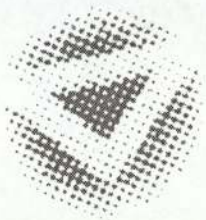
Neural Networks Applied to Ignition Timing Calibration

Krzysztof Zapart

Master of Science by Research
in Pattern Analysis and Neural Networks



The University of Aston in Birmingham

September 1996

THE UNIVERSITY OF ASTON IN BIRMINGHAM

Neural Networks Applied to Ignition Timing Calibration

KRZYSZTOF ZAPART

Master of Science by Research
in Pattern Analysis and Neural Networks, 1996

Thesis Summary

The present control methods for combustion parameters in engine management systems, such as ignition timing or desired air-to-fuel ratio, are based on "look-up" tables. Optimised engine parameters are accessed only for specific input values. In the intermediate points the output parameters are linearly interpolated, which results in the engine running in sub-optimal conditions.

This thesis discusses the feasibility of replacing these look-up tables with non-linear mappings produced by artificial neural networks.

The thesis reports the experiments for two data sets collected from two Rover internal combustion engines.

The preliminary experiments carried out for the air-to-fuel ratio and the ignition timing data show that non-linear models, such as the Radial Basis Function Networks, Multilayer Perceptron and the committees of these networks, can be used to produce smooth and accurate mappings of the engine parameters.

The neural network approach is feasible and provides a new and more efficient way to handle the problem of controlling the engine parameters.

The thesis also reports on the C++ neural network library created for use in this project.

Keywords: artificial neural networks, internal combustion engines, automatic calibration of the engine parameters

Acknowledgements

I would like to thank my supervisor, Professor David Lowe, for his help and words of encouregment that led me to completion of this thesis. I would also express thanks to Hector Sindano from Sagem Ltd., for his support and help with data collection.

I would also thank Cazhaow for fruitful discussions on Bayesian error bars.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

The problem of maintaining the optimal control of engine parameters has large implications in such diverse areas as environmental concerns and personal satisfaction of customers driving the cars.

While an internal combustion engine, widely used in vehicles, is running in non-optimal conditions it is producing an excessive amount of pollutants and may be achieving an inferior performance in terms of the output power and fuel consumption. This suggests that there is a strong need for a better calibration and control of the engine parameters, which can also help to improve the comfort of driving a vehicle.

The task of calibrating the engine management system can be divided into the following parts: fuelling, ignition, air management and emission (fuel vapour purge control and exhaust gas recirculation).

This thesis tries to address the problem of the ignition timing calibration and also to consider the issue of improving the fuelling section (controlling the air-to-fuel ratio).

The traditional way of controlling the engine parameters is to map the desired ignition timing and air/fuel ratio values as a function of speed, load and some other inputs (e.g. throttle change rate or coolant temperature). Because of the high cost of obtaining the data points on the test bed and the problem of the curse of dimensionality [1], one cannot fully span the whole input space. A large number of measurements would be required to accomplish this task. The large number of points measured implies that the mappings, which would contain these points, would need much more

memory to store in the EEPROM of electronic control system. This restricts their use in the engine management system, which is governed by very tight memory requirements.

Because of the above-mentioned problems, the mappings contain a number of gaps and uncovered regions. These "blank" spaces are next filled in manually by engineers who make judgements based on their experience and engineering intuition.

When an engine is operating the mappings are transformed from discrete to a continuous form using a linear interpolation between points in the tables.

This is hence a "table look-up" approach with local linear interpolation, which effectively tries to approximate the real smooth control surface.

Such a method introduces differential discontinuities to the mappings. As the result, the engine is not running in an optimal, smooth and efficient way.

Artificial neural networks are naturally suited to perform complex non-linear mappings [1] and therefore, it would be interesting to replace the tables currently in use with smooth mappings provided by neural networks.

This thesis forms a preliminary work carried out to test the feasibility of replacing linearly interpolated tables with neural networks.

1.2 Previous work

Recently there has been some interest in the use of neural networks in the domain of calibration and control of internal combustion engines, which resulted in a number of successful applications.

In [9] Stevens *et al.*, working in collaboration with the Ford Motor Company, have mapped the precatalysts, such as NO_x, CO, and HC rate, against an engine speed, load, air-to-fuel ratio, exhaust gas recirculation and a coolant temperature.

Having these interpolating maps, one can construct improved "look-up" tables for the air-to-fuel ratio and the ignition timing. The neural network can predict the pollution levels as a function of the A/F ratio and ignition timing at any given values of load, speed and other parameters. It is therefore possible to fill in the gaps in the

"look-up" tables. This approach also allows the engineers to use less measurements and to compensate for the insufficient quantity of data.

Another approach to the problem of automatic calibration has been proposed by Morita in [5].

In his recent work he considered making the neural mappings adapt over time to slight changes due to engine aging and to individual differences of characteristics among engines of the same type.

The mappings realised by neural models were the air-to-fuel ratio and ignition timing, based on the engine intake air mass flow and speed.

There were two generic types of those mappings: off-line and on-line.

In the off-line case, after training the pre-optimised networks were fixed and used as improved look-up tables. The on-line case is the most interesting one. These are the networks designed to change over time.

In general, it is difficult to realise systems that adapt itself to engine aging because the mass-market vehicles do not contain sensors able to detect such changes. It is also hard to evaluate in which direction and how the control surfaces should be modified in such cases.

The goal is to keep the objective performance index J optimum by the feedback control. The index J includes such parameters as the degree of thermal efficiency or the degree of variation of an engine speed.

If for a given operating point the performance index significantly decreases, the back-propagation algorithm is used to slightly alter the neural mapping.

Yet there is a major obstacle in practically implementing this method. The on-line adaptation of the networks with a back-propagation algorithm is not fast enough and therefore cannot be implemented in the engine management system.

1.3 Data issues

The preliminary aim of work was to demonstrate feasibility of producing smooth non-linear mappings of the ignition timing and the air-to-fuel ratio versus an engine load and speed, using neural networks.

The proper techniques have been developed on two data sets. The first one was for a Rover 4.6 litre engine. It consisted of two "look-up" tables, one for the air-to-fuel ratio and the other one for ignition timing. The tables were defined on a 16x16 grid of inputs load and speed. The range of input values was following: load $\in [0, 100]$ [%], speed $\in [500, 5000]$ [rpm]. It was sparse, under-sampled and contained many "blank" regions, not only in the boundaries but at the centre of the input space as well, which is illustrated in the Table 1. Thus the first problem was the missing data. If the statistical distribution of data is poor, one cannot expect to achieve good generalisation. The models are only as good as the data provided.

Also, one of the inputs to the models, the engine load, cannot be directly measured; it is instead calculated from the other measurable quantities such as the air flow mass.

Another problem faced was the data acquisition. The first data set was provided in two batches, one for the air-to-fuel ratio and another one for the ignition timing. They both comprised of a number of various engine characteristics such as load, speed, torque, engine power, coolant temperatures, exhaust gas emission level etc., taken for a range of values of the ignition timing and the air-to-fuel ratio.

From these characteristics, the previously-mentioned tables of optimum mappings were extracted by an experienced engineer. One extra problem was to transfer the data from provided hard-copy materials to the electronic form - some parts had to be typed in manually, other had to be scanned, recognised by an optical character recognition system, which used neural networks, and then revised and manually corrected.

It is also worth mentioning that the target mappings to realise are likely to be sub-optimum since they have been generated by a human expert in a non-principled way.

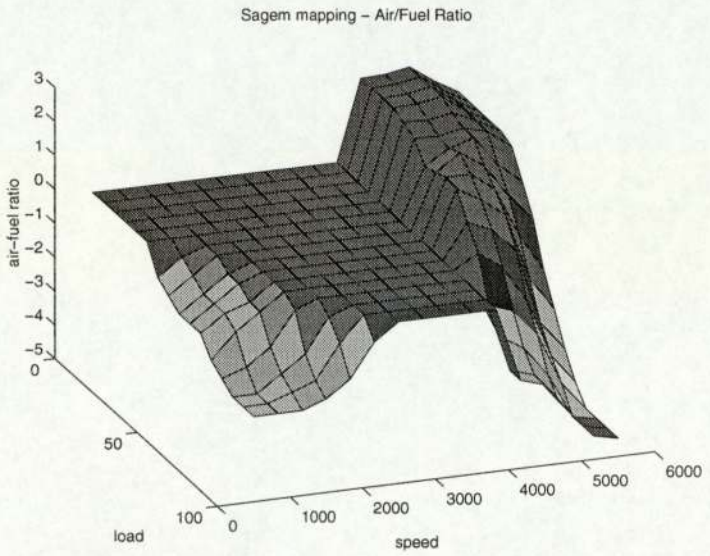


Figure 1.1: The current control surface for the air-to-fuel ratio based on a look-up table.

This task of creating optimum tables could be automated by optimising a cost function incorporating information about the emission level and the torque output, but this is not in the scope of this project.

The second set of data contained the A/F ratio and ignition timing mappings for the Rover 4.0 litre engine. The maps were completely defined on the same discrete grid of 16x16 as for the first data set, without any gaps which were pre-filled by Sagem engineers. (compare table 1 Figures 2 and 3). For these mappings, the crosses in the Table 1 were filled in with values.

The models trained on the latter data set were tested on the third data set gathered while driving a car with the Rover 4.0 litre engine. The set consisted of time-series with both steady and transient states of an engine. In the steady state the engine parameters, such as load, speed and throttle, remain near constant. The transient state is characterised by a variable throttle and therefore variable speed and/or load. There were ten parameters gathered: air temperature, engine load, speed, air-to-fuel ratio, manifold pressure, throttle position, coolant temperature, base ignition advance angle and ignition advance. The Figure 3 show the example time-series of the engine throttle while the Figure 4 present the example series of the base ignition timing.

	0	6.61	13.3	19.9	26.6	33.2	39.8	46.5	53.1	59.8	66.4	73.0	79.7	86
500	x	x	x	x	x	x	x	x	x	x	13.5	x	x	x
750	x	x	x	x	x	x	x	x	x	x	x	13.5	x	x
1000	x	x	x	14.7	x	15.5	x	15.7	x	15.0	14.5	13.5	x	x
1250	x	x	x	x	x	x	x	x	x	x	x	13.0	x	x
1500	x	x	x	14.5	14.5	x	15.7	x	15.3	x	15.0	13.0	x	x
1750	x	x	x	x	x	x	x	x	x	x	x	13.0	x	x
2000	x	x	x	15.0	x	15.6	x	16.0	x	14.8	14.5	13.5	x	x
2250	x	x	x	x	x	x	x	x	x	x	x	x	13.0	x
2500	x	x	x	15.0	15.5	x	16.0	x	15.0	x	14.5	x	13.0	x
2750	x	x	x	x	x	x	x	x	x	x	x	x	13.0	x
3250	x	x	x	15.5	x	16.0	x	15.5	x	14.7	14.0	x	x	12
3750	x	x	x	x	x	x	x	x	x	x	x	x	x	12
4000	x	x	x	15.7	16.0	x	16.0	x	13.5	x	12.2	11.8	x	11
4250	x	x	x	x	x	x	x	x	x	x	x	x	x	11
5000	x	x	x	15.0	x	13.8	x	12.3	x	11.9	x	11.2	11.0	x
5500	x	x	x	x	13.8	x	12.6	x	11.7	x	10.9	10.9	x	x

Table 1.1: The look-up table for the air-fuel ratio as defined on a 16x16 grid. Crosses indicate missing data points. The analogous table is provided for the ignition timing.

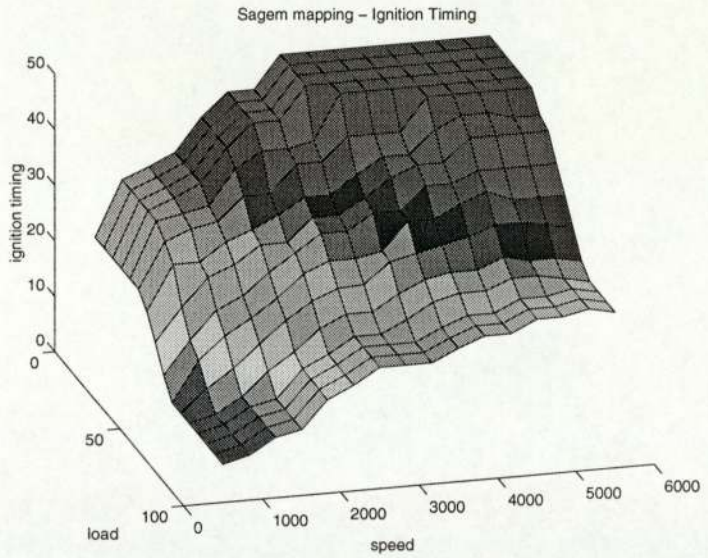


Figure 1.2: The current control surface for the ignition timing based on a look-up table.

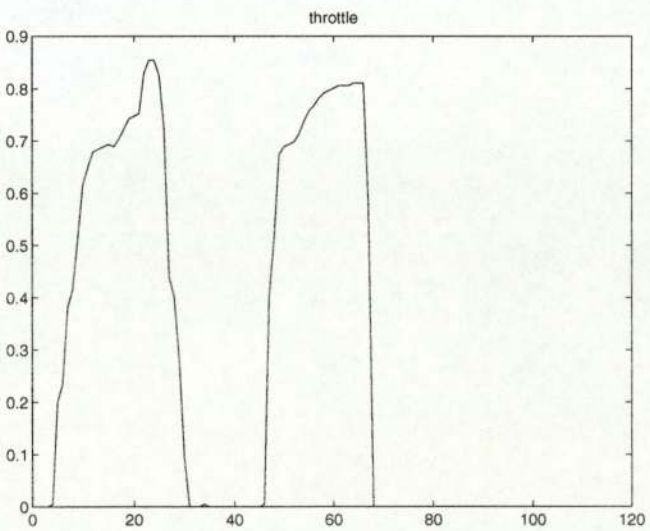


Figure 1.3: An example time-series of an engine throttle.

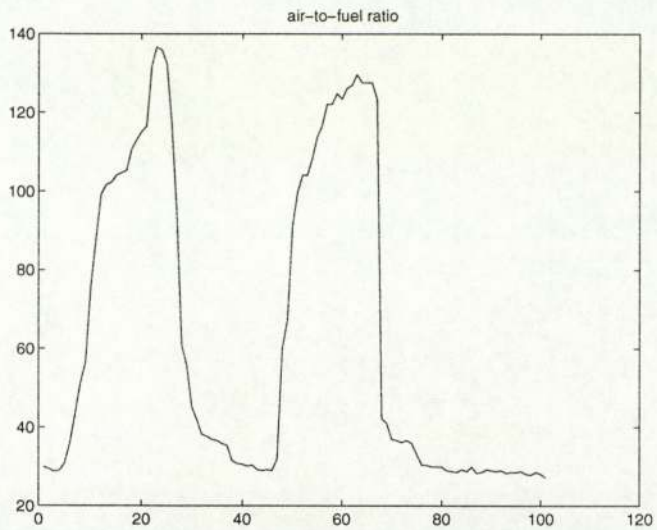


Figure 1.4: An example time-series of an engine air-to-fuel ratio.

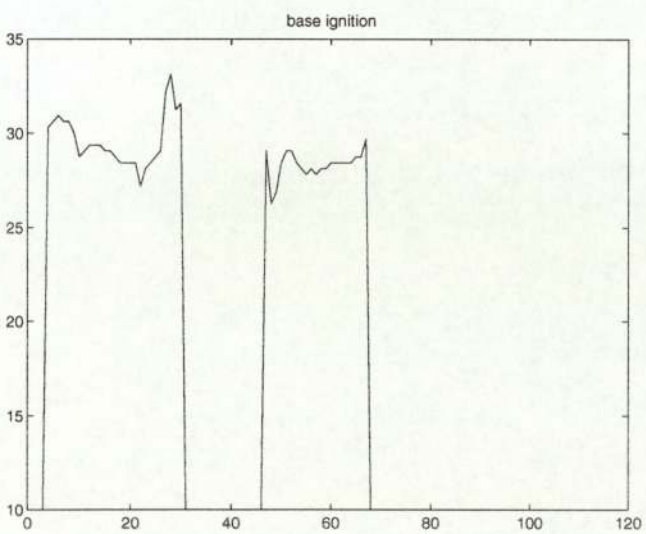


Figure 1.5: An example time-series of an engine base ignition timing.

2. Theory

This chapter briefly describes theoretical foundations of models used throughout the project. This include standard linear regression, which is used for benchmarking neural networks, the Gaussian Processes and a range of neural models such as the RBF, MLP and committees of them.

Also, a brief description of confidence intervals (error bars) and cross-validation techniques is provided.

2.1 Models

2.1.1 Linear models

A linear model with k inputs and l outputs is given in the form

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}, \tag{2.1}$$

where $\mathbf{x}$ is an input vector, $\mathbf{y}$ - output vector, $\mathbf{W}$ - a matrix of linear coefficients and $\mathbf{b}$ is a bias vector compensating for the mean in the data. The bias vector can be incorporated into the weights matrix; an additional input equal to unity has to be assumed then.

Given a training data matrix $\mathbf{X}$ with N examples and the associated target output matrix $\mathbf{T}$, where

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N, 1\}^T, \mathbf{T} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}^T, \tag{2.2}$$

the weights matrix can be calculated using a pseudo-inverse of the matrix $\mathbf{X}$:

$$\mathbf{W} = \mathbf{T}\mathbf{X}^\dagger, \tag{2.3}$$

where

$$\mathbf{X}^{\dagger} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T. \tag{2.4}$$

The data (or design) matrix $\mathbf{X}$ has dimension $N \times (k+1)$ and contains successive examples arranged in rows.

2.1.2 Non-linear models

Radial Basis Function Networks

The Radial Basis Functions Networks [1,4] perform a non-linear transformation of the $k-$dimensional input space into $l-$dimensional output space. Given an input vector $\mathbf{x}$, the kth output is given by

$$y_k = \sum_{j=1}^{M} w_{kj}\Phi_j(\mathbf{x}) + b_k. \tag{2.5}$$

Any arbitrary function $f(\mathbf{x})$ can be approximated as a linear superposition of basis functions $\Phi_j(\mathbf{x})$.

