UNIVERSITY OF LIVERPOOL

Model Structure Selection in Powertrain Calibration and Control

Thesis submitted in accordance with the requirements of the
University *of* Liverpool for the
Degree of Doctor in Philosophy

by

Zongyan Li

October 2012

Statement of Originality

This thesis is submitted for the degree of Doctor in Philosophy in the Faculty of Engineering at the University of Liverpool. The research project reported herein was carried out, unless otherwise stated, by the author in the Department of Engineering at the University of Liverpool between 1/11/2008 and 31/10/2012.

No part of this thesis has been submitted in support of an application for a degree or qualification of this or any other University or educational establishment. However, some parts of this thesis have been published, or submitted for publication, in the following papers:

- Li Z. and Shenton A.T. (2010) Nonlinear Model Structure Identification of Engine Torque and Air/Fuel Ratio. 6th IFAC Symposium Advances in Automotive Control. IFAC, Munich pp 1-6
- Li Z. and Shenton A.T. (2010) Structure Selection for Identified Multi-Model Dynamic Fuel Maps. 10th International Symposium on Advanced Vehicle Control. AVEC, Loughborough pp 1-6
- Shenton A.T. and Li Z. (2012) Multi-Modelling of Torque and Air/Fuel Ratio Based on Engine Operating Regions. 15th IFAC Workshop on Control Applications of Optimization, Volume.15, Part 1. IFAC, Rimini
- Li Z. and Shenton A.T. (2012) I.C. Engine Inverse Dynamic Multi-Model Identification. In Proceeding of Powertrain Modelling and Control Conference, Paper 16, Bradford

Zongyan Li
20th October 2012

Abstract

This thesis develops and investigates the application of novel identification and structure identification techniques for I.C. engine systems. The legislated demand for reduced vehicle fuel consumption and emissions indicates that improved model-based dynamical engine calibration and control methods are required in place of the existing static set-point based mapping methods currently used in industry. The choice of structure of any dynamical engine model has significant consequences for the accuracy and the calibration/optimization time of engine systems. This thesis primarily addresses the issue of this structure selection.

Linear models are well understood and relatively easy to implement however the modern I.C. engine is a highly nonlinear system which restricts the use of linear structures. Further the newer technologies required to achieve demanding fuel consumption and emission targets are increasingly more complex and nonlinear. The selection of appropriate nonlinear model regressor terms presents a combinatorial explosion problem which must be solved for accurate engine system modelling. In this thesis, two systematic nonlinear model structure selection techniques, namely stepwise regression with F-statistics and orthogonal least squares method with error reduction ratio, are accordingly investigated. SISO algebraic NARMAX engine models are then established in simulation studies with these methods and demonstrate the effectiveness of the approach.

The thesis also investigates the development and application of multi-modelling techniques and the expansion of the model structure selection techniques to the identification of the local models terms within the multi-model structures for the engine. Based on the engine operating regions, novel multi-model networks can be established and several alternative multi-modelling techniques, such as LOLIMOT, Neural Network, Gaussian and log-sigmoid function weighted multi-models, for the multi-model engine system identification are explored and compared. An experimental validation of the methods is given by a black box identifi-

cation of SISO engine models which are developed purely from the experimental engine test data sets. The results demonstrate that the multi-model structure selection techniques can be successfully applied on the engine systems, and that the multi-modelling techniques give good model accuracy and that good modelling efficiency can also be achieved.

The outcome is a set of techniques for the efficient development of accurate nonlinear black-box models which can be acquired from experimental dynamometer test-bed data which should assist in the dynamic control of future advanced technology engine systems.

Acknowledgments

I am very grateful to my supervisor Dr Tom Shenton. His guidance and support were indispensable and much appreciated in the completion of this work. I also want to thank my supervisor Prof. Huajiang Ouyang for his timely and valuable advice while I was progressing the PhD. I would like to thank Dr Ahmed Abass, Dr Paul Dickinson, Ke Fang, Ming-yen Chen and Kamil for kindly offering their valuable suggestions and assistance. I would like to thank my parents for their inspiration and selfless financial support for my study in the UK. Finally, my gratitude goes to my wife, Qin Chen, for her love and patience.

Contents

List of Figures

List of Tables

Acronyms

ABV	Air-Bleed Valve
ACF	Auto-Correlation Function
AFR	Air-Fuel Ratio
AIC	Akaikes Information Criterion
PRBS	Modulated Pseudo-Random Binary Sequence
ATDC	After Top Dead Centre
ARMAX	AutoRegressive Moving Average with eXogeneous inputs
ARX	AutoRegressive with eXogeneous inputs
BDC	Bottom Dead Centre
BIC	Bayesian Information Criterion
BJ	Box-Jenkins
BTDC	Before Top Dead Centre
DoE	Design of Experiment
ECU	Engine Control Unit
EGR	Exhaust Gas Recirculation
EMS	Engine Management System
FPE	Final Prediction Error
FPW	Fuel Pulse Width
FIR	Finite Impulse Response
GDI	Gasoline Direct Injection
GTDI	Gasoline Turbocharged Direct Injection
HEGO	Heated Exhaust Gas Oxygen
IC	Internal Combustion
INJ	Injected fuel mass
MAP	Manifold Air Pressure
MIMO	Multiple-Input-Multiple-Output
MISO	Multiple-Input-Single-Output
MLE	Maximum Likelihood Method
MSE	Mean Squared Error
NARX	Non-linear AutoRegressive Moving Average
NN	Neural Network

OE	Output Error
OLS	Ordinary Least Square
PEM	Prediction Error Method
PFI	Port Fuel Injection
PRBS	Pseudo Random Binary Sequence
PS	Pattern Search
PSD	Power Spectral Density
RBS	Random Binary Signal
RPM	Revolutions Per Minute
RT	Real Time
SA	Spark Advance
SAN	Simulated ANnealling
SEM	Simulation Error Method
SI	Spark Ignition
SISO	Single-Input-Single-Output
SQP	Sequential Quadratic Programming
TDC	Top Dead Centre
THR	Throttle Angle
TRR	Trust Region Reflective
TWC	Three Way Catalyst
UDRN	Uniformly Distributed Random Number
UEGO	Universal Exhaust Gas Oxygen
VGT	Variable Geometry Turbocharger

Chapter 1

Introduction

The modern automotive I.C. engine system requires accurate and multi-dimensional calibration in order to meet increasingly strict emission and fuel consumption regulations. The trend of engine development is to the use of additional actuators, sensors and of more complex technologies with more prevalent transient dynamics. As a result, the number adjustable variables in the engine operating space are increasing significantly which means that even more data samples clustered across the larger operating space are needed to determine sufficient information for statistically reliable engine calibration. Because the number of required data points and corresponding optimisation effort increases exponentially with the number of adjustable variables, this phenomena is known as 'curse of dimensionality' [1]. Notwithstanding the significant improvements brought by advanced numerical optimisation and design-of-experiment (DoE) methods, due to the 'curse of dimensionality' the cost of experimental based engine tests to support the conventional static calibration procedures is becoming exponentially time-consuming and costly. One potential way of addressing this issue is to capture experimental data in a dynamic model during non-steady-state testing [2]. An additional advantage of a dynamic modelling is that it may allow control of the engine dynamics which are likely to become of increasing significance in forthcoming legislated drive cycles which have significant engine transients and tightened emissions [3].

To capture the nonlinear dynamical behaviour of the advanced I.C. engine system how-

ever requires an efficient and reliable way to determine model type and subsequently model parameters in the system-identification. Dynamic black-box identified polynomial models have been widely considered by researchers for engine management system (EMS) mappings instead of the current steady-state static pseudo-dynamic methods [4]. Hybrid structured multi-models have been reported as being able to provide a capability to represent complex nonlinear characteristics with the desired simple structure, however there are many choices for the multi-model structure, including nonlinear terms, partition size, weighting choice, linear vs nonlinear etc. [5]. Such dynamic maps have the potential to improve emissions and fuel consumption in transient operations for reduced computational time and ease of implementation including comprehension and calibration. Accordingly, structure selection methods are required to choose the simplest effective models. Structure identification techniques to determine the optimal structures of black-box models are the principal subject of this thesis.

1.1 Systems identification of SISO and MIMO models

The study of Single-Input-Single-Output (SISO) systems offer a way to reveal the cause and effect interconnection between two variables and it is the central concern of system identification. For a newly designed system or a manufactured item, we would always ask the question like how will all the inputs affect the system outputs or what uncertainties the system has. SISO models can be used to relate variables of any component in the engine whether it is a small part such as a spark plug or a larger mechanism namely a turbocharger. The SISO models are generally identified based on input/output data and physical information gathered from the system [6]. Three significant areas are required to be considered before the identification:

- Prior knowledge about physical phenomena

  The behavior of the systems is guided by a series of physical principles such as Newton's laws and energy conservative laws, therefore, some outputs of the system are fully predictable according to the physical analysis. For example, in an engine intake system, the air flow rate can be derived from the throttle opening angle and intake manifold area

of the cross section. Engine speed can be derived from the brake torque and crankshaft inertia.

- Observed data

  The observed data is the key information source for a complex system because physical analysis becomes unfeasible due to the curse of dimensionality. The input signal design is a significant procedure to make sure the information contained in the data is rich enough. Meanwhile, the measured data is used to compare with the model output and thus validate the model quality. The observed data can be measured directly by the sensors in the system or indirectly calculated by measured data.

- Accuracy requirement of the model

  Most of the model structure determination process runs iteratively thus a stopping criteria is necessary for the identification process. This means the identified models are only acceptable when certain thresholds are satisfied. The model with high accuracy is significant but complex model structures may cost a long time in training the data and calculating the output. A balance must be found otherwise it would make the model difficult to be implemented in the real-time control systems.

In this thesis, SISO identification and structure selection techniques and also Multiple-Input-Single-Output (MISO) system identification are investigated. In a complex engine system, the output is always affected by various inputs and their cross relations. The proposed strategy for a MIMO identification problem is to convert the Multiple-Input-Multiple-Output (MIMO) identification problem into a network of MISO models which can be solved separately when the outputs of the system are relatively independent of each other. With proper data acquisition interface, such as dSPACE A/D conversion module, the data can be captured and processed when the target system is in operation. In this case, the time-domain MIMO models can also be identified in real-time by recursive least-squares estimation [7].

1.2 Overview

This thesis investigates various model structure selection techniques. Novel SISO and MISO dynamical models have been developed by using these techniques. The models acquired were

identified and validated on both a real and virtual engine setups.

Following this introduction, Chapter 2 introduces the general procedures of system identification. Both the static and dynamic system properties are discussed. The white-box, black-box and grey-box model categories are described, especially the linear and nonlinear black-box models. The model-based engine calibration is discussed as well.

Chapter 3 shows the infrastructures of four different engine test beds used in studies in this thesis and introduces the sensors and actuators mounted on them.

Chapter 4 gives a detailed study of the application of Least Squares algorithm in determining the parameters of dynamical polynomial models. A series of dynamical SISO engine models including linear and nonlinear ABV-RPM model are established and validated.

Chapter 5 demonstrates two major model selection techniques: the stepwise regression with F-statistic analysis and the orthogonal least squares with error reduction ratio analysis. The ANOVA algorithm is also introduced in this chapter. The dynamic models constructed with these techniques are currently limited to linear, square and cubic regressors terms but can be readily extended if necessary. A novel MISO dynamical model is developed by application of the model structure selection techniques.

Chapter 6 extends the scope of model structure selection to the multi-model identification. A number of multi-model structures are investigated, namely the LOLIMOT, and the Neural Network. The general procedure of multi-modelling is described in this chapter. Several novel dynamical multi-models are developed for different types of engines.

Chapter 7 applies an multi-model identification process to produce a novel forward and inverse multi-model. The WAVE-RT virtual engine is adopted as the experiment test bed. The forward model gives accurate estimation of the engine torque and air/fuel ratio whilst the inverse model is regarded as a controller by tracking the desired engine output with inverse

identified inputs.

Chapter 8 reviews the main outcomes of this thesis and discusses the thesis contributions together with the future possible developments of these efforts.

1.3 Contributions

The novel contributions of this thesis are as follows:

- SISO dynamical models were developed based on PFI gasoline engine in Chapter 4. The model structure are found by Matlab identification toolbox and validated by the measured data. The polynomial model structure was adopted, especially, the least-squares technique is adopted to determine the parameters of the models. Finally, both ARX and NARX models of the engine system has been obtained.
- MISO dynamical models for IC engine was developed using stepwise regression and orthogonal least squares techniques in Chapter 5. At the same time the detailed steps of the iterative structure selection procedure are presented.
- A LOLIMOT model and a log-sigmoid weighted multi-model was established for IC engines. The optimal number of local models is analysed according the multi-modelling results.
- An inverse multi-model has been identified on a state-of-the art virtual engine model. The inverse model has been implemented to track the engine outputs of torque and air/fuel ratio.

Chapter 2

Background

2.1 Dynamical System Identification for IC Engine

2.1.1 Static and Dynamical Modeling

A model for a system describes part or all of its characteristics depending on its application. If we define two time points t_1 and t_2 with $u(t_1) = u(t_2)$ as their corresponding inputs, for the static model, the model outputs would turn out to be $y(t_1) = y(t_2)$. The relationship between inputs and outputs remained unchanged with time in a static system. However, in a dynamical system, the relationship is constantly evolving with time. In other words, the outputs of static system only depend on the current inputs, but the outputs of dynamical system will be affected by both current and previous inputs along the time line. In detail, some modeling and control strategies for dynamic systems has been researched in [8, 9, 10].

2.1.2 Dynamic Engine Identification

System identification is a fundamental problem in systems engineering applications. The application of system identification overlaps the boundaries of engineering and physical science. Many other fields of study, such as economics and medicine, can also benefit from the employment of system identification technology. Generally, system identification is the de-

termination of a mathematical model for a system or a process by observing its input-output relationships [11]. In another word, a specific mathematical model which is equivalent in input-output behaviour to a physical system can be obtained by this technology.

As the foundation of system identification, the input and output data should be collected in a pre-designed experiment. Initially, the engine system is monitored by a set of sensors and actuators according to our a priori knowledge, then a mathematical model for the engine is developed by using structure selection and parameter estimation algorithms. The engine model should be capable of reproducing the dynamics of the system over different operating regimes. An accurate identified engine model is significantly beneficial for engine calibration and control purposes. A schematic flow chart of the identification process is given in Figure. 2.1 [12, 13, 14]

Design of Experiment

Dynamic system behaviour is represented by the current and past values of inputs and outputs especially the transient response between the inputs and outputs. The experiment design process applies clustering algorithm to locate a set of feasible operating points in order to established dynamical model on a pre-defined working range with satisfying accuracy [15, 16].

To start an identification and control task, the input and output signals are to be determined at the beginning. In the process of designing the experiment, a strategy of how to efficiently and accurately acquire the information from the available experimental setup should be established. In the case of engine test, the input and output signals are typically transmitted into an A/D converter and processed in the computer. This means the resolution and the sampling time of the sensor and actuator mounted on the engine must be considered in order to make sure the I/O data can be collected and managed in a sensible way. Some prior knowledge of the system is vital because any experimental system has its operating limitations and experiment should always be conducted within the safe operating range. Furthermore, the input and output signals need to be measurable within the experimental budget. It is wise to avoid some parameters which are difficult to measure by indirectly measuring related parameters. For instance, the throttle valve duty cycle is replaced by the Air Bleed Valve (ABV) duty as the input signal and the Fuel Pulse Width (FPW) is measured

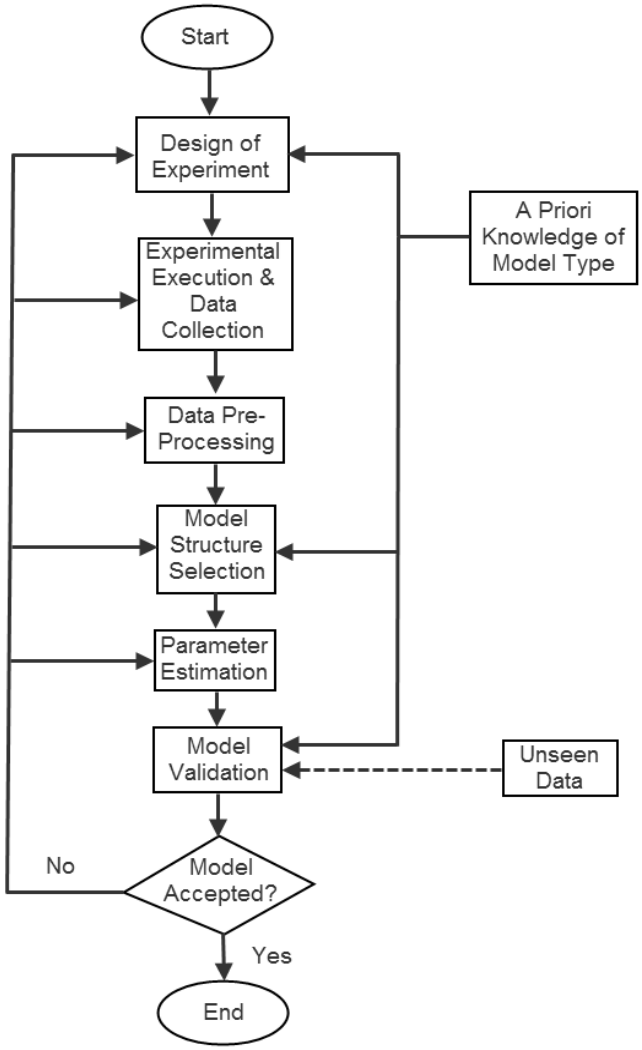

Figure 2.1: The Flow Chart of System Identification

instead of fuel injection mass. There are three major concerns in the design of the input signals to be used for identification:

- Frequency Spectrum and Perturbation Time

  The bandwidth of a dynamic system should be much wider than that of a static system. Therefore, a proper input perturbation signal should be capable of exciting the system across its bandwidth and revealing its nonlinearity as accurately as possible. In many cases, step-shaped input signals, which change abruptly from one input value to another, are preferred because they are equivalent to the superpositions of sinusoidal components across an infinite number of frequencies. However, it is necessary to be aware that instant change of the magnitude of input signal may result in damage to the test equipment. The Pseudo Random Binary Sequence (PRBS) is another preferred option in a digitally controlled experiment. The PRBS signal has the advantage of a wide range of frequency content and a flat power spectrum density which is suitable for dynamic identification. Other than the bandwidth of the PRBS signal which covers a large spectrum between its lower frequency and upper frequency, the perturbation time is also a deciding matter. The identified model of a dynamic system can only be recognized as a good representation when the perturbation time is longer than the delays of the system.

- Amplitude of the Signal

  The amplitude of the signal does not pose any significant threat to the model accuracy in linear identification applications because the gain of the system's transfer function is constant, in which case the output signal is proportional to the input. Therefore, the PRBS signal with its two distinct levels is well accepted. For largely nonlinear systems, the random walk signals which generated by uniformed random PRBS, a gain and an integrator is believed to be a good choice. The variation of the signal amplitude of the random walk signal aims to cover all the operating range of the nonlinear system and capture the uncertainty within the nonlinear system.

- Sampling rate

  The sampling rate refers to the time interval between the current and the subsequent record of the signal generated by the sensors. The main constraint on achieving an appropriate sampling rate is the availability of qualified instruments which are able to

achieve the speed of sampling required. It has been recommended [14] that the sampling rate to be 10 times that of the interested variable's bandwidth. For best results, the highest sampling rate is usually adopted in any engine test and the researcher will then be able to decide the down-sampling strategy depending on the modeling requirements. It is worth noting that the down-sampling can only be applied after any sort of digital anti-aliasing filtering [17]. At the same time, the test input signal designer should also keep the input signal rate lower than that of the sampling rate.

Data Processing

Due to various requirements of an identification algorithm, the experimental data collected from an engine test should be adjusted into suitable form. The oversampling of the recorded data, the associated sensors noise and the switching of electronic noise should all be considered as factors in the data processing.

1. Offsets Removal and Data Detrending

   In a general engine test system, the sensors for physical measurement could lead to bias or offset. To remove the offsets such as those existing in absolute pressure or temperature measurement from the data greatly improves the quality of the identification process [18]. Therefore, the mean value of the data is eliminated from the raw data samples respectively. Alternatively, the offset term can be deducted from the I/O signal in the identification process. Offsets can be revealed in the raw data or detected in general trends like linear drifts or seasonal trends, and thus the best fitted trend is subtracted from the I/O data in order to cancel these trends [17]. Usually, the sampling rate is taken as higher than necessary in order to improve the optimizing result with the amount of increase in rate based on the specific accuracy requirements. Since a system with high order dynamics can have both fast and slow modes such a system may need to be measured at high rate over a longer time requiring many more data samples than for a system with low order dynamics. Further a high order system may require a much faster sampling rate to capture its transient nonlinear behaviour, whereas a lower order model containing a similar nonlinearity can be obtained with a much reduced sampling rate.

2. Elimination of Outlier Points

   Outlier points are the non-normal data which may be caused by localized anomalous events or measurement errors involved in the system. These data points will break the regulation of the data and dramatically affect the accuracy of the results. Hence the outlier points ought to be eliminated before the identification. Due to the severe cost to the identification process, it is better to avoid it in the experiment design stage. Otherwise, the outlier points can be recognized as lost data which can be deleted from the raw data.

3. Pre-filtering

   A frequency domain filter is designed to modify the frequency response of a system, normally by either emphasizing or attenuating certain frequency ranges. A significant regulation has been found that, for the linear system, filtering both the input and output data by the identical filter does not impact the relationship between input and output signals [14]. Hence the filter can be used while collecting the original data and the type of filter is decided by the application region and the sort of the disturbance. For the data sampling applications, an anti-aliasing filter should be utilized, with a separate frequency in the middle position of the sampling frequency which named as Nyquist frequency. And then the ambiguity caused by aliasing of high frequency component at low frequency can be eliminated. However, if the noise and disturbances is in the region of well defined frequency, then the band-stop filter can be applied for identifying the attenuation of specific frequency bands. Moreover, a band-pass filter can be adopted while identifying the dynamics in a particular frequency range. Since the influence of unknown external noise sources or disturbances is always existent in the data used for identification. It can be concluded that the choice of filtering strategy is significant factor which concerns whether the satisfactory results can be obtained for system identification.

2.1.3 White-box Models

A white-box model is transparent from outside viewers. It usually integrates key physical elements and forges a clear mathematical representation that shows the transit states of

the dynamic system. In order to develop the model, various scientific laws can be used, including thermodynamic laws, Kirchhov's laws, Newton's laws and reaction kinetics [19]. An illustrative physical modelling which described a powertrain system was presented in [20]. An engine in-cylinder model based on the principles of thermodynamics are introduced in [21] and another kinematical model is established in [22]. Therefore, the white-box model is also recognized as phenomenological or physical model. With the confirmation of the physical parameters in the model, the engine system dynamics can be fully understood. However, in a complex system like a powertrain engine, the white-box models lacks the ability to make approximations in order to simplify the identification process thus difficult to be implemented in digital device such as Engine Control Unit (ECU) .

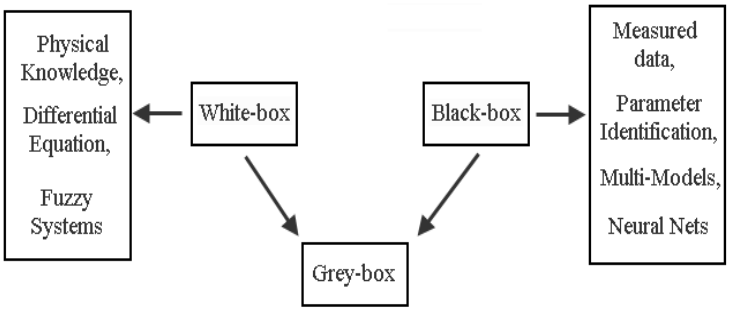

Figure 2.2: Black-box and White-box Models comparison

2.1.4 Black-box Models

The black-box approach of modelling relies entirely on the input/output data from the test bed. In this sense, system identification can be seen as a behavioural approach for model development. [13]. Since the system's physical property is assumed to be unknown, the model structures are mainly constructed by weighted combinations of basis functions or time delayed regressors obtained from the real system's I/O channels. Black-box models can be identified as either parametric or non-parametric models. Parametric models are developed by determining the parameters in relevant transfer functions or state-space matrices, while non-parametric models are focused on estimating a model in frequency domain, such as impulse responses and frequency responses. A predefined quality criterion, such as goodness of fit or normalized mean square error (NMSE), has to be met in order to accept the black-

box model because it is not interpretable in the physical sense. The trial and error tests are needed as stated in [23] because the black-box models are only valid over the operating range where the models are identified and could become unreliable elsewhere.

Linear black-box models

As demonstrated in [24], the simplest dynamical black-box model is called the Finite Impulse Response (FIR) model. In Equation 2.1, the FIR model is represented as:

$$\begin{aligned} y(t) &= B(q)u(t) + e(t) \\ &= b_1 u(t-1) + \cdots + b_n u(t-n) + e(t) \end{aligned} \tag{2.1}$$

where q is the shift operator and $B(q)$ is a polynomial in q^{-1}. Hence the predictor $\hat{y}(t|\theta) = B(q)u(t)$ of this FIR model are only formed by regressors of input delays. The regressor vector is then defined as

$$\phi(t) = [u(t-1), u(t-2), \cdots, u(t-n)] \tag{2.2}$$

As we involve more delayed terms in the regressor vector, the model is able to reflect more system dynamics. However, in practice, the regressors in the vector should be selected and the noise term e(t) are assumed to be Gaussian distributed or to be modelled in different ways.

The general structure of black-box model family is proposed in [14] as:

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \tag{2.3}$$

Specially, various classes of models can be derived from Equation 2.3. As shown in Figure 2.3, the linear black-models are categorised as Prediction Error Method (PEM) model as shown in Equation 2.3, the Box-Jenkins (BJ) model when $A = 1$, the ARX model when $F = C = D =$ 1,the ARMAX model when $F = D = 1$, the Output-Error (OE) model when $A = C = D = 1$.

The predictor of the model output can be defined as the regression form in [25]:

$$\hat{y}(t|\theta) = \theta^T \phi(t, \theta) \tag{2.4}$$

where the parameter vector $\theta = [\theta_1, \theta_2, \theta_3, ...]$.

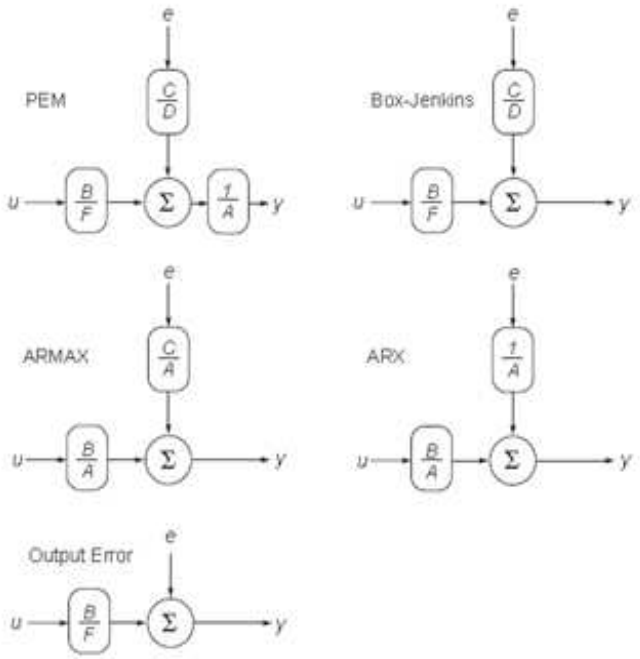

Figure 2.3: Linear black-box model classes

Nonlinear black-box models

The nonlinear black-box model estimator is able to be described as a nonlinear function $g(\cdot)$ of regressor vector $\phi(t)$ and corresponding parameter vector θ as

$$\hat{y}(t|\theta) = g(\phi(t), \theta) \tag{2.5}$$

In the case of discrete system, the time shift operator q is transformed into z and the output dynamics of the system is revealed by difference dynamics or tapped delays [6]. Figure 2.4 demonstrates the dynamical model with tapped delays. The past inputs and outputs of the system can be restored by these delay taps and recycled into the dynamical model when being called. When the outputs of real system or measured outputs $y(t), y(t -$

$1), y(t-2)...y(t-n)$ are used in the feedback, the dynamical model is named as NARX (Nonlinear AutoRegressive with eXogenous input) model. On the other hand, if the model outputs $\hat{y}(t), \hat{y}(t-1), ..., \hat{(}y-n)$ are adopted as the feedback, the model should be categorised as the NOE (Nonlinear Output Error) Model. From the prospect of model identification, it is likely that NOE mode will offer better simulation results than that from an ARX or NARX model which are identified using the measured data.

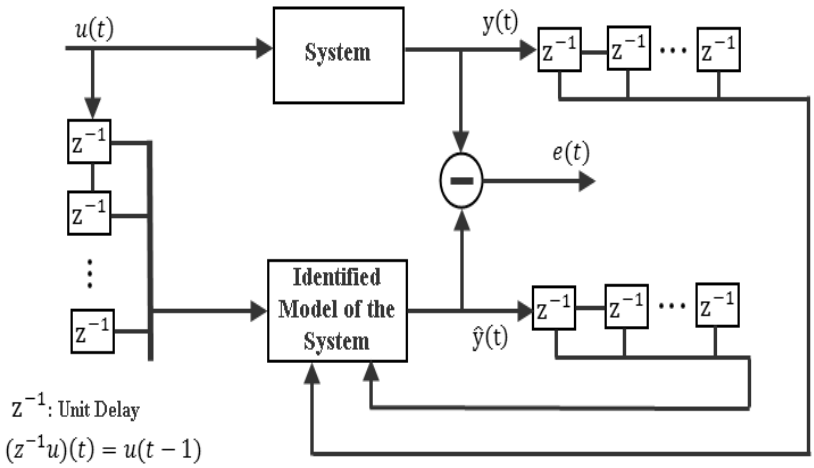

Figure 2.4: The identified dynamical model with a series of unit tapped delays

Here we define input and output regressor as $u(t-n_u)$,$y(t-n_y)$, the model predicted output as $\hat{y}(t-n_y|\theta)$. Then the simulated output $\hat{y}_u(t-n_y|\theta)$ can be defined as the model output $\hat{y}_u(t|\theta)$ at time t with all measured outputs $y(t-n_y)$ replaced by the simulated output $\hat{y}_u(t-n_y)|\theta)$ computed k-step ahead. Therefore, the model prediction error becomes $e(t-n_e) = y(t-n_y) - \hat{y}(t-n_y|\theta)$ and the simulation error is $e_u(t-n_e) = y(t-k) - \hat{y}_u(t-n_y|\theta)$.

Based on these expressions, the corresponding nonlinear version of the model types are defined as [26, 27]:

- NFIR models which only includes input regressors $u(t-n_u)$
- NARX models which contain both input and output regressors $u(t-n_u)$ and $y(t-n_y)$
- NARMARX models which consider input $u(t-n_u)$, output $y(t-n_y)$ and prediction errors $e(t-n_e|\theta)$

- NOE models which use $u(t-n_u)$ and $\hat{y}(t-n_y|\theta)$ as regressors when the simulated model output is calculated iteratively as $\hat{y}_u(t)|\theta)$
- NBJ models which adopt $u(t-n_u)$, $y(t-n_y)$, $e(t-n_e|\theta)$ and $e_u(t-n_e|\theta)$ as regressors. In this model class, the simulated output $\hat{y}_u$ is determined by using the structure of Equation 2.5 at the same time substituting e and e_u by zero vectors in the regression vector $\phi(t,\theta)$.

2.1.5 Grey-box Models

A comparison between Black-box and White-box models is shown in Fig. 2.2. The white-box model's high interpretability is most advantageous, but it requires exact knowledge of the system. On the other hand, the black-box model is capable of fast simulation of the system but not interpretable in physical sense. In reality, model identification lies in somewhere between the two ends. These model classes can be called the Grey-box models which offers higher flexibility in system identification process. When we has some a priori physical knowledge about the system but are not able to solve the parameters associated with it, the data-based black-box approach can be applied. In terms of interpretability, the physical implication of the model parameters can be retained and assessed by the designer of the control system. A series of grey-box models, namely neuro-fuzzy systems and semi-physical models can be found in [28, 29].

2.1.6 Dynamic Engine Calibration

Engine calibration aims to develop the optimal engine operating scheme under varies restrains, such as: emission legislation, fuel consumption and requests from the driver [30]. The best trade-off are required to be found because these requirements contradict in the engine mechanism. Some of the extensive applications of engine dynamical calibration are documented in [16]. The engine systems is conventionally calibrated by static mapping techniques which are based on multi-dimensional tables describing the relationship among the engine inputs and outputs. With more advanced technology being applied to the engine system, the relevant variables controllable in the engine management unit (EMS) increases dramatically.

Consequently, the calibration time of the engine will be rising enormously due to the curse of dimensionality. A lot of researchers suggest that the dynamic models are capable to overcome the shortcomings of the static calibration, mainly the time-consuming experiment [31, 32].

The model-based calibration is introduced to meet the challenge of more advanced engine calibration. The engine properties is represented by mathematical models instead of look-up tables. The engine control scheme is then developed based on the engine mapping generated by engine model testing. Consequently, it is essential for the engine model to be accurate enough to predict the engine dynamic behaviour. Various types of engine models were investigated extensively by researchers. The examples of physical models of the air, fuel and mechanical systems can be found in [33, 34]. The neural networks have also been applied to engine modelling. The manifold pressure and mass flow processes were modelled in [35] with eternal recurrent networks.

In this thesis, the engine system is recognized as a complex dynamical system which memorizes its previous states. From this point of view, the calibration of the engine can be based on a series of dynamic I/O models. With proper designed test cycle, namely New European Driving Cycle or random walk signal, the engine static or dynamical model can be developed solely from the information contained in the data set. It can be concluded that the model-based engine calibration can significantly reduce the pre-production time thus it is even more promising in the future industrial application.

Chapter 3

Experimental and Virtual Test Bed Setup

This chapter describes the experimental equipment and setup used for investigation of the identification techniques for multi-model development and local model structure identification.

3.1 Ford Port-Fuel-Injected (PFI) Zetec Engine

The Liverpool University low inertial dynamometer system installed with Ford Zetec 1.6L PFI 4 cylinder S.I. engine is used for the experimental studies in this thesis. A schematic picture of an engine dynamometer is represented in Figure 3.1. The engine dynamometer system comprises a test engine and a dynamometer with a connecting transmission shaft. The engine torque can be controlled by a combination of air, fuelling, spark angle inputs. An in-line torque transducer is available to measure the torque, and a total reaction torque measurement arrangement is also available. The other dynamometer parameter is the rotational speed of the engine shaft and it is measurable by a shaft encoder. A detailed comparison between PFI and GDI engine mechanism can be found in [36]. The direct injection techniques are offen used to achieve better fuel dispersion and homogeneity. The control strategies of GDI combustion is briefly introduced in [37, 38].

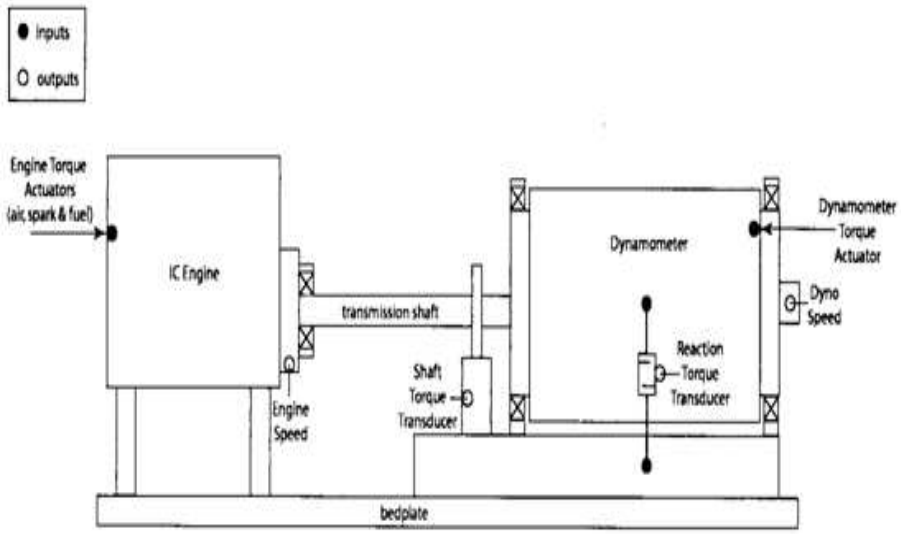

Figure 3.1: A schematic structure of the engine connecting with dynamometer

The results obtained from the engine dynamometer identification provide the data to determine the effectiveness for any engine modelling or control strategies. Four basic control modes may be adopted to control the dynamometer [39]:

- Independent Mode: In this mode, the dynamometer torque position is controlled open-loop, which means the disturbances to the model state will not affect the input to the dynamometer actuator.
- Speed: Here, the speed of the dynamometer is adjusted using closed-loop control with dynamometer speed as the tracked variable.
- Torque: the closed loop system is adopted in this mode to track a desired shaft torque. This mode is useful for the simulation of external loads to increment or decrement the tracked torque and is obtained by the addition of disturbance signals to the dynamometer torque actuation channel.
- Power: The engine power is determined by the product of torque and rotational speed. Hence the engine power is controlled by adjusting both of them simultaneously.

