Economics, "BP energy outlook: 2020 Edition," *BP PLC: London, UK*, 2020.

[13] A. Vuorinen, *Planning of Optimal Power Systems*. Vammalan Kirjapaino Oy, Vammala, Finland, 2009.

[14] S. Davis and R. G. Boundy, "Transportation Energy Data Book: Edition 39," Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), Tech. Rep., 2021.

[15] I. A. Resitoglu, K. Altinisik, and A. Keskin, "The pollutant emissions from diesel-engine vehicles and exhaust aftertreatment systems," *Clean Technologies and Environmental Policy*, vol. 17, no. 1, pp. 15–27, 2015.

[16] I. Blanco-Rodriguez, *Modelling and observation of exhaust gas concentrations for diesel engine control*. Springer, 2014.

[17] S. H. Cadle, P. A. Mulawa, E. C. Hunsanger, K. Nelson, R. A. Ragazzi, R. Barrett, G. L. Gallagher, D. R. Lawson, K. T. Knapp, and R. Snow, "Composition of Light-Duty Motor Vehicle Exhaust Particulate Matter in the Denver, Colorado Area," *Environmental Science & Technology*, vol. 33, no. 14, pp. 2328–2339, 1999.

[18] Y.-C. Chang, W.-J. Lee, T. S. Wu, C.-Y. Wu, and S.-J. Chen, "Use of water containing acetone–butanol–ethanol for NOx-PM (nitrogen oxide-particulate matter) trade-off in the diesel engine fueled with biodiesel," *Energy*, vol. 64, pp. 678–687, 2014.

[19] C. Thiel, J. Schmidt, A. Van Zyl, and E. Schmid, "Cost and well-to-wheel implications of the vehicle fleet CO2 emission regulation in the european union," *Transportation Research Part A: policy and practice*, vol. 63, pp. 25–42, 2014.

[20] L. D. Prockop and R. I. Chichkova, "Carbon monoxide intoxication: An updated review," *Journal of the neurological sciences*, vol. 262, no. 1-2, pp. 122–130, 2007.

[21] X. Gu, Z. Huang, J. Cai, J. Gong, X. Wu, and C.-f. Lee, "Emission characteristics of a spark-ignition engine fuelled with gasoline-n-butanol blends in combination with EGR," *Fuel*, vol. 93, pp. 611–617, 2012.

[22] A. Dewangan, A. Mallick, A. K. Yadav, and R. Kumar, "Combustion-generated pollutions and strategy for its control in CI engines: A review," *Materials Today: Proceedings*, vol. 21, pp. 1728–1733, 2020, International Conference on Mechanical and Energy Technologies.

[23] B. Heid, R. Hensley, and S. Knupfer, "What's sparking electric-vehicle adoption in the truck industry?," 2017.

[24] EuroVI, "Commission regulation (EU) 2016/646 of 20 april 2016 amending regulation (EC) NO692/2008 as regards emissions from light passenger and commercial vehicles (Euro 6)," in *Euro 6 regulation*, 2016.

[25] A. G. Konstandopoulos, M. Kostoglou, E. Skaperdas, E. Papaioannou, D. Zarvalis, and E. Kladopoulou, "Fundamental studies of diesel particulate filters: Transient loading, regeneration and aging," *SAE transactions*, pp. 683–705, 2000.

[26] M. K. Khair and W. A. Majewski, "Diesel emissions and their control," SAE Technical Paper, Tech. Rep., 2006.

[27] M. Zheng, G. T. Reader, and J. G. Hawley, "Diesel engine exhaust gas recirculation—-a review on advanced and novel concepts," *Energy conversion and management*, vol. 45, no. 6, pp. 883–900, 2004.

[28] A. Abedi, "The Effect of an Axial Catalyst Distribution on the Performance of a Diesel Oxidation Catalyst and Inverse Hysteresis Phenomena during CO and C3H6 Oxidation," 2012.

[29] A Aksikas, I Aksikas, R. Hayes, and J. Forbes, "Model-based optimal boundary control of selective catalytic reduction in diesel-powered vehicles," *Journal of Process Control*, vol. 71, pp. 63–74, 2018.

[30] K. Mollenhauer and H. Tschoke, *Handbook of diesel engines*. Springer Berlin, 2010, vol. 1.

[31] F. Tschanz, A. Amstutz, C. H. Onder, and L. Guzzella, "Feedback control of particulate matter and nitrogen oxide emissions in diesel engines," *Control engineering practice*, vol. 21, no. 12, pp. 1809–1820, 2013.

[32] L. Guzzella and C. Onder, *Introduction to modeling and control of internal combustion engine systems*. Springer Science & Business Media, 2009.

[33] R. Isermann, "Engine modeling and control," *Berlin: Springers Berlin Heidelberg*, vol. 1017, 2014.

[34] J. D. López, J. J. Espinosa, and J. R. Agudelo, "LQR control for speed and torque of internal combustion engines," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 2230–2235, 2011, 18th IFAC World Congress.

[35] R. Pfeiffer, G. Haraldsson, J.-O. Olsson, P. TunestAl, R. Johansson, and B. Johansson, "System identification and LQG control of variable-compression HCCI engine dynamics," in *Proceedings of the 2004 IEEE International Conference on Control Applications, 2004.*, IEEE, vol. 2, 2004, pp. 1442–1447.

[36] A. Norouzi, K. Ebrahimi, and C. R. Koch, "Integral discrete-time sliding mode control of homogeneous charge compression ignition (hcci) engine load and combustion timing," *IFAC-PapersOnLine*, vol. 52, no. 5, pp. 153–158, 2019, 9th IFAC Symposium on Advances in Automotive Control AAC 2019.

[37] M. R. Amini, M. Shahbakhti, S. Pan, and J. K. Hedrick, "Discrete adaptive second order sliding mode controller design with application to automotive control systems with model uncertainties," in *2017 American Control Conference (ACC 2017)*, IEEE, 2017, pp. 4766–4771.

[38] J. S. Souder and J. K. Hedrick, "Adaptive sliding mode control of air–fuel ratio in internal combustion engines," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 14, no. 6, pp. 525–541, 2004.

[39] B. K. Irdmousa, S. Z. Rizvi, J. M. Velni, J. Naber, and M. Shahbakhti, "Data-driven modeling and predictive control of combustion phasing for RCCI engines," *American Control Conference (ACC 2019)*, pp. 1–6, 2019.

[40] L. A. Basina, B. K. Irdmousa, J. M. Velni, H. Borhan, J. D. Naber, and M. Shahbakhti, "Data-driven modeling and predictive control of maximum pressure rise rate in RCCI engines," in *IEEE Conference on Control Technology and Applications (CCTA 2020)*, IEEE, 2020, pp. 94–99.