Such a system is a feed-forward neural network [1] since the information flows in the forward direction - from inputs to outputs - without internal feedbacks.

As mentioned in the section about software tools, there two common choices for the basis functions:

- Gaussians - $\Phi_j(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x}-\mu_j||^2}{2\delta_j^2}\right)$
- and cubic splines of the form $\Phi_j(\mathbf{x}) = z^2\log(z)$, where $z = ||\mathbf{x}-\mu_j||$ and μ_j is a centre of the jth basis function.

There are two paradigms to training these models. One is to use a full optimisation of the system. That is, to apply gradient-based techniques such as the steepest descent, conjugate gradients or BFGS to all centres and weights in order to minimise the error function (e.g. sum of the squares).

This is, however, a lengthy process and can be avoided by using another learning procedure.

The semi-supervised approach first sets the centres of the basis functions either to a random subset of input space or to clusters resulting from an unsupervised learning technique performed on input samples. Any clustering procedure can be applied, among many of them Kohonen network and K – means clustering together with fuzzy methods are worth mentioning.

After that the centres are fixed. In the next stage the linear coefficients (weights) are calculated using a pseudo-inverse of the design matrix $\mathbf{\Phi}$. This matrix contains activations of basis functions for each training example which are arranged in rows. Hence it has N rows and $k+1$ columns (one additional column equal to unity - for a bias value). The formal solution for the weights is given by

$$\mathbf{W} = \mathbf{\Phi}^{\dagger}\mathbf{T}, \tag{2.6}$$

where $\mathbf{T}$ is a target output matrix.

One of the crucial issues in the RBF models is choosing an optimum number of basis functions. To avoid over-fitting this number is usually much smaller than the number of training examples N. Indeed, as it could be observed in the cross-validation Figures 7 and 8, RBF networks with a large number of hidden nodes were giving large errors on input vectors deliberately excluded from the training set.

Multilayer Perceptron

The multilayer perceptron [1,3,4] is another example of a feed-forward neural network. Its architecture comprises of three main components: an input layer, a number of hidden layers with so-called hidden neurons and finally the output layer.

The input layer contains buffer neurons which pass input information forward to the non-linear hidden neurons. In the most function approximation problems only one hidden layer is considered. Each neuron is an elementary unit that performs a simple weighted summation of its inputs and transfers it through a non-linear transfer (activation) function.

Hyperbolic tangent or logistic sigmoid (equation 3) are common choices for the activation functions of these neurons. In the output layer there are linear neurons since we are trying to approximate unbounded functions.

Thus the whole network represents a non-linear transformation of inputs into outputs according to the equation

$$y_k = f^\star \left(\sum_{j=0}^{m} w_{kj}^\star f \left(\sum_{i=0}^{d} w_{ji} x_i \right) \right), \tag{2.7}$$

where $f(\cdot)$ is a transfer function, w is a set of hidden neurons weights and $w^\star$ is a set of weights connecting hidden nodes to output neurons. In the case of linear outputs $f^\star(x) = x$.

Training procedures for multilayer perceptrons employ gradient-based methods such as conjugate gradients or BFGS. Also, other methods like simulated annealing or genetic algorithms may be used for learning in non-linear perceptrons. Nevertheless, gradient methods are much faster than the latter techniques. The price that is paid for gains in speed is frequently getting stuck in local minima of an error function.

Committee of Networks

The concept of a committee of networks can be introduced following the Bayesian approach [1].

Any neural network model can be represented by a vector of parameters (weights) θ. Such a single solution is sub-optimal because it corresponds to one of many local minima (which is a peak in the posterior distribution of θ). Thus there is a number of different vectors θ dominating the posterior weight space. The Bayesian method provides an easy tool to integrating the solution over many competing weight sets.

By integrating over all possible model configuration the output is given by

$$\mathbf{y}(\mathbf{x}) = \int_\theta \mathbf{y}(\mathbf{x}|\theta) p(\theta|\mathbf{D}) d\theta. \tag{2.8}$$

Assuming that the posterior distribution of weights has m strong peaks around

local minima this integral can be approximated by

$$\mathbf{y}(\mathbf{x}) = \sum_m p(m|\mathbf{D})\mathbf{y}(\mathbf{x}|\theta_m), \tag{2.9}$$

or equivalently

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \mathbf{y}_i. \tag{2.10}$$

Hence, a committee of networks is formed by a linear combination of different models. The linear coefficients can either be explicitly set to values $\frac{1}{m}$ (plain average) or evaluated using Monte Carlo methods or other optimisation techniques. It seems, however, that these weights should rather be fixed to equal values since then we do not deepen the problem of over-fitting to training examples.

Also, a very interesting result can be obtained for the committee fitting error. Following [1], under strong assumptions that component models are independent, the committee error is smaller than the average error of the individual models:

$$E_{\text{COM}} = \frac{1}{m} E_{\text{AVE}}, \tag{2.11}$$

where m is the number of individual models.

It must be stressed that in practice the models are not fully independent and such a dramatic error reduction is not observed. Nevertheless, typically a certain decrease in error is achieved.

Gaussian Processes

The Gaussian Processes model, proposed by Chris Williams in [10], is another tool for non-linear modelling the data.

From the Bayesian perspective, this model is a limit of a neural network with an infinite number of hidden neurons, assuming the Gaussian process priors over the functions. Given a collection of random variables $\{Y(\mathbf{x})|\mathbf{x} \in X\}$, the Gaussian process is a stochastic process with a mean $\mu(\mathbf{x}) = E[Y(\mathbf{x})]$ and a covariance function $C(\mathbf{x}, \mathbf{x}') = E[(Y(\mathbf{x}) - \mu(\mathbf{x}))(Y(\mathbf{x}') - \mu(\mathbf{x}'))]$.

Given the training pairs $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \{t_i\}, i = 1..N$, the target distribution for an input point $\mathbf{x}$ is given by a Gaussian with the following mean and variance [10]:

$$y(\mathbf{x}) = \mathbf{k}^T(\mathbf{x})K^{-1}\mathbf{t} \tag{2.12}$$

$$\sigma_y^2(\mathbf{x}) = C(\mathbf{x}, \mathbf{x}) - \mathbf{k}^T(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x}), \tag{2.13}$$

where $\mathbf{k}(\mathbf{x}) = (C(\mathbf{x}, \mathbf{x}_1), \ldots, C(\mathbf{x}, \mathbf{x}_N))^T$, K is the covariance matrix for the training data $K_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{t} = (t_1, \ldots, t_N)^T$.

Chris Williams suggests the following choice for the covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$ although in the general case there may be other choices.

$$C(\mathbf{x}_i, \mathbf{x}_j) = v_0 \exp\{-\frac{1}{2}\sum_{l=1}^{d} w_l \left(x_i^l - x_j^l\right)^2\} + a_0 + \sum_{l=1}^{d} x_i^l x_j^l + v_1\delta(i, j), \tag{2.14}$$

where d is the input dimension.

Any particular choice of the function has to generate a non-negative definite covariance matrix for any set of inputs.

The model is very suitable for automatic input relevance determination as the weights w_l can express a relative strength of each input.

The last term in equation (17) is a noise term.

Hence the positive-scale parameters to optimise are the following:

$$\theta = \log\left(v_0, v_1, w_1, \ldots, w_d, a_0, \ldots, a_d\right). \tag{2.15}$$

By taking the logarithm it is ensured that the parameters on the right-hand side have only positive real values while θ can take any real values.

Training of the Gaussian Processes models is carried out by maximising the log likelihood of the training data, given by

$$l = -\frac{1}{2}\log\det K - \frac{1}{2}\mathbf{t}^T K^{-1}\mathbf{t} - \frac{N}{2}\log 2\pi. \tag{2.16}$$

This task can be accomplished by using either integration via Hybrid Monte Carlo or gradient-based methods (e.g. conjugate gradients). In this work the second approach

has been chosen, which is much faster than multi-dimensional integration and produces comparable results.

The derivation gradients of the likelihood function with respect to the parameters vector θ is following:

$$\frac{\partial l}{\partial \theta} = -\frac{1}{2}\text{Trace}\left(K^{-1}\frac{\partial K}{\partial \theta}\right) - \frac{1}{2}\mathbf{t}^T\left(-K^{-1}\frac{\partial K}{\partial \theta}K^{-1}\right)\mathbf{t} \tag{2.17}$$

$$\frac{\partial K_{ij}}{\partial \theta^0} = \frac{\partial K_{ij}}{\partial v_0}\exp\theta^0 = \exp\{-\frac{1}{2}\sum_{l=1}^{d} w_l\left(x_i^l - x_j^l\right)^2\} \tag{2.18}$$

$$\frac{\partial K_{ij}}{\partial \theta^1} = \frac{\partial K_{ij}}{\partial v_1}\exp\theta^1 = \delta(i,j) \tag{2.19}$$

$$\frac{\partial K_{ij}}{\partial \theta^{1+k}} = \frac{\partial K_{ij}}{\partial w_k}\exp\theta^1 = v_0\exp\{-\frac{1}{2}\sum_{l=1}^{d} w_l\left(x_i^l - x_j^l\right)^2\}\left(-\frac{1}{2}\left(x_i^k - x_j^k\right)^2\right) \tag{2.20}$$

$$\text{where } k = 1..d \tag{2.21}$$

$$\frac{\partial K_{ij}}{\partial \theta^{d+2}} = \frac{\partial K_{ij}}{\partial a_0}\exp\theta^{d+2} = 1 \tag{2.22}$$

$$\frac{\partial K_{ij}}{\partial \theta^{d+2+k}} = \frac{\partial K_{ij}}{\partial a_k}\exp\theta^{d+2+k} = x_i^k x_j^k, \tag{2.23}$$

$$\text{where } k = 1..d \tag{2.24}$$

There is one major disadvantage of a gradient-based method used to train Gaussian Processes: this model, unlike the MLP, is very prone to getting stuck in high-error local minima.

2.2 Confidence Intervals

When given a prediction from a neural network, it is always useful to know the "confidence" interval, or the error bar associated with that model answer.

There are many treatments of this problem. In this thesis we distinguish two types of error bars: Bayesian and predictive.

In the Bayesian approach [1,2] network errors arise from two sources: the intrinsic noise in the data and the posterior weights uncertainty. These two terms are independent and can be expressed by

$$\sigma_y^2(\mathbf{x}) = \sigma_w^2(\mathbf{x}) + \sigma_\mu^2(\mathbf{x}), \tag{2.25}$$

where $\sigma_w^2(\mathbf{x})$ is the variance of the output due to weights uncertainty, $\sigma_\mu^2(\mathbf{x})$ is the variance of the data noise and $\sigma_y^2(\mathbf{x})$ is the overall output variance.

The first term in the equation 28 can be evaluated using the Gaussian approximation to the posterior weights distribution as follows

$$\sigma_w^2(\mathbf{x}) = \mathbf{g}^T(\mathbf{x})\mathbf{H}^{-1}\mathbf{g}(\mathbf{x}), \tag{2.26}$$

where $\mathbf{g}(\mathbf{x}) = \partial \mathbf{y}(\mathbf{x}|\theta)/\partial\theta$ and $\mathbf{H}$ is the Hessian matrix of the model.

The noise variance is given by

$$\sigma_\mu^2(\mathbf{x}) = \frac{1}{\beta(\mathbf{x})} \tag{2.27}$$

and even though in principle depends on an input value $\mathbf{x}$, it is usually assumed to be constant and approximated by [1]

$$\sigma_\mu^2(\mathbf{x}) = \frac{1}{\beta} = \frac{2E_D}{(N-\gamma)}, \tag{2.28}$$

where N is the number of training examples, γ number of well-determined parameters in the model (e.g. weights in a neural network) and E_D is the error measured on the training set. At the moment γ is approximated by a number of weights k in the models. According to the full Bayesian treatment of the error bars, it should be set to

$$\gamma = k - \alpha \mathrm{Trace}(\mathbf{H}^{-1}). \tag{2.29}$$

Also, when implementing the full Bayesian approach, the training should be iterated until the values of α and β converge. This, however, significantly slows down training of the RBF networks since, at each iteration, the Hessian matrix needs to be evaluated and inverted, and the training is repeated with pseudo-inverting the design matrix each time.

In the case of Radial Basis Function networks with centres set in the first stage and weights obtained from minimising the desired error function, the Hessian matrix is given by an exact formula [2]

$$\mathbf{H} = \beta \mathbf{\Phi}^T \mathbf{\Phi} + \alpha \mathbf{I}, \tag{2.30}$$

where

$$\alpha = \frac{\gamma}{2E_W}, \tag{2.31}$$

$$E_W = \frac{1}{2}\mathbf{w}^T\mathbf{w}, \tag{2.32}$$

$\mathbf{w}$ is the weight vector.

The network outputs can be interpreted as $\mathbf{y}(\mathbf{x}) \pm 2\sigma_y$. However, Bayesian error bars provide us only with information concerning the input data density. Although they also depend on the global training error, this relation is not localised. In some regions both the training error and generalisation error may be low (hence the network should be very confident), but still the error bars may be unreasonably high.

Nevertheless they can indicate that in the regions where the input data is scarce the network predictions may have a larger error. This is only a qualitative information which can be used for gathering further data points in these high-error regions.

The second type of bars is called "predictive error bars" and is closely related to the error on the targets.

In the implementation of the second approach there are two neural networks. One produces required mapping whereas the second one (the error network) approximates the residual error surface of the first model. This surface is extracted from the first network by measuring the residual error on the training set (target values are known). Thus the second network predicts the noise variance $(y(\mathbf{x}^n) - t^n)^2$ of the main neural model.

When compared with Bayesian error bars, the level of the predictive errors is much lower while the Bayesian error bars tend to be very large on the data provided.

2.3 Cross-validation - leave-one out method

During the design of neural network models one question always arises: how to choose an appropriate model order complexity. The goal is to find an optimum network

that would perform with the smallest possible error on unseen data. Thus a natural approach is to have two independent sets of data - one for training and another one for validating the models. The networks performances are compared with respect to the validation set. This process is called the "hold out" method.

Unfortunately, the above method may often lead to over-fitting to the validation set. Hence the third - test set should used to confirm that the best possible network has been chosen.

When applying neural networks to many real-world problems, a designer often faces a problem of insufficient quantity of data. If the data set is split into three parts - training, validating and testing - there is risk of losing a vital information during training.

The solution to this problem is provided by the cross-validation procedure [1]. The initial data set is randomly divided into S segments. The networks are trained using data from $S-1$ segments and validated on the remaining segment. The procedure is carried out S times, each time with a different validation set. The network errors are averaged over all validation sets. In this way a large proportion of data if preserved for training and the method becomes more reliable. On the other hand, the overall validating process is much longer than with a single confirmation set.

In the special case, if we have N data points and take $S = N$ then the method is known as the "leave-one out" [1].

3. Experiments

In this chapter, results of experiments made on the data sets previously described are presented. Design and development issues are discussed, together with a comparison of models and test results.

3.1 Data Preprocessing

Input data have been scaled to lie in the interval $[0, 1]$. The *load* input was therefore divided by 99.6 (a maximum possible value of load in the mappings) and the *speed* input was scaled by 5500 (a maximum speed). As to the target data, it was pre-whitened so that the output data had zero mean and unit variance.

Such a preprocessing helps neural networks during the training procedure. If inputs represent different physical quantities and are of different order of magnitudes then it is very difficult for the training algorithm to find an appropriate set of weights. The same reasoning follows for rescaling the outputs. Indeed, in the initial experiments with neural networks, neural models could not learn the data if there was no such preprocessing. Also, in case of linear models and radial basis function networks, conditioning of design matrices (described in the *Theory* section) was greatly improved.

When producing final predictions, the network outputs had to be processed back to their original range by multiplying by the previous standard deviations and adding the means.

The final number of training examples was 56 for both mappings from the first data set and 256 for mappings from the second set. Thus all the data provided was used for training.

3.2 Prior Assumptions

There has been a number of assumptions made that affect both the design and development of neural networks used in the project.

First, the prior knowledge that the mappings should be smooth suggests restricting the neural models to small yet without compromising on the accuracy. Small changes in the inputs should result in small changes in the outputs.

There is a number of different ways to restrict the complexity of neural networks and to impose smoothness on mappings produced by them.

First, a number of hidden neurons can be deliberately limited. Special care has to be taken, however, to ensure that smaller networks do not have larger training and test errors.

During the development of regression models, the usual *sum-of-squares* error function is used:

$$E = \frac{1}{2}\sum_{i=1}^{N}, \tag{3.1}$$

where N is a number of examples.

There exists another method that tries to restrict the model complexity; it introduces the following regularisation term [1] to this error function:

$$E = \frac{1}{2}\sum_{i=1}^{N}\left(y(\mathbf{x}^i) - t^i\right)^2 + \alpha\frac{1}{2}\sum_{i=1}^{K} w_i^2, \tag{3.2}$$

where α is a regularisation coefficient and K - a number of weights. This additional term pushes all the weights in the networks towards small values. This helps to prevent models from over-fitting while maintaining high level of flexibility, since a model does not necessarily have to restricted to a small number of hidden neurons. However, one has to be cautious with setting unreasonably high values of α for then this may negatively influence the accuracy of neural mappings. With large α's the regularisation term will dominate over the main sum-of-squares error function to minimise. As a result, the training procedure will concentrate too much on penalising large weights and not optimising the model fits to the data.

Recently, the following modification to the regularisation term in the equation 3.1 has been introduced:

$$E = \frac{1}{2}\sum_{i=1}^{N}\left(y(\mathbf{x}^i) - t^i\right)^2 + \alpha\sum_{i=1}^{K}\frac{\frac{w_i^2}{w_0^2}}{1+\frac{w_i^2}{w_0^2}}. \tag{3.3}$$

The resulting regulariser acts as a selective weight pruner.

Large weights (compared to either a pre-fixed threshold w_0 or to other weights in a given neuron) are kept unchanged while small ones are being pruned out.

If, for a given weight w_i it happens that $w_i^2 \gg w_0^2$ then it follows that $\frac{w_i^2}{w_0^2} \gg 1$ and

$$\frac{\frac{w_i^2}{w_0^2}}{1+\frac{w_i^2}{w_0^2}} \approx 1, \tag{3.4}$$

which results in no change being made to that weight. Otherwise, the weight is decreased accordingly.

