Optimal control of diesel engines

School of Engineering

University of Liverpool

This thesis is submitted for the degree of

Doctor of Philosophy

Callum Moseley

July 2019

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Contents

List of Figures

Abstract

Diesel engine calibration is an increasingly complicated task that is very time-consuming and costly. This is due to additional sensors and actuators added over time to help meet stringent EU emissions legislation, each of which must be calibrated. In order to keep satisfying EU emissions limits, new methods must be developed.

A method for detecting nonlinear behaviour in dynamic systems is implemented and described as a way to inform the design of controllers. Then, a nonlinear dynamic neural network control methodology is described for application on LPEGR position control. This is compared to manual PI control and an optimised PI controller, resulting in similar performance for both neural network and optimised PI, performing far better than the manual PI and taking much less time to design.

The neural network control method is applied to a more complicated system, a validated model of a diesel engine air-path. This was done to control torque and the composition of the air using an observer to control via feedback quantities which cannot be measured using sensors on a production engine. An emissions model was implemented on the simulated engine and the control methodology was adapted to minimise emissions directly.

Randomised signals are used to drive the examined systems and derive the necessary models and controllers. Being random, the signals contain redundant data in order to make sure the whole operating envelope is explored. A natural progression of the described methodology, an optimisation-based method is adapted for the design of test signals. The method is demonstrated on a simple nonlinear system with the objective of maximising information content in the data collected from the system during experiments, therefore reducing testing time and improving the calibration results. The method is then applied to a test signal for a validated diesel engine air-path model, and a model is derived from it. The model showed an average improvement in performance over the initial test signal, as did the controller derived from that model.

Chapter 1

Introduction

1.1 Overview

A production engine must be calibrated to meet a number of legislative performance standards, which include emissions limits imposed by the European Union (EU). Static Design of Experiments (DoE) for diesel engine calibration is the standard approach used in industry. Ropke summarised the process as [1]:

1. Choose operating points and corresponding emissions targets;

2. Optimise engine response at each operating point by adjusting VGT and EGR;

3. Build continuous maps by smoothing between those settings.

where VGT is the variable geometry turbocharger and EGR is the exhaust gas recirculation valve. However, as EU emissions limits are made more strict over time, the number of sensors and actuators used by modern diesel engines have grown [1]. As this number has increased more and more time and effort has been required to calibrate engines due to the dimensions of their operating space [2]. Many of these systems used to regulate the engine can also have highly nonlinear dynamics and non-minimum phase behaviour [3], making them difficult to calibrate controllers for [4] [5] [6]. EU emissions standards are shown in figure 1.1 up to the 2008 EU5 standard, at which point the limits become smaller than would be visible given the scale of previous EU standards. Such representation highlights the importance of technology to keep satisfying legislation.

Typically legislation limits are met via a combination of turbocharging and exhaust gas recirculation loops, as illustrated in figure 1.2. Karagiorgis, Glover and Collings [6] discussed the control challenges facing modern diesel engines, supporting the assertion that an increasing number of calibratable parameters makes calibration exponentially more difficult due to the "curse of dimensionality". Typically these control challenges are approached using static lookup tables [1] and gain scheduling [9]. Gain scheduling can be used to approximate nonlinear controllers to handle difficult nonlinear behaviour, but its drawbacks include needing a "slow" variable to schedule with [10]. This is often not the case in automotive applications, as in the case of valve calibration where position is controlled using gains scheduled by its position. The limitations are often due to engine control unit (ECU) architecture

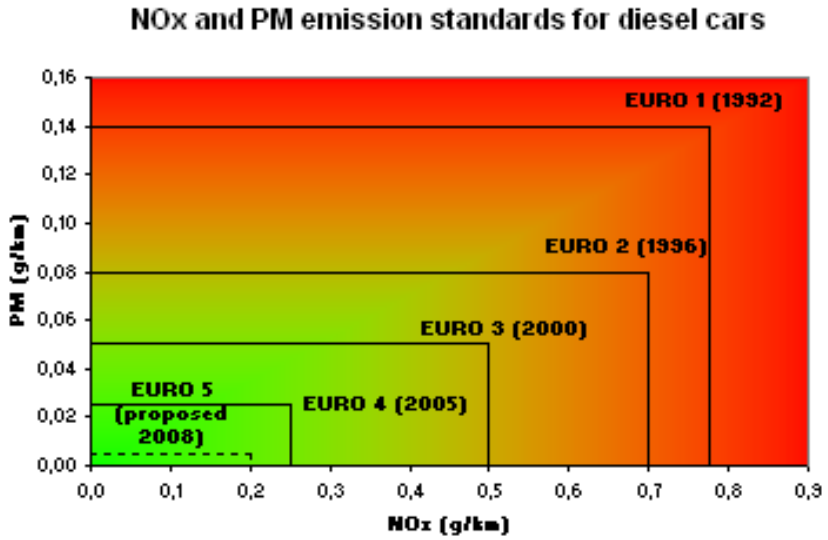

Figure 1.1: EU emissions standards [7]

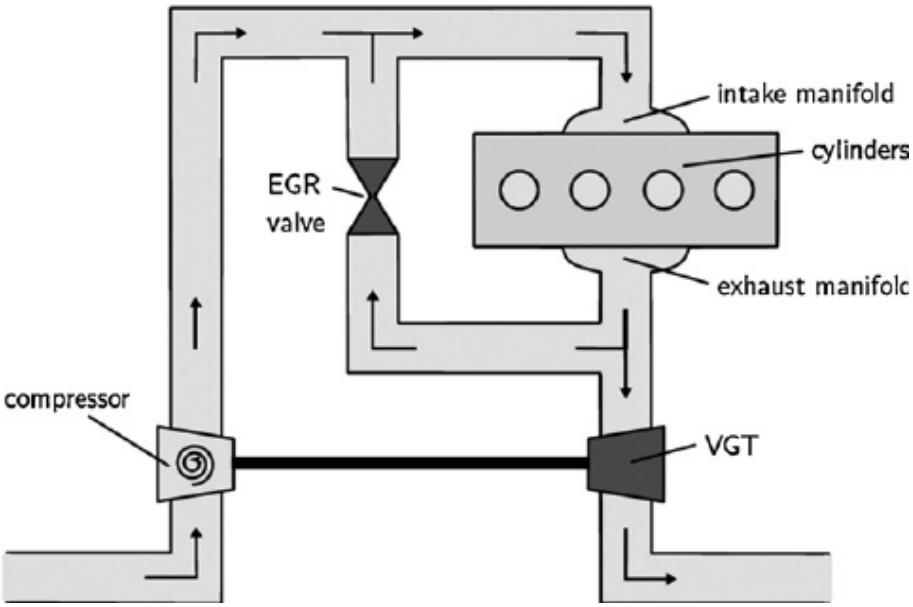

Figure 1.2: Diesel air path schematic [8]

being set by the manufacturer, forcing calibration engineers to work with set structures of lookup table, where often not even the number or spread of gain stages can be adjusted.

In this thesis a dynamic methodology is implemented to overcome the issues of static DoE and gain scheduling in diesel engine calibration. The method is based on deriving a dynamic model of the engine first, which is then used to design a nonlinear controller capable of controlling the whole operating envelope at once. This technique is adapted for and tested on three cases:

1. a low pressure exhaust gas recirculation valve (LPEGR);

2. a whole engine model, where torque and air composition are controlled as setpoints;

3. a whole engine where torque is controlled while emissions are minimised.

The LPEGR case was used as a practical demonstration of the nonlinear dynamic control method, and was successfully demonstrated on a test bench with two different valves. Later, it was adapted to whole-engine control for two cases: control of air composition setpoints, and direct emissions minimisation. An emissions model was developed to add to the existing validated diesel model used in these methods. In all cases, the controller was able to successfully control the system at hand, with performance often better than the one obtained with current approaches.

A related method for the detection of nonlinearity in the behaviour of a system is also presented, which allows a system to be defined as nonlinear according to statistical metrics. This method was explored to help design the structure of calibration models without resorting to trial and error. A method for optimising test signals is also adapted for identification of engine systems, and together these could form the basis of a full dynamic DoE method in future work.

1.2 Aim and objectives

The aim of the project was to explore and develop dynamic methods for calibrating diesel engines in a way which remains feasible as emissions limits decrease over time. To this end, the broad objectives are listed below:

1. demonstrate the control method on an LPEGR system by comparing its performance to that of a standard manually-tuned controller;

2. apply the method to a DoE-style setpoint based controller on a whole simulated engine;

3. use the same method to reduce emissions directly on the whole simulated engine;

4. outline how this method can be adapted into a full dynamic DoE approach.

1.3 Thesis structure

The thesis is structured as follows:

- Chapter 1 reviews the current state of the art to identify the research gap and puts the methodology demonstrated in the thesis into context;
- Chapter 2 reviews literature on relevant topics in control theory and engines;
- Chapter 3 describes a nonlinearity detection algorithm which was developed to assess the degree of dynamic nonlinearity in a system's behaviour before designing a controller;
- Chapter 4 applies the neural network control method to two LPEGRs and compares the applied controller against current standard calibration methods;
- Chapter 5 applies the neural network control method to a whole-engine model in order to control operating setpoints;
- Chapter 6 describes the implementation of an emissions model in the whole-engine model, then applies the neural network control method to the direct reduction of emissions;
- Chapter 7 adapts a method of optimising test signals for the refinement of the methodology described in chapters 4-6;
- Chapter 8 summarises conclusions made and suggests potential future research avenues.

Chapter 2

Literature Review

In this chapter a brief overview of technologies important to engine calibration is given in section 2.1. The most relevant state-of-the-art approaches to engine calibration are also presented, in particular, the traditional approach based on static design of experiments (DoE) is reviewed in Section 2.2, whereas the more modern approach based on dynamic calibration is discussed in Section 2.3. Finally, given that the neural network control methodology is based on obtaining an accurate model of the system to be controlled, the application of System Identification techniques to engine calibration is discussed in Section 2.4. The system identification method of choice for most of the research in this thesis used neural networks, which are discussed in Section 2.5. Finally, research contributions are summarised in Section 2.6. The state of the art related to specific aspects and systems involved in engine calibration is discussed in more detail in each of the following chapters.

2.1 Engine technologies

Engines are divided into two broad categories: gasoline and diesel. Besides the type of fuel used in each, the primary difference is the method of ignition [2]. Gasoline engines use spark ignition (SI) and diesel engines use compression ignition (CI). Moreover, there are novel ignition methods under development, such as laser ignition (LI), examples of which can be found in [11], [12] and [13]. The basic structure of the diesel engine cylinder is shown by the diagram in figure 2.1.

The stages of the four-stroke cycle, illustrated by figure 2.1 and described, for example, by [15], are:

1. a. *Intake stroke.* Under the inertia of the engine, the piston head moves from top dead centre (TDC) to top dead centre (BDC), expanding the cylinder volume as the intake valve opens, drawing in fresh air;

2. b. *Compression stroke.* With the intake valve closed, the piston moves from BDC to TDC, compressing the air in the cylinder to the clearance volume V_c and raising its temperature;

3. c. *Injection/Power stroke.* The fuel injector sprays fuel into the cylinder and the heat of the compressed air ignites it. The burning fuel increases pressure in the cylinder, driving the piston;

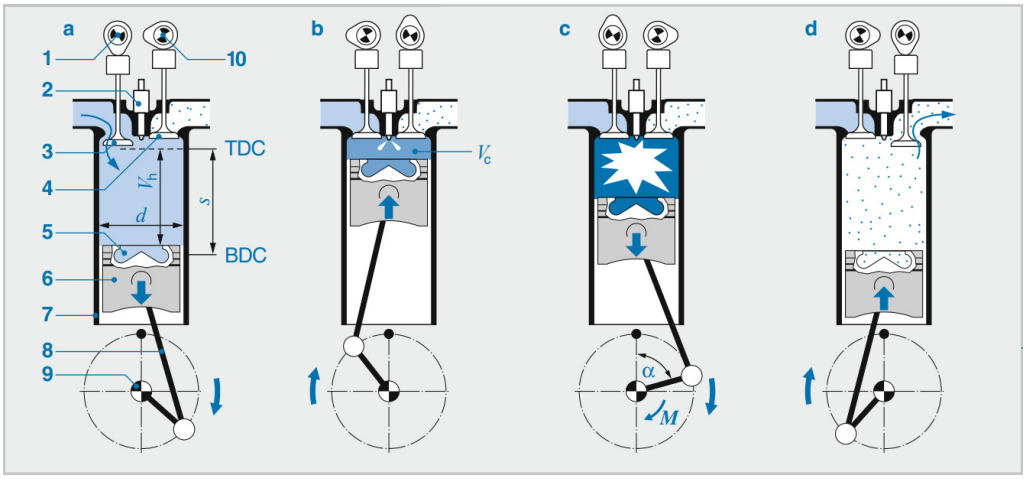

Figure 2.1: Structure of a CI diesel engine cylinder [14]. (1) Inlet-valve camshaft, (2) fuel injector, (3) inlet valve, (4) exhaust valve, (5) combustion chamber, (6) piston, (7) cylinder wall, (8) connecting rod, (9) crankshaft, (10) exhaust-valve camshaft.

4. d. *Exhaust stroke.* As the piston moves back up to TDC a second time, the exhaust valve opens and exhaust gas is driven from the cylinder.

As the engine goes through its combustion cycle, pressure and volume vary in the cylinder. A plot of pressure and volume variations in the cylinder during its 2- or 4-stroke cycle is called an indicator diagram, or P-V diagram. The P-V curve, shown in figure 2.2, represents the thermodynamic character of the engine, as the integral of the curve gives the final work output over a single cycle.

Gasoline SI engines use a spark plug to initiate combustion of the fuel-air mixture in the engine cylinders, whereas diesel CI engines ignite the fuel-air mixture via compression with the piston. As such, diesel engines typically operate at higher compression ratios than gasoline engines. Compression ratio r is defined as the ratio of maximum volume V_d displaced by the piston moving between BDC and TDC, to clearance volume V_c left above the piston at TDC:

$$r = \frac{V_{max}}{V_{min}} = \frac{V_d + V_c}{V_c} \tag{2.1}$$

This is related to air cycle efficiency [16] by equation (2.2),

$$\eta_{ac} = 1 - r^{(1-\gamma)} \tag{2.2}$$

where γ is the ratio for specific heats. For air, γ is equal to 1.4.

Air cycle efficiency is one factor in brake thermal efficiency, which is shown in an experimental evaluation by [18], see figure 2.4. Gasoline and diesel are compared for three different compression ratios 7, 8.5 and 10, showing that for the same fuel type, efficiency will typically increase with compression ratio. Further experimental results were gathered by Aldhaidhawi *et al.* [19] purely for diesel engines of more typical operating compression ratios in the range of 18-22. The results are summarised in table 2.1 and show that brake specific fuel consumption decreases with compression ratio, in line with an increase in efficiency. As such, the higher compression ratios at which diesel engines operate is the source of one of their advantages over gasoline engines.

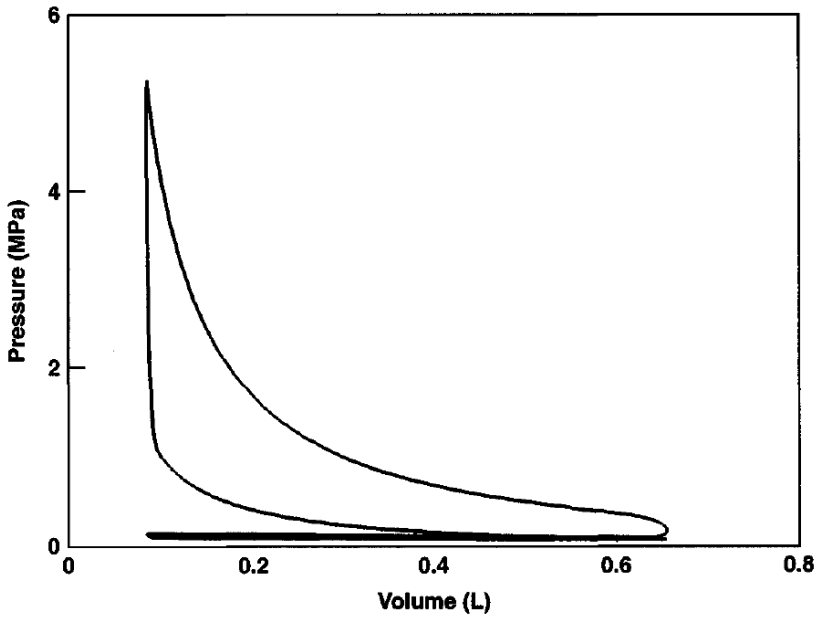

Figure 2.2: P-V curve for the Jaguar XK (4-stroke, 3.4 L , r=8) [16].

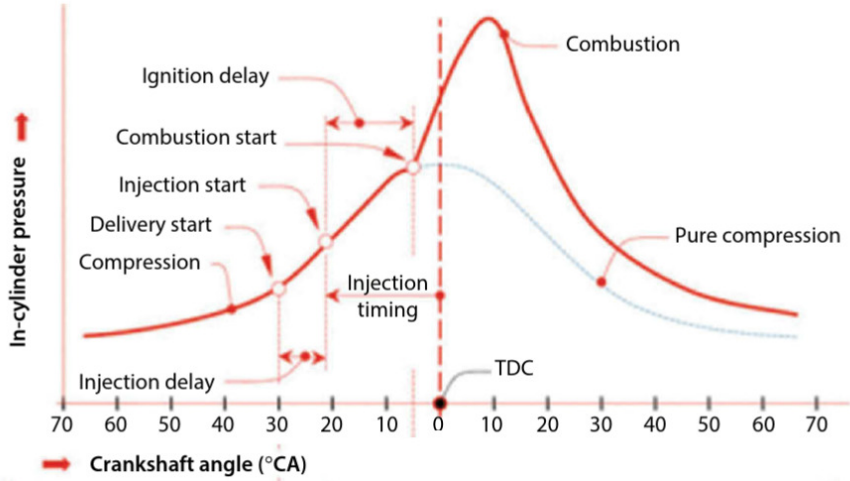

Figure 2.3: Plot of in-cylinder pressure variation with crankshaft angle [17].

Compression ratio	Brake specific fuel consumption (g/kWh)
22	370.396
20	403.17
18	456.69

Table 2.1: Summary of experimental results for compression ratio and BSFC [19].

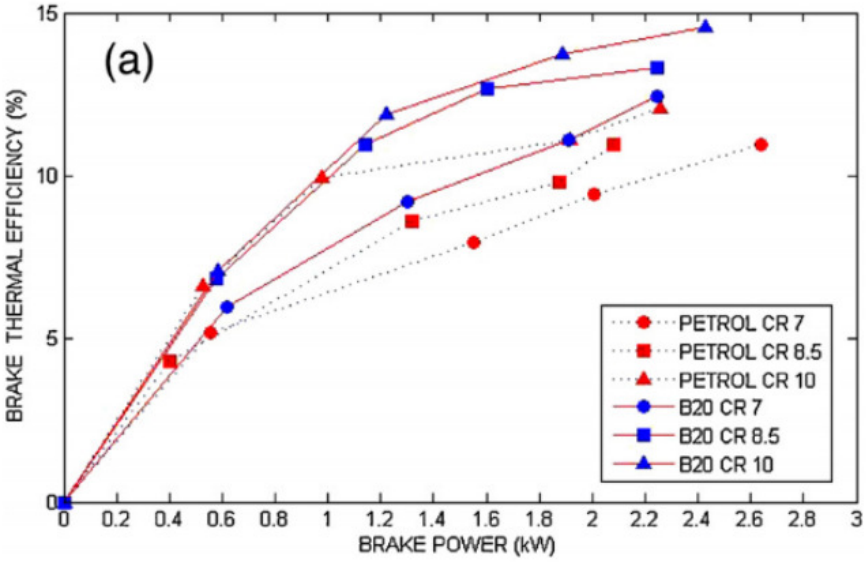

Figure 2.4: Brake efficiency for varying fuel type and compression ratio (CR) [18].

A disadvantage of diesel engines is their higher production of emissions compared to gasoline engines of similar size [20]. The focus of this thesis is to control two such emissions: nitrogen oxides (NO_x) and particulate matter (PM). NO_x includes NO and NO_2, but primarily consists of NO [16], and is commonly treated as such in modelling literature [21]. However, NO undergoes a photochemical reaction with unburnt hydrocarbons to form NO_2, which dissociates back into NO while producing an oxygen free radical, which bonds with atmospheric oxygen (O_2) to form ozone (O_3) [22]. Ozone is a primary constituent of photochemical smog, which causes respiratory illnesses and reduces crop productivity, among other ecologically undesirable phenomena [23].

Two key technologies in controlling these emissions through air-path management are turbocharging and exhaust gas recirculation (EGR). A simple schematic of a typical diesel air-path management system is given in figure 2.5. The turbocharger is made up of a turbine which is driven by the exhaust gas pressure and by a compressor which forces air into the intake manifold. The use of turbochargers promotes the burning of fuel and PM, but also increases the formation of NO_x. In a variable geometry turbocharger, the vanes of the turbine can be actuated to vary the angle, varying the flow. The EGR system uses a valve to recirculate exhaust gas into the intake manifold, reducing its oxygen content and, conversely, reducing the formation of NO_x.

Control of turbochargers and EGRs is problematic because, as well as having to balance the use of the two to keep both emissions below an acceptable limit, flow rates through the turbine and EGR are also both dependent on pressure differentials across them, and so both affect each other [25]. Control valves also bring the complication of static friction (or stiction), which introduces a nonlinearity into their dynamics even before considering the influence of pressure variations on either side of the valve. This motivates the investigation of controller tuning to compensate for stiction [26], as well as for multivariate design objectives [27]. This in part has contributed to the increasingly complicated calibration process for engines. The techniques adapted in this thesis aim at simplifying and automating this process.

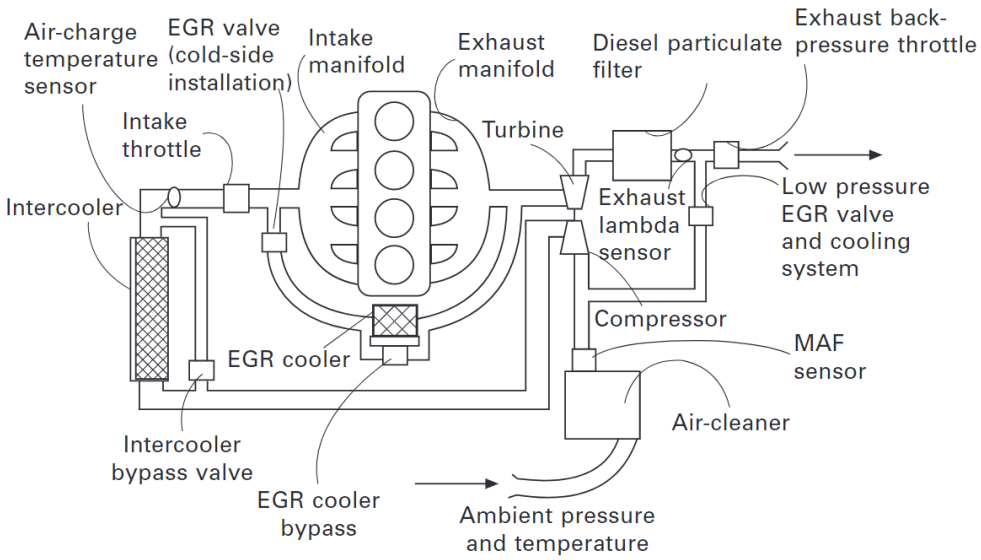

Figure 2.5: Overview of a diesel air-path management system [24].

2.2 Static DoE-based calibration

The need for meeting stringent emission and consumption targets has encouraged engine manufacturers to embed an increasing number of sensors and actuators within the engine itself, so that the combustion process can be controlled more accurately [28]. The data collected by these sensors is fed into the engine's control unit (ECU) which uses such information to control the actuators. Traditionally, such control task is implemented via a series of data and look-up tables, whose entries are filled manually by skilled calibration engineers [1].

When calibrating a production engine, it is standard practice to perform such calibration at a set of operating points defined in terms of engine torque and engine speed [29]. The choice of number and location of such operating points is critical for ensuring good overall performance, as too few (or ill-placed) points will lead to poor performance, whereas too many points would lead to larger look-up tables and more complex control strategies. Design of Experiments (DoE) methods are therefore used to select a set of operating points capable of representing the whole operating envelope of the engine without requiring testing of every possible combination of torque and speed [30]. DoE methods have been successfully exploited to calibrate several types of engines and to gain a better understanding of subsystems within the engine [31], [32]. Figure 2.6 shows an example of maps obtained following this approach, whereas a detailed example of the procedure used in traditional static DoE-based calibration can be found in [1].

In a traditional static DoE calibration procedure, the engine is driven to steady state at each operating point. Once the steady state is reached, the relevant look-up tables and controller gains are calibrated manually, resulting in a very time-consuming process. Moreover, as the number of sensors and actuators (and hence dimension of look-up and gain tables) increases, the time required to perform such tasks significantly increases. The difficulty and costs associated to calibration is well recognised in the literature [2], [4] and it is the main reason why calibration often appears on the critical path for engine production [1].

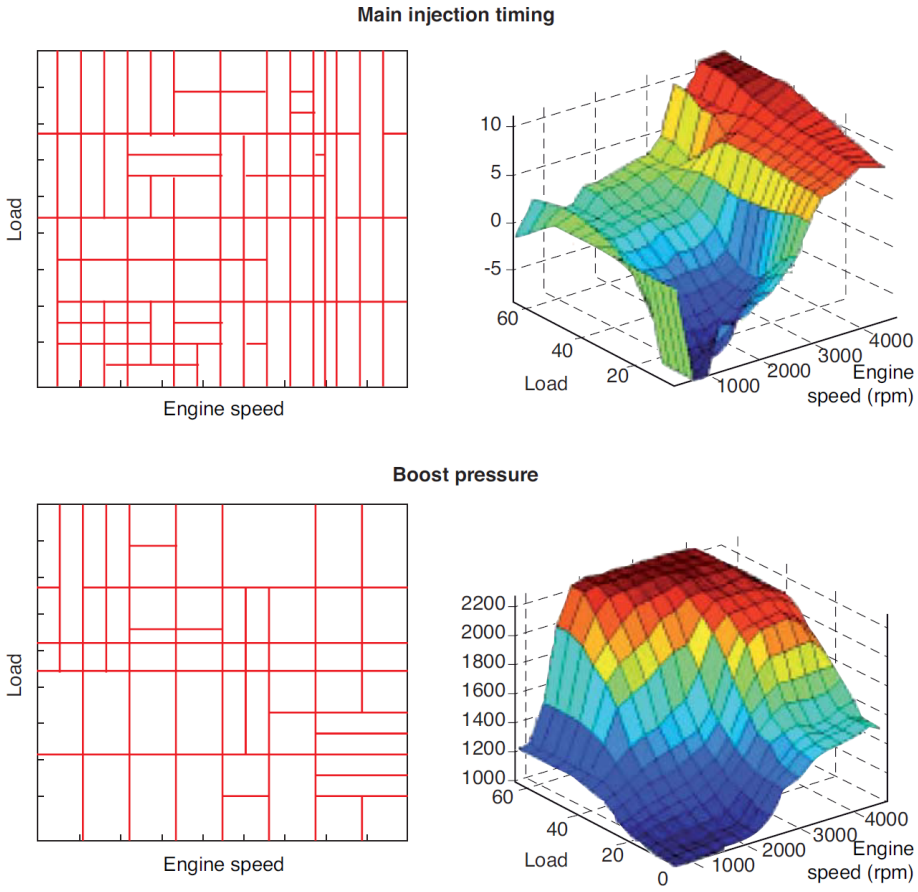

Figure 2.6: Example of calibration maps obtained via a static DoE approach [32]

Furthermore, systems added to control engine emissions also exhibit highly nonlinear dynamics as well as non-minimum phase behaviour, where the direction of response is initially in the opposite direction than is expected [3]. The inherently dynamic behaviour of non-minimum phase systems can not be characterised using a static DoE as such behaviour manifests itself only during transient responses. Similarly, it is difficult to use static DoE to accurately calibrate highly nonlinear systems, e.g. valves with strong stiction such as the ones described in Chapter 4, as the steady-state response is strongly affected by what happens during transients. Therefore, performing manual DoE-based calibration on these systems is extremely challenging.

2.3 Dynamic calibration

Dynamic calibration has been proposed as a solution to significantly reduce complexity and cost associated with traditional static DoE. By considering both transient and steady state responses during the calibration process, the data collection phase can be much shorter as no settling time is required. Moreover, the derived models used for control design at a later stage are more accurate because they can capture the dynamic response of the engine. Additional benefits include [33], [34]:

- Intrinsic data smoothing to achieve good driveability and to promote controller optimization;
- Intrinsic interpolation capability;
- The integration of control processes within the calibration activity.

Prior work conducted at the University of Liverpool has investigated the use of dynamic feedforward optimal controllers to calibrate whole engines, see for example [35]–[37]. This activity has proven that the effort required to calibrate the engine using dynamic calibration is significantly reduced thanks to the use of dynamic controllers and time-series data. This can be compared to static DoE [1], where each measurement is taken at steady state, thus requiring the engine to first settle to a steady stage for each operating point. Fang [35] proposed a dynamic calibration approach based on optimisation of parameters in a feedforward map in order to minimise tracking error and fuel consumption. Such an approach creates the feedforward "Dynamic Calibration Map" shown in figure 2.7 based on the following variables:

- Demand torque, T_d;
- Brake torque, T_b;
- Rate of formation of nitrogen oxides, NO_x;
- Rate of formation of particulate matter, PM;
- Fuel injection per engine cycle, INJ;
- Start of injection angle, SOI;
- Exhaust gas recirculation valve percentage opening, EGR;

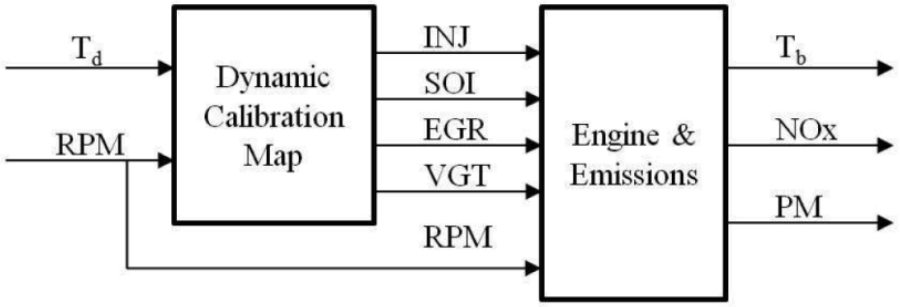

Figure 2.7: Calibration scheme based on feedforward maps proposed in [35]

- Variable geometry turbocharger percentage deflection, VGT;
- Engine speed, RPM.

The feedforward Dynamic Calibration Map strategy uses demand torque T_d and engine speed RPM to minimise NO_x, PM and the difference between T_d and T_b by adjusting INJ, SOI, EGR and VGT. Note that in this case, knowledge of a good engine model, or use of a real engine during controller calibration, is essential, as the feedforward controller cannot cope with significant discrepancies between modelled and real engine behaviour. An alternative approach was proposed by Ostrowski [37] and was based on developing a feedforward dynamic controller via inverse identification, and it was suggested for further work that a method be adapted for combined feedforward-feedback control in order to improve performance. The techniques implemented for this thesis optimally mix feedforward with feedback actions. Similar approaches to calibrate engines using dynamic data have been recently proposed in [38], [39] as well.

A critical step in any dynamic calibration approach is the derivation of an offline model of the system to be controlled, as this allows for an optimal controller to be designed. Note that any model used for optimisation needs to be computationally light, as the optimisation routine will run the same models a large number of times to find the optimum values of the controller gains and/or structure. Neural networks are used extensively in research for both modelling and control of automotive systems [37], [40], [41] due to their relative ease of training and reduced computational complexity, but also due to them being data-driven approaches that are commonly trusted in industry. A review of neural networks is reported in section 2.5 below for reference. Omran, Younes and Champoussin [40] performed calibration by optimising both static and dynamic feedforward maps using neural networks in order to maximise engine power while minimising particulate matter (PM). They also compared the dynamic neural network controller to the results generated by the ECU using statically calibrated maps [40] and found that the effective power of the engine was increased while particulate matter formation, represented by the detected opacity of the exhaust gas, was reduced. Neve *et al.* used radial basis function networks to estimate the optimal engine maps native to the stock engine control unit (ECU) [41], the advantage of which is that the ECU architecture need not be changed.

Neural networks and machine learning have mostly been used only to model certain aspects of the engine response that are hard to capture with physics-based models, such as fuel consumption and NO_x production [42], soot emissions [43], generation of particulate matter [44] and volumetric efficiency [45], just to cite a few

examples. Whole engine models using neural networks have also been developed for engine calibration [46], [47] and hardware-in-the-loop testing [48].

However, there are limitations to how neural networks can be used for practical engine control and modelling. One drawback is that a neural network that is sufficiently complex to capture the strong nonlinear behaviour of an engine will extrapolate outside its training range poorly, so care must be taken to collect data for behaviour across the whole operational range of the system, which can be difficult to define in practice. They must also be non-adaptive, in part because an adaptive neural network controller could behave unpredictably, but also because ECU hardware is limited to production specifications, which typically only have limited processing capability which can be dedicated to any one controller. Real-time adaptive training may exceed this capability. Furthermore, the instruction set of an ECU may not support certain controller or model features to be installed at all.

Once a model of the system has been derived (for example in the form of a neural network), the simplest control strategy would be to "invert" that model and implement a so-called *feedforward controller*. Such a controller derives the value of the engine inputs so that the engine outputs match the demand signals. However, any discrepancy between the model and the real engine would significantly affect the performance of this control scheme, as the controller does not use any real-time information about the engine itself. For this reason feedforward control is limited by uncertainty in the system being controlled [49].

For example, pollutants such as particulate matter (PM) are notoriously difficult to model due to their nonlinear and stochastic nature [50], making physics-based models highly approximate. While physics-based models relating the production of NO_x chemicals to local environmental variables exist, its behaviour also presents a challenge to data-driven modelling due to strongly nonlinear behaviour. For example, the proportional relationship is given by [20]

$$\frac{d[NO]}{dt} \propto \frac{6 \times 10^{16}}{T^{\frac{1}{2}}} \exp\left(\frac{-69090}{T}\right) [O_2]_e^{\frac{1}{2}} \tag{2.3}$$

Where $[O_2]_e$ represents local equilibrium concentration of oxygen, and T denotes local temperature. Equation (2.3) indicates proportionality to three nonlinear terms, one of which is an exponential function with a large exponent within reasonable temperature ranges for a diesel engine cylinder, making it highly nonlinear. Due to the uncertainty in models derived for NO_x and PM, feedback of measurements to the controller are typically necessary to keep predictions realistic. Therefore, feedforward control schemes aimed at minimising PM and/or soot do not usually achieve good performance. For this reason, pure feedforward calibration strategies do not provide satisfactory performance in emission minimisation, which is one of the objectives for this thesis.