For identification purposes, the engine system is regarded as a black box. Hence the main objective of the experiment is to provide the information to reveal the characteristics of the engine system dynamics. A dSPACE system, a data pre-processing device, is applied

to collect and record the raw data from the engine test. The data thus obtained from the experiment are inputs to the identification process to determine the mathematical model and provide the evidence for the validation of the resulting model. The directly measured variables include ABV (Air Bleed Valve) Duty Cycle, MAP (Manifold Air Pressure), ABV, SA and RPM of the engine crank. Among these parameters, for modelling in this thesis, the ABV Duty Cycle and MAP are regarded as measured input signal while the RPM and UEGO reading λ are chosen as the measured output. The experimental set is shown in Figure 3.2.

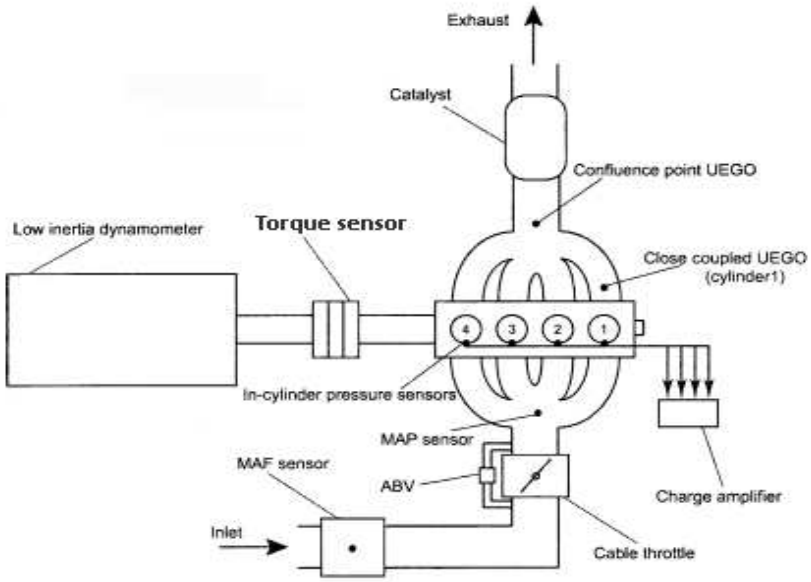

Figure 3.2: The key control devices for the engine test

The test engine is a four cylinder engine with four strokes cycle providing four strokes every 720°. The number of raw data, the samples can achieve 1 million, since samples are collected by the software each degree of crank angle. In order to simplify the input control, the throttle switch is turned off which corresponds to a fixed throttle angle of 8°, thus almost all the entered air is bypassed through the ABV. In order to reduce the error, it is necessary to collect adequate amount of data samples for data processing. Before the test is started, the engine speed is adjusted to around the idle speed of 1000 rpm and a braking load is applied by the dynamometer. The dSPACE system is used to collect all the signals and sample them with the 1 degree interval.

3.2 Variable Cam Timing(VCT) Engine

For conventional gasoline engines, the intake and exhaust valves are driven by the camshafts and the cam timing is adjusted by a timing belt which binds the crankshaft and the camshaft. By adjusting the phase difference between the camshaft and the crankshaft angle, the timing belt can synchronize the valve and piston movements in a fixed timing. With optimized timing of inlet and outlet valve opening in variable valve timing (VVT) engine, the volumetric efficiency is boosted and the torque output can be increased in all the engine operating regions thus the fuel economy is improved [40]. Meanwhile, the retard of exhaust valve opening will reduce or replace the use of exhaust gas recirculation system which would generate more CO and NOx emissions [41].

Variable Camshaft Timing (VCT) is an automotive variable valve timing technology in current advanced engines used by many manufacturers including the Ford Motor Coperation. The VCT technology is an innovation used to minimize the engine emissions whilst providing also satisfactory fuel economy and driver experience. It has been proved effective and applied extensively in the industry. This technology has offered considerable merits: it is able to reduce emissions such as oxides of nitrogen(NOx) and unburned hydrocarbons(HC) [42], and also to improve the full load performance of the engine [43]. The specific feature of the VCT engine, the cam timing is precisely advanced on retarded according to the engine speed as determined by an optimized engine mapping.

VCT offers another effective way to reduce NOx and HC emissions other than the exhaust gas recirculation(EGR) valve systems. However, the VCT mechanism requires more sophisticated control schemes to achieve the emission reduction target. As shown in Fig 3.3 , the exhaust gas is sucked back in to the cylinder after TDC is reached. The gas mixture in the cylinder is diluted by the retained exhaust gas which reduces the combustion temperature. Therefore, the NOx generation is suppressed under lower temperature. Meanwhile, the recirculated exhaust gas containing unburned HC from the cylinder or piston wall interface will undergo combustion in the next engine cycle. The camshaft of the VCT engine is actuated by a hydraulic mechanism and the valve timing is adjusted according to the engine inlet and exhaust cycle. There are several cam timing schemes, namely Dual Variable Cam Timing (Dual-VCT) and Twin Independent Variable Cam Timing (Ti-VCT). The cam tim-

ing of Dual-VCT is adjusted simultaneously with the fixed phase differences. On the other hand, the Ti-VCT adjusts the intake and exhaust camshaft separately in order to produce higher power and torque and at the same time keep the emissions to a minimum. However, the Ti-VCT scheme will increase the complexity of the timing actuator and the cost of the engine.

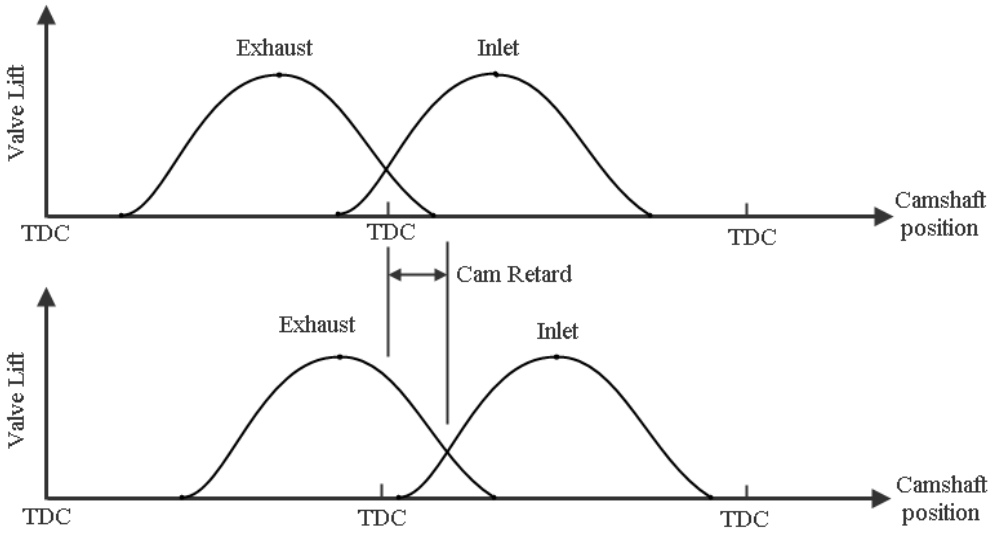

Figure 3.3: Variable Camshaft Timing scheme

In this thesis, a VCT engine model developed by Stefanopoulou [44] is investigated by establishing a Matlab Simulink model and simulating the dynamical properties obtained from the empirical polynomial model given in Stefanopoulou's work.

The calibration and control target of our study is to maintain the stoichiometric Air/Fuel Ratio while minimizing the HC and NOx emissions. The key features of the engine, such as the throttle angle, engine pumping rate, torque generation and NOx and HC emissions are included as the calibration variables. In detail, the manifold air pressure is physically determined by the mass flow rate into the throttle body and cylinder as shown in Equations (3.1) (3.2).

$$K_m = \frac{R \cdot T}{V_m} = \frac{287 \frac{J}{Kg \cdot K} \cdot 288\ K}{0.007\ m^3} = 0.118\ bar/g \tag{3.1}$$

where R is the specific gas constant ($287\ \frac{J}{Kg \cdot K}$), T represents the nominal temperature

(288 K), V_m represents the manifold volume (0.007 m^3), and K_m is the resulting constant. The rate of change of the manifold pressure is determined by:

$$\frac{d}{dt}P_m = K_m(\dot{m}_\theta - \dot{m}_{cyl}) \tag{3.2}$$

where $\dot{m}_\theta$ represents the mass air flow rate into the manifold and $\dot{m}_{cyl}$ is the engine pumping mass air flow rate. These two air flow rates can be calculated by Equations (3.3) and (3.4) respectively:

$$\dot{m}_\theta = g_1(P_m) \cdot g_2(\theta) \tag{3.3}$$

$$\dot{m}_{cyl} = F(1, CAM, CAM^2, CAM^3, P_m, P_m^2, P_m^3, N, N^2, N^3) \tag{3.4}$$

where CAM is the camshaft angle, N is the engine speed, P_m is the manifold pressure, θ represents the throttle angle and

$$g_1(P_m) = \left\{ \begin{array}{cc} 1 & if \quad P_m \leq P_0/2, P_0 = 1\ atm \\ 2/P_0\sqrt{P_m P_0 - P_m^2} & if \quad P_m > P_0/2 \end{array} \right\} \tag{3.5}$$

and

$$g_2(\theta) = F(1, \theta, \theta^2, \theta^3) \tag{3.6}$$

The cylinder mass air flow, torque and NOx and HC emissions are modeled by the static algebraic functions of cylinder air charge (CAC), camshaft angle (CAM), manifold air pressure (Pm), air/fuel ratio (A/F) and engine speed (N). The schematic output model structure is outlined in Equation (3.4) (3.7) (3.8) and (3.9):

$$T_q = F(1, CAC, CAC^2, CAC^3, A/F, A/F^2, A/F^3, N, N^2, N^3) \tag{3.7}$$

$$NO_x = F_1(1, N, N^2) \cdot F_2(1, CAM) \cdot F_3(1, A/F, A/F^2, A/F^3) \cdot F_4(1, P_m, P_m^2) \tag{3.8}$$

$$HC = F_1(1, N, N^2) \cdot F_2(1, A/F, A/F^2, A/F^3) \cdot (F_3(1, \frac{1}{P_m}) + F_4(1, CAM, \frac{CAM}{P_m^{\frac{1}{16}}})) \tag{3.9}$$

A new Simulink realization which simulates this VCT engine model is shown in Figure 3.5. The Simulink model is constructed in four subsystems which determine the four-stoke phases: throttle and manifold intake, air/fuel compression, ignition and combustion, exhaust generation. Each subsystem has its own physical and empirical sub-models with corresponding inputs and outputs. For the intake system, air flow rate, throttle angle and manifold air pressure are recognized as inputs. In the final stage, the brake torque is a modelled output of the combustion system while the NOx and HC emissions are estimated outputs of the exhaust system.

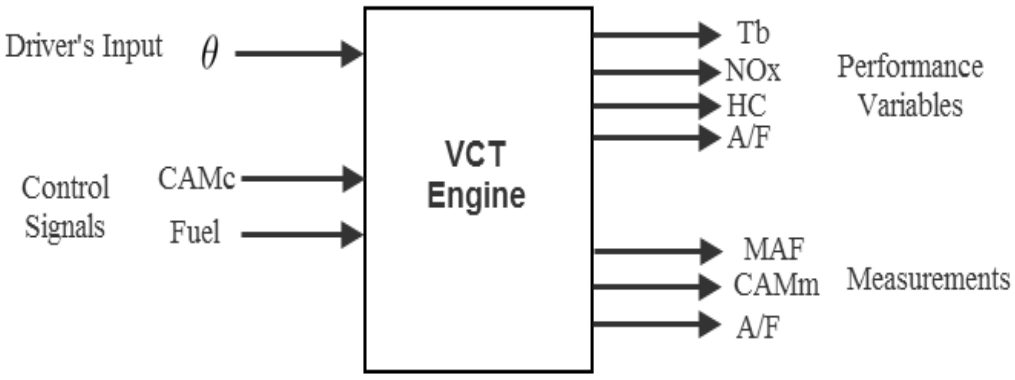

Figure 3.4: Inputs and Outputs of the VCT Engine

3.3 Diesel Engine with VGT and EGR

Diesel Engines are widely used in heavy-duty vehicles due to its higher torque output and fuel economy. Two types of compression enhancement devices are widely used in industry:

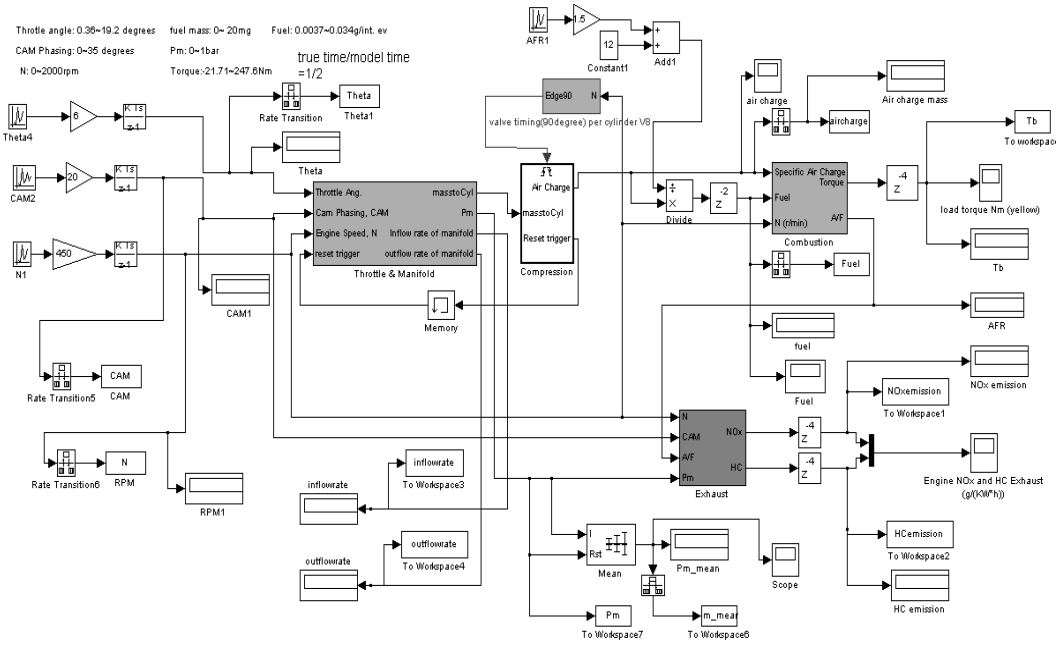

Figure 3.5: VCT Engine Simulation Model in Simulink

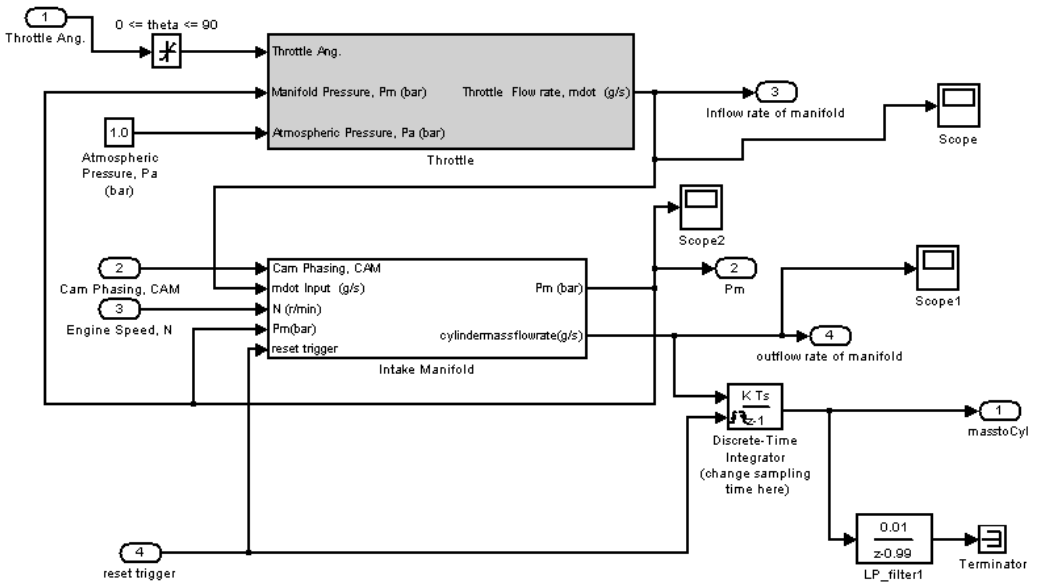

Figure 3.6: Throttle and Manifold Simulation Model

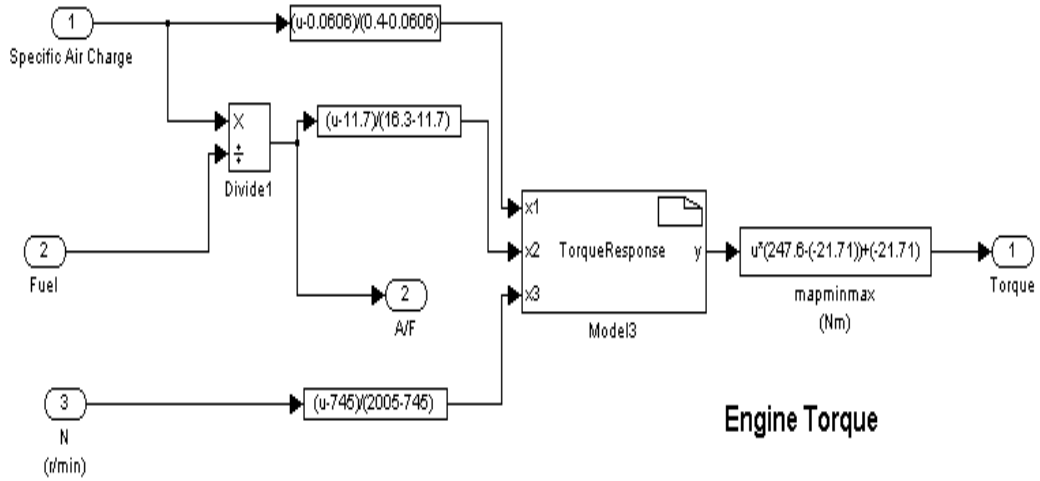

Figure 3.7: VCT Engine Torque Model

the turbocharger and the supercharger. However, these two mechanism adopts different working strategy. The turbochargers utilize the kinetic energy of the exhaust gas flow and improved the energy efficiency of the engine, but it only works effectively when the engine operates at relatively high speed. Meanwhile, the superchargers are directly connected with the crankshaft which is driven by the engine main combustion power, therefore its response time is shorter and working consistently during the engine operating time. Nevertheless, the supercharger requires to keep drawing power from the engine which reduce the power and torque output on the wheels [45].

For the purpose of minimizing the emission within the legislated limit, exhaust recirculation and variable-geometry turbochargers have been introduced into the Diesel engine design. A well-established Diesel engine model which is presented in [46] was adopted as a test template for the multi-model. This model was peer-reviewed and validated from the industrially obtained experimental data [47] [48] [49]. As shown in Figure 3.8, eight state variables are involved in the engine model structure: the intake manifold pressure p_{im}, the exhaust manifold pressure p_{em}, the intake oxygen mass fraction X_{Oim}, the exhaust oxygen mass fraction X_{Oem}, the turbocharger speed ω_t, and three actuator state variable u_δ,u_{egr},u_{vgt} where u_δ is the mass of injected fuel, u_{egr} represents the EGR valve position and the u_{vgt} is the VGT valve position. The valve position value varies from 0(closed) to 1(fully open). The adjustable input variables are shown in Equation 3.10 and the model has the state space

form of Equation 3.11:

$$\mathbf{u} = (u_\delta \quad u_{egr} \quad u_{vgt})^T \tag{3.10}$$

where u_δ is the injected fuel,u_{egr} is the position of EGR valve, and u_{vgt} is the actuator position of the VGT.

The state-space model:

$$\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u}, N) \tag{3.11}$$

where the function $F(\cdot)$ is developed by combining the dynamic physical properties of all the key components shown in Figure 3.8 [50]. The dynamic testing of the Diesel engine was conducted by generating randomly input sequences for $\mathbf{u}$ and then observing the output states $\mathbf{x}$ at different engine speed(N) through the time of the test.

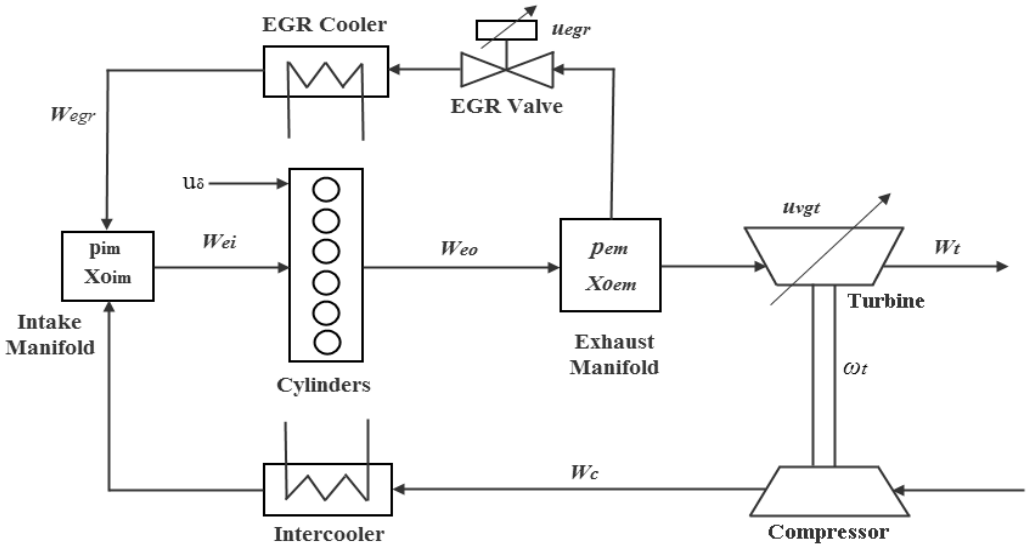

Figure 3.8: Inputs and Outputs of the Diesel Engine with VGT mechanism

3.4 WAVE-RT Petrol Engine Model

The model structure identification techniques is applied to a high accurate industrially produced and validated virtual engine model implemented as a WAVE-RT system. Some major

disturbances, namely, temperature and moisture, could cause system uncertainty and compromise the validation results however in terms of experimental effectiveness, the computer-aided engine simulator is able to conduct highly repeatable engine test within relative low time and cost budget. At the same time, the engine model maintains good flexibility to changing component units and parameters. The subject engine for WAVE-RT model is an EcoBoost 2.0-Litre GTDI engine provided by the Ford Motor Co.. The WAVE real-time model is built on an ISO approved software package named RICARDO WAVE platform [51, 52]. The WAVE platform contains a library of modelling components in the area of compressible-flow fluid networks and machinery. In terms of engine modelling, it offers elements such as piston compressor, engine cylinder, throttle, turbocharger and turbine. The time-variant variables for the engine, including temperature, cylinder pressure and air mass flow rates, can be measured directly in SI or Imperial units and monitored within the simulation process.

WAVE is recognized as a state-of-the-art industrial design tool and has extensive application in engine performance assessment. From early concept studies to detailed engine production investigations, the WAVE model can be used throughout the engine design process. Common applications include torque response, fuel consumption, turbocharger response, EGR studies, valve profile design and timing optimization. As shown in Figure 3.9, for the sake of simplicity of use, the Wave-RT model is mainly comprised of a series of connection of physical models of engine components along with the engine breathing pipeline, including throttle and cylinder intake mechanism. As a concept engine design tool, the wave model is recognized by Ford engineers as a realistic alternative to the real engine for conducting initial stages of a control scheme. The engine simulation results obtained from the WAVE model can be plotted in various standard graphical formats and validated by different types of real engine test data.

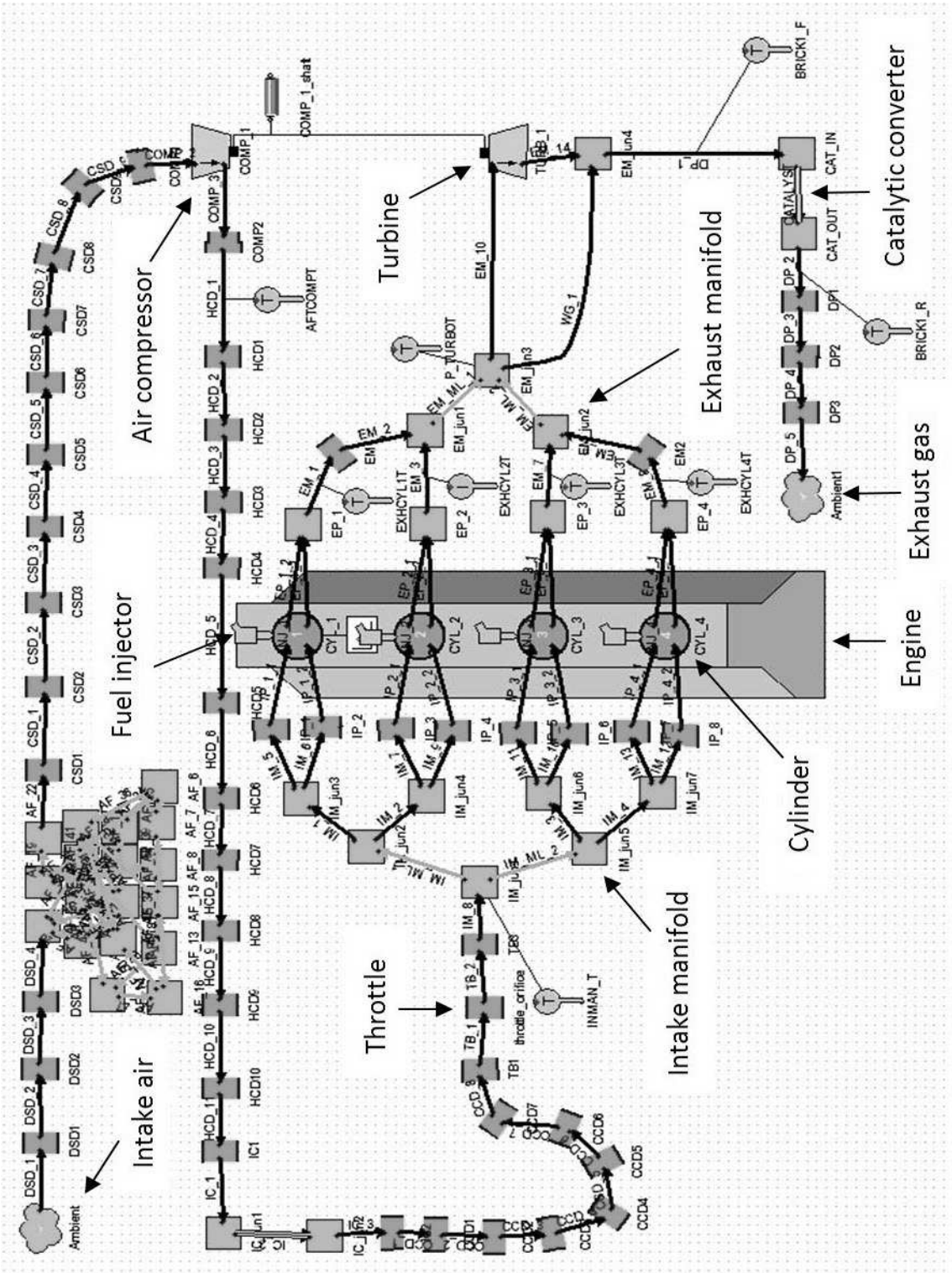

Figure 3.9: WAVE model developed for GTDI engine

Chapter 4

Engine SISO Model Identification

4.1 Method of Least Squares Parameter Identification

The methods used for the data processing are the most significant factors which affect the accuracy and reliability of the identification results. The basic algorithm for parameter identification, the least-squares algorithm, is introduced in this chapter.

Least-squares (LS) theory was first proposed by Karl Gauss for predicting the orbit of the planets. Since then, the LS theory has become a principle tool for parameter estimation from experimental data. The LS method is easy to comprehend and does not require an extensive knowledge of mathematical statistics; thus it is widely used among scientists and engineers. Furthermore, estimates obtained by LS methods have optimal statistical properties which are consistent, unbiased and efficient [11] . Before applying LS theory, the choice of input and output signal of the system are required to be decided.

4.1.1 Definition of the vector norm

The concept of vector norms is described in the mathematical representation. A vector norm on R^n is a function $f : R^n \rightarrow R$ that satisfies the following properties [53]:

$$f(x) \geq 0 \qquad x \in R^n, \quad (f(x) = 0 \quad iff \ x = 0) \tag{4.1}$$

$$f(x+y) \leq f(x) + f(y) \qquad x, y \in R^n \tag{4.2}$$

$$f(\alpha x) = |\alpha|\, f(x) \qquad \alpha \in R, x \in R^n \tag{4.3}$$

Such a function is denoted with a double bar notation: f(x)=$\|\mathbf{x}\|$. With different subscripts on the double bar, various norms can be distinguished. Then the p-norms are defined by:

$$\|\mathbf{x}\|_p = (|x_1|^p + \cdots + |x_n|^p)^{\frac{1}{p}} \qquad p \geq 1 \tag{4.4}$$

Three of these norms are the most significant:

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n| \tag{4.5}$$

$$\|\mathbf{x}\|_2 = (|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2)^{\frac{1}{2}} = (x^T x)^{\frac{1}{2}} \tag{4.6}$$

$$\|\mathbf{x}\|_\infty = max_{1 \leq i \leq n} |x_i| \tag{4.7}$$

Note that a unit vector is a vector x which satisfies $\|\mathbf{x}\|_p$=1.

4.1.2 Least-squares via the minimum error-squares

an n-tuple set $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)$ are related linearly to a variable $\mathbf{y}$:

$$\mathbf{y} = \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + \cdot + \theta_n \mathbf{x}_n \tag{4.8}$$

where $\theta = (\theta_1, \theta_2, \cdots, \theta_n)^T$ is an n-tuple constants so that $\mathbf{y} = \mathbf{X}\theta$.
Let the sum of squares of errors be determined as follows [54]:

$$\begin{aligned} J = \sum_{t=1}^{N} e^2(t) = \mathbf{e}^T\mathbf{e} \\ = (\mathbf{y} - \mathbf{X}\theta)^T(\mathbf{y} - \mathbf{X}\theta) \\ = \mathbf{y}^T\mathbf{y} - \theta^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\theta + \theta^T\mathbf{X}^T\mathbf{X}\theta \end{aligned} \tag{4.9}$$

The equation to determine the stationary point for derivation of the minimum of J with respect to θ is:

$$\frac{dJ}{d\theta} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\theta = 0 \tag{4.10}$$

which leads to

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\theta \tag{4.11}$$

The second derivative of J can be represented as:

$$\frac{d^2J}{d\theta^2} = 2\mathbf{X}^T\mathbf{X} \tag{4.12}$$

which requires $\mathbf{X}^T\mathbf{X} > 0$ for a minimum and therefore the stationary point obtained from Equation 4.11 is able to make the sum of squares J minimum. Hence the least square estimator (LSE) for the polynomial model 4.8 is

$$x_{LS} = \hat{\theta} = \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{y} \tag{4.13}$$

4.1.3 Least-squares via QR algorithm

Let $X \in R^{m\times n}$ with $m \geq n$ and $b \in R^m$ and suppose $Q \in R^{m\times m}$ is an orthogonal matrix such that

$$\begin{aligned} Q^T = Q^{-1} \\ Q^TQ = QQ^T = I \end{aligned} \tag{4.14}$$

where I is the identity matrix. The QR algorithm starts from the matrix formation that

$$Q^T X = R = \begin{bmatrix} R_{1,n\times 1} \\ 0_{(m-n)\times 1} \end{bmatrix} \tag{4.15}$$

where the matrix X is formed as $X = [x_1, x_2, x_3..., x_n]^T$ and if

$$Q^T b = \begin{bmatrix} h_{n\times 1} \\ g_{(m-n)\times 1} \end{bmatrix} \tag{4.16}$$

In the least-squares algorithm, the aim is to determine $\hat{\theta}$ which is able to minimize $\|X\theta - b\|_2^2$. Obviously, this problem is equivalent to the minimization of $\left\|Q^T X\theta - Q^T b\right\|_2^2$. In this case

$$\|X\theta - b\|_2^2 = \left\|Q^T X\theta - Q^T b\right\|_2^2 = \|R_1\theta - h\|_2^2 + \|g\|_2^2 \tag{4.17}$$

For any $x \in R^n$, if rank(X)=rank(R_1)=n, then $\hat{\theta}$ is defined by the upper triangular system such that

$$\begin{aligned} R_1\hat{\theta} &= h \\ \hat{\theta} &= R_1^{-1}h \end{aligned} \tag{4.18}$$

Based on this QR Algorithm, the coefficient vector of $\hat{\theta}$ can be obtained.

4.1.4 Least-squares via SVD algorithm

In linear algebra, the SVD is a vital factorization of a rectangular real or complex matrix, with several applications in signal processing and statistics. Applications which employ the SVD include computing the pseudo-inverse, matrix approximation, and determining the rank, range and null space of a matrix. In the SVD factorization, suppose $X \in R^{m\times n}$, the matrix

Q and Z are orthogonal matrices such that

$$Q^T XZ = T = \begin{bmatrix} T_{11} & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} \qquad r = rank(X)$$

$$\|X\theta - b\|_2^2 = \left\|(Q^T XZ)Z^T\theta - Q^T b\right\|_2^2 = \|T_{11}a - c\|_2^2 + \|d\|_2^2 \tag{4.19}$$

where in the Equation 4.19, $X = [x_1, x_2, x_3..., x_n]^T$,

$$Z^T\theta = \begin{bmatrix} a_{r\times 1} \\ b_{(n-r)\times 1} \end{bmatrix} \qquad Q^T b = \begin{bmatrix} c_{r\times 1} \\ d_{(m-r)\times 1} \end{bmatrix}$$

If the 2-norm of x is minimum, b must be zero and $a = T_{11}^{-1}c$. Therefore,

$$x_{LS} = \hat{\theta} = Z \begin{bmatrix} T_{11}^{-1}c \\ 0 \end{bmatrix} \tag{4.20}$$

4.1.5 Least squares via Recursive Least-Squares (RLS) algorithm

The requirement for recursive solutions arise when fresh experimental data are continuous in supply and on-line results are required. With recursive formula, the LS estimates can be updated at each experimental data sample. The recursive methods are often called sequential or on-line estimation methods.