There are also other, more complicated techniques available, such as weight sharing, the Optimal Brain Damage and the Optimal Brain Surgeon [4], which are best used when simple weight regularisation does not produce satisfactory results.

In this project, a regularisation term may be used to impose smoothness on the solutions, if it proves necessary. This may possibly prove inevitable when moving to high-dimensional mappings.

Next, the collected data sets are assumed to be stationary. The conditions under which the data was collected ideally should be stable. This holds only approximately since during the data collection on the test bed it is difficult to keep all the parameters constant (e.g. load was varying to a large extent). The speed is not so problematic on the test bed.

However, these issues have a great influence on the test data recorded while an experimental car was being driven. The driver had problems with maintaining the speed and, in particular, load constant. Thus there are some fluctuations present in the test data.

After identifying steady states from this data, the mean values over a steady period were used in the experiments. This has reduced the fluctuations to a large extent.

It has also be stressed that the optimum mappings itself should be made adaptive to changes as an engine is aging. As briefly mentioned in the first chapter, this requires having proper means to track changes in engine characteristics and being able to follow these changes while modifying the optimum control surfaces.

In the following preliminary experiments the whole input space is treated as equally important for the solution. In practice, however, there are some regions in the input space which are almost never visited during normal operational conditions. Therefore, in the next stage, a distribution of inputs should be estimated (e.g. using a mixture of Gaussians trained by the EM algorithm [1]) and used to give a proper weight for each training sample in the error function.

By using such a weighted error function, the most of the "efforts" during training will be spent on the most significant input regions. For instance, the effective emission level should be mostly reduced in the regions within which the operational point is located for most of the time.

3.3 Design Issues

The model based approach is considered in the project. The current system in operation also uses models since it is very difficult to design proper algorithms to deal with highly non-linear engine characteristics.

Artificial neural networks have been proved to be universal approximators, moreover, they are very flexible and easy to design. Also, to some extent, they can deal with uncertain and missing non-linear data. They are very well-suited to model highly dimensional and non-linear mappings [1]. Their use to an engine calibration mappings is therefore well-justified.

When approaching real-world problems it is always advisable to apply linear regression models to check whether they are suitable for a specific task before moving to more sophisticated methods, such as neural networks. This provides a benchmark for assessing minimum expectation of a neural system performance. It can be expected

that neural systems will perform at least as well as linear regression models.

In the following preliminary experiments, two inputs have been used: engine load [%] and speed [rpm]. The air-to-fuel ratio and ignition timing have been used as outputs.

A variety of models is considered in the project. At first standard linear regression, which is followed by Radial Basis Function networks and Multilayer Perceptron. Next, a more complex model is used: a committee of RBF and MLP networks. The Gaussian Processes model is also tested if suitable for implementing.

The aim of using committees of networks is to reduce the error bars and improve a generalisation capability of the system.

Two separate networks with linear output nodes (target values are continuous) were used to realise two separate mappings: one for the air-to-fuel ratio and another one for the ignition timing.

Because the number of points in both data sets is very small, all the data is used for training the models. There has been a test data gathered from driving an experimental car that is used as an independent test set. No proper data for testing the emission level is available at this stage.

As the mean to assess the network operational performance and to get a level of confidence in the network predictions, two kinds of error bars are used: approximated Bayesian error bars and predictive error bars (appendix .3).

3.4 Development Issues

The standard use of the sum-of-squares error function is being optimised:

$$E = \frac{1}{2}\sum_{i=1}^{N}\left(y(\mathbf{x}^i) - t^i\right)^2, \tag{3.5}$$

where N is the total number of training cases.

The error performance of all models is measured using a *Root Mean Square* error

$$E_{\mathrm{RMS}} = \sqrt{\frac{\sum_{i=1}^{N}\left(y(\mathbf{x}^i) - t^i\right)^2}{N}} \tag{3.6}$$

and is used in the comparison tables included in the sections *Comparison*. The same measure is used for comparing both training and cross-validation errors.

The conjugate gradients [1] procedure was used to optimise the parameters in the MLP and Gaussian Processes models. In the committees, the MLP components were separately trained using the conjugate gradients, too.

The minimum number of iterations used was 100 and did not exceed 200. After the first 100 iterations the training errors were reaching convergence.

The weights in the MLP networks were randomly drawn from a uniform distribution defined on the interval $[-1, 1]$.

The RBF networks were trained in two stages: first the centres were randomly chosen as a subset of the training inputs set, next the second-layer weights were determined using a pseudo-inverse of the design matrix (see appendix .2) using a singular value decomposition method [12].

Special care has been taken to ensure that the resulting models do not over-fit. Since there was a small quantity of data available, a proper testing set could not have been constructed as a subset of the training data. Instead, the leave-one out cross-validation method (see appendix .4) was used for the first data set (the one with 56 samples). The second data set (256 examples) was randomly divided into sixteen segments and each segment was used for cross-validating. Hence it was a cross-validation technique with $S = 16$. The leave-one out method is not suitable for large data sets and could not have been used for the second data set.

For each model complexity, the experiments were repeated ten times and the average values were used in further analysis and for making plots.

When choosing the optimum model complexity, one question arises naturally: the bias/variance dilemma [2]. As the number of hidden nodes increases, the models get very flexible and the bias is being reduced. On the other hand, the variance of the models is also increasing, and the over-fitting is present.

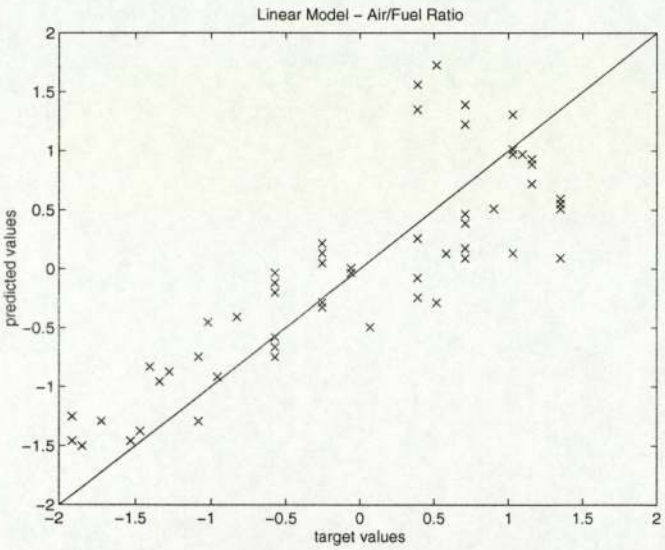


Figure 3.1: Linear model: a plot of the predicted air/fuel ratio versus the actual A/F ratio. The desired solution should be distributed close to the diagonal shown in the diagram.

The model selection within the same model type (e.g. RBF) is based on three criteria: the training set error, the cross-validation error and the decisive - a proper test set error. Among the different types of models, the choice depends on the operational speed and memory storage requirements. The smaller and faster the model is, the more likely it is to be selected.

3.5 Preliminary Results on the 1st Data

3.5.1 Linear Models

Linear regression models are described in detail in the appendix (section B.1).

Figures 3.1 and 3.2 show performance of linear models on air/fuel and ignition timing problems. Plots of target versus predicted values from the training set ideally should be close to a diagonal line beginning at the origin. As it can be seen, these models are not capable of correctly capturing the required mappings. Because of presumed non-linearity of the data, neural networks techniques have to be applied.

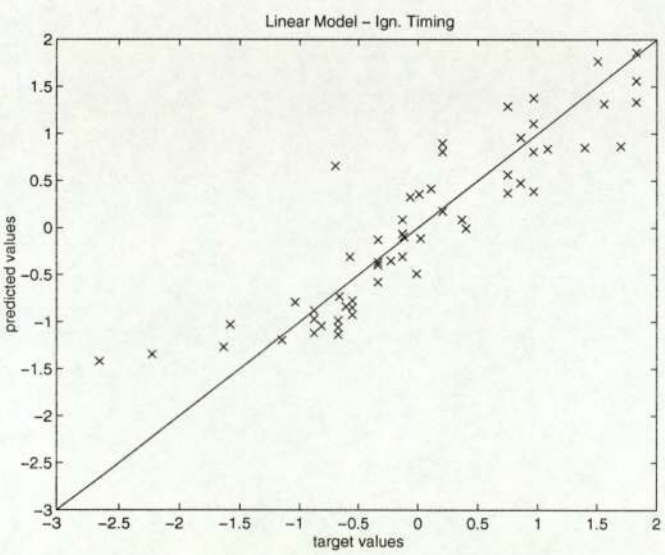


Figure 3.2: Linear model: a plot of the predicted ignition timing versus the actual values. The desired solution should be distributed close to the diagonal shown in the diagram.

3.5.2 Radial Basis Function Networks

An introduction to Radial Basis Function Networks is given in appendix B.2.1.

Spline basis functions instead of Gaussians have been used for this application since then the issue of appropriately setting the widths in Gaussians is avoided. Apart from this, spline functions have better interpolation capabilities.

First, an appropriate model complexity has to be found such that the models are neither poor nor over-fitting. Figures 5 and 6 show training errors and figures 7 and 8 present cross-validations of the RBF network with a "leave-one out" method (see appendix B.4). Plots are made for both air-to-fuel ratio and ignition timing.

Second, two radial basis function networks with 20 and 25 hidden nodes have been used for air-to-fuel ratio and ignition timing respectively.

Such a specific choice has been made based on plots of both training complexities (Figures 5 and 6) and leave-one out cross-validation (Figures 7 and 8 respectively). It can be observed that when the number of basis functions exceeds forty five, the models start to over-fit to the training set. Hence the mappings become very sharp and do not pass smoothly between training points. Such over-fitting networks cannot be used for interpolating the smooth air-to-fuel ratio and the ignition timing mappings. Also, when increasing the number of hidden nodes to after twenty for the A/F ratio and twenty five for the ignition timing, the cross-validation error stops decreasing and maintains a steady level.

Therefore, the number of hidden nodes was limited to twenty for the A/F ratio and twenty five for the ignition timing.

The target/predicted plots are presented in Figures 9 and 10. By comparing these figures with Figures 1 and 2 for linear models, it is observed that radial basis function networks produce more accurate fits than simple linear models.

It is noticed that with RBF networks the data can be more accurately fitted to produce smooth mappings, which are shown in Figures 13 and 14.

Artificial neural networks are not only capable of learning required mappings but

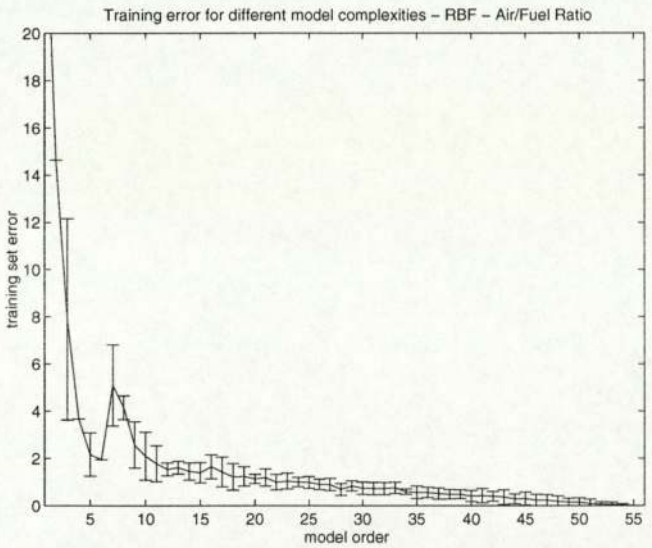


Figure 3.3: The training set error obtained as a function of increasing model order measured by the number of basis functions used. Error bars indicate maximum and minimum values obtained during ten different runs. The solid line is the mean performance.

as well can assess their performance by producing confidence intervals (appendix B.3) around the predicted values. Figures 15 and 16 show predictive error bars for both the air-to-fuel ratio and the ignition timing.

In the regions where there is not a sufficient quantity of data (e.g. boundaries), the networks tend to give rather larger error bars compared to other areas. Hence the networks are fairly confident about their predictions in the regions where there is a large number of training examples.

For comparison, in the RBF case approximate Bayesian error bars were calculated. They are presented in the Figures 17 and 18. This approach, however, gives much larger values than the observed residual error as given by predictive error bars.

There are many possible causes for this. Mainly, the Bayesian method works well only with a large quantity of data, of which there is shortage in this case.

Nevertheless, the Bayesian method is too slow to be practically implemented in the engine management system since it involves evaluating the gradient vector each time and multiplication by an inverse of a large Hessian matrix.

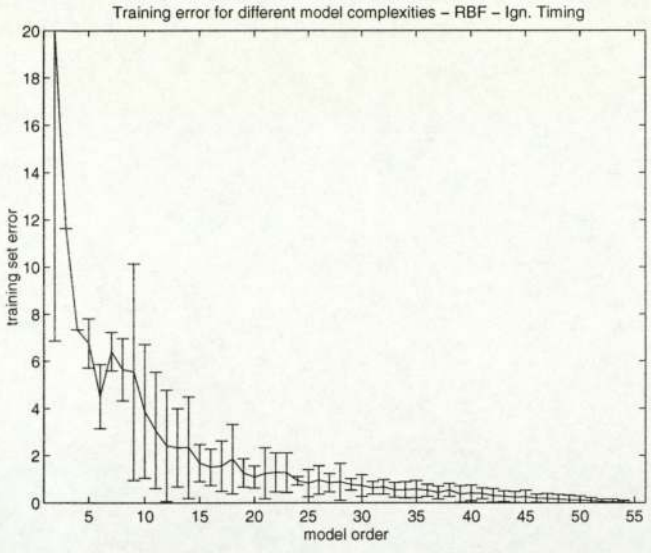


Figure 3.4: The training set error obtained as a function of increasing model order measured by the number of basis functions used. Error bars indicate maximum and minimum values obtained during ten different runs. The solid line is the mean performance.

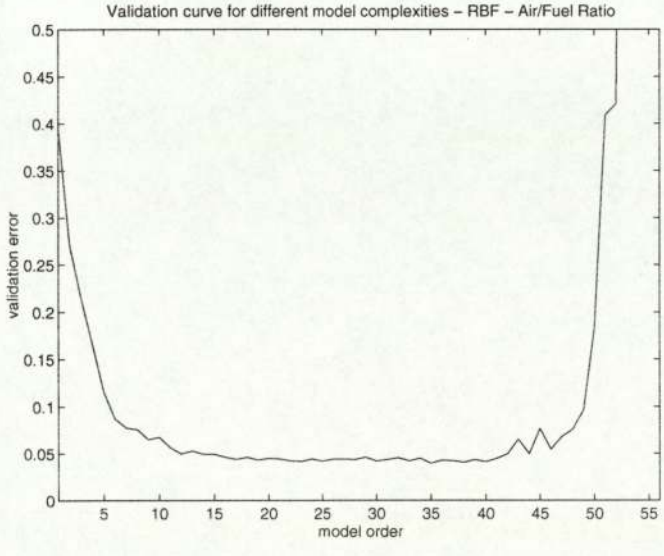


Figure 3.5: Cross-validation of the radial basis function network with a leave-one out method - air to fuel ratio.

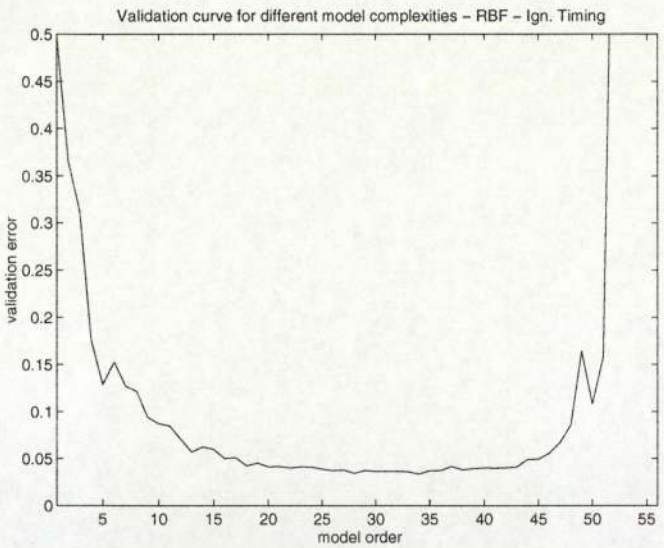


Figure 3.6: Cross-validation of the radial basis function network with a leave-one out method - ignition timing.

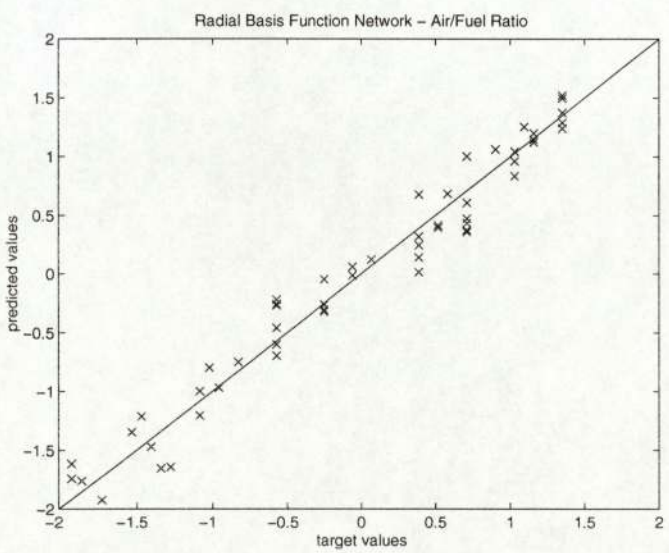


Figure 3.7: RBF: a plot of the predicted air/fuel ratio versus the actual A/F ratio. The desired solution should be distributed close to the diagonal shown in the diagram. Number of radial basis functions used: 20.