Besides the challenges of deriving dynamical data-driven models for highly nonlinear or stochastic systems, controlling them is also made difficult by the relationship between the production rates of NO_x and PM themselves. Figure 2.8 plots the concentrations of soot (which correlates with PM [50]) and NO_x against oxygen concentration. It is shown that PM correlates *negatively* with oxygen concentration [51], while NO_x correlates *positively* with oxygen concentration, which agrees with equation (2.3). As the aim of such a controller should be to minimise these emissions, the calibration procedure becomes a compromise to keep both both below an

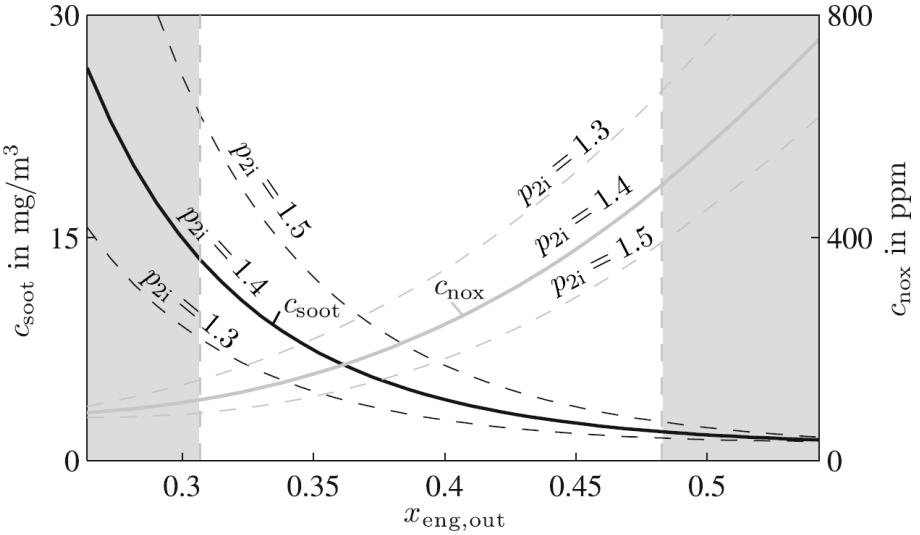

Figure 2.8: Trade-off between NO_x and PM, taken from [51].

acceptable level.

On the other hand, as outlined in [52], feedback control uses real-time sensor data to gather information about the internal state of the engine. By minimising the error between the demand signals and the corresponding measurable outputs using feedback, any discrepancy between the actual engine response and model response due to uncertainty is minimised. Therefore, control performance is less dependent on the error in the model used to derive the controller. A simple controller comprising four linear PID feedback loops is demonstrated in [53], where feedforward control is only provided for a portion of the target output set, showing good results. A similar approach using linear parameter-varying (LPV) gain scheduled controllers was proposed in [54], using measurable outputs as feedback. A controller proposed by [55] demonstrates the benefit of direct feedback of NO_x measurements compared to typical methods, as traditionally such measurements are either too slow or unavailable in production engines.

Of course, implementation of feedback control requires the desired outputs to be measurable, and this is a major drive for the increasing number of sensors installed on diesel engines [56] [57]. For example, emissions sensors are not traditionally built into production engines and, even when they are installed to implement NO_x feedback [55], they are slow to update and may have unfavourable dynamics. Particulate matter (PM) sensors remain unavailable to production engine units, and as such an observer needs to be designed to estimate PM emissions from other signals, and then use such estimate to implement feedback control.

A combined feedforward-feedback controller, as shown in Figure 2.9, allows for corrections to be done in real-time, while at the same time the feedforward action can drive fast closed-loop response. This scheme is capable of accounting for uncertainty in the system behaviour, which includes unknown disturbances and error in the models used to generate the controller. Uncertainty also originates from the difference in operating environment when an engine is calibrated on a testbed versus running in a car, which feedback can help alleviate. Linear feedback control methods such as H_2 and H_∞ [58] have been adapted to highly nonlinear systems including diesel engines through gain scheduling as in Wei and del Re [59], and also jet engine

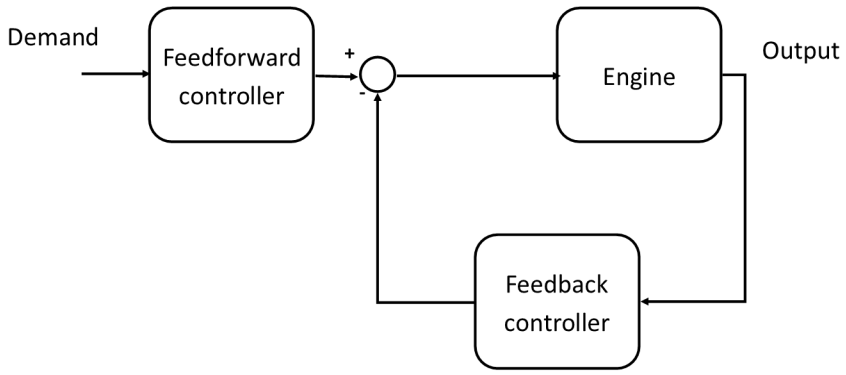

Figure 2.9: Combined feedforward-feedback control architecture.

compressors as in Hardt, Helton and Kreutz-Delgado [60].

Shamma discusses gain scheduling in [10], where multiple linear controllers are defined for different operating points and then interpolated so that their collective response approximate a nonlinear controller. The main benefit of gain scheduling is that traditional linear control design techniques can be used to control highly nonlinear systems. The downsides associated with gain scheduling include that, in order to closely approximate a nonlinear controller, enough local linear controllers need to be derived. In the automotive industry sometimes it may not be possible to represent enough operating points, especially on an ECU which is not particularly powerful from a computational and a memory storage point of view. Another notable downside is that for gain scheduling to work well, the scheduling variable should be slowly changing in comparison to the controlled variable. This is not always possible, e.g. for valves with significant stiction [61] whose nonlinear behaviour is dependent on the deflection of the valve itself. This means that in practice, engine valves have controllers scheduled by position, the same quantity that must be controlled, thus violating one of the main assumptions of gain scheduling techniques.

2.4 System identification

System identification comprises methods which develop models from experimental measurements. This is useful when complex physical phenomena prevent the accurate modelling of a system from fundamental physical principles, or when physics-based models will be too complex to run in real-time. Moreover, as mentioned in the previous section, accurate models are required to design a controller that achieves robust performance over a range of operating conditions, as control performance are strictly linked to the quality of the models used during the design of the controller. Furthermore, optimisation-based control design techniques require models to be run a large number of times during optimisation, therefore the availability of offline models of reduced computational complexity is key to make the use of such techniques viable. A general procedure for system identification is summarised in figure 2.10, taken from [37].

A very common class of models used for system identification purposes is the Nonlinear AutoRegressive Moving Average with eXogenous input (NARMAX) model,

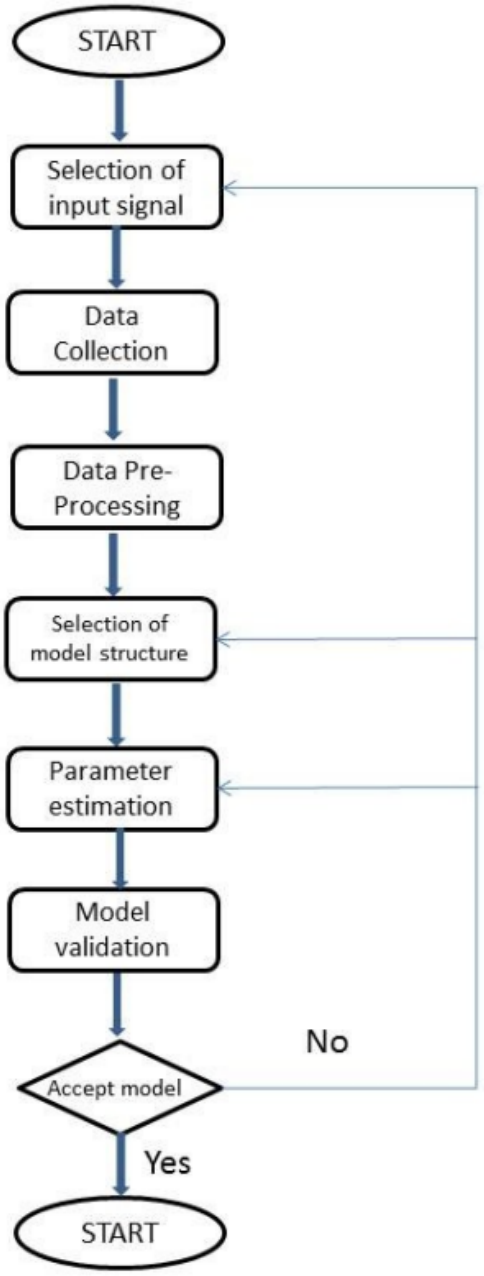

Figure 2.10: General procedure for system identification [37]

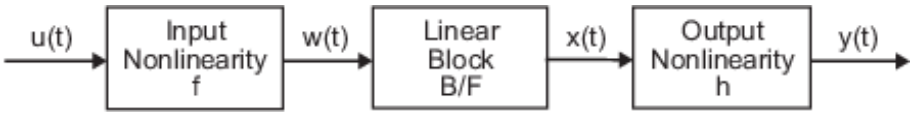

Figure 2.11: Hammerstein-Wiener model [67]

where the output at time step k is expressed as a function of past input, output and noise terms as

$$\begin{aligned} y(k) = f(&y(k-1), y(k-2), \ldots, y(k-q), \\ &u(k-1), u(k-2), \ldots, u(k-p), \\ &e(k-1), e(k-2), \ldots, e(k-r)) + e(k) \end{aligned} \tag{2.4}$$

where y is the output, u is the input and e is the system noise. In the linear case, this simplifies to an ARMAX model [62] of the form

$$\begin{aligned} y(k) =& a_1 y(k-1) + a_2 y(k-2) + \cdots + a_q y(k-q) \\ &+ b_1 u(k-1) + b_2 u(k-2) + \cdots + b_p u(k-p) \\ &+ e(k) + c_1 e(k-1) + c_2 e(k-2) + \cdots + c_r e(k-r) \end{aligned} \tag{2.5}$$

NARMAX models have been used to derive models of engines, see for example [63]. They have been specifically applied to the modelling of NO_x emissions using air-path inputs in [64], which is relevant to the aim of minimising emissions. It is notable that these methods have also been used for development of virtual sensors for NO_x as in [65], and as a motivation for the work in [66] about the formation of particulate matter. Virtual sensors take the place of sensors not present on production engines for the purposes of feedback for control systems.

Neural networks are also a popular framework for system identification, but implementation in real-time in the automotive industry has sometimes been reported as a problem for large neural networks. To overcome this issue, Ostrowski [37] investigated the use of Hammerstein-Wiener models, whose structure is shown in figure 2.11, to simplify the controller when implemented on the ECU. The distinguishing feature of a Hammerstein-Wiener model is its linear dynamic model, B/F, sandwiched between two static nonlinear functions, f and h. The static functions can be of polynomial or lookup table format, both of which are already used in standard ECU architecture. Although more modern approaches for system identification such as neural networks and fuzzy models have been proposed [33], they often take too many ECU resources. Therefore, classical models such as Hammerstein-Wiener can still be used to approximate, for example, the obtained neural network and make the identified model compatible with resource-limited ECUs.

Wahlström and Eriksson have developed a mean-value diesel simulated engine which was well-validated against a 12L Scania engine on a testbed [68]. This allows whole-engine control methodology to be developed offline more quickly before applying it to a real engine. However, the simulated engine did not include emissions models, so for the purpose of this thesis NO_x and particulate matter (PM) models were adapted from [20] and [50] respectively using chemical phenomenological models to augment the simulated engine. It is shown in [69] that PM is particularly sensitive to the full crank angle domain temperature history inside the cylinder, which is not modelled by the Wahlström model. The temperature model described

by [70] provides this, but proved unfeasible to implement in the whole engine model. This was due to large differences in required time step for each, with crank angle varying much more quickly than the mean-value models comprising the simulated engine (see also chapter 6). Due to access to the real engine testbed at Jaguar Land Rover being denied early in the PhD project, the Eriksson simulated engine is treated as a real engine for all whole-engine modelling and control in this thesis. As the relevant control methods described in chapters 5 and 6 rely on producing black-box offline models of the real engine, a distinction is made between the *simulated engine* produced by Wahlström and Eriksson and any *engine models* derived from it via system identification.

A key step for successful system identification and calibration is deciding whether the system to be controlled exhibits linear or nonlinear behaviour. Lee, White and Granger [71] compared multiple identification frameworks, including neural networks, to develop a systematic approach for deciding if the system to be modelled/controlled is linear or nonlinear. For the neural network method, a neural network with linear and nonlinear components was trained on a rich time-series. The system was considered nonlinear if the nonlinear weights were non-zero and of comparable size to the linear weights [71]. Knudsen describes a statistical F-test method based on the superposition principle for linear systems [72]. This method is useful for selecting controller complexity, or deciding if a linear controller will suffice over a complex nonlinear controller. An extension of this method to make it applicable to automotive systems is presented in chapter 3.

Moreover, similar to a DoE, in dynamic calibration the dynamic test sequence needs to drive the system through a representative operating envelope. It is well known that to obtain an accurate model the excitation sequence used for calibration needs to be "persistently exciting" [73], meaning that it must excite all of the frequencies in the range of interest. Moreover, when identifying nonlinear systems, the input must be adequately rich also in amplitude so that the full amplitude range of the nonlinearities is also explored [74]. This has often been achieved using amplitude-modulated pseudo-random binary sequences (APRBS), see for example the inverse dynamic feedforward control method developed by [37], and and example of an APRBS signal is shown in figure 2.12. An APRBS works well for this purpose, as it changes at random time intervals and to random amplitudes, and is generated according to [75]. This means that it is capable of exciting multiple dynamic modes and of exploring the system nonlinearity by varying amplitude [76], as required for nonlinear systems [52].

However, the intrinsic stochastic nature of APRBS signals - not exploiting any knowledge of the system to be identified - implies that long test sequences may be needed to ensure that the whole operating envelope is explored. To overcome this issue and decrease the time needed for identification, optimisation of the test signal can be performed. For example, a dynamic analogue of static DoE is proposed in [77], where an augmented Lagrangian pattern search algorithm is used to optimise an APRBS to increase its information content. This was used to train a dynamic neural network torque model and was found to improve model performance significantly. Optimal test signals could be used for both model and controller training and is a logical next step for further work by reducing the calibration time even further. This is investigated in Chapter 7 for a simple SISO system and for a MIMO simulated engine.

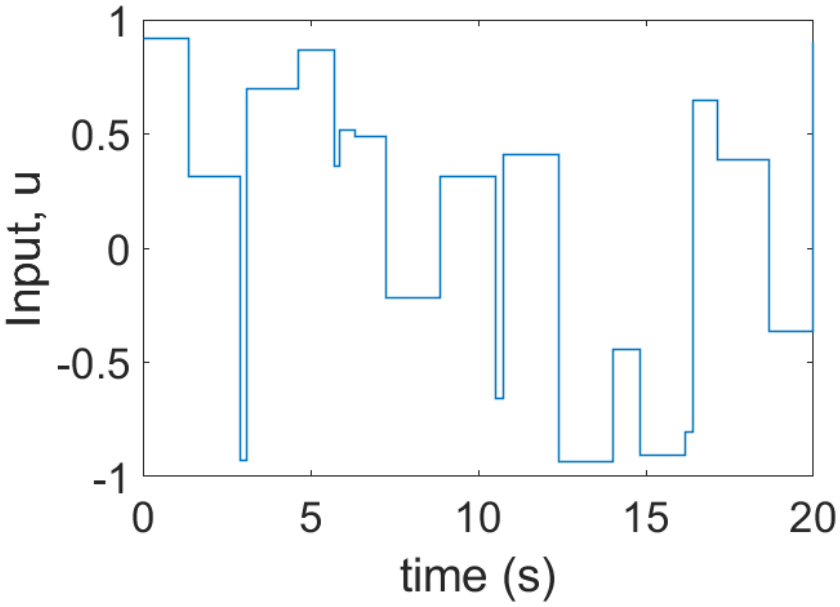

Figure 2.12: An amplitude modulated pseudo-random binary signal (APRBS).

Finally, quantifying the accuracy of the model in describing the behaviour of the real system is crucial for choosing the best model for a given application. The main metrics used for validation in this thesis are R^2 and MSE, defined respectively as [78]

$$R^2(Y, \hat{Y}) = 1 - \sum_{i=1}^{n} \frac{(\hat{Y}_i - Y_i)^2}{(Y_i - \bar{Y}_i)^2} \tag{2.6}$$

$$MSE(Y, \hat{Y}) = \sum_{i=1}^{n} \frac{(\hat{Y}_i - Y_i)^2}{n} \tag{2.7}$$

where $\hat{Y}$ is the modelled value of Y, and $\bar{Y}$ is the mean value of Y. While MSE is a dimensional measure of error and needs to be minimised, R^2 ranges between $-\infty$ (poor fit) and 1 (perfect fit). Both metrics have been used to evaluate goodness of fit of engine models in [36].

An alternative metric demanded in some industrial contexts is mean absolute percentage error $MAPE$, defined as

$$MAPE(Y, \hat{Y}) = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{\hat{Y}_i - Y_i}{Y_i} \right| \tag{2.8}$$

$MAPE$ represents a percentage error and its interpretation is readily available for the user. However, it suffers from the drawback is that for measured values close to zero, $MAPE$ tends to infinity. This makes it a non reliable metrics for data sets which include a value of zero, or small numbers approaching zero, as common in this thesis.

2.5 Neural networks

The structure of artificial neural networks, as used for system identification, is based on the biological neural networks comprising the animal brain. The biological neu-

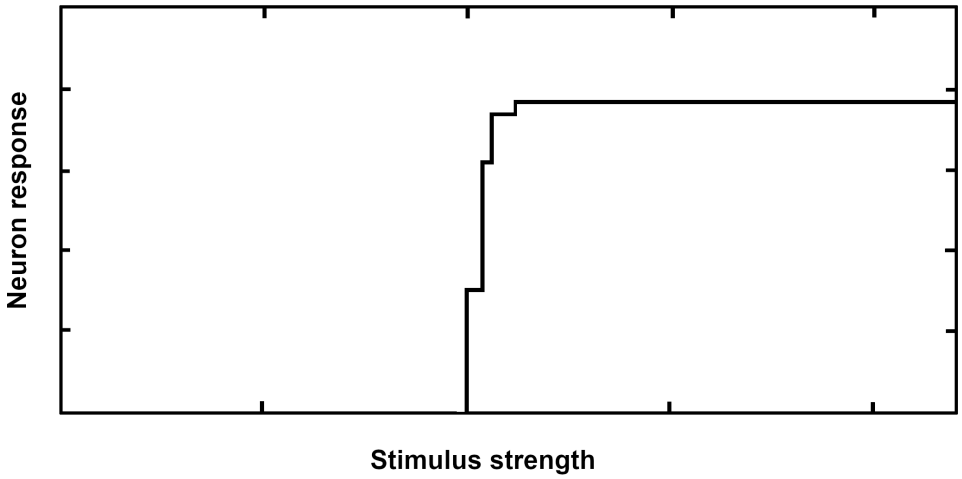

Figure 2.13: Relation between biological neuron response and stimulus strength, digitised from experiments by Lucas [80].

ron, or the nerve cell, is the basic processing unit of the brain which reacts to stimuli from connected neurons. Of interest is that biological neurons do not respond to stimuli in a proportional manner; a larger stimulus above the threshold of the neuron does not result in a larger response. Instead, neurons are "activated" when a stimulus exceeds a threshold, so they exhibit the so called "all-or-none" response [79] illustrated by experimental results in figure 2.13 [80]. By arranging the connections between biological neurons in a network, complex behaviour can be produced. Hebb's rule [81] states that repeated and persistent excitation of one neuron by a connected neuron increases the efficiency with which that excitation happens in future, the basis on which behaviours can be learned.

It was proposed by McCulloch and Pitts [82] that, due to the "all-or-none" response of neurons to stimuli, events in a neural network can be treated by propositional logic. The McCulloch-Pitts (MCP) model is a mathematical construct, based on the description of biological neural networks, which exploits such character to perform logical decisions. The basic unit of the MCP model is the McCulloch-Pitts neuron, which is shown diagrammatically in figure 2.14. Each input u_i is multiplied by a corresponding weight w_i, all the weighted inputs are then summed together and compared to an activation threshold value β. If the result is greater than the threshold, the neuron is fired and the output y is 1. Otherwise, the neuron does not fire and the output y is 0. As both input and output are boolean values, the MCP neuron mirrors the "all-or-none" behaviour of the biological neuron. This behaviour is given by

$$y = \begin{cases} 0 & \sum_{i=1}^{n} w_i u_i \leq \beta \\ 1 & \sum_{i=1}^{n} w_i u_i > \beta \end{cases} \tag{2.9}$$

The simplest possible neural network is the MCP model with a single neuron. Setting aside the question of how to determine the weights and biases of the model, to be fit for purpose the model should be able to represent an arbitrary boolean function. However, Worden [83] showed that, in this form, the MCP model is insufficient to

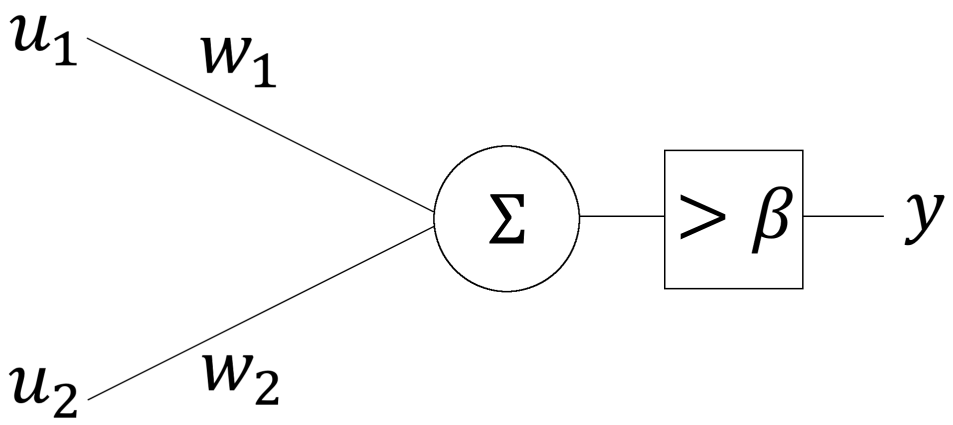

Figure 2.14: The McCulloch-Pitts neuron with two inputs.

accurately reproduce all simple logical operators. Take, for example, a two-input MCP model given by

$$y = \begin{cases} 0 & w_1 u_1 + w_2 u_2 \leq \beta \\ \\ 1 & w_1 u_1 + w_2 u_2 > \beta \end{cases} \tag{2.10}$$

In this simple example, the weights and threshold of the model are to be set in such a way that the output is the result of the logical equality operator; The output should be 1 if both inputs are equal and should be set to 0 if the inputs are distinct. In table 2.2 all four possible combinations of boolean inputs u_1 and u_2 are tabulated. The result of the comparison $u_1 = u_2$ is given according to Gensler [84] along with the conditions which must be met for the MCP model to return the correct result.

Table 2.2: Truth table for the equality operator [84]

Case	u_1	u_2	$\{u_1 = u_2\}$	$\{w_1 u_1 + w_2 u_2 > \beta\} = \{u_1 = u_2\}$ if:
(a)	0	0	1	$0 > \beta$
(b)	0	1	0	$w_2 \leq \beta$
(c)	1	0	0	$w_1 \leq \beta$
(d)	1	1	1	$w_1 + w_2 > \beta$

To satisfy cases (a) and (b) it follows that

$$0 > \beta \geq w_2 \tag{2.11}$$

which imposes a negative value of w_2. Similarly, w_1 must be negative due to (a) and (c), which demands that

$$0 > \beta \geq w_1 \tag{2.12}$$

This means that the sum of the two weights must be less than the threshold (and less than zero),

$$w_1 + w_2 < \beta \tag{2.13}$$

However, this contradicts case (d). Therefore, in this format the single neuron MCP model cannot accurately perform the given operation for all possible cases, no matter

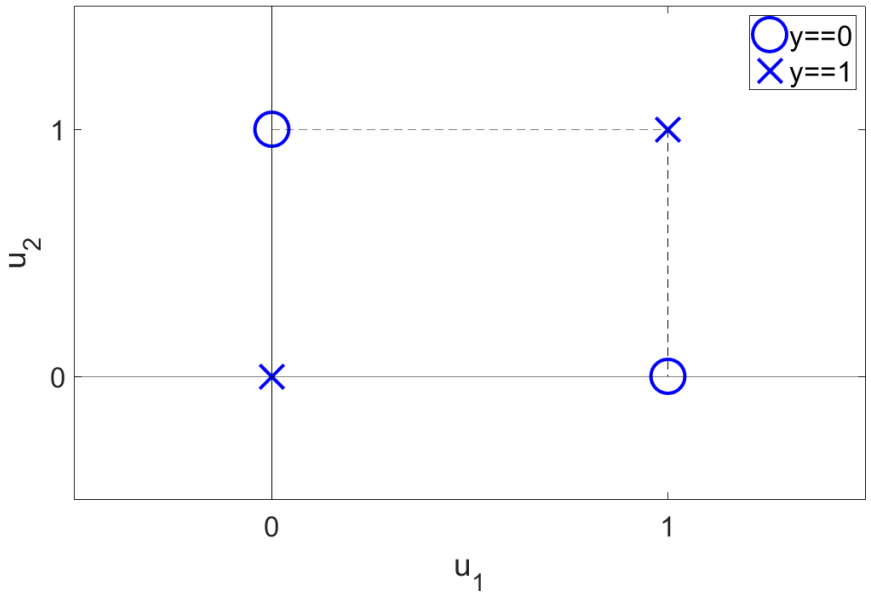

Figure 2.15: The outcome of the equality operation in input space

what the weights and threshold are set to. While this example examines the two-input, i.e. two-dimensional, case, it can also be shown that the same conclusion is valid for the general n-dimensional case. Indeed, let us consider a function with n inputs given by

$$l(u_1, u_2, \ldots, u_{(n-1)}, u_n) = \sum_{i=1}^{n} w_i u_i - \beta \tag{2.14}$$

If each input is considered a dimension in n-dimensional space, there exist two regions in that space: one where the value of l is positive and the function is true, the other where the value of l is negative and the function is false. These regions are separated by a hyperplane where

$$0 = \sum_{i=1}^{n} w_i u_i - \beta \tag{2.15}$$

which corresponds to the boundary between the two outcomes of the equation (2.9). Therefore, rendered in input-space, there must exist an MCP model whose boundary is a hyperplane bisecting the true and false outcomes of the operator being modelled. In the two dimensional case given by equation (2.10), this hyperplane is reduced to a straight line through the input space described by

$$0 = w_1 u_1 + w_2 u_2 - \beta \tag{2.16}$$

Representing the outcome of the logical equality operator used in the earlier example in input space as in figure 2.15, the problem can be seen: there is no line that can be drawn which divides the true and false outcomes of the equality operator into distinct regions. As this is a requirement of equations (2.9) and (2.14), the conclusion is that no matter what values of w_1, w_2 or β are used, the MCP model cannot accurately represent logical equality.

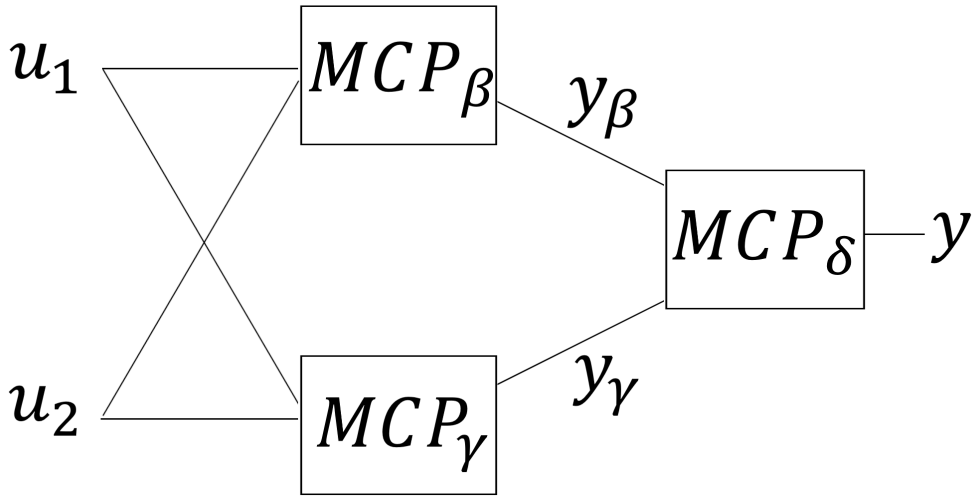

Figure 2.16: A network of MCP neurons

There are 2^n points in the domain of an n-input boolean function, with two possible values (0 or 1). So, for n inputs there are $(2^2)^n$ possible functions. Therefore, there are 16 possible boolean functions in the two-input domain. In fact, only two of these cannot be representing by the single neuron MCP model, equality and "Exclusive Or" (XOR). As the number of inputs increases, the number of possible distinct functions increases exponentially and the single neuron MCP model becomes increasingly unable to represent the growing number of functions. Using multiple MCP units in a network solves this problem, which allows the definition of multiple hyperplanes bounding true and false function values in the input space [85].

To show this, let us consider a network of three layers: the input layer, one "hidden" layer, so called because it does not communicate with anything outside the network, and one output layer which produces the final output of the network. The input layer serves only to distribute the inputs to the rest of the network, and has no adjustable parameters. The hidden layer consists of two MCP neurons, and the output layer consists of one MCP neuron. Each neuron is of the same format as shown in figure 2.14. The three MCP neurons are denoted as:

1. MCP_β, the first neuron of the hidden layer;
2. MCP_γ, the second neuron of the hidden layer;
3. MCP_δ, the output layer neuron.

The output of hidden neuron MCP_β is given by

$$y_\beta = \begin{cases} 0 & w_1u_1 + w_2u_2 \leq \beta \\ 1 & w_1u_1 + w_2u_2 > \beta \end{cases} \tag{2.17}$$

The output of hidden neuron MCP_γ is given by

$$y_\gamma = \begin{cases} 0 & v_1u_1 + v_2u_2 \leq \gamma \\ 1 & v_1u_1 + v_2u_2 > \gamma \end{cases} \tag{2.18}$$

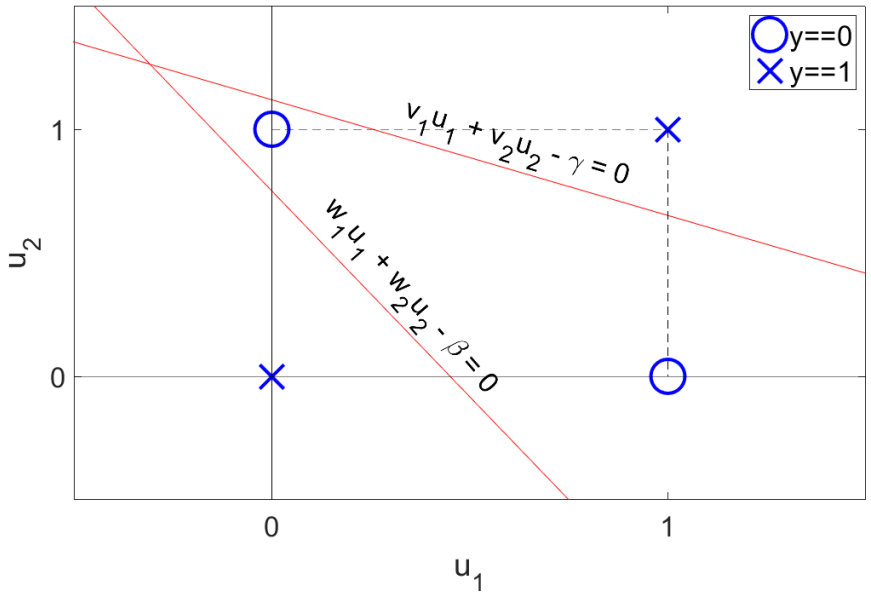

Figure 2.17: The outcome of the equality operation in input space with the bounding lines of each neuron represented.

The outputs of both hidden neurons are fed as inputs to the output neuron MCP_δ and the final output is given by

$$y_\delta = \begin{cases} 0 & z_1 y_\beta + z_2 y_\gamma \leq \delta \\ \\ 1 & z_1 y_\beta + z_2 y_\gamma > \delta \end{cases} \tag{2.19}$$

For the two-input equality example, figure 2.17 shows that the input space can be divided by two separate lines corresponding to the two hidden layer neurons. The output neuron effectively selects the appropriate region, allowing regions of true and false function values to be separated by the three neuron model. It is notable that, as long as the regions of true and false are defined correctly, the weights and threshold of each neuron have infinitely many valid values.

Having defined the MCP neuron, and addressed its limitations with the multi-layer structure, leads to the generalisation of the "multi-layer perceptron" (MLP). This is defined by Rosenblatt as a feedforward network with one or more hidden layers and an output layer [85]. The activation function, given earlier by the discontinuous boolean equation (2.10), is generalised to be any nonlinear transfer function f. Each k-th layer of the network takes the output of the previous $(k-1)$-th layer as input, the output of each layer with N_k neurons is an $N_k \times 1$ vector given by

$$\hat{\boldsymbol{y}}_k = \begin{bmatrix} \hat{y}_{k,1} \\ \hat{y}_{k,2} \\ \vdots \\ \hat{y}_{k,(N_k-1)} \\ \hat{y}_{k,N_k} \end{bmatrix} \tag{2.20}$$

For a network with a total of L layers, indicating $L-1$ hidden layers, the output of

a layer k with N_k neurons and input $\hat{\boldsymbol{y}}_{(k-1)}$ is given by

$$\boldsymbol{y_k} = \begin{bmatrix} f\left(\boldsymbol{w_{k,1}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,1}\right) \\ f\left(\boldsymbol{w_{k,2}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,2}\right) \\ \vdots \\ f\left(\boldsymbol{w_{k,(N_k-1)}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,(N_k-1)}\right) \\ f\left(\boldsymbol{w_{k,N_k}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,N_k}\right) \end{bmatrix} \tag{2.21}$$

where $\boldsymbol{w_{k,n}}$ for a given neuron n is an $N_{(k-1)} \times 1$ vector, the same dimensions as the layer input $\hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}}$, such that their scalar product can be calculated in equation (2.21).

The network structure described so far is known as "feedforward neural network", as all connections are fed towards the output layer. On the other hand, a "feedback neural network" can have connections which loop back to previous layers. Note that in the case of $k = 1$, i.e. in the first layer, the input $\hat{y}_{(k-1)}$ is the external input to the network, as no previous layer exists but the input. A bias $\boldsymbol{b_k} = [b_{k,1}, b_{k,2} \dots]^T$ is added to the weighted sums to allow a constant offset in the activation function argument. It has been shown that networks of neurons can represent arbitrary functions [83], so long as the network has a sufficient number of neurons and its weights are set appropriately.