If the first m groups of data are embedded in matrix $\mathbf{X}_m$ and $\mathbf{y}_m$, then

$$\mathbf{y}_m = \mathbf{X}_m\theta \tag{4.21}$$

Hence the least squares estimator can be written as:

$$\hat{\theta}(m) = (\mathbf{X}_m^T\mathbf{X}_m)^{-1}\mathbf{X}_m^T\mathbf{y}_m \tag{4.22}$$

Assume a vector of new experimental data $\mathbf{x}(m+1)$ and $\mathbf{y}(m+1)$ are then obtained, where

$$\mathbf{y}(m+1) = \theta_1 x_1(m+1) + \theta_2 x_2(m+1) + \cdots + \theta_n x_n(m+1) \tag{4.23}$$

if we define

$$\mathbf{x}^T(m+1) = [x_1(m+1), x_2(m+1), \cdots, x_n(m+1)] \tag{4.24}$$

Then the output of next time point is

$$y(m+1) = \mathbf{x}^T(m+1)\theta \tag{4.25}$$

Consequently, the new data system including the $(m+1)th$ equation can be written as

$$\mathbf{y}_{m+1} = \mathbf{x}_{m+1}\theta \tag{4.26}$$

in which

$$\mathbf{y}_{m+1} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(m) \\ y(m+1) \end{bmatrix} = \begin{bmatrix} \mathbf{y}_m \\ \\ \\ \\ y(m+1) \end{bmatrix} \tag{4.27}$$

$$\mathbf{X}_{m+1} = \begin{bmatrix} x_1(1) & \cdots & x_n(1) \\ \vdots & \ddots & \vdots \\ x_1(m) & \cdots & x_n(m) \\ x_1(m+1) & \cdots & x_n(m+1) \end{bmatrix} = \begin{bmatrix} \mathbf{X}_m \\ \\ \\ \mathbf{x}^T(m+1) \end{bmatrix} \tag{4.28}$$

Hence the new least-squares estimator of next iteration is

$$\hat{\theta}(m+1) = [\mathbf{X}_{m+1}^T \mathbf{X}_{m+1}]^{-1} \mathbf{X}_{m+1}^T \mathbf{y}_{m+1} \tag{4.29}$$

Without matrix inversion, the new estimator $\hat{\theta}(m+1)$ can be calculated by simply updating the previous estimate $\hat{\theta}(m)$. A well-known matrix inversion lemma is the key to achieve

this. Let **A**, **C** and **A+BCD** be nonsingular square matrices; then the following Sherman-Morrison-Woodbury formula holds [11]:

$$(\mathbf{A}+\mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1} \tag{4.30}$$

where A is n-by-n matrix, U is n-by-k, C is k-by-k and V is k-by-n matrix. If we define the matrix $\mathbf{P}(m)$ as

$$\mathbf{P}(m) = (\mathbf{X}_m^T\mathbf{X}_m)^{-1} \tag{4.31}$$

then

$$\mathbf{P}(m+1) = (\mathbf{X}_{m+1}^T\mathbf{X}_{m+1})^{-1} = (\mathbf{X}_m^T\mathbf{X}_m + \mathbf{x}(m+1)\mathbf{x}^T(m+1))^{-1} \tag{4.32}$$

Applying the matrix inversion lemma described in Equation 4.30, let $\mathbf{A} = \mathbf{P}(m)^{-1}$, $\mathbf{B} = \mathbf{x}(m+1)$, $\mathbf{C} = 1$ and $\mathbf{D} = x^T(m+1)$, then $\mathbf{P}(m+1)$ can be written as follows:

$$\mathbf{P}(m+1) = [\mathbf{P}(m)^{-1} + \mathbf{x}(m+1)\mathbf{x}^T(m+1)]^{-1} = \mathbf{P}(m) -$$
$$\mathbf{P}(m)\mathbf{x}(m+1)[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)]^{-1}\mathbf{x}^T(m+1)\mathbf{P}(m) \tag{4.33}$$

From Equation 4.27 and 4.28

$$\mathbf{X}_{m+1}^T\mathbf{y}_{m+1} = \mathbf{X}_m^T\mathbf{y}_m + \mathbf{x}(m+1)y(m+1) \tag{4.34}$$

Substituting Equation 4.32 and 4.34 into Equation 4.29, the following relation holds:

$$\hat{\theta}(m+1) = \mathbf{P}(m+1)[\mathbf{X}_m^T\mathbf{y}_m + \mathbf{x}(m+1)y(m+1)]$$
$$= (\mathbf{P}(m) - \mathbf{P}(m)\mathbf{x}(m+1)[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)]^{-1}$$
$$\cdot\mathbf{x}^T(m+1)\mathbf{P}(m))(\mathbf{X}_m^T\mathbf{y}_m + \mathbf{x}(m+1)y(m+1)) \tag{4.35}$$

The above equation can be transformed into

$$\begin{aligned}\mathbf{P}(m)\mathbf{X}_m^T\mathbf{y}(m) - \mathbf{P}(m)\mathbf{x}(m+1)[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)^{-1}\\ \cdot\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{X}_m^T\mathbf{y}_m + \mathbf{P}(m)\mathbf{x}(m+1)y(m+1)\\ -\mathbf{P}(m)\mathbf{x}(m+1)[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)]^{-1}\\ \cdot\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)y(m+1) \qquad (4.36)\end{aligned}$$

The last two terms of Equation 4.36 can then be rearranged into the form

$$\begin{aligned}\mathbf{P}(m)\mathbf{x}(m+1)[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)]^{-1}\\ \cdot[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)\\ -\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)]\cdot y(m+1)\\ = \mathbf{P}(m)\mathbf{x}(m+1)[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)]^{-1}y(m+1) \qquad (4.37)\end{aligned}$$

From Equation 4.22 and 4.31, the following relations can be obtained:

$$\hat{\theta}(m) = \mathbf{P}(m)\mathbf{X}_m^T\mathbf{y}_m \qquad (4.38)$$

Finally, the $\hat{\theta}(m+1)$ can be written as

$$\begin{aligned}\hat{\theta}(m+1) = \hat{\theta}(m) - \mathbf{P}(m)\mathbf{x}(m+1)[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)]^{-1}\\ \cdot\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{X}_m^T\mathbf{y}_m\\ +\mathbf{P}(m)\mathbf{x}(m+1)[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)]^{-1}y(m+1)\\ = \hat{\theta}(m) + \mathbf{P}(m)\mathbf{x}(m+1)[1+\mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)]^{-1}\\ \cdot[y(m+1) - \mathbf{x}^T(m+1)\hat{\theta}(m)] \qquad (4.39)\end{aligned}$$

The result obtained in Equation 4.39 shows that the next estimator is calculated by the previous one plus a correction term. Therefore, the recursive least-squares estimation can be executed by the following equations:

$$\hat{\theta}(m+1) = \gamma(m+1)\mathbf{P}(m)\mathbf{x}(m+1)[y(m+1) - \mathbf{x}^T(m+1)\hat{\theta}(m)] \qquad (4.40)$$

$$\mathbf{P}(m+1) = \mathbf{P}(m) - \gamma(m+1)\mathbf{P}(m)\mathbf{x}(m+1)\mathbf{x}^T(m+1)\mathbf{P}(m) \tag{4.41}$$

where

$$\gamma(m+1) = \frac{1}{1 + \mathbf{x}^T(m+1)\mathbf{P}(m)\mathbf{x}(m+1)} \tag{4.42}$$

4.2 Engine SISO Model identification using LS algorithm

In order to validate the least squares method in engine system identification, an engine test was conducted on the 1.6L PFI gasoline engine with the objective to identify a single-input-single output (SISO) ARX model with optimized terms and orders.

The primary parameters concerned with the experiment are the ABV duty and the rotational speed of the engine crank. The signal of the ABV duty was taken as input and was varied between 0.380 and 0.410, and the response of the rotational speed was determined as output. When the number of the raw data samples is very large, it may cause low efficiency of the computation. Hence, the raw data for the test were required to be down-sampled before analyzing the experimental data in the Matlab. In this application, the interval between samples is changed to 180 degrees after down sampling and around 10000 samples were extracted from the raw data. The resulting data for identification and validation are shown in Figure 4.1 and Figure 4.2. The validation data shown in Figure 4.2 is collected under the same input ABV range with that of idenfication data in Figure 4.1.

In this application, two criteria have been adopted for model validation:

- *Goodness of Fit* using the fit ratio

$$fit\ ratio = 100\% \times (1 - norm(\hat{\mathbf{y}} - \mathbf{y})/norm(\mathbf{y} - \bar{\mathbf{y}})) \tag{4.43}$$

- *Coefficient of Determination* (R^2)

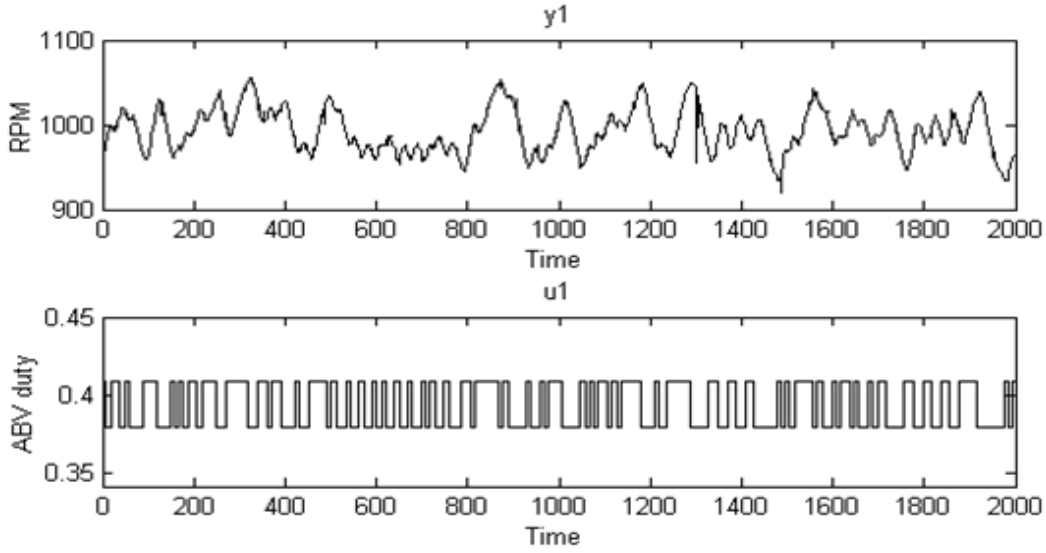

Figure 4.1: The data used for the SISO model identification

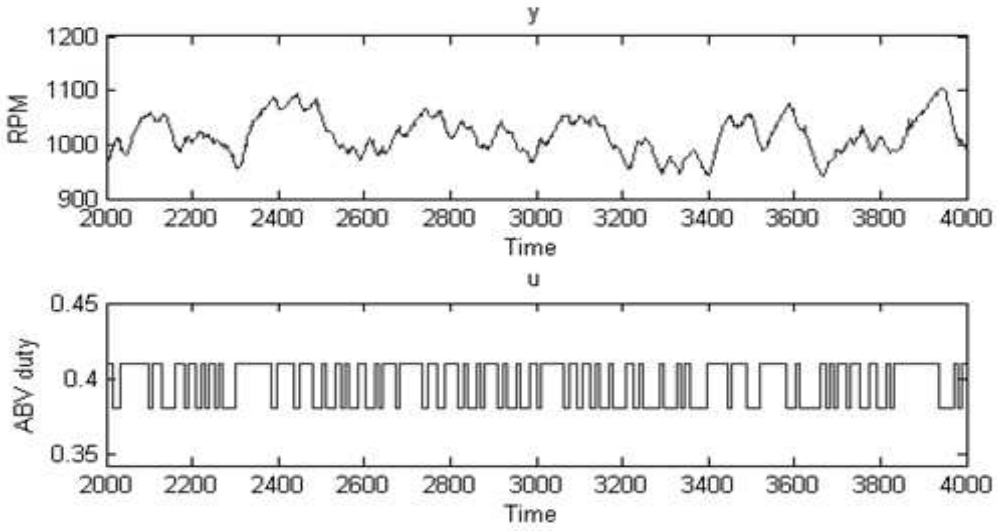

Figure 4.2: The data used for the SISO model validation

$$residual = \mathbf{y} - \hat{\mathbf{y}}$$

$$R^2 = 1 - \frac{\sum_i (\mathbf{y}_i - \hat{\mathbf{y}})^2}{\sum_i (\mathbf{y}_i - \bar{\mathbf{y}})^2} \tag{4.44}$$

where $\hat{\mathbf{y}}$ is the model estimated output and $\bar{\mathbf{y}}$ is the mean value of the measured output vector $\mathbf{y}$.

4.2.1 The coherence of the SISO input/output

The coherence function can be used to describe the linearity between the input and output signal. It is a function of frequency with values between 0 and 1 that indicate how well the input X corresponds to the output Y at each frequency. Figure 4.3 is obtained by using the coherence function MSCOHERE (X, Y) in Matlab. By using Welch's averaged periodogram method, the magnitude-squared coherence of the system is estimated with input X and output Y. In this test, the raw data is downsampled into 180 degree/sample while the engine speed is around 1000 rpm, so the sampling rate is therefore 0.03 sec/sample. In the range of normalized frequency from 0 to 0.1π rad/sample, the coherence magnitude is close to 1 hence the ABV duty and the rotational speed are approximately linear correlated in this range. Therefore, it is feasible to use the linear ARX model as a primary engine model.

4.2.2 The transfer function of the ARX model

For an ARX model, the optimum solution should be a compromise between minimum orders and the highest fit quality of the model. In the Matlab system identification toolbox, several groups of parameters of the ARX model have been tested. The structure of the transfer function can be represented as:

$$G(z) = \frac{a_{n_b-1} + a_{n_b-2}z^{-1} + \cdots + a_1 z^{2-n_b} + a_0 z^{1-n_b}}{1 + b_{n_a-1}z^{-1} + b_{n_a-2}z^{-2} + \cdots + b_1 z^{1-n_a} + b_0 z^{-n_a}} \cdot z^{-n_k} \tag{4.45}$$

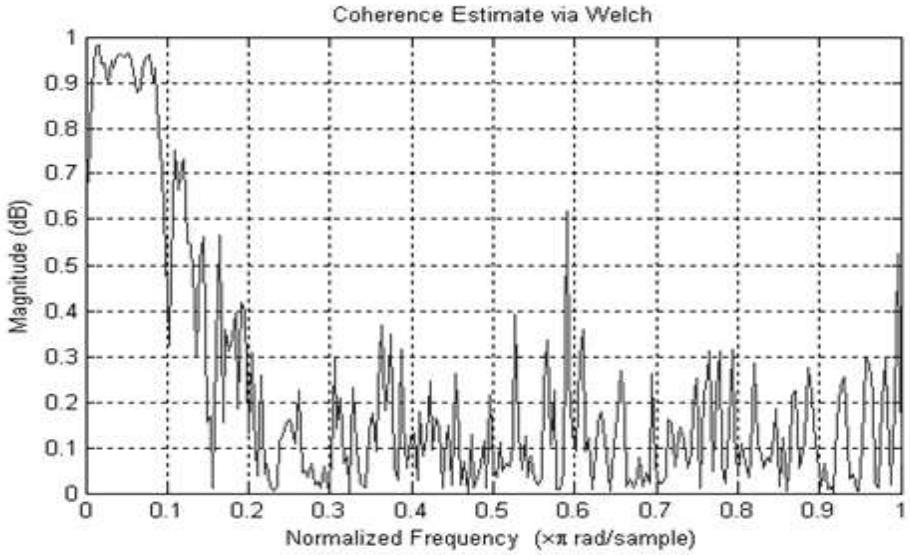

Figure 4.3: Coherence estimate via Welch's method

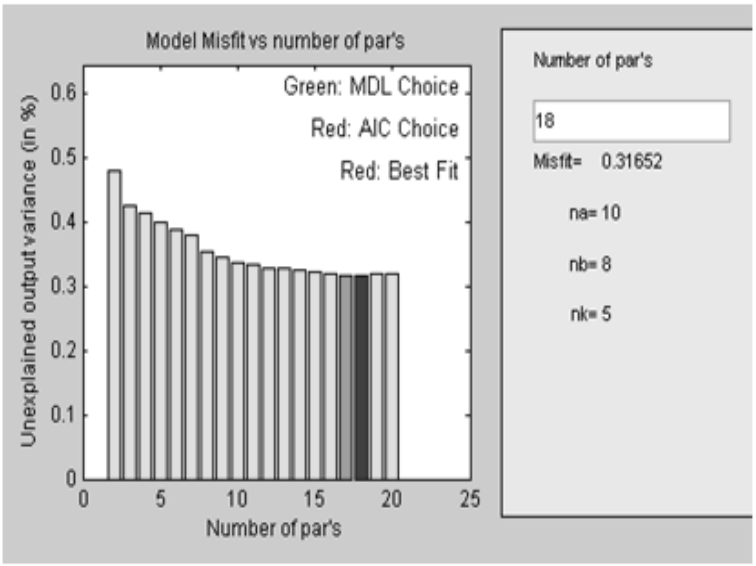

Figure 4.4: Order selection analysis

The ARX order selection plot is represented in Figure 4.4. The x-axis shows the number of parameters in the respective models. The y-axis shows the part of the output variance, which is not explained by the model, that is, the ratio between the prediction error variance and the output variance in percentage. The bar chart demonstrates that the best fit of the ARX model is $(1 - 0.316) = 0.684$. Meanwhile, the bar chart also suggests that both the Akaike Information Criteria (AIC) [55] and Rissanen's Minimum Description Length (MDL) [56] criteria has been adopted for the model selection and the best model fit is obtained by the AIC criteria.

The system identification usually does not have a unique solution. It is necessary to obtain compromises between testing effort (hours on the engine), model order and model quality. The fact is that the fit would not get significantly better for an increase in model order. Finally, the ARX model which implements the orders of $n_a = 6$,$n_b = 6$,$n_k = 6$ is adopted, since the corresponding simulated output curve has the highest fit ratio with the measured output curve. The transfer function obtained from the identification toolbox function $[sys = arx(data, n_a, n_b, n_k)]$ is

$$G(z) = \frac{40.98z^{-6} + 15.69z^{-7} + 23.6z^{-8} + 18.99z^{-9} + 20.27z^{-10} + 65.12z^{-11}}{1 - 0.4303z^{-1} - 0.3422z^{-2} - 0.2126z^{-3} - 0.1174z^{-4} - 0.0136z^{-5} + 0.143z^{-6}} \quad (4.46)$$

As shown in Figure 4.5, the predicted output signal matches well with the real output. According to Equation 4.43, the fit ratio of this ARX model is 75.8% which is defined by 4.43.

The least squares method is also applied to determine the transfer function. The least squares via QR, SVD and minimum error-squares have been implemented to build the models respectively. As shown in Figure 4.6, the transfer function developed from different least squares methods are linked into a validation Simulink model in parallel form.
The result shows that the output curves of different linear models matches well (<0.5% mismatch). Therefore, it can be concluded that the linear ARX model is not sensitive to the different LS algorithms. As a result, the transfer function can be unified as the one with best fit:

$$G(z) = \frac{40.3954z^{-6} + 20.4388z^{-7} + 24.7377z^{-8} + 13.2827z^{-9} + 19.6590z^{-10} + 60.7806z^{-11}}{1 - 0.4403z^{-1} - 0.3465z^{-2} - 0.2125z^{-3} - 0.1131z^{-4} - 0.0075z^{-5} + 0.1468z^{-6}} \quad (4.47)$$

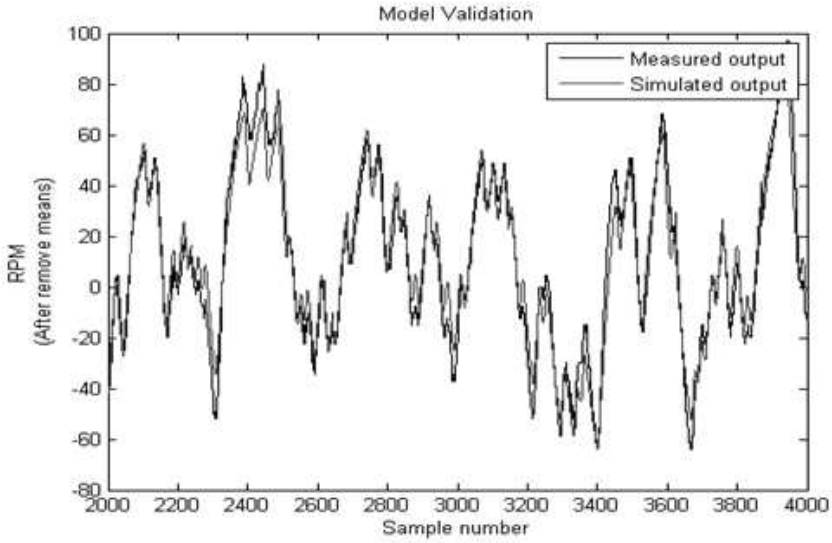

Figure 4.5: The ARX model validation

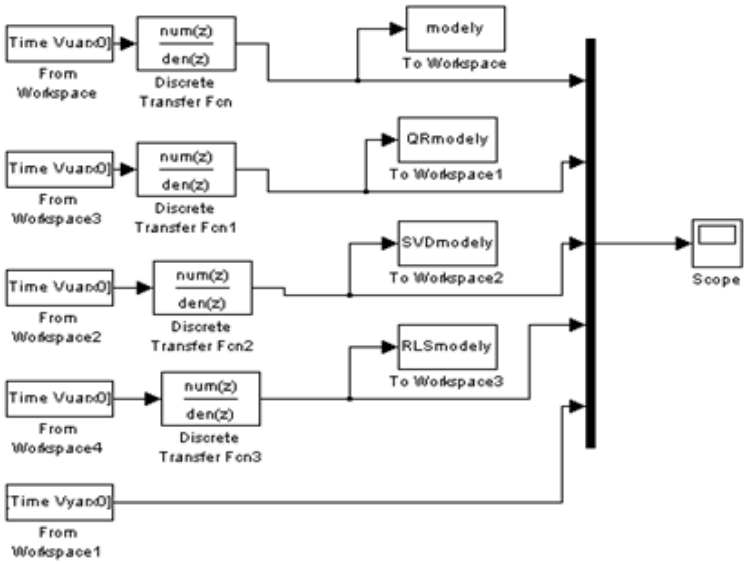

Figure 4.6: The investigation of the models obtained by different algorithms

From the least-squares methodology, the fit ratio of the model is 73.93%, which is very close with the model obtained from the toolbox. Meanwhile, the value of R^2 is calculated as 99.8%. Of course, the linear model is actually just a simplification of the engine system behaviour. In practice, appropriate nonlinear factors should be incorporated into the basic model to give more representative dynamic characteristics.

4.2.3 Nonlinear SISO model

The NARX model structure has been adopted in this application since it is known to be able to well approximate many nonlinear dynamical systems and exhibit a wide range of nonlinear dynamical behaviour [57]. A NARX model can be developed from a ARX model with nonlinear regressors being added subsequently. As illustrated in Figure 4.7, in the NARX structure the next output value is given by the past output (and input) measurements. One of the key NARX model type is the polynomial NARX model which is developed by using algebraic regressor terms and their corresponding parameters.

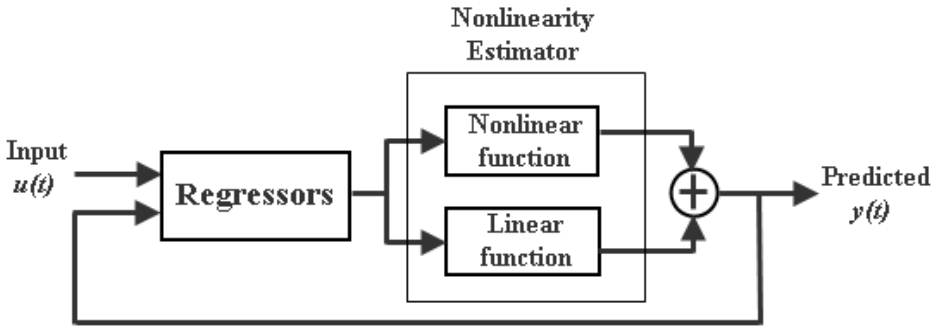

Figure 4.7: The structure of NARX model estimator

For establishing the NARX model, two key problems are required to be tackled in advance:

- Scaling

  Since the ratio of the magnitude of the ABV duty (0.38 0.41) to the rotational speed (around 1000) is approximated 1:2500, the X-matrix formed by the two groups of data vectors may confront the problem of singularity. Hence a scaling strategy is necessary

to be applied to the raw experimental data. The method is to make a scaling vector which based on the mean value of each data vector respectively. Before the identification is started, the regressors (u1, $u1^2$, y1 etc.) are normalized by multiplying the scaling vector such that the values of the data vector are adjusted to a magnitude of approximately 1. At the end of the procedure, the real parametric vector of theta is required to be unscaled by dividing the scaling vector.

- Choice of Regressors

  Another factor which causes the X-matrix become singular is that the regressors are 'too similar'. For preventing this problem, the solution is to add the nonlinear terms into the X-matrix one at a time. This strategy can help to identify which regressor is causing the problem.

Square terms are the most basic nonlinear terms in the NARX model. As part of the trial-and-error strategy, the square terms of the output are inserted into the linear equation. Then, the parameters of the NARX model can be determined by the Least-squares algorithm. The proposed NARX model for test is:

$$\begin{aligned} y(t) = &-4\times10^{-4}y^2(t-1)+3.5\times10^{-5}y^2(t-2)+2\times10^{-4}y^2(t-3)-7\times10^{-4}y^2(t-4)\\ &+1.1\times10^{-3}y^2(t-5)-3\times10^{-4}y^2(t-6)+1.199y(t-1)+0.2742y(t-2)-0.2188y(t-3)\\ &+1.4538y(t-4)-2.2375y(t-5)+0.3696y(t-6)+40.608u(t-6)\\ &+20.4847u(t-7)+22.9675u(t-8)+14.1312u(t-9)+19.6108u(t-10)\\ &+62.4311u(t-11)+23.4279 \end{aligned} \tag{4.48}$$

Based on this polynomial NARX model, a simulink block diagram for the validation of the NARX model can be developed as represented in Figure 4.8.

The quality of the NARX model can be reflected by the residual. Figure 4.9 shows the plot of the residual. It can be seen that the residual of the model is limited between 30 rpm and -10 rpm. The range is around 4% of the mean value of the RPM. The residual value peaks at low speed point of 450rpm and high speed point 1900rpm and this indicates that the model is more accurate in the range between 600 to 1500 rpm. The average of the residual is

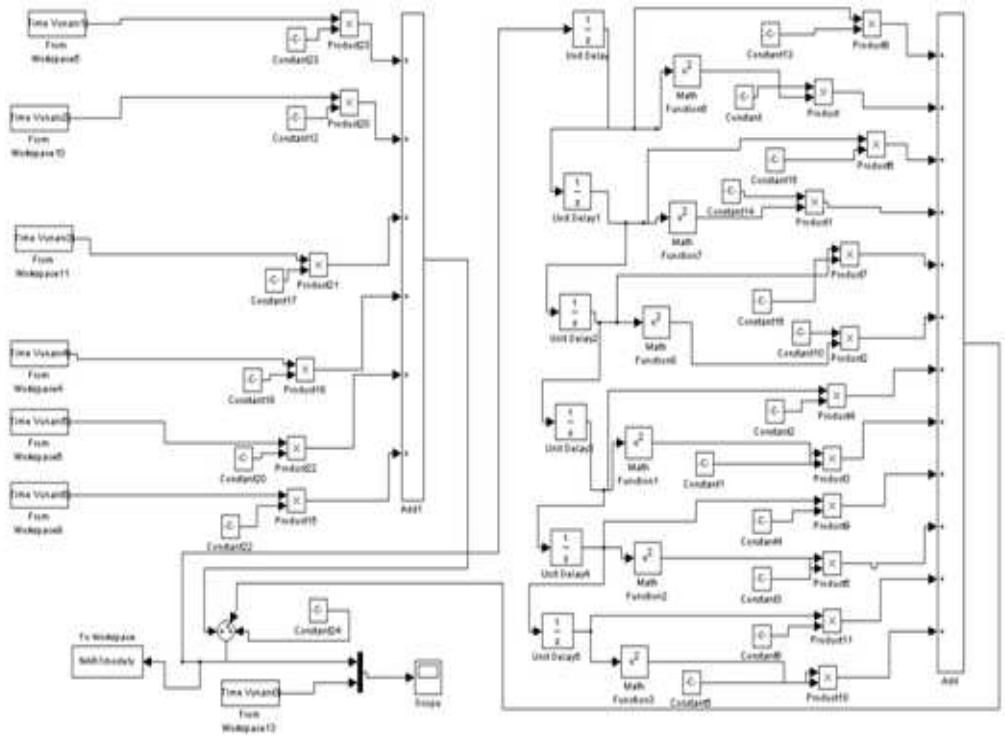

Figure 4.8: The NARX model validation in Simulink

7.5848 which is a positive value. This indicates that the model has somewhat underestimated the output of the engine system. However, it may be acceptable for controller design for the engine system. On the other hand, the result shows that the NARX model fit is improved by 5% compared with the linear model. As a result, it can be seen that the NARX model can represent the engine system more accurate than the ARX model.

4.3 Conclusions

The results presented in this chapter involves linear and nonlinear SISO ABV to engine speed model of the PFI engine. The least-squares technique is adopted to determine the parameters of the models. Both ARX and NARX model of the engine system are obtained.

However, there is no single best model for the engine system. It can be said that the compromise between the complexity of the model and the quality of the model fit is significant. Generally, system identification is carried out as a priori step of controller design. Therefore, the accuracy of the identification model corresponds to the requirements of the controller and

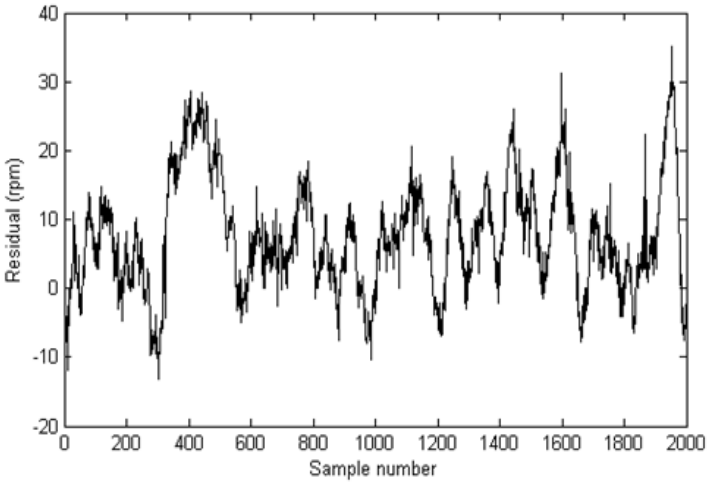

Figure 4.9: The NARX model residual

control design technique.

Based on the results represented in this chapter, it can be concluded that the results of identification of linear ARX models are not sensitive to different LS identification algorithms. Secondly, the square terms of the output RPM have been inserted to construct the NARX model. By adding the nonlinear terms into the ARX model equation, the fit ratio of the model can be improved. The NARX model represented in this chapter has 5% improvement than the ARX model.

Chapter 5

Identification of MISO Dynamic Engine Systems

This chapter mainly introduces two model structure identification techniques including stepwise regression with F-statistic and orthogonal least squares with error reduction ratio. For the purpose of vehicle performance analysis, a MISO engine torque and a MISO air/fuel dynamical model are established by using polynomial NARX model structures. The stepwise regression and the orthogonal least squares methods are both successfully applied in searching for the optimal model regressors.

5.1 Structure Identification Technique

5.1.1 Model Representation

Any structure identification algorithm depends on the nonlinear dynamical model class and the experiment category performed on the unknown process. Most structure identification methods assume that the structure of the system is given a priori. In reality, parameter estimation algorithms and structure identification is usually performed by repeated parameter estimation [20].

In recent years, models describing the engine torque dynamics have been investigated by several researchers: Lee [58] have applied the crankshaft position,angular speed fluctuation and angular acceleration as the measured data to estimate engine in-cylinder pressure which can then be used to derive the indicated torque by using the engine geometry. A schematic single cylinder model has been derived and validated by Filipi and Assanis [59]. Connolly and Rizzoni. [60] have investigated the theory and test results for estimating the torque output generated from individual cylinder in order to achieve real-time estimation. They also emphasised the importance of on-line estimation of engine performance variables for detecting different types of malfunctions. Ingram [61] developed a torque controller based on a steady-state torque function derived from the torque value at different engine working points. The work of Rizzoni and Zhang [62] consists of a detailed physical model and a nonlinear parametric model for a single cylinder IC engine. The indicated torque is determined by minimizing the error ϵ in the estimate of the crankshaft speed.

So far, most of the researchers estimate the engine torque via an interim measured variable such as cylinder pressure and crankshaft angular velocity, in which case, some of the engine nonlinearity may be lost in the identification process. One of the objectives of this thesis is to apply polynomial model structure identification methods for engine torque and air/fuel ratio estimation. In the model structure determination process, the stepwise regression and the orthogonal least squares techniques are employed as regressor selection approaches. The stepwise regression method has been widely utilized in aircraft system identification and is introduced in detail in Klein and Morelli [63] work and practical application of this approach has been made by Cordell and Clayton [64]. In the work described in this chapter, a novel parametric MISO model is identified directly from the measured input/output of the engine and validated by the unseen data using both stepwise regression and orthogonal least squares methods.

In the context of the discrete-time dynamic system, the general NARX(Nonlinear autoregressive with exogenous inputs) model or NARMAX(Nonlinear auto regressive moving average with exogenous inputs) model can be represented by the relationship between its inputs u and outputs y with time delays as shown in Equation 5.1. It has been demonstrated that both NARX model and NARMAX models are capable of describing a very wide form of dynamic

nonlinear behaviour, however, they are linear-in-the-parameters models[65].

$$y(k) = F(y(k-1),...,y(k-p),u(k),...,u(k-q),e(k-1),...,e(k-r)) + e(k) \quad (5.1)$$

It can be seen from Equation 5.1 that the current system response y(k) is determined by its previous inputs/output and noise level. Specifically, p,q and r are known as the order numbers of the model which indicate the maximum number of time-shifted terms in the model. The function F$(\cdot)$ represents the dynamical relationship between the inputs and outputs of the model and can be found by several means if it has a polynomial model structure. $e(k)$ is defined as the noise signal which is generally assumed to be an independent and zero-mean random variable, such as Gaussian distributed noise.

If we assume that e(k)=0, the model is only concerned with the input/output behaviour of these models. The Equation 5.1 is then reduced to

$$y(k) = F(y(k-1),...,y(k-p),u(k),...,u(k-q)) \quad (5.2)$$

In this chapter, the algebraic polynomial NARX model structure is adopted for model development. For linear-in-parameter nonlinear models, several ways for searching the best structure have been introduced in [20]. The MISO modeling process for the torque model starts from a regressor pool, which exhausts all the combinations formed by 1 output and 4 inputs. The terms considered comprises:

Linear terms:

$y(k-1),...y(k-p),u1(k),...,u1(k-q1),u2(k),...,u2(k-q2),u3(k),...,u3(k-q3),u4(k),...,u4(k-q4)$

Quadratic terms:

$y^2(k-1),...y^2(k-p),u_1^2(k),...,u_1^2(k-q1),u_2^2(k),...,u_2^2(k-q2),u_3^2(k),...,u_3^2(k-q3),u_4^2(k),...,u_4^2(k-q4),y(k-1)y(k-2),...,y(k-1)y(k-p),y(k-2)y(k-3),...,y(k-p+1)y(k-p),y(k-1)u_1(k),...,y(k-p)u_1(k-q1),...,y(k-p)u_4(k-q4),...,u_1(k)u_1(k-1),...u_1(k-q1)u_2(k-q2),...,u_4(k-q4+1)u_4(k-q4)$

Cubic terms:

$y^3(k-1), ...y^3(k-p), u_1^3(k), ..., u_1^3(k-q1), u_2^3(k), ..., u_2^3(k-q2), u_3^3(k), ..., u_3^3(k-q3), u_4^3(k), ..., u_4^3(k-q4), y^2(k-1)y(k-2), ..., y^2(k-1)y(k-p), ..., y^2(k-2)y(k-3), ..., y^2(k-p+1)y(k-p), ..., y^2(k-1)u_1(k), ..., y^2(k-p)u_1(k-q1), ..., y^2(k-p)u_4(k-q4), ..., u_1^2(k)u_1(k-1), ...u_1^2(k-q1)u_2(k-q2), ..., u_4^2(k-q4+1)u_4(k-q4), ..., u_1(k-q1)u_2(k-q2)u_3(k-q3), ..., u_4^3(k-q3)$

The range of the candidate regressor terms can be extended according to estimated order of the target system.

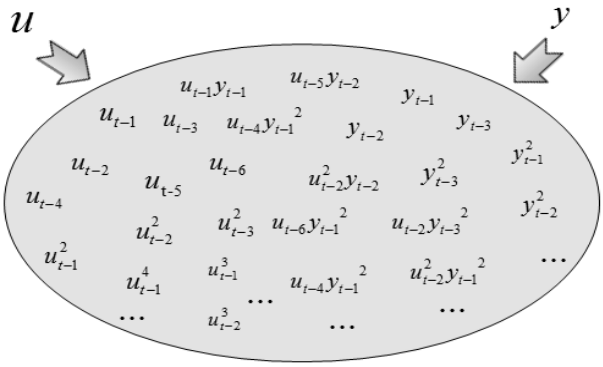

Figure 5.1: Pool of Regressors

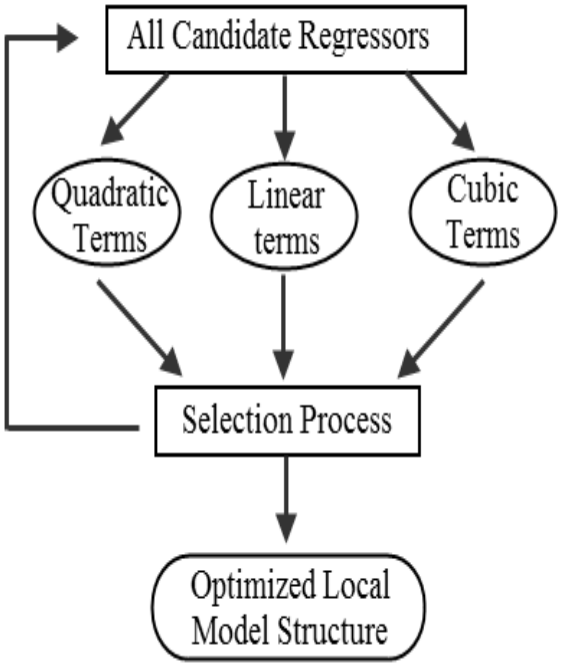

Figure 5.2: Structure Selection Flow Chart

An X matrix can be formed for the selected regressors from the regressor pool, and then

the parameter vector θ can be estimated using ordinary least squares:

$$\hat{\theta} = (X^T \cdot X)^{-1} \cdot X \cdot y \tag{5.3}$$

$$\hat{y} = X \cdot \hat{\theta} \tag{5.4}$$

where $X = [regressor1, regressor2, ...]$, $\theta = [\theta_1, \theta 2, ...]'$ and the regressor vectors are the selected terms from the regressor pool.

5.1.2 Stepwise Regression with F-statistics

The whole regressor selection process runs in iterations. According to the statistical theory, the regressor with the highest correlation factor will have the highest partial F-ratio. With the default offset term in the model and its parameter defined as θ_0, we are able to establish the following two hypotheses:

$$H_0 : \theta_1 = \theta_2 = ... = \theta_n = 0 \tag{5.5}$$

$$H_1 : \theta_j \neq 0 \tag{5.6}$$

where H_0 and H_1 are known as null hypothesis and alternative hypothesis respectively. The alternative hypothesis indicates that at least one j is inserted in the model.The F-statistic analysis of these hypotheses determines whether a new jth regressor is included into the model or not.