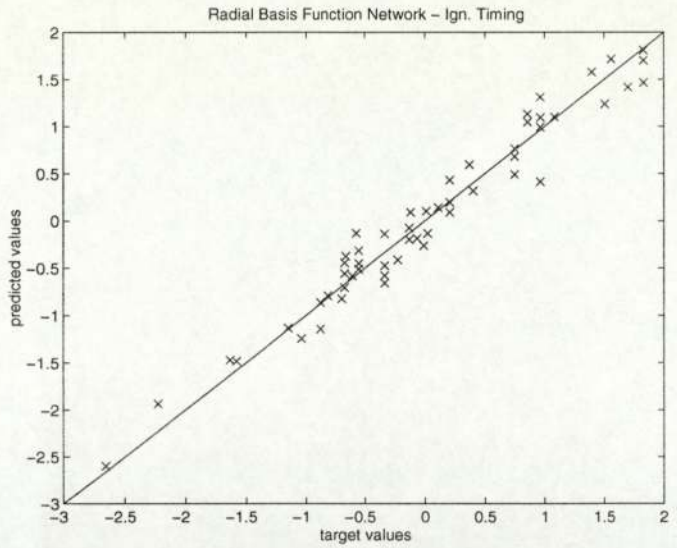


Figure 3.8: RBF: a plot of the predicted ignition timing ratio versus the actual timing. The desired solution should be distributed close to the diagonal shown in the diagram. Number of radial basis functions used: 25.

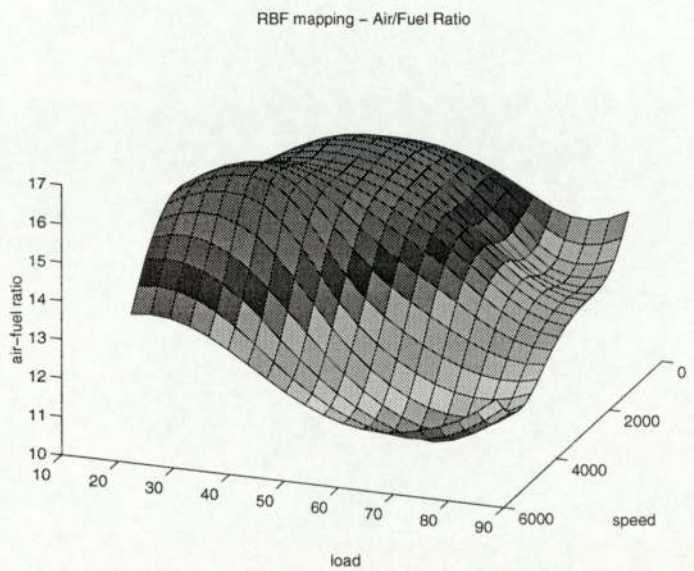


Figure 3.9: Full mapping surface for the predicted air to fuel ratio as obtained by the RBF model with 20 centers, determined as a simultaneous function of both load and speed.

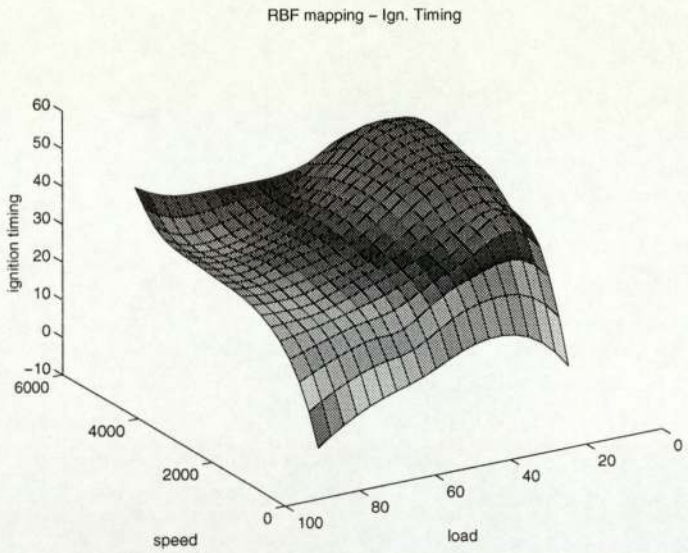


Figure 3.10: Full mapping surface for the predicted ignition timing as obtained by the RBF model with 25 centres, determined as a simultaneous function of both load and speed.

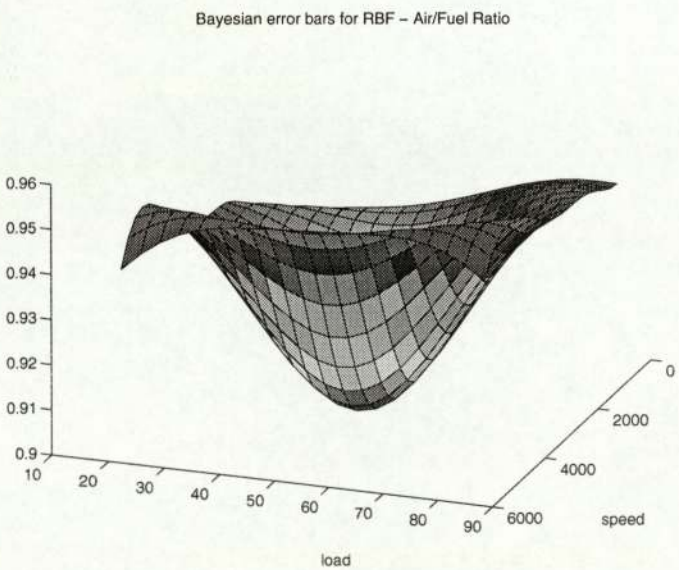


Figure 3.11: RBF Bayesian error bars with 20 hidden nodes for the air to fuel ratio.

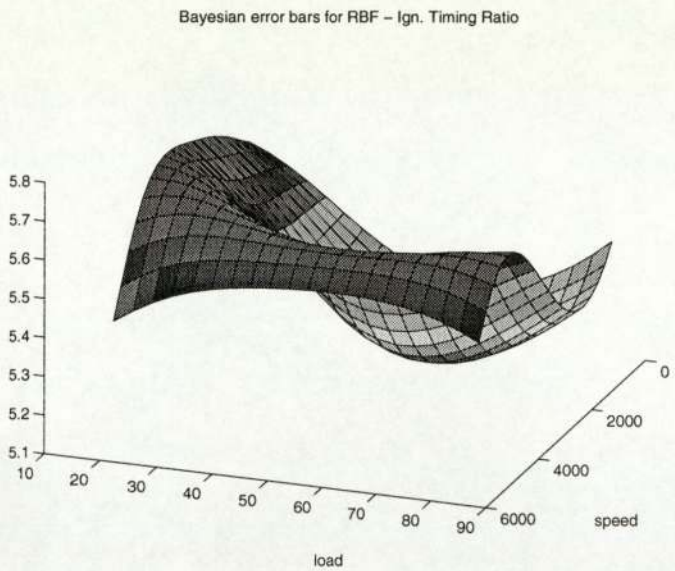


Figure 3.12: RBF Bayesian error bars with 20 hidden nodes for the air to fuel ratio.

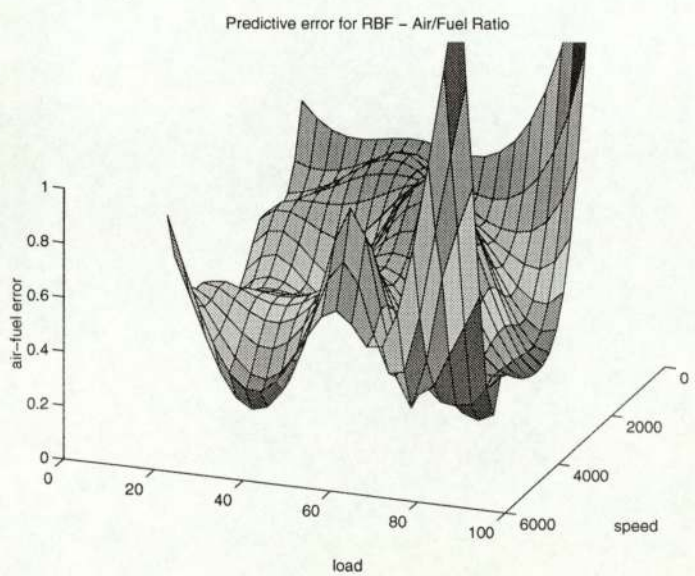


Figure 3.13: RBF predictive error surface with 20 nodes for the air to fuel ratio. Note that the error increases drastically where there is little or no data.

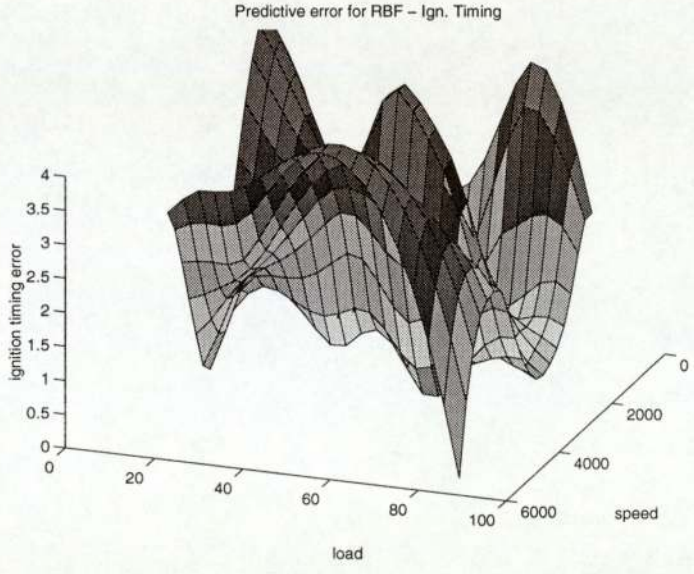


Figure 3.14: RBF predictive error surface with 25 nodes for ignition timing. Note that the error increases drastically where there is little or no data.

3.5.3 Multilayer Perceptron

In this section results obtained from multilayer perceptrons are presented. A brief introduction to this kind of network is included in the appendix (section B.2.2).

Figures 17 and 18 show target/predicted plots for multilayer perceptrons with six and four sigmoidal hidden neurons for the air-to-fuel ratio and ignition timing mappings respectively.

The number of hidden neurons has been chosen to be six for the air to fuel ratio and four for the ignition timing. Such a specific choice has been made based on plots of both training complexities (Figures 21 and 22) and leave-one out cross-validation (Figures 23 and 24 respectively). After exceeding five neurons in the hidden layer, there is little, if any, gain in the fitting accuracy for the A/F ratio and after four neurons for ignition timing. Also, flat regions (plateaus) can be observed in the cross-validation curves after that number of hidden nodes. Thus there is no reason to significantly increase the network complexity over six neurons for the air-to-fuel ratio and four for the ignition timing.

The smooth and accurate mappings produced by the MLP networks are presented in the Figures 25 and 26.

By comparing analogous plots from linear models, radial basis function networks and multilayer perceptrons, it can be concluded that the both latter models produce the most accurate fits to the training data.

However, because the original mappings provided by Sagem contain a certain level of "human" noise, the models should not nessesarily produce exact fits to all data points.

The predictive error bars can also be easily evaluated for the MLP model. Figures 27 and 28 present these error bars for two modelled surfaces. It can be observed that they are much lower than analogous error bars obtained from the RBF networks.

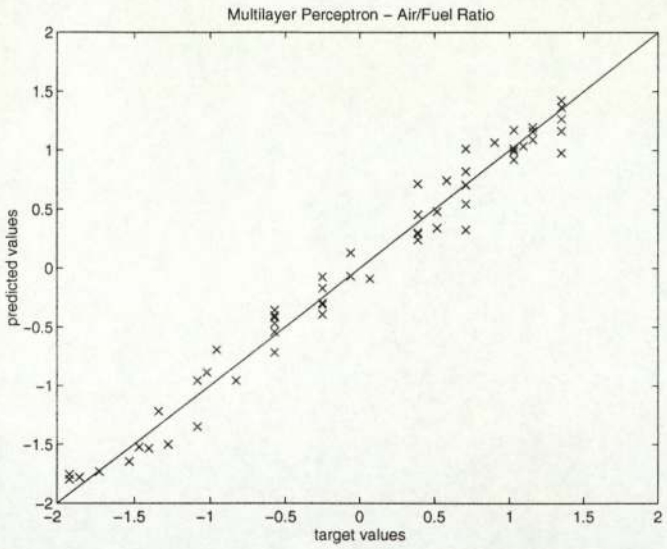


Figure 3.15: MLP: a plot of the predicted air/fuel ratio versus the actual A/F ratio. The desired solution should be distributed close to the diagonal shown in the diagram. Number of hidden neurons used: 6.

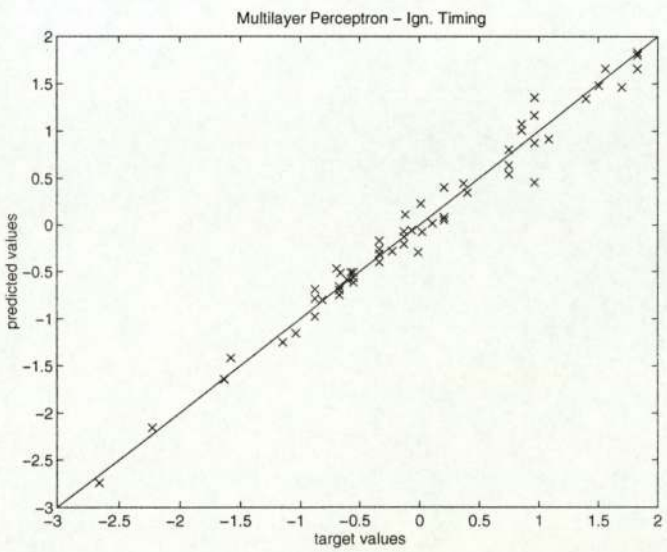


Figure 3.16: MLP: a plot of the predicted ignition timing versus the actual timing. The desired solution should be distributed close to the diagonal shown in the diagram. Number of hidden neurons used: 4.

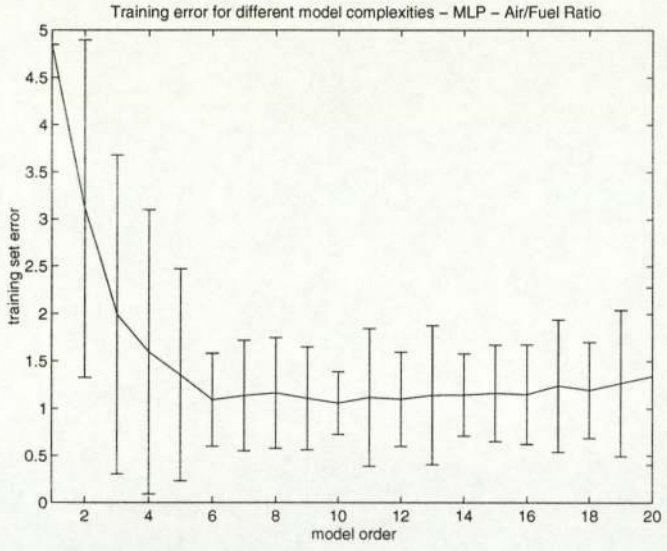


Figure 3.17: The training set error obtained as a function of increasing model order measured by the number of hidden neurons used. Error bars indicate maximum and minimum values obtained during fifty different runs. The solid line is the mean performance.

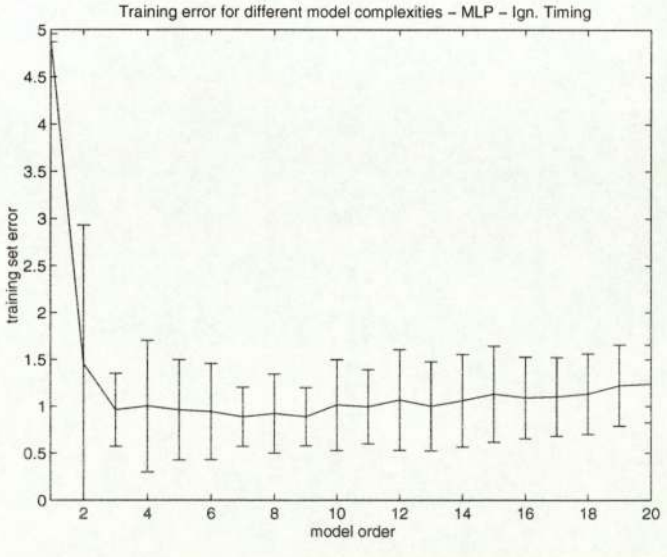


Figure 3.18: The training set error obtained as a function of increasing model order measured by the number of hidden neurons used. Error bars indicate maximum and minimum values obtained during fifty different runs. The solid line is the mean performance.

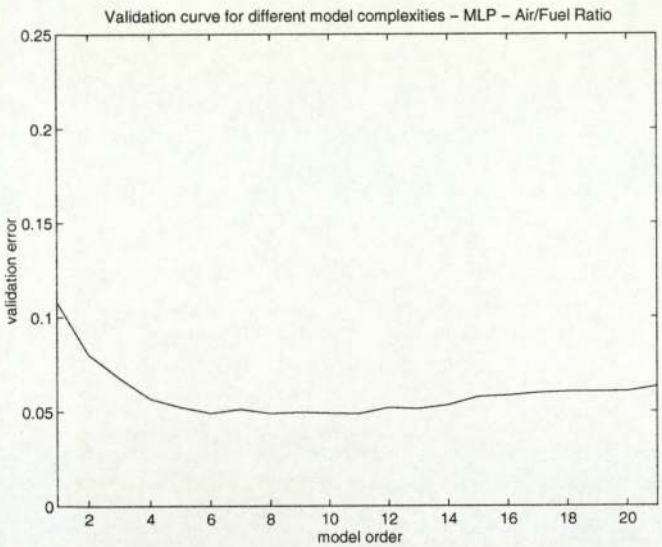


Figure 3.19: Cross-validation of the multilayer perceptron with a leave-one out method - air to fuel ratio.