For a data set of target outputs y with t_{max} elements, an appropriate training algorithm would minimise the difference between the measured data set and the final output of the network, $\hat{y}$, making it an example of supervised learning [33]. The typical cost function to be minimised during training is given by as the sum of squared errors over the whole data set

$$J = \frac{1}{2} \sum_{t=1}^{t_{max}} \delta(t)^2 \tag{2.22}$$

where δ is defined as the difference in desired output y and predicted output $\hat{y}$

$$\delta(t) = y(t) - \hat{y}(t) \tag{2.23}$$

and generally for each layer k with desired layer output $\boldsymbol{y_k}$,

$$\boldsymbol{\delta}_k(t) = \hat{\boldsymbol{y}}_{(k+1)}(t) - \hat{\boldsymbol{y}}_k(t) \tag{2.24}$$

For simplicity of notation, in the following it is assumed there is a single neuron in the final layer, resulting in a multi-input single-output (MISO) system. The training algorithm must minimise the cost function through nonlinear optimisation due to the presence of the nonlinear transfer function f in (2.21). The parameters $\boldsymbol{\theta}$ of the neural network layer k to be adjusted to minimise (2.22) are a collection of the

connection weights and biases for the layer, i.e.

$$\boldsymbol{\theta}_k = \begin{bmatrix} \boldsymbol{w}_{k,1} \\ \boldsymbol{w}_{k,2} \\ \vdots \\ \boldsymbol{w}_{k,(N_k-1)} \\ \boldsymbol{w}_{k,N_k} \\ \hline b_{k,1} \\ b_{k,2} \\ \vdots \\ b_{k,(N_k-1)} \\ b_{k,N_k} \end{bmatrix} = \begin{bmatrix} \theta_{k,1} \\ \theta_{k,2} \\ \vdots \\ \theta_{k,(2N_k-1)} \\ \theta_{k,2N_k} \end{bmatrix} \tag{2.25}$$

and the set of all weights and biases for all layers is denoted θ

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_k \\ \vdots \\ \boldsymbol{\theta}_{(L-1)} \\ \boldsymbol{\theta}_L \end{bmatrix} \tag{2.26}$$

To minimise the cost function (2.22), the parameters of the neural network are adjusted iteratively by gradient descent, as is standard in nonlinear optimisation [86]. Considering the gradient operation defined by (2.27), the direction of *steepest descent* $\boldsymbol{p}_k$ in parameter space is a unit vector given by (2.28), as described by [87]

$$\nabla_k J = \begin{bmatrix} \frac{\partial J}{\partial \theta_{k,1}} \\ \\ \frac{\partial J}{\partial \theta_{k,2}} \\ \\ \vdots \\ \\ \frac{\partial J}{\partial \theta_{k,(2N_k-1)}} \\ \\ \frac{\partial J}{\partial \theta_{k,2N_k}} \end{bmatrix} \tag{2.27}$$

$$\boldsymbol{p}_k = -\frac{\nabla_k \cdot J}{\|\nabla_k \cdot J\|} \tag{2.28}$$

The steepest possible gradient $\boldsymbol{p}_k$ is negated so that the gradient descends, bringing the cost function J closer to its minimum, hence the name *gradient of steepest descent*. The final change in the parameter step in this direction is given by

$$\Delta\boldsymbol{\theta}_k = \eta \boldsymbol{p}_k = -\eta \nabla_k J \tag{2.29}$$

where the size of the step η is called the learning coefficient in this context [87]. An appropriate value for the learning coefficient η depends on the application, and

its tuning if often performed by trial-and-error by the user [83]. If the learning coefficient is too small, learning is slow and the algorithm may take a long time to converge. If η is too large, the parameters may oscillate or diverge. To mitigate this, an additional term may be introduced so that high-frequency changes are damped, i.e.

$$\Delta\boldsymbol{\theta}_k(i) = -\eta\nabla_k J(i) + \mu\Delta\boldsymbol{\theta}_k(i-1) \tag{2.30}$$

The additional *momentum* term with a small value of μ forces changes from the previous learning iteration $(i-1)$ to persist, therefore mitigating the problem of oscillation.

However, determination of the gradient ∇J is made difficult for multi-layer networks, as the input to each layer is the output of the previous layer, see equation (2.21). Each layer must minimise the difference between its output and its desired output as in (2.24), so the desired output for each layer must be determined. This problem was addressed in Paul Werbos' doctoral thesis [88] with a method called *back-propagation*, which propagates the output error from the final L-th layer backwards through the previous layers. The method is described by Werbos in [89] and briefly summarised in the following. By applying the chain rule to the differential in equation (2.28), the cost gradient can be rewritten as

$$\nabla_k J = \frac{\partial J}{\partial \hat{\boldsymbol{y}}_k}\nabla_k \hat{\boldsymbol{y}}_k \tag{2.31}$$

where

$$\frac{\partial J}{\partial \hat{\boldsymbol{y}}_k} = -\boldsymbol{\delta}_k \tag{2.32}$$

The gradient of $\hat{\boldsymbol{y}}_k$ with respect to the parameter space gradient is defined as

$$\nabla_k \hat{\boldsymbol{y}}_k = \begin{bmatrix} \frac{\partial \hat{\boldsymbol{y}}_{k,1}}{\partial \boldsymbol{w}_{k,1}} \\ \\ \frac{\partial \hat{\boldsymbol{y}}_{k,2}}{\partial \boldsymbol{w}_{k,2}} \\ \\ \vdots \\ \\ \frac{\partial \hat{\boldsymbol{y}}_{k,(N_k-1)}}{\partial \boldsymbol{w}_{k,(N_k-1)}} \\ \\ \frac{\partial \hat{\boldsymbol{y}}_{k,N_k}}{\partial \boldsymbol{w}_{k,N_k}} \\ \\ \text{———} \\ \\ \frac{\partial \hat{\boldsymbol{y}}_{k,1}}{\partial b_{k,1}} \\ \\ \frac{\partial \hat{\boldsymbol{y}}_{k,2}}{\partial b_{k,2}} \\ \\ \vdots \\ \\ \frac{\partial \hat{\boldsymbol{y}}_{k,(N_k-1)}}{\partial b_{k,(N_k-1)}} \\ \\ \frac{\partial \hat{\boldsymbol{y}}_{k,N_k}}{\partial b_{k,N_k}} \end{bmatrix} \tag{2.33}$$

and, by further application of the chain rule, the gradient of the layer output with

respect to the parameter space can be written as

$$\nabla_k \hat{\boldsymbol{y}}_k = \begin{bmatrix} f'\left(\boldsymbol{w_{k,1}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,1}\right) \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} \\ f'\left(\boldsymbol{w_{k,2}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,2}\right) \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} \\ \vdots \\ f'\left(\boldsymbol{w_{k,(N_k-1)}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,(N_k-1)}\right) \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} \\ f'\left(\boldsymbol{w_{k,N_k}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,N_k}\right) \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} \\ \hline f'\left(\boldsymbol{w_{k,1}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,1}\right) \\ f'\left(\boldsymbol{w_{k,2}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,2}\right) \\ \vdots \\ f'\left(\boldsymbol{w_{k,(N_k-1)}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,(N_k-1)}\right) \\ f'\left(\boldsymbol{w_{k,N_k}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,N_k}\right) \end{bmatrix} \tag{2.34}$$

where f' indicates the first derivative of the neuron transfer function. Equation (2.34) highlights an important constraint of gradient descent back-propagation method: the transfer function f must be analytically differentiable. This was one of the limits of boolean activation function MCP neural networks described earlier: their discontinuous behaviour could not be differentiated. Many functions are commonly used, for example a linear function $f(x) = x$ is a typical choice for the output layer, and is trivial to differentiate. For layers with nonlinear transfer functions, the hyperbolic tangent $\tanh x$ (or tangent sigmoid/tansig) is commonly used [83], but other sigmoid-type functions are used too. For example, differentiating the hyperbolic tangent with respect to x yields

$$\frac{d(\tanh x)}{dt} = 1 - \tanh^2 x \tag{2.35}$$

Substituting (2.32) and (2.34) into (2.31) gives the final parameter space gradient

of the cost function. Substituing that into (2.30) gives the update rule:

$$\Delta\boldsymbol{\theta}_k(i) = \eta \begin{bmatrix} f'\left(\boldsymbol{w_{k,1}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,1}\right) \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} \delta_{k,1} \\ f'\left(\boldsymbol{w_{k,2}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,2}\right) \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} \delta_{k,2} \\ \vdots \\ f'\left(\boldsymbol{w_{k,(N_k-1)}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,(N_k-1)}\right) \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} \delta_{k,(N_k-1)} \\ f'\left(\boldsymbol{w_{k,N_k}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,N_k}\right) \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} \delta_{k,N_k} \\ \hline f'\left(\boldsymbol{w_{k,1}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,1}\right) \delta_{k,1} \\ f'\left(\boldsymbol{w_{k,2}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,2}\right) \delta_{k,2} \\ \vdots \\ f'\left(\boldsymbol{w_{k,(N_k-1)}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,(N_k-1)}\right) \delta_{k,(N_k-1)} \\ f'\left(\boldsymbol{w_{k,N_k}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{(k-1)}} + b_{k,N_k}\right) \delta_{k,N_k} \end{bmatrix} + \mu\boldsymbol{\theta}_k(i-1) \quad (2.36)$$

As discussed in section 2.4, neural networks are a popular method when deriving models for use in automotive research. For the purpose of this thesis, the MATLAB Neural Network Toolbox [67] was used to design and train neural network based models and controllers. The toolbox allows networks to be defined by number of hidden layers, the size (number of neurons) of each layer and their respective connections. For time-series training data, each connection can have a set of time step delays so that discrete dynamic system transients can be modelled and controlled, which was necessary for the objective of dynamic calibration in the research discussed in this thesis. An example of neural network created with this toolbox is shown in figure 2.18, where the numbers in the circles represent the time step delays and the numbers outside the boxes indicate how many neurons each box contains.

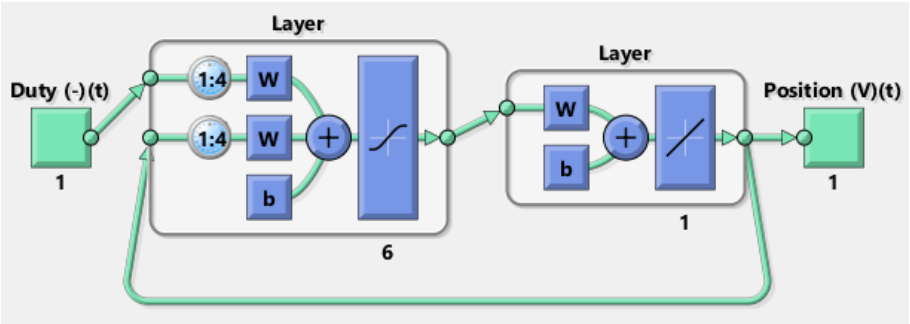

Figure 2.18: Example of neural network, as visualised by the MATLAB Neural Network Toolbox [67].

The Neural Network toolbox also provides a number of training algorithms.

The recommended one for most applications is the *Levenburg-Marquardt Back-propagation* algorithm, which was found to be adequate for modelling and controlling engine systems. This training algorithm modifies Werbos' back-propagation method [88] discussed earlier with the Marquardt algorithm for least-squares estimation of nonlinear parameters [90]. A limitation of the original back-propagation discussed earlier was its tendency to oscillate when learning rate was set too high, and to converge slowly when learning rate was set too low. A solution was given, and then stated by equation (2.30), that introducing the *momentum* term μ helped mitigate the problem, without much guidance on how to set the parameter. The Maquardt modification adjusts μ by factor β as the algorithm iterates. The result is more robust for a constant learning rate η and takes considerably fewer floating point operations to convergence than other implementations of the back-propagation algorithm [91]. The modified algorithm is given as:

1. Compute the sum of squares of errors J as in (2.22);

2. Compute $\Delta\boldsymbol{\theta}$ as in (2.36);

3. Recompute error J for parameter set $\boldsymbol{\theta} + \Delta\boldsymbol{\theta}$. Calculate the change in error:

   (a) If the magnitude of the change is smaller than that of a predetermined tolerance, the algorithm is assumed to have converged;

   (b) If the change is negative, divide μ by β, update $\boldsymbol{\theta}$ by $\Delta\boldsymbol{\theta}$ and go back to step 1;

   (c) If the change is positive, multiply μ by β and go back to step 3;

The MATLAB Neural Network toolbox uses default initial values $\mu = 0.01$, $\beta = 10$ and tolerance for change in error of 10^{-7}. Typically inputs and outputs are normalised to values between -1 and 1 so that the absolute magnitudes of the training data are not a factor in determining these settings and do not significantly affect performance.

2.6 Research gap and thesis contribution

The overview of the current state-of-the-art in engine calibration reported in this chapter has highlighted the great potential of dynamic calibration techniques for reducing the time and cost associated to engine calibration. Therefore, there is a need to develop efficient and simple-to-use dynamic calibration techniques, harnessing the results from dynamical system identification and control, which can be implemented in a production environment. Preliminary steps in this direction have been made, and the work presented in this thesis will leverage and extend such results to create a dynamic calibration technique capable of controlling complex systems such as whole diesel engines. More specifically, the main methodological contributions of this thesis are:

- application of a statistical test to assess linearity in dynamical system, prior to proceeding with control design (Chapter 3);

- application of a dynamic calibration technique applying optimal control and neural network methods capable of dealing with highly non-linear EGR valves (Chapter 4);
- extension of such technique to automotive air-path system where some of the internal states or variables to be controlled are not measurable (Chapter 5 and Chapter 6);
- adaptation of an optimisation-based test signal design to improve the accuracy of system identification techniques and controllers derived from such, applied to automotive dynamic Design of Experiments (Chapter 7).

Chapter 3

Nonlinearity Detection

3.1 Introduction

A common assumption in system identification and control is that the dynamical system at hand is linear. If this assumption is approximately true, the task of identifying a black-box model or deriving a control scheme for the system is simplified. This is due to linear methods having analytical solutions, whereas nonlinear methods must be solved numerically. Therefore, the preferred control and system identification methods are linear, as commonly stated in literature such as [49].

In practice, the decision to treat an unknown physical system as linear or nonlinear is often made by empirically testing different identification or control methods and assessing which works the best. However, the decision can be better informed by performing a preliminary nonlinearity test. This approach allows the decision to be made with statistical confidence, and it may potentially save time by narrowing the selection of identification and control methods to either linear or nonlinear methods.

A method to perform such a statistical nonlinearity detection test is presented in [72], and is based on the principle of superposition of linear systems. This method builds upon previous efforts such as the Subba Rao method [92] and neural network method described in [71]. The former method analyses a timeseries as a sum of Gaussians to test linearity, the latter uses optimal network weights as a test.

In this chapter, the approach described in [72] is adapted to make use of rich system identification data collected using pseudo-random inputs, allowing the data to be used for both preliminary nonlinearity testing and system identification. This is beneficial when data is either expensive to collect, time-consuming, contingent on availability of apparatus, or a combination of all three. The method is applied to three SISO systems, one linear and two nonlinear. It is then applied to a published diesel engine air path model described in [68] to show its effectiveness on a real-world application.

The Subba Rao method makes use of spectral analysis and the principle that the response of a linear system can be constructed by a sum of Gaussians [92]. For a linear timeseries $y(t)$,

$$y(t) = \sum_{i=-\infty}^{\infty} a(i)e(t-i) \tag{3.1}$$

where a is a set of constant values and e is a Gaussian function. If the signal can

be reconstructed as in equation 3.1 to sufficient accuracy, the linear hypothesis is accepted. This method does not take account of the input used to drive the system. The method described in [4] makes use of artificial neural networks. For a vector $\boldsymbol{u}$ consisting of an input and a set of its value at past time steps 1—k given by

$$\boldsymbol{u} = [u_k, u_{k-1}, \ldots, u_2, u_1]^T \tag{3.2}$$

the output of a feedforward neural network with one hidden layer with q neurons at time step k is given by

$$y_k = \boldsymbol{u}^T\boldsymbol{\theta} + \sum_{i=1}^{q} \boldsymbol{\beta}_i \psi(\boldsymbol{u}^T\boldsymbol{\gamma}_i) \tag{3.3}$$

where $\boldsymbol{\theta}$ is a set of linear weights and $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are a set of weights for the nonlinear function ψ. When the null hypothesis of linearity is true, the optimal network weights $\boldsymbol{\beta}$ are found to be 0. This method neglects feedback terms, limiting the accuracy of the neural network model and therefore limiting the validity of the nonlinearity test.

3.2 Method

The nonlinearity detection method applied in this chapter is based on two main principles: superposition principle to assess variance of system response due to nonlinearity, and a statistical test to assess its significance.

3.2.1 Superposition principle

The applied method tests the linear hypothesis by checking if the underlying superposition principle holds. The superposition principle states that the response of a linear system subject to multiple superimposed input sources is the sum of the system response to each input source individually [52]. For a dynamical system with response G it can be said that

$$G(2\boldsymbol{u}) = 2G(\boldsymbol{u}) \tag{3.4}$$

is true for any input signal u if the system is linear, where $\boldsymbol{u}$ is a vector of input values from the present and a finite number of time step delays into the past. Testing the equality in equation 3.4 requires three data sets denoted by y_1, y_2 and y_3. Each data set corresponds to input signals $\boldsymbol{u}_1$, $\boldsymbol{u}_2$ and $\boldsymbol{u}_3$, defined as

$$\boldsymbol{u}_1 = \boldsymbol{u}_2 = \boldsymbol{u} \tag{3.5}$$

$$\boldsymbol{u}_3 = 2\boldsymbol{u} \tag{3.6}$$

where u is an input signal which is sufficiently information-rich to excite all relevant system behaviour. Output signals corrupted by noise are then collected for each input:

$$y_1 = G(\boldsymbol{u}_1) + v_1 \tag{3.7}$$

$$y_2 = G(\boldsymbol{u}_2) + v_2 \tag{3.8}$$

$$y_3 = G(\boldsymbol{u}_3) + v_3 \tag{3.9}$$

where v_1, v_2 and v_3 are uncorrelated noise terms with zero mean and variance σ_v^2. Output signals y_1 and y_2 are generated by the same input signal u, but affected by different realisation of the measurement noise. Therefore, subtracting the two signals gives the difference in their respective measurement noises v_1 and v_2,

$$d_1 = y_2 - y_1 = v_2 - v_1 \tag{3.10}$$

Similarly, another signal d_2 is defined as

$$d_2 = y_3 - y_2 - y_1 = G(2\boldsymbol{u}) - 2G(\boldsymbol{u}) + v_3 - v_2 - v_1 \tag{3.11}$$

incorporating the superposition principle from equation 3.4. For a linear system, equation 3.4 is true and d_2 simplifies to $v_3 - v_2 - v_1$. The variance sum law states that the variance of a signal is the sum of the variances of its constituent signals [78]. Therefore, the expected variances of d_1 and d_2 are given by

$$\mathrm{var}(d_1) = 2\sigma_v^2 \tag{3.12}$$

and

$$\mathrm{var}(d_2) = 3\sigma_v^2 + \sigma_g^2 \tag{3.13}$$

where σ_g^2 is the variance of $G(2\boldsymbol{u}) - 2G(\boldsymbol{u})$. for a linear system, the superposition principle implies that σ_g^2 is zero.

3.2.2 Statistical test

Following from equations 3.12 and 3.13, the criterion for nonlinearity detection reads

$$\mathrm{var}(D_1) < \mathrm{var}(D_2) \tag{3.14}$$

where

$$D_1 = d_1/\sqrt{2} \tag{3.15}$$

$$D_2 = d_2/\sqrt{3} \tag{3.16}$$

to simplify notation equation 3.14 is rewritten as:

$$\sigma_1^2 < \sigma_2^2 \tag{3.17}$$

A two-sample F-test is then performed to assess if the difference between σ_1^2 and σ_2^2 is statistically significant. The null hypothesis for such a test is the linearity of the system at hand. The F-score for the timeseries' is calculated by

$$F = \frac{D_1^T D_1}{D_2^T D_2} \tag{3.18}$$

giving a measure of similarity in variance. As F deviates from unity, σ_g^2 increases, increasing the likelihood that the system is showing nonlinear behaviour. In the following, the linear hypothesis was tested with a confidence level of 98%, using the criterion for similarity described in [78].

3.3 Benchmark Examples & Analysis

The described nonlinearity detection method was tested on three SISO systems for benchmarking and on a published MIMO model of the airpath of a diesel engine. System A represents a simple linear damped harmonic oscillator described by the second-order linear ordinary differential equation

$$\ddot{y} + 0.3\dot{y} + y = u \tag{3.19}$$

where u is the system input and y is its output.

System B represents a nonlinear damped pendulum described by

$$\ddot{y} + 0.3\dot{y} + \sin(y) = u \tag{3.20}$$

where y is the angle of the pendulum and u is an angular acceleration applied to it by an external moment.

System C is a nonlinear oscillator described by the cubic Duffing equation, which is a standard nonlinear system

$$\ddot{y} + 0.3\dot{y} + y - 1.5y^3 = u \tag{3.21}$$

where y approximately represents the deflection of a beam forced periodically between two magnets by external force u. This is a standard benchmark system in the study of nonlinear systems and chaos, as in [93], and is used to model certain physical systems such as spring pendula.

System D is a model of a 12 litre diesel air-path model, developed and validated in [68]. Specifically, the relationship between its variable geometry turbocharger (VGT) setting and compressor flow rate was analysed. This is known to be highly nonlinear and exhibits non-minimum phase behaviour.

The superposition method is formulated for arbitrary input signals. However, for the results to be relevant to real-world applications, the input signals should excite behaviours which are representative of the system's typical operation. This is beneficial in practical applications, as this is also a requirement of data used for system identification. As a result, the same data can be used for both nonlinearity testing and system identification. Therefore, the signal should be information-rich, exciting the full range of relevant dynamical modes. For this study, an amplitude modulated pseudo-random binary signal (APRBS) was used. The amplitude of input signal u is varied by maximum amplitude U and the superposition method is used to test the nonlinearity of each system. The APRBS signals used were constructed by defining 10 random uniformly distributed time intervals. In each time interval, the amplitude was chosen from a random uniform distribution between $-U$ and U (for the SISO systems) or between 0 and U for the engine model. This builds on the method proposed in [72] by using signals which are useful for system identification. This also mitigates the need for additional, potentially expensive, data collection. Such a signal is composed of a wide range of frequencies and amplitudes, making it information-rich and exciting a range of dynamical behaviours. The signal used provided enough information to produce representative black box models (e.g. polynomial models), and so it can be considered to excite the relevant dynamical behaviour of each system [49].

3.4 Results

Each SISO system was driven for 10 simulated minutes with an APRBS input signal. This was done according to the procedure described in section 3.2. Figure 3.1 shows the time response of system C to the pseudo-random input signal. Timeseries y_1 and y_2 differ only in terms of random measurement noise, which is added artificially to the model. These series are used to estimate the variance of measurement noise. Series y_3 is subject to $2u$, so its response is larger. The three timeseries are used to test nonlinearity as described in section 3.2.

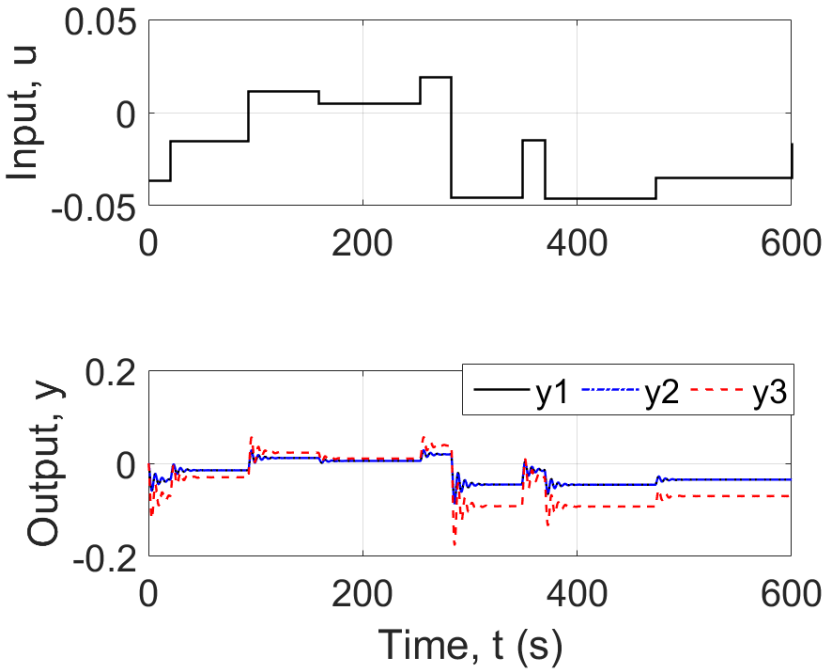

Figure 3.1: Time response of system C to APRBS input.

Figure 3.2 shows the time response of the engine model to APRBS input to its VGT when the maximum amplitude is 1% and the minimum is 0%.

Figure 3.3 shows the F-score and decision variable for the three SISO systems. A test decision of 0 indicates probable linearity, whereas 1 indicates nonlinearity. It is shown that for all tested input amplitudes, system A stays at a relatively flat level at $F \approx 1$. This indicates that this system behaves linearly, as expected, given that it represents a simple harmonic oscillator. As shown in figure 4, the method rejects linearity for the engine model at about 0.6% VGT. This is extremely low, with typical real VGT actuators saturating below 10%. This suggests that the examined input-output relation between VGT and compressor flow rate cannot be considered linear for any realistic purposes, which is commonly expected for diesel airpath behaviours described in literature, for example [56]. Other input-output relationships in diesel engines may vary in nonlinear dynamics, and this method could be helpful for characterising this.

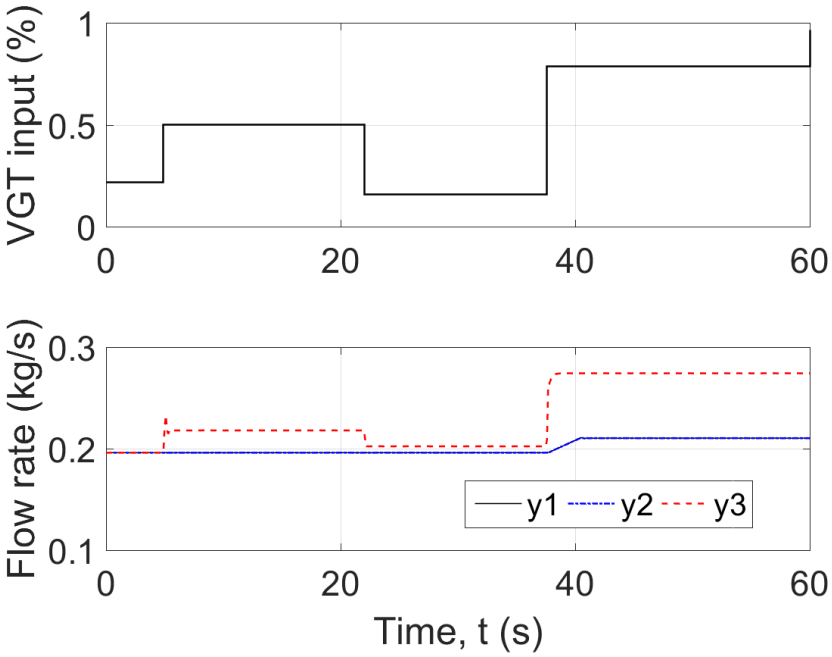

Figure 3.2: Time response of the engine model to APRBS input.

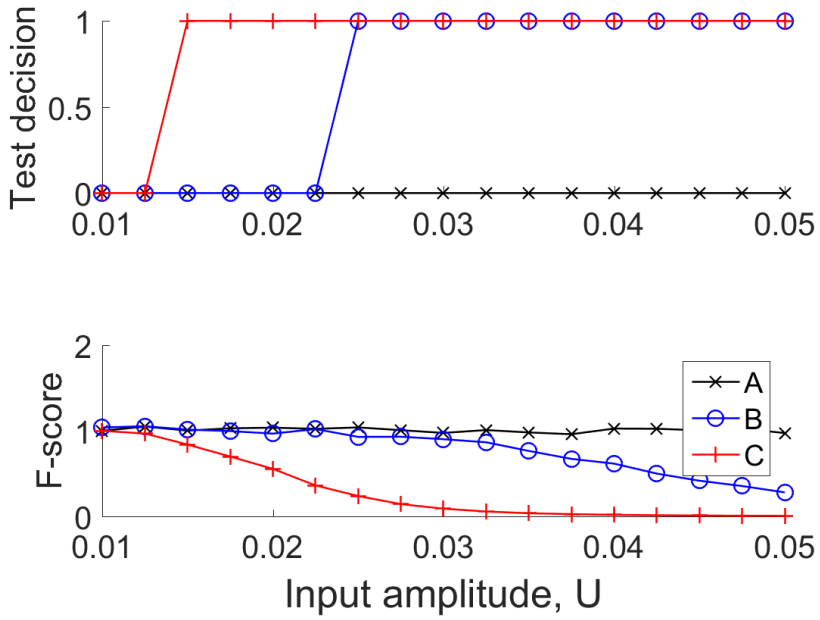

Figure 3.3: Test decision and F—score for varying input amplitude for each SISO system.

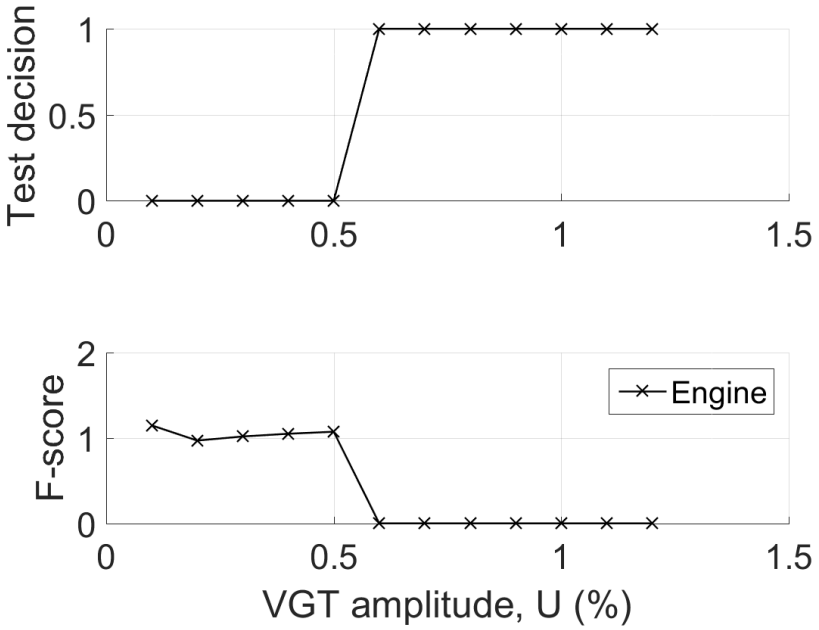

Figure 3.4: F-score and test decision for varying input amplitude for the engine model.

3.5 Conclusions

The superposition method for testing nonlinearity in dynamical systems provides mathematical justification for the acceptance or rejection of dynamical systems. The applied method makes use of data appropriate also for system identification, forgoing the need for additional, potentially expensive, measurements.

Four dynamical systems were analysed using the method, with varying amounts of nonlinearity. A linear model was found to always accept the null hypothesis of linearity for all examined input amplitudes. A pendulum model was found to accept linearity at small perturbations, and Duffing oscillator with cubic stiffness term was found to accept linearity only for even smaller perturbations. The relationship between turbocharger setting and compressor flow rate in a diesel engine model was found to reject linearity for all realistic input amplitudes.

The trend for the measured F-score, indicating the prevalence of nonlinear behaviour, is relatively smooth and therefore convincing when justifying approximating a system as linear. This smoothness makes F-score a reliable measure of nonlinearity which could be used to relate optimal model and control structures to the strength of a system's nonlinear behaviour. A reported shortcoming of the method in [72] is its sensitivity to non-normally distributed measurement noise. Future work could make the method more robust to non-Gaussian noise. Another potential extension would be considering the effect of process noise on the system dynamics.

Chapter 4

Control of an Exhaust Gas Recirculation Valve

Modern diesel engines are equipped with a growing number of components to help meet legislative limits on emissions. In addition to advanced turbocharging techniques which help prevent the formation of particulate matter and reduce unburnt fuel waste, exhaust gas can be recirculated into the intake manifold to reduce oxygen concentration, which reduces the formation of NO_x [2] [20]. This is done via a system of exhaust gas recirculation (EGR) valves, typically a high- and low-pressure loop. The distinction comes from the source of the exhaust gas, with low pressure gas being recirculated downstream of the turbocharger turbine, and high pressure coming directly from the exhaust manifold.

Each added component must be calibrated individually, which means tuning a position controller for each valve. Ideally this would be accomplished with a linear PI controller, however nonlinearity due to stiction is significant [94]. In practice, the PI gains of the controller are staged by position to approximate the behaviour of a nonlinear controller. This is typical of many control loops in an engine control unit [9] [10]. As also stated by [10], gain staging works best with a scheduling variable which is not a state variable, or at least slowly-changing, which is not the case when using valve position.

In this chapter a dynamic nonlinear controller is implemented and then demonstrated on a low-pressure EGR (LPEGR). Being dynamic allows the controller to use time-series data to improve calibration speed, and being nonlinear allows it to deal with prominent stiction. The performance of the nonlinear controller was then compared to the typical staged PI controller for two valves, denoted valve A and valve B, identical except valve A was more worn.

4.1 Methodology

One of the benefits of an optimisation-based approach is that controller tuning can be done much faster than a manual, online calibration. However, this means the tuning must be done offline using a model therefore the process consists of a modelling step and a controller tuning step. Another potential problem is the need for time-series data, as taking many static measurements would increase the time taken to comparable static design of experiments (DoE) methods as in [1]. This is because the engine must be left to settle at each static operating point before taking

each measurement. In order to use time-series data, both model and controller are made to be dynamic by introducing time delays into the structure. The methodology was designed to deviate as little as possible from the typical online valve calibration process, automating anything different.

4.1.1 Valve modelling

The valve shown in figure 4.1 was excited by an amplitude-modulated pseudo-random binary sequence (APRBS) and its position sensor output recorded and used to train a neural network model using the MATLAB Neural Network Toolbox [67] to derive a model of the valve. Using neural networks was a simple black-box modelling solution to compensate for unique characteristics between valves of the same type, either due to manufacturing tolerances or wear, the latter of which will be seen when examining two valves of different ages. The calibration software INCA allows users to modify calibration variables in the ECU, also pictured in figure 4.1, however this is designed with static DoE in mind. In order to record dynamic data as a time-series, simulated key-presses modify the motor duty cycle in real-time via the graphical user interface of the calibration software.