[41] W. H. Kwon and S. H. Han, *Receding horizon control: model predictive control for state models*. Springer Science & Business Media, 2006.

[42] J. Powell, "A review of IC engine models for control system design," *IFAC Proceedings Volumes*, vol. 20, no. 5, Part 3, pp. 235–240, 1987, 10th Triennial IFAC Congress on Automatic Control - 1987 Volume III, Munich, Germany, 27-31 July.

[43] B. Lennox, G. A. Montaguet, A. M. Frith, and A. J. Beaumont, "Non-linear model-based predictive control of gasoline engine air-fuel ratio," *Transactions of the Institute of Measurement and Control*, vol. 20, no. 2, pp. 103–112, 1998.

[44] P. Bromnick, "Development of a model predictive controller for engine idle speed using cpower," *SAE Paper No. 1999-01-1171*, 1999.

[45] D. Liao-McPherson, M. Huang, S. Kim, M. Shimada, K. Butts, and I. Kolmanovsky, "Model predictive emissions control of a diesel engine airpath: Design and experimental evaluation," *International Journal of Robust Nonlinear Control*, vol. 30, no. 17, pp. 7446–7477, 2020.

[46] A. Raut, B. Irdmousa, and M. Shahbakhti, "Dynamic modeling and model predictive control of an RCCI engine," *Control Engineering Practice*, vol. 81, pp. 129–144, 2018.

[47] M. Karlsson, K. Ekholm, P. Strandh, R. Johansson, and P. Tunestål, "Multiple-input multiple-output model predictive control of a diesel engine," *IFAC Proceedings Volumes*, vol. 43, no. 7, pp. 131–136, 2010, 6th IFAC Symposium on Advances in Automotive Control.

[48] J Dahl, H Wassén, O Santin, M Herceg, L Lansky, J Pekar, and D Pachner, "Model predictive control of a diesel engine with turbo compound and exhaust after-treatment constraints," *IFAC-PapersOnLine*, vol. 51, no. 31, pp. 349–354, 2018, 5th IFAC Conference on Engine and Powertrain Control, Simulation and Modeling E-COSM 2018.

[49] D. Zhao, C. Liu, R. Stobart, J. Deng, E. Winward, and G. Dong, "An explicit model predictive control framework for turbocharged diesel engines," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 7, pp. 3540–3552, 2014.

[50] Y. Yashiro, V. Jiwariyavej, Y.Yamashita, T. Hoshi, K. Terakado, and S. Ibaraki, "High-speed model predictive control for next-generation turbocharging system," *Mitsubishi Heavy Industries Technical Review*, vol. 54, no. 1, 2017.

[51] M. Huang, D. Liao-McPherson, S. Kim, K. Butts, and I. Kolmanovsky, "Toward real-time automotive model predictive control: A perspective from a diesel air path control development," in *American Control Conference (ACC 2018)*, IEEE, 2018, pp. 2425–2430.

[52] A. Bemporad, D. Bernardini, R. Long, and J. Verdejo, "Model predictive control of turbocharged gasoline engines for mass production," in *WCX World Congress Experience*, SAE International, 2018.

[53] 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*, IEEE, 2008, pp. 5692–5697.

[54] T. Broomhead, C. Manzie, P. Hield, R. Shekhar, and M. Brear, "Economic model predictive control and applications for diesel generators," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 388–400, 2016.

[55] 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, 2011.

[56] K. Ebrahimi and C. B. Koch, "Real-time control of HCCI engine using model predictive control," in *American Control Conference (ACC 2018)*, IEEE, 2018, pp. 1622–1628.

[57] A. Widd, H.-H. Liao, J. C. Gerdes, P. Tunestål, and R. Johansson, "Control of exhaust recompression HCCI using hybrid model predictive control," in *American control conference (ACC 2011)*, IEEE, 2011, pp. 420–425.

[58] N. Ravi, H.-H. Liao, A. F. Jungkunz, A. Widd, and J. C. Gerdes, "Model predictive control of HCCI using variable valve actuation and fuel injection," *Control Engineering Practice*, vol. 20, no. 4, pp. 421–430, 2012.

[59] L. Yin, G. Turesson, P. Tunestål, and R. Johansson, "Model predictive control of an advanced multiple cylinder engine with partially premixed combustion concept," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 2, pp. 804–814, 2020.

[60] H. Ferreau, G Lorini, and M. Diehl, "Fast nonlinear model predictive control of gasoline engines," in *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, IEEE, 2006, pp. 2754–2759.

[61] P. Majecki, G. M. van der Molen, M. J. Grimble, I. Haskara, Y. Hu, and C.-F. Chang, "Real-time predictive control for si engines using linear parameter-varying models," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 94–101, 2015, 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015.

[62] S. Di Cairano, D. Yanakiev, A. Bemporad, I. V. Kolmanovsky, and D. Hrovat, "An mpc design flow for automotive control and applications to idle speed regulation," in *2008 47th IEEE Conference on Decision and Control*, IEEE, 2008, pp. 5686–5691.

[63] N. Rajaei, X. Han, X. Chen, and M. Zheng, "Model predictive control of exhaust gas recirculation valve," in *SAE 2010 World Congress & Exhibition*, SAE International, 2010.

[64] G. Stewart and F. Borrelli, "A model predictive control framework for industrial turbodiesel engine control," in *2008 47th IEEE Conference on Decision and Control*, IEEE, 2008, pp. 5704–5711.

[65] H Borhan, G Kothandaraman, and B Pattel, "Air handling control of a diesel engine with a complex dual-loop egr and vgt air system using mpc," in *American Control Conference (ACC 2015)*, IEEE, 2015, pp. 4509–4516.

[66] P. Ortner and L. Del Re, "Predictive control of a diesel engine air path," *IEEE transactions on control systems technology*, vol. 15, no. 3, pp. 449–456, 2007.

[67] H. J. Ferreau, P. Ortner, P. Langthaler, L. Del Re, and M. Diehl, "Predictive control of a real-world diesel engine using an extended online active set strategy," *Annual Reviews in Control*, vol. 31, no. 2, pp. 293–301, 2007.

[68] P. Drews, K. Hoffmann, R. Beck, R. Gasper, A. Vanegas, C. Felsch, N. Peters, and D. Abel, "Fast model predictive control for the air path of a turbocharged diesel engine," in *2009 European Control Conference (ECC)*, IEEE, 2009, pp. 3377–3382.