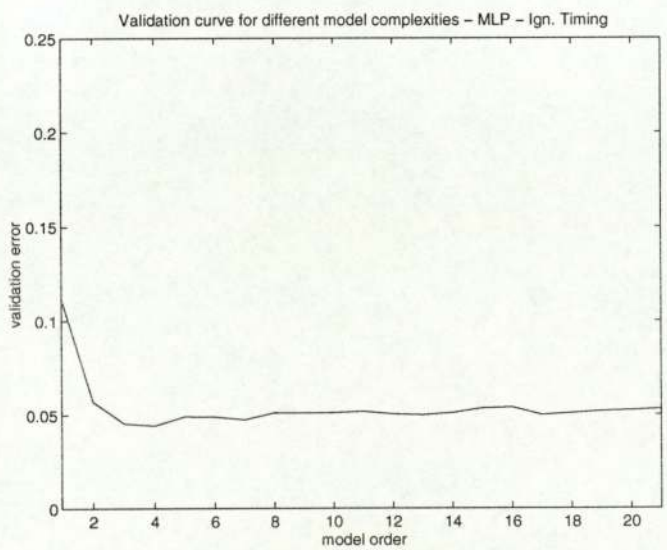


Figure 3.20: Cross-validation of the multilayer perceptron with a leave-one out method - ignition timing.

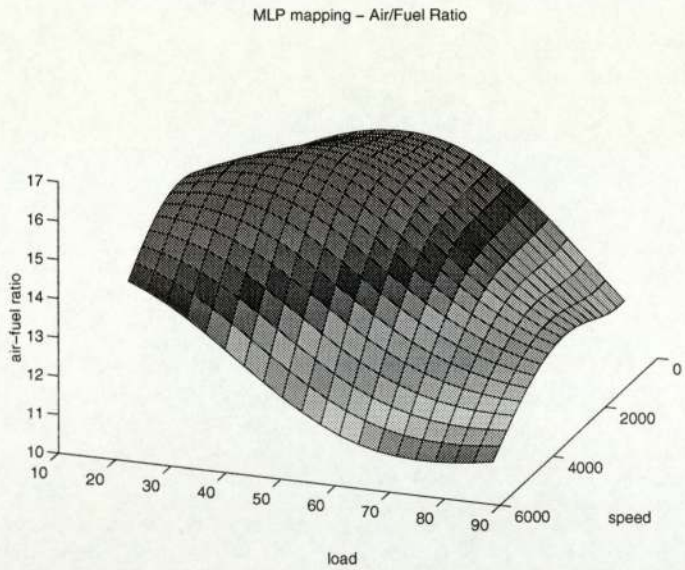


Figure 3.21: Mapping surface for the air to fuel ratio produced by the MLP with 7 hidden units determined as a simultaneos function of both load and speed.

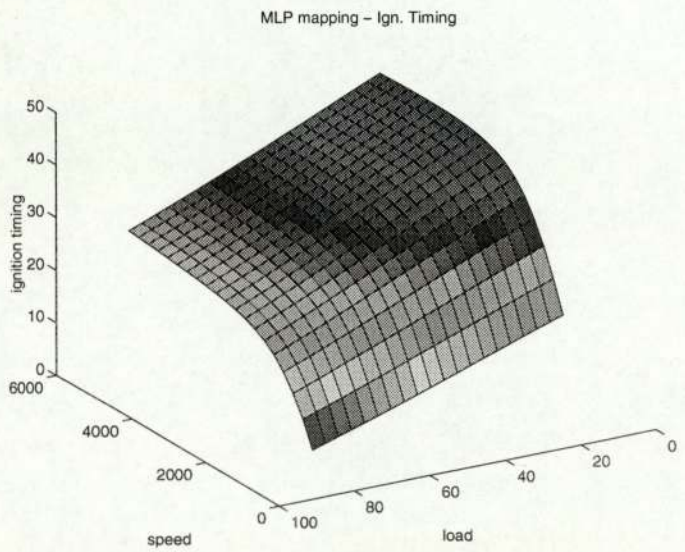


Figure 3.22: Mapping surface for the ignition timing produced by the MLP with 12 hidden units determined as a simultaneous function of both load and speed.

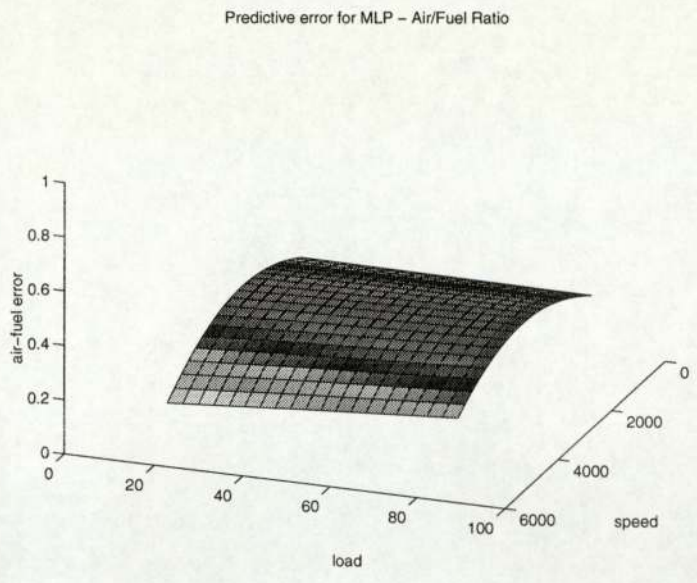


Figure 3.23: MLP predictive error surface with 7 hidden neurons for the air to fuel ratio.

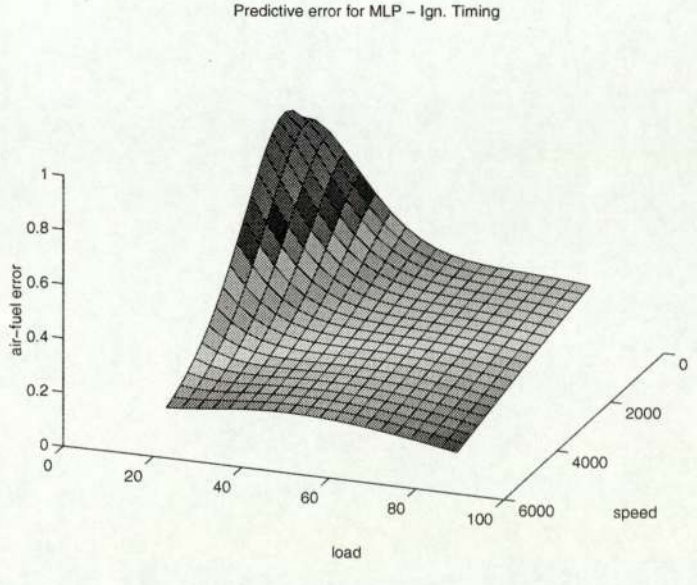


Figure 3.24: MLP predictive error surface with 7 hidden neurons for the ignition timing.

3.5.4 Committee of Networks

After applying Multilayer Perceptrons and Radial Basis Functions separately, the focus can be moved on to form a committee of networks. A detailed description of these systems can be found in the appendix (section B.2.3).

There are two committee networks. One is for the air to fuel ratio and consists of five Multilayer Perceptrons with the following number of hidden neurons: $\{3, 5, 7, 9, 11\}$ and six Radial Basis Function models with $\{15, 17, 19, 21, 24\}$ centres.

The second structure is used for modelling the ignition timing, and has the following architecture: five Multilayer Perceptrons with $\{7, 9, 11, 13, 15\}$ hidden nodes and six Radial Basis Function Networks with $\{19, 21, 22, 24, 26, 27\}$ centres.

In theory (see theoretical section 2.3) such a committee network should significantly outperform single models, but when dealing with this topic researchers often find only a slight improvement. This is due to the fact that component models are not independent. Thus in the worst case it can be expected that on average the committee will not perform worse than a single MLP or RBF.

Figures 27 and 28 show target/predicted plots for the A/F ratio and ignition timing respectively.

As it can be observed, these fits are only slightly better than the best Multilayer Perceptron. The result was achieved without adapting the weighting coefficients. Later they can be adjusted according to the generalisation capability of each component model. This should give a gain in the overall performance of the committee. Hence, when applied to a good quality data, committee networks should be able to outperform other neural models.

The predictive error bars (Figures 29 and 30) appear to have values between the lowest bound from the MLP and the highest one given by the RBF components.

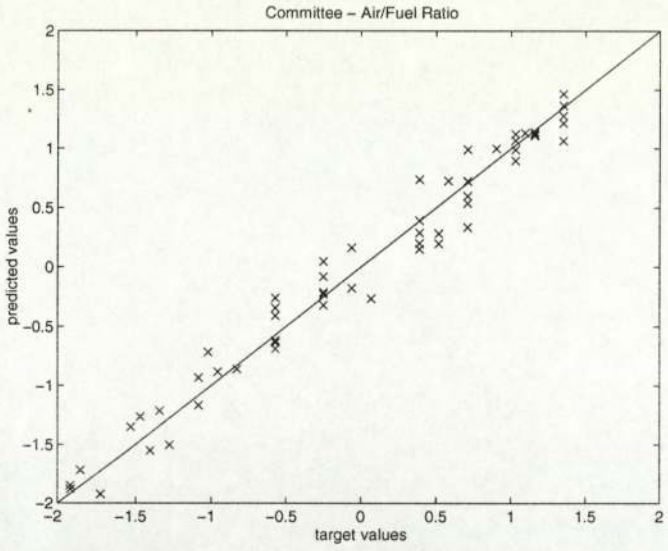


Figure 3.25: Committee: a plot of the predicted air/fuel ratio versus the actual A/F ratio. The desired solution should be distributed close to the diagonal shown in the diagram.

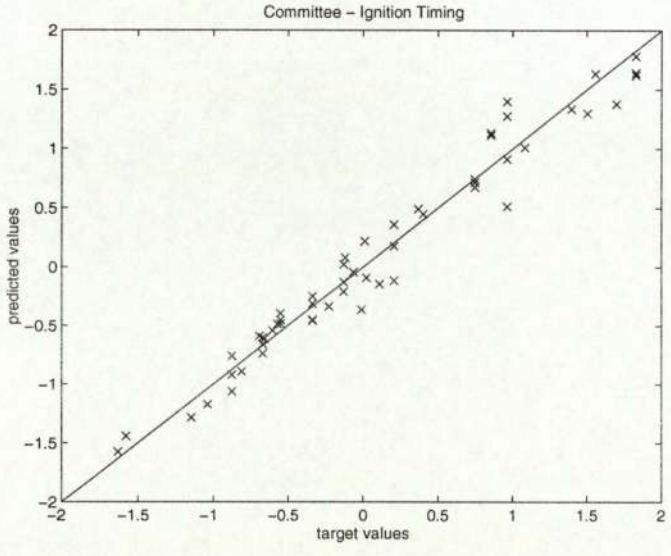


Figure 3.26: Committee: a plot of the predicted ignition timing versus the actual timing. The desired solution should be distributed close to the diagonal shown in the diagram.

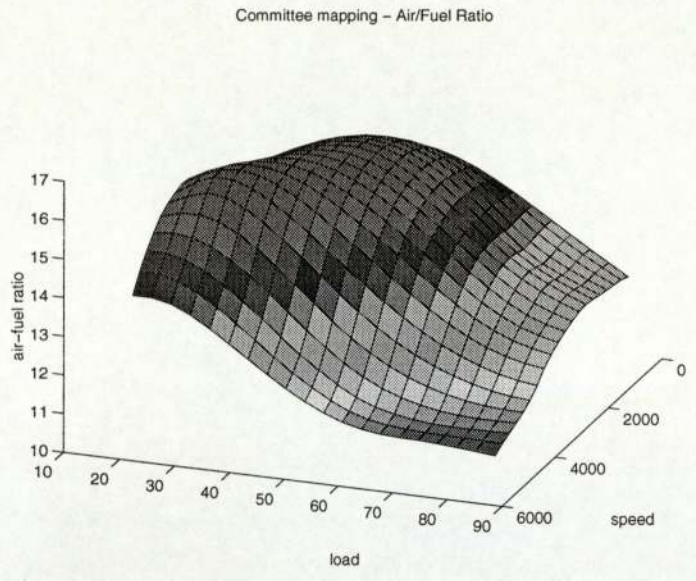


Figure 3.27: Mapping surface for the air to fuel ratio produced by the Committee determined as a simultaneos function of both load and speed.

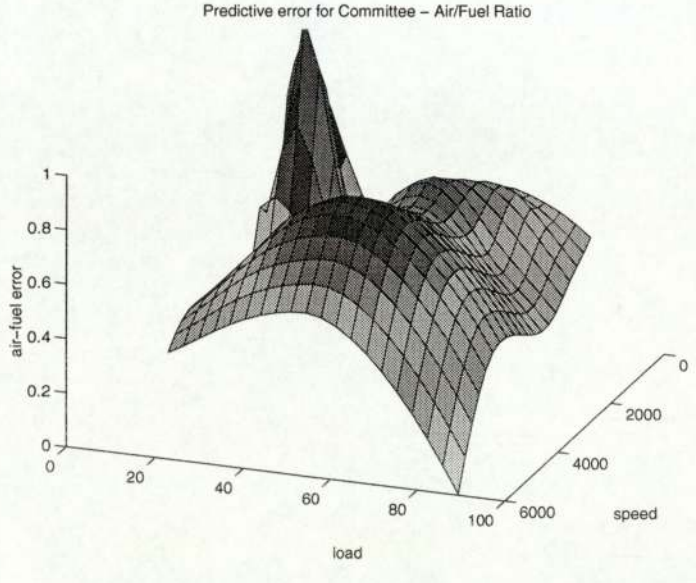


Figure 3.28: Committe predictive error surface for the air to fuel ratio.

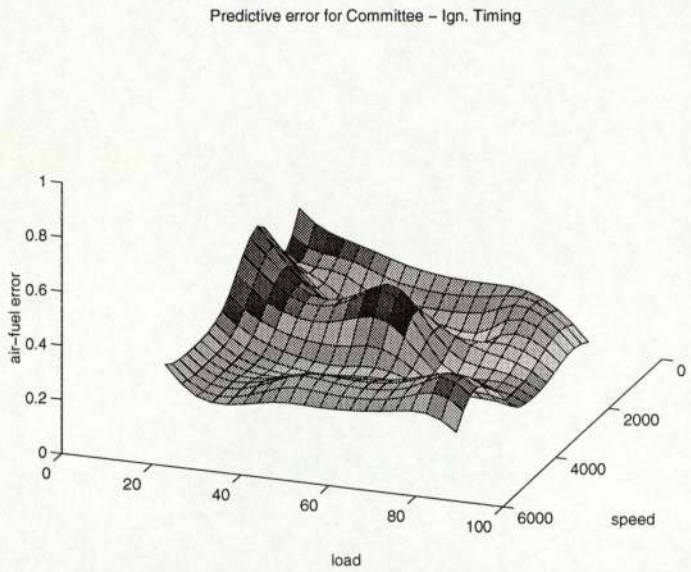


Figure 3.29: Committe predictive error surface for the ignition timing.

3.5.5 Gaussian Processes

This section presents the results obtained using the Gaussian Processes models. A brief introduction to these models is included in the section 2.4.

The comparison of Gaussian Processes and other models is given in the next section.

Figures 17 and 18 show target/predicted plots for the air-fuel ratio and ignition timing, which are not very different from the RBF and MLP models.

The mappings (Figures 25 and 26) produced by these models are very similar to the MLP ones.

The error bars produced by the Gaussian Processes models are low in the centre of the mappings and, when moving towards the edges, rapidly increase (see Figures 36 and 37). This is a "proper" behaviour since there is little data available at the boundaries and the models become less confident.

3.5.6 Comparison of Models

The table 2.1 presents the comparison of training errors for different models on the air-to-fuel ratio and the ignition timing data.

The committees of RBF and MLP networks have the lowest error on the both

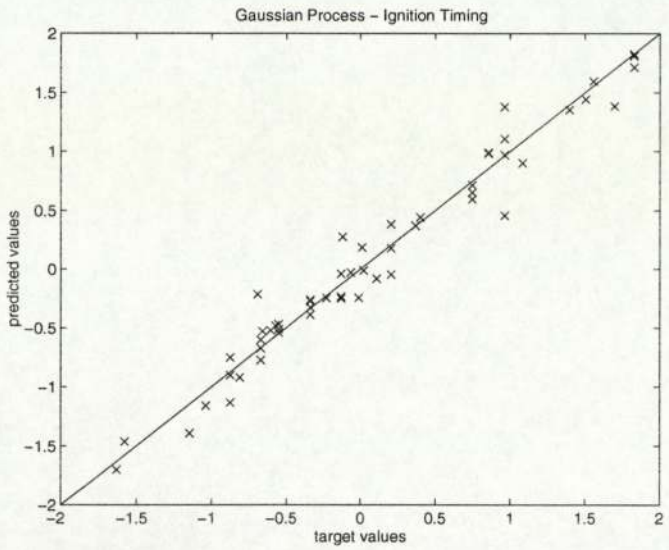


Figure 3.30: Gaussian Processes: a plot of the predicted air/fuel ratio versus the actual A/F ratio. The desired solution should be distributed close to the diagonal shown in the diagram.

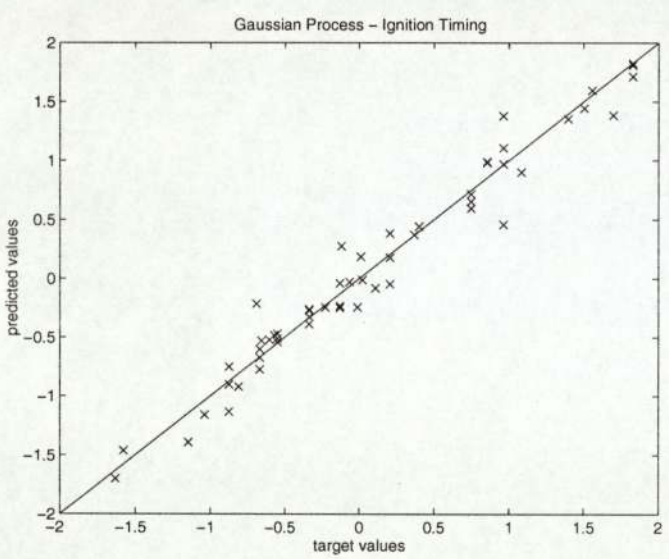


Figure 3.31: Gaussian Processes: a plot of the predicted ignition timing versus the actual timing. The desired solution should be distributed close to the diagonal shown in the diagram.