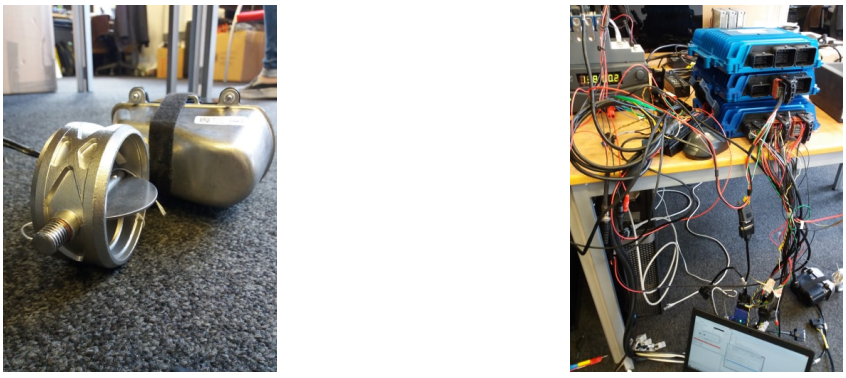

Figure 4.1: LPEGR experimental setup: (left) valve used to collect data and perform validation, (right) ECU used to control the valve.

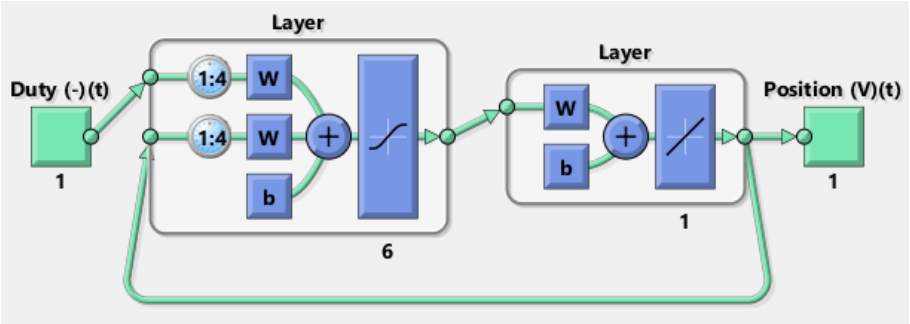

Figure 4.2: Neural network valve model structure

The structure of the valve model is shown in figure 4.2, with one input and one output. Duty cycle input and feedback position output have time delays 1:4 based on trial and error, starting from 1 and stopping when model error started to increase. Each time delay is equal to the model time step of 0.05s. The hidden layer neuron size of 6 was chosen in the same way by trial and error, increasing from 1 neuron

iteratively until error started to increase. A hyperbolic tangent transfer function, denoted *tansig* by the toolbox, both bounds the output of the layer and allows for representation of nonlinear features such as stiction.

4.1.2 Controller tuning

The neural network (NN) investigated for controlling the valve, represented in figure 4.3, was structured to have the important features of a combined feedforward and PI controller, namely nonlinear layers for proportional error feedback W_{FB}, integral error feedback W_I and demand signal feedforward W_{FF}. Duty cycle is determined by weight matrices which convert each layer output to the appropriate dimension, 1x1 in this case, then sums them. This is an adaptation of the standard linear feedforward-feedback controller as described by [49] and builds on the feedforward control approach of [37].

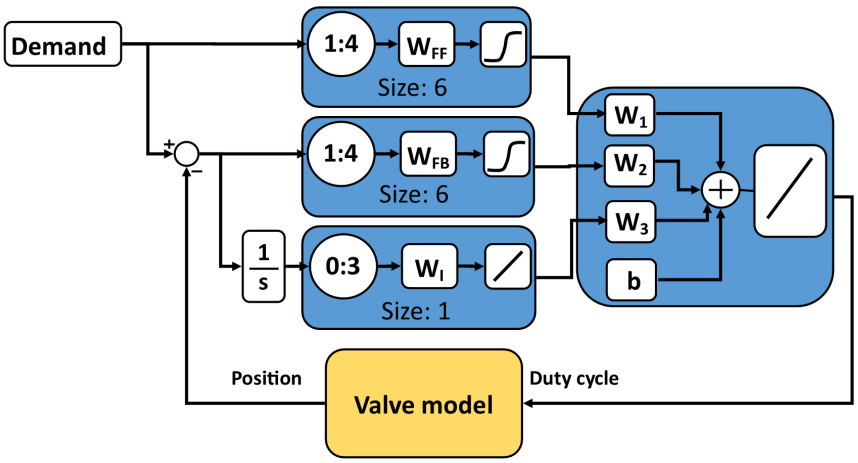

Figure 4.3: Neural network valve controller structure

A pure feedforward controller would ideally characterise the inverse dynamics of the system to control, so its time delays and layer size are set to the same as the valve model. To account for uncertainty in valve behaviour, a proportional feedback layer with the same feedback output time delays and size as the valve model is added. Integral action is achieved by a simple linear dynamic layer with the same time delays.

The valve model shown is the same neural network model derived from the valve input-output measurements in section 4.1.1. Its weights and biases are fixed in the controller tuning step, forcing the rest of the neural network to control it in order to minimise the difference between demanded position and modelled position. Demanded position for the purposes of training is an APRBS signal as before, though the original model training data can be used as a more realistic demand signal. An APRBS signal was found to work for all examined cases, and was chosen for being more challenging for the controller to track than the original training data.

4.1.3 PI controller

Two valves were examined, A and B, the former with a standard manually tuned PI controller. Valve B was swapped in when valve A was broken, and needed a new PI controller. This was done by using the offline ANN valve model to optimise each gain using the MATLAB Optimization Toolbox [67] function *fmincon* set to use the nonlinear interior-point method. There were 12 proportional (P) gains and 12 integral (I) gains, for a total of 24, each staged by a linearly spaced series of valve positions from 0 to 100%. As well as replacing the original PI controller lost when valve A was broken, this served as an alternative optimisation-based calibration using standard ECU infrastructure for the sake of comparison to the NN method. The optimal PI gains are shown in 4.4.

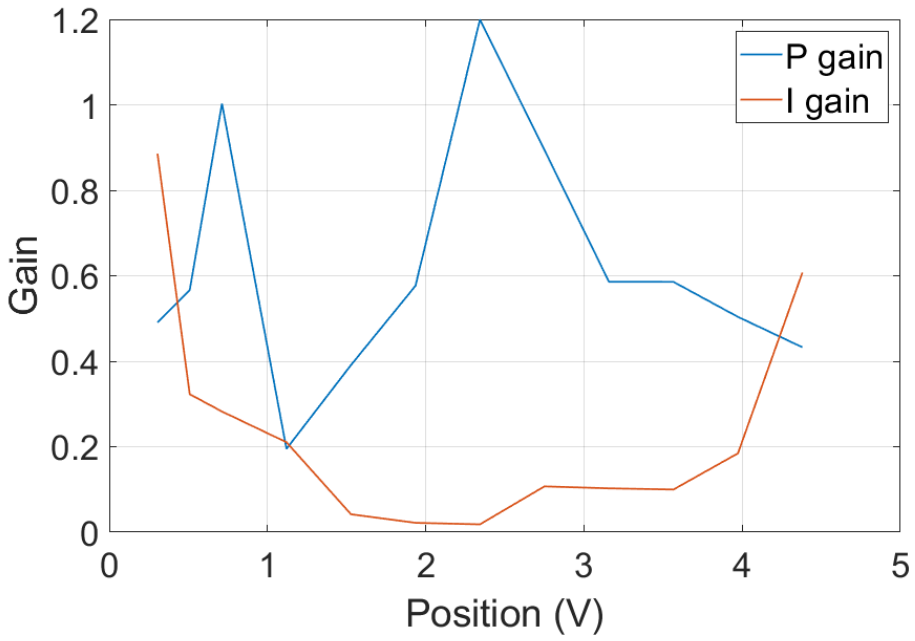

Figure 4.4: Optimal PI gains for the LPEGR

4.2 Results

The model used to tune each controller is validated first against its training data and then its validation data. Then results for the neural network (NN) and manual PI (PI) controllers are compared for both valves, including mean-squared error (MSE) calculations. For valve B, these are also compared to an optimised PI (PIopt) for a partial European Drive Cycle (EDC) [95] derived from a JLR research vehicle. Specifically the first 3 minutes of the EDC is used, according to the behaviour of the calibrated LPEGR in the JLR research vehicle, and simulated using automated key-press in the standard calibration software. It should be noted that the actual signal as supplied to the valve may differ between experiments due to key-press delays depending on computer load, but the signal as supplied to the valve is as shown in the figures. This is why there are some differences in demand signal between controllers.

4.2.1 Valve A

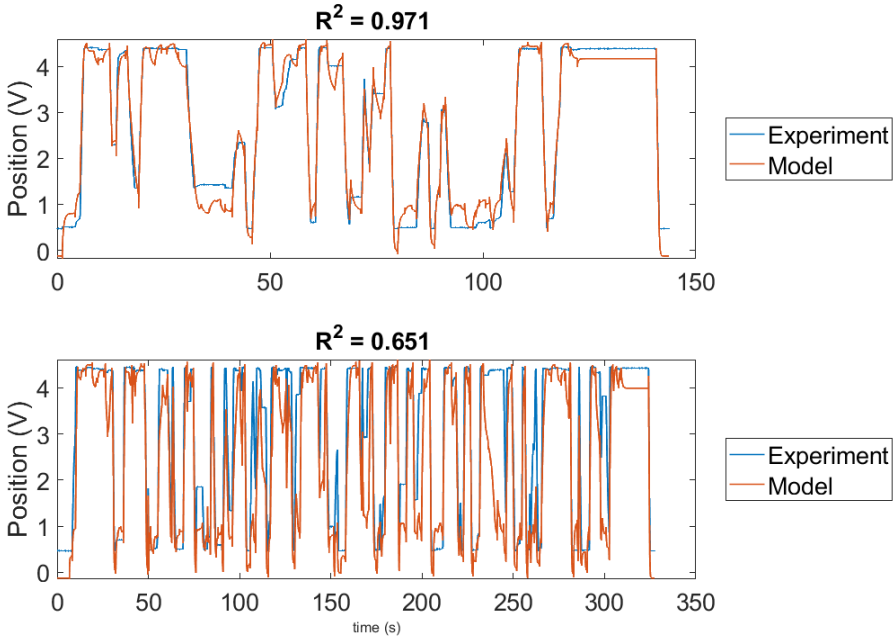

Figure 4.5: Validation of valve A model (Top: training set, bottom: validation set)

The neural network controller was found to achieve significantly better tracking than the standard PI controller. This is reflected in the mean-squared error (MSE) values in figure 4.6, but is most easily seen with a slower demand signal as shown in figure 4.7. Notably, the implemented NN controller significantly reduces overshoot in valve position compared to the PI controller. It is shown in figure 4.5 that against the training data set, the valve A model performs considerably worse against its validation set. It is shown that the model predicts motion which, in practice, is prevented by stiction despite a change in duty cycle supplied to the motor. However, the controllers tuned against the model are shown to perform well on the real motor, demonstrating robustness of the method.

4.2.2 Valve B

It is notable that this valve was considerably easier to model and generalised much better to the separate validation data set, as shown in figure 4.8. It is possible that valve A, being considerably older, was worn out from the beginning, and therefore had more pronounced nonlinear behaviour due to increased stiction or other sources. This would explain it needing to be replaced partway through the project.

Figure 4.9 shows comparable performance to the same type of controller as used on valve A. No data is available for the equivalent PI controller for this particular demand, but similar MSE performance between valves suggests the NN method will work on different valves of similar design, and is therefore fit for its purpose as a general calibration method.

The first few minutes of an EDC were simulated for the PI, NN and PIopt controllers. The results shown in 4.10 demonstrate that while the NN method produces significantly better results than a manually tuned PI, similar performance

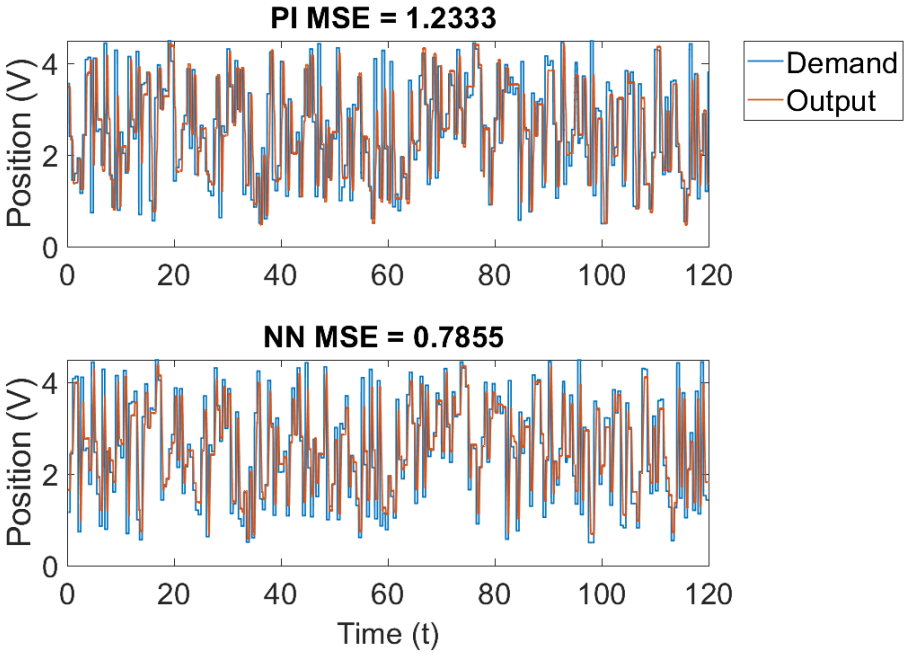

Figure 4.6: Training data for the valve A controller

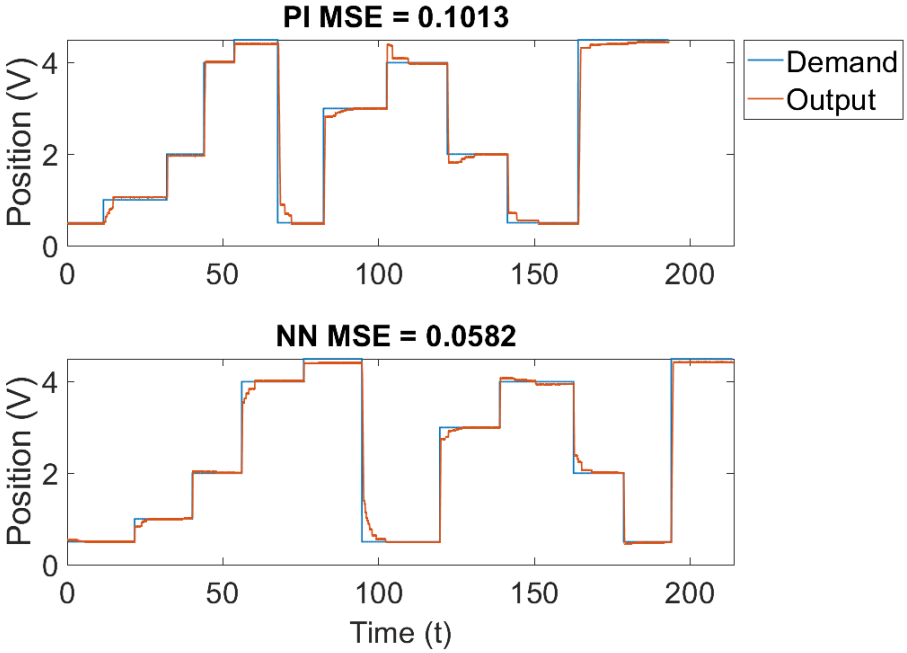

Figure 4.7: Validation data for the valve A controller

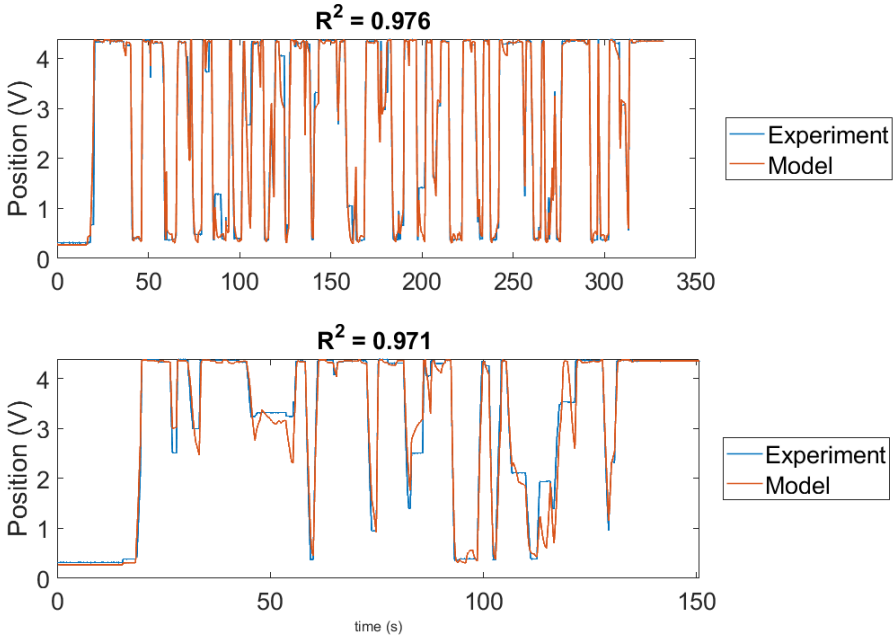

Figure 4.8: Validation of valve B model (Above: Training, below: validation)

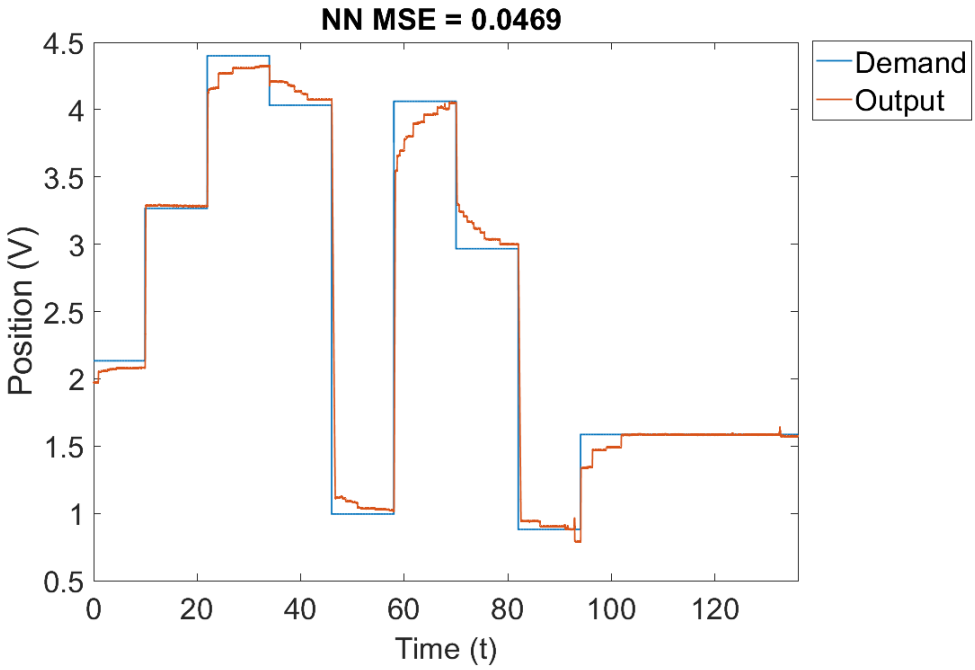

Figure 4.9: Step response for the valve B NN controller

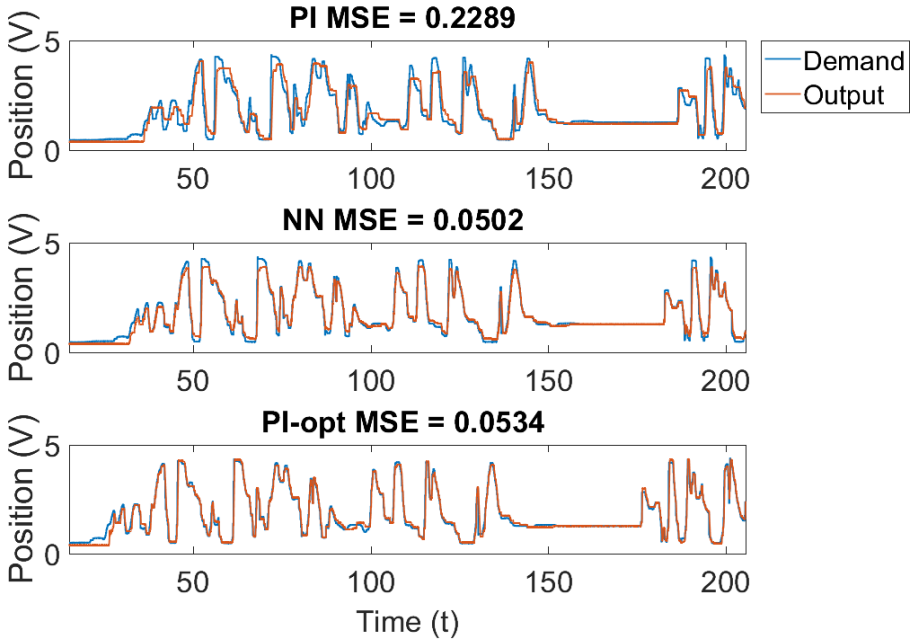

Figure 4.10: Partial simulated EDC data for the valve B controller

can be achieved by tuning the same PI structure using optimisation techniques. The trade-off is longer computation time, as the PIopt controller takes around 30 minutes compared to 10 minutes for the NN controller on a standard laptop used on-site for calibration.

4.3 Conclusion

A method for neural network control of LPEGR valves was implemented then tested on-line and compared with a standard manually-tuned PI controller. The NN controller was found to perform much better than the PI controller, but had comparable performance to an optimised PI controller. This suggests that for simple (albeit nonlinear) systems, a gain staged PI controller is sufficient, though it would be very difficult to tune by hand to the same level of performance.

Neural network controller parameters such as hidden layer size and time delays were selected by trial and error, method refinement might include setting these parameters algorithmically. As the controller is tuned offline using a model, improvements to the valve model would improve the controller quality. It was shown that the valve models used, while able to produce valid controllers for the real valve, had somewhat poor accuracy due to the effects of stiction, and in the case of valve A, age. As a result, the use of an offline model for controller tuning could have been a trade-off with controller quality. However, significant errors in the model used to tune the valve A controllers did not prevent the neural network method nor the optimised PI from outperforming standard manual methods, suggesting these methods provide better performance despite uncertainties between the two different valves.

Chapter 5

Engine setpoint control

The number of actuators in diesel engines has significantly increased over time [20] [2] [1] and the control loops required to operate them has motivated research into whole-engine calibration methods [56] [37] [53], treating the many components of the engine as one whole system to control. For this, the valve control method described in chapter 4 was adapted to whole-engine control on a simulated engine, as described in this chapter.

Due to access to the real engine testbed at Jaguar Land Rover being denied early in the project, a simulated engine is treated as a real engine in this chapter. As the following method relies on black-box offline models of the real engine, a distinction is made between the *simulated engine* produced by Wahlström and Eriksson [68] and any neural network *engine models* derived from it via system identification. The simulated engine is briefly described in section 5.2.

5.1 Engine Design of Experiments

In typical static design of experiments (DoE) calibration [1] the engine is driven to steady state at a set of operating points (OP), where ECU maps and controllers are tuned to meet legislative emissions limits. This is done manually for each point and so, similarly to the valve calibration, the benefits of a dynamic offline calibration method include less time spent tuning. Emissions are largely controlled in practice by turbocharging (VGT) and exhaust gas recirculation (EGR) [2] . In figure 1.2 in Chapter 1 a schematic of a typical configuration of EGR and VGT in a diesel engine is shown.

By deflecting the VGT downstream of the exhaust manifold, the compressor is made to drive more air into the intake manifold, which provides more oxygen for combustion, but also for formation of NO_x pollutants. The air can be diluted to reduce NO_x by opening the EGR valve, recirculating exhaust gas into the intake manifold. This has the adverse effect of reducing oxygen concentration and increasing the formation of particulate matter PM due to incomplete combustion.

The calibration process aims to balance these two strategies to meet emissions legislation at each OP, and the spread of OPs is determined using optimisation methods. A DoE in terms of engine load and engine speed for a JLR 2.0L engine is shown in figure 5.1, with corresponding emissions limits shown by figure 5.2 (data provided by JLR).

The summarised process is as follows [1]:

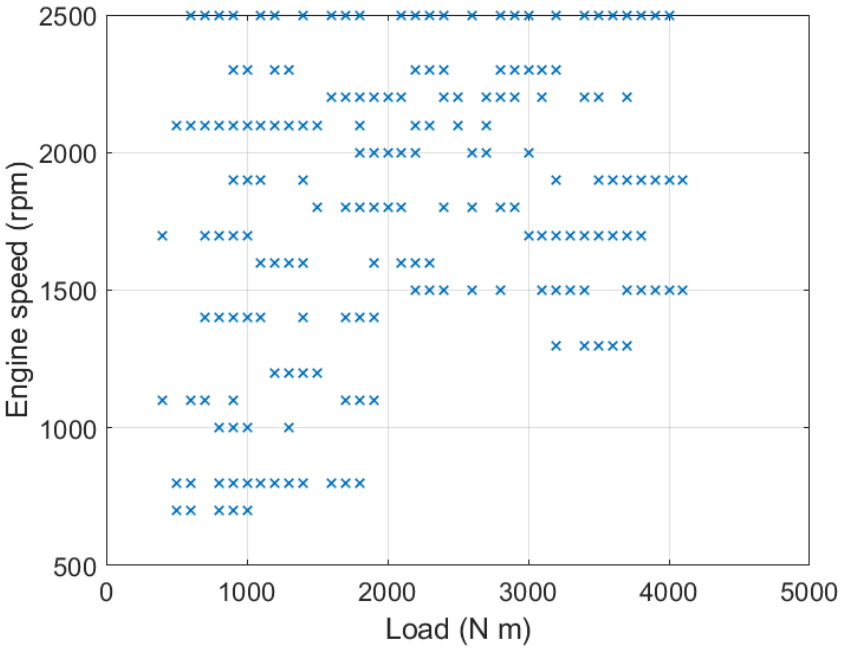

Figure 5.1: JLR 2.0L diesel DoE

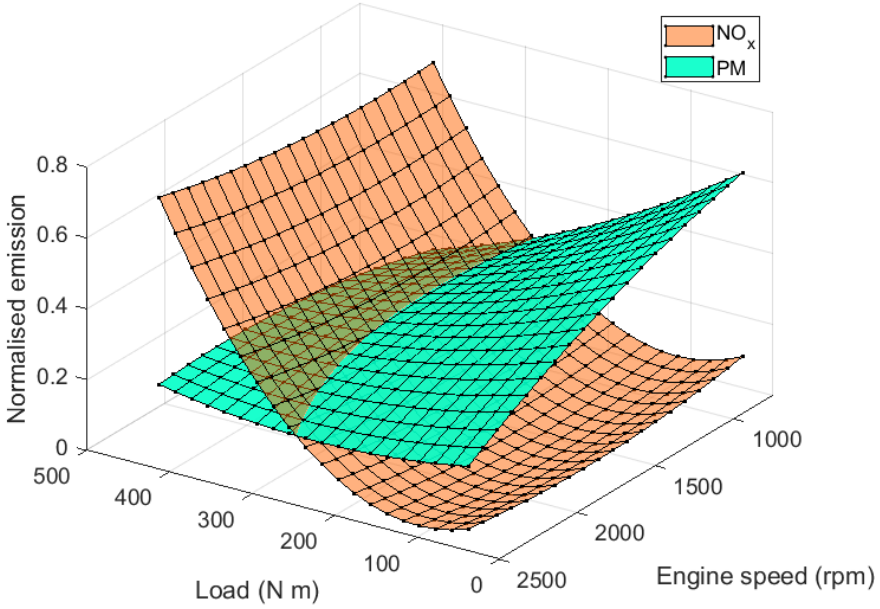

Figure 5.2: JLR 2.0L diesel DoE with emissions

1. Choose operating points and corresponding emissions targets;

2. Optimise engine response at each OP by adjusting VGT and EGR;

3. Build continuous maps by smoothing between those settings.

Typically emissions are not the direct quantity to control, instead related quantities are controlled, i.e. air-fuel equivalence ratio (λ) and EGR fraction (x_{egr}). Optimal settings for λ and x_{egr} to meet emissions limits are set in a separate procedure. In addition, torque demand must also be met, so in total there are three targets:

1. Air-fuel equivalence ratio, λ, where $\lambda = 1$ is a stoichiometric mixture,

2. EGR ratio, x_{egr}, the ratio of recirculated exhaust gas to total air mixture in the intake manifold and

3. engine torque, M_e.

Mathematically, air-fuel equivalence ratio and EGR ratio are defined as [68]

$$\lambda = \frac{AFR}{AFR_s} \tag{5.1}$$

$$x_{egr} = \frac{W_{egr}}{W_c + W_{egr}} \tag{5.2}$$

where AFR is air-to-fuel ratio and AFR_s is the stoichiometric AFR, W_{egr} is the EGR flow rate and W_c is the intake air flow rate, or flow rate through the turbocharger compressor.

For the purpose of developing the methodology, a validated simulated diesel engine [68] was used in place of an engine and testbed. The tunable inputs that will be supplied by the controller to the engine are

1. Variable geometry turbocharger, u_{VGT}, expressed as a percentage where 100% is full deflection;

2. exhaust gas recirculation valve setting, u_{EGR}, expressed as a percentage where 100% is fully open;

3. fuel injection per cycle per cylinder, u_δ, in $mg/cycle$.

5.2 Methodology

The method was applied to a validated simulation of a 12L diesel engine implemented in Simulink by Wahlström and Eriksson [68], its top level is shown in figure 5.3. The methodology for whole-engine calibration closely resembles that of the valve calibration in chapter 4. Unlike the valve's position case, the outputs to control, x_{egr}, λ and M_e, cannot typically be measured in a production engine. This makes them unavailable for feedback control, and requires the use of an observer to predict the internal states of the engine. To differentiate between directly measurable sensor

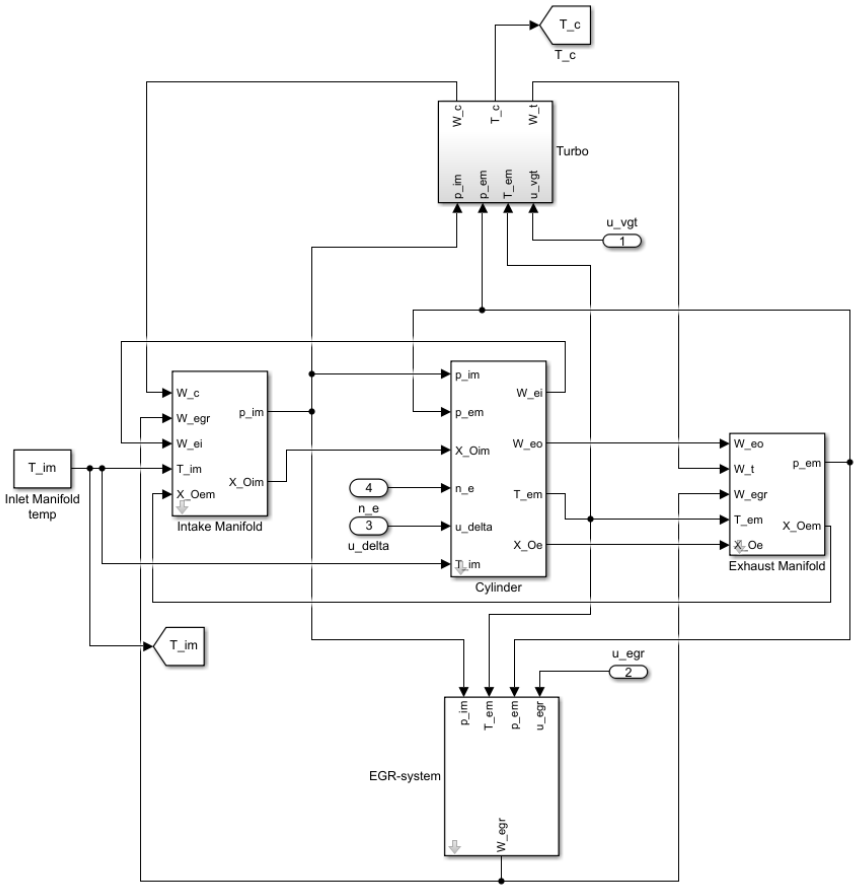

Figure 5.3: Simulated engine top level, taken from [68]

outputs and outputs indirectly predicted by the observer, two output vectors were defined as

$$\boldsymbol{y_{meas}} = [W_c, W_{EGR}, n_e]^T \tag{5.3}$$

$$\boldsymbol{y_{obs}} = [\lambda, x_{EGR}, M_e]^T \tag{5.4}$$

where $\boldsymbol{y_{meas}}$ is a vector of dynamic quantities which can be measured on a production engine or approximated via maps, and $\boldsymbol{y_{obs}}$ is a vector of dynamic quantities to be observed. The observer predicts the values of $\boldsymbol{y_{obs}}$ using direct measurements of $\boldsymbol{y_{meas}}$. Engine input was defined as

$$\boldsymbol{u} = [u_{VGT}, u_{EGR}, u_\delta]^T \tag{5.5}$$

5.2.1 Engine modelling

A surrogate model of the engine must be derived in order to tune the controller offline, as with the valve control. This is done using a neural network shown in figure 5.4. As part of the calibration procedure, the neural network was trained

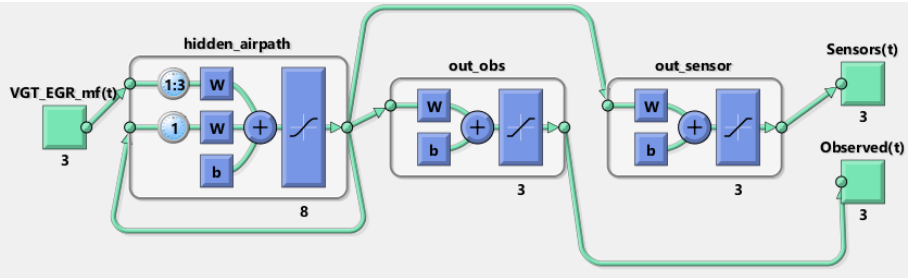

Figure 5.4: Engine neural network model

using the MATLAB neural network toolbox [67] using data from the simulated engine. The engine was excited by amplitude-modulated pseudo-random binary signals (APRBS) as a timeseries test signal $\boldsymbol{u}(t)$, shown in figure 5.5. To generate

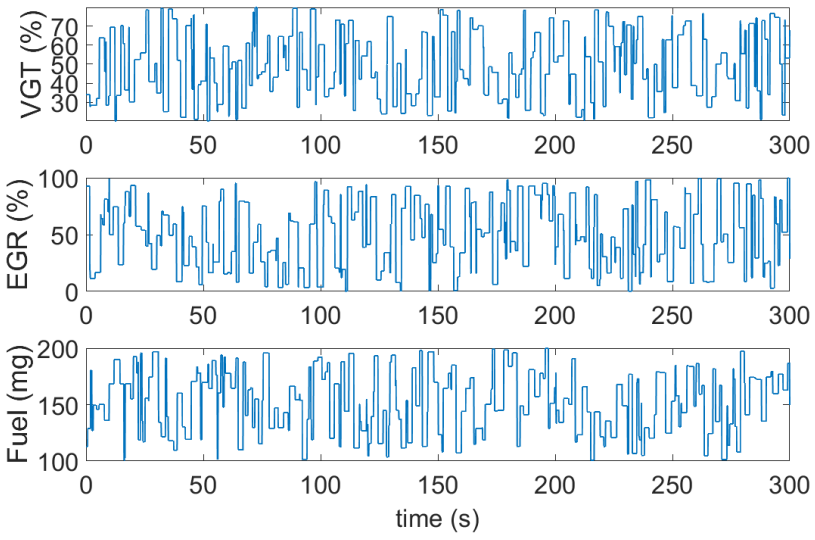

Figure 5.5: Test signal for engine model training

a fresh data set for validation, the simulated engine was then driven with a newly generated signal shown in figure 5.6.