[69] M. E. Emekli and B. A. Güvenç, "Explicit mimo model predictive boost pressure control of a two-stage turbocharged diesel engine," *IEEE transactions on control systems technology*, vol. 25, no. 2, pp. 521–534, 2016.

[70] S. Sudhakar, A. Hansen, and J. K. Hedrick, "Algorithmic performance of receding horizon sliding control for engine emission reduction," in *2016 IEEE Conference on Control Applications (CCA)*, IEEE, 2016, pp. 1398–1403.

[71] Q. Zhu, S. Onori, and R. Prucka, "An economic nonlinear model predictive control strategy for SI engines: Model-based design and real-time experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 1, pp. 296–310, 2017.

[72] B. Shin, Y. Chi, M. Kim, P. Dickinson, J. Pekar, and M. Ko, "Model predictive control of an air path system for multi-mode operation in a diesel engine," in *WCX SAE World Congress Experience*, SAE International, 2020.

[73] Q. Zhu, R. Prucka, M. Prucka, and H. Dourra, "A nonlinear model predictive control strategy with a disturbance observer for spark ignition engines with external EGR," *SAE International Journal of Commercial Vehicles*, vol. 10, no. 1, pp. 360–372, 2017.

[74] E. Lee and L Markus, "Foundations of optimal control theory," *New York: Wiley*, 1967.

[75] J Rault, A Richalet, J. Testud, and J Papon, "Model predictive heuristic control: Application to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[76] S. J. Qin and T. A. Badgwell, "An overview of industrial model predictive control technology," in *AIche symposium series*, New York, NY: American Institute of Chemical Engineers, 1971-c2002., vol. 93, 1997, pp. 232–256.

[77] J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, no. 3, p. 415, 2011.

[78] A. Bemporad, F. Borrelli, and M. Morari, "Piecewise linear optimal controllers for hybrid systems," in *American Control Conference (ACC 2000)*, IEEE, vol. 2, 2000, pp. 1190–1194.

[79] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[80] B. Alrifaee, D. Abel, and C. Ament, "Networked model predictive control for vehicle collision avoidance," Lehrstuhl und Institut für Regelungstechnik, Tech. Rep., 2017.

[81] T. F. Yusaf, D. Buttsworth, K. H. Saleh, and B. Yousif, "CNG-diesel engine performance and exhaust emission analysis with the aid of artificial neural network," *Applied Energy*, vol. 87, no. 5, pp. 1661–1669, 2010.

[82] S. Roy, R. Banerjee, and P. K. Bose, "Performance and exhaust emissions prediction of a CRDI assisted single cylinder diesel engine coupled with egr using artificial neural network," *Applied Energy*, vol. 119, pp. 330–340, 2014.

[83] S. Javed, Y. S. Murthy, R. U. Baig, and D. P. Rao, "Development of ann model for prediction of performance and emission characteristics of hydrogen dual fueled diesel engine with jatropha methyl ester biodiesel blends," *Journal of Natural Gas Science and Engineering*, vol. 26, pp. 549–557, 2015.

[84] S. Dharma, M. H. Hassan, H. C. Ong, A. H. Sebayang, A. S. Silitonga, F. Kusumo, and J. Milano, "Experimental study and prediction of the performance and exhaust emissions of mixed jatropha curcas-ceiba pentandra biodiesel blends in diesel engine using artificial neural networks," *Journal of cleaner production*, vol. 164, pp. 618–633, 2017.

[85] A. Paul, S. Bhowmik, R. Panua, and D. Debroy, "Artificial neural network-based prediction of performances-exhaust emissions of diesohol piloted dual fuel diesel engine under varying compressed natural gas flowrates," *Journal of Energy Resources Technology*, vol. 140, no. 11, p. 112 201, 2018.

[86] S. Yıldırım, E. Tosun, A. Çalık,
bibinitperiodI. Uluocak, and E. Avşar, "Artificial intelligence techniques for the vibration, noise, and emission characteristics of a hydrogen-enriched diesel engine," *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, vol. 41, no. 18, pp. 2194–2206, 2019.

[87] S. Uslu and M. B. Celik, "Prediction of engine emissions and performance with artificial neural networks in a single cylinder diesel engine using diethyl ether," *Engineering science and technology, an international journal*, vol. 21, no. 6, pp. 1194–1201, 2018.

[88] A. Domínguez-Sáez, G. A. Rattá, and C. C. Barrios, "Prediction of exhaust emission in transient conditions of a diesel engine fueled with animal fat using artificial neural network and symbolic regression," *Energy*, vol. 149, pp. 675–683, 2018.

[89] A. Mohammad, R. Rezaei, C. Hayduk, T. O. Delebinski, S. Shahpouri, and M. Shahbakhti, "Hybrid physical and machine learning-oriented modeling approach to predict emissions in a diesel compression ignition engine," in *SAE World Congress Experience, SAE Paper No. 2021-01-0496*, 2020.

[90] A. Silitonga, H. Masjuki, H. C. Ong, A. Sebayang, S. Dharma, F. Kusumo, J. Siswantoro, J. Milano, K. Daud, T. Mahlia, W.-H. Chen, and B. Sugiyanto, "Evaluation of the engine performance and exhaust emissions of biodiesel-bioethanol-diesel blends using kernel-based extreme learning machine," *Energy*, vol. 159, pp. 1075 –1087, 2018.

[91] M. Aghbashlo, S. Shamshirband, M. Tabatabaei, L. Yee, and Y. N. Larimi, "The use of ELM-WT (extreme learning machine with wavelet transform algorithm) to predict exergetic performance of a DI diesel engine running on diesel/biodiesel blends containing polymer waste," *Energy*, vol. 94, pp. 443–456, 2016.

[92] P. K. Wong, K. I. Wong, C. M. Vong, and C. S. Cheung, "Modeling and optimization of biodiesel engine performance using kernel-based extreme learning machine and cuckoo search," *Renewable Energy*, vol. 74, pp. 640 –647, 2015.

[93] B. Liu, J. Hu, F. Yan, R. F. Turkson, and F. Lin, "A novel optimal support vector machine ensemble model for NOx emissions prediction of a diesel engine," *Measurement*, vol. 92, pp. 183–192, 2016.

[94] H. Duan, Y. Huang, R. K. Mehra, P. Song, and F. Ma, "Study on influencing factors of prediction accuracy of support vector machine (SVM) model for NOx emission of a hydrogen enriched compressed natural gas engine," *Fuel*, vol. 234, pp. 954–964, 2018.

[95] K. Wong, P. Wong, C. Cheung, and C. Vong, "Modeling and optimization of biodiesel engine performance using advanced machine learning methods," *Energy*, vol. 55, pp. 519 –528, 2013.