Model	RBF	MLP	Committee	Gaussian Processes	Linear
RMS error for air-to-fuel ratio	0.2018	0.1973	0.1793	0.1909	0.5352
RMS error for ign. timing	0.1722	0.1890	0.1512	0.1711	0.4268

Table 3.1: The average training error for the air-fuel ratio and ignition timing for the first data set.

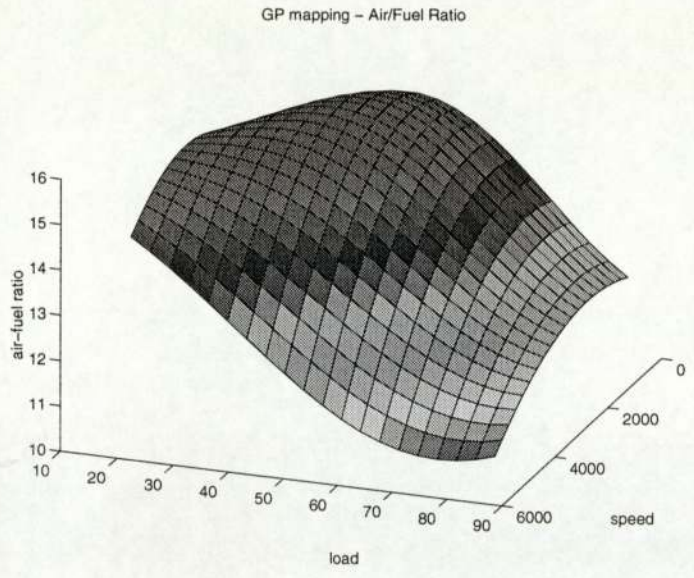


Figure 3.32: Mapping surface for the air to fuel ratio produced by the Gaussian Processes determined as a simultaneous function of both load and speed.

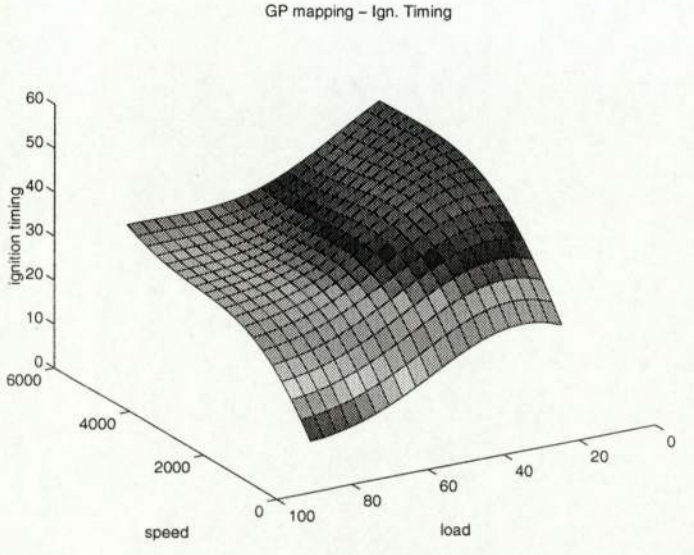


Figure 3.33: Mapping surface for the ignition timing produced by the Gaussian Processes determined as a simultaneous function of both load and speed.

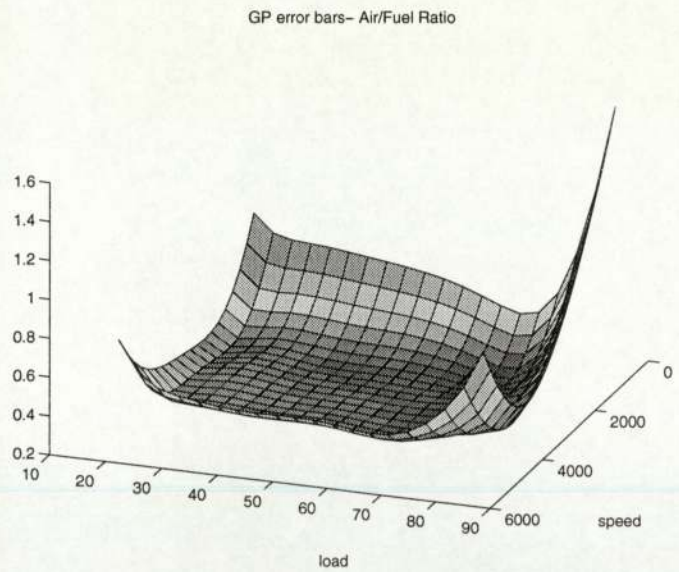


Figure 3.34: Gaussian Processes predictive error surface for the air to fuel ratio. Note that the error increases drastically where there is little or no data.

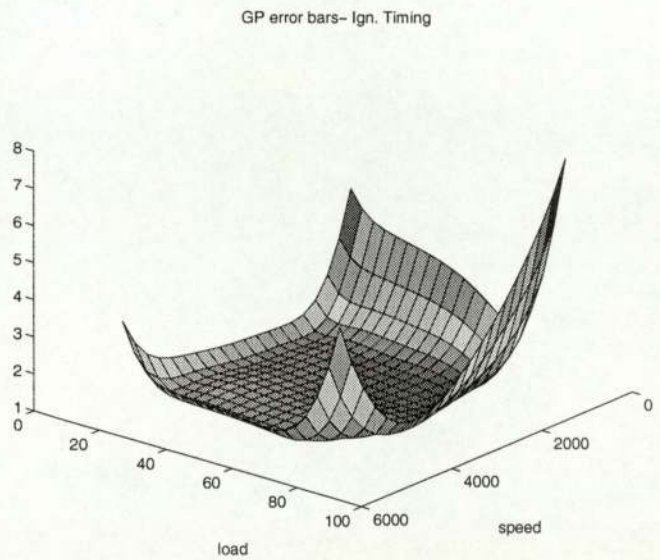


Figure 3.35: Gaussian Processes predictive error surface for the ignition timing. Note that the error increases drastically where there is little or no data.

Model	RBF	MLP	Committee
Cross-validation error - A/F ratio	0.2018	0.1909	0.1701
Cross-validation error - ign. timing	0.1813	0.2104	0.1936

Table 3.2: Leave-one out cross-validation error for the first data set - RMS error.

training sets, next is the MLP models followed by the RBF network. The Gaussian Processes model is slightly better than the RBF and MLP. At the far end is the standard linear model, which is only used for benchmarking. Its fits (a three-dimensional planes) seem to be inadequate to model non-linear surfaces, which is not surprising.

When comparing the cross-validation errors, the committee appears to exhibit the lowest error for the air-to-fuel ratio, whereas on the ignition timing data the RBF network has the lowest error.

The Gaussian Processes have not been included for the cross-validating since this procedure requires a large number of model retraining, which for this model results in getting stuck in many bad local minima.

3.6 Preliminary Results on the 2nd Data

This section presents results from the analogous procedure as in the previous section, carried out for the second set of A/F ratio and ignition timing mappings.

The investigated models include: the linear regression, RBF, MLP and the committee of networks. The Gaussian Processes models have not been included for the technical reasons: on this data the covariance matrix inversion procedure, used in training, could not converge.

The same general procedure as for the first data set is used. The complexity of a model is selected based on cross-validation and training error curves.

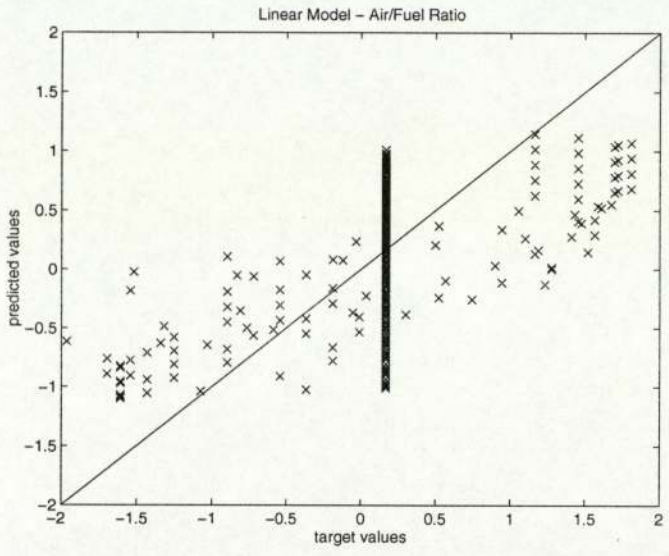


Figure 3.36: Linear model: a plot of the predicted air/fuel ratio versus the actual A/F ratio. The desired solution should be distributed close to the diagonal shown in the diagram.

3.6.1 Linear Models

Figures 3.1 and 3.2 show performance of linear models on air/fuel and ignition timing problems. Plots of target versus predicted values from the training set ideally should be close to a diagonal line beginning at the origin. Again, as for the first data set, these models are not capable of correctly capturing the required mappings.

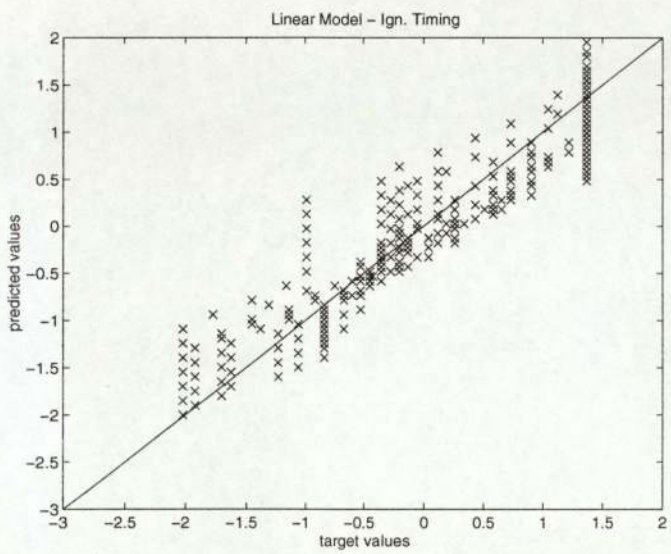


Figure 3.37: Linear model: a plot of the predicted ignition timing versus the actual values. The desired solution should be distributed close to the diagonal shown in the diagram.

3.6.2 Radial Basis Function Networks

Also for this data set spline basis functions are used.

Figures 5 and 6 show training errors and figures 7 and 8 present cross-validations of the RBF network with a "leave-one out" method (see appendix B.4).

The number of basis functions has been set to 50 for the air-to-fuel ratio and 30 for ignition timing.

Such a choice has been made based on plots of training complexities (Figures 5 and 6) and cross-validation (Figures 7 and 8 respectively).

The target/predicted plots are presented in Figures 9 and 10.

The RBF networks better fits the data than the linear model and smoothes out large differential discontinuities present in the air-to-fuel mapping (see Figures 13 and 14).

Figures 15 and 16 show predictive error bars for both the air-to-fuel ratio and the ignition timing.

In the regions where there is not a sufficient quantity of data (e.g. boundaries), the networks tend to give rather larger error bars compared to other areas. On the other hand, the networks are fairly confident about their predictions in the regions with a large number of training examples.

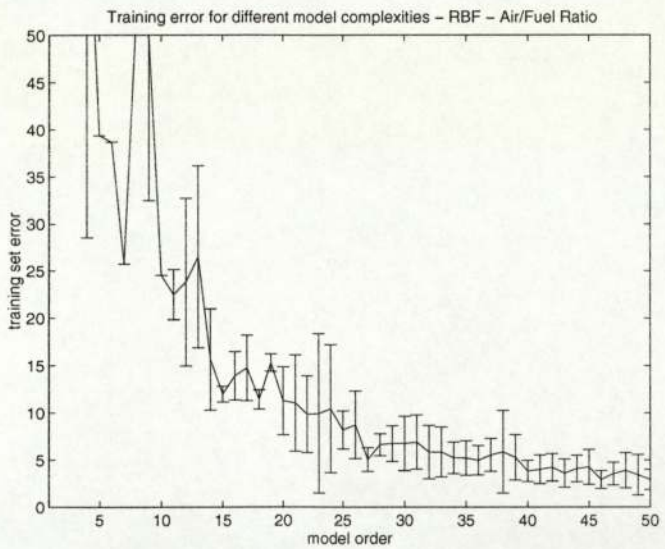


Figure 3.38: The training set error obtained as a function of increasing model order measured by the number of basis functions used. Error bars indicate maximum and minimum values obtained during ten different runs. The solid line is the mean performance.

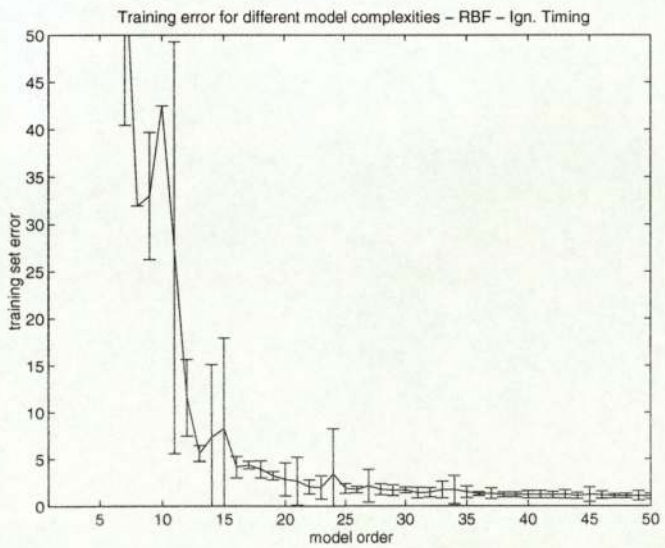


Figure 3.39: The training set error obtained as a function of increasing model order measured by the number of basis functions used. Error bars indicate maximum and minimum values obtained during ten different runs. The solid line is the mean performance.

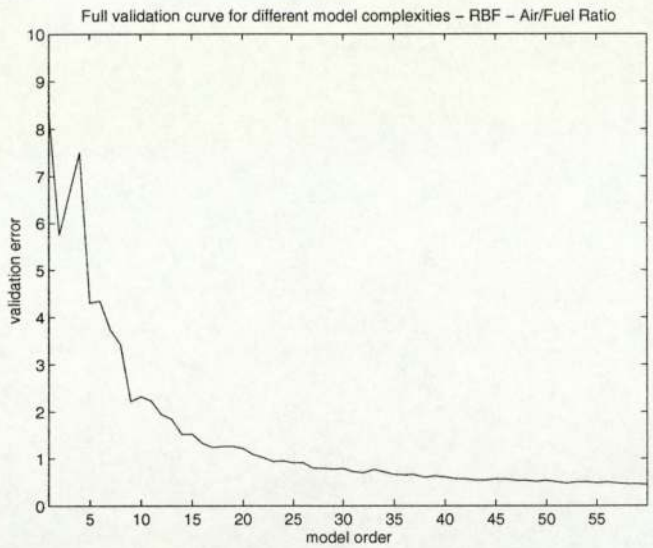


Figure 3.40: Cross-validation of the radial basis function network with 16x16 segments - air to fuel ratio.

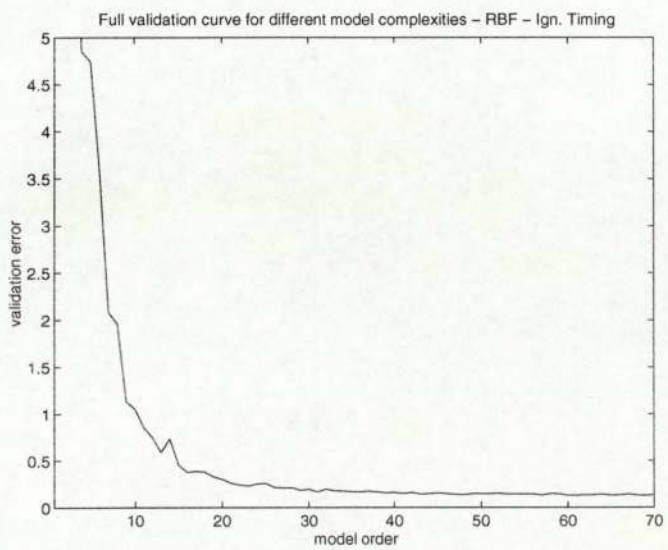


Figure 3.41: Cross-validation of the radial basis function network with 16x16 segments - ignition timing.

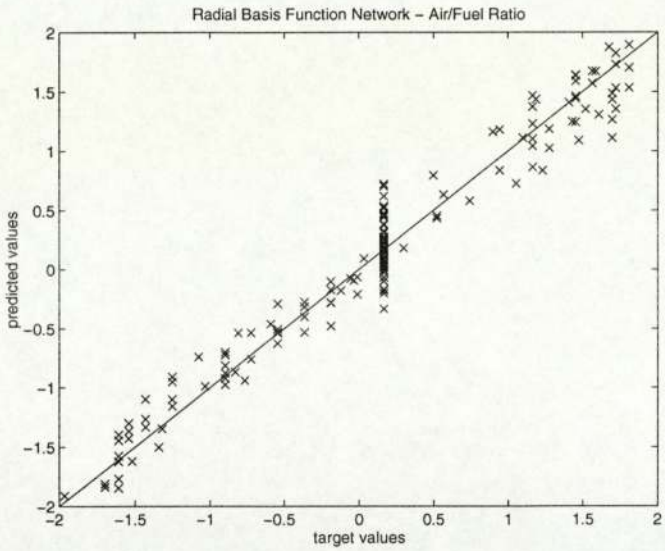


Figure 3.42: RBF: a plot of the predicted air/fuel ratio versus the actual A/F ratio. The desired solution should be distributed close to the diagonal shown in the diagram. Number of radial basis functions used: 20.