In order to decide which hypothesis is true, there are three important statistical quantities which must be evaluated for F-statistic analysis:

$$SS_T = \sum_{n=1}^{N} [y(n) - \bar{y}]^2 = \mathbf{y}^T \mathbf{y} - N\bar{y}^2 \tag{5.7}$$

$$SS_R = \sum_{n=1}^{N} [\hat{y}(n) - \bar{y}]^2 \tag{5.8}$$

$$SS_E = \sum_{n=1}^{N} [y(n) - \hat{y}(n)]^2 = \mathbf{y}^T\mathbf{y} - \hat{\theta}^T\mathbf{X}^T\mathbf{y} \tag{5.9}$$

where N is the number of sample points of the regressor vector and $\bar{y}$ is the mean value of the output variable. SS_T is known as the total sum of squares, SS_R represents the regression sum of squares and SS_E is named as the residue sum of the squares. According to their statistical definitions, the three assessments are related as:

$$SS_T = SS_R + SS_E \tag{5.10}$$

Equation (5.10) can be substituted with the terms from Equation (5.7) (5.8) and (5.9), and thus the following equation is derived:

$$SS_R = \hat{\theta}\mathbf{X}^T\mathbf{y} - N\bar{y}^2 \tag{5.11}$$

With the above statistical measurements, the model structure can be generated by iterations as shown in Figure 5.3.

Correlation Analysis

At the start of each iteration, the proper regressors have to be found based on their correlation with the dependent output variable z. The dependent variable z_i is an iteratively changing variable which is equal to the output variable y at the beginning of the first iteration when $i = 1$. The correlation factor between each regressor and the output variable indicates how significantly the regressor can affect the output variable. It involves three key statistical measurements which are the average value of the regressor $\bar{x_j}$, the covariance of the regressor S_{jj} and the predicted dependent variable S_{zz}. Consider the case when we have M regressors in the pool, the correlation factor between the jth regressor x_j and output z

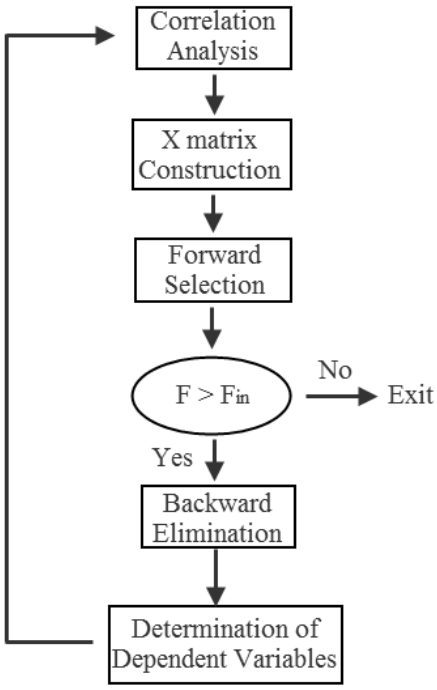

Figure 5.3: Stepwise Regression with F-statistics

can be represented by [63]:

$$r_{i,j} = \sum_{n=1}^{N} \frac{[x_{j(i)}(n) - \bar{x}_{j(i)}][z_i(n) - \bar{z}_i]}{\sqrt{S_{jj(i)} S_{zz(i)}}}, j = 1, 2, 3, ..., M \tag{5.12}$$

where

$$\bar{x}_j = \frac{1}{N} \sum_{n=1}^{N} x_j(n) \tag{5.13}$$

$$S_{jj(i)} = \sum_{n=1}^{N} [x_{j(i)}(n) - \bar{x}_{j(i)}]^2 \tag{5.14}$$

$$S_{zz(i)} = \sum_{n=1}^{N} [z_i(n) - \bar{z}_i]^2 \tag{5.15}$$

Note that z_i is the dependent output vector which has been modified in ith iteration and $z_1 = y$. $\bar{x}_j$ is the mean value of the original regressor vector and $\bar{x}_{j(i)}$ is the mean value of

the jth regressor vector which is modified in the ith iteration. The value of $r_{i,j}$ lies between -1 and 1 and it is an indicator of the degree of linear dependence between two variables. The absolute value of $r_{i,j}$ is known to be closer to 1 if the variables are strongly linearly correlated which means the corresponding term is more suitable in an affine polynomial model. Based on these calculations, we can find the candidate regressor with the largest correlation coefficient in this step.

Formation of Regressor Matrix

The regressor inserted into the model should be the one with the highest correlation. Initially, the regressor matrix X_1 only contains a column of 1s as the offset term:

$$X_1 = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1} \tag{5.16}$$

Then we define x_j as the jth selected regressors and the regressor with the highest correlation factor is added into the X matrix such that:

$$X_{i+1} = [X_i, x_j] \tag{5.17}$$

where i is the iteration number, x_j is the jth regressor selected in the ith iteration.

Subsequently, in terms of assessing the regressors, the forward selection and backward elimination strategies are considered in advance of adding each regressor into the model:

Forward Selection

The partial F-ratio decides the significance of the regressor. In the situation that the model already contains m regressors, the jth regressor can be brought into the model if

$$F = \frac{SS_R(\hat{\theta}_j \mid \hat{\theta}_m)}{s^2} = \frac{SS_R(\hat{\theta}_{m+j}) - SS_R(\hat{\theta}_m)}{s^2} > F_{in} \tag{5.18}$$

where

$$SS_R(\hat{\theta}_{m+j}) = \hat{\theta}_{m+j} * X_{m+j}^T * y - N * \bar{y}^2 \tag{5.19}$$

$SS_R(\hat{\theta}_{m+j})$ is the regression sum of squares obtained by adding the jth regressor to the original m terms. The index $m + j$ means that the jth regressor is inserted into the model of previous iteration which contains m regressors. F_{in} is a predefined F-distribution threshold which is normally set as less than 4 if the confidence level of the selected term is required to be higher than 95%.

Backward Elimination

The regressors already entered into the model are reassessed from their partial F-ratios in each iteration, since a regressor added in the model at the early stage may become redundant when it involves some relationship with the regressors added subsequently. With a model that already involves p regressors, the jth regressor with the lowest partial F-ratio is eliminated if

$$F = \min_j \frac{SS_R(\hat{\theta}_p) - SS_R(\hat{\theta}_{p-j})}{s^2} < F_{out} \tag{5.20}$$

Where

$$SS_R(\hat{\theta}_{p-j}) = \hat{\theta}_{p-j} * X_{p-j}^T * y - N * \bar{y}^2 \tag{5.21}$$

$SS_R(\hat{\theta}_{p-j})$ is the regression sum of squares obtained by removing the jth regressor from the p terms which are already in the model. The index $p - j$ represents that the jth regressor is removed from the model of previous iteration with p regressors. F_{out} is the F-distribution threshold which is normally defined as equal or a little lower than F_{in} which indicates it is equally hard or a little easier to eliminate a term in the model than to select one into it.

Determination of Dependent Variables

The dependent output variable and regressors will be modified corresponding to the regressors already in the model, i.e., in the ith iteration, the dependent variables are altered as

$$z_{i+1} = z_i - \hat{\theta} * X_i \tag{5.22}$$

In terms of offsetting the influence of the regressors already in the model for the next iteration, the unselected regressors in the pool are modified to

$$x_{j(i+1)} = x_{j(i)} - \hat{\beta} * X_i, j = 1, 2, 3... \tag{5.23}$$

where

$$\hat{\beta} = (X_i^T * X_i)^{-1} * X_i * x_{j(i)}, j = 1, 2, 3... \tag{5.24}$$

At the end of the iteration, i increases 1 for next iteration.

The iteration from correlation analysis stage to modification of dependent variables stage continues until no other candidate regressor in the regressor pool possesses a partial F-ratio higher then F_{in} and no regressor in the model has the partial F-ratio less then F_{out}. F_{in} and F_{out} are the pre-selected stopping criterion for the iteration. Essentially, at 95 percent confidence level, we use the criterion $F(0.05, 1, N - m) \approx 4$. Normally, $F_{in} = F_{out}$, however, $F_{in} > F_{out}$ indicates it's harder to accept a regressor than delete one. As a result, all the significant terms in the regressor pool are found and inserted into the model through the iteration process.

5.1.3 Regression Analysis Using Orthogonal Model Components

The forward orthogonal least squares(OLS) algorithm with the error reduction ratio(ERR) analysis was originally introduced to determine which terms should be included in the model [66]. This approach is suitable for determining the significant terms in the linear-in-the-parameters models. The ith error reduction ratio, ERR_i, introduced by the ith orthogonalized regressor, can be defined as [67]:

$$ERR_i = \frac{g_i^2 < x_i, x_i >}{< Y, Y >} = \frac{< Y, x_i >^2}{< Y, Y >< x_i, x_i >} \tag{5.25}$$

where ERR_i should vary between 0 and 1, $< Y, Y >= Y^T \cdot Y$.

The structure detection can be executed iteratively as shown in Figure 5.4. The steps in this algorithm are:

Step 1: all the candidate regressors are examined by the ERR criterion in this step. The regressor which maximises the ERR_i will be chosen as the optimal regressor and added into X matrix in this iteration.

Step 2: all the remaining candidate regressors are orthogonalized by a Gram-Schmidt transformation. According to [20], the regressor orthogonalization is determined by:

$$\hat{x}_j(k) = x_j(k) - \sum_{i=1}^{j-1} \frac{\sum_{k=1}^{N} x_j(k)\hat{x}_i(k)}{\sum_{k=1}^{N} \hat{x}_i^2(k)} \hat{x}_i(k) \tag{5.26}$$

where $x_j(k)$ is the jth non-orthogonal regressor and $\hat{x}_i(k), i = 1, 2, ..., (j-1)$ are the already orthogonalized regressors. In this step, a predefined threshold $\hat{x}_j^T \hat{x}_j >= \tau$ should be satisfied otherwise the candidate regressor should be deleted from the regressor pool to avoid numerical ill conditioning. Here τ is varied from the magnitude of 10^{-10} to 1 according to the practical application of the model [67].

The loop formed by ERR calculation and regressor orthogonalization will be repeated

until the sum of the ERR of all selected regressors (devoted as SERR) meets $1-\sum_{i=1}^{M_{in}} ERR_i < \rho$ where M_{in} is the number of selected regessors and ρ represents the error tolerance limit. The calculation of parameter vector $\Theta^{(OLS)}$ of the X matrix is described in [67].

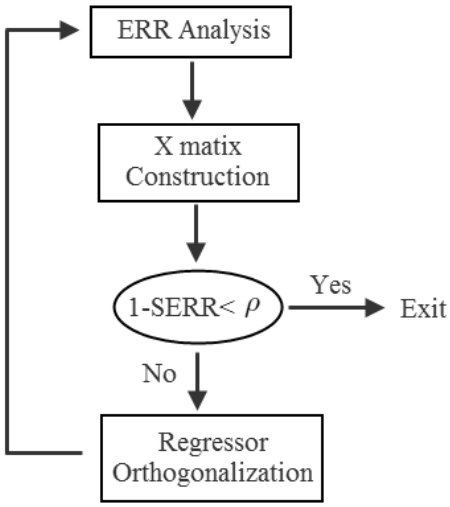

Figure 5.4: Orthogonal Least Squares with ERR detection

5.1.4 Analysis of Variance (ANOVA)

The Analysis of Variance (ANOVA) method is a well-known structure selection technique for the identification of non-linear dynamical black box models. It has been a general tool for quality control and medicine analysis. The basic principles of ANOVA are founded on the hypothesis tests with F-distribution variables derived from the residual quadratic sum [68]. In recent years, several novel tests of ANOVA in [69, 70] were successfully conducted on nonlinear finite impulse response (NFIR) model types whose outputs depends solely on current and past inputs. Further structured utilization of ANOVA was given in [71]. The ANOVA method assumes all the sample data can be described as:

$$y_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \epsilon_{ijk} \tag{5.27}$$

where μ is the overall mean and ϵ_{ijk} are the independent Gaussian noise with $(0, \sigma^2)$ distribution. Based on this assumption, the sampled data and the regressors are quantised into

influence levels. For example, $\tau_i,\ i = 1, ..., p$ is the ith influence factor for regressor $\phi_1(t)$ and $\beta_j,\ j = 1, ..., q$ is the jth influence factor for regressor $\phi_2(t)$. The interaction between the regressors are further denoted as $(\tau\beta)_{ij}$.

The assumption of two regressors can be expanded into an axis-orthogonal regressor space. An ANOVA function expansion is proposed in [72] using piecewise constant basis functions:

$$\begin{aligned}\rho(\mathbf{c},\theta,\phi) = c_0\theta_0 + \sum_{k=1}^{d} c_k(\sum_{i_1=1}^{m_k} \theta_{k;i_1}\mathbf{I}_{b_{(k,i_1)}}(\phi_k)) \\ + \sum_{k=1}^{d-1}\sum_{l=k+1}^{d} c_{k,l}(\sum_{i_1=1}^{m_k}\sum_{i_2=1}^{m_l} \theta_{k,l;i_1,i_2}\mathbf{I}_{b_{(k,i_1)}}(\phi_k)\mathbf{I}_{b_{(l,i_2)}}(\phi_l)) \\ + \cdots + c_{1,2,\ldots,d}(\sum_{i_1=1}^{m_1}\sum_{i_2=1}^{m_2}\cdots\sum_{i_d=1}^{m_d} \theta_{1,2,\ldots,d;i_1,i_2,\ldots,i_d} \times \prod_{k=1}^{d}\mathbf{I}_{b_{(k,i_k)}}(\phi_k))\end{aligned} \tag{5.28}$$

In Equation 5.28, $b_{k,i}$ represents the ith interval linked with the regressor ϕ_k . Meanwhile, $\mathbf{I}_b(u) = 1$ if $u \in\ b$ and $\mathbf{I}_b(u) = 0$ if $u \notin\ b$. The parameter vector $\mathbf{c}$ decides whether an elemental sum is included, so it only has the value 0 or 1. The term $c_0\theta_0$ is the overall mean value which is independent of all regressors, the main effects of the regressors are the first summation and the interaction effects are expressed in the second summation. The interaction degree of the effect is defined as the number of regressors involved in an interaction effect. Every effect has its own parameter in vector $\mathbf{c}$ but can contain multiple parameters in θ for each basis function.

$\rho(\mathbf{c},\theta,\phi(t))$ can then be derived as a linear-in-parameters model, the structure selection objective can be described as:

$$L(\mathbf{c},\theta) = \sum_{t=1}^{N}(y(t) - \rho(\mathbf{c},\theta,\phi(t)))^2 \tag{5.29}$$

The estimated $\hat{\theta}$ should minimise $L(\mathbf{1},\theta)$ and then be optimized by solving

$$min_{\mathbf{c}}L(\mathbf{c},\hat{\theta}) + J\left\|\mathbf{Fc}\right\|_1 \tag{5.30}$$

where $\mathbf{c} \in 0, 1$. The penalty function J and $\mathbf{F}$ are calculated from L($\mathbf{1}$,$\hat{\theta}$) together with the

statistical F-tables. Due to the highly over-parameterised model structure, linear constraints are required to guarantee identifiability. These constraints can be expressed as:

$$M\theta = 0 \tag{5.31}$$

where the detailed content of the M matrix are defined in [73].

5.2 PFI Engine Modelling

5.2.1 Experiment Data

The 1.6V Ford Zetec Engine was used for obtaining the Input/Output data. In order to build a MISO model for the engine torque response, five I/O channels have been adopted. The directly measured variables include ABV (Air Bleed Valve) Duty, FPW (Fuel Pulse Width), RPM and AFR (Air/Fuel Ratio) with the output torque measured by a dynamometer coupled directly the engine crankshaft. The delayed linear regressors are shown in Table 5.1. The experiment data sets are shown in Figure 5.5, with an angular resolution of 360 degrees of crank angle (2 data points per engine cycle). In Total, 2500 data samples were collected.

5.2.2 Estimation of Torque and Air/Fuel Ratio

The MISO models are identified via stepwise regression method using correlation analysis selection(CORR) and error reduction selection(ERR) respectively. For torque model identification, the regressor selection process of the first 7 regressors is represented in Table 5.2. The values in the table are the largest partial F-ratio for the candidate regressors in each iteration. As shown in Table 5.2, the values in the table indicates that the partial F-ratio is varying along with iterations. Meanwhile, the most significant regressors will maintain the highest F-ratio in each iteration. The regressor with the highest partial F-ratio is selected from the regressor pool in each iteration. In this application, the partial F-ratio thresholds are defined as F_{in}=5.5, F_{out}=4. On the other hand, in order to keep the term number within a reasonable range for pursuing the calculation of error reduction ratio, the threshold τ is

defined as 1 , which is higher than the threshold given in [67]. After the candidate regressors are selected, the corresponding parameters are determined using least squares estimation. The MISO models for torque and air/fuel ratio are given in Table 5.3 and Table 5.4 respectively. For a fair comparison, identical numbers of iterations are applied in the CORR and ERR approach.

5.2.3 Model Validation

The first 1000 samples of measurement data are utilized for identifying the model structure, while the latter 1000 samples are considered as the unseen data and prepared for cross-validation. The results obtained from two different methods are compared in Figure 5.6 and Figure 5.7, including torque and air/fuel ratio prediction results. The model quality is estimated by the following three criterion:

1. R^2 Multiple Correlation Coefficient

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{k=1}^{N}[y(k)-\hat{y}(k)]^2}{\sum_{k=1}^{N}[y(k)-\bar{y}]^2} \tag{5.32}$$

   The model is considered better with larger R^2 $(0 < R^2 < 1)$.

2. Mallow's C_p statistic

$$C_p = \frac{\sum_{k=1}^{N}[y(k)-\hat{y}(k)]^2}{\hat{\sigma}^2} - (N - 2n_\theta) \tag{5.33}$$

   where n_θ represents the degrees of freedom for the regression or the number of the regressors and $\hat{\sigma}^2$ represents the variance of the source noise which is approximated by

$$\hat{\sigma} \approx min(\frac{1}{N}\sum_{k=1}^{N}[y(k)-\hat{y}(k)]^2) \tag{5.34}$$

   In terms of model analsis, the model is accepted when C_p is the closest with the number of parameters in the model and the model with the same C_p value but least parameters should be selected.

3. FPE: The Prediction Error Technique [55]

$$FPE = N \cdot \ ln[\frac{1}{N}\sum_{k=1}^{N}[y(k) - \hat{y}(k)]^2] + N \cdot \ ln[\frac{N + n_\theta}{N - n_\theta}] \tag{5.35}$$

Therefore, lower FPE value indicates better model quality.

The results in Table 5.5 shows that both error reduction ratio(ERR) method and correlation(CORR) analysis approach can develop the torque and air/fuel ratio model with good prediction ability (higher R^2 value), however F-statitic with $CORR$ estimation gives better predictive torque model. Meanwhile, the model quality of the two methods get closer when the number of regressor increases.

Table 5.1: Input and Output Channels

Measured Variables	Notation	Delays
ABV	$u1$	$u1(t-1), u1(t-2), ..., u1(t-9)$
FPW	$u2$	$u2(t-1), u2(t-2), ..., u2(t-9)$
RPM	$u3$	$u3(t-1), u3(t-2), ..., u3(t-9)$
AFR	$u4$	$u4(t-1), u4(t-2), ..., u4(t-9)$
TRQ	$y1$	$y1(t-1), y1(t-2), ..., y1(t-5)$

Measured Variables	Notation	Delays
ABV	$u1$	$u1(t-1), u1(t-2), ..., u1(t-9)$
FPW	$u2$	$u2(t-1), u2(t-2), ..., u2(t-9)$
RPM	$u3$	$u3(t-1), u3(t-2), ..., u3(t-9)$
AFR	$y2$	$y2(t-1), y2(t-2), ..., y2(t-5)$

Table 5.2: Stepwise Regression Using F-statistics (Reg: Regressor number)

Iter	Reg1	Reg2	Reg3	Reg4	Reg5	Reg6	Reg7
1	379350						
2	6678.7	665.0					
3	6075.4	502.4	68.7				
4	6047.5	491.5	75.3	30.6			
5	4541.3	466.2	71.0	30.5	5.9		
6	4485.5	472.1	77.3	36.2	8.8	6.3	
7	5600.4	448.9	28.6	39.4	44.2	36.1	34.6

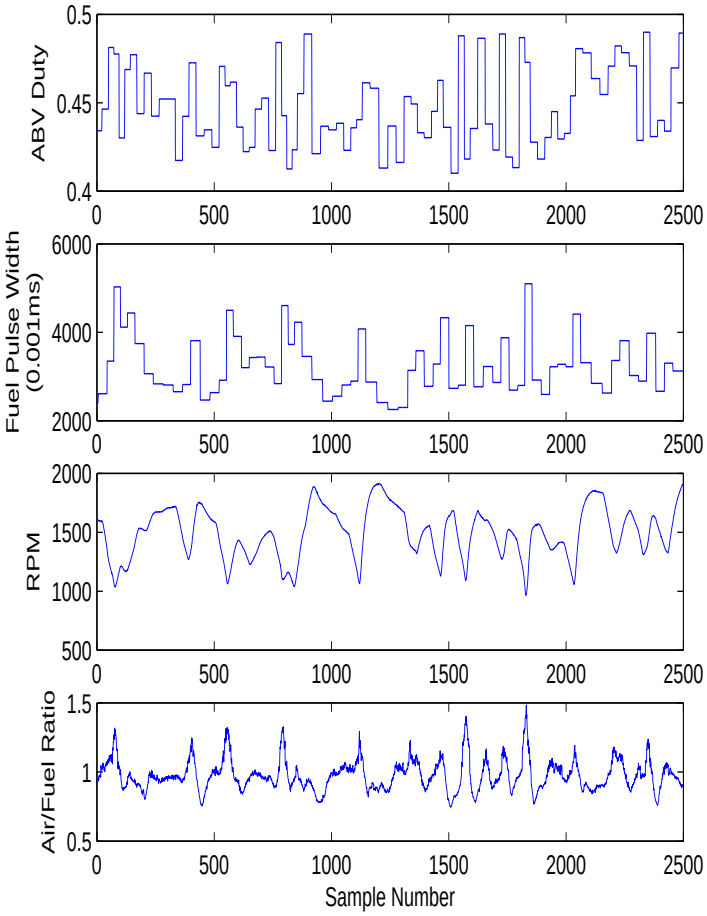

Figure 5.5: Input Channels

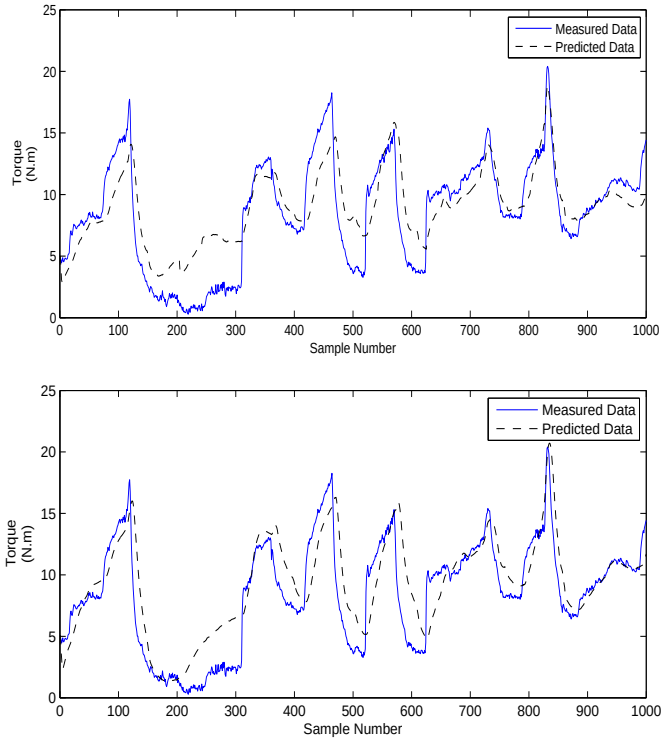

Figure 5.6: Prediction of torque using correlation criteria and ERR criteria

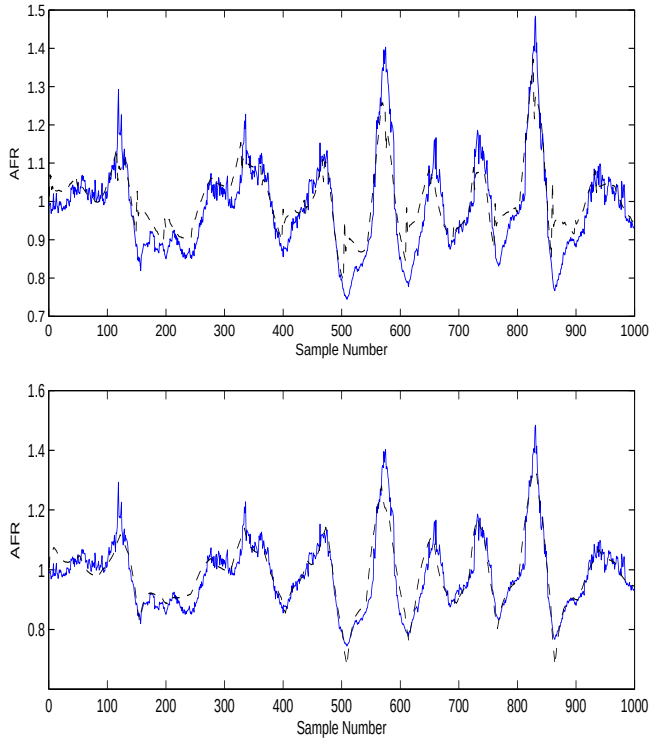

Figure 5.7: Prediction of air/fuel ratio using correlation criteria and ERR criteria

5.3 Conclusion

Due to the complexity of the engine, nonlinear models are able to predict the engine characteristics more accurately than linear models. It can be concluded that both stepwise regression and orthogonal least squares(OLS) techniques can be applied to the torque and air/fuel ratio model identification. In terms of the time-domain discrete system identification, both methods are able to develop parametric model structures efficiently. In practice, the directly identified model may have the capability to bypass various engine geometry problems and integrate the engine nonlinearities via the model regressor selection process. As shown in the validation results, the model prediction ability of the two methods gets closer with increasing model term number. Meanwhile, the programming and calculation process of the OLS approach is less time-consuming than that of stepwise regression method and the OLS approach can achieve better prediction performance when fewer parameters are required in the model.

Table 5.3: Regressers and Parameters in the Torque Model

$\theta(CORR)$	Regressor($CORR$)
0.3827	constant offset
1.0711	$y1(t-1)$
-0.1062	$y1(t-2)$
-0.0423	$u3(t-1) * u3(t-2)^2$
6.0809	$u3(t-2) * u1(t-2) * u1(t-2)$
-7.3791	$u3(t-3) * u1(t-1) * u1(t-1)$
0.0071	$u1(t-2) * u2(t-2) * u4(t-3)$
13.3492	$u1(t-1) * u1(t-2) * u3(t-3)$
-5.9645	$u1(t-2) * u1(t-2) * u3(t-3)$
7.4953	$u1(t-1) * u1(t-1) * u3(t-2)$
-13.5404	$u1(t-2) * u1(t-1) * u3(t-2)$
-0.0002	$u2(t-1) * u4(t-3)$
-0.0065	$u1(t-1) * u2(t-2) * u4(t-3)$
-8.1077e-05	$u2(t-6) * u4(t-2) * u4(t-3)$
6.3294e-08	$u2(t-1) * u3(t-9) * y1(t-1)$
-5.9159e-08	$y1(t-2) * u2(t-1) * u3(t-8)$
$\theta(ERR)$	Regressor(ERR)
0.4607	constant offset
1.1762	$y1(t-1)$
-0.2849	$y1(t-2)$
0.9959	$u1(t-3) * u1(t-2)^2$
0.0419	$u1(t-1) * y1(t-1)$
1.5668e-05	$u3(t-3) * u4(t-5) * y1(t-1)$
0.0002	$u2(t-6)$
-0.0096	$u3(t-1)$
0.0094	$u3(t-2)$
9.0540e-09	$u2(t-1) * u3(t-9) * y1(t-1)$
-6.4347e-06	$u2(t-6) * u3(t-1)$
5.4901e-06	$u2(t-6) * u3(t-2)$
-1.2049e-06	$u2(t-6) * u3(t-3)$
1.9621e-06	$u2(t-6) * u3(t-4)$
7.1609e-06	$u2(t-5) * u3(t-1)$
-7.1039e-06	$u2(t-5) * u3(t-2)$

Table 5.4: Regressers and parameters in the Air/Fuel Ratio Model

$\theta(CORR)$	Regressor($CORR$)
-0.1628	constant offset
1.2792	$y2(t-1)$
-4.9611e-08	$u3(t-4)*u2(t-4)*y2(t-1)$
0.2408	$y2(t-2)*u1(t-1)*y2(t-3)$
-0.2154	$y2(t-1)*y2(t-1)$
1.6116e-08	$u2(t-2)*u3(t-9)*y2(t-5)$
-1.1484e-08	$y(t-1)*u2(t-6)*u3(t-4)$
1.2062e-07	$u2(t-4)*u3(t-1)$
-8.1273e-08	$u2(t-2)*u3(t-4)$
0.1553	$u1(t-3)*y2(t-2)$
-1.7930e-07	$u2(t-4)*u1(t-1)*u3(t-2)$
1.5379e-07	$u1(t-1)*u2(t-2)*u3(t-5)$
$\theta(ERR)$	Regressor(ERR)
0.0249	constant offset
0.8785	$y2(t-1)$
0.2033	$u1(t-1)*y2(t-1)$
0.6381	$u1(t-1)^3$
-9.2037e-09	$y2(t-1)*u2(t-1)*u3(t-9)$
-1.2922e-07	$u2(t-1)*u1(t-1)*u3(t-2)$
5.7844e-08	$u1(t-1)*u2(t-1)*u3(t-4)$
-9.1311e-09	$u1(t-1)*u2(t-3)*u3(t-1)$
4.5637e-08	$u2(t-1)*u3(t-1)$
2.9951e-08	$u2(t-8)*u3(t-1)$
1.9999e-09	$u2(t-7)*u3(t-1)$
-4.3731e-08	$u2(t-8)*u3(t-2)$

Table 5.5: Model Prediction Quality

Torque model	R^2	C_p	FPE
CORR	67.6	15	8750
ERR	66.8	15	8770
AFR model	R^2	C_p	FPE
CORR	73.8	11	1361
ERR	86.5	11	696

Chapter 6

Multi-Model Identification

6.1 Introduction

Multiple models have been applied in several branches of science and engineering independently in recent years. Multiple model methods are being investigated by several researchers to address the increasing complexity of control and modelling problem can be caused by advances in information technology, and increases in environmental constraints and the economical regulations [5]. For highly complex systems, precision and significance tend to become mutually exclusive properties along with the increasing complexity [74]. Therefore, it is a good approach to disintegrate the system into segments, which is also known as the divide-and-conquer strategy. As shown in Figure 6.1, the multi-model structure is defined by the local models' structures and the weighting function which determines the relationship among local models and the overall output. Depending on the practical objective, different multi-models have their corresponding inherit advantages. For instance, the static mean value models possess straightforward structure and fast output response [75, 76] while the recurrent neural network models are superior in learning and predicting the dynamic nonlinearity [77, 78].

The entire operating range of the system is partitioned into multiple regimes which are attached with corresponding local models or controllers. Hence, a complex nonlinear model can be simplified and become more transparent. Figure 6.2 represents a input/output

scheduling multi-model structure, each of the local model has it own weighting function to decide its fraction of influence on the output. Meanwhile, the weighting function is determined by the operating regime partition and selected scheduling variables and algorithms. The multi-models also reduce the computational requirement of the digital devices, since they incorporate adapting and learning mechanisms which enable the use of prior qualitative knowledge and keep the dimension of regime scheduling in a reasonable range if not to a minimum. Various iterative multi-model approaches have been proposed by researchers, including Piecewise models, Tagaki-Sugeno fuzzy models [79], Local linear model tree (LoLiMoT), Local model networks, Operating regime based models etc. The neural network models of cylinder pressure are represented in [80, 81]. In the area of gasoline engine, a series of neural network models for air/fuel ratio were established in [82, 83, 84].

In this chapter, the procedure of the multi-model identification is discussed, including input signal design and optimization, choice of sheduling variables, data partitioning, weighting functions and local model identification. The LOLIMOT algorithm for developing the multi-model network is investigated and an engine multi-model identified by LOLIMOT algorithm is presented. Based on the procedure discussed in this chapter, an engine torque and air/fuel ratio multi-model is developed with MIMO affine local models.

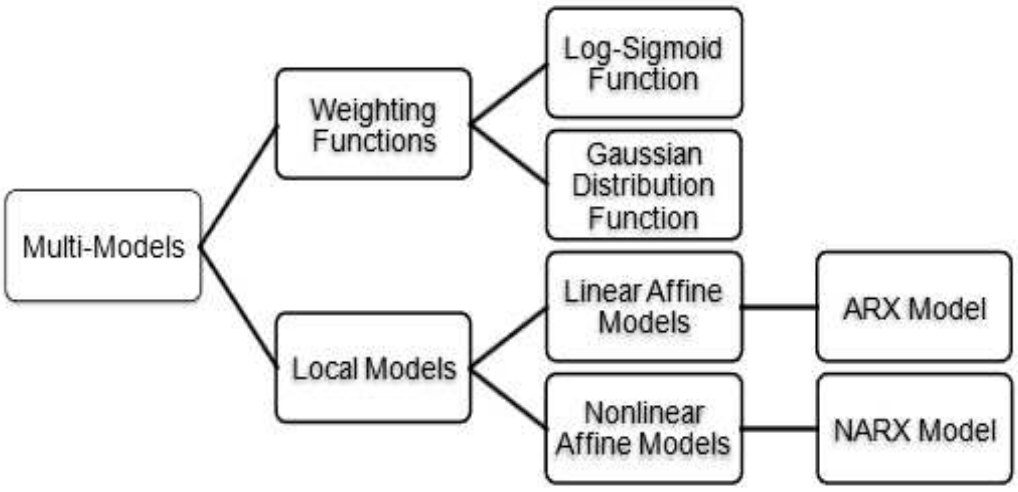

Figure 6.1: The taxonomy of multi-models

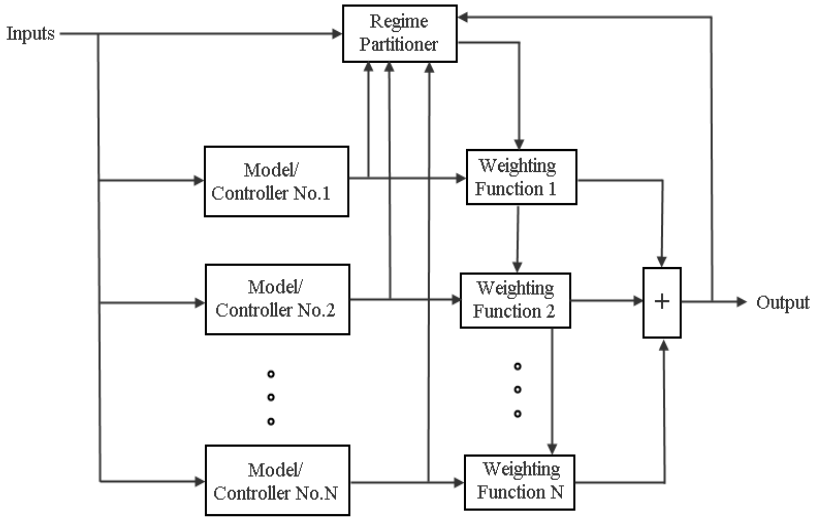

Figure 6.2: The generic multi-model structure

6.2 General Procedure of Multi-Model Identification

6.2.1 Input Signal Design and Optimization

Input signal design is the initial essential step for model identification. For Identification of MIMO and SISO models, the step test signal is frequently applied to perturb the inputs [85]. Good input test signals can broadly excite dynamic nonlinear behaviour of the engine and produce accurate and reliable experimental results.

At the same time, in a multi-variable dynamic process, the input and output signals are also required to be restrained within certain boundaries in order to keep the engine operating smoothly and safely. As introduced by Vinson and Georgakis [86], the available input/output space is bounded by the assigned range of input/output variables. The fundamental principles of optimal input design and objective functions are detailed by Mehra [87] and Goodwin [88]. The main current approaches signal optimization include A-optimal [89] and G-optimal [90], and these techniques have been successfully applied on engine dynamic models in Zaglauer and Delflorian's work [91]. It can be noted that the optimal signal design is always based on the assumption that a dynamic model is available at hand ahead of the experiment design. Consequently, an initial dynamic model of the engine is needed and this can be obtained from data-based structure selection techniques. Usually, the various series of amplitude modulated

pseudo-random binary signals(APRBS) are used as initial test signals because of their widely spanned frequency spectrum which can span the expected system dynamics [92, 93]. For APRBS signals, the input optimization process is conducted by adjusting the APRBS signals into an optimized form which best suits the identified system.

6.2.2 Choice of Scheduling Variables

In a multi-model structure, the scheduling variables define how the data set is to be partitioned. If two variables are adopted to divide the data set, the scheduling space is 2-dimensional. These variables are the key properties of the multi-model because the local models are developed on these data segments defined by value space of the scheduling variables. Consequently, the scheduling variables should reflect the major factors affecting the related output, such as brake torque and emissions. Both input and output variables of the system can be used as the scheduling variables depending on the modeling priority and accuracy.

6.2.3 Data Partitioning

Data partitioning is a significant process for determining which data segments and characteristics are utilized at certain data site. The data set collected from the system is filtered and sampled before entering the partitioning process. The use of data segments can be better maintained and accessed and facilitate the efficient use of multi-modeling and improves the quality of the local models. The overall data set can be collected through the whole operating region and then break into segments according to a pre-defined feature or boundary, namely the Gaussian distribution center or average value center. On the other hand, the data set can be collected on different operating ranges and the local models are identified based on their corresponding data sets. Moreover, the programming strategies of data partitioning are discussed [94, 95, 96].