[96] S. Shamshirband, M. Tabatabaei, M. Aghbashlo, L. Yee, and D. Petković, "Support vector machine-based exergetic modelling of a DI diesel engine running on biodiesel–diesel blends containing expanded polystyrene," *Applied Thermal Engineering*, vol. 94, pp. 727–747, 2016.

[97] X. Niu, C. Yang, H. Wang, and Y. Wang, "Investigation of ANN and SVM based on limited samples for performance and emissions prediction of a CRDI-assisted marine diesel engine," *Applied Thermal Engineering*, vol. 111, pp. 1353–1364, 2017.

[98] D. Hao, R. K. Mehra, S. Luo, Z. Nie, X. Ren, and M. Fanhua, "Experimental study of hydrogen enriched compressed natural gas (HCNG) engine and application of support vector machine (SVM) on prediction of engine performance at specific condition," *International Journal of Hydrogen Energy*, vol. 45, no. 8, pp. 5309–5325, 2020.

[99] M Ghanbari, G Najafi, B Ghobadian, R Mamat, M. Noor, and A Moosavian, "Support vector machine to predict diesel engine performance and emission parameters fueled with nano-particles additive to diesel fuel," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 100, 2015, p. 012 069.

[100] K. I. Wong, P. K. Wong, and C. S. Cheung, "Modelling and prediction of diesel engine performance using relevance vector machine," *International journal of green energy*, vol. 12, no. 3, pp. 265–271, 2015.

[101] M. Lang, P. Bloch, T. Koch, T. Eggert, and R. Schifferdecker, "Application of a combined physical and data-based model for improved numerical simulation of a medium-duty diesel engine," *Automotive and Engine Technology*, vol. 5, no. 1, pp. 1–20, 2020.

[102] 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.

[103] Y. Bao, J. Mohammadpour Velni, and M. Shahbakhti, "An Online Transfer Learning Approach for Identification and Predictive Control Design With Application to RCCI Engines," in *Dynamic Systems and Control Conference*, ser. Dynamic Systems and Control Conference, American Society of Mechanical Engineers, vol. 84270, Oct. 2020, V001T21A003.

[104] Y. Hu, H. Chen, P. Wang, H. Chen, and L. Ren, "Nonlinear model predictive controller design based on learning model for turbocharged gasoline engine of passenger vehicle," *Mechanical Systems and Signal Processing*, vol. 109, pp. 74–88, 2018.

[105] Y. Bao, J. M. Velni, A. Basina, and M. Shahbakhti, "Identification of state-space linear parameter-varying models using artificial neural networks," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5286–5291, 2020, 21th IFAC World Congress.

[106] Y. Bao, J. M. Velni, and M. Shahbakhti, "Epistemic Uncertainty Quantification in State-Space LPV Model Identification Using Bayesian Neural Networks," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 719–724, 2020.

[107] A. Vaughan and S. Bohac, "Real-time, adaptive machine learning for non-stationary, near chaotic gasoline engine combustion time series," *Neural Networks*, vol. 70, pp. 18 –26, 2015.

[108] 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.

[109] S Batool, J Naber, and M Shahbakhti, "Data-Driven Modeling and Control of Cyclic Variability of an Engine Operating in Low Temperature Combustion Modes," *IFAC-PapersOnLine*, 2021, Modeling, Estimation and Control Conference (MECC 2021).

[110] B Khoshbakht Irdmousa, J Naber, J Mohammadpour Velni, H Borhan, and M Shahbakhti, "Input-output Data-driven Modeling and MIMO Predictive Control of an RCCI Engine Combustion," *IFAC-PapersOnLine*, 2021, Modeling, Estimation and Control Conference (MECC 2021).

[111] A Moosavian, H Ahmadi, A Tabatabaeefar, and M Khazaee, "Comparison of two classifiers; K-nearest neighbor and artificial neural network, for fault diagnosis on a main engine journal-bearing," *Shock and Vibration*, vol. 20, no. 2, pp. 263–272, 2013.

[112] I. Morgan, H. Liu, G. Turnbull, and D. Brown, "Predictive unsupervised organisation in marine engine fault detection," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, IEEE, 2008, pp. 249–256.

[113] T. Chan and C. Chin, "Data analysis to predictive modeling of marine engine performance using machine learning," in *2016 IEEE Region 10 Conference (TENCON)*, IEEE, 2016, pp. 2076–2080.

[114] X. Bi, J. Lin, D. Tang, F. Bi, X. Li, X. Yang, T. Ma, and P. Shen, "VMD-KFCM Algorithm for the Fault Diagnosis of Diesel Engine Vibration Signals," *Energies*, vol. 13, no. 1, p. 228, 2020.

[115] P. Shih, B. C. Kaul, S. Jagannathan, and J. A. Drallmeier, "Reinforcement-learning-based output-feedback control of nonstrict nonlinear discrete-time systems with application to engine emission control," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 5, pp. 1162–1179, 2009.

[116] P. Shih, B. C. Kaul, S. Jagannathan, and J. A. Drallmeier, "Reinforcement-learning-based dual-control methodology for complex nonlinear discrete-time systems with application to spark engine egr operation," *IEEE transactions on neural networks*, vol. 19, no. 8, pp. 1369–1388, 2008.

[117] J. Huotari, A. Ritari, R. Ojala, J. Vepsäläinen, and K. Tammi, "Q-Learning based autonomous control of the auxiliary power network of a ship," *IEEE Access*, vol. 7, pp. 152 879–152 890, 2019.

[118] J. Czarnigowski, "A neural network model-based observer for idle speed control of ignition in si engine," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 1, pp. 1–7, 2010.

[119] S. A. Rahman, H. Masjuki, M. Kalam, M. Abedin, A Sanjid, and H. Sajjad, "Impact of idling on fuel consumption and exhaust emissions and available idle-reduction technologies for diesel vehicles–a review," *Energy Conversion and Management*, vol. 74, pp. 171–182, 2013.

[120] J. Xue, Q. Gao, and W. Ju, "Reinforcement learning for engine idle speed control," in *2010 International Conference on Measuring Technology and Mechatronics Automation*, IEEE, vol. 2, 2010, pp. 1008–1011.

[121] M. N. Howell and M. C. Best, "On-line PID tuning for engine idle-speed control using continuous action reinforcement learning automata," *Control Engineering Practice*, vol. 8, no. 2, pp. 147–154, 2000.

[122] J. Ren, "Ann vs. SVM: Which one performs better in classification of MCCs in mammogram imaging," *Knowledge-Based Systems*, vol. 26, pp. 144–153, 2012.