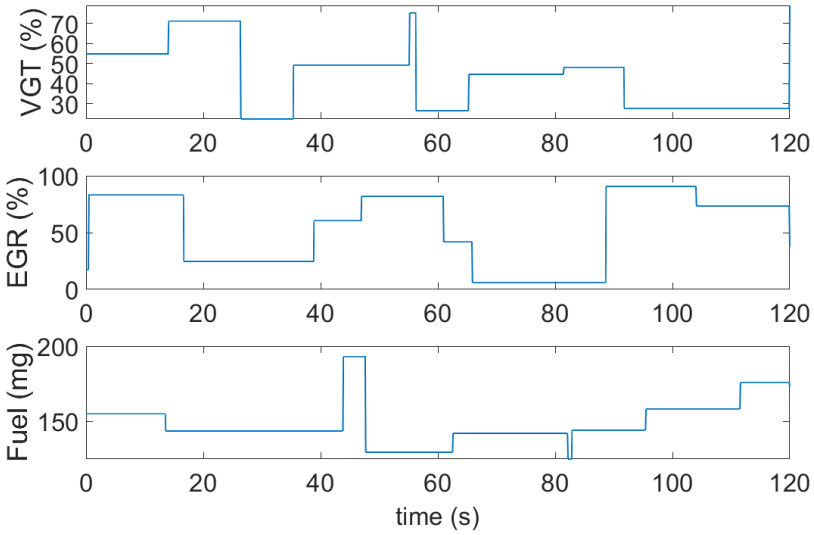

Figure 5.6: Test signal for engine model validation

5.2.2 Controller tuning

As previously mentioned, direct measurements of the outputs to control for this case are not available for feedback to the controller. Therefore the controller structure of chapter 4 was modified to include an observer implemented as a neural network. The final controller structure, shown in figure 5.7, can be broken down as

1. Feedforward layer, representing the inverse dynamics of the engine;
2. State demand layer, which converts the demanded outputs into observer state-space;
3. State observer layer, which predicts the internal state of the engine based on feedback;
4. Feedback layer, which feeds back the estimated state error calculated by layers 2 and 3;
5. Integral feedback layer, which provides integral action;
6. Engine input layer, which produces the final engine input.

Additionally, the NN engine model is embedded in the structure with its weights and biases fixed during controller training. The structure was designed to resemble typical linear controllers with feedforward, feedback, integral action and state observer [49]. The derivation of observer and control law at the same time is based on robust and optimal control methods described by Zhou et al [58], and resembles nonlinear implementations of H_2 control as in Hardt, Helton and Kreutz-Delgado [60].

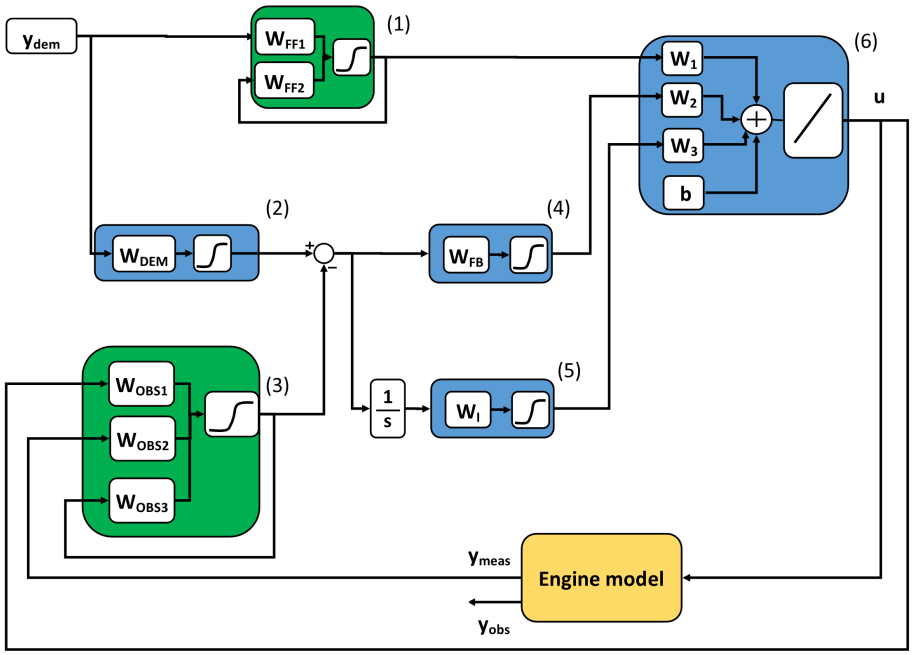

Figure 5.7: NN engine controller structure

The observer was configured to have the same time delays as the NN engine model, and has the same number of neurons as the model hidden layer, i.e. 8. Measurements from available sensors are fed back to the observer which predicts the system internal state, allowing the controller to indirectly predict quantities for which no sensors are available, e.g. torque.

The feedforward layer would ideally exhibit the inverse dynamics of the system to control [49], allowing it produce the correct system input for the demanded output. Therefore it is expected that such a controller would have time delays and complexity of similar order to the engine model, so the feedforward layer was configured by reversing the NN engine model time delays so that input delays became output delays and vice versa. In practice, the number of neurons was reduced from 8 to 5 by trial and error for the feedforward layer, since perfect representation of inverse dynamics is not necessary when feedback is introduced. This improved smoothness of response and reduced the time to tune the controller.

The numbers of neurons in the feedback layers for proportional and integral error were set to 3, the number of controllable inputs and therefore the number of error signals to feed back. The proportional feedback layer size was increased until the tracking of the controller stopped improving, finally setting it to 5.

Feedforward, proportional feedback and integral feedback are converted to the correct input dimensions and summed by the engine input layer. As implemented in MATLAB, the controller structure looks like in figure 5.8, with the layers corresponding to the components in figure 5.7 numbered accordingly.

The demand signal to the controller is the target output to achieve in the engine. This defined by

$$\boldsymbol{y_{dem}} = [\lambda^{dem}, x_{EGR}^{dem}, M_e^{dem}]^T \tag{5.6}$$

where λ^{dem}, x_{EGR}^{dem} and M_e^{dem} denote the demanded values of their corresponding

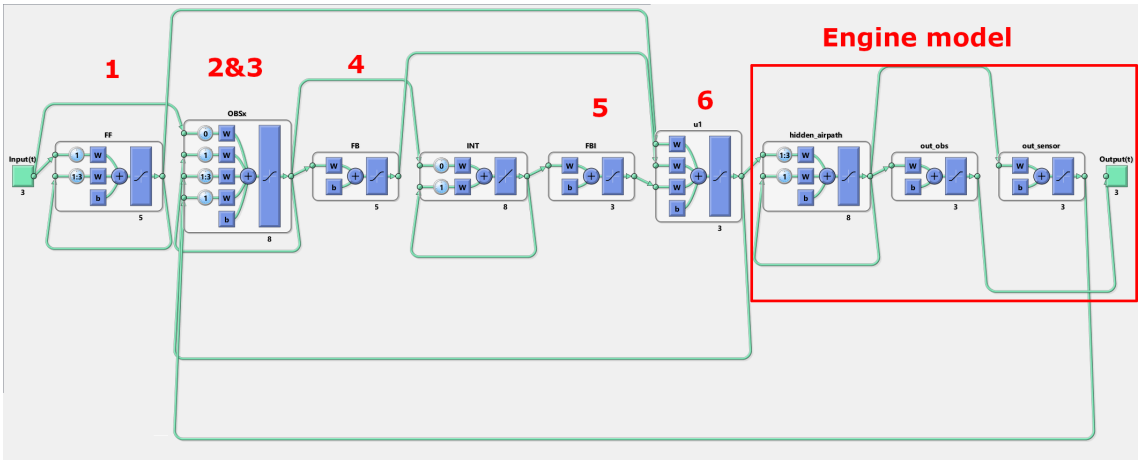

Figure 5.8: MATLAB engine controller structure implementation

observed variables in equation (5.4). When training the controller, the error between $\boldsymbol{y_{obs}}$ and demand signal $\boldsymbol{y_{dem}}$ should be minimised, therefore training is formulated as the optimisation problem

$$\min_{\boldsymbol{w},\boldsymbol{b}} \sum_{t=0}^{t_{max}} \boldsymbol{\Theta}(\boldsymbol{y_{dem}}(t) - \boldsymbol{y_{obs}}(t))^2 \tag{5.7}$$

where $\boldsymbol{w}$ and $\boldsymbol{b}$ are the weights and biases respectively of the neural network controller shown in Figure 5.7. Adjusting these during optimisation affects the controller output in order to bring the engine model output close to the demand signal. $\boldsymbol{\Theta}$ is a weighting matrix of appropriate dimensions which allows the optimisation to be tuned. e.g. if the closed loop torque error is too high, its weighting can be increased relative to the others.

It should be noted that the internal state of the system as defined by the observer is abstract, and the state values do not necessarily represent any particular physical quantity. The state of the system is selected to be of a higher order than the number of controlled outputs, so that more information about the system is made available to the controller. However, there is no constraint dictating that the observer structure should behave as an observer, it is treated the same as all other layers by the training algorithm. It is possible and probable for the algorithm to discover this on its own and work as intended, but requires the method be repeated until it does, varying the random seed used to initialise training. Therefore equation 5.7 was modified to impose this in the cost function as

$$\min_{\boldsymbol{w},\boldsymbol{b}} \sum_{t=0}^{t_{max}} \boldsymbol{\Theta} \begin{bmatrix} (\boldsymbol{y_{dem}}(t) - \boldsymbol{y_{obs}}(t))^2 \\ (\boldsymbol{x_{dem}}(t) - \boldsymbol{x_{obs}}(t))^2 \end{bmatrix} \tag{5.8}$$

where $\boldsymbol{x_{obs}}$ is the abstract observed state of the system corresponding to $\boldsymbol{y_{obs}}$ and $\boldsymbol{x_{dem}}$ is the equivalent system state corresponding to $\boldsymbol{y_{dem}}$, the respective outputs of layers 2 and 3. By including both in the cost function, it is imposed that both demanded and observed system states correspond to demanded and observed outputs, ensuring the observer structure is treated as an actual observer.

5.3 Results

An NN model of the engine was trained using data collected from the simulated engine and validated agaist its training data and also a new set of validation data.

5.3.1 Engine modelling

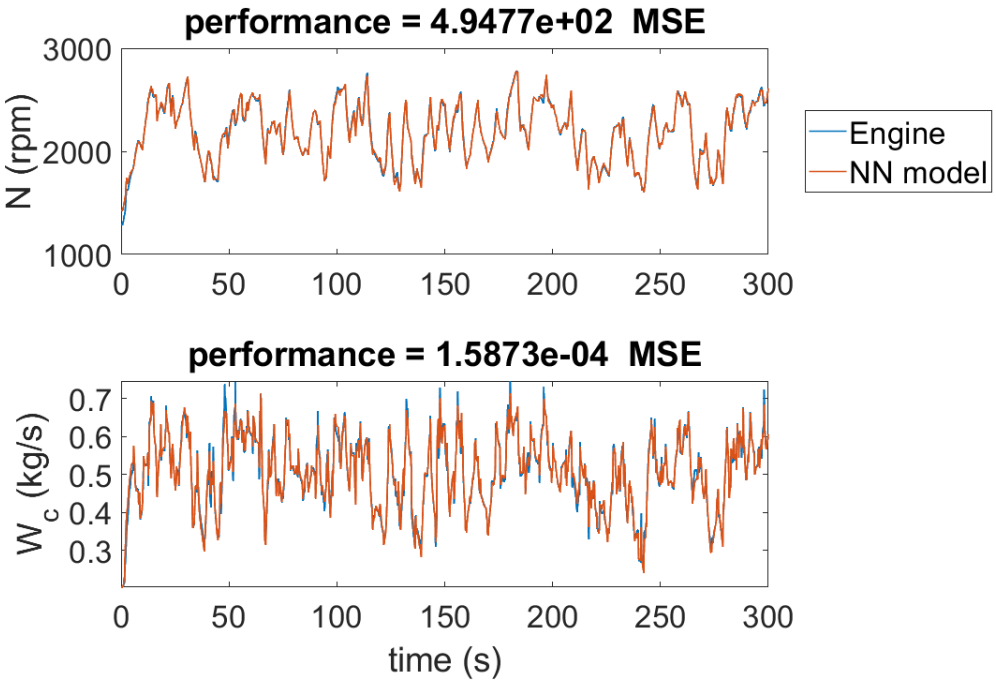

Figure 5.9: Engine model response to training signal (top: engine speed, bottom: compressor intake rate)

The NN model of the engine derived from data taken from the simulated engine fit the training data well as shown in figures 5.9, 5.10 and 5.11. This is supported by performing similarly against the validation data set in figures 5.12, 5.13 and 5.14. It was found in chapter 4 that the applied NN control method is robust enough to reject considerably larger errors than are shown here and produce well-performing controllers, so the model was accepted for offline tuning of the controller.

5.3.2 Engine control

The controller was validated by supplying as a demand the engine free-response to the APRBS signal in figure 5.6, and the response is shown in figure 5.15. The controller replicates the signal very well for all controlled quantities with equal weighting set via the $\mathbf{\Theta}$ variable when tuning. The exception is the high error during the initial settling period, which can be eliminated by initialising the controller before use or allowing it time to settle by itself. This initial high error can cause problems with the training algorithm, as it overstates the gradient used during the optimisation, therefore the first 10 seconds of response are ignored during controller training.

To illustrate the benefit of the combined feedforward-feedback approach, the controller was re-tuned with only feedforward action and validated in figure 5.16.

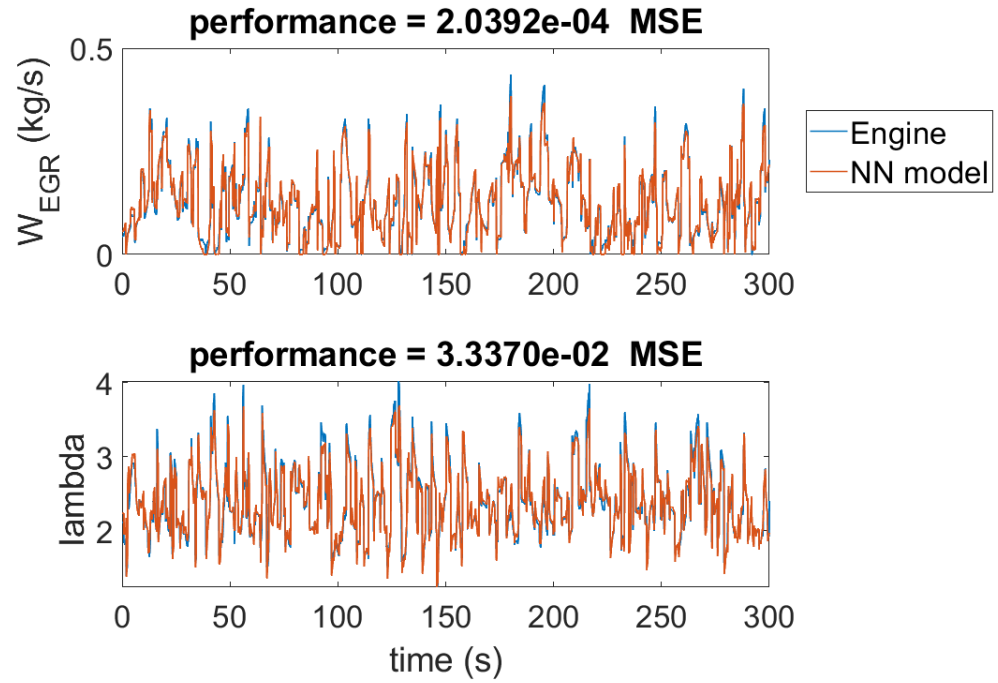

Figure 5.10: Engine model response to training signal (top: EGR flow rate, bottom: air-fuel equivalence ratio)

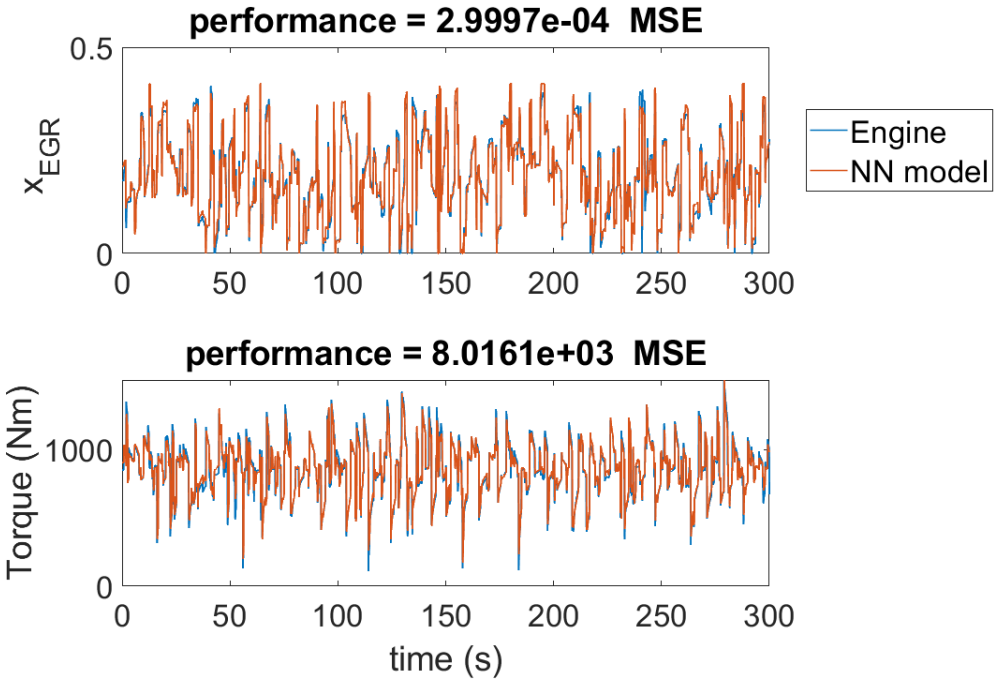

Figure 5.11: Engine model response to training signal (top: EGR fraction, bottom: engine torque)

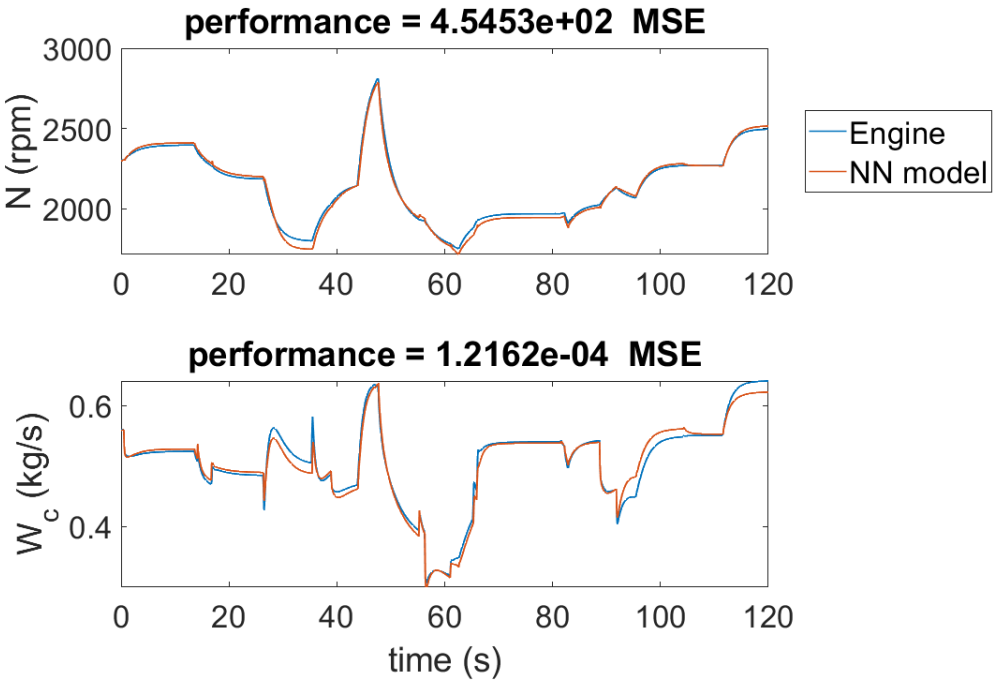

Figure 5.12: Engine model response to validation signal (top: engine speed, bottom: compressor intake rate)

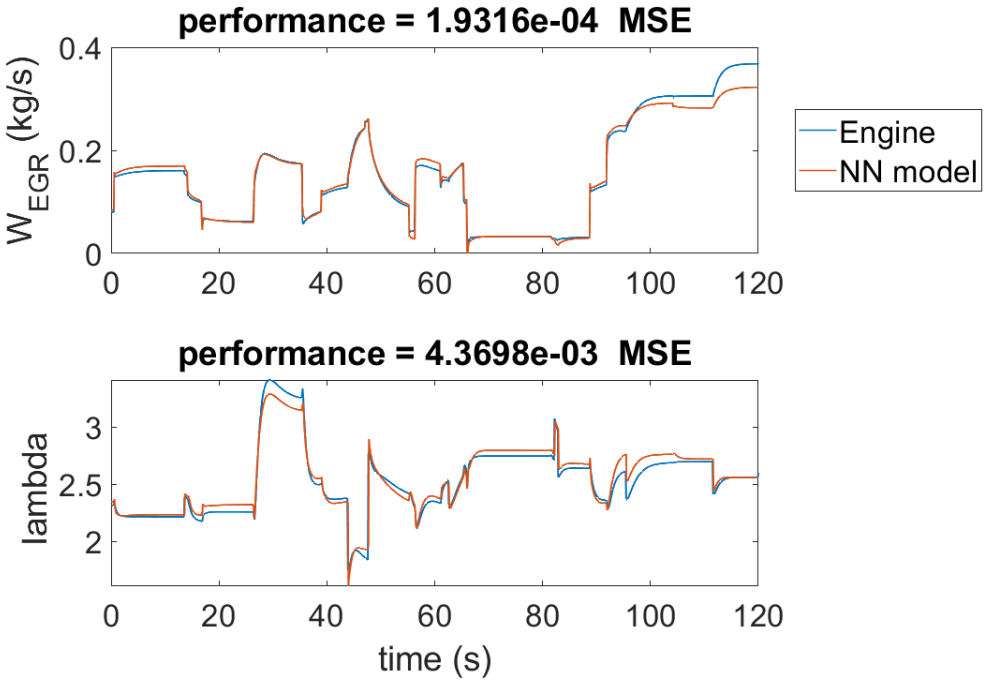

Figure 5.13: Engine model response to validation signal (top: EGR flow rate, bottom: air-fuel equivalence ratio)

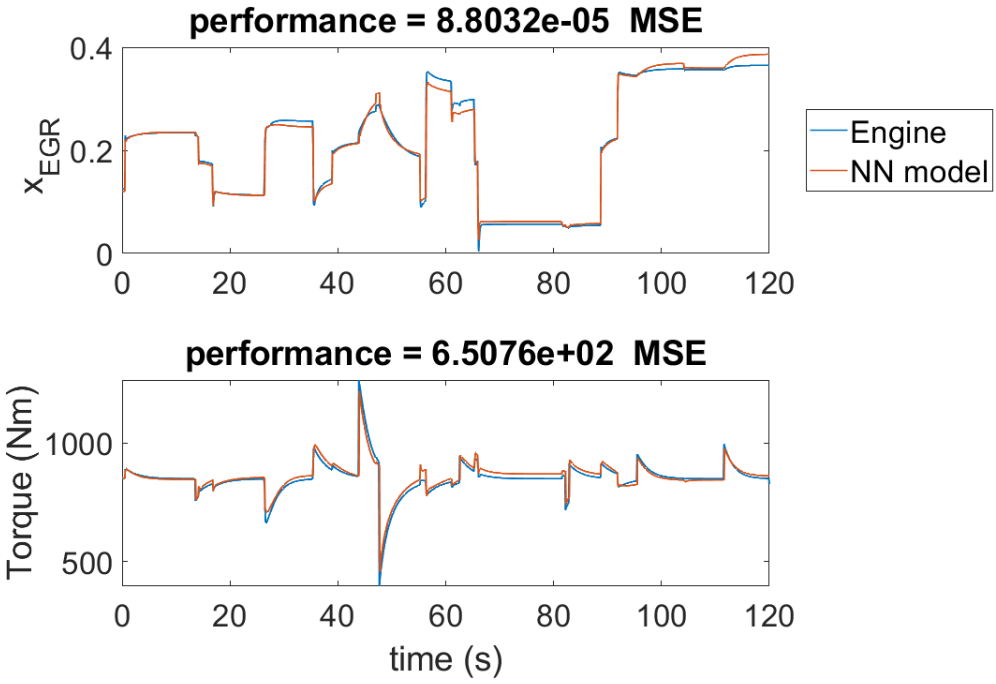

Figure 5.14: Engine model response to validation signal (top: EGR fraction, bottom: engine torque)

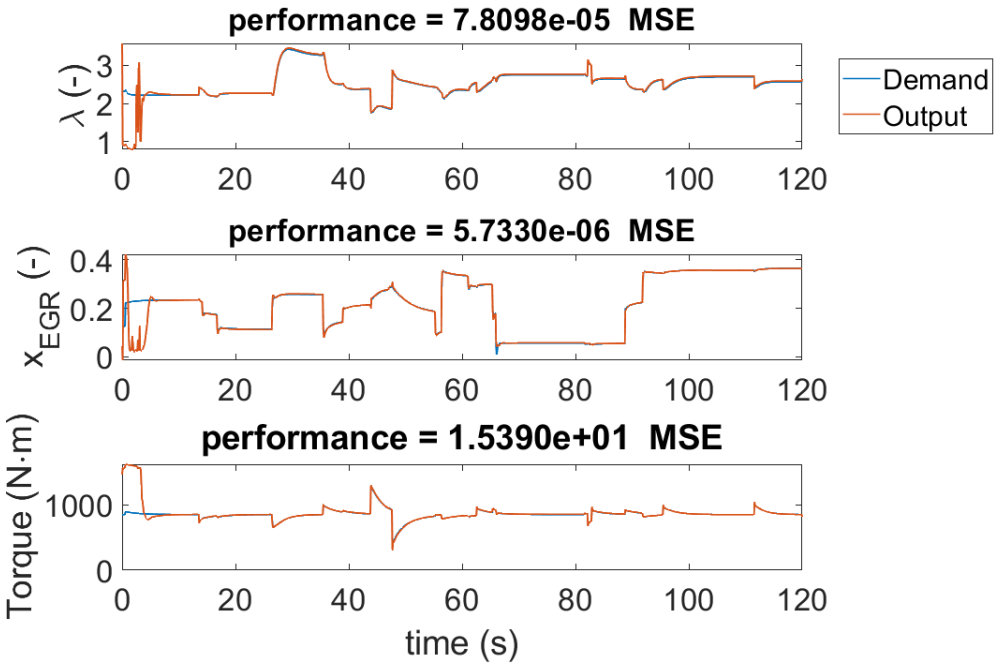

Figure 5.15: Engine full controller validation

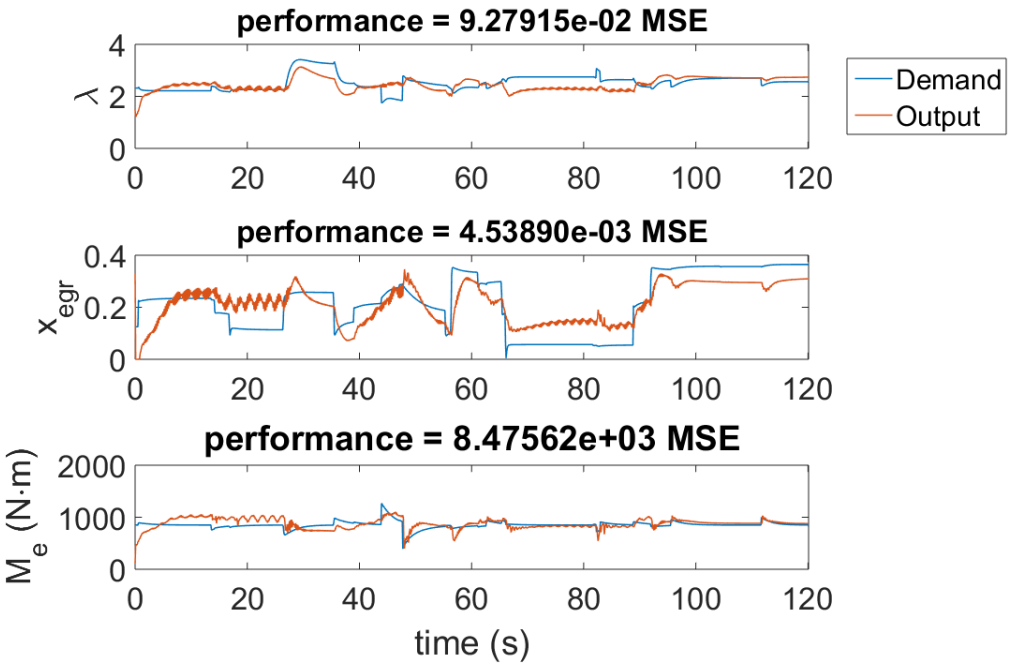

Figure 5.16: Engine feedforward only controller validation

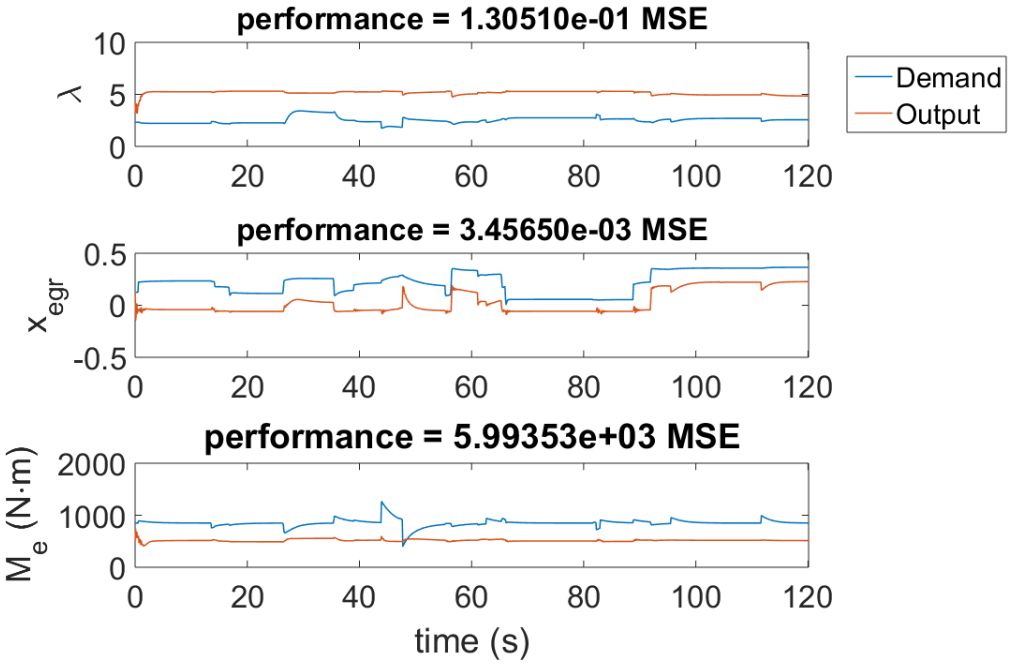

Figure 5.17: Engine feedback only controller validation

The feedforward controller tracks the general shape of the signal provided as would be expected, but uncertainty in the state of the system limits controller performance. The same was done with only feedback action in figure 5.17. In this configuration, the controller is essentially a nonlinear PI controller, which is evidently insufficient to control a system of this complexity.

5.4 Conclusion

The control methodology for valve calibration described in chapter 4 was adapted for a validated multiple-input multiple-output (MIMO) simulated engine. The outputs to control were treated as unmeasurable, and were instead estimated by a nonlinear observer taking feedback from other sensor measurements. The controller tracked a validation signal with high accuracy, barring an initial settling period which can be removed easily by allowing the controller to settle before use. However, this could be done more quickly with a method to initialise the internal state of the controller, and could constitute further work. The full controller was then compared to both a pure feedforward controller and a pure feedback controller with both showing much poorer performance. Therefore the proposed combined feedforward-feedback architecture was proven to be a robust control structure for complex problems, such as full engine control.

Chapter 6

Engine emissions control

In chapter 5 a neural network control method was applied to control three outputs - EGR fraction, x_{egr}, air-fuel ratio, λ and engine torque, M_e. The three together form a dynamic analogue of a setpoint as defined in engine Design of Experiments (DoE). In static DoE setpoints are determined for a range of operating points (OP) in order to meet emissions targets. The general static approach follows the steps defined in [1]:

1. Choose operating points and corresponding emissions targets;
2. Optimise engine response at each OP by adjusting VGT and EGR;
3. Build continuous maps by smoothing between those settings.

The controller applied in chapter 5 assumes that the optimal values of x_{egr} and λ are pre-determined by a DoE, as is typically the case. However, pre-determination of setpoints using DoE and manual calibration is time consuming and prone to error. In this chapter, the setpoint control method was adapted to control torque while minimising emissions directly. As in chapter 5, a *simulated engine* [68] was used as a proof of concept instead of a testbed and engine. A physical emissions model was embedded in the simulated engine [68] as it lacked one. The simulated engine, described briefly in section 5.2, is distinct from any neural network models derived from it in this chapter or the last, and is treated as a real engine on a testbed would.