6.2.4 Weighting Function

The local models identified from different data segments requires a feasible weighting function to be integrated into a global model. The weighting function decides which local model is activated and to what extent it will affect the global model. Therefore, the weighting functions are also called membership functions. Generally, the following weighting function types are applicable in multi-models:

Gaussian Function

Gaussian functions are widely used in industrial quality control and experiment design. The Gaussian function has the form:

$$\rho(x) = f(x, c, \sigma^2) = \frac{1}{\sqrt{2\pi}} e^{\frac{-(x-c)^2}{2\sigma^2}} \tag{6.1}$$

where the symmetric bell-shape function is defined by the centre c and the standard deviation σ. As shown in Figure 6.3, the centre is decided by the middle point of the alleged data partition and σ is either predefined or calculated via the data cluster. However, the Guassian function is conveniently normalized by Equation 6.2 into the range of $[0, 1]$ thus keeping the overall sum as 1.

$$\phi(x) = \frac{\rho_i(x, c, \sigma^2)}{\sum_{j=1}^{M} \rho_j(x, c, \sigma^2)} \tag{6.2}$$

Piecewise Constant Function

The piecewise function creates a unit step change at the boundary between two distinct area thus only one local model will be activated at a time. The expression of the function is of the form:

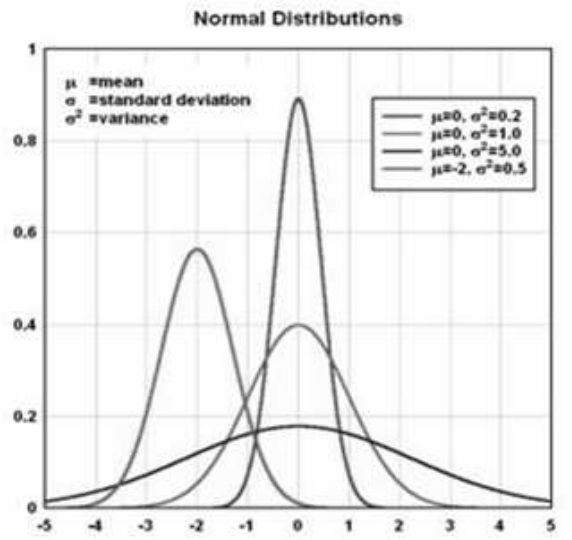

Figure 6.3: Gaussian bell curves

$$\rho(x-\beta) = \left\{ \begin{array}{ll} 0 & if \quad x-\beta \leq 0 \\ 1 & if \quad x-\beta > 0 \end{array} \right\} \tag{6.3}$$

where β is the displacement from the base point. The piecewise weighting function has the simplest structure of possible weighting functions and requires the lowest calculation time and fully employs the individual local model. However, the piecewise weighting function ignores the interconnection among the local models, which could cause non-smooth and inaccurate nonlinear behaviour.

Log-sigmoid Function

An obvious disadvantage of the piecewise function is that it is non-smooth at the border between different data regimes. To avoid this problem, the log-sigmoid function offers a smooth transition from one partition to another. The general expression of the log-sigmoid function is given by:

$$\rho(x) = f(x; \alpha, \beta) = \frac{1}{1 + e^{-\alpha(x-\beta)}} \tag{6.4}$$

where α determines the steepness and β determines the displacement of the curve. As shown in Fig 6.4, the sigmoid function has a value ranged from 0 to 1 and so is automatically normalized.

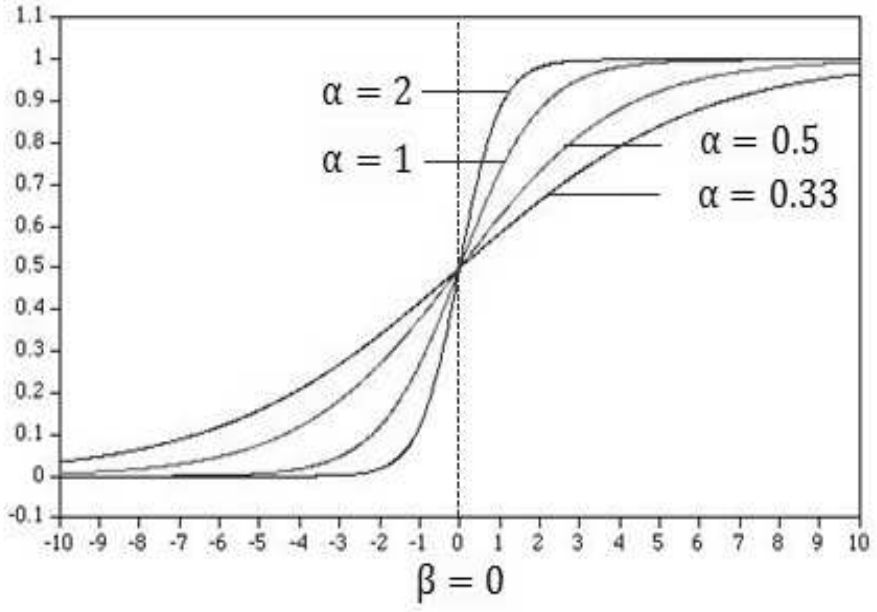

Figure 6.4: Log-sigmoid curves

For piecewise weighting where membership function only switch between 0 and 1, the 'switch' effect occurs at the border of neighbouring regions. however, for Gaussian and log-sigmoid membership functions, the local models are scheduling by their corresponding membership weighting, so in certain valid region, one or several local model output may become dominant and others may fade away. Therefore, the Gaussian and log-sigmoid functions are able to create a smooth transition region rather than a 'switch' point.

6.2.5 Local Models

In a multi-model structure, the local models determine the local accuracy and the computational time. Using polynomial models, the local model can use fewer terms than for a global model because the dynamics represented by a limited data partition is much less varied than that represented by all data. In the automotive industry, the empirical and data driven model types are already recognized as an economically feasible and accurate approach for producing static models. With the tightening fuel efficiency and emission legislation, nonlinear dynamic models are gaining momentum in applications to model-based calibration and optimization [97]. Structure selection techniques which can be applied in identifying both linear and nonlinear local models are consequently of great interest for use in local model structure identification.

ARX and NARX Models

The AutoRegressive model with eXternal inputs (ARX) and Nonlinear AutoRegressive model (NARX) with eXternal inputs are described in [14] and [65]. Generally, the inputs and outputs are modulated by a series of unit delays in the local subsystems. As demonstrated in Figure 6.5, the ARX and NARX model are readily built as block diagram structures such as in the Matlab S-function blocks.

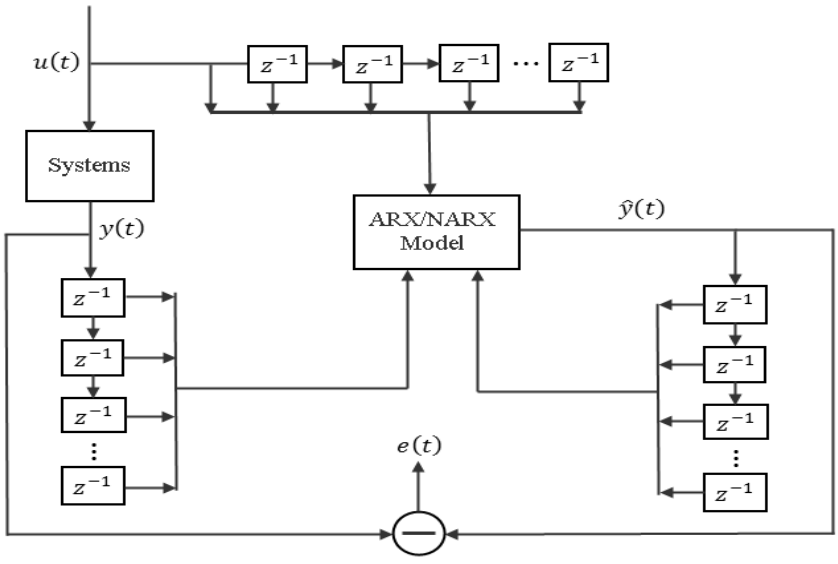

Figure 6.5: ARX/NARX model as block diagram with tapped time delays

LPV Models

The linear parameter varying (LPV) systems are defined as a class of linear time-varying systems and the LPV modelling and control is a well-established technique in engineering area. LPV approaches have been applied in various nonlinear identification. The basic principles of LPV model are well documented in [98]. A turbo fan engine model appliation identified by LPV technique was presented in [99].

6.3 One Dimensional Multi-modelling

In this section, a one dimensional scheduled multi-model is developed for a 4 cylinder Zetec 1.6L PFI gasoline engine. MIMO linear affine and nonlinear affine models are adopted as local models which can be developed using multiple inputs and outputs (u1: air bleed valve duty, u2: manifold air pressure, u3: engine speed, u4: fuel pulse width, y1: torque, y2: air/fuel ratio). The partition of the local multi-models can be fixed a priori or estimated along with the local models [100]. In this application, for a comparison exercise, we define 3 partitions which are centred by the local models. The local models are selected on engine speed, centred on 4 different speed points, 1150 rpm, 1350 rpm, 1550 rpm, 1750 rpm. Two groups of local affine models are identified by testing the whole range of the nonlinear system using the stepwise regression and orthogonal least squares respectively. The multi-affine local models with different number of linear regressors are analyzed. As is shown in Figure 6.6, the model fitness decreases when over 5 regressors (20 multiplications) are added into the local models. This phenomenon can be caused by overfitting as too many linear regressors are involved in the model. Figure 6.7 is obtained if we consider the total multiplication numbers of the 4 local models where each linear regressor creates 1 multiplication in each local model.

In order to improve the model quality, nonlinear regressors, which are able to track the nonlinearity of the system to some extent, are added into the model. The structure selection programme starts from linear terms, then goes to quadratic and cubic terms. The selection of quadratic terms is based on the linear terms and the selection of cubic terms is based on the previous selected linear and quadratic terms. For instance, the optimum quadratic terms

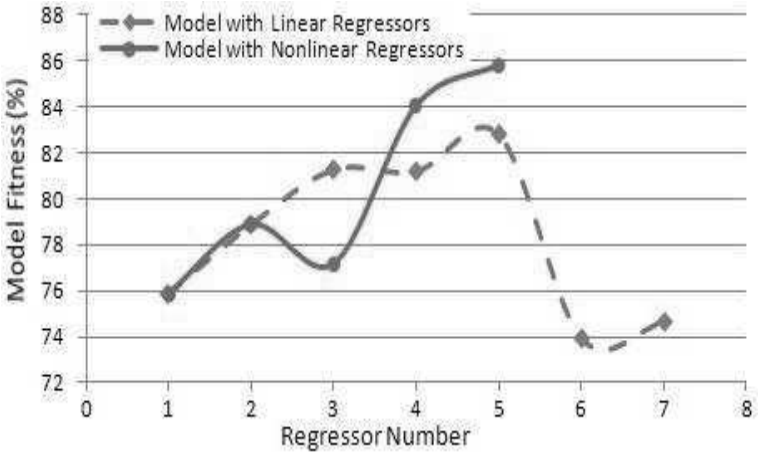

Figure 6.6: Relations between model fitness and regressor number

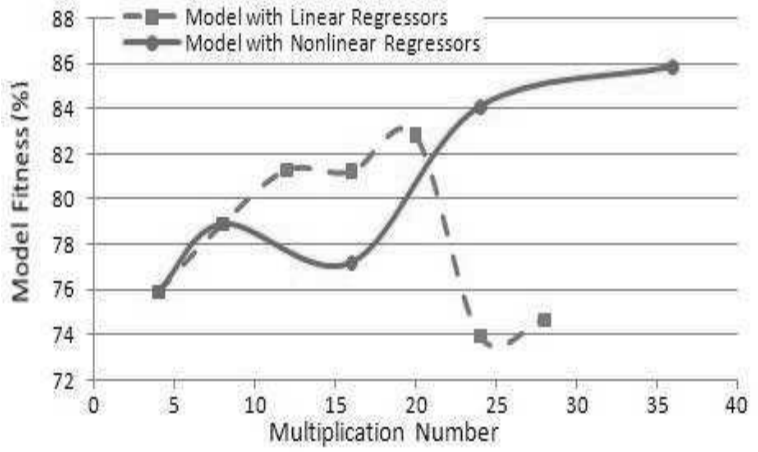

Figure 6.7: Relations between model fitness and multiplication number

are determined based on the optimum linear terms via the dependent variable (the output variable that is modified by the regressors already in the model). So it becomes a kind of relaxed version of structure optimization. On the other hand, the best number of linear, quadratic and cubic terms in the model can be determined empirically by using the fitness vs regressor or fitness vs multiplication curve located by the peak points of fitness. In the CORR method, every added regressors (linear and nonlinear term) will then be assessed by a partial F-ratio in the combined model which will make sure the regressor selected in each category is also a significant term for the combined model. However, the nonlinear model requires more multiplications which costs more computation time. As shown in Figure 6.7, the global model's fitness keeps on rising when the multiplication number increases to 36. It indicates that the potential of the nonlinear multi-affine models can be developed if a good model structure is chosen. Therefore, a compromise needs to be made between computational cost and model quality. An example of structure identification of a local model is given by Table 6.1 which gives the CORR and ERR values for the nonlinear structure selection process for a torque model at 1750rpm. For overall comparison, the global models for torque and air/fuel ratio using nonlinear affine local models are shown in Table 6.2.

Table 6.1: CORR and ERR Structure Selection Process at 1750rpm

Iter.1	2	3	4	5
CORR				
0.9899	0.3377	0.3886	0.2686	0.1814
y1(t-1)	u3(t-1)	u3(t-2)* u3(t-3)	u1(t-1)* y1(t-3)	u4(t-1)*u3(t-1)* u3(t-3)
F-ratio				
48840				
33555	128			
36304	252	177		
1629.6	321.6	232.9	77.4	
1275.0	268.5	203.4	88.7	33.8
ERR				
0.9994	0.0158	3.0e-4	1.55e-5	9.93e-6
y1(t-1)	y1(t-3)	u2(t-1)* u4(t-1)	u4(t-1)* u4(t-1)	u4(t-5)*u1(t-5)* u4(t-5)

The multi-model validity region for this study is 1150-1750 rpm however the region is extendable when more local models beyond this range are established. The weighting function

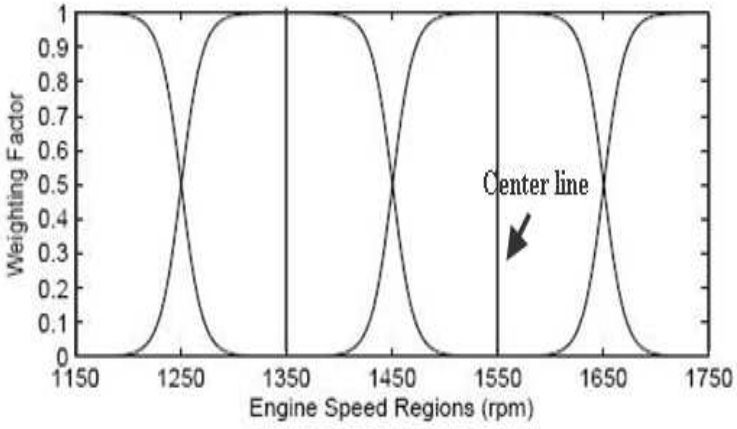

Figure 6.8: Log-Sigmoid weighting function

is developed based on the symmetric log-sigmoid function:

$$\rho(x) = \left\{ \begin{array}{l} \frac{1}{1+e^{-\alpha(200-x)}}, x < 100 \ (right\ half) \\ \\ \frac{1}{1+e^{-\alpha(x)}}, x \geq 100 \ \ (left\ half) \end{array} \right\}$$

where α determines the convergence speed to 0 or 1. In the tests reported, we employ α=0.10 and x= engine speed - central speed of the corresponding speed partition. In order to determine the weighting factors of different local models, a uniform random speed signal is generated to range across the validity region and applied to the multi-affine model. The form of these weighting functions is shown in Figure 6.8. The convergence parameter α defines the saturation speed of the log-sigmoid function. An increase in the value of α will result in faster convergence, in which case the weighting factors in the middle of each speed regions will be affected more by the central local model. In the application we have taken the distance between two mid-points of the neighbouring speed regions as 200rpm. The weighting function switches among local models whilst the sum of the weighting factors remains 1 at any point of the valid range. The weighting function thus converts the separate local linear models into a continuous nonlinear model. With our weightings, the nonlinear model in the speed region is effectively determined only by its immediately adjacent local models; the output of the model at 1250 rpm is affected by the local models at 1150 rpm and 1350 rpm.

For model validation and analysis, a neural network model is developed using the MATLAB Neural Network Toolbox. The model parameters are composed from 2 input weighting

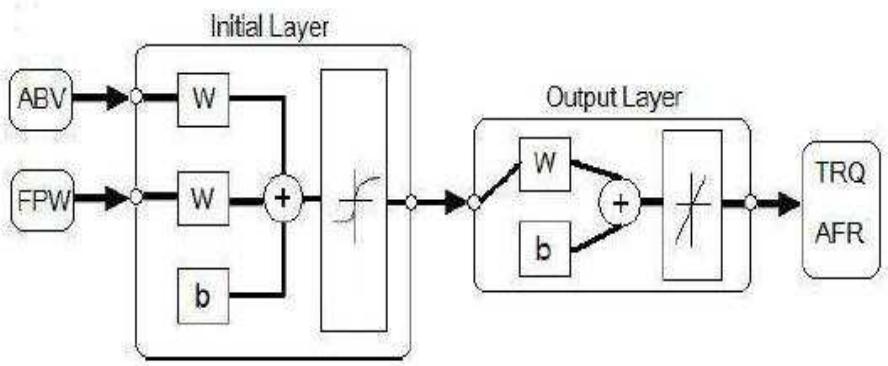

Figure 6.9: The infrastructure of the neural network model

matrices and 2 bias vectors in the initial layer, and 1 layer for the weighting matrix in the output layer as shown in Figure 6.9. Two inputs, the engine air bleed valve duty (ABV) and fuel pulse width (FPW), are used to determine the weighting matrices. Finally, the outputs of torque and air/fuel ratio are predicted separately but by the same structure of neural network layer.

Model validation requires the application of unseen experimental data to the model. For this validation, the engine torque and air/fuel ratio are chosen as output variables. The estimated results from the models are ploted along with 1000 engine speed data samples shown in Figure 6.10. Figure 6.11 and Figure 6.12 show the torque estimation results, Figure 6.13 and Figure 6.14 show the air/fuel ratio estimation results.

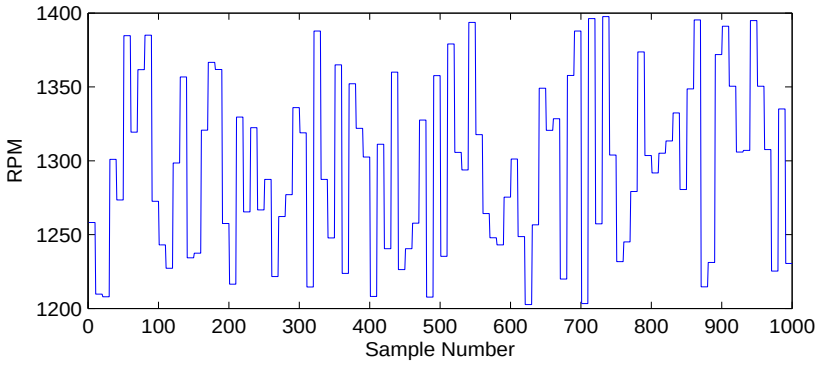

Figure 6.10: The test data of engine speed

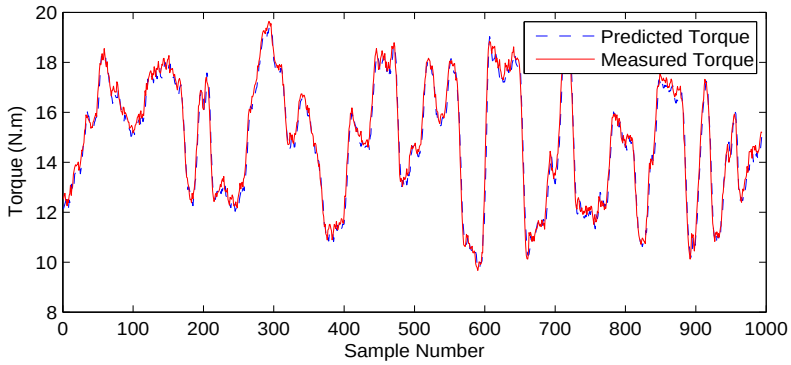

Figure 6.11: Torque Estimation using Nonlinear Multi-Affine Model

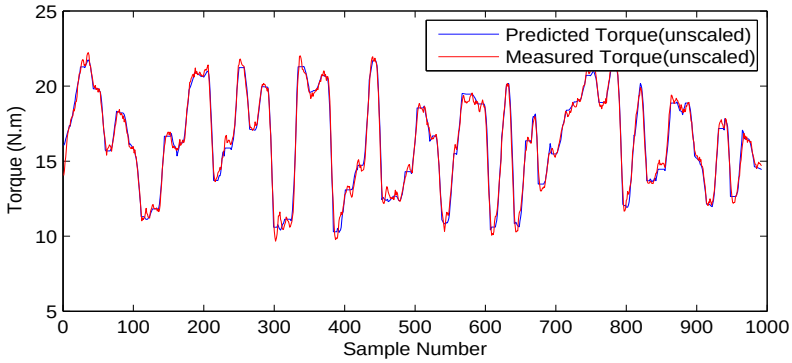

Figure 6.12: Torque Estimation using Neural Network Model

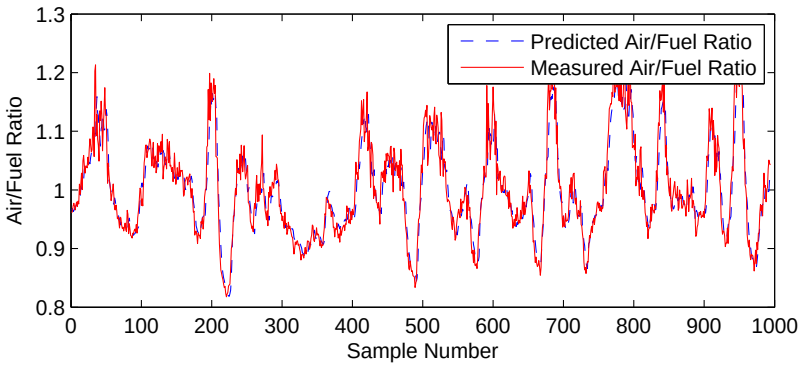

Figure 6.13: Air/fuel ratio Estimation using Nonlinear Multi-Affine Model

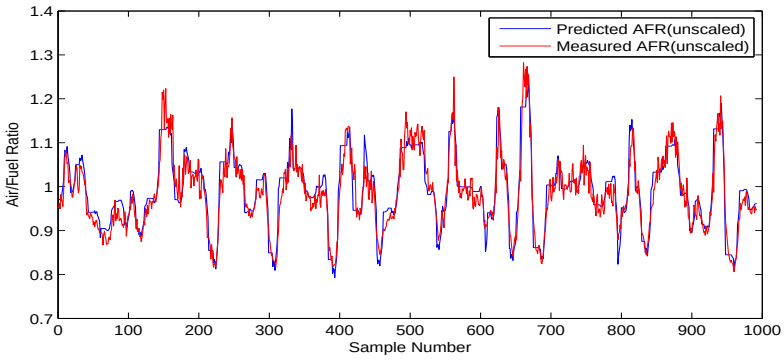

Figure 6.14: Air/fuel ratio estimation using neural network model

6.4 LOLIMOT Identification

6.4.1 Definition of LOLIMOT algorithm

The LOLIMOT (Local Linear Model Tree) algorithm was developed by Oliver Nelles [101] to identify a local model network which best fits an input and output data set. For nonlinear dynamic systems with unknown properties, the LOLIMOT algorithm offers a feasible way to cope with complex data-based identification [102]. The LOLIMOT model can also be recognized as a fast learning neural network model and in this context has been applied to an emission model of an internal combustion engine [103]. The LOLIMOT algorithm is able to generate a model structure equivalent to Sugeno-Takagi fuzzy system with a local model network, however, the LOLIMOT algorithm is easier to implement and is faster in training [104]. As shown in Figure 6.15, at each iteration, two new local models are created and the one with worse fitting capacity or model accuracy subjects to division in the next step. From the point of view of engine system identification, the algorithm starts with identifying an overall global linear model; then the data sets from the engine are split into two segments within the engine operating range. Within respective segment, a local model is identified with a least squares algorithm and the output of this model is properly weighted and accumulated into the global output. This means each of the local models in the branches has a fair share to determine the output of the global tree. In the next iteration, the model with the lowest fitness to the data is divided into two sub-models. The process continues while the model tree is growing and new refined model structures will be obtained until a pre-defined threshold on fitness is reached. Finally, the overall LOLIMOT model output is the accumulation of all

the neurons which are formed by the local model and the corresponding weighting functions as illustrated in Figure 6.17.

The LOLIMOT algorithm can be executed iteratively as shown in Figure 6.16. However, this flow chart also reveals that this algorithm still has its limitations. It only considers the best choice of local model level in each iteration and the influence newly selected local models exercise on the global model is ignored. Therefore the final result is not optimal because once a decision is made in an iteration, it is not reversible and we have no idea if the fitness of the global model is compromised by previous decisions. Secondly, as long as the complexity of the local model networks is largely determined by the scheduling dimensions of the hypercuboid space and the dividing strategy, the partitioning process of 2-D or more dimensional scheduling space can be very time-consuming. To avoid the curse of dimensionality, usually the local regimes are decided by distribution of data clusters or just by an equal-division strategy which is called binary tree planning or dyadic partitioning. The membership function for the LOLIMOT algorithm is derived from the n dimensional Gaussian function of the form:

$$\mu_j(\mathbf{X}) = exp(-0.5 \times (\frac{(\mathbf{x}_1 - c_{1j})^2}{\sigma_{1j}^2} + \frac{(\mathbf{x}_2 - c_{2j})^2}{\sigma_{2j}^2} + \frac{(\mathbf{x}_3 - c_{3j})^2}{\sigma_{3j}^2} + \cdots + \frac{(\mathbf{x}_n - c_{nj})^2}{\sigma_{nj}^2})) \quad (6.5)$$

where $\mu_j(\mathbf{X})$ represents the membership function of the j-th local model in scheduling space X, $\mathbf{x}_n$ represents the n-th scheduling variable of the membership function, c_{nj} is the expected mean value for the n-th scheduling variable of the j-th local model and σ_{nj}^2 is the corresponding variance.

The membership function indicates the significance of a local model considering the n dimensional space of the scheduling variables and the overall validity/weighting function for the i-th local region is determined as:

$$\phi(X, c_i, \sigma_i) = \frac{\mu_i(X)}{\sum_{j=1}^{M} \mu_j(X)} \quad (6.6)$$

It can be seen that the weighting functions are able to emphasise different local models

depending on the selecting criterion to reproduce nonlinear characteristics of the engine.

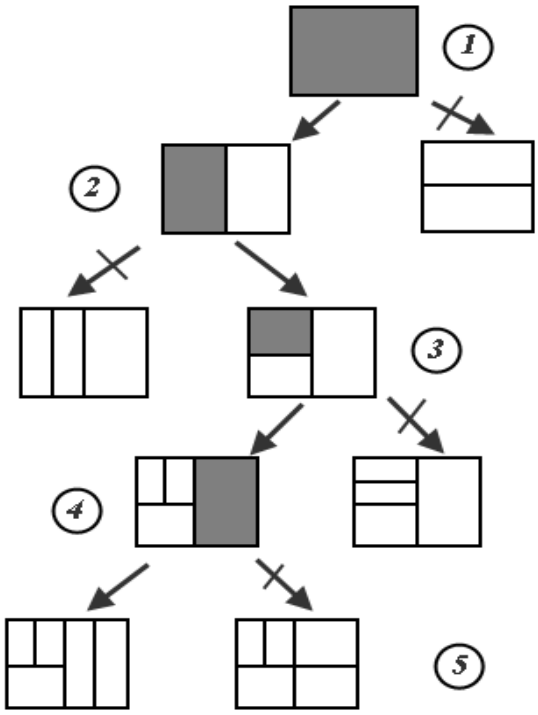

Figure 6.15: 2D partitioning of LOLIMOT algorithm

6.4.2 Engine MISO Identification using LOLIMOT strategy

The data sets shown in Figure 6.18 were collected from the 4 cylinder Zetec 1.6L PFI gasoline engine in the Powertrain Control Laboratory of University of Liverpool for the purposes of LOLIMOT identification.

Subsequently, the engine data was normalized into the range of $[0, 1]$ by Equation 6.7 in order to facilitate the partitioning and training process.

$$y_{normalized} = \frac{y - y_{min}}{y_{max} - y_{min}} \tag{6.7}$$

In this application, the engine speed N and the air/fuel ratio(λ) L are selected as scheduling variables. The model quality is calculated by the normalized mean square error (NMSE)

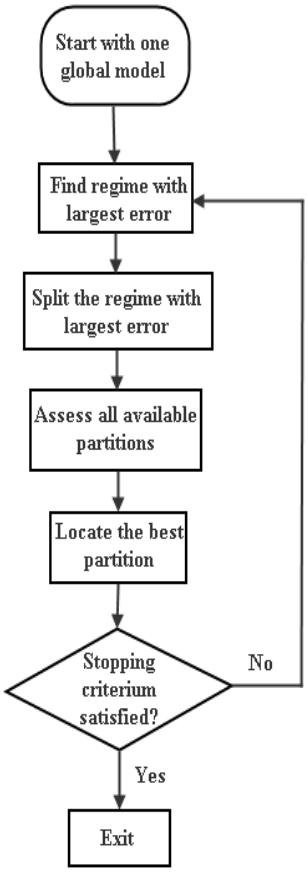

Figure 6.16: Flow chart of the LOLIMOT algorithm

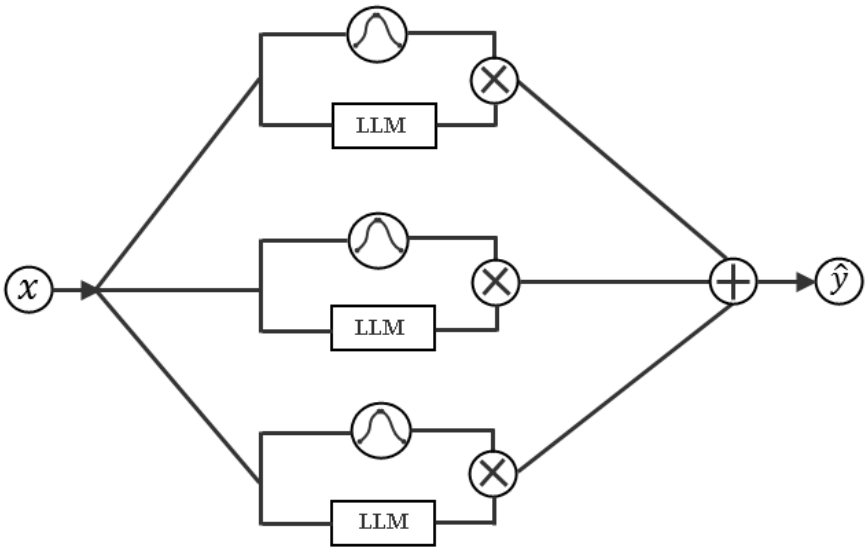

Figure 6.17: The LOLIMOT structure as a neural network

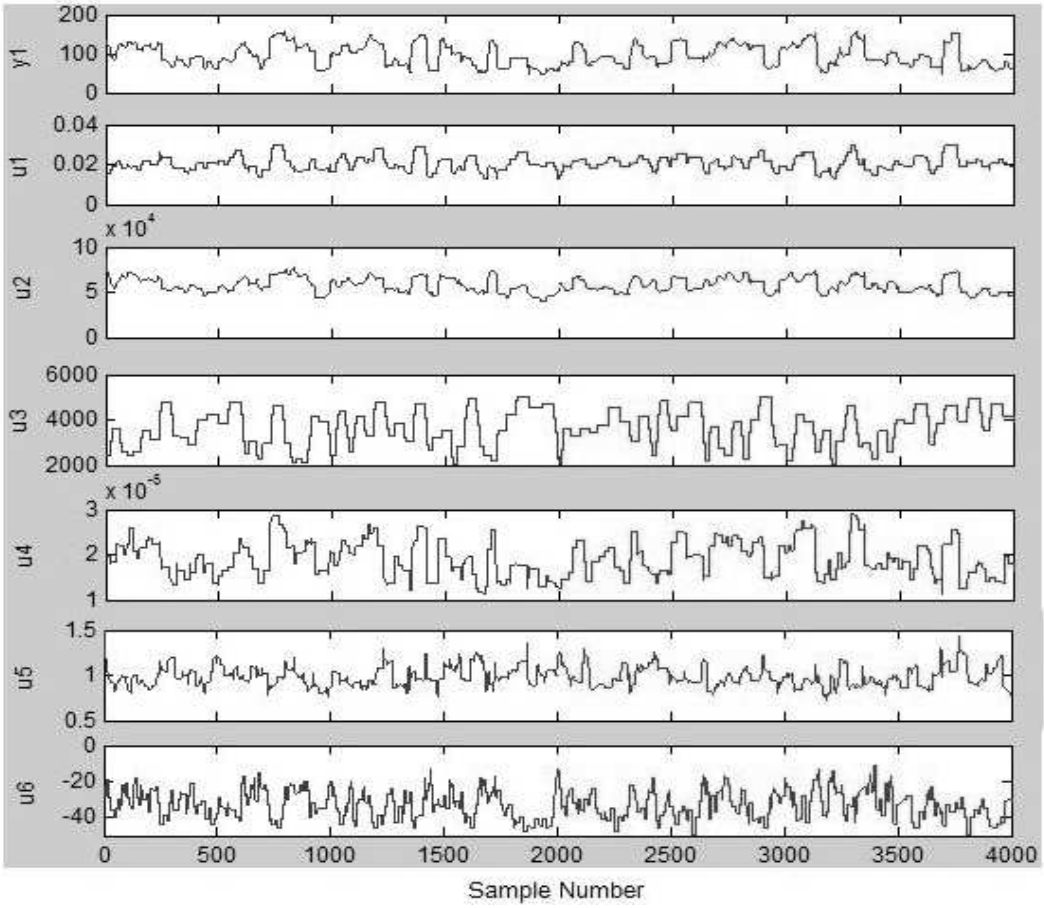

Figure 6.18: Test data sets for LOLIMOT identification

which is defined as:

$$NMSE(y) = \frac{Var(y - \hat{y})}{Var(y)} \tag{6.8}$$

where the variance is defined as

$$Var(X) = \frac{1}{N}\sum_{i=1}^{N}(X(i) - \bar{X})^2 \tag{6.9}$$

Therefore, the 2-D partitioning process is that shown in Figure 6.19. Starting from the first iteration, each local model has its own normalized mean square error (NMSE) and the local model with highest NMSE will be separated into two local models in each iteration.

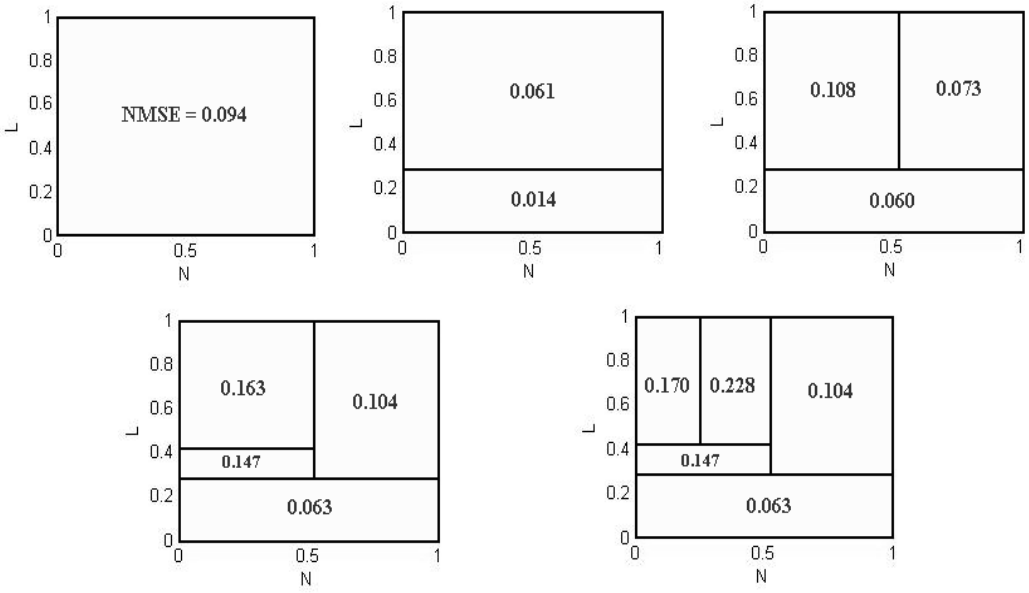

Figure 6.19: Data partitioning process (N: Engine speed L: Lambda)

The Gaussian curves of the weighting functions for the five local models are plotted in Figure 6.20. It can be seen that the weighting function curves for smaller data regions are steeper.