[123] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489 –501, 2006, Neural Networks.

[124] Y. Xu, R. Guo, and L. Wang, "A twin multi-class classification support vector machine," *Cognitive computation*, vol. 5, no. 4, pp. 580–588, 2013.

[125] M. Tanveer, "Robust and sparse linear programming twin support vector machines," *Cognitive Computation*, vol. 7, no. 1, pp. 137–149, 2015.

[126] C. Kavuri and S. L. Kokjohn, "Exploring the potential of machine learning in reducing the computational time/expense and improving the reliability of engine optimization studies," *International Journal of Engine Research*, vol. 21, no. 7, pp. 1251–1270, 2020.

[127] A. Hanuschkin, S. Schober, J. Bode, J. Schorr, B. Böhm, C. Krüger, and S. Peters, "Machine learning–based analysis of in-cylinder flow fields to predict combustion engine performance," *International Journal of Engine Research*, vol. 0, no. 0, p. 1 468 087 419 833 269, 0.

[128] P. K. Wong, X. H. Gao, K. I. Wong, and C. M. Vong, "Online extreme learning machine based modeling and optimization for point-by-point engine calibration," *Neurocomputing*, vol. 277, pp. 187–197, 2018.

[129] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[130] M. Seeger, "Gaussian processes for machine learning," *International Journal of Neural Systems*, vol. 14, no. 02, pp. 69–106, 2004.

[131] B. Berger, F. Rauscher, and B. Lohmann, "Analysing gaussian processes for stationary black-box combustion engine modelling," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 10 633–10 640, 2011.

[132] S. Castric, L. Denis-Vidal, Z. Cherfi, G. J. Blanchard, and N. Boudaoud, "Modeling pollutant emissions of diesel engine based on kriging models: A comparison between geostatistic and gaussian process approach," *IFAC Proceedings Volumes*, vol. 45, no. 6, pp. 1708–1715, 2012.

[133] B. Berger and F. Rauscher, "Robust gaussian process modelling for engine calibration," *IFAC Proceedings Volumes*, vol. 45, no. 2, pp. 159–164, 2012.

[134] K Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 3, pp. 433–439, 1999.

[135] *Reinforcement Learning with MATLAB*. MathWorks, 2019.

[136] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[137] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

[138] K. J. Åström and B. Wittenmark, *Adaptive control*. Courier Corporation, 2013.

[139] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic systems*, vol. 1, no. 2, pp. 123–140, 1984.

[140] S. Su and G. Chen, "Lateral robust iterative learning control for unmanned driving robot vehicle," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 234, no. 7, pp. 792–808, 2020.

[141] N. Shakeri, Z. Rahmani, A. R. Noei, and M. Zamani, "Direct methanol fuel cell modeling based on the norm optimal iterative learning control," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 235, no. 1, pp. 68–79, 2021.

[142] A. Heinzen, P. Gillella, and Z. Sun, "Iterative learning control of a fully flexible valve actuation system for non-throttled engine load control," *Control Engineering Practice*, vol. 19, no. 12, pp. 1490–1505, 2011.

[143] T. Nagata and M. Tomizuka, "Robust engine torque control by iterative learning control," in *2009 American Control Conference*, 2009, pp. 2064–2069.

[144] C. Slepicka and C. R. Koch, "Iterative learning on dual-fuel control of homogeneous charge compression ignition," *IFAC-PapersOnLine*, vol. 49, no. 11, pp. 347–352, 2016, 8th IFAC Symposium on Advances in Automotive Control AAC 2016.

[145] R. Hedinger, N. Zsiga, M. Salazar, and C. Onder, "Model-based iterative learning control strategies for precise trajectory tracking in gasoline engines," *Control Engineering Practice*, vol. 87, pp. 17–25, 2019.

[146] R. Noack, T. Jeinsch, A. H. A. Sari, and N. Weinhold, "Data-driven self-tuning control by iterative learning control with application to optimize the control parameter of turbocharged engines," in *2014 19th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2014, pp. 839–844.

[147] R. Zweigel, F. Thelen, D. Abel, and T. Albin, "Iterative learning approach for diesel combustion control using injection rate shaping," in *2015 European Control Conference (ECC)*, 2015, pp. 3168–3173.

[148] K. Min, M. Sunwoo, and M. Han, "Iterative Learning Control Algorithm for Feedforward Controller of EGR and VGT Systems in a CRDI Diesel Engine," *International journal of automotive technology*, vol. 19, no. 3, 2018.

[149] X. V. Nguyen, J. Chan, S. Romano, and J. Bailey, "Effective global approaches for mutual information based feature selection," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014, pp. 512–521.

[150] M. A. Hall, "Correlation-based Feature Selection for Machine Learning. PhD thesis. The University of Waikato (1999),"

[151] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Advances in bioinformatics*, vol. 2015, 2015.

[152] M. Bidarvatan, V. Thakkar, and M. Shahbakhti, "Grey-box modeling and control of HCCI engine emissions," in *American Control Conference (ACC 2014)*, IEEE, 2014, pp. 837–842.

[153] S. Z. Rizvi, J. Mohammadpour, R. Tóth, and N. Meskin, "An IV-SVM-based approach for identification of state-space LPV models under generic noise conditions," in *2015 54th IEEE Conference on Decision and Control (CDC)*, IEEE, 2015, pp. 7380–7385.

[154] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[155] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," *Applied Energy*, vol. 218, pp. 199–216, 2018.

[156] B. Karg and S. Lucia, "Deep learning-based embedded mixed-integer model predictive control," in *2018 European Control Conference (ECC)*, IEEE, 2018, pp. 2075–2080.

[157] L. Sun, C. Peng, W. Zhan, and M. Tomizuka, "A fast integrated planning and control framework for autonomous driving via imitation learning," in *Dynamic Systems and Control Conference*, American Society of Mechanical Engineers, vol. 51913, 2018, V003T37A012.

[158] X. Zhang, M. Bujarbaruah, and F. Borrelli, "Safe and near-optimal policy learning for model predictive control using primal-dual neural networks," in *2019 American Control Conference (ACC)*, IEEE, 2019, pp. 354–359.

[159] B. Cera and A. M. Agogino, "Multi-cable rolling locomotion with spherical tensegrities using model predictive control and deep learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1–9.

[160] M. Novak and T. Dragicevic, "Supervised imitation learning of finite set model predictive control systems for power electronics," *IEEE Transactions on Industrial Electronics*, 2020.