6.1 Methodology

6.1.1 Emissions Modelling

The emissions examined were nitrogen oxides, NO_x and particulate matter, PM. As the vast majority of nitrogen oxides produced by diesel engines are of the chemical species NO [20], NOx formation is given by

$$\frac{d[NO]}{dt} = \frac{6 \times 10^{16}}{T^{\frac{1}{2}}} \exp\left(\frac{-69090}{T}\right) [O_2]_e^{\frac{1}{2}} [N_2]_e \tag{6.1}$$

where $[O_2]_e$ and $[N_2]_e$ are respectively the equilibrium concentrations of oxygen and nitrogen given local conditions.

Oxygen concentration is calculated by the engine model, and nitrogen was assumed to constitute the remainder of the gas flow into the cylinders. However, the

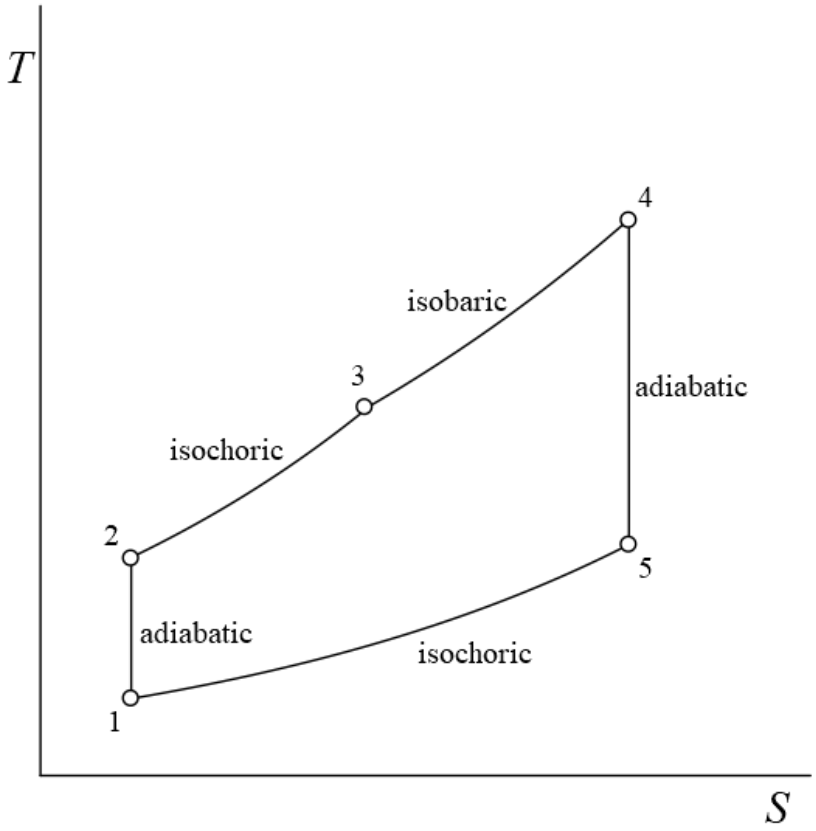

Figure 6.1: The Seiliger cycle T-S (temperature-entropy) diagram [97]

diesel engine model used was a mean-value model, which neglects discrete cycles of the engine and assumes all processes are spread out over the engine cycle [57], so the temperature history over the engine cycle is not known.

Therefore the idealised Seiliger cycle (AKA dual cycle, or mixed cycle) shown by figure 6.1 was used to estimate local cylinder temperature after combustion, as this is the phase during which most NO is formed [96].

The Seiliger cycle consists of

- Process 1-2: Isentropic compression;
- Process 2-3: Addition of heat at constant volume;
- Process 3-4: Addition of heat at constant pressure;
- Process 4-5: Isentropic expansion;
- Process 5-1: Removal of heat at constant volume.

For all temperature T_i and volume v_i definitions used hereafter, the subscript i corresponds to the stage in the cycle according to figure 6.1. The temperature T in equation (6.1) is considered to be the maximum temperature, after combustion at step 4, meaning $T = T_4$. To estimate this value, a ratio x_v defined by the cylinder

volume between the start and end of combustion given by

$$x_v = \frac{v_4}{v_2} = \frac{T_4}{T_2} \tag{6.2}$$

which is already calculated by the simulated engine according to the heat added by combustion of fuel. Another ratio relating compression ratio r_c and ratio of specific heats for air γ_a to temperatures T_1 and T_2 is defined by

$$r_c{}^{\gamma_a - 1} = \frac{T_2}{T_1} \tag{6.3}$$

Multiplying equation (6.2) by equation (6.3) gives an expression for T_4 given by

$$T_4 = T_1 x_v r_c{}^{\gamma_a - 1} \tag{6.4}$$

Intake temperature T_1 is known, γ_a is a property of air equal to 1.4 and compression ratio r_c is a design feature of the engine. As a result, equation (6.4) provides a way to model the peak cycle temperature to use with (6.1).

Particulate matter (PM) production is highly dependent on discrete events in the crank-angle domain such as fuel spray formation, cylinder temperature and pressure histories [57] [50]. Therefore, a temperature model in terms of crank angle was implemented in the engine model [70]. However it was found that the temperature model, due to engine speeds ranging from 700rpm - 2500rpm in a typical diesel engine, required a time step much smaller than that of the rest of the mean-value model. Variable time-step solvers were unable to converge due to the large difference in rate of dynamic behaviours, and a fixed-step solution for the whole model required a time step so small that it was not feasible to solve. With a real engine this would not be a limitation as sensors for PM are standard on testbeds for calibration.

The trade-off between NO_x and PM is described in Chapter 2 and illustrated by figure 2.8. Due to the reciprocal relationship between NO_x and PM, both must be represented in the model to avoid over-production of either by minimisation of the other. In short, production of particulate matter correlates negatively with oxygen concentration due to burning [98], and correlates positively with the amount of fuel burned [50] [99]. Therefore, the controller was optimised to minimise the following, indirectly imposing a soft constraint on the production of PM:

$$PM = \frac{u_\delta}{0.1 + [O_2]_e} \tag{6.5}$$

Equation (6.5) is roughly equal to fuel-to-air ratio, with a small constant added to the denominator to prevent singularities, which prevents the training algorithm from minimising NO_x indefinitely without regard for production of particulate matter. Henceforth, references to PM in this chapter will refer to equation (6.5) rather than "true" particulate matter.

6.1.2 Engine modelling

As in chapter 5, two output vectors are defined, however λ and x_{EGR} don't need be be controlled, and therefore don't need to be modelled.

$$\boldsymbol{y_{meas}} = [W_c, W_{EGR}, n_e]^T \tag{6.6}$$

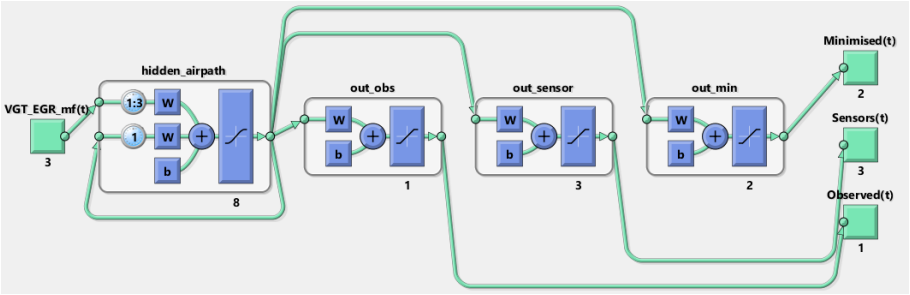

Figure 6.2: Engine neural network model with emissions.

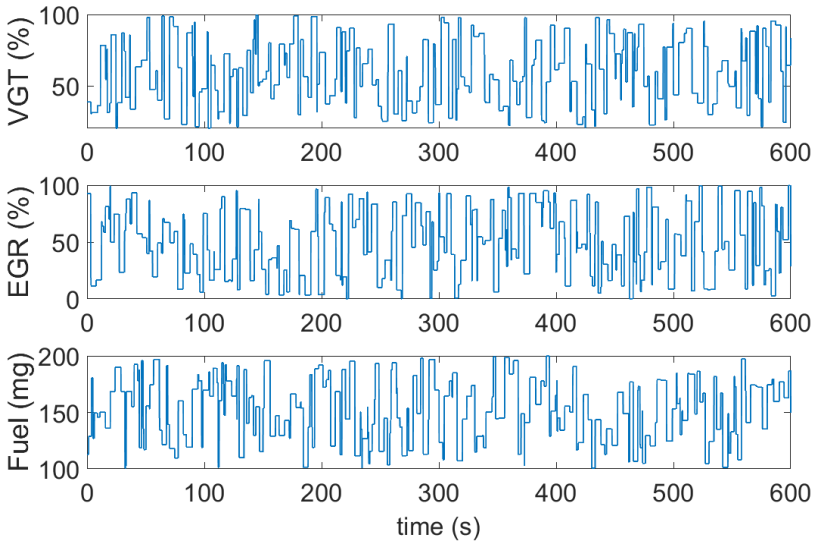

Figure 6.3: Test signal for engine model training with emissions.

$$y_{obs} = M_e \tag{6.7}$$

A third output vector is defined as

$$\boldsymbol{y_{min}} = [NO_x, PM]^T \tag{6.8}$$

which is the set of outputs which must be minimised, rather than controlled. This simplifies the method in chapter 5 as these quantities do not need to be estimated by the observer. The augmented model (figure 6.2) has a third set of outputs to minimise ($\boldsymbol{y_{min}}$), in addition to the existing sensor outputs ($\boldsymbol{y_{meas}}$) and outputs to observe and control (y_{obs}). Otherwise, the NN modelling process is identical to that described in chapter 5. The model was excited by the training signal in figure 6.3 and then by the validation signal in figure 6.4, with the training data being used to train the neural network model.

6.1.3 Controller tuning

The controller structure in figure 6.5 has the same structure of the controller used in chapter 5, but has an extra output $\boldsymbol{y_{min}}$ from the engine model which the algorithm attempts to minimise.

The structural breakdown is the same, and is reported below for clarity:

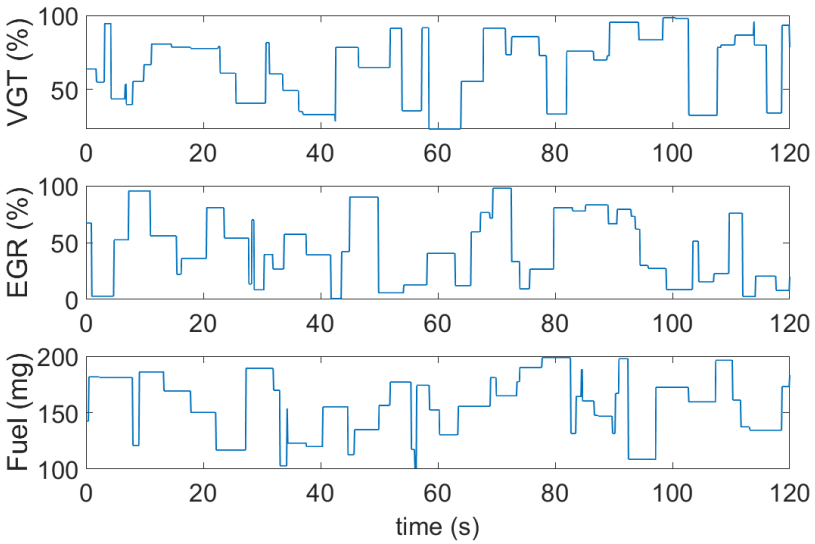

Figure 6.4: Test signal for engine model validation with emissions.

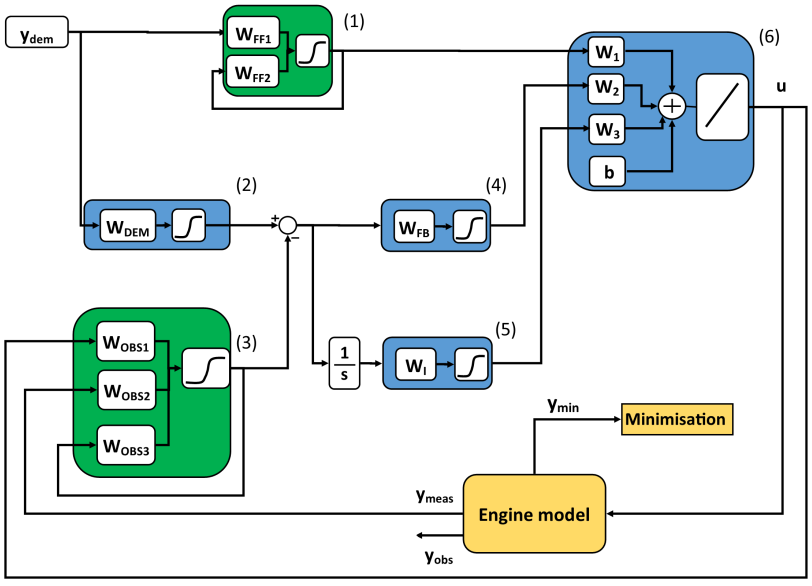

Figure 6.5: NN engine controller structure with minimisation output.

1. Feedforward layer, representing the inverse dynamics of the engine;
2. State demand layer, which converts the demanded outputs into observer state-space;
3. State observer layer, which predicts the internal state of the engine based on feedback;
4. Feedback layer, which feeds back the estimated state error calculated by layers 2 and 3;
5. Integral feedback layer, which provides integral action;
6. Engine input layer, which produces the final engine input.

A demand signal in the form of an amplitude-modulated pseudo-random binary signal (APRBS), $y_{dem} = M_e$ was supplied to the controller and trained to track it. The minimisation problem is modified from equation 5.8 to be

$$\min_{\boldsymbol{w},\boldsymbol{b}} \sum_{t=0}^{t_{max}} \Theta \begin{bmatrix} (y_{dem}(t) - y_{obs}(t))^2 \\ (\boldsymbol{x_{dem}}(t) - \boldsymbol{x_{obs}}(t))^2 \\ \boldsymbol{y_{min}}(t)^2 \end{bmatrix} \tag{6.9}$$

where the first term encourages demand tracking (weighted by Θ), the second imposes the observer portion of the NN be treated as an actual observer and the new third term is added to minimise emissions. The purpose of the second term is explained more fully in chapter 5. As before, the first 10 seconds of data is ignored during training so that settling period error does not influence the algorithm.

6.2 Results

6.2.1 Engine modelling

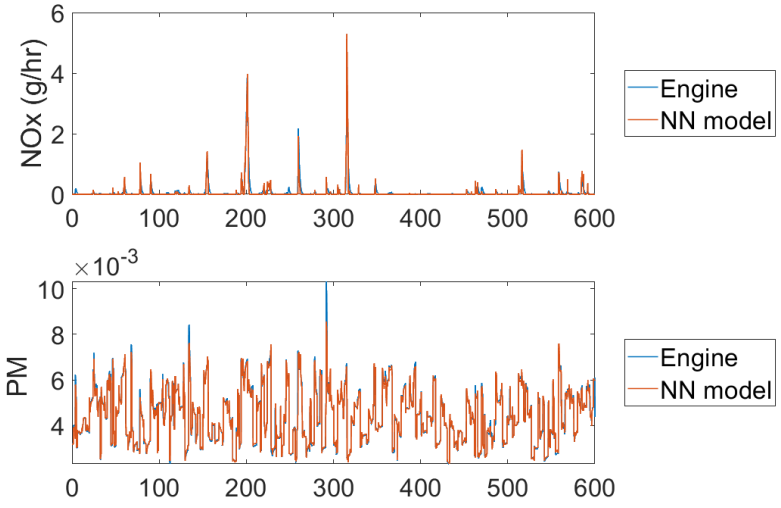

Figure 6.6: Engine model response to training signal. Top: NO_x, bottom: PM.

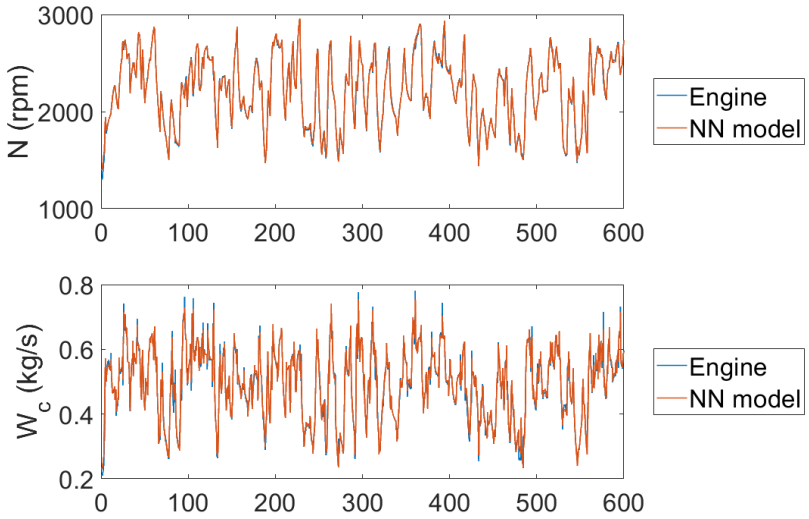

Figure 6.7: Engine model response to training signal. Top: engine speed, bottom: compressor intake rate.

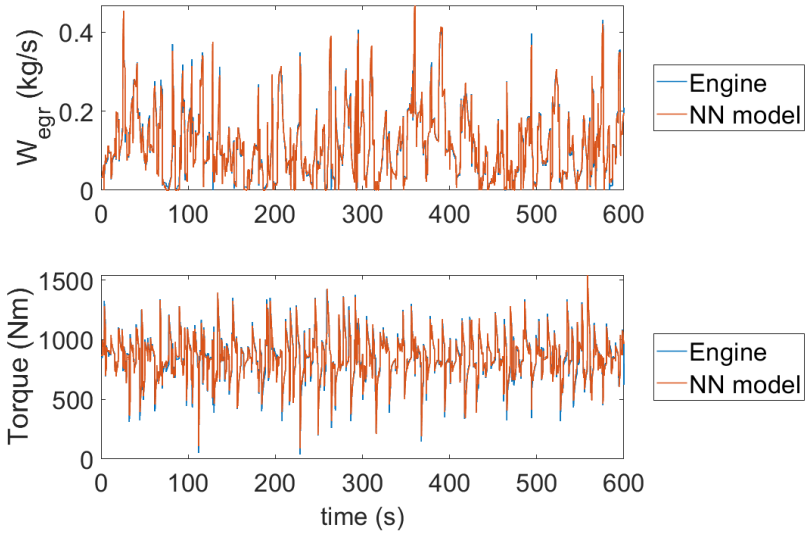

Figure 6.8: Engine model response to training signal. Top: EGR flow rate, bottom: engine torque.

The NN model follows the output of the simulated engine closely for the training range, demonstrating that neural network models are a feasible way of capturing the highly nonlinear dynamics of the engine for control.

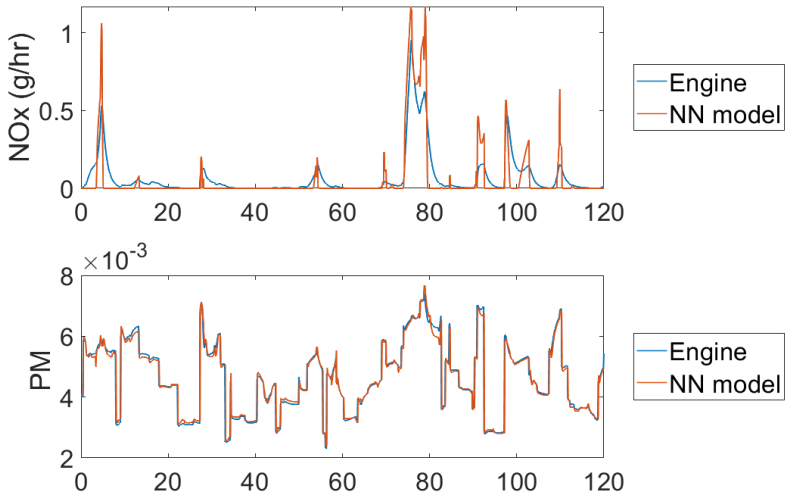

Figure 6.9: Engine model response to validation signal. Top: NO_x, bottom: PM.

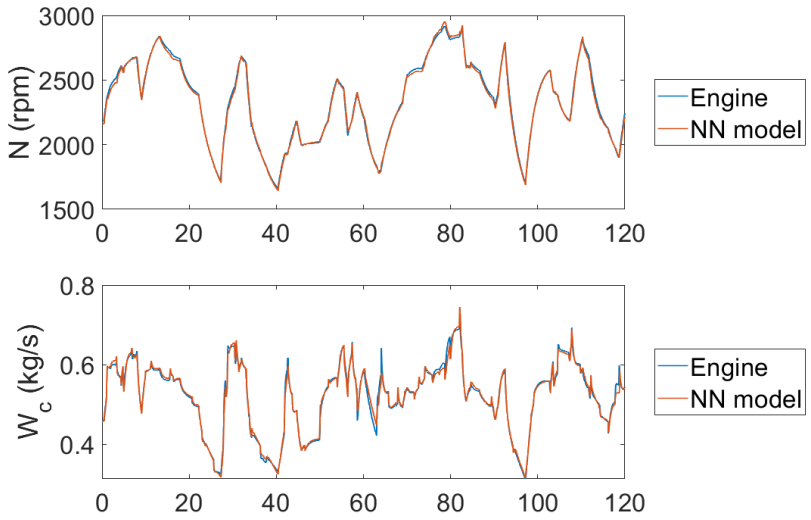

Figure 6.10: Engine model response to validation signal. Top: engine speed, bottom: compressor intake rate.

The NN model of the engine was derived from the training data in figures 6.6, 6.7 and 6.8. The model was then validated against a new set of data in figures 6.9, 6.10 and 6.11. It is shown that NO_x validates more poorly than the other engine outputs, and it is expected that this would also be true when modelling real PM behaviour in an engine. This validates the hypothesis that emissions are particularly difficult to model. However, it is known from chapter 4 that even a low accuracy model will produce a good controller due to its robustness.

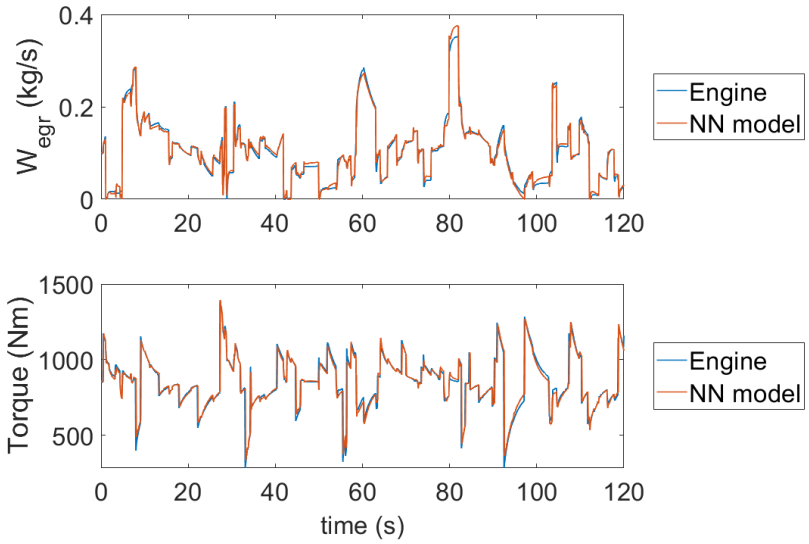

Figure 6.11: Engine model response to validation signal. Top: EGR flow rate, bottom: engine torque.

6.2.2 Engine control

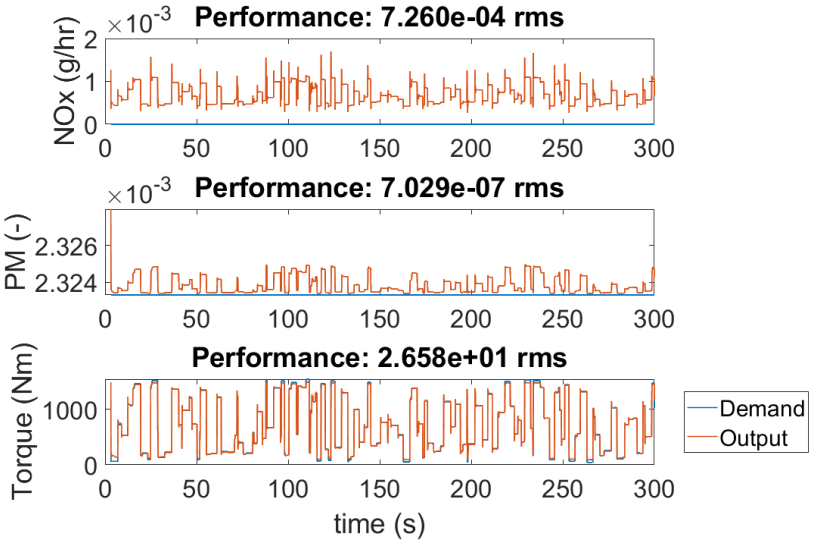

Figure 6.12: Engine controller training results for emissions minimisation

The controller was trained using data in figure 6.12, and a new set of data was used as validation in figure 6.13. The resulting controller tracked torque well, and emissions were kept very close to the minimum the model would allow, of the order of 10^{-3} compared to maximum measured values of the order of 1. Weightings in $\mathbf{\Theta}$ were kept equal as in the setpoint tracking control in chapter 5.

6.3 Conclusion

The control methodology for setpoint tracking described in chapter 5 was adapted to track only torque, but minimising emissions directly at the same time. The controller

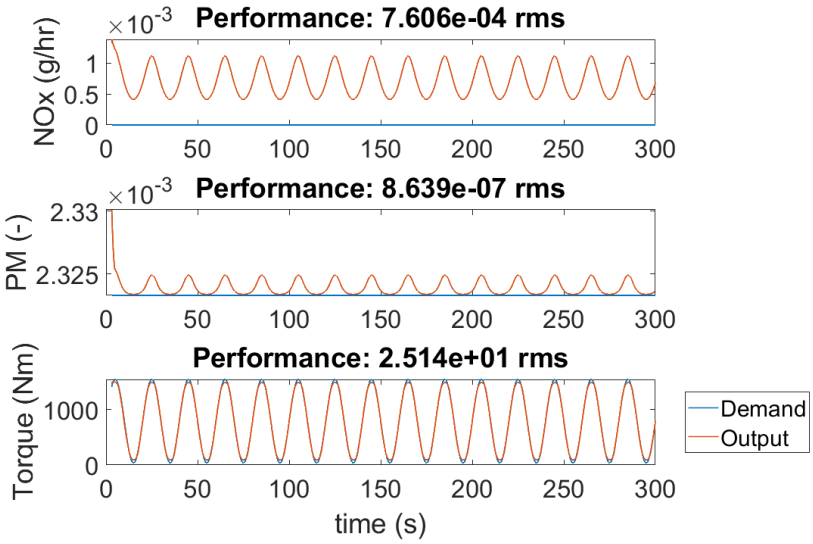

Figure 6.13: Engine controller validation for emissions minimisation

was generated and tested on a validated mean-value MIMO engine model augmented with an emissions model. The NO_x model was based on empirical chemical relationships as a function of peak cylinder temperature and local gas concentrations. A crank-angle domain cylinder temperature model was one of a few steps required to model PM due to its dependence on cylinder history, spray formation and more, but due to vastly different time-scales of the mean-value engine dynamics and in-cylinder temperature, this was not feasible to solve. Instead, PM was represented as a function of fuel injection quantity and oxygen concentration which encourages the controller to minimise quantities correlated to actual PM.

The applied method tracked torque well, and kept NO_x and representative PM close to their respective minima. Further work would validate the controller synthesis method on a real engine, allowing PM to be modelled and therefore minimised directly.

Chapter 7

Test signal optimisation

The neural network based control methodology described in this thesis was initially applied to control LPEGR valves in chapter 4. This method was designed for single-input single-output systems, but then adapted to the control of setpoints for a whole engine in chapter 5, and further extended to directly minimise engine emissions in chapter 6. In all of such incarnations, the method consists of the following three main steps:

1. Measure the relevant system outputs $\boldsymbol{y}$ given a test signal $\boldsymbol{u}$;
2. Derive a neural network model of the system;
3. Derive a neural network controller from the model.

In previous chapters the focus has been on adapting and applying efficient, optimal methods for performing the second and third steps. Up to now, a simple amplitude-modulated pseudo-random binary sequence (APRBS) was used as the test signal in step 1. An APRBS is a signal whose value changes at random time intervals to a random amplitude. An example of APRBS signal is shown in figure 7.1 for reference. As the time interval between changes in amplitude is random, the signal has a broad frequency content which enables the excitation of multiple dynamic modes in the system under test. Furthermore, by varying amplitude randomly between bounds covering the whole operating region, nonlinearities in the system are excited as well [52] [49]. Such features make APRBS extremely useful for generating representative data sets for nonlinear systems identification purposes, e.g. the neural network modelling stage of the methods described in chapters 4, 5 and 6.

However, limiting test signals to standard APRBS means the user is left with only two choices in the design of test signals:

1. Randomly generate APRBS signals, test if a good model can be generated from them, and iterate by trial and error;
2. Generate an APRBS signal that is excessively long, so that enough information is made available to the training algorithm, with a risk of some of this information being redundant.

In both cases, there is no guarantee that all the dynamics modes and nonlinearities of the system under test have been excited during the data collection and identification phases.

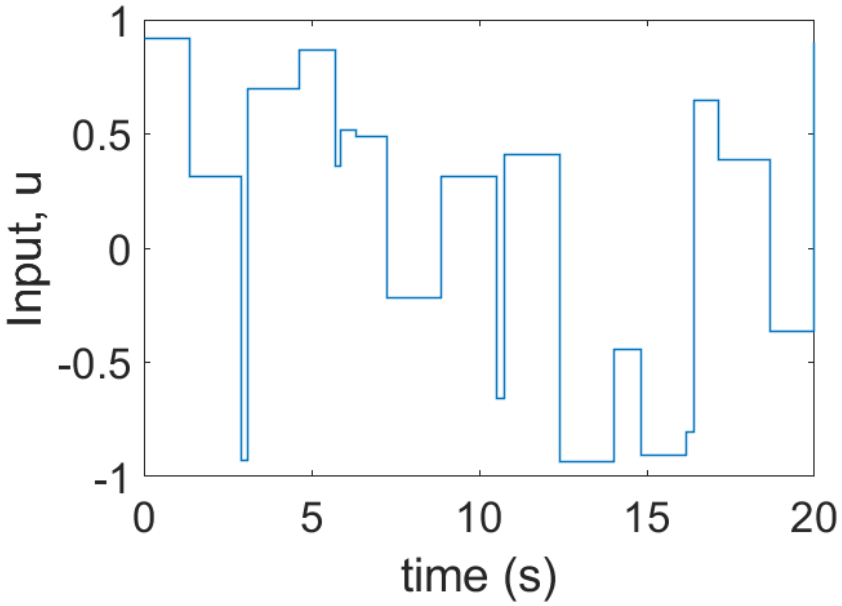

Figure 7.1: Example APRBS signal.

In the case of this project, the second option was used for all chapters so far, as iterating on randomly generated signals by trial and error is time consuming. The resulting signals, and associated data collection phases, still took less time than traditional static calibration, so this was acceptable (and indeed one of the main benefits of the methodology described by this thesis). However, this means that the signal contains redundant information, and could be reduced in size considerably by a more representative test signal design method. This is akin to a dynamic implementation of design of experiments (DoE) [1]. Shorter signals save time during (the traditionally expensive) data collection, and the training algorithm for deriving a model would run faster by having fewer data points to process.

Therefore, a method for designing optimal test signals is applied in this chapter, enabling identification of good, representative models and controllers with a shorter identification process. Initially, the methodology is illustrated for a simple nonlinear SISO system as a proof of concept. The method is then adapted for MIMO systems and used on the whole-engine model.

7.1 Benchmark SISO system

To better illustrate the methodology for test signal optimisation, its application to a simple nonlinear SISO oscillator is described here at first. Such a system is related to the Duffing oscillator model used in chapter 3. However, the gain of the nonlinear cubic term was set to a much higher value to make the system have more pronounced nonlinearity, and the linear stiffness term was removed for the same reason. The oscillator dynamics then reads

$$\ddot{y} + \dot{y} + 20y^3 = u \tag{7.1}$$

7.1.1 Optimisation method

The method described in this chapter builds on the method described by [36], and makes use of pattern search optimisation [67] to design test signals for system identification. Let us then consider a discrete-time nonlinear dynamical system described by

$$y(k) = f(\boldsymbol{\theta}, \boldsymbol{u}(k), \boldsymbol{y}(k)) \tag{7.2}$$

$$\boldsymbol{u}(k) = [u(k), u(k-1), u(k-2), ..., u(k-n)]^T \tag{7.3}$$

$$\boldsymbol{u}(k) = [y(k-1), y(k-2), ..., y(k-n)]^T \tag{7.4}$$

where k is the time step, $\boldsymbol{\theta}$ is the set of system parameters, $u(k)$ and $y(k)$ are the input and output values at time step k. $\boldsymbol{u}(k)$ and $\boldsymbol{u}(k)$ are vectors containing the respective inputs and outputs of the system at time step k and their corresponding values when shifted by incrementally many time steps up to a total of n.

As the objective of the optimisation is to maximise information, a measure of information content called an *information matrix* is used. The information matrix $\boldsymbol{M}$ of a data set with N_k points is calculated according to the Cramer-Rao law [76], [77] as

$$\boldsymbol{M} = \frac{1}{\sigma^2} \sum_{k=1}^{N_k} [\nabla y(k)] [\nabla y(k)]^T \tag{7.5}$$

where $\nabla y(k)$ is the gradient at time step k with respect to the parameters of the model, expressed as a row vector. For a set of N_p parameters, this is given by

$$\nabla y(k) = \left[\frac{\partial y(k)}{\partial \theta_1}, \frac{\partial y(k)}{\partial \theta_2}, \dots \frac{\partial y(k)}{\partial \theta_{(N_p-1)}}, \frac{\partial y(k)}{\partial \theta_{N_p}} \right]^T \tag{7.6}$$

In order to maximise information density of the test signal, a scalar quantity representing information content must be defined. Here, an A-optimal design objective was used, where the trace of the inverse information matrix is minimised. This gives the cost function

$$J_A = tr(\boldsymbol{M}^{-1}) \tag{7.7}$$

At each iteration of the optimisation procedure, equation (7.5) was approximated numerically by simulating the model response with its nominal parameters $\boldsymbol{\theta}$ at first, and the values of y_k recorded for each time step k. Then the parameters $\boldsymbol{\theta}$ were perturbed by their respective floating point relative accuracies $\delta\boldsymbol{\theta}$, and the model was evaluated again to calculate

$$\delta y(k) = f(\boldsymbol{\theta} + \boldsymbol{\delta\theta}, \boldsymbol{u}(k), \boldsymbol{y}(k)) - f(\boldsymbol{\theta}, \boldsymbol{u}(k), \boldsymbol{y}(k)) \tag{7.8}$$

which is the difference between the timeseries over the same set of time steps. The gradient in (7.6) is discretised as

$$\hat{\nabla} y(k) = \left[\frac{\delta y(k)}{\delta \theta_1}, \frac{\delta y(k)}{\delta \theta_2}, \dots \frac{\delta y(k)}{\delta \theta_{(N_p-1)}}, \frac{\delta y(k)}{\delta \theta_{N_p}} \right]^T \tag{7.9}$$

As the parameter set variance σ^2 in equation (7.5) is a constant, it was discarded because the goal of the optimiser is simply to minimise the objective function, so

the absolute value of $\boldsymbol{M}$ is not important. $\boldsymbol{M}$ was therefore calculated as

$$\boldsymbol{M} = \sum_{k=1}^{N_k} \left[\hat{\nabla} y(k)\right] \left[\hat{\nabla} y(k)\right]^T \tag{7.10}$$

and used to calculate the objective function as in (7.7).