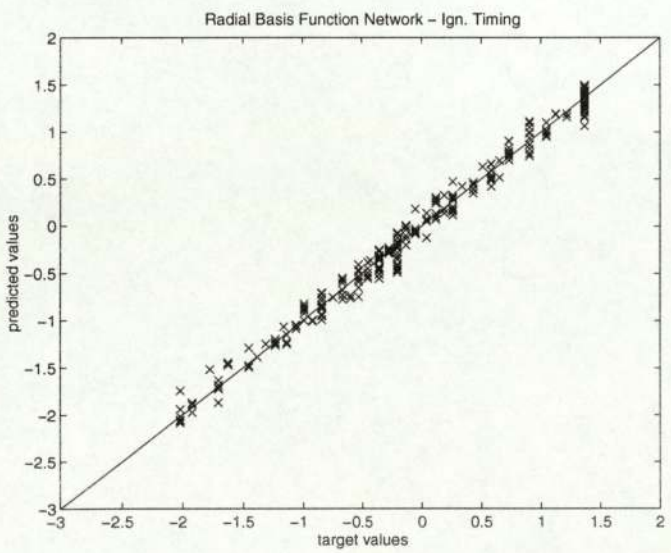


Figure 3.43: RBF: a plot of the predicted ignition timing ratio versus the actual timing. The desired solution should be distributed close to the diagonal shown in the diagram. Number of radial basis functions used: 25.

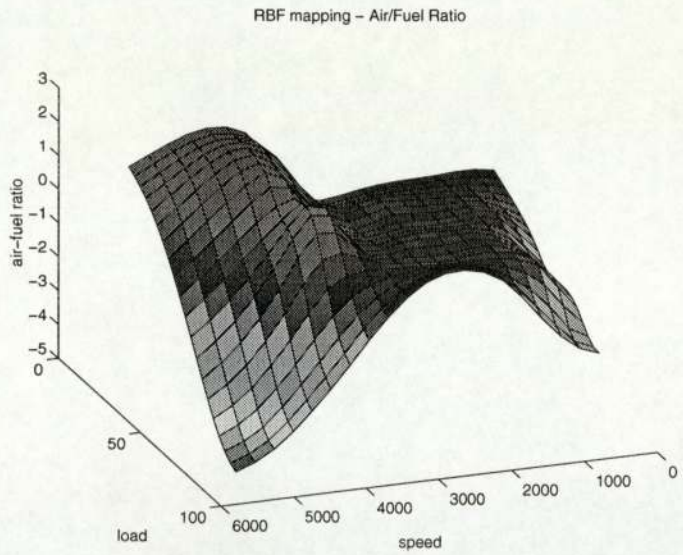


Figure 3.44: Full mapping surface for the predicted air to fuel ratio as obtained by the RBF model with 20 centres, determined as a simultaneous function of both load and speed.

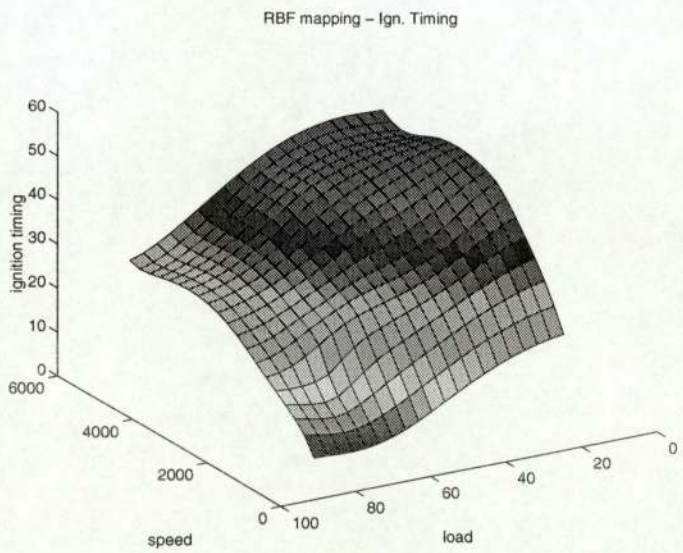


Figure 3.45: Full mapping surface for the predicted ignition timing as obtained by the RBF model with 25 centres, determined as a simultaneous function of both load and speed.

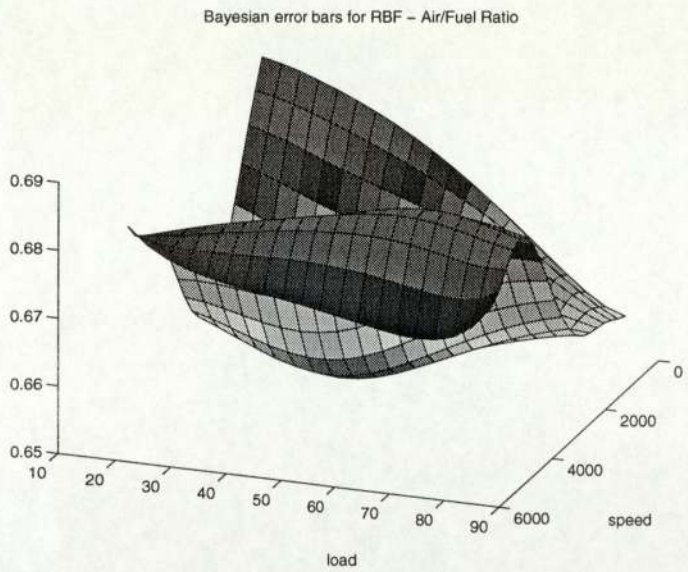


Figure 3.46: RBF Bayesian error bars with 20 hidden nodes for the air to fuel ratio.

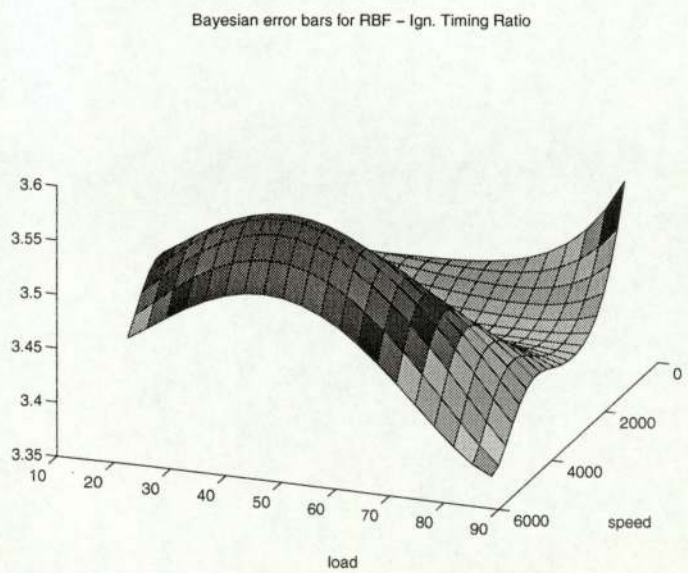


Figure 3.47: RBF Bayesian error bars with 20 hidden nodes for the air to fuel ratio.

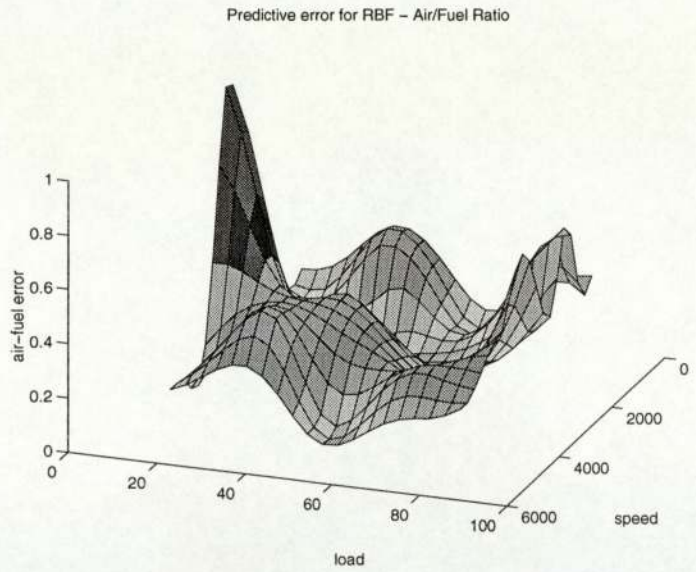


Figure 3.48: RBF predictive error surface with 20 nodes for the air to fuel ratio. Note that the error increases drastically where there is little or no data.

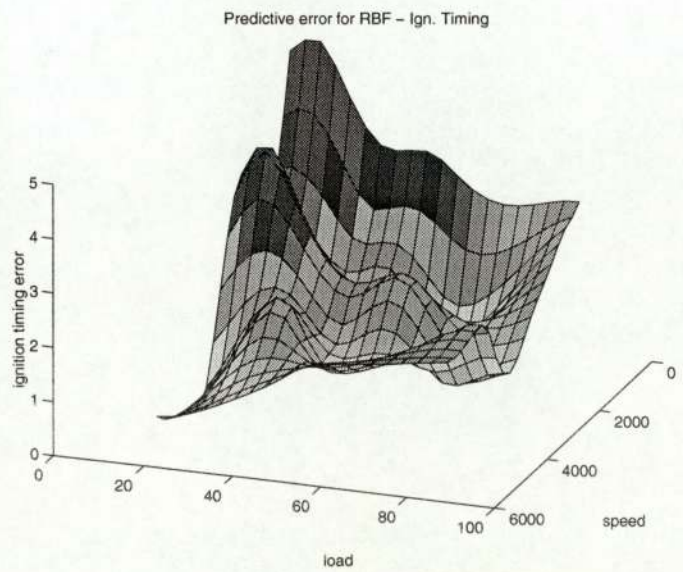


Figure 3.49: RBF predictive error surface with 25 nodes for ignition timing. Note that the error increases drastically where there is little or no data.

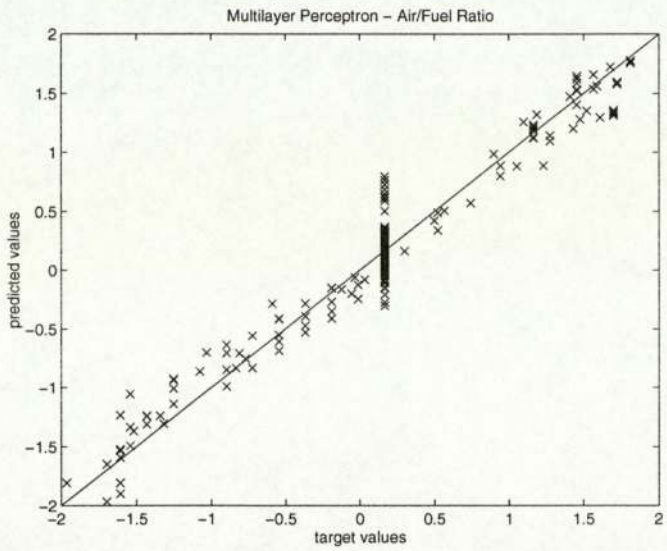


Figure 3.50: MLP: a plot of the predicted air/fuel ratio versus the actual A/F ratio. The desired solution should be distributed close to the diagonal shown in the diagram. Number of hidden neurons used: 6.

3.6.3 Multilayer Perceptron

Figures 17 and 18 show target/predicted plots for multilayer perceptrons with 10 and five sigmoidal hidden neurons respectively.

The number of hidden neurons has been chosen to be six for the air to fuel ratio and four for the ignition timing.

Such a choice has been made based on plots of both training complexities (Figures 21 and 22) and leave-one out cross-validation (Figures 23 and 24 respectively). There is no reason to significantly increase the network complexity beyond those sizes since there is no gain in performance achieved by doing so.

The smooth and accurate mappings produced by the MLP networks are presented in the Figures 25 and 26.

By comparing analogous plots from linear models, radial basis function networks and multilayer perceptrons, it can be concluded that the both latter models produce the most accurate fits to the training data.

The Figures 27 and 28 show predictive error bars that are significantly lower of analogous ones given by the RBF models.

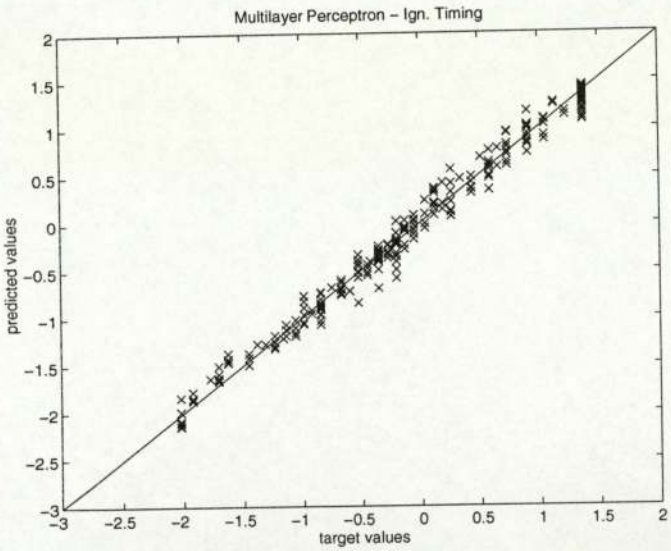


Figure 3.51: MLP: a plot of the predicted ignition timing versus the actual timing. The desired solution should be distributed close to the diagonal shown in the diagram. Number of hidden neurons used: 4.

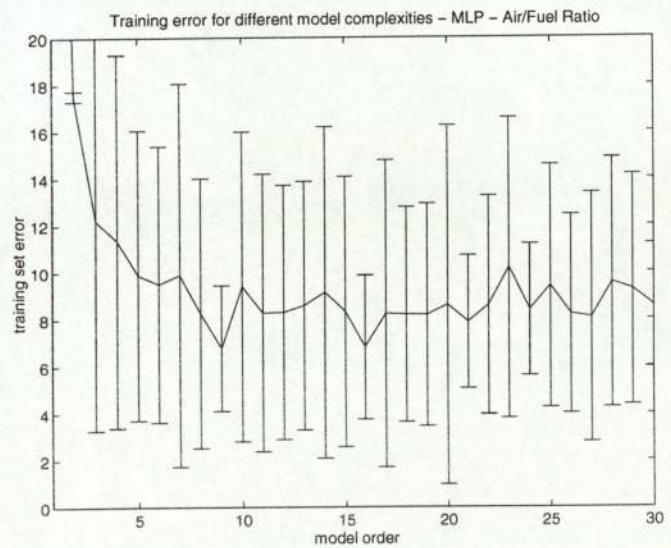


Figure 3.52: The training set error obtained as a function of increasing model order measured by the number of hidden neurons used. Error bars indicate maximum and minimum values obtained during fifty different runs. The solid line is the mean performance.

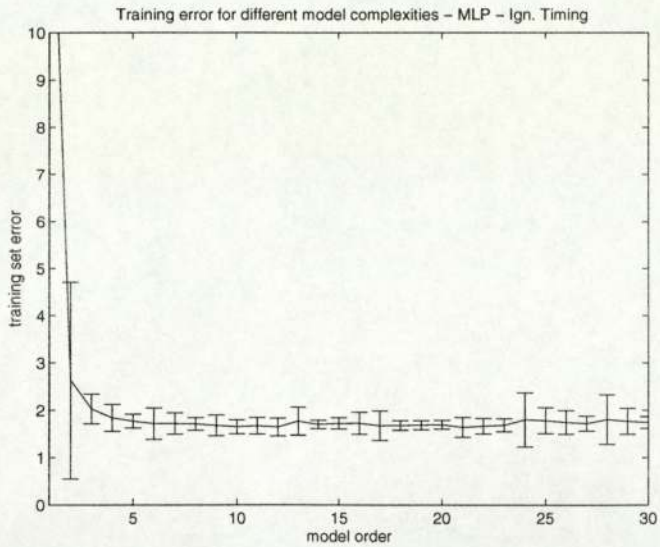


Figure 3.53: The training set error obtained as a function of increasing model order measured by the number of hidden neurons used. Error bars indicate maximum and minimum values obtained during fifty different runs. The solid line is the mean performance.

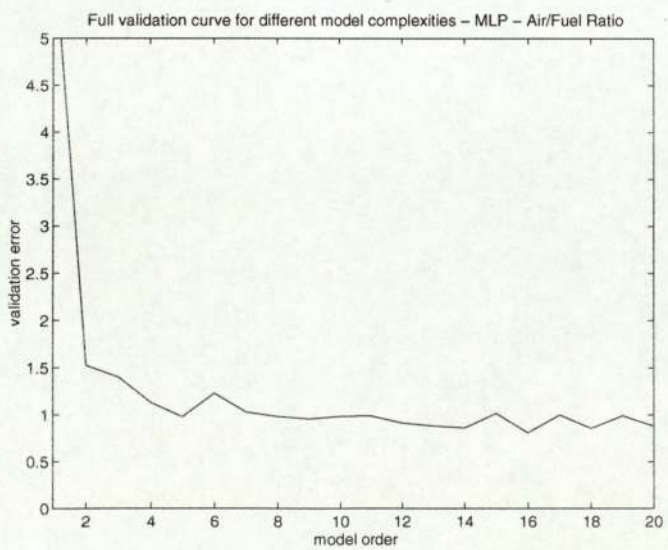


Figure 3.54: Cross-validation of the multilayer perceptron with a leave-one out method - air to fuel ratio.

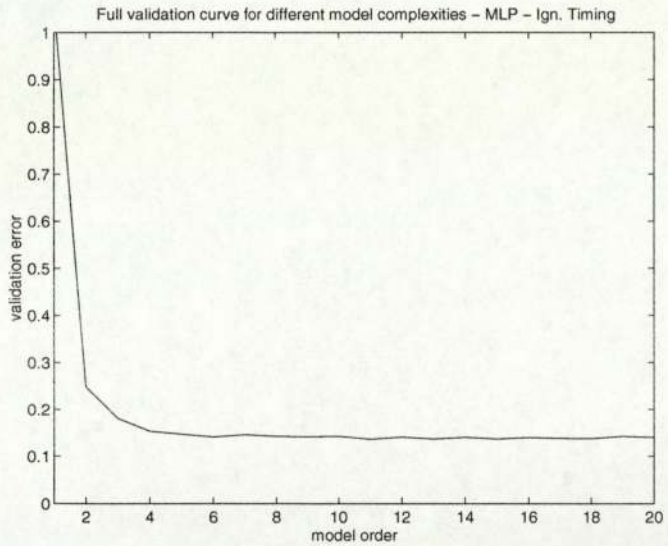


Figure 3.55: Cross-validation of the multilayer perceptron with a leave-one out method - ignition timing.