[161] Y. Hu, W. Li, K. Xu, T. Zahid, F. Qin, and C. Li, "Energy management strategy for a hybrid electric vehicle based on deep reinforcement learning," *Applied Sciences*, vol. 8, no. 2, p. 187, 2018.

[162] X. Qi, G. Wu, K. Boriboonsomsin, and M. J. Barth, "A novel blended real-time energy management strategy for plug-in hybrid electric vehicle commute trips," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2015, pp. 1002–1007.

[163] A. J. Smith, "Applications of the self-organising map to reinforcement learning," *Neural networks*, vol. 15, no. 8-9, pp. 1107–1124, 2002.

[164] K. P. Wabersich and M. N. Zeilinger, "Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning," *arXiv preprint arXiv:1812.05506*, 2018.

[165] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, 2021.

[166] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Transactions on Automatic Control*, 2020.

[167] S. Gros, M. Zanon, and A. Bemporad, "Safe Reinforcement Learning via Projection on a Safe Set: How to Achieve Optimality?" *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8076–8081, 2020, 21st IFAC World Congress.

[168] D. Gordon, C. Wouters, M. Wick, F. Xia, B. Lehrheuer, J. Andert, C. R. Koch, and S. Pischinger, "Development and experimental validation of a real-time capable FPGA based gas-exchange model for negative valve overlap," *International Journal of Engine Research*, 2018.

[169] J. Pfluger, J. Andert, H. Ross, and F. Mertens, "Rapid control prototyping for cylinder pressure indication," *MTZ Worldwide*, vol. 73, no. 11, pp. 38–42, 2012.

[170] R. Klikach, "Investigation and analysis of RCCI using NVO on a converted Spark Ignition engine," 2018.

[171] L Tarabet, K. Loubar, M. Lounici, K Khiari, T Belmrabet, and M. Tazerout, "Experimental investigation of DI diesel engine operating with eucalyptus biodiesel/natural gas under dual fuel mode," *Fuel*, vol. 133, pp. 129–138, 2014.

[172] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints," *IEEE transactions on evolutionary computation*, vol. 18, no. 4, pp. 577–601, 2013.

[173] V. Vapnik and A. Lerner, "Generalized portrait method for pattern recognition," *Automation and Remote Control*, vol. 24, no. 6, pp. 774–780, 1963.

[174] V. Vapnik and A. Chervonenkis, "A note on class of perceptron," *Automation and Remote Control*, vol. 24, 1964.

[175] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, 1997, pp. 155–161.

[176] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[177] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[178] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[179] R. Bellman *et al.*, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954.

[180] W. Karush, "Minima of functions of several variables with inequalities as side constraints," *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago*, 1939.

[181] R. A. Brualdi, *Introductory combinatorics*. Pearson Education India, 1977.

[182] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[183] P. Juszczak, D Tax, and R. P. Duin, "Feature scaling in support vector data description," in *Proc. ASCI*, Citeseer, 2002, pp. 95–102.

[184] A. Norouzi, M. Aliramezani, and C. R. Koch, "Diesel engine NOx reduction using a PD-type fuzzy iterative learning control with a fast response NOx sensor," *Proceedings of Combustion Institute-Canadian Section (CICS 2019)*, 2019.

[185] Y He and C. Rutland, "Application of artificial neural networks in engine modelling," *International Journal of Engine Research*, vol. 5, no. 4, pp. 281–296, 2004.

[186] S. Shalev-Shwartz and N. Srebro, "SVM optimization: Inverse dependence on training set size," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 928–935.

[187] M. Aliramezani, A. Norouzi, C. R. Koch, and R. E. Hayes, "A control oriented diesel engine NOx emission model for on board diagnostics and engine control with sensor feedback," *Proceedings of Combustion Institute-Canadian Section (CICS 2019), Kelowna, Canada*, 2019.

[188] H. Hiroyasu, T. Kadota, and M. Arai, "Development and use of a spray combustion modeling to predict diesel engine efficiency and pollutant emissions: Part 1 combustion modeling," *Bulletin of JSME*, vol. 26, no. 214, pp. 569–575, 1983.

[189] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 569–575, 2009.

[190] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges: Chapter 3- Neural Architecture Search*. Springer Nature, 2019.

[191] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[192] R. A. Berk, *Statistical learning from a regression perspective: Chapter 3- Classification and regression trees (CART)*. Springer, 2008, vol. 14.

[193] M. Aliramezani, A. Norouzi, and C. R. Koch, "A grey-box machine learning based model of an electrochemical gas sensor," *Sensors and Actuators B: Chemical*, vol. 321, p. 128 414, 2020.

[194] M. H. Hassoun *et al.*, *Fundamentals of artificial neural networks*. MIT press, 1995.

[195] F. D. Foresee and M. T. Hagan, "Gauss-newton approximation to bayesian learning," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, IEEE, vol. 3, 1997, pp. 1930–1935.

[196] J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017.

[197] H. Omidvarborna, A. Kumar, and D.-S. Kim, "Recent studies on soot modeling for diesel combustion," *Renewable and Sustainable Energy Reviews*, vol. 48, pp. 635–647, 2015.

[198] R. Rezaei, C. Hayduk, E. Alkan, T. Kemski, T. Delebinski, and C. Bertram, "Hybrid phenomenological and mathematical-based modeling approach for diesel emission prediction," in *SAE World Congress Experience, SAE Paper No. 2020-01-0660*, 2020.

[199] S. Shahpouri, A. Norouzi, C. Hayduk, R. Rezaei, M. Shahbakhti, and C. R. Koch, "Soot emission modeling of a compression ignition engine using machine learning," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 826–833, 2021, Modeling, Estimation and Control Conference MECC 2021.

[200] A. Domahidi and J. Jerez, *Forces professional*, Embotech AG, url=https://embotech.com/FORCES-Pro, 2014–2019.

[201] 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*, vol. 93, no. 1, pp. 13–29, 2020.

[202] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "Acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[203] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, and M. Diehl, "Towards a modular software package for embedded optimization," in *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2018.

[204] 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, 21th IFAC World Congress.

[205] A. Winkler, J. Frey, T. Fahrbach, G. Frison, R. Scheer, M. Diehl, and J. Andert, "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.

[206] 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*, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 391–417.

[207] G. Frison, D. Kouzoupis, J. Jørgensen, and M. Diehl, "An efficient implementation of partial condensing for nonlinear model predictive control," Dec. 2016, pp. 4457–4462.

[208] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[209] D. Gordon, C. Wouters, M. Wick, B. Lehrheuer, J. Andert, C. Koch, and S. Pischinger, "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.

[210] *acados Documentation: Embedded Workflow*, https://docs.acados.org/embedded_workflow/index.html, Accessed: 2022-04-06.