The overall validation of the LOLIMOT model is shown in Figure 6.21. Based on

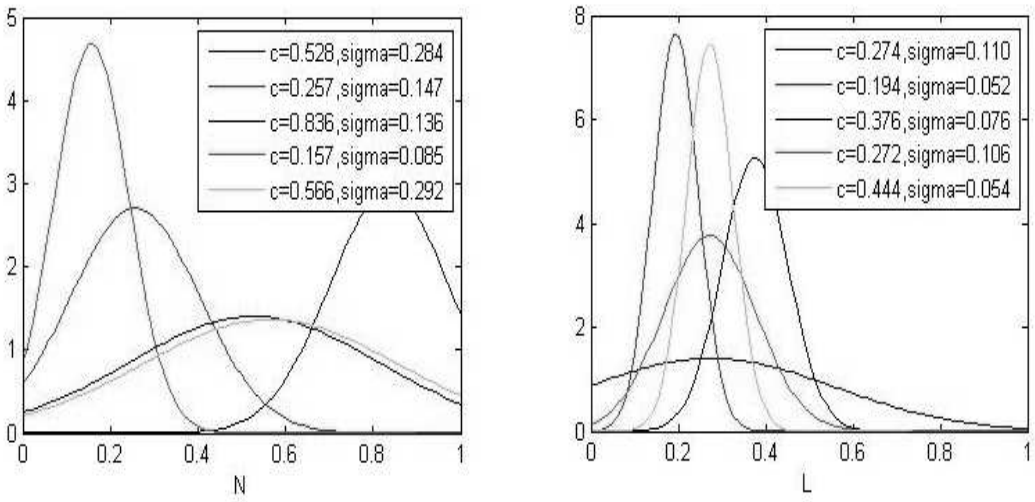

Figure 6.20: Weighting function curves

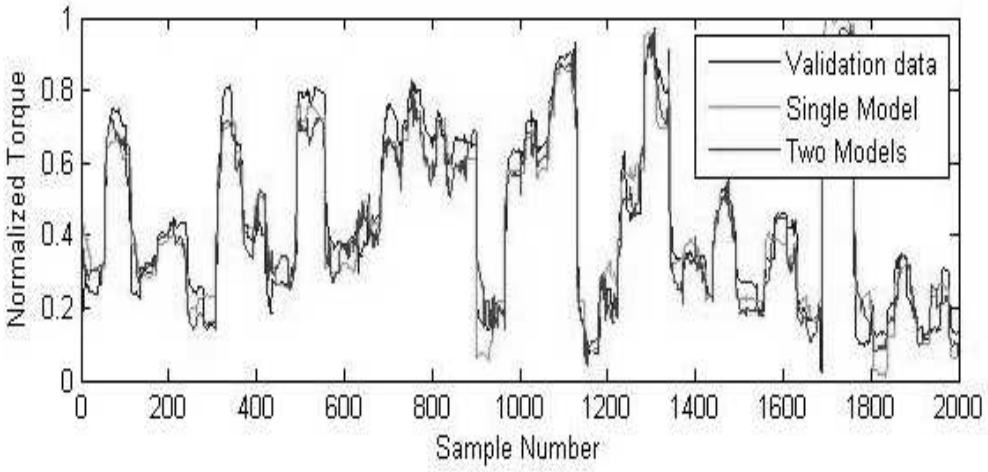

Figure 6.21: Validation of the LOLIMOT model

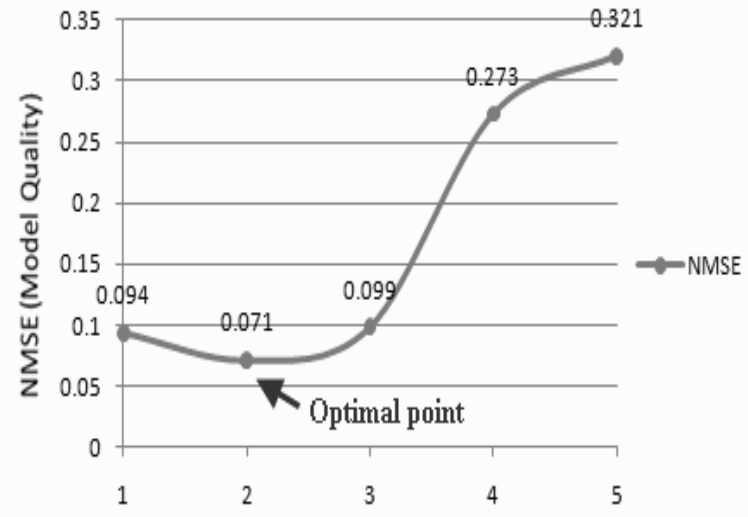

Figure 6.22: Local model number analysis

the validation results, an analysis of the local model numbers was carried out as shown in Figure 6.22. According to the NMSE assessment, two overall local models are found to be the optimal number of model partitions.

6.5 Conclusion

Multi-models are being considered as an efficient and understandable modelling approach to achieve high accuracy dynamic models and filters. Their ease of implementation makes them highly suited to applications within ECU software. Nevertheless to ensure the full benefits of the approach, it is necessary to ensure that partitioned domains for the different models and thus where one model becomes more weighted relative to the others are determined properly. This issue has received little previous attention. In this chapter, a one-dimensional scheduling algorithm and a multi-dimensional LOLIMOT algorithm has been used for establishing a multi-model network for a 4 cylinder 1.6L PFI Zetec engine. Structure identification methods have been applied to develop the local models. A forward and an inverse identified multi-model has been developed for the WAVE-RT virtual engine.

The multi-model is able to represent complex nonlinear behaviour yet retains a simple implementable structure for EMS engine mappings. Both stepwise regression and orthogonal least squares (OLS) techniques can be applied successfully to the torque and Air/Fuel

ratio model identification. Based on the results represented in this chapter, the multi-model structures is proposed as an alternative black box dynamic approach for the case of non-linear engine map and a systematic method of developing engine mappings can be applied to the engine calibration process which can save significant amount of time. The relationship between model fitness and likely EMS computational overhead as measured by number of multiplications were investigated. The results showed an appropriate trade-off relation between the computational cost and the model quality can be established by the structure identification techniques. In the case of the multi-models, the associated local models can be developed from the engine experiments conveniently at different operating points and we can apply linear identification and control techniques to the linear local models which is a significant advantage.

Table 6.2: Local Linear Models for Torque(y1)and Air/Fuel Ratio(y2)

local models	$\theta(CORR)$	$Reg(CORR)$	$\theta(ERR)$	$Reg(ERR)$
y1	-2.8677	offset	-0.8392	offset
@1150rpm	0.852	y1(t-1)	1.1181	y1(t-1)
	8.0822	u1(t-2)	-0.2287	y1(t-3)
	-0.0009	u3(t-5)y1(t-3)	6.8513	u1(t-1)u1(t-1)
	0.0009	u3(t-1)y1(t-3)	2.75e-8	u4(t-1)u4(t-1)
	0.0011	u4(t-1)u1(t-1) u1(t-1)	8.8549	u1(t-3)u1(t-3) u1(t-5)
@1350rpm	-3.26	offset	-0.8452	offset
	0.7549	y1(t-1)	1.1047	y1(t-1)
	8.4551	u1(t-2)	-0.2181	y1(t-3)
	-0.0011	u3(t-6)y1(t-3)	5.5309	u1(t-1)u1(t-1)
	0.0012	u3(t-1)y1(t-3)	3.91e-8	u4(t-1)u4(t-1)
	0.0015	u4(t-1)u1(t-4) u1(t-6)	9.038	u1(t-4)u1(t-4) u1(t-6)
@1550rpm	-32.4711	offset	-1.8704	offset
	0.6179	y1(t-1)	0.6925	y1(t-1)
	0.0274	u3(t-1)	0.1102	y1(t-2)
	1.12e-5	u2(t-1)u4(t-2)	7.83e-8	u4(t-1)u4(t-1)
	4.07e-3	u2(t-2)y1(t-2)	9.975	u1(t-2)u1(t-2)
	-2.82e-9	u3(t-5)u3(t-4) u3(t-4)	2.1e-7	u4(t-5)u1(t-5) u4(t-5)
@1750rpm	-56.1132	offset	-1.6023	offset
	0.7620	y1(t-1)	0.8417	y1(t-1)
	0.0579	u3(t-1)	-0.0202	y1(t-3)
	-1.48e-5	u3(t-2)u3(t-3)	2.15e-5	u2(t-1)u4(t-1)
	0.3312	u1(t-1)y1(t-3)	-1.28e-7	u4(t-1)u4(t-1)
	1.15e-10	u4(t-1)u3(t-1) u3(t-3)	1.60e-7	u4(t-5)u1(t-5) u4(t-5)
y2	0.3792	offset	0.3381	offset
@1150rpm	0.4380	y2(t-1)	0.3726	y2(t-1)
	-3.74e-5	u4(t-5)	0.2074	y2(t-2)
	0.5763	u1(t-6)y2(t-2)	-6.69e-9	u4(t-5)u4(t-5)
	0.0022	u2(t-4)y2(t-3)	0.25	u1(t-5)u1(t-5)
	-2.43e-5	y2(t-1)u4(t-6) y2(t-2)	0.6386	y2(t-3)u1(t-6) u1(t-6)
@1350rpm	0.2704	offset	0.1045	offset
	0.4442	y2(t-1)	0.5236	y2(t-1)
	0.1935	u1(t-5)	0.3118	y2(t-2)
	-8.83e-9	u4(t-4)u4(t-6)	-2.57e-9	u4(t-4)u4(t-4)
	0.0024	u2(t-5)y2(t-2)	0.6419	u1(t-5)u1(t-5)
	1.54e-6	y2(t-3)u2(t-4)* u3(t-1)	-4.21e-9	u4(t-6)u4(t-6) y2(t-2)
@1550rpm	0.3521	offset	0.3305	offset
	0.3934	y2(t-1)	0.4290	y2(t-1)
	0.1394	y2(t-2)	0.1818	y2(t-2)
	-7.46e-5	u4(t-6)y2(t-3)	-1.44e-8	u4(t-6)u4(t-6)
	4.05e-6	y2(t-3)u2(t-6)* u3(t-3)	4.75e-7	y2(t-3)u2(t-6) u4(t-2)
			0.5795	u1(t-5)u1(t-5)
@1750rpm	-0.0605	offset	0.3291	offset
	0.6384	y2(t-1)	0.4094	y2(t-1)
	0.1782	y2(t-2)	0.1694	y2(t-2)
	-7.526e-5	u4(t-6)y2(t-1)	-1.05e-8	u4(t-6)u4(t-6)
	4.018e-6	u2(t-4)u3(t-1)	2.38e-5	u2(t-1)u2(t-1)
	2.64e-8	y2(t-3)u3(t-1) u3(t-3)	3.67e-5	y2(t-3)u2(t-5) u2(t-5)
u1:abv	u2:map	u3:rpm	u4:fpw	
CORR:	Correlation Method	ERR:	Error Reduction Ratio Method	

Chapter 7

Forward and Inverse IC Engine Multi-modelling

7.1 Introduction

The engine identification process usually has various objectives which determine the approach of multi-modelling. The forward multi-modelling aims to find a multi-model which can best represent the dynamical behaviour of the engine branch. A novel forward multi-model for WAVE-RT virtual engine test bed is developed and presented in this chapter. The structure selection techniques are applied in the local model identification process at the same time. For the subject of inverse identification, several research works have been published: a nonlinear direct-inverted engine plant model was developed and a inverse compensated system was then established accordingly via the nonlinear parameter uncertainty estimation in the work of Horowitz [105]. Related mathematical inverse techniques were proposed by Petridis and Shenton [106, 107] for the nonlinear identification of continuous time systems. In this chapter, a novel direct inverted engine model and the resulting control test is presented.

7.2 Multi-model Identification of WAVE-RT Engine

An experimental multi-modeling application was conducted on a WAVE-RT virtual engine and the influences of the multi-model complexity was assessed based on the output-error prediction quality of the model. The experiment was designed to establish an engine operating trajectory in a 2-D space formed by two sheduling variables. The data set is then partitioned and located into several local data segments. This approach of data partitioning is found to capture more of the transient dynamics when the engine traverses operation from one regime to another. Both the stepwise regression and the orthogonal least squares methods are used in the structure identification process and a binary tree structure is adopted as the form of the overall model network. As a result, the multi-models for air/fuel ratio and torque are developed and validated separately.

The WAVE Real-Time (RT) engine model is used to simulate the Ford EcoBoost 2.0-Litre GTDI engine. The engine simulation software 'WAVE' is an ISO approved 1D computer-aided package used for creating virtual models. The WAVE-RT engine model, also known as the virtual engine, is taken as a surrogate for the actual engine for the initial stage of engine test as it is able to closely approximate the real engine dynamics and reduce the experimental cost. In order to build MISO model for the engine torque and air/fuel ratio response, six I/O channels have been adopted. The directly measured variables include Fuel-Injection (FUL/mg/stroke), Spark-Advance angle ($SA1/^{\circ}$), Throttle angle ($THR/^{\circ}$),and Engine Speed (RPM/rpm). The Torque (TRQ/Nm) and Air/Fuel Ratio (AFR) are estimated by MISO models and validated by measured data. The delayed basic linear regressors are shown in Table 7.1. After data down-sampling, 1800 data points are finally located. The data sets used for identification are shown in Figure 7.1, and the validation data is a set of 1800 samples collected in an independent virtual engine test.

Table 7.1: Input and Output Channels

Measured Variables	Notation	Delays
FUL	$u1$	$u1(t-1), u1(t-2), ..., u1(t-7)$
THR	$u2$	$u2(t-1), u2(t-2), ..., u2(t-7)$
SA	$u3$	$u3(t-1), u3(t-2), ..., u3(t-7)$
RPM	$u4$	$u4(t-1), u4(t-2), ..., u4(t-7)$
TRQ	$y1$	$y1(t-1), y1(t-2), y1(t-3)$
AFR	$y2$	$y2(t-1), y2(t-2), y2(t-3)$

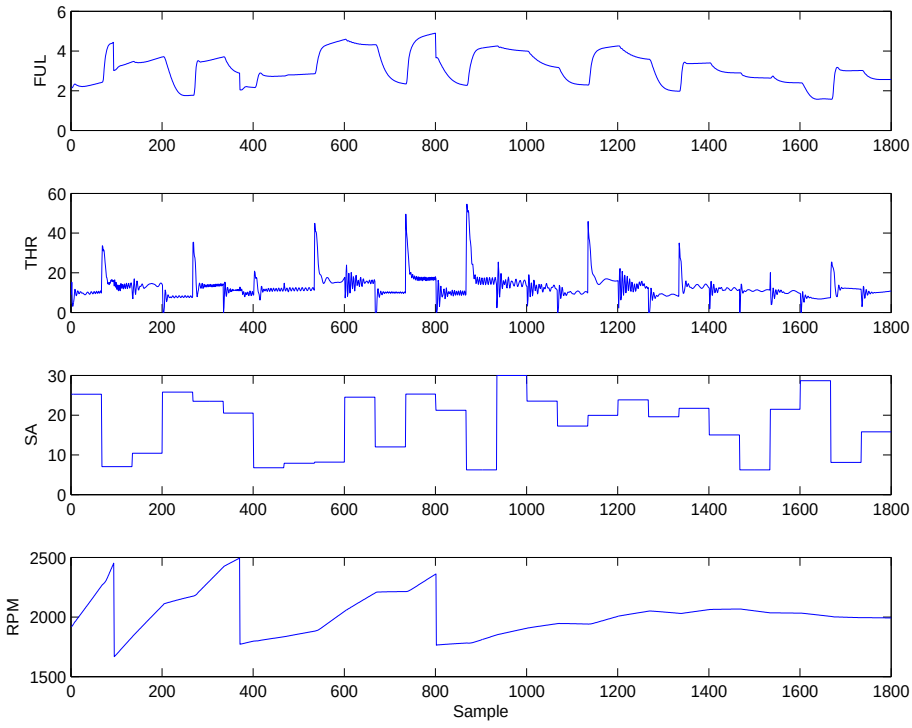

Figure 7.1: Input channels for multi-model identification

7.2.1 Weighting Functions for Local Models

In this application, engine speed (RPM) and throttle angle (THR) are selected to form the 2D scheduling space. As shown in Figure 7.3, the spark advance (SA) and throttle angle (THR) are chosen as the two dimensions of the scheduling operating region for air/fuel ratio estimation. The corresponding local model is required to be activated when the engine operates in the valid local area. The multi-model is incorporated by a logistic sigmoid activation function to avoid the side effects of normalization and the sum of weighting partitions is guaranteed to be unity. As presented in Figure 7.3, the operating regions are partitioned into 3 local regions for the estimation of AFR and torque. This weighting function approach was adopted for multi-modelling of a EURO V engine in [97]. The weighting function used is represented as:

$$\rho(x) = \frac{1}{1 - e^{-\alpha(x-\beta)}} \tag{7.1}$$

$$\rho^{'}(x) = 1 - \rho(x) \tag{7.2}$$

where α is the coefficient which defines the overlaps between the neighbouring regions and β is the border of separated operating regions (see Figure 7.2).

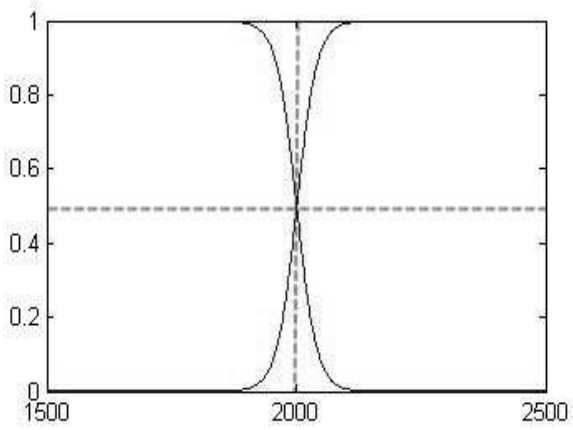

Figure 7.2: Sigmoid weighting functions (α=0.05,β=2000)

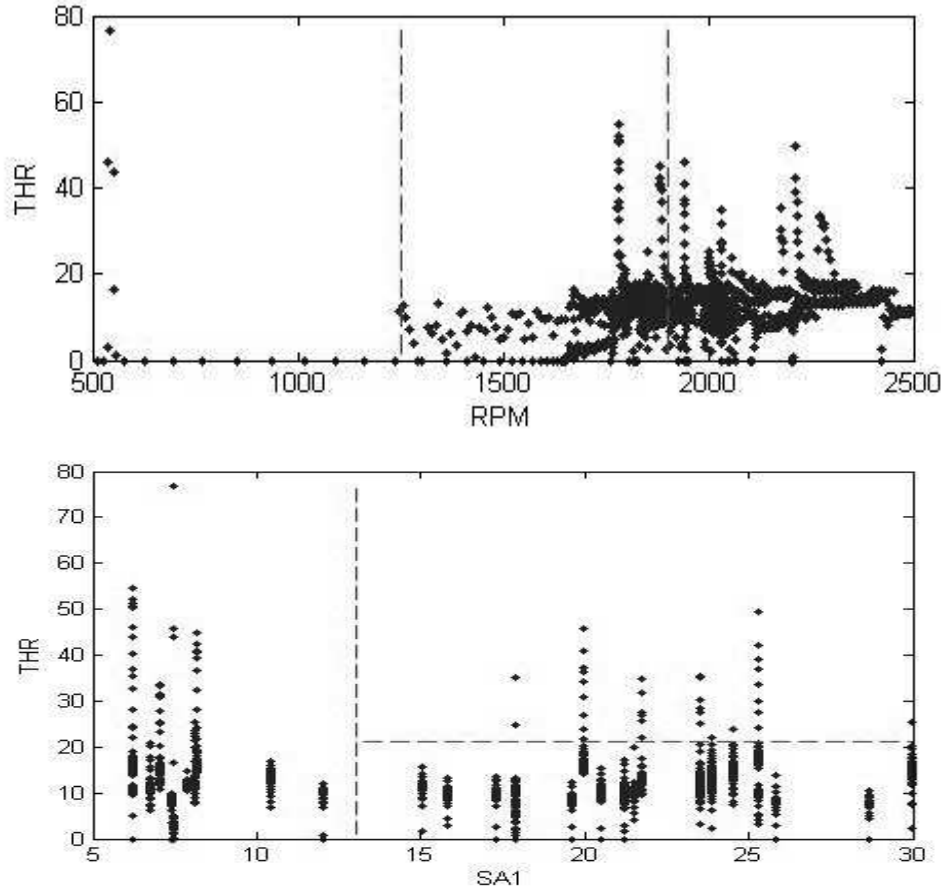

Figure 7.3: 2D Scheduling for Torque and Air/Fuel Ratio

7.2.2 The Estimation of Torque and Air/Fuel Ratio

Both the F-statistic analysis and error reduction ratio analysis were used to generate the multi-nonlinear model of the engine torque and air/fuel ratio. The model validation results for these two techniques are presented in Figure 7.4 and Figure 7.5 respectively. The inaccuracy which appeared on the estimated torque graph is caused by the overfit of the model when a sudden change occurs in the sequence of the input signal. This inaccuracy could be avoided by using smoother input signal or reduce the engine model order. For the purposes of model validation, the quality of the models are assessed by their R^2 statistics in the form:

$$R^2 = 1 - \frac{\sum_{k=1}^{N}(y(k) - \hat{y}(k))^2}{\sum_{k=1}^{N}(y(k) - \overline{y})^2} \tag{7.3}$$

As shown in Table 7.2, the model estimated by F-statistic analysis gives better torque validation result, however for air/fuel ratio (λ) estimation, the error reduction ratio technique offers better results. It is worth noting that the quality of the multi-model will be improved with more local models until it reaches a certain bottleneck (see Figure 7.6).

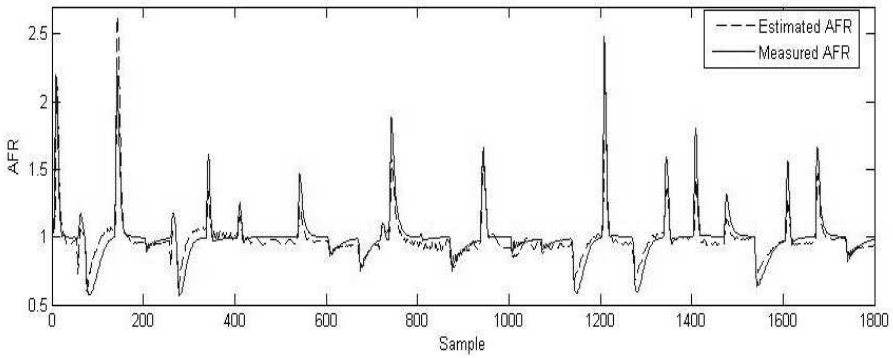

Figure 7.4: Estimated vs Measured Air/Fuel Ratio by Error Reduction Ratio Analysis

Table 7.2: Model Prediction Quality

Output Channel		Torque	AFR
F-statistic	R^2	0.7899	0.8249
ERR	R^2	0.7045	0.9070

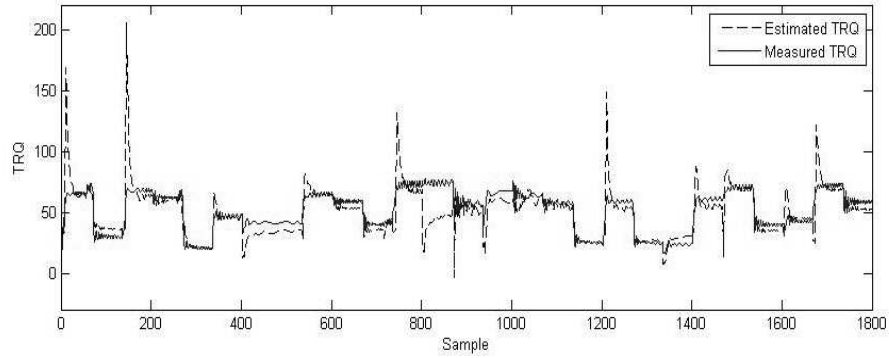

Figure 7.5: Estimated vs Measured Torque Output by F-statistic Analysis

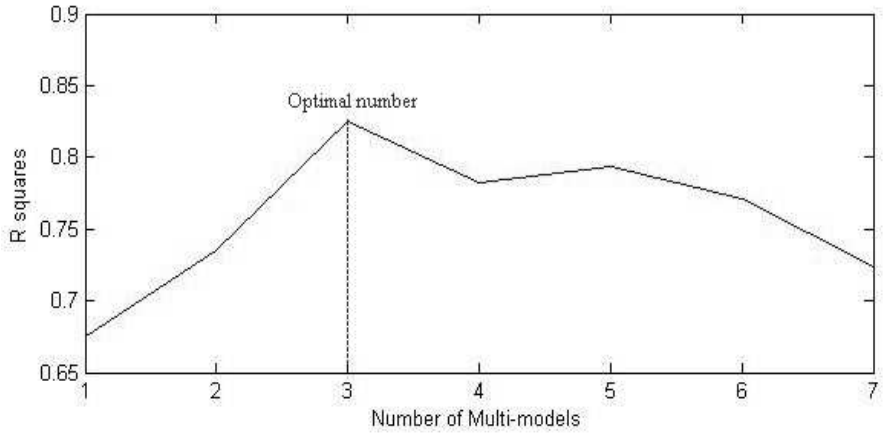

Figure 7.6: R^2 vs Multi-model Number for Air/Fuel Ratio Estimation

7.3 Inverse Multi-modelling for Throttle Angle and Spark Advance

Inverse engine identification swaps the measured input/output as the inverse model's output/input. Consequently, the inverse models are generally used as feedforward compensators to track the desired outputs [108]. With the inverse compensators, the resulting overall open loop system is able to greatly offset the nonlinear uncertainty and nonlinearity and the feedback controllers can be developed using linear controller design techniques [109, 110, 23].

In this section, the multi-model identification approach is applied to a virtual engine test-bed implemented as a WAVE-RT model, which is a state-of-the-art industrial level representation of a Ford EcoBoost 2.0-Litre GTDI engine. The additional information about the WAVE-RT model can be found in Chapter 3. Figure 7.7 gives the general structure of the inverse multi-model. Three dynamic nonlinear affine component models are developed to fit the measured spark advance and throttle angle in the corresponding partitions and the overall output is obtained by accumulating the weighted output from each of the local models. In a more nonlinear system more component models could be added. Figure 7.8 shows the setup of the WAVE-RT engine and the inverse MISO multi-model. By using the inverse model as a feedforward compensator, the T_b and λ can be controlled through the spark advance and throttle angle by feedforward of the desired values, $T_b^{'}$ and $\lambda^{'}$

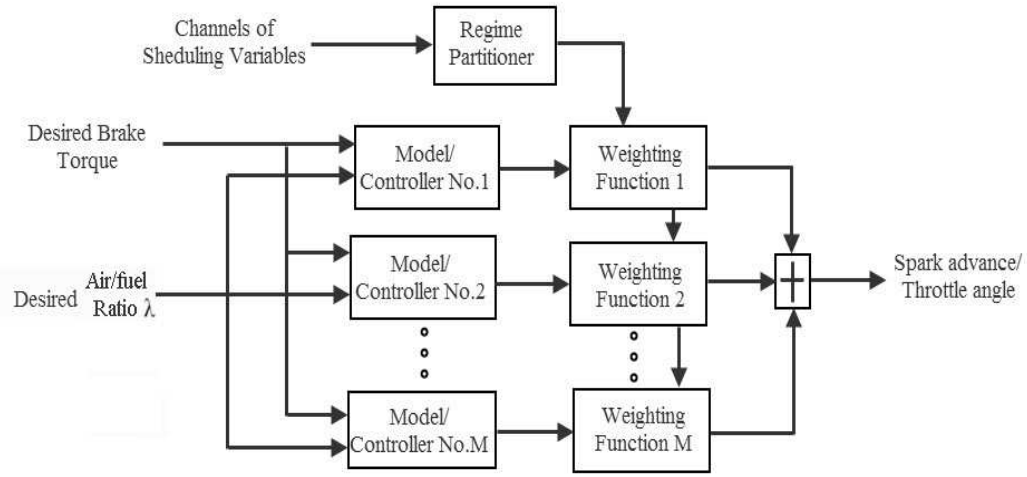

Figure 7.7: Inverse MISO multi-model structure

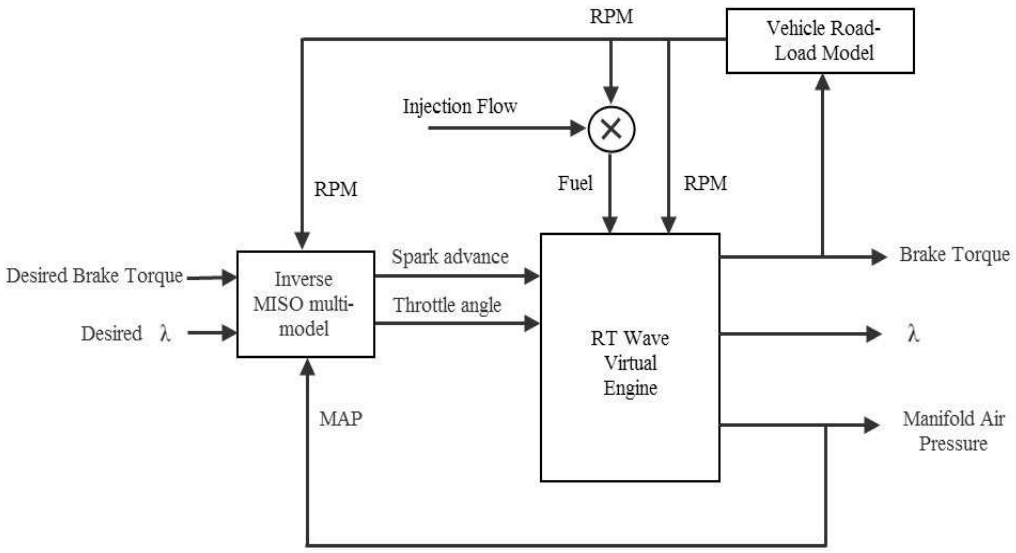

Figure 7.8: WAVE-RT engine test setup for inverse MISO multi-model

The relevant test signals are shown in Figure 7.9. Engine and desired engine brake torque T_b, $T_b^{'}$ and lambda and desired lambda λ, $\lambda^{'}$ are selected as inputs for the inverse multi-model. As indicated in Figure 7.10 and Figure 7.11, the collected engine data is separated into 3 local regions in a 2-D space with manifold air pressure (MAP) and RPM selected as scheduling variables. The boundaries among the regions are optimized via multiple model fitness evaluations. The weighting functions are chosen as logistic sigmoid functions to guarantee that the weighting factors sums to unity. The simulation results for spark advance and throttle angle controller from the multi-models are presented in Figure 7.12 and Figure 7.13 respectively. The goodness of fit of spark advance and throttle angle are 68.6% and 63.8% and the R^2 values calculated from model outputs and measured data can achieve 90% at the optimized combinations of scheduling variables. An open-loop test has been conducted for the inverse controller and the filtered engine torque and Lambda output together with the desired torque and Lambda are presented in Figure 7.14.

7.4 Conclusion

In this chapter, nonlinear forward multi-model have been developed for the purpose of simulation of the WAVE-RT engine. The engine RPM and throttle angle has been adopted to

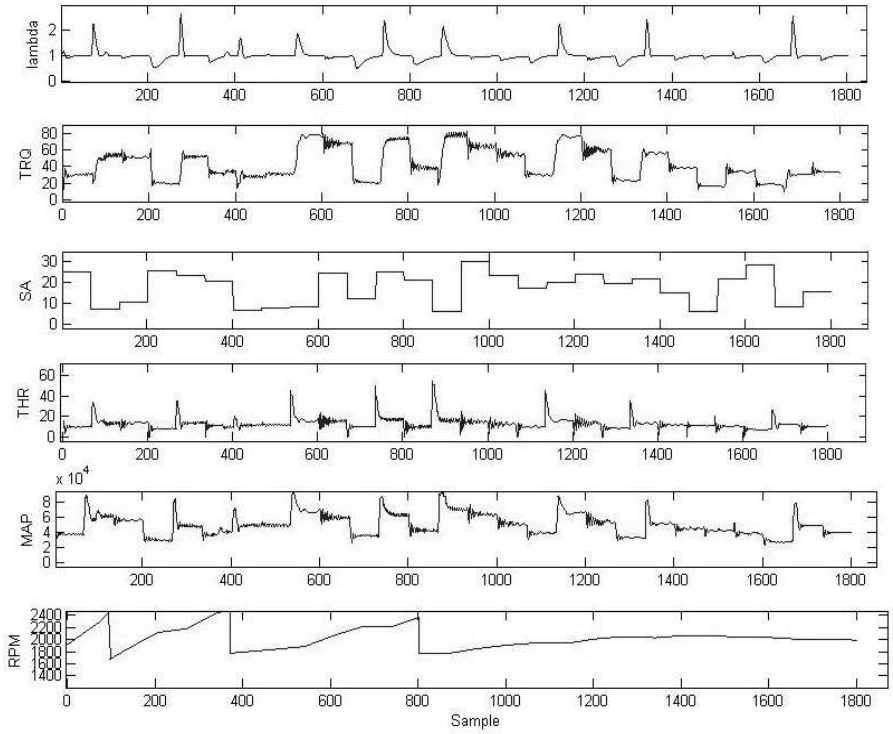

Figure 7.9: Data channels of the inverse multi-model

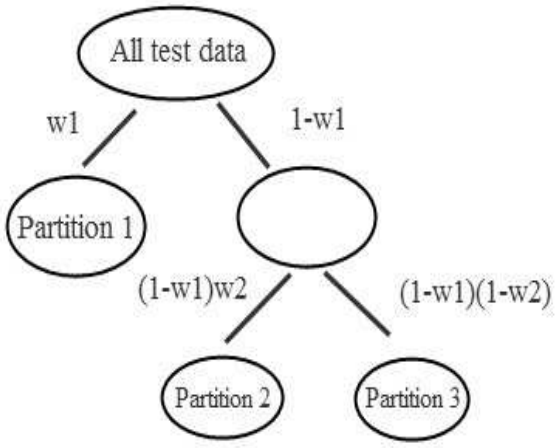

Figure 7.10: Data partitioning tree

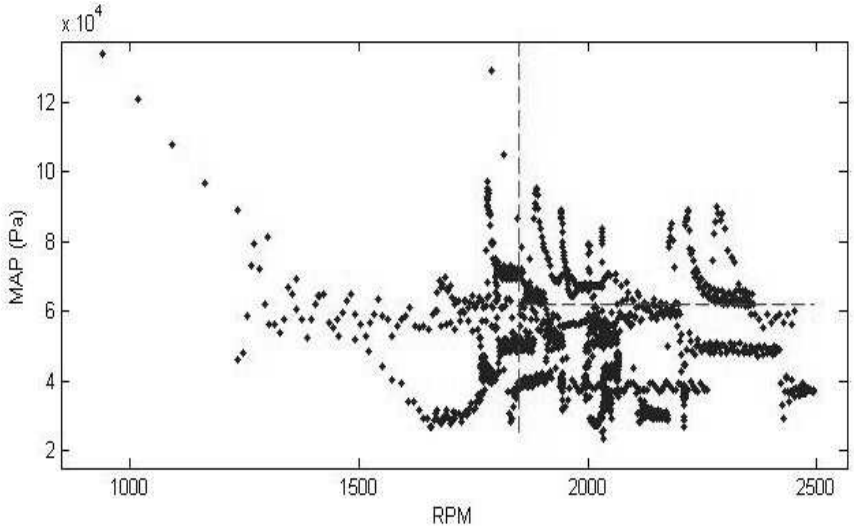

Figure 7.11: 2D data partitioning with MAP and RPM

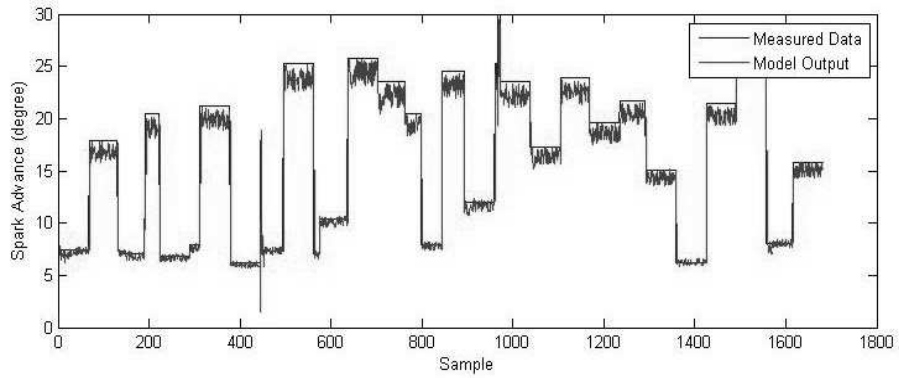

Figure 7.12: The validation result of Inverse multi-model for spark advance (SA)

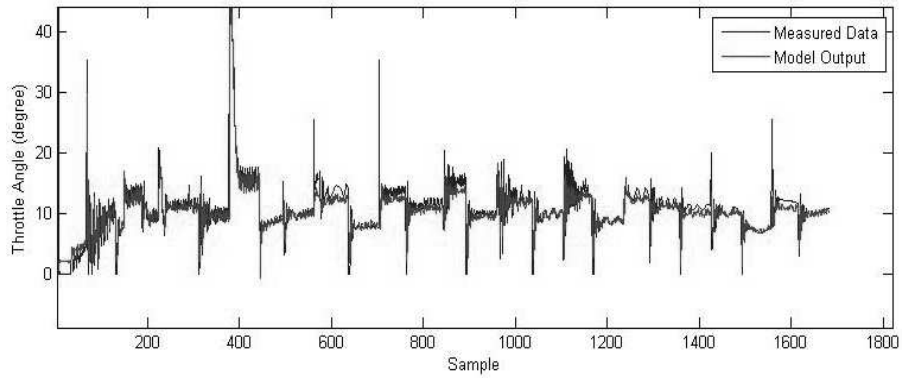

Figure 7.13: The validation result of Inverse multi-model for throttle angle (THR)

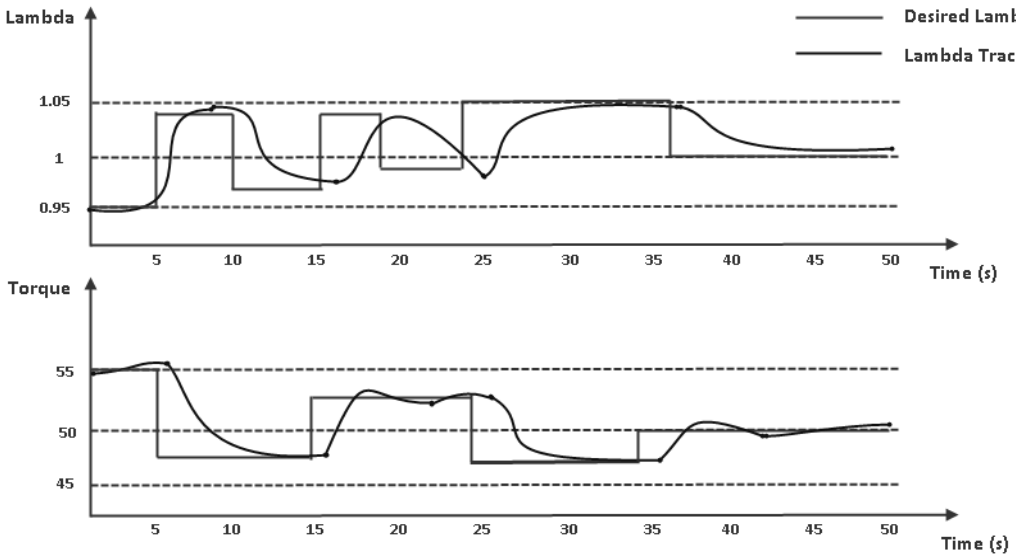

Figure 7.14: A open-loop simulation of the inverse control

form the 2-D scheduling space and the local NARX model structures are selected by the F-statistic and orthogonal least squares methods. The validation test results have shown that the estimated engine brake torque (Tb) and air/fuel ratio (λ) fits well with the measured data.