Results available in the literature suggest that an optimisation problem of this nature cannot be easily solved with standard gradient-based optimisation methods [76], [77], [100] due to difficult gradients and the large number of optimisation inputs. Therefore, the gradientless pattern search optimisation described in [101], [102], and implemented in the *patternsearch* function in the MATLAB Optimization Toolbox [67], was used. The algorithm works by evaluating an objective function in a *mesh* of points around the initial point $\boldsymbol{x}_0$. An example four point mesh with size η for an optimisation with two degrees of freedom is given by

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \eta \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{7.11}$$

$$\boldsymbol{x}_2 = \boldsymbol{x}_0 + \eta \begin{bmatrix} -1 \\ 0 \end{bmatrix} \tag{7.12}$$

$$\boldsymbol{x}_3 = \boldsymbol{x}_0 + \eta \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{7.13}$$

$$\boldsymbol{x}_4 = \boldsymbol{x}_0 + \eta \begin{bmatrix} 0 \\ -1 \end{bmatrix} \tag{7.14}$$

In the case of this research, each point in the mesh is a timeseries test signal, and each time step in the signal is a degree of freedom in the optimisation. In brief, the pattern search algorithm can be summarised as follows:

1. Set current test signal to the initial APRBS signal

2. Set mesh size as 1

3. Evaluate cost function J_A for the current test signal

4. Add the current mesh size to the first point in the test signal, evaluate J_A again. Repeat for each point in the signal.

5. If a lower value for the cost function is found, set that signal as the current signal, multiply mesh size by 2 and iterate.

6. If a lower value is not found, divide the current mesh size by 2 and iterate.

The algorithm iterates until stopped manually or until mesh size falls below 0.25.

7.1.2 Dynamic polynomial modelling

In order to derive the information matrix for the current system and test signal, the system must be modelled and parameterised by $\boldsymbol{\theta}$. An initial test signal was used to drive the system described by equation (7.1), yielding a set of timeseries data $\boldsymbol{u}$ and $\boldsymbol{y}$. This data was used to derive a dynamic polynomial model as a linear least squares regression problem

$$\boldsymbol{X\theta} = \boldsymbol{y} \tag{7.15}$$

where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X_u} & \boldsymbol{X_y} & \boldsymbol{X_c} \end{bmatrix} \tag{7.16}$$

$$\boldsymbol{X_u} = \begin{bmatrix} \boldsymbol{u}(1) & \boldsymbol{u}(1)^2 & \dots & \boldsymbol{u}(1)^{d-1} & \boldsymbol{u}(1)^d \\ \boldsymbol{u}(2) & \boldsymbol{u}(2)^2 & \dots & \boldsymbol{u}(2)^{d-1} & \boldsymbol{u}(2)^d \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{u}(N_k-1) & \boldsymbol{u}(N_k-1)^2 & \dots & \boldsymbol{u}(N_k-1)^{d-1} & \boldsymbol{u}(N_k-1)^d \\ \boldsymbol{u}(N_k) & \boldsymbol{u}(N_k)^2 & \dots & \boldsymbol{u}(N_k)^{d-1} & \boldsymbol{u}(N_k)^d \end{bmatrix} \tag{7.17}$$

$$\boldsymbol{X_y} = \begin{bmatrix} \boldsymbol{y}(1) & \boldsymbol{y}(1)^2 & \dots & \boldsymbol{y}(1)^{d-1} & \boldsymbol{y}(1)^d \\ \boldsymbol{y}(2) & \boldsymbol{y}(2)^2 & \dots & \boldsymbol{y}(2)^{d-1} & \boldsymbol{y}(2)^d \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{y}(N_k-1) & \boldsymbol{y}(N_k-1)^2 & \dots & \boldsymbol{y}(N_k-1)^{d-1} & \boldsymbol{y}(N_k-1)^d \\ \boldsymbol{y}(N_k) & \boldsymbol{y}(N_k)^2 & \dots & \boldsymbol{y}(N_k)^{d-1} & \boldsymbol{y}(N_k)^d \end{bmatrix} \tag{7.18}$$

and d represents the maximum polynomial degree set by the user. The polynomial model parameters $\boldsymbol{\theta}$ in (7.15) can be solved using the linear regression capabilities of MATLAB [67].

$$\boldsymbol{X_c} = \begin{bmatrix} \boldsymbol{y}(1)\boldsymbol{y}(2) & \boldsymbol{y}(1)^2\boldsymbol{y}(2) & \boldsymbol{y}(1)\boldsymbol{y}(2)^2 & \dots & \boldsymbol{y}(1)^{\frac{d}{2}}\boldsymbol{y}(2)^{\frac{d}{2}} & \dots \\ \vdots & \ddots & \vdots & \vdots & \vdots & \\ \boldsymbol{y}(N_k-1)\boldsymbol{y}(N_k) & \boldsymbol{y}(N_k-1)^2\boldsymbol{y}(N_k) & \boldsymbol{y}(1)\boldsymbol{y}(2)^2 & \dots & \boldsymbol{y}(1)^{\frac{d}{2}}\boldsymbol{y}(2)^{\frac{d}{2}} & \dots \end{bmatrix} \tag{7.19}$$

The full matrix $\boldsymbol{X_c}$ is too large to display in the thesis, but basically $\boldsymbol{X_c}$ contains all the pairwise multiplications between elements in $\boldsymbol{u}(k)$ and $\boldsymbol{y}(k)$ up to exponent d. In all cases, the time step argument k is limited to be no less than 1, as this would lie outside the data set. Therefore values in previous time steps are assumed to be equal to those at $k = 1$, i.e. the first data point in the timeseries is assumed to be an initial steady state.

Given the large set of parameters to be estimated, rather than selecting polynomial terms by trial and error and/or knowledge of the system, the model structure was selected algorithmically. The model selection algorithm was designed to perform the following steps:

1. Fit model to data for current parameter set;
2. Validate on full model and record mean squared error;
3. For each parameter in the set $\boldsymbol{\theta}$:
   (a) Set current parameter coefficient to 0,
   (b) Re-validate model and record error.

4. Permanently remove from the set $\boldsymbol{\theta}$ the parameter that caused the smallest increase in error;

5. Repeat from step 1 until tolerance or minimum parameter set length is met.

It should be noted that by reducing the parameter set and re-fitting, the model error will tend to decrease, so tolerance is defined in terms of change in error, rather than absolute error. Therefore when error has not decreased sufficiently by reducing the parameter set, the algorithm stops.

Once a polynomial model structure has been selected using the described algorithm, the derivatives in information matrix $\boldsymbol{M}$ are calculated by finite difference, i.e. by perturbing each polynomial parameter by their corresponding floating point minimum increments and evaluating the model. Once an optimal test signal is generated, measurements can be taken using the new test signal and the whole method repeated to iteratively improve the signal by starting with a better initial polynomial model. In this case, three iterations were done, and in practice is limited by the number of repeat experiments that are feasible.

7.1.3 Results

An initial model of the system described by equation (7.1) was modelled as a dynamic polynomial using maximum degree $d = 3$ and time delays 1, 2 and 3 included for input and output terms. To initialise the optimisation procedure, an initial test signal consisting of two steps of amplitudes 1 and -1 respectively was considered, as shown by figure 7.2.

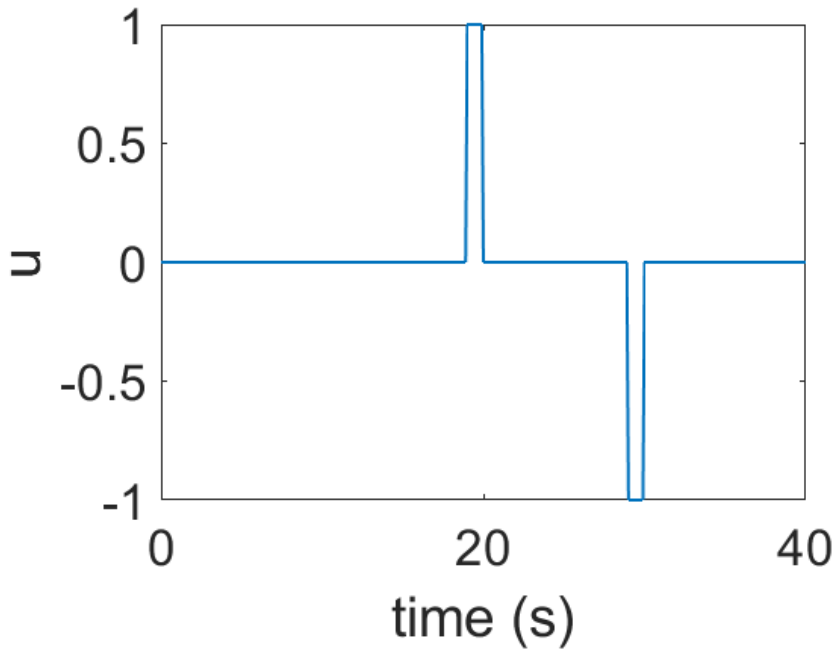

Figure 7.2: Test signal used to initialise optimisation procedure.

Such signal was used to drive the system in simulation and its output was recorded. The initial polynomial model parameter set is given in table 7.1, and the model response versus simulated data is shown in figure 7.3.

Table 7.1: Parameter set for initial polynomial model

Parameter	Polynomial term	Value
θ_1	$u(k-1)^3$	-0.0691
θ_2	$u(k-1)^2u(k-2)$	0.7825
θ_3	$u(k-1)^2$	-0.1485
θ_4	$u(k-2)^3$	-0.2170
θ_5	$u(k-2)^2$	0.1604
θ_6	$y(k-1)^3$	0.0362
θ_7	$y(k-1)y(k-2)^2$	-0.1323
θ_8	$y(k-1)y(k-2)y(k-3)$	-0.1131
θ_9	$y(k-1)y(k-3)^2$	0.1442
θ_{10}	$y(k-1)$	1.0427
θ_{11}	$y(k-2)^3$	0.1519
θ_{12}	$y(k-2)y(k-3)^2$	-0.0873
θ_{13}	$y(k-2)y(k-3)$	0.0006
θ_{14}	$y(k-2)$	0.2921
θ_{15}	$y(k-3)$	-0.3479

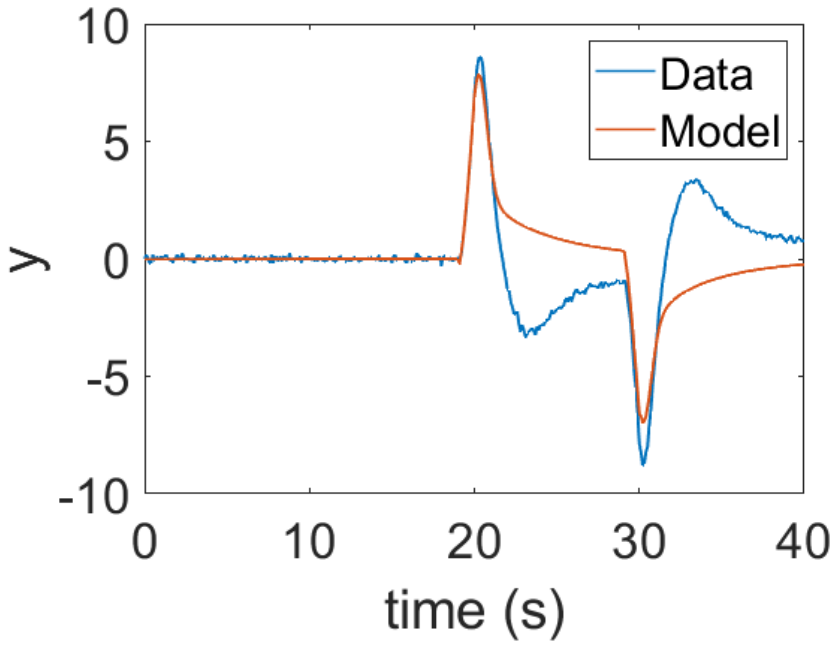

Figure 7.3: Initial model response

The initial test signal from figure 7.2 was optimised for the given model by implementing the process described in sections 7.1.1 and 7.1.2. Then the system was driven by the new signal, data was collected and a new model was derived. These steps were repeated 3 times, resulting in 3 sets of experiments in total.

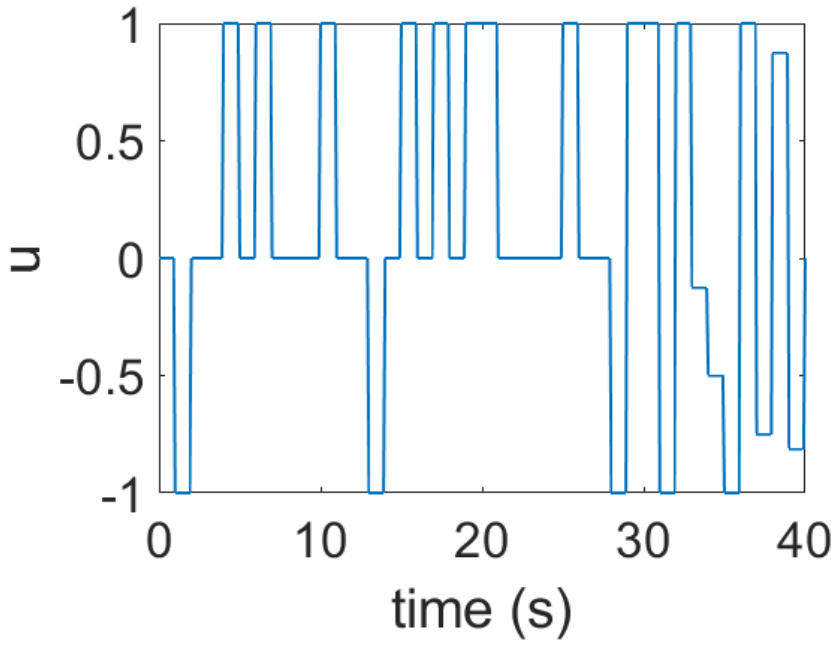

Figure 7.4: Optimised test signal.

The optimised test signal shown in figure 7.4 has increased complexity compared to the initial test signal from figure 7.2. As a result of better test signal design, the final model in figure 7.5 validates qualitatively better than the initial one in figure 7.3.

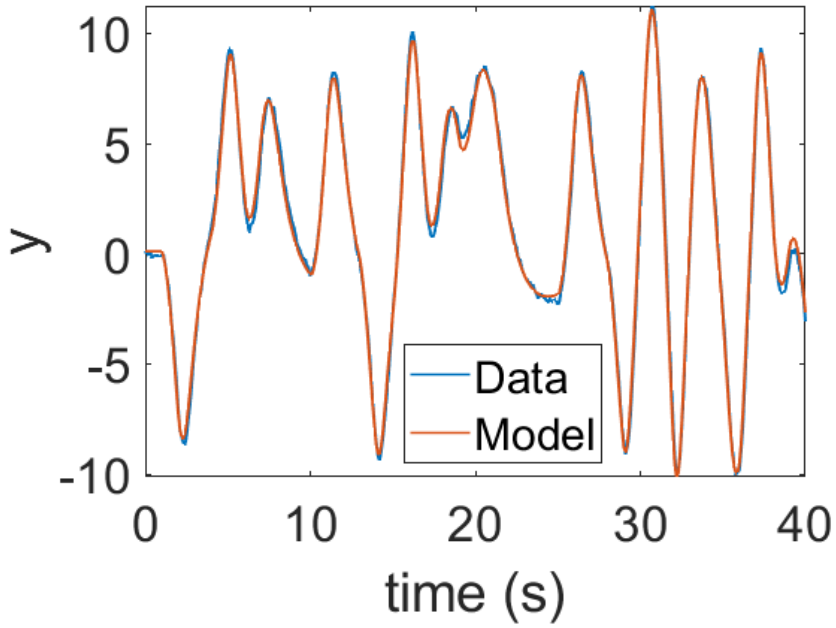

Figure 7.5: Optimised model response.

The coefficients of the optimised model are reported in table 7.2. This follows the expectation that by minimising $tr(\boldsymbol{M}^{-1})$, the information density of the signal is increased, providing richer datasets for the purposes of system identification.

Table 7.2: Parameter set for final polynomial model.

Parameter	Polynomial term	Value
θ_1	$u(k-1)^2u(k-2)$	-0.0012
θ_2	$u(k-1)^2$	-0.0701
θ_3	$u(k-1)u(k-2)$	0.0006
θ_4	$u(k-1)$	0.1488
θ_5	$u(k-2)^2$	0.0699
θ_6	$u(k-2)$	0.1657
θ_7	$y(k-1)^2y(k-3)$	0.0255
θ_8	$y(k-1)y(k-2)y(k-3)$	-0.1157
θ_9	$y(k-1)y(k-3)^2$	0.0646
θ_{10}	$y(k-1)$	1.2198
θ_{11}	$y(k-2)^2y(k-3)$	0.1239
θ_{12}	$y(k-2)y(k-3)^2$	-0.1359
θ_{13}	$y(k-2)$	0.3915
θ_{14}	$y(k-3)^3$	0.0368
θ_{15}	$y(k-3)^2$	-0.0001
θ_{16}	$y(k-3)$	-0.6135

7.2 Whole-engine system

The method described in section 7.1 for optimising test signals of SISO systems can be further extended to MIMO systems as well. To illustrate this, the whole-engine Simulink model introduced in chapter 5 is considered in this section. Within this scenario, the MIMO system to be identified has the form

$$\boldsymbol{y_{meas}} = [W_c, W_{EGR}, n_e]^T \tag{7.20}$$

$$\boldsymbol{y_{obs}} = [\lambda, x_{EGR}, M_e]^T \tag{7.21}$$

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y_{meas}} \\ \boldsymbol{y_{obs}} \end{bmatrix} \tag{7.22}$$

$$\boldsymbol{u} = [u_{VGT}, u_{EGR}, u_\delta]^T \tag{7.23}$$

where $\boldsymbol{y}$ at time step k is given by the dynamic function f

$$\begin{aligned} \boldsymbol{y}(k) = f(&\boldsymbol{u}(k), \boldsymbol{u}(k-1), \ldots, \boldsymbol{u}(2), \boldsymbol{u}(1), \\ &\boldsymbol{y}(k-1), \boldsymbol{y}(k-2), \ldots, \boldsymbol{y}(2), \boldsymbol{y}(1)) \end{aligned} \tag{7.24}$$

Given that the final aim of the identification process is facilitating controller design in this thesis, the outputs of interest are split into two categories: measurable quantities, $\boldsymbol{y_{meas}}$, and quantities which must be estimated by an observer, $\boldsymbol{y_{obs}}$. The controller is optimised based on a neural network model identified from data measured on an engine test bed, where a greater range of sensors are available, so for the identification stage they are grouped together into vector $\boldsymbol{y}$.

The test signal optimisation for the aforementioned SISO system was repeated for the identification portion of the controller synthesis method described in chapter 5. The information matrix is defined in the same way as equation (7.5), except that

$\boldsymbol{\theta}$ is not a set of polynomial coefficients. Due to the complexity of the full engine model, polynomial modelling was not suitable. Instead, the parameter set is defined as all the weights and biases of the neural network from figure 5.4

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{b} \end{bmatrix} \tag{7.25}$$

where $\boldsymbol{w}$ and $\boldsymbol{b}$ are the sets of all network weights and all biases respectively. The derivatives in equation (7.5) are otherwise calculated in the same way as described in subsection 7.1.1.

7.2.1 Results

After running the optimisation starting with the initial test signal shown in figure 7.6, the result is the new optimised test signal in figure 7.7.

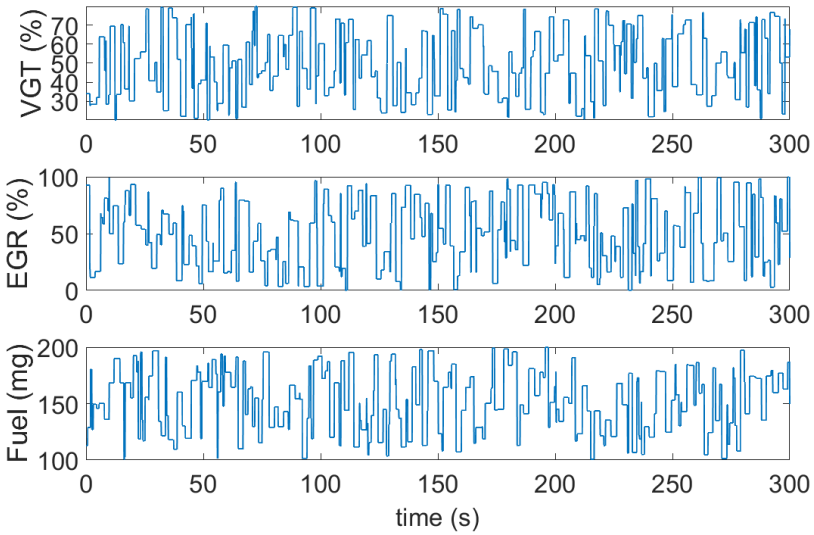

Figure 7.6: Initial test signal for engine model training

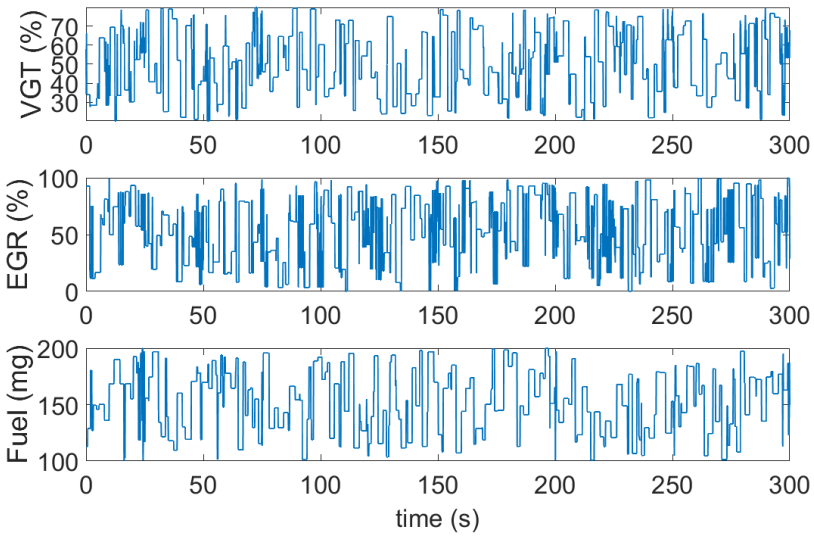

Figure 7.7: Optimised test signal for engine model training

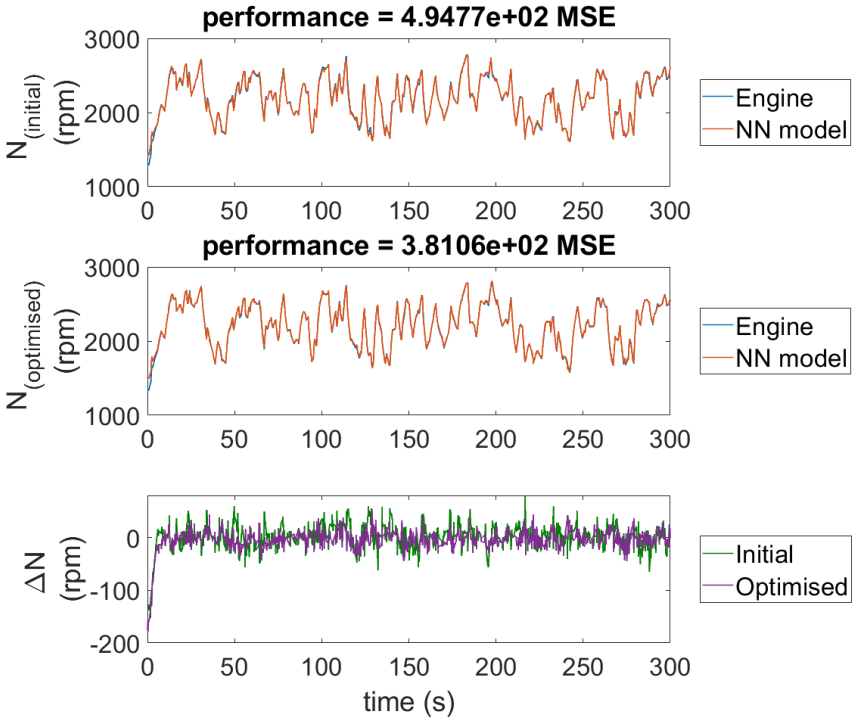

Figure 7.8: Comparison of initial and optimised test signals on engine speed modelling

The neural network models identified using the initial and optimised test signals are compared. Figure 7.8 compares the engine speed output of the two neural network models to the output of the whole-engine system. The performance of the neural network derived from an optimised test signal in the second subplot achieves an improvement in mean-squared error compared to the that of the initial neural network in the first subplot and the corresponding error signal in the third subplot is shown to be flatter overall.

Compressor flow rate accuracy is compared in figure 7.9 and shows that mean-squared error increased slightly. However, the error signal of the initial neural network model is shown to have larger instantaneous spikes compared to that of the optimised neural network, offsetting the small overall increase in error.

Error predicting mass flow rate through the EGR is significantly improved as indicated by figure 7.10, with a large reduction in mean-squared error and the elimination of many large spikes in error which were characteristic of the initial neural network model. As with engine speed, the decrease in error is more significant than the increase in error shown for W_c in the previous figure 7.9, suggesting a trade-off in accuracy but an overall improvement when considering all outputs of both models.

The optimised test signal model for air-fuel equivalence ratio λ in figure 7.11 had increased mean-squared error compared to the initial model, with spikes in the error signal comparable to or exceeding those of the initial model.

The new model improves on EGR fraction accuracy significantly in terms of mean-squared error, reducing it from 3.0×10^{-4} to 1.8×10^{-4}. Furthermore, in figure 7.12 the initial model showed large instantaneous spikes in its error signal which are mostly eliminated by the new model, notably one at around 90 seconds and another at around 150 seconds.

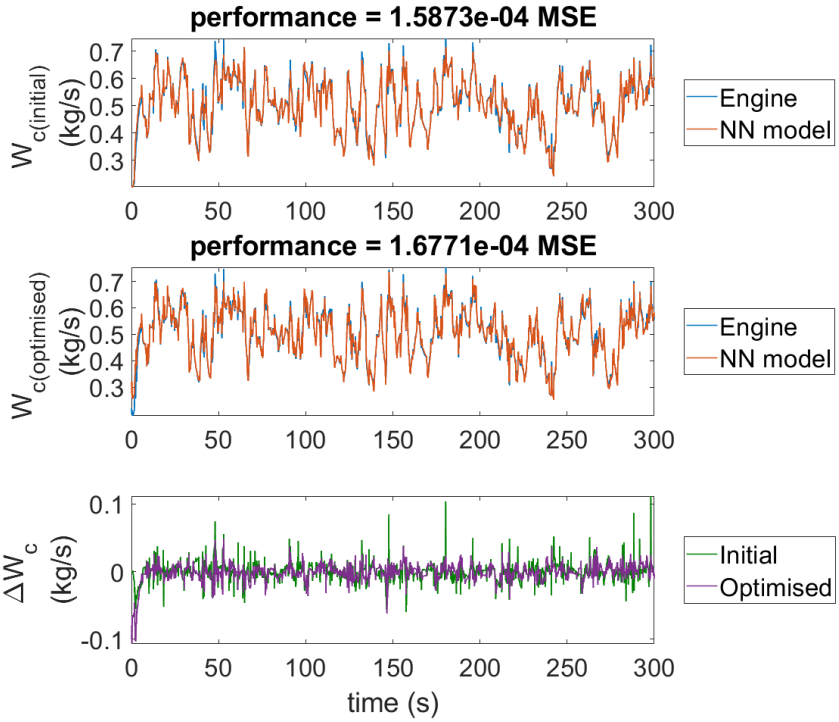

Figure 7.9: Comparison of initial and optimised test signals on compressor flow rate modelling

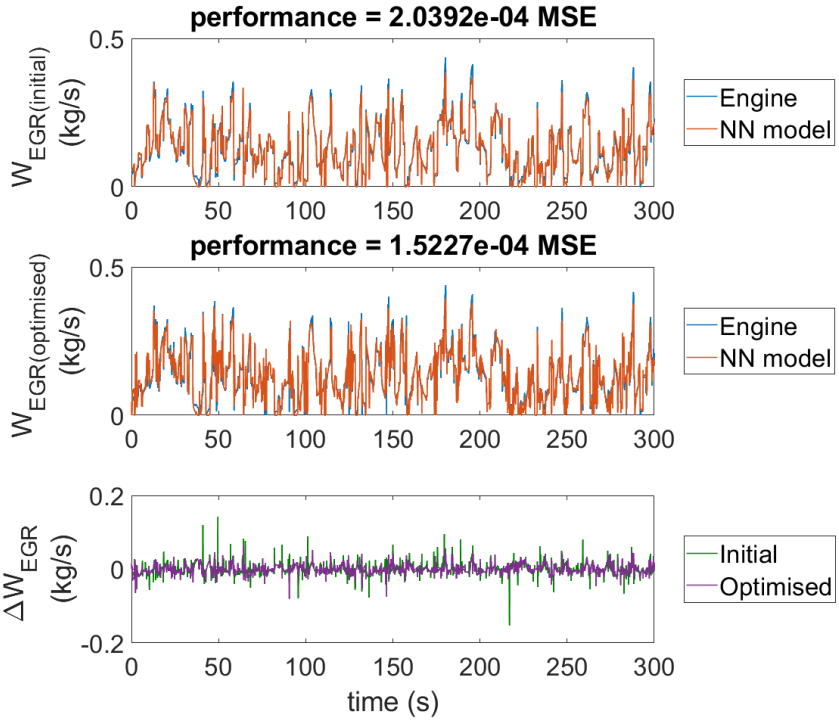

Figure 7.10: Comparison of initial and optimised test signals on EGR flow rate modelling

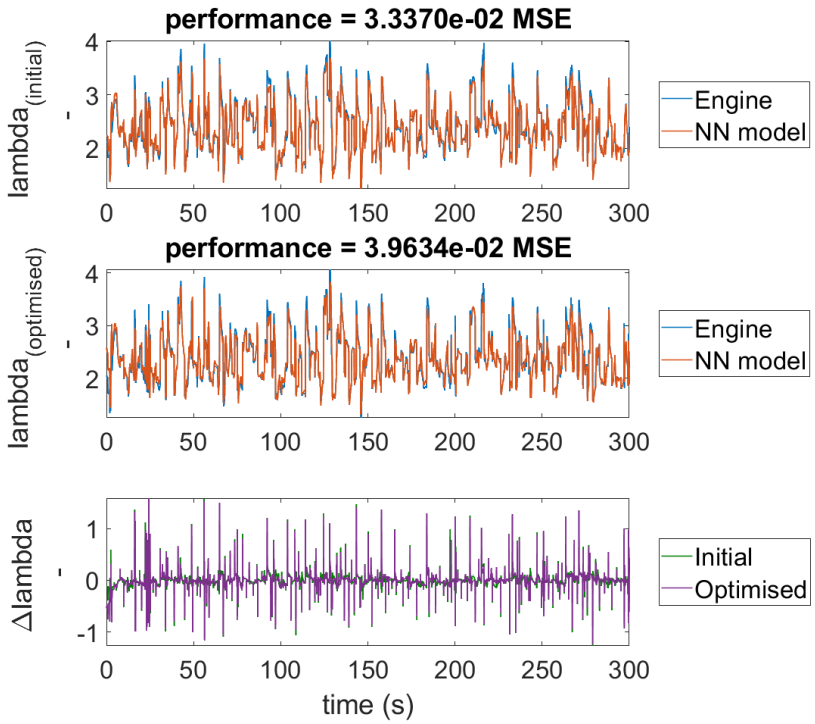

Figure 7.11: Comparison of initial and optimised test signals on air-fuel equivalence ratio modelling

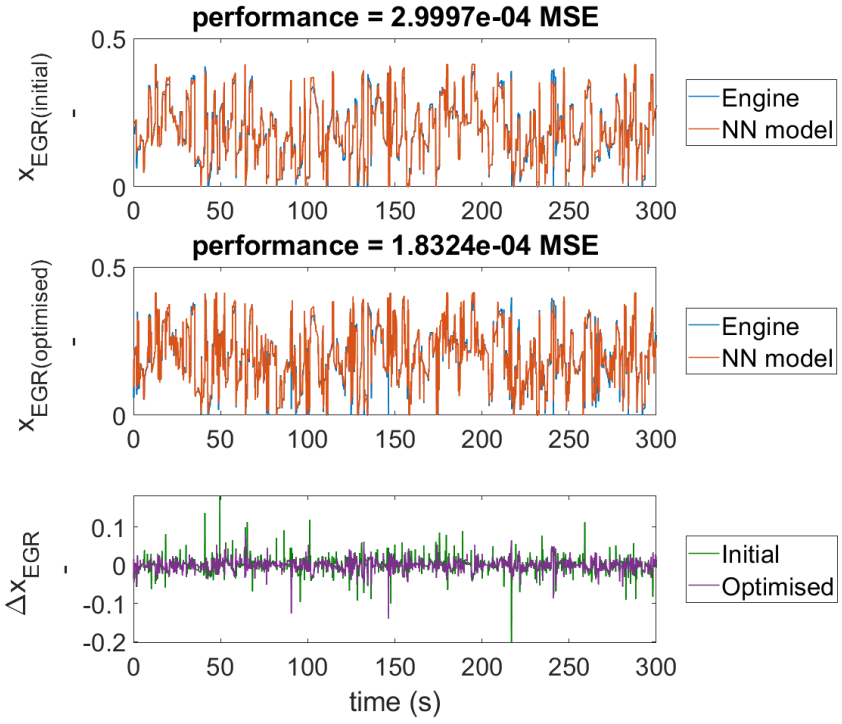

Figure 7.12: Comparison of initial and optimised test signals on EGR fraction modelling

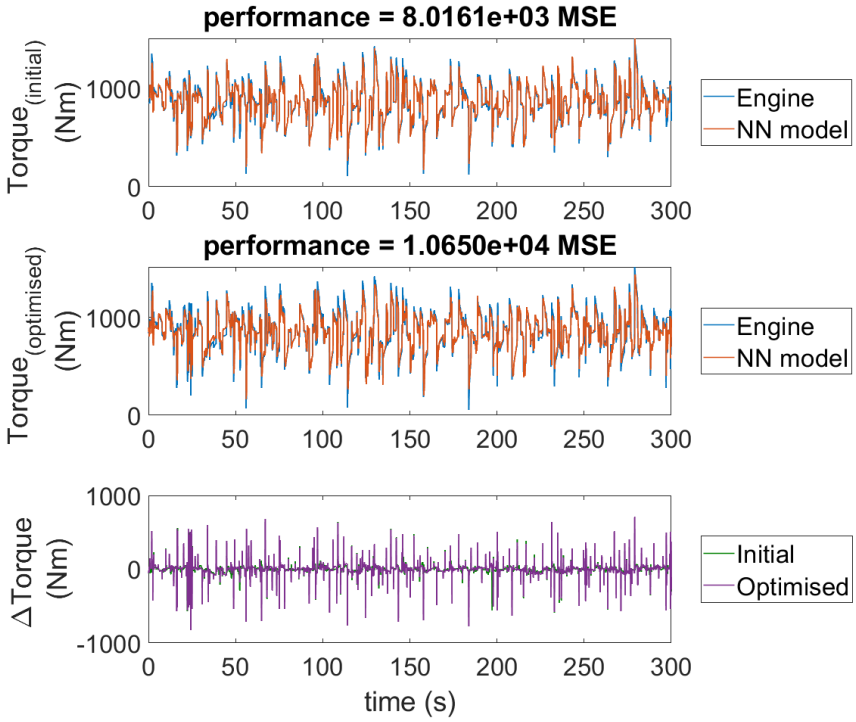

Figure 7.13: Comparison of initial and optimised test signals on torque modelling

Like with air-fuel equivalence ratio, torque modelling error was increased significantly, as shown by figure 7.13.