[211] M. Aliramezani, C. R. Koch, and M. Shahbakhti, "Modeling, diagnostics, optimization, and control of internal combustion engines via modern machine learning techniques: A review and future directions," *Progress in Energy and Combustion Science*, vol. 88, p. 100 967, 2022.

[212] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[213] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, no. 5, p. 823, 1930.

[214] P. Wawrzynski, "Control policy with autocorrelated noise in reinforcement learning for robotics," *International Journal of Machine Learning and Computing*, vol. 5, no. 2, p. 91, 2015.

[215] D. Gordon, A. Norouzi, G. Blomeyer, J. Bedei, M. Aliramezani, J. Andert, and C. R. Koch, "Support vector machine based emissions modeling using particle swarm optimization for homogeneous charge compression ignition engine," *International Journal of Engine Research*, eprint: https://doi.org/10.1177/14680874211055546.

[216] A. Norouzi and C. R. Koch, "Robotic manipulator control using pd-type fuzzy iterative learning control," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, 2019, pp. 1–4.

[217] A. Norouzi and C. R. Koch, "Integration of pd-type iterative learning control with adaptive sliding mode control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6213–6218, 2020, 21st IFAC World Congress.

[218] M. Aliramezani, A. Norouzi, and C. R. Koch, "Support vector machine for a diesel engine performance and NOx emission control-oriented model," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 13 976–13 981, 2020, 21st IFAC World Congress.

Appendix A: Ph.D. Publications

A.1 Peer Reviewed Journal Papers

1. **A. Norouzi**, H. Heidarifar, A. Borhan, M. Shahbakhti, C.R. Koch, Integrating Machine Learning and Model Predictive Control for Automotive Applications: A Review and Future Directions, Engineering Applications of Artificial Intelligence (Submitted on June 24, 2022).

2. **A. Norouzi**, S. Shahpouri, M. Shahbakhti, and C. R. Koch, Safe Deep Reinforcement Learning in Diesel Engine Emission Control, Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering (Submitted on April 3, 2022).

3. **A. Norouzi**, D. Gordon, A. Winkler, J. McNally, E. Nuss, D. Abel, J. Andert, M. Shahbakhti, and C. R. Koch, End-to-End Deep Neural Network-based Nonlinear Model Predictive Control: Experimental Implementation on Diesel Engine Emission Control, Transaction on Control Systems Technology (Submitted on May 26, 2022).

4. S. Shahpouri, **A. Norouzi**, C. Hayduk, A. Fandakov, R. Rezaei, C. R. Koch, M. Shahbakhti, Laminar Flame Speed Modeling for Low Carbon Fuels Using Methods of Machine Learning, Fuel (Submitted on May 9, 2022).

5. **A. Norouzi**, S. Shahpouri, D. Gordon, A. Winkler, E. Nuss, D. Abel, J. Andert, M. Shahbakhti, and C. R. Koch, Deep Learning based Model Predictive Control for Compression Ignition Engines, Control Engineering Practice (In press).

6. S. Shahpouri, **A. Norouzi**, C. Hayduk, R. Rezaei, M. Shahbakhti, and C. R. Koch, Hybrid Machine Learning approaches and a systematic model selection process for predicting soot emissions in compression ignition engines, *Energies*, 14(23) (2021), 7865.

7. D.C. Gordon, **A. Norouzi**, G. Blomeyer, J. Bedei, M. Aliramezani, J. Andert, and C.R. Koch, Support Vector Machine Based Emissions Modeling using Particle Swarm Optimization for Homogeneous Charge Compression Ignition Engine, *International Journal of Engine Research*, OnlineFirst, November 2021.

8. **A. Norouzi**, H. Heidarifar, A. Borhan, M. Shahbakhti, C.R. Koch, Application of Model Predictive Control for Internal Combustion Engines (ICEs) Control: A review and future directions, *Energies*, 14(19) (2021): 6251.

9. **A. Norouzi**, M. Aliramezani, C.R. Koch, A correlation based model order reduction approach for a diesel engine NO_x and BMEP dynamic model using machine learning, *International Journal of Engine Research*, 22.8 (2021): 2654-2672.

10. M. Aliramezani, **A. Norouzi**, C.R. Koch, A grey-box machine learning based model of an electrochemical gas sensor, *Sensors and Actuators B: Chemical* 321 (2020): 128414.

A.2 Refereed Conference Papers in Proceedings

1. **A. Norouzi**, S. Shahpouri, D. Gordon, A. Winkler, E. Nuss, D. Abel, J. Andert, M. Shahbakhti, and C. R. Koch, Machine Learning Integrated with Model Predictive Control for Imitative Optimal Control of Compression Ignition Engines, 10th Symposium on Advances in Automotive Control (AAC22) (In press).

2. S. Shahpouri, **A. Norouzi**, C. Hayduk, R. Rezaei, M. Shahbakhti, and C. R. Koch, Soot emission modeling of a compression ignition engine using machine

learning, *Modeling, Estimation and Control Conference (MECC 2021), 24-27 October 2021, University of Texas at Austin, Texas, United States.* .

3. **A. Norouzi**, D. Gordon, M. Aliramezani, C.R. Koch, Machine Learning-based Diesel Engine-Out NOx Reduction Using a plug-in PD-type Iterative Learning Control, *4th IEEE Conference on Control Technology and Applications (CCTA 2020)*, August 24-26, 2020, Montreal, QB, Canada.

4. **A. Norouzi**, C.R. Koch, Integration of PD-type iterative learning control with adaptive sliding mode control, *IFAC World Congress 2020*, July 12-77, 2020, Berlin, Germany.

5. M. Aliramezani, **A. Norouzi**, C.R. Koch, Support vector machine for a diesel engine performance and NO_xemission control-oriented model, *IFAC World Congress 2020*, July 12-77, 2020, Berlin, Germany.

6. **A. Norouzi**, KH. Ebrahimi, C.R. Koch, Integral Discrete-time Sliding Mode Control of Homogeneous Charge Compression Ignition (HCCI) Engine Load and Combustion Timing, *9th Symposium on Advances in Automotive Control (AAC19)*, June 23-27, 2019, Orleaon, France.

7. **A. Norouzi**, C.R. Koch, Robotic manipulator control using PD-type fuzzy iterative learning control, *32nd Canadian Conference on Electrical & Computer Engineering (CCECE)*, May 5-8, 2019, Edmonton, AB, Canada.

A.3 Technical Presentations & workshops (refereed abstract)

1. **A. Norouzi**, M. Shahbakhti, and C. R. Koch, Machine Learning Control Workshop, *Canadian Society for Mechanical Engineering International Congress (CSME)*, June 5-8, 2022, Edmonton, Canada.