Secondly, a directly inverted nonlinear multi-model has been developed and an inverse control system is then established. The inverse compensated system output is able to track the desired engine output and thus achieve real-time control. It can be concluded that the inverse controller can offset the nonlinear uncertainties of the engine plant thus create a unit gain between the desired outputs and the controlled engine outputs.

Chapter 8

Conclusions and Future Work

8.1 Introduction

This thesis aims to develop dynamical nonlinear models especially nonlinear polynomial models for IC engine with various model structure selection techniques and validated by the engine testing results.

The primary conclusion is that the dynamic model structure selection techniques based on stepwise regression and orthogonal least squares can be applied successfully on the engine identification process and produced both good polynomial ARX and NARX models for the engine system. Secondly, these model structure selection technique can be used in identifying the polynomial local models when developing global multi-models for the engine. The best trade-off are required to be found in calibrating any engine system because contradictory requirements exist in the engine controller optimization. The engine system is conventionally calibrated by static mapping techniques which are based on multi-dimensional tables describing the relationship among the engine inputs and outputs. The model-based calibration is nowadays introduced to meet the challenge of more advanced engine calibration where the engine properties are represented by dynamical mappings instead of the old method of look-up tables. The engine control scheme can be developed based on the engine mapping generated by the simulation of the engine model . Consequently, it is essential for the engine model to have the adequate accuracy to predict the engine dynamic behaviour.

8.2 Conclusions

- Due to the complexity of the engine, nonlinear models are able to predict the engine characteristics more accurately than linear models. Novel SISO dynamical models were developed based on PFI gasoline engine in Chapter 4. The model structures were found by Matlab identification toolbox and validated by the measured data. The polynomial model structure was adopted, especially, the least-squares technique is adopted to determine the parameters of the models. Finally, both ARX and NARX models of the engine system has been obtained. It can be concluded that both stepwise regression and orthogonal least squares(OLS) techniques can be applied to the torque and air/fuel ratio model identification. In terms of the time-domain discrete system identification, both methods are able to develop parametric model structures efficiently. In practice, the directly identified model may have the capability to bypass various engine geometry problems and integrate the engine nonlinearities via the model regressor selection process. As shown in the validation results, the model prediction ability of the two methods gets closer with increasing model term number. Meanwhile, the programming and calculation process of the OLS approach is less time-consuming than that of stepwise regression method and the OLS approach can achieve better prediction performance when fewer parameters are required in the model.

- A comparison between different types of least squares parameter estimation has been made in this thesis. The results presented in chapter 4 involves linear and nonlinear SISO ABV to engine speed model of the PFI engine. The least-squares technique is adopted to determine the parameters of the models. Both ARX and NARX model of the engine system are obtained. However, there is no single best model for the engine system. It can be said that the compromise between the complexity of the model and the quality of the model fit is significant. Generally, system identification is carried out as a priori step of controller design. Therefore, the accuracy of the identification model corresponds to the requirements of the controller and control design technique. Based on the results represented in chapter 4, it can be concluded that the results of identification of linear ARX models are not sensitive to different LS identification algorithms. Secondly, the square terms of the output RPM have been inserted to construct the NARX model. By adding the nonlinear terms into the ARX model

equation, the fit ratio of the model can be improved. The NARX model represented in this chapter has 5% improvement than the ARX model.

- Novel MISO dynamical models for IC engine was developed using stepwise regression and orthogonal least squares techniques in Chapter 5. At the same time the detailed steps of the iterative structure selection procedure are presented. Their ease of implementation makes them highly suited to applications within ECU software. Nevertheless to ensure the full benefits of the approach, it is necessary to ensure that partitioned domains for the different models and thus where one model becomes more weighted relative to the others are properly. This issue has received little previous attention. The multi-model is able to represent complex nonlinear behaviour yet retains a simple implementable structure for EMS engine mappings. Both stepwise regression and orthogonal least squares (OLS) techniques can be applied successfully to the torque and Air/Fuel ratio model identification.

- Based on the results represented in chapter 6, the multi-model structures is capable to be developed as an alternative black box dynamic approach for the case of a nonlinear engine map and a systematic method of developing engine mappings can be applied to the engine calibration process which can save a significant amount of time. The relationship between model fitness and likely EMS computational overhead as measured by number of multiplications have been investigated. The results show an appropriate trade-off relation between the computational cost and the model quality can be established by the structure identification techniques. In the case of the multi-models, the associated local models can be conveniently developed from the engine experiments at different operating points and we can apply linear identification and control techniques to the linear local models which is a significant advantage.

- Multi-models are currently being considered by several automotive researchers as an efficient and understandable modelling approach to achieve high accuracy dynamic models and filters. The general process of the multi-modelling includes input design, choice of scheduling variables, data partitioning, determination of weighting functions and local model identification. In chapter 6, they are introduced and executed on the IC engine test data. A novel LOLIMOT model and a log-sigmoid weighted multi-model was established for IC engines. The optimal number of local models is analysed

according the multi-modelling results.

- A novel inverse multi-model has been identified on a state-of-the art virtual engine model. The inverse model has been used to track the engine outputs of torque and air/fuel ratio.

8.3 The Perspective of Future Work

- It is promising to further the application of the model structure selection techniques to other types of models. The selection techniques of model regressors can be extended to identify other classes of dynamical models including Output Error (OE) models and Nonlinear Output Error(NOE) models. Such a different approach is worthwhile because these models utilize k-step prediction outputs as the regressors instead of the measured output used to identify ARX and NARX models.

- The data partitioning strategies for the multi-models still requires extensive effort in order to achieve a structured and effective methodology to give the optimal solution. On the other hand, it is also important to test different partitioning strategies on the engine and investigate their respective benefit. Therefore, the data partitioning method is an interesting subject in the future.

- The weighting functions introduced in this thesis, including Piecewise, Gaussian and log-sigmoid functions, are validated on separated data segments. As demonstrated in Figure 8.1, the neighbouring Gaussian and log-sigmoid curves are overlapped at some point with the valid curve. The influence of the weighting outside the bound will be diminished in this case. Therefore, it is important to find how to define the the upper bound and the lower bound of the valid curve, since they decide to what extent one local model interrelates with the others or whether the local models are entirely independent of each other. This area is worth looking at in the future.

- The switching mechanism between the different kinds of local models is an important concern in the future application for multi-modelling of the engine systems. If different classes of local models, such as an output error model and polynomial NARX model, are to be incorporated, one or other model is required to be selected depending on

the priority of the application. Switching is also needed when the model is providing real-time simulation because the optimal local model could be different when a new operating range is reached.

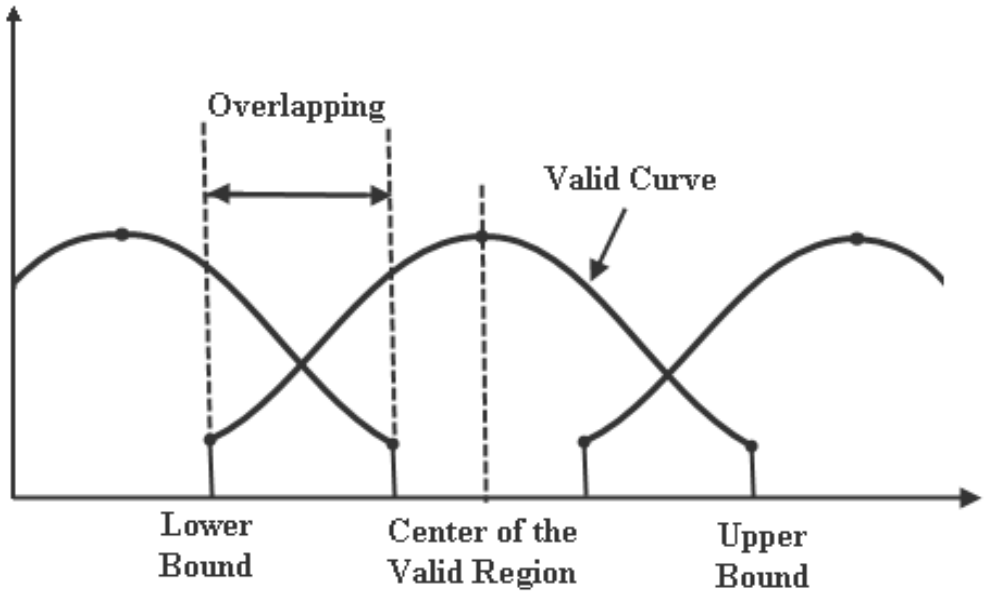

Figure 8.1: The boundaries for the curve of the weighting function

References

[1] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton Uninversity Press, 1961.

[2] Ropke K., Knaak M., Nessler A., and Schaum S. Rapid measurement: Basic combustion engine calibration in one day? *MTZ Worldwide, Vieweg Verlag GWV Fachverlage GmbH Germany*, 88:276–279, 2007.

[3] R. Sykes and A. T. Shenton. Prospects of transient calibration for forthcoming legislated drive cycles. *Powertrain Modelling and Control (PMC 2012) Conference, Bradford University*, 2012.

[4] K.J. Astrom and B. Wittenmark. *Computer-Controlled Systems, 3rd ed.* Prentice Hall, 1997.

[5] R. Murray-Smith and T. A. Johansen. *Multiple Model Approaches to Modelling and Control*. Taylor and Francis, New York, 2010.

[6] J. Hauth. Grey-box modelling for nonlinear systems. *Ph.D. Thesis*, Department of Mathmatics, Kaiserslautern University of Technology, Germany, 2008.

[7] Kaplanoglu E., Safak K.K., and H.S. Varol. Real-time parameter estimation of a mimo system. *In the Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2, 2008.

[8] B. Fabien. *Analytical System Dynamics:Modelling and Simulation*. Springer Science+Business Media, New York, 2009.

[9] N.U. Ahmed. *Dynamic Systems and Control with Applications*. Hackensack, NJ : World Scientific, 2006.

[10] F. Haugen. *Dynamic Systems : Modelling, Analysis and Simulation*. Trondheim : Tapir Academic, 2004.

[11] Hsia T.C. *System identification: Least-squares methods*. Lexington Books, Mass, 1977.

[12] A. C. Atkinson and A. N. Donev. *Optimum Experimental Designs*. Clarendon Press, Oxford, 1992.

[13] T. Soderstrom and P. Stoica. *System Identification*. Prentice-Hall, UK, 1989.

[14] L. Ljung. *System Identification-Theory For The User*. PTR Prentice Hall, New Jersey, USA, 1999.

[15] Karlsson M., Ekholm K., Strandh P., Johansson R., and Tunestal P. Dynamic mapping of diesel engine through system identification. *In Proceedings of American Control Conference, June*, pages 3015–3020, 2010.

[16] K. (ed.) Ropke. *Design of Experiments (DoE) in Engine Development V*. ExpertVerlag, 2011.

[17] Verhaegen M. and Verdult V. *Filtering and System Identification, A Least Squares Approach*. Cambridge University Press, New York, 2007.

[18] Ljung L. *System Identification Toolbox, Getting Started Guide*. The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, 2010.

[19] P. E. Wellstead. *Introduction to Physical System Modelling*. Academic Press, London, 1979.

[20] R. Haber and H. Unbehauen. Structure identification of nonlinear dynamic systems-a survey on input/output approaches. *Automatica*, 26, 4:651–677, 1990.

[21] Chan S.H and Zhu J. Modelling of engine in-cylinder thermodynamics under high values of ignition retard. *International Journal of Thermal Sciences*, 40:94–103, 2001.

[22] Wu Y.Y. Chen B.C. and Hsieh F.C. Estimation of engine rotational dynamics using kalman filter based on a kinematic model. *IEEE Transactions on Vehicular Technology*, 59:3728–3735, 2010.

[23] Enns D., D. Bugajski, R. Hendrick, and Stein G. Dynamic inversion: an evolving methodology for flight control design. *International Journal of Control*, 59:71–91, 1994.

[24] Sjoeberg J., Zhang Q., Benveniste A. Ljung L., Delyon B., Glorennec P., Hjalmarsson H., and Juditsky A. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31,12:1691–1724, 1995.

[25] L. Ljung and T. Soderstrom. *Theory and practice of recursive identification*. MIT Press, Cambridge, Massachusetts, 1983.

[26] S. Chen, S. Billings, and P. Grant. Non-linear system identification using neural networks. *International Journal of Control*, 51,6:1191–1214, 1990.

[27] S. Chen and S. Billings. Neural networks for nonlinear dynamic system modelling and identification. *International Journal of Control*, 56,2:319–346, 1992.

[28] Lindskog P. and Ljung L. Tools for semiphysical modelling. *International Journal of Adaptive Control and Signal Processing*, 9,6:509–523, 1995.

[29] Sjoberg J. A nonlinear grey-box example using a stepwise system identification approach. *In Proceedings of the 11th IFAC Symposium on Identification, Santa Barbara,USA*, 2000.

[30] H. Langouet, L. Metivier, D. Sinoquet, and Q.H. Tran. Optimization for engine calibration. *International Conference on Engineering Optimization*, pages 34–42, 2008.

[31] Liu F., Amaratunga G., Collings N., and Soliman A. An experimental study on engine dynamics model based in-cylinder pressure estimation. *SAE Technical Paper, 2012-01-0896*, 2012.

[32] Ropke K., W. Baumann, B.U. Kohler, S. Schaum, R. Lange, and M. Knaak. Engine calibration using nonlinear dynamic modeling. *Identification for Automotive Systems*, 418:165–182, 2012.

[33] Guzzella L. and Onder C.H. *Introduction to Modelling and Control of Internal Combustion Engine Systems*. Springer-Verlag, 2004.

[34] Sun J., Kolmanovsky I., Cook J.A., and Buckland J.H. Modelling and control of automotive powertrain systems: a tutorial. *In Proceedings of American Control Conference*, pages 3271–3283, 2005.

[35] Tan Y. and Saif M. Nonlinear dynamic modelling of automotive engines using neural networks. *In Proceedings of the IEEE International Control Conference on Control Application*, pages 408–410, 1997.

[36] Zhao F., D.L. Harrington, and Lai M.D. *Automotive Gasoline Direct-injection Engines*. SAE International, 2002.

[37] Hiromitsu A. Combustion control for mitsubishi gdi engine. *In Proceedings of the 2nd International Workshop on Advanced Spray Combustion*, pages 225–235, 1998.

[38] Kazunari K. and Hiromitsu A. Control of mixing and combustion for mitsubishi gdi engine. *Proc. JSAE Annual Congress*, 984:35–38, 1998.

[39] Martyr A.J. and Plint M.A. *Engine Testing - Theory and Practice,3rd edition*. Butterworth-Heinemann, Oxford, 2007.

[40] Nagumo S. and Hara S. Study of fuel economy improvement through control of intake valve closing timing: cause of combustion deterioration and improvement. *JSAE Review*, 16,1:13–19, 1995.

[41] Sher E. and Bar-Kohany T. Optimization of variable valve timing for maximizing performance of an unthrottled si engine-a theoretical study. *Energy*, 27,8:757–775, 2002.

[42] Meacham G. B. Variable cam timing as an emission control tool. *SAE Paper No.700645*, 1970.

[43] Lenz H. P, Wichart K., and Gruden D. Effects of variable engine valve timing on fuel economy. *SAE Paper No.880390*, 1988.

[44] Stefanopoulou A. G., Cook J. A., Freudenberg J.S., Grizzle J. W., Haghgooie M., and Szpak P.S. Modeling and control of a spark ignition engine with variable cam timing. *Proceeding of American Control Conference*, 4:2576–2581, 1995.

[45] Bae J.I. and Bae S.C. A study on the engine downsizing using mechanical supercharger. *Journal of Mechanical Science and Technology*, 19,12:2321–2329, 2005.

[46] J. Wahlstrom. Control of egr and vgt for emission control and pumping work minimization in diesel engines. *Ph.D. Thesis*, Linkoping University, Linkoping, Sweden, 2009.

[47] A. Froberg and L. Nielsen. Efficient drive cycle simulation. *IEEE Trans. Veh. Technol.*, 57,2:1442–1453, 2008.

[48] J. Wahlstrom, L. Eriksson, and L. Nielsen. Egr-vgt control and tuning for pumping work minimization and emission control. *IEEE Trans. Control Systems Technol.*, 18,4:993–1003, 2010.

[49] L. Eriksson, J. Wahlstrom, and M. Klein. Physical modeling of turbocharged engines and parameter identification. *Automotive model predictive control: Models, Methods and Applications.*

[50] J. Wahlstrom 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*, 225:960–986, 2011.

[51] H.R. Ricardo. *Engines and Enterprise, 2nd Edition.* Blackie and Son limited, 1941.

[52] H.R. Ricardo and J.G. Hempson. *High Speed Internal Combustion Engine, 5th Edition.* Blackie and Son limited, 1968.

[53] Golub G. and Van Loan C. *Matrix Computations, 3rd ed.* Johns Hopkins University Press, USA, 1996.

[54] Wellstead P. and Zarrop M. *Self-tunning systems: Control and signal processing.* John Willey and Sons, Inc., UK, 1991.

[55] H. Akaike. Statistical predictor identification. *Ann. Inst. Statist. Math.*, 22, 1:203–217, 1970.

[56] J. Rissanen. Modeling by shortest data description. *Automatica*, 14,5:465–471, 1978.

[57] Basso M., Giarre L., Groppi S., and Zappa G. Narx models of an industrial power plant gas turbine. *IEEE transactions on control systems technology*, 13,4:599–604, 2005.

[58] G. Lee, B.and Rizzoni, Y. Guezennec, Solomon A., Cavalletti M., and Waters J. Engine control using torque estimation. *SAE*, 1, 2001.

[59] Z. Filipi and D. Assanis. A nonlinear, transient, single-cylinder diesel engine simulation for predictions of instantaneous engine speed and torque. *J. Eng. Gas Turbines Power*, 123, 4:951–960, 2001.

[60] F. Connolly and G. Rizzoni. Real time estimation of engine torque for the detection of engine misfires. *J. Dyn. Sys., Meas., Control*, 116, 4:875–687, 1994.

[61] G. Ingram, M. Franchek, and V. Balakrishnan. Spark ignition engine torque management. *Proc. American Control Conference*, 1, 4:767–772, 2003.

[62] G. Rizzoni and Y. Zhang. Identification of a non-linear internal combustion engine model for on-line indicated torque estimation. *Mechanical Systems and Signal Processing*, 8, 3:275–287, 1994.

[63] V. Klein and E. Morelli. *Aircraft System Identification: Theory and Practice.* American Institute of Aeronautics and Astronautics, 2006.

[64] H. Cordell and D. Clayton. A unified stepwise regression procedure for evaluating the relative effects of polymorphisms within a gene using case/control or family data: Application to hla in type 1 diabetes. *The American Society of Human Genetics*, 70, 1:124–141, 2002.

[65] R. Pearson. *Discrete-Time Dynamic Models.* Oxford university Press,Oxford, 1999.

[66] S. Billings, M. Korenberg, and S. Chen. Identification of nonlinear output-affine systems using an orthogonal least-squares algorithm. *International Journal of Systems Science*, 19, 1:1559–1568, 1988.

[67] H. L. Wei, S. Billings, and J. Liu. Term and variable selection for non-linear system identification. *International Journal of Control*, 77, 1:86–110, 2004.

[68] R. G. Jr. Miller. *Beyond ANOVA*. Chapman and Hall, London, 1997.

[69] I. Lind. Model order selection of n-fir models by the analysis of variance method. *In Proceedings of the 12th IFAC symposium on system identification, Santa Barbara*, pages 367–372, 2000.

[70] I. Lind and L. Ljung. Regressor selection with the analysis of variance method. *Automatica*, 41,4:693–700, 2005.

[71] Lind I. and Ljung L. Regressor and structure selection in narx models using a structured anova approach. *Automatica*, 44:383–395, 2008.

[72] J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19,1:1–67, 1991.

[73] J. Roll, I. Lind, and L. Ljung. Connections between optimisation-based regressor selection and analysis of variance. *In the 45th IEEE conference on decision and control, San Diego, CA*, pages 4907–4914, 2006.

[74] Zadeh L. A. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Systems, Man, and Cybernetics*, 3, 1973.

[75] E. Hendricks. Isothermal vs. adiabatic mean value si engine models. *In Proceedings of IFAC Workshop on Advances in Automotive Control*, 2001.

[76] Skogestad S. and Postlethwaite I. *Multivariable Feedback Control: Analysis and Design*. John Wiley and Sons, 2001.

[77] N. Rivara. *IC Engine Control by Ionization Current Sensing*. PhD Thesis, University of Liverpool,UK, 2009.

[78] S. Zhao. *Nonparametric Robust Control Methods for Powertrain Control*. PhD Thesis, University of Liverpool,UK, 2011.

[79] Angelov P. and Filev D. An approach to online identification of takagi-sugeno fuzzy models. *IEEE Transactions on Systems, Man, And Cybernetics-Part B: Cybernetics*, 34,1:960–986, 2004.

[80] Saraswati S. Reconstruction of cylinder pressure for si engine using recurrent neural network. *Journal of Neural Computing and Applications*, 19:935–944, 2010.

[81] Xia Y., Hao G., Shan C., Ni Z., and Zhang W. Reconstruction of cylinder pressure of i.c. engine based on neural networks. *In Proceedings of the 1st International Conference on Pervasive Computing, Signal Processing and Applications, September*, pages 924–927, 2010.

[82] Y. Zhang, L. Xi, and Liu J. Transient air-fuel ratio estimations in spark ignition engine using recurrent neural networks. *KES 2007/WIRN 2007, Part II*, pages 240–246, 2007.

[83] Hou Z., Sen Q., and Wu Y. Air-fuel ratio identification of gasoline engine during transient conditions based on elman neural networks. *IEEE International Conference on Intelligent Systems and Design Applications*, (0-7695-2528-8/06), 2006.

[84] I. Arsie, M.M. Marotta, C. Pianese, and Sorrentino M. Experimental validation of a recurrent neural newtwork for air-fuel ratio dynamic simulation in si ic engines. *In Proceedings of ASME International Mechanical Engineering Congress and Exposition, November*, pages 127–136, 2004.

[85] T. Li and Christos G. Dynamic input signal design for the identification of constrained systems. *Journal of Process Control*, 18,3-4:332–364, 2008.

[86] Vinson D. and Georgakis C. A new measure of process output controllability. *Journal of Process Control*, 10:185–194, 2000.

[87] Mehra R.K. Optimal input signals for parameter estimation in dynamic systems-survey and new results. *IEEE Transactions on Automatic Control*, AC-19(6):753–768, 1974.

[88] Goodwin C.G. and Payne L.R. *Dynamic System Identification: Experiment Design and Data Analysis*. Academic press, 1977.

[89] Aoki M. and Staley R.M. On input signal synthesis in parameter identification. *Automatica*, 6:431–440, 1970.

[90] Lizama E. and Surdilovic D. Designing g-optimal experiments for robot dynamics identification. *Proceedings of IEEE International Conference on Robotics and Automation*, 1996.

[91] Zaglauer S. and Delflorian M. Bayesian d-optimal design. *Proceedings of the 6th Conference Design of Experiments in Engine Development*, 2011.

[92] R. Isermann. *Identifikation dynamischer Systeme2 (2nd ed.)*. Springer Verlag, 1992.

[93] O. Nelles. *Nonlinear system identification (1st ed.)*. Springer Verlag, 2002.

[94] I. Zliobaite. Three data partitioning strategies for building local classifiers. *Studies in Computational Intelligence*, 373:233–250, 2011.

[95] L. Kuncheva. Clustering-and-selection model for classifier combination. *In Proceedings of the 4th International Conference: Knowledge-Based Intell. Eng. Syst. and Allied Technologies, Brighton, UK*, pages 185–188, 2000.

[96] M. Lim and S. Sohn. Cluster-based dynamic scoring model. *Expert Systems with Appl.*, 32:427–431, 2007.

[97] C. Hametner and M. Nebel. Operating regime based dynamic engine modelling. *Control Engineering Practice*, 20:397–407, 2012.

[98] M. Nemani, R. Ravikanth, and B. Bamieh. Identification of linear parametrically varying systems. *In Proceedings of the 34th IEEE control and decision conference*, 3:2990–2995, 1995.

[99] F. Bruzelius, Breitholtz C., and Pettersson S. Lpv-based gain scheduling technique applied to a turbo fan engine model. *Proceedings of the 2002 International Conference on Control Applications*, 2:713–718, 2002.

[100] Paoletti S., Juloski A., Ferrari trecate G., and Vidal R. Identification of hybrid systems: a tutorial. *European Journal of Control*, 13:242–260, 2007.

[101] O. Nelles. *Nonlinear System Identification.* Springer-Verlag,Berlin, 2001.

[102] J. Rezaie, B. Moshiri, and B. Araabi. Land vehicle state estimation using a modified lolimot algorithm. *5th IFAC Symposium on Advances in Automotive Control: Advances in Automotive Control*, 5,1, 2007.

[103] J.D. Martinez, E. Palacios, and C. Velaazquez. Modeling of internal combustion engine emissions by lolimot algorithm. *In Proceedings of the 34th IEEE control and decision conference*, 3:251–258, 2012.

[104] O. Nelles and Isermann R. Basis function networks for interpolation of local linear models. *In the proceedings of the 35th conference on decision and control, Kobe, Japan*, 1:470–475, 1996.

[105] I. Horowitz. Improvement in quantitative non-linear feedback design by cancellation. *International Journal of Control*, 34,3:547–560, 1981.

[106] A.P. Petridis and A.T. Shenton. Non-linear inverse compensation of an si engine by system identification for robust performance control. *Inverse Problems in Engineering*, 8,2:163–176, 2000.

[107] A.P. Petridis and A.T. Shenton. Inverse-narma: a robust control method applied to si engine idle-speed regulation. *Control Engineering Practice*, 11,3:279–290, 2003.

[108] Hover F. Inversion of a distributed system for open-loop trajectory following. *International Journal of Control*, 60:671–686, 1994.

[109] A.P. Petridis and A.T. Shenton. Non-linear inverse compensation of an si engine by system identification for robust performance control. *Inverse Problems in Engineering*, 8:163–176, 2000.

[110] Shenton A.T. and Petridis A.P. Nonlinear miso direct-inverse compensation for robust performance speed control of an si engine. *Nonlinear and Adaptive Control*, 281:337–349, 2003.