It is shown that the error for some outputs is decreased and for others it is increased. The mean-squared errors of both initial and optimised models are summarised in table 7.3 with a percentage change in error for each model output. The mean average change in error is also shown, indicating an overall reduction in error of 5%.

Table 7.3: Summary of neural network model output errors

Output variable	Before (MSE)	After (MSE)	Change in error
N	4.948e+02 $(rpm)^2$	3.811e+02 $(rpm)^2$	-23 %
W_c	1.587e-04 $(kg/s)^2$	1.677e-04 $(kg/s)^2$	6 %
W_{EGR}	2.039e-04 $(kg/s)^2$	1.523e-04 $(kg/s)^2$	-25 %
$lambda$	3.337e-02	3.963e-02	19 %
x_{EGR}	3.000e-04	1.832e-04	-39 %
Torque	8.016e+03 $(Nm)^2$	1.065e+04 $(Nm)^2$	33 %
		Mean	**-5 %**

As the purpose of the neural network model of the whole-engine is to derive a controller, a controller is derived for each as described in chapter 5 and compared. It is shown in figure 7.15 that the controller derived using a neural network trained using an optimised test signal achieves an improvement in tracking error, in contrast to the initial controller which was derived using a neural network trained with an APRBS signal in figure 7.14.

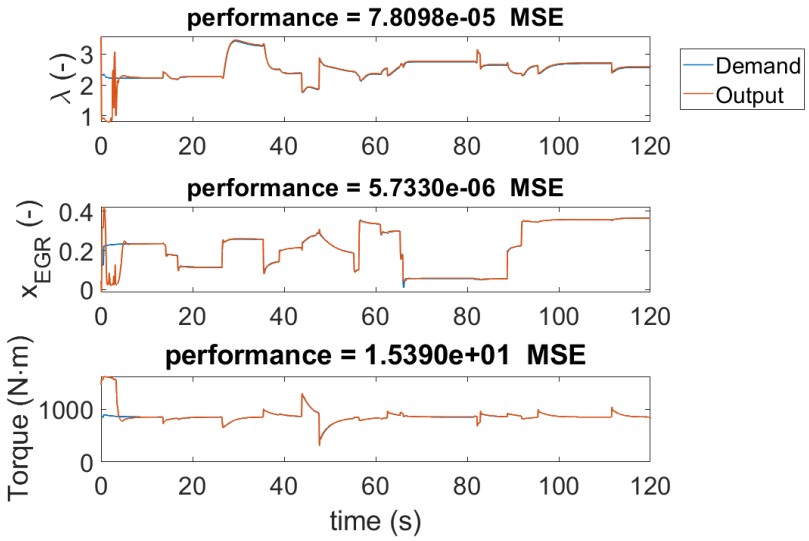

Figure 7.14: Initial test signal engine controller validation

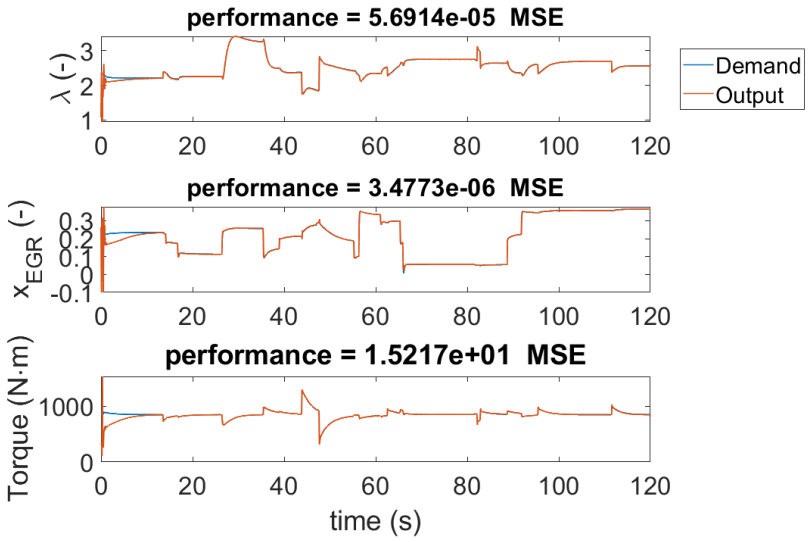

Figure 7.15: Optimal test signal engine controller validation

Table 7.4: Error summary for initial and optimised test signal controllers

Controlled output	Before (MSE)	After (MSE)	Change in error
λ	7.81e-05	5.69e-05	-27%
x_{EGR}	5.73e-06	3.48e-06	-39%
Torque	1.54e+01 $(Nm)^2$	1.52e+01 $(Nm)^2$	-1%
		Mean	**-23%**

The changes in tracking error are summarised in table 7.4, where the tracking for variables λ and x_{EGR} were found to be particularly improved. A considerably smaller improvement was seen in torque tracking, which could be due to the neural network model losing accuracy in predicting torque, compared to the initial model.

Accounting for the very small improvement rather than a decrease in tracking error, torque control is in part determined by the observer, which uses all model outputs to predict the system's state. By improving the mean average accuracy of the whole model, torque control was improved very slightly instead of becoming worse. It was concluded previously in chapter 4 on valve control that controllers could be derived from relatively poor accuracy models. In the case of the valves examined, this was due to the influence of static friction and age of the valve, as a newer valve proved to be easier to model. In this case, it is shown that sacrificing accuracy in some of the neural network outputs in order to increase average accuracy of all outputs is still beneficial for the final derived controller, where a small decrease in model error of 5% was shown to decrease controller tracking error by an average of 23%.

7.3 Conclusion

A method for optimal design of test signals for system identification was implemented and validated using a simple nonlinear SISO system. This is done by characterising the system as a polynomial model using an initial test signal of low information density. Information content was quantified using the trace of the information matrix M, composed of derivatives which represent the sensitivity of the model output in changes to its parameters. This was repeated 3 times from taking data, deriving a model and optimising the signal, using the previous optimised signal as the new initial one.

In practice, an optimal signal could be produced in fewer iterations by providing a higher information density signal as the initial signal, however an intentionally poor initial signal was used here to test the method on a simple SISO system. It is shown in previous chapters that APRBS signals are naturally information dense, so these would be a sensible starting point.

The test signal optimisation method was adapted for a MIMO whole-engine system, represented by a mean-value Simulink model [68]. The APRBS test signal used in chapter 5 was used as an initial signal to be optimised. A neural network model derived from the optimised test signal was found to reduce the error of some output variables, and increase others, resulting in a mean average improvement of a -5% decrease in error.

A controller was derived using the new neural network model and shown to significantly decrease tracking error for λ and x_{EGR}, with a smaller decrease for torque tracking. The smaller improvement in torque control is attributed to the higher torque error in the new neural network model, but is offset by the overall improvement in error for other output variables. It is suggested that this is due to the observer using all outputs, resulting in an overall improvement when the controller comes to predict the system's internal state.

Chapter 8

Conclusions and potential further work

To meet increasingly strict legislative limits for diesel engines, while reducing the effort required to calibrate them, a set of methods were adapted to improve the performance of airpath controllers. Current practice in static design of experiments (DoE) and engine control unit (ECU) calibration were examined, then extended to a dynamic approach which captures the behaviour of transients while reducing the time taken to perform a calibration.

To achieve this, a dynamic calibration method was initially applied to a single engine component, an exhaust gas recirculation valve (EGR), to test the calibration method. Having successfully calibrated a single component, the method was extended to calibrating a multi-component simulated engine, with the aim of controlling three quantities important for controlling emissions. In standard static DoE, such quantities are defined as setpoints and used as targets when performing a static calibration. The same method was then applied to minimise emissions directly without the use of pre-defined setpoints. Finally, the pseudorandom test signals initially used to derive these controllers were replaced with an optimised test signal using a method adapted to resemble a dynamic form of engine DoE.

In Chapter 3, a statistical method was applied to the analysis of dynamic systems, including a simulated engine, to test the confidence that a system exhibits purely linear behaviour, which offered insight into the appropriate complexity of nonlinear models or controllers for a given system, or if a nonlinear model or controller is appropriate at all.

In Chapter 4, a nonlinear, dynamic feedforward-feedback neural network control method was implemented and applied to an exhaust gas recirculation valve. The experimental rig consisted of an ECU with the standard PI controller for the valve disabled, and a laptop to modify calibration parameters in real-time. The calibration parameter to vary was duty-cycle to the valve motor, recording its position response as a time-series. The input signal was designed to be information-dense by varying a step input as an amplitude-modulated pseudorandom signal (APRBS). To collect this data, the duty cycle field was adjusted in real-time via scripted simulated keypresses, as standard engine calibration software does not easily allow this. The data was used to train a neural network model of the valve, which was then used to train a feedforward-feedback integral action neural network controller offline.

The performance of the derived controller was validated in the loop and docu-

mented, showing a significant improvement in position tracking. Notably, standard staged PI-controllers for engine valves must be tuned online manually, which is a very time-consuming process involving 24 cross-coupled parameters. By automating the initial data collection, as well as using neural network training algorithms for the controller tuning, the time taken to calibrate the valve was greatly reduced. On a cheap office laptop the total time for controller design was reduced to 10-15 minutes, with minimal manual interruption, satisfying objective 1 of section 1.2.

In Chapter 5, the method for the control of valve position was adapted for the control of multiple air-path parameters on a validated whole-engine simulation, representing an engine on a testbed. Unlike with the valve position control, the parameters of interest here cannot be sensed by a production engine. To adapt the method, the neural network controller was augmented with an observer to estimate the internal state of the engine based on sensors or maps that are available for feedback in the ECU. The adapted method was applied to two cases: setpoint control and emissions reduction.

Setpoint control is closer to the static DoE methods used in production, but uses time-series like with the valve control in order to reduce labour and time. This method controls torque, air-to-fuel ratio and exhaust gas ratio simultaneously, according to the operating points considered important by a standard DoE. These are typically set in order to meet emissions limits. The controller tracked well for an APRBS of varying setpoint demands, which included all setpoints specified by DoE, satisfying objective 2 of section 1.2.

The second case explored in Chapter 6 is a more direct approach to emissions reduction by taking measurements of NOx and PM during the experimental phase. These are included during the model training, and specified as quantities to minimise, rather than control with a demand. The controller tracked torque well for an APRBS test signal, while emissions varied closely to the model lower bounds of both NOx and particulate matter (PM), satisfying objective 3 of section 1.2. For this method, an emissions model had to be implemented for the whole-engine simulation. NOx was simulated using an ideal Seiliger cycle for temperature and pressure, and standard chemical rate constants which vary with pressure, temperature and oxygen concentration. These rate constants are widely published in introductory texts and are empirical in nature.

As models of PM are highly stochastic, nonlinear and fast, this created a stiffness problem in the whole-engine simulation which could not be easily resolved. NOx and PM are related to temperature, fuel and oxygen concentration in inverse ways, so excluding the PM model entirely would result in excessive PM emissions in a production engine, albeit with unrealistically low NOx emissions. PM was instead represented by fuel-to-oxygen ratio, as more fuel results in more PM, and more oxygen results in less. This has the additional benefit of penalising fuel consumption during optimisation. It is recommended for further work that this method be carried out on a testbed and engine as originally intended, or at least with a more detailed emissions model.

Finally, in Chapter 7 a method for designing test signals without relying on randomisation via APRBS generation was applied to two systems. The method made use of gradientless pattern search optimisation to design a step signal which maximised its information content. This was applied to benchmark SISO systems and to a MIMO simulated engine, showing a mean reduction in error for models

derived from the optimised signal. Furthermore, a mean reduction in tracking error for a controller generated from the derived model was shown. This is comparable to a dynamic approach to static DoE, and represents a huge step needed if dynamic methods are to be adopted, satisfying objective 4 in section 1.2.

Bibliography

[1] K. Röpke and C. V. Essen, "DoE in engine development," in *The European Network for Business and Industrial Statistics (ENBIS)*, IAV GmbH, vol. 24, John Wiley and Sons Ltd, 2008, ch. 6, pp. 643–651, ISBN: 07488017. [Online]. Available: http://dx.doi.org/10.1002/qre.941.

[2] R. Stone, *Introduction to Internal Combustion Engines*, 2nd ed. Hampshire: Macmillan Press LTD, 1992.

[3] I. Kolmanovsky, M. V. N. P. Morall, and A. Stefanopoulou, "Issues in modelling and control of intake flow in variable geometry turbocharged engines," *Chapman and Hall CRC research notes in mathematics*, pp. 436–445, 1999.

[4] H. Uchida, "Trend of turbocharging technologies," version 41, *R&D Review of Toyota CRDL*, no. 3, pp. 1–8, 2006.

[5] A. G. Ulsoy, H. Peng, and M. Çakmakci, *Automotive control systems*. Cambridge University Press, 2012.

[6] S. Karagiorgis, K. Glover, and N. Collings, "Control challenges in automotive engine management," Department of Engineering, University of Campbidge, United Kingdom, vol. 13, Lavoisier, 2007, ch. 2-3, pp. 92–104, ISBN: 09473580. [Online]. Available: http://dx.doi.org/10.3166/ejc.13.92-104.

[7] Wikipedia contributors, *European emission standards — Wikipedia, the free encyclopedia*, [Online; accessed 30-July-2019], 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=European_emission_standards&oldid=903564893.

[8] 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, pp. 293–301, Dec. 2007. DOI: 10.1016/j.arcontrol.2007.09.001.

[9] E Alfieri, A Amstutz, and L Guzzella, "Gain-scheduled model-based feedback control of the air/fuel ratio in diesel engines," *Control Engineering Practice*, vol. 17, no. 12, pp. 1417–1425, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.conengprac.2008.12.008.

[10] J. S. Shamma and M. Athans, "Gain scheduling: Potential hazards and possible remedies," *IEEE Control Systems*, vol. 12, no. 3, pp. 101–107, 1992.

[11] N. Pavel, M. Bärwinkel, P. Heinz, D. Brüggemann, G. Dearden, G. Croitoru, and O. V. Grigore, "Laser ignition - spark plug development and application in reciprocating engines," *Progress in Quantum Electronics*, vol. 58, pp. 1 –32, 2018, ISSN: 0079-6727. DOI: 10.1016/j.pquantelec.2018.04.001.

[12] N. Pavel, R. Chiriac, A. Birtas, F. Draghici, and M. Dinca, "On the improvement by laser ignition of the performances of a passenger car gasoline engine," *Opt. Express*, vol. 27, pp. 385–396, 2019.

[13] P. Patane and M. Nandgaonkar, "Review: Multipoint laser ignition system and its applications to ic engines," *Optics & Laser Technology*, vol. 130, 2020, ISSN: 0030-3992. DOI: 10.1016/j.optlastec.2020.106305.

[14] K. Reif, *Fundamentals of Automotive and Engine Technology*. Wiesbaden, Germany: Springer Vieweg, 2014. DOI: 10.1007/978-3-658-03972-1.

[15] K. Mollenhauer and H. Tschöke, *Handbook of Diesel Engines*. Berlin, Germany: Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-540-89083-6.

[16] J. Lumley, *Engines: An Introduction*. Cambridge: Cambridge University Press, 1999. DOI: 10.1017/CBO9781139175135.

[17] A. Gupta, S. Sharma, and S. Narayan, *Combustion Engines : An Introduction to Their Design, Performance, and Selection*. Hoboken, New Jersey: John Wiley and Sons Inc., 2016.

[18] R. Thomas, M. Sreesankaran, J. Jaidi, D. M. Paul, and P. Manjunath, "Experimental evaluation of the effect of compression ratio on performance and emission of si engine fuelled with gasoline and n-butamol blend at different loads," *Perspectives in Science*, vol. 8, pp. 743 –746, 2016.

[19] M. Aldhaidhawi, R. Chiriac, V. Bădescu, H. Pop, V. Apostol, A. Dobrovicescu, M Prisecaru, A. A. Alfaryjat, M. Ghilvacs, and A. Alexandru, "Performance and emission of generator diesel engine using methyl esters of palm oil and diesel blends at different compression ratio," *IOP Conf. Series: Materials Science and Engineering*, vol. 147, 2016. DOI: 10.1088/1757-899X/147/1/012135.

[20] J. B. Heywood, *Internal Combustion Engine Fundamentals*. New York: Macmillan Press LTD, 1988.

[21] Y. Park, K. Min, J. Chung, and M. Sunwoo, "Control of the air system of a diesel engine using the intake oxygen concentration and the manifold absolute pressure with nitrogen oxide feedback," *Journal of Automobile Engineering*, 2016. DOI: 10.1177/0954407015584130.

[22] Z. Warhaft, *An introduction to thermal-fluid engineering: the engine and the atmosphere*. New York: Cambridge University Press, 1997.

[23] D. L. Greene and D. J. Santini, "Transportation and global climate change," *CDIAC Communications*, vol. 20, 2016. DOI: 10.1088/1757-899X/147/1/012135.

[24] H. Zhao, *Advanced direct injection combustion engine technologies and development*. Cambridge, UK: Woodhead Publishing Ltd, 2010. DOI: 10.1016/B978-1-84569-744-0.50020-6.

[25] C. L. control design for turbocharged diesel engines, *IEEE Transactions on Control Systems Technology*, vol. 8, no. 2, 2000. DOI: 10.1109/87.826800.

[26] M. Ale Mohammad and B. Huang, "Compensation of control valve stiction through controller tuning," *Journal of Process Control*, vol. 22, no. 9, pp. 1800 –1819, 2012, ISSN: 0959-1524. DOI: 10.1016/j.jprocont.2012.08.006.

[27] R. Bacci di Capaci, M. Vaccari, and G. Pannocchia", "Model predictive control design for multivariable processes in the presence of valve stiction," *Journal of Process Control*, vol. 71, pp. 25 –34, 2018, ISSN: 0959-1524. DOI: 10.1016/j.jprocont.2018.09.006.

[28] H. Ma, Z. Li, M. Tayarani, G. Lu, H. Xu, and X. Yao, "Computational intelligence nonmodel-based calibration approach for internal combustion engines," *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 4, p. 041 002, 2018. DOI: 10.1115/1.4037835.

[29] L. Guzzella and A. Amstutz, "Control of diesel engines," *Control Systems IEEE*, 18(5):53–71, 1998.

[30] A. C. Atkinson and A. N. Donev, *Optimum Experimental Designs*. Oxford: Clarendon Press, 2002.

[31] M. Castagné, Y. Bentolila, A. Halle, F. Nicolas, and D. Sinoquet, "Engine calibration: Towards an integrated approach," in *3rd Conference on Design of Experiments (DoE) in Engine Development*, Berlin, Germany: Expert-Verlag, 2007, pp. 56–65.

[32] Castagné, M., Bentolila, Y., Chaudoye, F., Hallé, A., Nicolas, F., and Sinoquet, D., "Comparison of engine calibration methods based on design of experiments (doe)," *Oil & Gas Science and Technology - Rev. IFP*, vol. 63, no. 4, pp. 563–582, 2008. DOI: 10.2516/ogst:2008029.

[33] O. Nelles, *Nonlinear System Identification From Classical Approaches to Neural Networks and Fuzzy Models*. Berlin, Germany: Springer-Verlag, 2001.

[34] M. Knaak, U. Schoop, and B. Barzantny, "Dynamic modelling and optimisation - the natural extension to classical DoE," in *Design of experiments in engine development III*, K. Röpke, Ed., Renningen: expert-Verlag, 2007, pp. 10–21.

[35] K. Fang, Z. Li, K. Ostrowski, A. T. Shenton, P. G. Dowell, and R. M. Sykes, "Optimal-behavior-based dynamic calibration of the automotive diesel engine," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 979–991, 2016, ID: 1.

[36] K. Fang, "Optimal test signal design and estimation for dynamic powertrain calibration and control," PhD thesis, University of Liverpool, 2012.

[37] K. Ostrowski, "Optimal dynamic calibration methods for powertrain controllers," PhD thesis, University of Liverpool, 2015.

[38] W. Gu, D. Zhao, and B. Mason, "Real-time modelling and parallel optimisation of a gasoline direct injection engine," in *2019 American Control Conference (ACC)*, Jul. 2019, pp. 5544–5549.

[39] S. D. Le Corre, B. Mason, T. Steffen, E. Winward, Z. Yang, T. Childs, M. Cary, and R. Lygoe, "Real-time modelling and parallel optimisation of a gasoline direct injection engine," in *WCX SAE World Congress Experience*, 2019.

[40] R. Omran, R. Younes, and J. C. Champoussin, "Optimal control of a variable geometry turbocharged diesel engine using neural networks: Applications on the etc test cycle," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 2, pp. 380–393, 2009, ID: 1.

[41] M. Neve, G. D. Nicolao, G. Prodi, and C. Siviero, "Estimation of engine maps: A regularized basis-function networks approach," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 3, pp. 716–722, 2009, ID: 1.

[42] J. D. Martínez-Morales, E. R. Palacios-Hernández, and G. A. Velázquez-Carrillo, "Modeling engine fuel consumption and nox with rbf neural network and mopso algorithm," *International Journal of Automotive Technology*, vol. 16, no. 6, pp. 1041–1049, 2015. DOI: 10.1007/s12239-015-0106-2.

[43] G. Alcan, E. Yilmaz, M. Unel, V. Aran, M. Yilmaz, C. Gurel, and K. Koprubasi, "Estimating soot emission in diesel engines using gated recurrent unit networks," *9th IFAC International Symposium on Advances in Automotive Control*, Jun. 2019.

[44] Y.-H. Pu, J. K. Reddy, and S. Samuel, "Machine learning for nano-scale particulate matter distribution from gasoline direct injection engine," *Applied Thermal Engineering*, vol. 125, pp. 336 –345, 2017. DOI: https://doi.org/10.1016/j.applthermaleng.2017.07.021.

[45] F. de Nola, G. Giardiello, A. Gimelli, A. Molteni, M. Muccillo, and R. Picariello, "Volumetric efficiency estimation based on neural networks to reduce the experimental effort in engine base calibration," *Fuel*, vol. 244, pp. 31 –39, 2019. DOI: https://doi.org/10.1016/j.fuel.2019.01.182.

[46] R. F. Turkson, F. Yan, M. K. A. Ali, and J. Hu, "Artificial neural network applications in the calibration of spark-ignition engines: An overview," *Engineering Science and Technology*, vol. 19, no. 3, pp. 1346 –1359, 2016. DOI: https://doi.org/10.1016/j.jestch.2016.03.003.

[47] K. Nikzadfar and A. H. Shamekhi, "Investigating a new model-based calibration procedure for optimizing the emissions and performance of a turbocharged diesel engine," *Fuel*, vol. 242, pp. 455 –469, 2019. DOI: https://doi.org/10.1016/j.fuel.2019.01.072.

[48] A. D. Sediako, J. Andric, J. Sjoblom, and E. Faghani, "Heavy duty diesel engine modeling with layered artificial neural network structures," in *SAE Technical Paper*, SAE International, Apr. 2018. DOI: 10.4271/2018-01-0870. [Online]. Available: https://doi.org/10.4271/2018-01-0870.

[49] G. F. Franklin, J. Powell, and M. L. Workman, *Digital control of dynamic systems*, 2nd ed. Reading, MA: Addison-Wesley, 1980, p. 837.

[50] H. Hiroyasui, 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 the JSME*, vol. 26, no. 214, pp. 569–575, 1983. [Online]. Available: http://dx.doi.org/10.1299/jsme1958.26.569.

[51] H. Sequenz, "Emission modelling and model-based optimisation of the engine control," University of Darmstadt, 2013.

[52] J. P. Hespanha, *Linear systems theory*. Princeton University Press, 2009.

[53] J. Wahlström and L. Eriksson, "EGR-VGT control and tuning for pumping work minimization and emission control," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 4, pp. 993–1003, 2011.

[54] M. Jung and K. Glover, "Calibratable linear parameter-varying control of a turbocharged diesel engine," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 1, pp. 45–62, 2006. DOI: 10.1109/TCST.2005.860513.

[55] F. Tschanz, A. Amstutz, C. H. Onder, and L. Guzzella, "Control of diesel engines using NO_x-emission feedback," *International Journal of Engine Research*, 14(1):44–56, 2006.

[56] M. Catagne, Y. Bentolila, A. Halle, F. Nicolas, and D. Sinoquet, "Engine calibration: Towards a integrated approach," in *Design of experiments in engine development III*, K. Röpke, Ed., Renningen: expert-Verlag, 2007, pp. 56–75.

[57] L. Guzzella and C. H. Onder, *Introduction to Modeling and Control of Internal Combustion Engine Systems*, 2nd ed. Heidelberg, Berlin: Springer-Verlag Berlin Heidelberg, 2010.

[58] K. Zhou, J. C. Doyle, and K. Glover, *Robust and optimal control*. Upper Saddle River, New Jersey: Prentice Hall, 1996, vol. 40.

[59] X. Wei and L. del Re, "Gain scheduled H_∞ control for air path systems of diesel engines using LPV techniques," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 3, pp. 406–415, 2007.

[60] M. Hardt, J. W. Helton, and K. Kreutz-Delgado, "Numerical solution of nonlinear H_2 and H_∞ control problems with application to jet engine compressors," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 1, pp. 98–111, 2000, ID: 1.

[61] R. B. di Capaci, M. Vaccari, G. Pannocchia, and C. Scali, "Identification and estimation of valve stiction by the use of a smoothed model," *IFAC-PapersOnLine*, vol. 51, no. 18, pp. 684–689, 2018. [Online]. Available: http://dx.doi.org/10.1016/j.ifacol.2018.09.344.

[62] G. V. Stuart, "A system identification approach to the modelling of engine transients," *International Congress and Exposition*, 1990. DOI: 10.4271/900237.

[63] G. Zito and I. D. Landau, "NARMAX model identification of a variable geometry turbocharged diesel engine," *American Control Conference*, pp. 1021 –1026, 2005.

[64] T. Boz, M. Unel, V. Aran, M. Yilmaz, C. Gurel, C. Bayburtlu, and K. Koprubasi, "Diesel engine nox emission modeling with airpath input channels," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, Nov. 2015, pp. 003 382–003 387. DOI: 10.1109/IECON.2015.7392622.

[65] S. M. Winkler, M. Hirsch, M. Affenzeller, L. del Re, and S. Wagner, "Virtual sensors for emissions of a diesel engine produced by evolutionary system identification," in *Computer Aided Systems Theory - EUROCAST 2009*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 657–664, ISBN: 978-3-642-04772-5.

[66] J. Deng, M. Bastian, and R. K. Stobart, "Particulate matter prediction in both steady state and transient operation of diesel engines," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 226, no. 2, pp. 260–274, 2012. DOI: 10.1177/0954407011418029.

[67] MATLAB, *version 9.5.0.1033004 (R2018b) Update 2*. Natick, Massachusetts: The MathWorks Inc., 2018.

[68] J. Wahlström and L. Eriksson, "Modelling diesel engines with a variable-geometry turbocharger and exhaust gas recirculation by optimization of model parameters for capturing non-linear system dynamics," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 225, no. 7, pp. 960–986, 2011.

[69] F. Ibrahim, W. M.F. W. Mahmood, S. Abdullah, and M. R. A. Mansor, "Comparison of simple and detailed soot models in the study of soot formation in a compression ignition diesel engine," SAE Technical Paper, Tech. Rep., 2017. DOI: 10.4271/2017-01-1006.

[70] R. Sindhu, G. A. P. Rao, and K. M. Murthy, "Real-time single zone model for simulation of a diesel engine in Simulink environment," *International Conference on Computing, Communication and Automation (ICCCA2015)*, pp. 806–811, 2015.

[71] T. Lee, H. White, and C. W. J. Granger, "Testing for neglected nonlinearity in time series models," *Journal of econometrics*, vol. 56, pp. 269–290, 1993.

[72] T. Knudsen, "Test for nonlinear input output relations in siso systems by preliminary data analysis," 2000. [Online]. Available: http://vbn.aau.dk/files/169526/fulltext/.

[73] L. Ljung, *System Identification: Theory for the User*, ser. Prentice Hall information and system sciences series. Prentice Hall PTR, 1999, ISBN: 9780136566953.

[74] R. D. Nowak, "Nonlinear system identification," *Circuits, Systems and Signal Processing*, vol. 21, no. 1, pp. 109–122, 2002. DOI: 10.1007/BF01211655.

[75] Ai Hui Tan and K. R. Godfrey, "The generation of binary and near-binary pseudorandom signals: An overview," *IEEE Transactions on Instrumentation and Measurement*, vol. 51, no. 4, pp. 583–588, 2002. DOI: 10.1109/TIM.2002.802243.

[76] M. Deflorian and S. Zaglauer, "Design of experiments for nonlinear dynamic system identification," in *18th IFAC World Congress, August 28, 2011 - September 2*, BMW Group, vol. 18, Milano, Italy: IFAC Secretariat, 2011, pp. 13 179–13 184. [Online]. Available: http://dx.doi.org/10.3182/20110828-6-IT-1002.01502.

[77] K. Fang and A. T. Shenton, "Constrained optimal test signal design for improved prediction error," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 4, pp. 1191–1202, 2014. [Online]. Available: http://dx.doi.org/10.1109/TASE.2013.2264810.

[78] W. J. Conover, *Pratical nonparametric statistics*. John Wiley and Sons, 1999.

[79] E .D Adrian and Y. Zotterman, "The impulses produced by sensory nerve-endings: Part II," *Journal of Physiology*, vol. 61, no. 2, pp. 151–171, 1926. DOI: 10.1113/jphysiol.1926.sp002281.

[80] K. Lucas, "The "all or none" contraction of the amphibian skeletal muscle fibre," *Journal of Physiology*, vol. 38, no. 2-3, pp. 113–133, 1909. DOI: 10.1113/jphysiol.1909.sp001298.

[81] D. O. Hebb, *The organization of behaviour: A neuropsychological theory*. 2002.

[82] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943. DOI: https://doi.org/10.1007/BF02478259.

[83] K. Worden and G. R. Tomlinson, *Nonlinearity in Structural Dynamics*. Temple Way, Redcliffe: IOP Publishing Ltd, 2001.

[84] H. Gensler, *Introduction to Logic*, 1st ed. London: Routledge, 2002, pp. 41–42. DOI: 10.4324/9780203204887.

[85] F. Rosenblatt, *Principles of Neurodynamics Perceptrons and the Theory of Brain Mechanisms*. Washington D.C.: Spartan Books, 1962.

[86] K. Lange, *Optimization*. New York: Springer, 2013. DOI: 10.1007/978-1-4614-5838-8.

[87] F. Aragón, M. A. Goberna, M. A. López, and M. M. L. Rodríguez, *Nonlinear Optimization*. Cham, Switzerland: Springer, 2019. DOI: 10.1007/978-3-030-11184-7.

[88] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," PhD thesis, Harvard University, 1975.

[89] ——, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990. DOI: 10.1109/5.58337.

[90] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963. DOI: 10.1137/0111030.

[91] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994. DOI: 10.1109/72.329697.

[92] T. S. Rao and M. M. Gabr, "A test for linearity of stationary time series," *Journal of time series analysis*, vol. 1, pp. 145–158, 1980.

[93] S. H. Strogatz, *Nonlinear dynamics and chaos, with applications to physics, biology, chemistry and engineering*. Birkhauser Verlag, 2001.

[94] H. Bhuiyan and J. Lee, "Low cost position controller for exhaust gas recirculation valve system," *Energies*, vol. 11, no. 8, 2018, ISSN: 1996-1073. DOI: 10.3390/en11082171. [Online]. Available: https://www.mdpi.com/1996-1073/11/8/2171.

[95] T. J. Barlow, S. Latham, I. S. McCrae, and P. G. Boulter, *A reference book of driving cycles for use in the measurement of road vehicle emissions*. TRL Limited, 2009.

[96] C. D. Rakopoulos and E. G. Giakoumis, *Diesel Engine Transient Operation*, 1st ed. Holborn, London: Springer-Verlag London Limited, 2009.

[97] W. contributors, *Mixed/dual cycle — Wikipedia, the free encyclopedia*, [Online; accessed 23-July-2019], 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mixed/dual_cycle&oldid=875512187.

[98] J. Nagle and R. F. Strickland-Constable, "Oxidation of carbon between 1000-2000deg C," *Proceedings of the 5th Conference on Carbon*, pp. 154–164, 1962.

[99] H. Hiroyasu and M. Arai, "Development and use of a spray combustion modeling to predict diesel engine efficiency and pollutant emissions," *Minutes of the Meeting - Pennsylvania Electric Association, Engineering Section*, vol. 1, pp. 264–288, 1980, Compilation and indexing terms, Copyright 2019 Elsevier Inc.

[100] T. Ng, G. Goodwin, and R. Payne, "Optimal input design for an AR model with output constraints," *Automatica*, vol. 20, pp. 359–363, 1977.

[101] R. M. Lewis and V. Torczon, "A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds," Institute for Computer Applications in Science and Engineering, Tech. Rep., 1998.

[102] T. G. Kolda, R. M. Lewis, and V. Torczon, "A generating set direct search augmented lagrangian algorithm for optimization with a combination of general and linear constraints," Sandia National Laboratories, Tech. Rep., 2006.