2. J. Mcnally, D. Gordon, **A. Norouzi**, M. Shahbakhti, and C. R. Koch, Experimental Study of Hydrogen Diesel Dual Fuel Engine Characterization, *Canadian Society for Mechanical Engineering International Congress (CSME)*, June 5-8, 2022, Edmonton, Canada.

3. **A. Norouzi**, S. Shahpouri, D. Gordon, A. Winkler, E. Nuss, D. Abel M. Shahbakhti, and C. R. Koch, Deep Learning and Nonlinear Model Predictive Control Integration for Compression Ignition Engine Emission Reduction, *Canadian Society for Mechanical Engineering International Congress (CSME)*, June 5-8, 2022, Edmonton, Canada.

4. **A. Norouzi**, S. Shahpouri, D. Gordon, M. Shahbakhti, and C. R. Koch, Deep Reinforcement Learning for Emission Control in Diesel Engines, *Canadian Society for Mechanical Engineering International Congress (CSME)*, June 5-8, 2022, Edmonton, Canada.

5. S. Shahpouri, **A. Norouzi**, C. Hayduk, A. Fandakov, R. Rezaei, M. Shahbakhtia, C. R. Koch, Laminar Flame Speed Modeling of Hydrogen, Methanol and Ammonia Using Machine Learning of Machine Learning, *Canadian Society for Mechanical Engineering International Congress (CSME)*, June 5-8, 2022, Edmonton, Canada.

6. S. Shahpouri, **A. Norouzi**, C. Hayduk, R. Rezaei, M. Shahbakhtia, C. R. Koch, Machine Learning Modeling of Soot Emissions in a Medium Duty Diesel Engine, *Canadian Society for Mechanical Engineering International Congress (CSME)*, June 5-8, 2022, Edmonton, Canada.

7. S. Shahpouri, **A. Norouzi**, C. Hayduk, R. Rezaei, C. R. Koch, and M. Shahbakhtia, Modeling of a Medium Duty Dual Fuel Diesel-Hydrogen Engine, *Proceedings of Combustion Institute Canadian Section (CICS)*, May 16-19, 2022, Ottawa, ON, Canada

8. D. Gordon, J. Mcnally, **A. Norouzi**, and C. R. Koch, Experimental Investigation of Hydrogen Diesel Dual Fuel Combustion, *Aachen Hydrogen Colloquium*, May 3-4, 2022, Aachen Germany.

9. D. Gordon, **A. Norouzi**, C.R. Koch,AI-based Advance Control Methods for next generation combustion engines,*Autonomous Systems Initiative (ASI) Annual Symposium*, June 2, 2021, Edmonton, Canada (Best presentation award).

10. **A. Norouzi**, M. Shahbakhti, C.R. Koch, Machine Learning-Based Diesel Engine-Out Emissions Model and Control Using the Learning-Based Control Technique,*WCX SAE World Congress*, April 13, 2021, Detroit, USA.

11. M. Aliramezani, **A. Norouzi**, C.R. Koch, R. E. Hayes, A control oriented diesel engine NOx emission model for on board diagnostics and engine control with sensor feedback, *Proceedings of Combustion Institute Canadian Section (CICS)*, May 13-16, 2019, Kelowna, BC, Canada.

12. **A. Norouzi**, M. Aliramezani, C.R. Koch, Diesel Engine NOx Reduction Using a PD-type Fuzzy Iterative Learning Control with a Fast Response NOx Sensor, *Proceedings of Combustion Institute Canadian Section (CICS)*, May 13-16, 2019, Kelowna, BC, Canada.

A.4 Technical Posters

1. D. Gordon, **A. Norouzi**, C.R. Koch, AI-based Advance Control Methods for next generation combustion engines,*2021 Future Energy Systems Research Symposium*, Sept 20, 2021, Edmonton, Canada.

2. M. Aliramezani, **A. Norouzi**, D. Gordon, C.R. Koch, Emission reduction of internal combustion engines with advanced control and machine learning techniques, *Future Energy Systems Real World Industry Mixer*, Feb 20, 2020.

3. D. Gordon, **A. Norouzi**, M. Aliramezani, C.R. Koch, Combustion Control Research –*University of Alberta, Canadian Graduate Engineering Consortium*, Sept 2019

4. D. Gordon, **A. Norouzi**, M. Aliramezani, C.R. Koch, Real-time Engine Control Utilizing Emission Measurement with FPGA Controller, *2nd annual Future Energy Systems Open house*, Oct 3, 2018

Appendix B: Research Source File

All of the research source codes exist in shared repository with supervisors of this thesis. The following is a list of the repositories and brief descriptions of each repositories.

- `https://gitlab.com/arminny/gt-power-model-soot-model`
  - **Descriptions:** Hybrid soot emission model and GT-power model implementation
  - **Papers:** [5, 199]
- `https://gitlab.com/arminny/hcci-model-control`
  - **Descriptions:** Homogeneous charge compression ignition (HCCI) engine control using sliding model control based on a model developed by Khashayar Ebrahimi and HCCI engine-out emission models using a support vector machine
  - **Papers:** [36, 215]
- `https://gitlab.com/arminny/ilc-engine-robatic-papers`
  - **Descriptions:** Iterative learning control (ILC) implementation in robotic including Quanser Qube real-time implementation and manipulated robotics. ILC implementation on engine to reduce NOx emission based on an SVM-based model
  - **Papers:** [184, 187, 216, 217]

- `https://gitlab.com/arminny/mpc-real-time`
  - Real-time implementation of long-short term memory-based nonlinear model predictive control (LSTM-NMPC) using acados
  - **Paper:** [8]
- `https://gitlab.com/arminny/nmpc-gt-simulation`
  - **Descriptions:** NMPC, Linear Parameter Varying (LPV) MPC, and linear MPC implementation in GT-power model
  - **Papers:** [6, 7]
- `https://gitlab.com/arminny/nox-sensor-modeling`
  - **Descriptions:** Gray-box NOx sensor modeling including raw-data and NOx sensor gray-box modeling codes
  - **Paper:** [193]
- `https://gitlab.com/arminny/diesel-svm-model`
  - **Descriptions:** Support vector machine-based modeling for NOx emission including model-order reduction algorithm, raw data, and NOx modeling
  - **Papers:** [3, 218]
- `https://gitlab.com/arminny/rl-ilc-gt-simulations`
  - **Descriptions:** Reinforcement learning and iterative learning control implementation using GT-power model (ESM)
  - **Papers:** [9]