Appendices

.1 A. M-Function for Stepwise Regression with F-statistics

```
function [OptimizedReg,OptimizedX,Optimizedtheta,F_partial,flag,error_est,y_est,reg]
=f_selection(MAX_CH,MAX_ORDER,TITLE,DATA,ITERATION_NUM)

%F_selection of regressors for dynamical models
%written by Zongyan Li
%version: November, 2011
%Inputs of the function:
%TITLE=[' C ';' F ';' Tb';' Pm';' N ';' Q ';'NOx'];
%Note: TITLE string should be the same length.
%DATA=[C(I0),F(I0),T(I0),P(I0),N(I0),Q(I0),NOx(I0)];
%MAX_CH=7; %experiment data channel, adjustable, Input and Output Channel
%MAX_ORDER=[3;3;3;3;3;3;1];
%Two Different Search methods are used:
%1. ITERATION_NUM=[9]; All-in search iterations
%2. ITERATION_NUM=[2,3,3] %linear,quadratic,cubic terms are searched seperately
%Structure of F_partial: [F_partial of new selected reg, F_partial of No.1
%reg,F_partial of No.2reg,...]
%CAUTION!!output data must be in the last channel, sample numbers of all the channels
%must match
%====================Data Training====================
%Input
channel=MAX_CH; %experiment data channel, adjustable
if  size(DATA,1)<size(DATA,2)
disp('the vectors in DATA matrix should be coloum vectors')
return
end
%===============DEFINITIONS OF I/O CHANNELS=========
```

```
u=struct('var',{},'order',{},'data',{});

for ii=1:channel
u(ii).var=TITLE(ii,:);
u(ii).order=MAX_ORDER(ii,1);
u(ii).data=DATA(:,ii);
end
%====================Pool of Regressors===========
N=length(u(channel).data); %sample number
reg=struct('var',{},'data',{}); %Structure Definition
flag=struct('var',{},'data',{}); % flag of the regressors
num(1)=0; % linear regressor number
num(2)=0; % quadratic regressor number
num(3)=0; % cubic regressor number
%====================Linear Terms===============
%reg(i,j): i=1 represents original regressors, j is the index of the regressor
for ii=1:channel %if output terms are not included, set channel-1 to represent input
channels
for jj=1:u(ii).order
num(1)=num(1)+1;
reg(1,num(1)).var=strcat(u(ii).var,'(t-',num2str(jj),')');
reg(1,num(1)).data=delay(u(ii).data,jj);
flag(num(1)).var=reg(1,num(1)).var;
flag(num(1)).para=1; %set the flag
end
end
%======Quadratic Terms(Including Square and Cross product)=====
for ii=1:num(1)
for jj=ii:num(1)
num(2)=num(2)+1;
reg(1,num(1)+num(2)).var=strcat(reg(1,ii).var,'&',reg(1,jj).var);
reg(1,num(1)+num(2)).data=reg(1,ii).data.*reg(1,jj).data;
flag(num(1)+num(2)).var=reg(1,num(1)+num(2)).var;
flag(num(1)+num(2)).para=1;   %set the flag
end
end
%==============Cubic Terms(Including Square and Cross product)=====
for ii=1:num(1)
```

```
for jj=num(1)+1:5:num(1)+num(2) %1/5 of the cubic regressors
num(3)=num(3)+1;
reg(1,sum(num)).var=strcat(reg(1,ii).var,&,reg(1,jj).var);
reg(1,sum(num)).data=reg(1,ii).data.*reg(1,jj).data;
flag(sum(num)).var=reg(1,num(1)+num(2)+num(3)).var;
flag(sum(num)).para=1; %set the flag
end
end
%==========Variable Definitions=========================
a0=ones(N,1); %offset term
beta=struct('para',{}); %OLS estimator of dependend variable
X=struct('var','ones','matrix',a0); % Regression matrix
optimumreg=struct('var',{},'data',{}); %optimal regressor selected in each iteration
theta=struct('para',{}); %OLS estimator
r=struct('para',{}); % Correlation Factor
z=struct('para',{}); %Dependent Output Vatiable
Sjj=struct('para',{}); %variance of regressors*N
Szz=struct('para',{}); %variance of (dependent z)*N
maxr=struct('para',{});% Recorder of the maximum correlation
SSR=struct('para',{}); %Regression sum of squares
F_partial=struct('para',{});
sigmasquare=struct('para',{});
flagnum=0; % counter of selected regressors
z(1).para=u(channel).data; %Output data vector
% SST=z(1).para'*z(1).para-N*mean(z(1).para)^ 2;

F_in=5; % Forward Selection Criterion
F_out=4;% Backward Elimination Criterion
%structure used in backward elimination
XX=struct('var',{},'matrix',{});
thetaXX=struct('para',{});
X_nonewreg=struct('var',{},'matrix',{});
theta_nonewreg=struct('para',{});

if  size(ITERATION_NUM,2)==1 %all _in  seletion
%=================Main Iterations===============
for i=1:sum(ITERATION_NUM) %Iteration searching optimal regressors
%==========initial  condition  for  each  iteration=======
```

```
Szz(i).para=sum((z(i).para-mean(z(i).para)).^ 2);
Sjj(i,1).para=sum((reg(i,1).data-mean(reg(i,1).data)).^ 2);
r(i).para(1)=sum((reg(i,1).data-mean(reg(i,1).data)).*(z(i).para-mean(z(i).para))
./(sqrt(Sjj(i,1).para*Szz(i).para)));
maxr(i).para=abs(r(i).para(1));
% Start from the first regressor
optimumreg(i).var=reg(1,1).var;
optimumreg(i).data=reg(1,1).data;
t=1; % recorder of the index
%===== Searching the term with the maximum correlation factor==========
for j=1:sum(num)
if flag(j).para =0 %skip the turn if this regressor has been picked
% (i,j): ith iteration, jth regressor
Sjj(i,j).para=sum((reg(i,j).data-mean(reg(i,j).data)).^ 2);
%Corelation factor r(i).para(j), Only use dependent variable in correlation calculation
r(i).para(j)=sum((reg(i,j).data-mean(reg(i,j).data)).*(z(i).para-mean(z(i).para))./
(sqrt(Sjj(i,j).para*Szz(i).para)));
if maxr(i).para<abs(r(i).para(j)) % Find the regressor with highest correlation
maxr(i).para=abs(r(i).para(j));
t=j;
end
end
end
optimumreg(i).var=reg(1,t).var; %set the regressor
optimumreg(i).data=reg(1,t).data;
flag(t).para=0; % Remove the flag of the selected regressor
flagnum=flagnum+1; % a new regressor is selected


    %==========adding Regressor to X matrix==========
X.matrix=[X.matrix,optimumreg(i).data];
X.var=[X.var;optimumreg(i).var];
% scale the X matrix in order to avoid singularity problem
size_x = size(X.matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x;
% Calculate the parameters for the new model
```

```
theta(i).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*z(1).para;
%Unscaled theta and X matrix
theta(i).para=scale_x * theta(i).para;
X.matrix=X.matrix/scale_x;
%===============F-statistics calculation============
loop_switch=1; %swtich on the loop
while loop_switch==1
column=size(X.matrix,2); % Identify current regressor number in the X matrix
%==initializing the iteration================
if i==1   %When the first regressor is just added into the X matrix
sigmasquare(1).para=1/(N-1-1)*((z(1).para-X.matrix*theta(1).para)'*(z(1).para-
X.matrix*theta(1).para));
SSR(1).para(1)=(X.matrix*theta(i).para)'*z(1).para-N*mean(z(1).para)^ 2;
F_partial(1).para(1)=SSR(1).para(1)/sigmasquare(1).para;
% partial F_ratio for the 1st new regressor in 1st iteration
else   % a new regressor is added into X matrix
sigmasquare(i).para=1/(N-i-1)*((z(1).para-X.matrix*theta(i).para)'*(z(1).para-
X.matrix*theta(i).para));
SSR(i).para(1)=(X.matrix*theta(i).para)'*z(1).para-N*mean(z(1).para)^ 2;
%F_in for new added regressor with existed regressors in X matrix
F_partial(i).para(1)=(SSR(i).para(1)-SSR(i-1).para(1))/sigmasquare(i).para;
end
%==============Forward selection==================
if F_partial(i).para(1)>F_in
loop_switch=0; %switch off the recycle if the new added regressor meets F_in else
note=strcat('No.',num2str(i),' new regressor does not satify F_in');
display(note);
% Recover to previous X and Return the Optimized X, regressors and
%corresponding parameters without the newly added regressor
X.matrix = X.matrix(:,1:end-1);
X.var = X.var(1:end-1,:);
theta(i).para =theta(i-1).para;
OptimizedReg = X.var;
OptimizedX = X.matrix;
Optimizedtheta =theta(i).para;
y_est=OptimizedX*Optimizedtheta;
display(OptimizedReg);
display(' ');
```

```
return
end
%Check the Partial F_ratios of the rest of regressors after the new regressor is added
for k=2:column-1    %only works when column>=3, the first regressor column
%is defined as ones(:,1), hence at least two regressors are already in the model
%=====Construct the matrix XX without regressor======
%=======X.matrix(:,1)=a0=============
% Create matrix XX which is the X matrix without the kth regressor
% XX matrix is used for partial F-ratio calculation in order to reject
%the null hypothesis for the kth regressor
XX(i-1,k-1).matrix=[X.matrix(:,1:k-1),X.matrix(:,k+1:column)];
%reconstruct X matrix into XX(i-1,k-1) matrix
XX(i-1,k-1).var=[X.var(1:k-1,:);X.var(k+1:column,:)];
%name the regressors remain in the XX matrix
%==========Calculate theta without the kth regressor======
size_x1 = size(XX(i-1,k-1).matrix); cols_x1 = size_x1(2); scale_x1=zeros(cols_x1);
%size_x1 by size_x1 matrix filled by zeros
for w=1:cols_x1
scale_x1(w,w) = 1/norm(XX(i-1,k-1).matrix(:,w)); % diagonal scale factors
end
XX(i-1,k-1).matrix=XX(i-1,k-1).matrix*scale_x1;

thetaXX(i-1,k-1).para=(XX(i-1,k-1).matrix'*XX(i-1,k-1).matrix)^ (-1)*XX(i-1,k-1).matrix'*z(1).para;
thetaXX(i-1,k-1).para=scale_x1 * thetaXX(i-1,k-1).para;
XX(i-1,k-1).matrix=XX(i-1,k-1).matrix/scale_x1;
%note: In the partial F-ratio section, the regressors and dependent variable
% are in the original form
SSR(i).para(k)=(XX(i-1,k-1).matrix*thetaXX(i-1,k-1).para)'*z(1).para-N*mean(z(1).para).^ 2;
% The partial F_ratio of the regressors already in the model excluding
%the kth regressor(k=2:column-1)
F_partial(i).para(k)=(SSR(i).para(1)-SSR(i).para(k))/sigmasquare(i).para;
% SSR with all regressors minus SSR without (k-1)th regressor
end
%Update the vector of partial F_ratio in case of previously eliminated
% regressors being recorded
if length(F_partial(i).para)>column-1
F_partial(i).para=F_partial(i).para(1:column-1);
end
```

```
%==============Backward Elimination=================
[F_min, index_min] = min(F_partial(i).para); %identify the mininum F_partial
if F_min<F_out
%delete the regressor already in the model until all partial F_ratios of remaining
%regressor satisfies F_out
if index_min==1
X.matrix=X.matrix(:,1:end-1);
X.var=X.var(1:end-1,:);
else
X.matrix=[X.matrix(:,1:index_min-1),X.matrix(:,index_min+1:column)];
X.var=[X.var(1:index_min-1,:);X.var(index_min+1:column,:)];
end
%recalculate the parameters of the model after removing a regressor
size_x = size(X.matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x;
theta(i).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*z(1).para;
theta(i).para=scale_x * theta(i).para;
X.matrix=X.matrix/scale_x;


    loop_switch=1; %switch on the loop in order to refresh this cycle if a regressor is deleted
%recalculate SSR(i-1).para(1) with the new X matrix
X_nonewreg(i-1).matrix=X.matrix(:,1:end-1);
size_x = size(X_nonewreg(i-1).matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X_nonewreg(i-1).matrix(:,w));
end
X_nonewreg(i-1).matrix=X_nonewreg(i-1).matrix*scale_x;
theta_nonewreg(i-1).para=(X_nonewreg(i-1).matrix'*X_nonewreg(i-1).matrix)^ (-1)*
X_nonewreg(i-1).matrix'*z(1).para;
theta_nonewreg(i-1).para=scale_x * theta_nonewreg(i-1).para;
X_nonewreg(i-1).matrix=X_nonewreg(i-1).matrix/scale_x;
SSR(i-1).para(1)=(X_nonewreg(i-1).matrix*theta_nonewreg(i-1).para)'*z(1).para-
N*mean(z(1).para)^ 2;
%SSR(i).para(1) will be recalculated in the beginning of next loop in
%order to recalculate the F_partial ratios of all remaining regressors in X matrix
```

```
end
end

    %===Adjust all candidate regressors and dependent output variable
% with the offset parameter and previous added regressors====
%===reg(1,j)===The original regressors==================
%===reg(i,j)===The Adjusted regressors in iteration i============
%===z(i+1)====The Adjusted dependent output variable for the next iteration
for j=1:sum(num)
%express the new regressor with the previous regressors in X.matrix
size_x2 = size(X.matrix);cols_x2 = size_x2(2);scale_x2=zeros(cols_x2);
for w=1:cols_x2
scale_x2(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x2;
beta(i,j).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*reg(1,j).data;
beta(i,j).para=scale_x2 * beta(i,j).para;
X.matrix=X.matrix/scale_x2;
%Modifying the regressors based on beta parameters
reg(i+1,j).data=reg(1,j).data-X.matrix*beta(i,j).para;
reg(i+1,j).var=reg(1,j).var;
end
z(i+1).para=z(1).para-X.matrix*theta(i).para; %Dependent Variable
end

    else % Entering second catogory method
%=============================================
if  ITERATION_NUM(1)>0
%==========Linear Regressors=================
%====================Main Iterations============
for i=1:ITERATION_NUM(1) %Iteration searching optimal regressors
%==========initial condition for each iteration==========
Szz(i).para=sum((z(i).para-mean(z(i).para)).^ 2);
Sjj(i,1).para=sum((reg(i,1).data-mean(reg(i,1).data)).^ 2);
r(i).para(1)=sum((reg(i,1).data-mean(reg(i,1).data)).*(z(i).para-mean(z(i).para))
./(sqrt(Sjj(i,1).para*Szz(i).para)));
maxr(i).para=abs(r(i).para(1));
```

```
% Start from the first regressor
optimumreg(i).var=reg(1,1).var;
optimumreg(i).data=reg(1,1).data;
t=1; % recorder of the index
%==================================================
% Searching the term with the maximum correlation factor
for j=1:num(1)
if  flag(j).para =0 %skip the turn if this regressor has been picked
% (i,j):ith iteration, jth regressor
Sjj(i,j).para=sum((reg(i,j).data-mean(reg(i,j).data)).^ 2);
%Corelation factor r(i).para(j),Only use dependent variable in correlation calculation
r(i).para(j)=sum((reg(i,j).data-mean(reg(i,j).data)).*(z(i).para-mean(z(i).para))
./(sqrt(Sjj(i,j).para*Szz(i).para)));
if  maxr(i).para<abs(r(i).para(j)) % Find the regressor with highest correlation
maxr(i).para=abs(r(i).para(j));
t=j;
end
end
end
optimumreg(i).var=reg(1,t).var; %set the regressor
optimumreg(i).data=reg(1,t).data;
flag(t).para=0; % Remove the flag of the selected regressor
flagnum=flagnum+1; % a new regressor is selected
%==============adding Regressor to X matrix===============
X.matrix=[X.matrix,optimumreg(i).data];
X.var=[X.var;optimumreg(i).var];
% scale the X matrix in order to avoid singularity problem
size_x = size(X.matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x;
% Calculate the parameters for the new model
theta(i).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*z(1).para;
%Unscaled theta and X matrix
theta(i).para=scale_x * theta(i).para;
X.matrix=X.matrix/scale_x;
%============F-statistics calculation===============
```

```
loop_switch=1; %swtich on the loop
while loop_switch==1
column=size(X.matrix,2); % Identify current regressor number in the X matrix
%==initializing the iteration================
if  i==1 %When the first regressor is just added into the X matrix
sigmasquare(1).para=1/(N-1-1)*((z(1).para-X.matrix*theta(1).para)'*
(z(1).para-X.matrix*theta(1).para));
SSR(1).para(1)=(X.matrix*theta(i).para)'*z(1).para-N*mean(z(1).para)^ 2;
F_partial(1).para(1)=SSR(1).para(1)/sigmasquare(1).para;
% partial F_ratio for the 1st new regressor in 1st iteration
else % a new regressor is added into X matrix
sigmasquare(i).para=1/(N-i-1)*((z(1).para-X.matrix*theta(i).para)'*(z(1).para-
X.matrix*theta(i).para));
SSR(i).para(1)=(X.matrix*theta(i).para)'*z(1).para-N*mean(z(1).para)^ 2;
%F_in for new added regressor with existed regressors in X matrix
F_partial(i).para(1)=(SSR(i).para(1)-SSR(i-1).para(1))/sigmasquare(i).para;
end
%==============Forward Selection=======================
if  F_partial(i).para(1)>F_in
loop_switch=0; %switch off the recycle if the new added regressor meets F_in
else
note=strcat('No.',num2str(i),' new regressor does not satify F_in');
display(note);
% Recover to previous X and Return the Optimized X, regressors and corresponding
%parameters without the newly added regressor
X.matrix = X.matrix(:,1:end-1);
X.var = X.var(1:end-1,:);
theta(i).para =theta(i-1).para;
OptimizedReg = X.var;
OptimizedX = X.matrix;
Optimizedtheta =theta(i).para;
y_est=OptimizedX*Optimizedtheta;
display(OptimizedReg);
display(' ');
% display('Original Model self-validation:');
error_est=gfit2(DATA(:,size(DATA,2)),y_est,'all');
% display(' ');
return
```

```
end
for k=2:column-1 %only works when column>=3, the first regressor column is defined as
ones(:,1), %hence at least two regressors are already in the model
%============Construct the matrix XX without kth regressor
%=======X.matrix(:,1)=a0==================
% Create matrix XX which is the X matrix without the kth regressor
% XX matrix is used for partial F-ratio calculation in order to reject the null
hypothesis %for the kth regressor
XX(i-1,k-1).matrix=[X.matrix(:,1:k-1),X.matrix(:,k+1:column)];
%reconstruct X matrix into XX(i-1,k-1) matrix
XX(i-1,k-1).var=[X.var(1:k-1,:);X.var(k+1:column,:)];
%name the regressors remain in the XX matrix
%====Calculate theta without the kth regressor======
size_x1 = size(XX(i-1,k-1).matrix); cols_x1 = size_x1(2); scale_x1=zeros(cols_x1);
%size_x1 by size_x1 matrix filled by zeros
for w=1:cols_x1
scale_x1(w,w) = 1/norm(XX(i-1,k-1).matrix(:,w)); % diagonal scale factors
end
XX(i-1,k-1).matrix=XX(i-1,k-1).matrix*scale_x1;
thetaXX(i-1,k-1).para=(XX(i-1,k-1).matrix'*XX(i-1,k-1).matrix)^ (-1)*XX(i-1,k-1)
.matrix'*z(1).para;
thetaXX(i-1,k-1).para=scale_x1 * thetaXX(i-1,k-1).para;
XX(i-1,k-1).matrix=XX(i-1,k-1).matrix/scale_x1;
%NOTE: In the partial F-ratio section, the regressors and dependent variable are
%in the original form
SSR(i).para(k)=(XX(i-1,k-1).matrix*thetaXX(i-1,k-1).para)'*z(1).para
-N*mean(z(1).para).^ 2;
% The partial F_ratio of the regressors already in the model excluding the kth
%regressor(k=2:column-1)
F_partial(i).para(k)=(SSR(i).para(1)-SSR(i).para(k))/sigmasquare(i).para;
% SSR with all regressors minus SSR without (k-1)th regressor
end
%Update the vector of partial F_ratio in case of previously eliminated
%regressors being recorded
if  length(F_partial(i).para)>column-1
F_partial(i).para=F_partial(i).para(1:column-1);
end
%============Backward  Elimination============
```

$[F\_min, index\_min] = min(F\_partial(i).para)$;
if F_min<F_out
if index_min==1 %delect the newly selected regressor
X.matrix=X.matrix(:,1:end-1);
X.var=X.var(1:end-1,:);
else
X.matrix=[X.matrix(:,1:index_min-1),X.matrix(:,index_min+1:column)];
X.var=[X.var(1:index_min-1,:);X.var(index_min+1:column,:)];
end
%recalculate the parameters of the model after removing a regressor
size_x = size(X.matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x;
theta(i).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*z(1).para;
theta(i).para=scale_x * theta(i).para;
X.matrix=X.matrix/scale_x;
loop_switch=1; %switch on the loop in order to refresh this cycle if a regressor is deleted
%recalculate SSR(i-1).para(1) with the new X matrix
X_nonewreg(i-1).matrix=X.matrix(:,1:end-1);
size_x = size(X_nonewreg(i-1).matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X_nonewreg(i-1).matrix(:,w));
end
X_nonewreg(i-1).matrix=X_nonewreg(i-1).matrix*scale_x;
theta_nonewreg(i-1).para=(X_nonewreg(i-1).matrix'*X_nonewreg(i-1).matrix)^ (-1)*X_nonewreg(i-1).matrix'*z(1).para;
theta_nonewreg(i-1).para=scale_x * theta_nonewreg(i-1).para;
X_nonewreg(i-1).matrix=X_nonewreg(i-1).matrix/scale_x;
SSR(i-1).para(1)=(X_nonewreg(i-1).matrix*theta_nonewreg(i-1).para)'*z(1).para-N*mean(z(1).para)^ 2;
end
end
for j=1:sum(num)
size_x2 = size(X.matrix);cols_x2 = size_x2(2);scale_x2=zeros(cols_x2);
for w=1:cols_x2
scale_x2(w,w) = 1/norm(X.matrix(:,w));

```
end
X.matrix=X.matrix*scale_x2;
beta(i,j).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*reg(1,j).data;
beta(i,j).para=scale_x2 * beta(i,j).para;
X.matrix=X.matrix/scale_x2;
reg(i+1,j).data=reg(1,j).data-X.matrix*beta(i,j).para;
reg(i+1,j).var=reg(1,j).var;
end
z(i+1).para=z(1).para-X.matrix*theta(i).para;
end
end
%==========quadratic terms================
if ITERATION_NUM(2)>0
%==========Main Iterations=========
for i=ITERATION_NUM(1)+1:ITERATION_NUM(1)+ITERATION_NUM(2) %Iteration searching
% optimal regressors
Szz(i).para=sum((z(i).para-mean(z(i).para)).^ 2);
Sjj(i,1).para=sum((reg(i,1).data-mean(reg(i,1).data)).^ 2);
r(i).para(1)=sum((reg(i,1).data-mean(reg(i,1).data)).*(z(i).para-mean(z(i).para))
./(sqrt(Sjj(i,1).para*Szz(i).para)));
maxr(i).para=abs(r(i).para(1));
optimumreg(i).var=reg(1,1).var;
optimumreg(i).data=reg(1,1).data;
t=1;
%===== Searching the term with the maximum correlation factor========
for j=num(1)+1:num(1)+num(2)
if flag(j).para =0
Sjj(i,j).para=sum((reg(i,j).data-mean(reg(i,j).data)).^ 2);
r(i).para(j)=sum((reg(i,j).data-mean(reg(i,j).data)).*(z(i).para-mean(z(i).para))
./(sqrt(Sjj(i,j).para*Szz(i).para)));
if maxr(i).para<abs(r(i).para(j))
maxr(i).para=abs(r(i).para(j));
t=j;
end
end
end
optimumreg(i).var=reg(1,t).var; %set the regressor
```

```
optimumreg(i).data=reg(1,t).data;
flag(t).para=0; % Remove the flag of the selected regressor
flagnum=flagnum+1; % a new regressor is selected


    %========adding Regressor to X matrix=========
X.matrix=[X.matrix,optimumreg(i).data];
X.var=[X.var;optimumreg(i).var];
size_x = size(X.matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x;
theta(i).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*z(1).para;
theta(i).para=scale_x * theta(i).para;
X.matrix=X.matrix/scale_x;
%======F-statistics calculation==========
loop_switch=1; %swtich on the loop
while loop_switch==1
column=size(X.matrix,2); % Identify current regressor number in the X matrix
%==initializing the iteration================
if i==1 %When the first regressor is just added into the X matrix
sigmasquare(1).para=1/(N-1-1)*((z(1).para-X.matrix*theta(1).para)'*(z(1).para-
X.matrix*theta(1).para));
SSR(1).para(1)=(X.matrix*theta(i).para)'*z(1).para-N*mean(z(1).para)^ 2;
F_partial(1).para(1)=SSR(1).para(1)/sigmasquare(1).para;
else
sigmasquare(i).para=1/(N-i-1)*((z(1).para-X.matrix*theta(i).para)'*(z(1).para-
X.matrix*theta(i).para));
SSR(i).para(1)=(X.matrix*theta(i).para)'*z(1).para-N*mean(z(1).para)^ 2;
F_partial(i).para(1)=(SSR(i).para(1)-SSR(i-1).para(1))/sigmasquare(i).para;
end
%==============Forward Selection================
if F_partial(i).para(1)>F_in
loop_switch=0;
else
note=strcat('No.',num2str(column-1),' new regressor does not satify F_in');
display(note);
X.matrix = X.matrix(:,1:end-1);
```

```
X.var = X.var(1:end-1,:);
theta(i).para =theta(i-1).para;
OptimizedReg = X.var;
OptimizedX = X.matrix;
Optimizedtheta =theta(i).para;
y_est=OptimizedX*Optimizedtheta;
display('F-statistic selection:');
display(OptimizedReg);
return
end
for k=2:column-1
XX(i-1,k-1).matrix=[X.matrix(:,1:k-1),X.matrix(:,k+1:column)];
XX(i-1,k-1).var=[X.var(1:k-1,:);X.var(k+1:column,:)];
size_x1 = size(XX(i-1,k-1).matrix); cols_x1 = size_x1(2); scale_x1=zeros(cols_x1);
%size_x1 by size_x1 matrix filled by zeros
for w=1:cols_x1
scale_x1(w,w) = 1/norm(XX(i-1,k-1).matrix(:,w));
end
XX(i-1,k-1).matrix=XX(i-1,k-1).matrix*scale_x1;
thetaXX(i-1,k-1).para=(XX(i-1,k-1).matrix'*XX(i-1,k-1).matrix)^ (-1)*XX(i-1,k-1).matrix'*z(1).para;
thetaXX(i-1,k-1).para=scale_x1 * thetaXX(i-1,k-1).para;
XX(i-1,k-1).matrix=XX(i-1,k-1).matrix/scale_x1;
SSR(i).para(k)=(XX(i-1,k-1).matrix*thetaXX(i-1,k-1).para)'*z(1).para-N*mean(z(1).para).^ 2;
F_partial(i).para(k)=(SSR(i).para(1)-SSR(i).para(k))/sigmasquare(i).para;
end
if  length(F_partial(i).para)>column-1
F_partial(i).para=F_partial(i).para(1:column-1);
end
%==============Backward  Elimination========
[F_min, index_min] = min(F_partial(i).para);
if  F_min<F_out
if  index_min==1
X.matrix=X.matrix(:,1:end-1);
X.var=X.var(1:end-1,:);
else
X.matrix=[X.matrix(:,1:index_min-1),X.matrix(:,index_min+1:column)];
X.var=[X.var(1:index_min-1,:);X.var(index_min+1:column,:)];
end
```

```
size_x = size(X.matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x;
theta(i).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*z(1).para;
theta(i).para=scale_x * theta(i).para;
X.matrix=X.matrix/scale_x;
loop_switch=1;
X_nonewreg(i-1).matrix=X.matrix(:,1:end-1);
size_x = size(X_nonewreg(i-1).matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X_nonewreg(i-1).matrix(:,w));
end
X_nonewreg(i-1).matrix=X_nonewreg(i-1).matrix*scale_x;
theta_nonewreg(i-1).para=(X_nonewreg(i-1).matrix'*X_nonewreg(i-1).matrix)^ (-1)*X_nonewreg(i-
1).matrix'*z(1).para;
theta_nonewreg(i-1).para=scale_x * theta_nonewreg(i-1).para;
X_nonewreg(i-1).matrix=X_nonewreg(i-1).matrix/scale_x;
SSR(i-1).para(1)=(X_nonewreg(i-1).matrix*theta_nonewreg(i-1).para)'*z(1).para-
N*mean(z(1).para)^ 2;
end
end
for j=1:sum(num) %express the new regressor with the previous regressors in X.matrix
size_x2 = size(X.matrix);cols_x2 = size_x2(2);scale_x2=zeros(cols_x2);
for w=1:cols_x2
scale_x2(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x2;
beta(i,j).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*reg(1,j).data;
beta(i,j).para=scale_x2 * beta(i,j).para;
X.matrix=X.matrix/scale_x2;
%Modifying the regressors based on beta parameters
reg(i+1,j).data=reg(1,j).data-X.matrix*beta(i,j).para;
reg(i+1,j).var=reg(1,j).var;
end
z(i+1).para=z(1).para-X.matrix*theta(i).para; %Dependent Variable
end
```

```
end
%==========cubic terms==================
if ITERATION_NUM(3)>0

    %==============Main Iterations============
for i=ITERATION_NUM(1)+ITERATION_NUM(2)+1:sum(ITERATION_NUM) %Iteration searching optimal regressors
%==========initial condition for each iteration====
Szz(i).para=sum((z(i).para-mean(z(i).para)).^ 2);
Sjj(i,1).para=sum((reg(i,1).data-mean(reg(i,1).data)).^ 2);
r(i).para(1)=sum((reg(i,1).data-mean(reg(i,1).data)).*(z(i).para-mean(z(i).para))
./(sqrt(Sjj(i,1).para*Szz(i).para)));
maxr(i).para=abs(r(i).para(1));
optimumreg(i).var=reg(1,1).var;
optimumreg(i).data=reg(1,1).data;
t=1; % recorder of the index
%===== Searching the term with the maximum correlation factor======
for j=num(1)+num(2)+1:sum(num)
if flag(j).para =0
% (i,j):ith iteration, jth regressor Sjj(i,j).para=sum((reg(i,j).data-mean(reg(i,j).data)).^ 2);
r(i).para(j)=sum((reg(i,j).data-mean(reg(i,j).data)).*(z(i).para-mean(z(i).para))./
(sqrt(Sjj(i,j).para*Szz(i).para)));

    if maxr(i).para<abs(r(i).para(j))
maxr(i).para=abs(r(i).para(j));
t=j;
end
end
end
optimumreg(i).var=reg(1,t).var;
optimumreg(i).data=reg(1,t).data;
flag(t).para=0;
flagnum=flagnum+1;

    %=======adding Regressor to X matrix=======
X.matrix=[X.matrix,optimumreg(i).data];
X.var=[X.var;optimumreg(i).var];
```

```
% scale the X matrix in order to avoid singularity problem
size_x = size(X.matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x;
% Calculate the parameters for the new model
theta(i).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*z(1).para;
%Unscaled theta and X matrix
theta(i).para=scale_x * theta(i).para;
X.matrix=X.matrix/scale_x;
loop_switch=1;
while loop_switch==1
column=size(X.matrix,2);
%==initializing the iteration=======
if i==1
sigmasquare(1).para=1/(N-1-1)*((z(1).para-X.matrix*theta(1).para)'*(z(1).para-
X.matrix*theta(1).para));
SSR(1).para(1)=(X.matrix*theta(i).para)'*z(1).para-N*mean(z(1).para)^ 2;
F_partial(1).para(1)=SSR(1).para(1)/sigmasquare(1).para;
else
sigmasquare(i).para=1/(N-i-1)*((z(1).para-X.matrix*theta(i).para)'*(z(1).para-
X.matrix*theta(i).para));
SSR(i).para(1)=(X.matrix*theta(i).para)'*z(1).para-N*mean(z(1).para)^ 2;
F_partial(i).para(1)=(SSR(i).para(1)-SSR(i-1).para(1))/sigmasquare(i).para;
end
%==============Forward Selection==============
if F_partial(i).para(1)>F_in
loop_switch=0;
else
note=strcat('No.',num2str(i),' new regressor does not satify F_in');
display(note);
% Recover to previous X and Return the Optimized X, regressors and
%corresponding parameters without the newly added regressor
X.matrix = X.matrix(:,1:end-1);
X.var = X.var(1:end-1,:);
theta(i).para =theta(i-1).para;
OptimizedReg = X.var;
```

```
OptimizedX = X.matrix;
Optimizedtheta =theta(i).para;
y_est=OptimizedX*Optimizedtheta;
display(OptimizedReg);
display(' ');
return
end
%Check the Partial F_ratios of the rest of regressors after the new regressor is added========
for k=2:column-1 %only works when column>=3, the first regressor column is
%defined as ones(:,1), hence at least two regressors are already in the model
%Construct the matrix XX without regressor
%=======X.matrix(:,1)=a0==============
% Create matrix XX which is the X matrix without the kth regressor
% XX matrix is used for partial F-ratio calculation in order to reject the
%null hypothesis for the kth regressor
XX(i-1,k-1).matrix=[X.matrix(:,1:k-1),X.matrix(:,k+1:column)];
%reconstruct X matrix into XX(i-1,k-1) matrix
XX(i-1,k-1).var=[X.var(1:k-1,:);X.var(k+1:column,:)];
%name the regressors remain in the XX matrix
%=======Calculate theta without the kth regressor======
size_x1 = size(XX(i-1,k-1).matrix); cols_x1 = size_x1(2); scale_x1=zeros(cols_x1);
for w=1:cols_x1
scale_x1(w,w) = 1/norm(XX(i-1,k-1).matrix(:,w)); % diagonal scale factors
end
XX(i-1,k-1).matrix=XX(i-1,k-1).matrix*scale_x1;
thetaXX(i-1,k-1).para=(XX(i-1,k-1).matrix'*XX(i-1,k-1).matrix)^ (-1)
XX(i-1,k-1).matrix'*z(1).para;
thetaXX(i-1,k-1).para=scale_x1 * thetaXX(i-1,k-1).para;
XX(i-1,k-1).matrix=XX(i-1,k-1).matrix/scale_x1;
%Note: In the partial F-ratio section, the regressors and dependent variable
%are in the original form
SSR(i).para(k)=(XX(i-1,k-1).matrix*thetaXX(i-1,k-1).para)'*z(1).para-N*mean(z(1).para).^ 2;
% The partial F_ratio of the regressors already in the model
%excluding the kth regressor(k=2:column-1)
F_partial(i).para(k)=(SSR(i).para(1)-SSR(i).para(k))/sigmasquare(i).para;
% SSR with all regressors minus SSR without (k-1)th regressor
end
```

```
if  length(F_partial(i).para)>column-1
F_partial(i).para=F_partial(i).para(1:column-1);
end
%=========Backward Elimination=========
[F_min, index_min] = min(F_partial(i).para);
if  F_min<F_out
if  index_min==1
X.matrix=X.matrix(:,1:end-1);
X.var=X.var(1:end-1,:);
else
X.matrix=[X.matrix(:,1:index_min-1),X.matrix(:,index_min+1:column)];
X.var=[X.var(1:index_min-1,:);X.var(index_min+1:column,:)];
end
%recalculate the parameters of the model after removing a regressor
size_x = size(X.matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x;
theta(i).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*z(1).para;
theta(i).para=scale_x * theta(i).para;
X.matrix=X.matrix/scale_x;
loop_switch=1;
X_nonewreg(i-1).matrix=X.matrix(:,1:end-1);
size_x = size(X_nonewreg(i-1).matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x
scale_x(w,w) = 1/norm(X_nonewreg(i-1).matrix(:,w));
end
X_nonewreg(i-1).matrix=X_nonewreg(i-1).matrix*scale_x;
theta_nonewreg(i-1).para=(X_nonewreg(i-1).matrix'*X_nonewreg(i-1).matrix)^ (-1)*X_nonewreg(i-
1).matrix'*z(1).para;
theta_nonewreg(i-1).para=scale_x * theta_nonewreg(i-1).para;
X_nonewreg(i-1).matrix=X_nonewreg(i-1).matrix/scale_x;
SSR(i-1).para(1)=(X_nonewreg(i-1).matrix*theta_nonewreg(i-1).para)'*z(1).para-
N*mean(z(1).para)^ 2;
end
end
```

```
%Adjust all candidate regressors and dependent output variable
%with the offset parameter and previous added regressors
%===reg(1,j)===The original regressors============
%===reg(i,j)===The Adjusted regressors in iteration i=========
%===z(i+1)====The Adjusted dependent output variable for the next iteration
for j=1:sum(num)
%express the new regressor with the previous regressors in X.matrix
size_x2 = size(X.matrix);cols_x2 = size_x2(2);scale_x2=zeros(cols_x2);
for w=1:cols_x2
scale_x2(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x2;
beta(i,j).para=(X.matrix'*X.matrix)^ (-1)*X.matrix'*reg(1,j).data;
beta(i,j).para=scale_x2 * beta(i,j).para;
X.matrix=X.matrix/scale_x2;
%Modifying the regressors based on beta parameters
reg(i+1,j).data=reg(1,j).data-X.matrix*beta(i,j).para;
reg(i+1,j).var=reg(1,j).var;
end
z(i+1).para=z(1).para-X.matrix*theta(i).para; %Dependent Variable
end
end
end
% Final step, Return the structure if all the regressor meets F_in
note=strcat('All (',num2str(column-1),') regressors satify F_in');
display(note);
OptimizedX = X.matrix;
OptimizedReg = X.var;
Optimizedtheta =theta(i).para;
y_est=OptimizedX*Optimizedtheta;
display(OptimizedReg);
display(' ');
return
%==========coefficients calculation==========
Rsquare(i)=SSR(i).para(1)/SST;
Rsquare_adjusted(i)=1-((1-SSR(i).para(1)/SST)*(N-1)/(N-column));
%Adjusted multiple correlation coefficient of determination
RSS(i)=sum((trq0-X.matrix*theta(i).para).^ 2);
```

```
Cp(i)=RSS(i)/sigmasquare(i).para-(N-2*column); %Mallow's Cp statistic
TSS(i)=sum((trq0-mean(trq0)).^ 2);
OVF(i)=(TSS-RSS)/RSS*(N-column)/(column-1);% Overall F-test
```

.2 B. M-Function for Orthogonal Least Squares with ERR

```
function [ERRReg,ERRX,ERRtheta,maxERR,flag,error_est,y_est,reg,w0,a,g,
orthogonal_theta,orthogonal_y]
=ERR_selection(MAX_CH,MAX_ORDER,TITLE,DATA,ITERATION_NUM)
%Inputs of the function:
%TITLE=[' C ';' F ';' Tb ';' Pm';' N ';' Q ';'NOx'];
%Note: TITLE string should be the same length.
%DATA=[C(I0),F(I0),T(I0),P(I0),N(I0),Q(I0),NOx(I0)];
%MAX_CH=7; %experiment data channel, adjustable, Input and Output Channel
%MAX_ORDER=[3;3;3;3;3;3;1];
%ITERATION_NUM=[9];  All-in search iterations
%CAUTION!!output data must be in the last channel, sample numbers of all the
%channels must match
%====================Data Training==================
%Input
channel=MAX_CH;  %experiment data channel, adjustable
if size(DATA,1)¡size(DATA,2)
disp('the vectors in DATA matrix should be coloum vectors')
return
end
%=============DEFINITIONS OF I/O CHANNELS=============
u=struct('var',{},'order',{},'data',{});
for ii=1:channel
u(ii).var=TITLE(ii,:);
u(ii).order=MAX_ORDER(ii,1);
u(ii).data=DATA(:,ii);
end
%================Pool of Regressors=================
N=length(u(channel).data);  %sample number
reg=struct('var',{},'data',{});  %Structure Definition
flag=struct('var',{},'para',{});  % flag of the regressors
num(1)=0;  % linear regressor number
```

```
num(2)=0;  % nonlinear regressor number
num(3)=0;  % cubic regressor number
%====================Linear Terms===============
for ii=1:channel  %if output terms are not included, set channel-1 to represent input channels for jj=1:u(ii).order
num(1)=num(1)+1;
reg(1,num(1)).var=strcat(u(ii).var,'(t-',num2str(jj),')');
reg(1,num(1)).data=delay(u(ii).data,jj);
flag(num(1)).var=reg(1,num(1)).var;
flag(num(1)).para=1;  %set the flag
end
end
%======Quadratic Terms(Including Square and Cross product)=======
for ii=1:num(1)
for jj=ii:num(1)
num(2)=num(2)+1;
reg(1,num(1)+num(2)).var=strcat(reg(1,ii).var,'&',reg(1,jj).var);
reg(1,num(1)+num(2)).data=reg(1,ii).data.*reg(1,jj).data;
flag(num(1)+num(2)).var=reg(1,num(1)+num(2)).var;
flag(num(1)+num(2)).para=1;  %set the flag
end
end
%==========Cubic Terms(Including Square and Cross product)=====
for ii=1:num(1)
for jj=num(1)+1:5:num(1)+num(2)  %1/5 of the cubic regressors
num(3)=num(3)+1;
reg(1,sum(num)).var=strcat(reg(1,ii).var,'&',reg(1,jj).var);
reg(1,sum(num)).data=reg(1,ii).data.*reg(1,jj).data;
flag(sum(num)).var=reg(1,num(1)+num(2)+num(3)).var;
flag(sum(num)).para=1;  %set the flag
end
end
a0=ones(N,1);  %offset term
X=struct('var','ones','matrix',a0);
optimumreg=struct('var',{},'data',{});
ERR=struct('para',{});  %Error Reduction Ratio
w=struct('para',{});  % orthogonalized regressor
w0=struct('para',{});  % orthogonal basis wk
```

```
z=struct('data',{});  %Dependent Output Vatiable
maxERR=struct('para',{});
z(1).data=u(channel).data;
M=sum(num);


    if size(ITERATION_NUM,2)==1   %all_in seletion
%====Step 1==============
for i=1:M
w(i).para=reg(1,i).data;
ERR(1).para(i)=((z(1).data'*w(i).para)^2)/((z(1).data'*z(1).data)*(w(i).para'*w(i).para));
a(1,1)=1;
end
[maxERR(1).para,flagnum]=max(ERR(1).para); %Find the regressor with the largest ERR
w0(1,1).para=reg(1,flagnum).data;  %the first orthogonal basis
optimumreg(1).data=reg(1,flagnum).data;  %the corresponding selected regressor
optimumreg(1).var=reg(1,flagnum).var;
g.para(1)=(z(1).data'*w0(1).para) ^ 2 / (w0(1).para'*w0(1).para);
tau(1)=w0(1).para'*w0(1).para;
X.matrix=[X.matrix,optimumreg(1).data];
X.var=[X.var;optimumreg(1).var];
flag(flagnum).var=optimumreg(1).var;
flag(flagnum).para=0;  %mark as selected
%==========Step 2: iter>=2================
iter=2;  % iteration number
while iter<=ITERATION_NUM  % condition
for i=1:M
if flag(i).para==1  %only test the regressors satisfying the criterion w'*w>10e-10
%=====regressor orthogonalization for this iteration=====
sum_oth=zeros(N,1);
for k=1:size(X.matrix,2)-1
sum_oth= sum_oth +(reg(1,i).data'*w0(1,k).para)/(w0(1,k).para'*w0(1,k).para) * w0(k).para;
end
reg(iter,i).data=reg(1,i).data-sum_oth;  %Orthogonalized candidate basis function
%========Calculate the ERR in each iteration=========
ERR(iter).para(i)=(z(1).data'*reg(iter,i).data)^ 2 / ((z(1).data'*z(1).data)*
(reg(iter,i).data'*reg(iter,i).data));
if (reg(iter,i).data'*reg(iter,i).data)<10e-10
```

```
flag(i).para=0;  %eliminate the candidate basis function which are less than
%the threshold tau to avoid ill conditioning
ERR(iter).para(i)=0;
end
else
ERR(iter).para(i)=0;
end
end
%=====Locate the orthogonal basis function w0 with the largest ERR====
[maxERR(iter).para,flagnum]=max(ERR(iter).para);  %memorize the selected regressor
optimumreg(iter).data=reg(1,flagnum).data;  %selected candidate regressor
optimumreg(iter).var=reg(1,flagnum).var;
w0(1,iter).para=reg(iter,flagnum).data;  % orthogonal basis function
g.para(iter)=(z(1).data'*w0(1,iter).para)^ 2/(w0(1,iter).para'*w0(1,iter).para);
tau(iter)=w0(iter).para'*w0(iter).para;
a(iter,iter)=1;  %diagonal Element in A matrix
for k=1:size(X.matrix,2)-1
a(k,iter)=(w0(1,k).para'*reg(1,flagnum).data)/(w0(1,k).para'*w0(1,k).para);
end
%=======================================================
X.matrix=[X.matrix,optimumreg(iter).data];
X.var=[X.var;optimumreg(iter).var];
flag(flagnum).para=0;  % mark as selected
iter=iter+1;  %next iteration
end
orthogonal_theta=a*(g.para');
orthogonal_y=X.matrix(:,2:end)*orthogonal_theta;
size_x = size(X.matrix);cols_x = size_x(2);scale_x=zeros(cols_x);
for w=1:cols_x; scale_x(w,w) = 1/norm(X.matrix(:,w));
end
X.matrix=X.matrix*scale_x;
ERRtheta=(X.matrix'*X.matrix)^(-1)*X.matrix'*z(1).data;
ERRtheta=scale_x*ERRtheta;
X.matrix=X.matrix/scale_x;
ERRX=X.matrix;
ERRReg=X.var;
y_est=X.matrix*ERRtheta;
display('ERR selection:');
```

```
display(ERRReg);
end
```