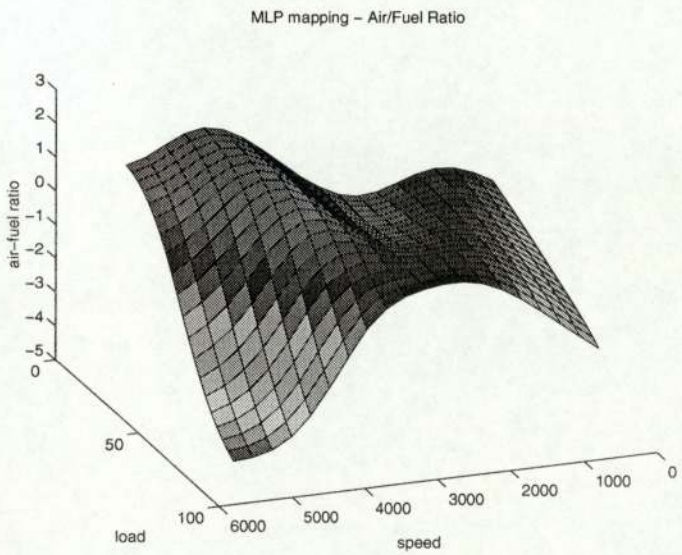


Figure 3.56: Mapping surface for the air to fuel ratio produced by the MLP with 7 hidden units determined as a simultaneos function of both load and speed.

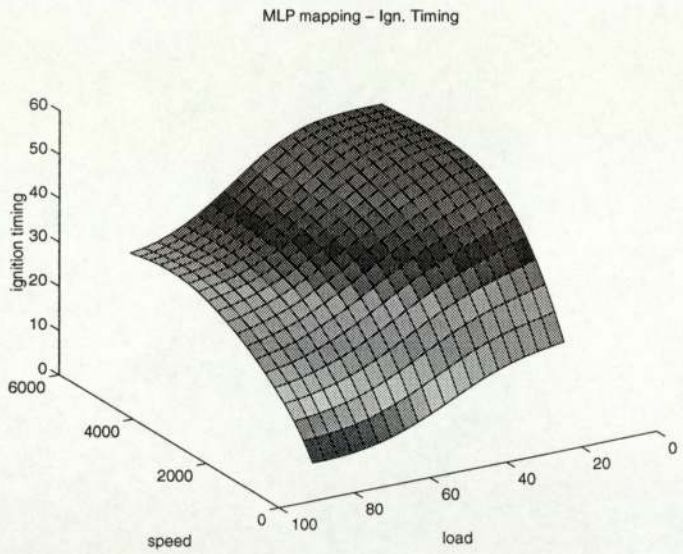


Figure 3.57: Mapping surface for the ignition timing produced by the MLP with 12 hidden units determined as a simultaneous function of both load and speed.

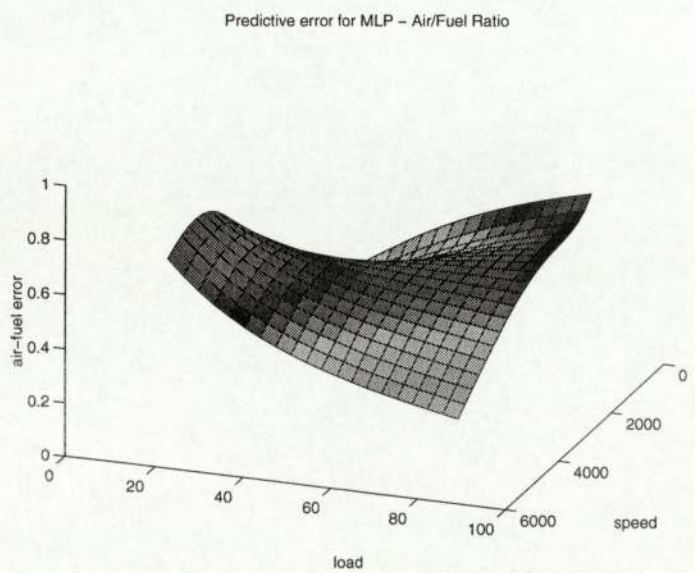


Figure 3.58: MLP predictive error surface with 7 hidden neurons for the air to fuel ratio.

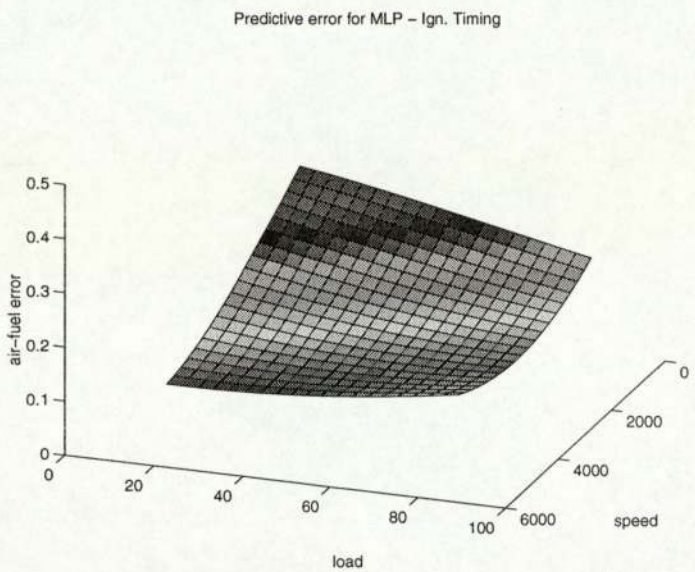


Figure 3.59: MLP predictive error surface with 7 hidden neurons for the ignition timing.

3.6.4 Committee of Networks

As previously, there are two committee networks. One is for the air to fuel ratio and consists of five Multilayer Perceptrons with a following number of hidden neurons: $\{7, 9, 11, 12, 13\}$ and five Radial Basis Function models with $\{40, 45, 50, 55, 60\}$ centres.

The second structure is used for modelling the ignition timing, and has the following architecture: five Multilayer Perceptrons with $\{3, 4, 5, 6, 7\}$ hidden nodes and five Radial Basis Function Networks with $\{20, 25, 30, 35, 40\}$ centres.

Figures 27 and 28 show target/predicted plots whereas in the Figures 29 and 30 mapping outputs versus sample index are shown.

The predictive error bars, presented in the Figures 31 and 32, are in-between the ones from MLP (a lower bound) and RBF (a higher bound).

3.6.5 Comparison of Models

The table 3.3 presents the comparison of training errors for different models on the air/fuel ratio and the ignition timing data.

The Radial Basis Function network has the lowest average error on the A/F training

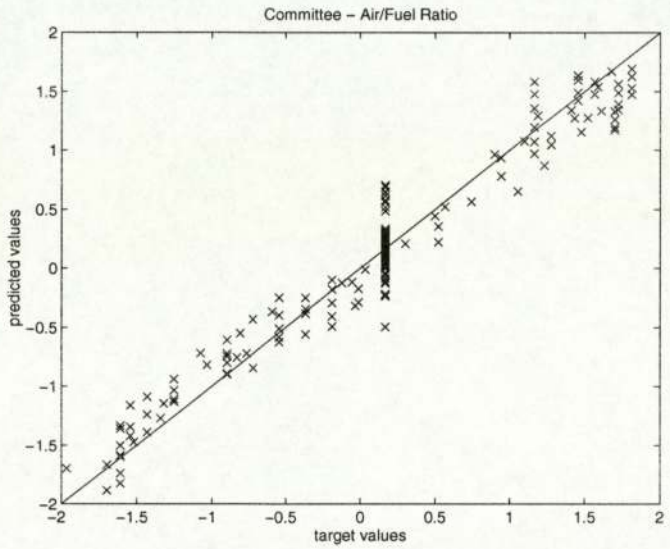


Figure 3.60: Committee: a plot of the predicted air/fuel ratio versus the actual A/F ratio. The desired solution should be distributed close to the diagonal shown in the diagram.

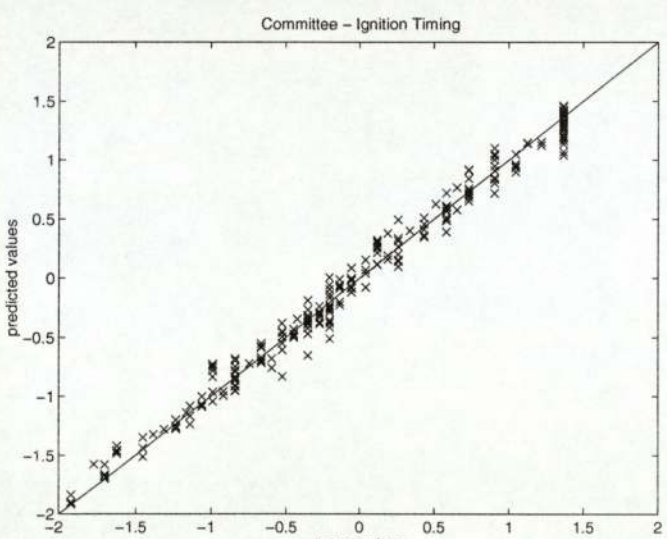


Figure 3.61: Committee: a plot of the predicted ignition timing versus the actual timing. The desired solution should be distributed close to the diagonal shown in the diagram.

Model	RBF	MLP	Committee	Linear
RMS error for air-to-fuel ratio	0.1521	0.2710	0.1936	0.7946
RMS error for ign. timing	0.1156	0.1176	0.1115	0.3767

Table 3.3: Training error for the air-to-fuel ratio and ignition timing.

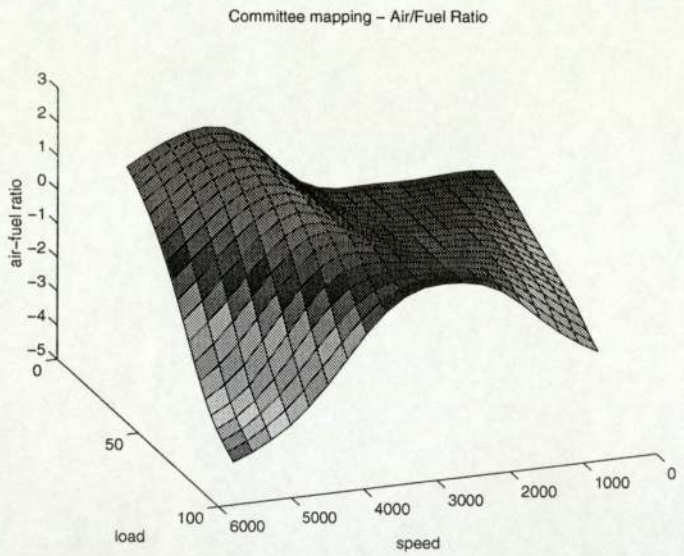


Figure 3.62: Mapping surface for the air to fuel ratio produced by the Committee determined as a simultaneous function of both load and speed.

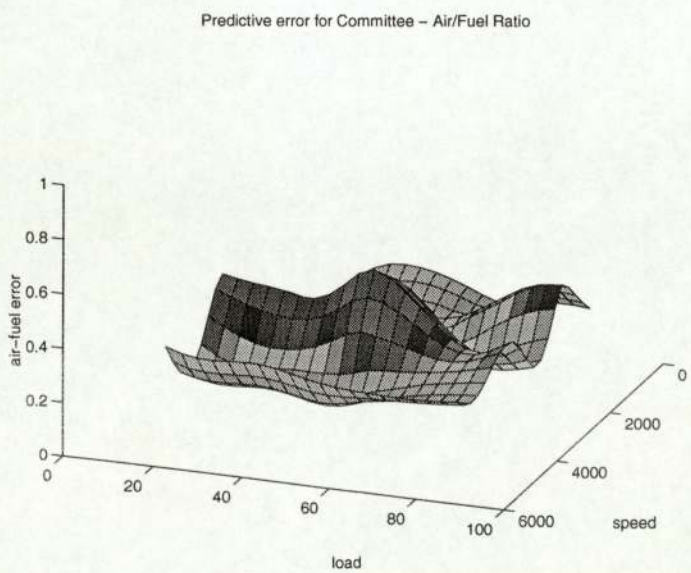


Figure 3.63: Committe predictive error surface for the air to fuel ratio.

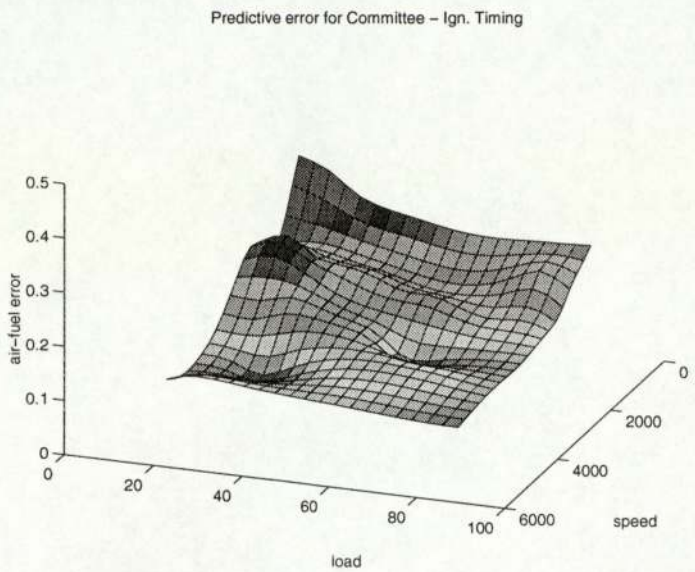


Figure 3.64: Committe predictive error surface for the ignition timing.

Model	RBF	MLP	Committee
Cross-validation error - A/F ratio	0.1346	0.2262	0.1517
Cross-validation error - ign. timing	0.0864	0.0929	0.0820

Table 3.4: Leave-one out cross-validation error for the second data set.

set, next is the committee of RBF and MLP models followed by the MLP network. Again, the linear regression models, used for benchmarking, give large training errors. For the ignition timing, the committee of networks is the best, which is followed by the RBF and MLP respectively.

Exactly the same order of models is reflected in the cross-validation error, shown in the table 3.4.

3.7 Test set results

This section presents results on the test set set for the ignition timing. The test set for the air-to-fuel ratio could not be obtained because of serious technical problems with

processing the raw air-to-fuel data from the third data set.

The models trained on the ignition timing table from the second data set were tested on the third set - gathered from driving a car. Tests were carried out for both steady and transient states, although the models were trained only for the steady states.

As a comparison, the current linear interpolation method was also used for this data.

The Figures 36-38 show target-predicted plots for all considered models: RBF, MLP, committee and linear interpolation.

For all models, including linear, outliers can be noticed for the target ignition value of 10. The further inspection revealed that all these points occurred for small values of load and speed, which corresponded for an idle state of the engine (e.g. waiting at traffic lights or changing gears). In these cases a special a rule must have been applied to reduce the ignition timing to 10.

The table 3.5 presents RMS errors for all steady and transient states, excluding the outliers.

For the steady state, the linear interpolation model appears to give the lowest RMS error, which is followed by the RBF, committee and MLP networks.

It should not be surprising, however, that the linear model gives the lowest fit since the data itself was generated using the same interpolating model. The aim of the project is to replace the interpolating tables with neural networks, which produce smoother control surfaces.

On the transient data, the linear model is again the best, and among neural models the committee provides the best fit. It is followed by the MP and RBF network.

The fact that models trained for the steady state are capable of controlling the transient ignition timing equally well proves that this mappings does not heavily depend on the throttle change rate, which is near zero for the steady state.

On the other hand, the desired air-to-fuel ratio is very sensitive to changes in the throttle.

Model	RBF	MLP	Committee	Linear
RMS error for steady state	0.7155	0.9718	0.7783	0.4970
RMS error for transient state	1.0467	0.9985	0.8346	0.7201

Table 3.5: The RMS errors for both steady and transient states.

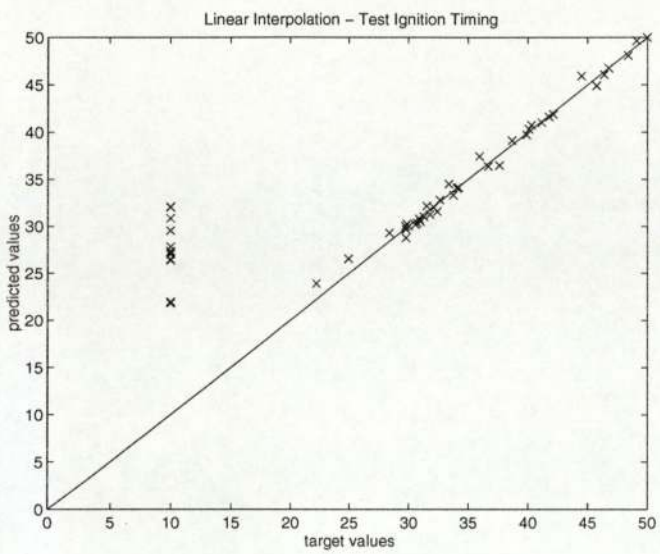


Figure 3.65: The target/predicted plot for linear interpolation .

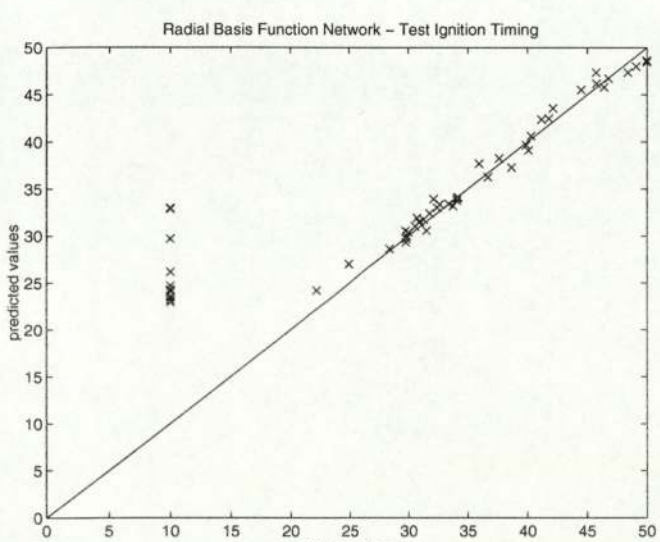


Figure 3.66: Committe predictive error surface for the ignition timing.

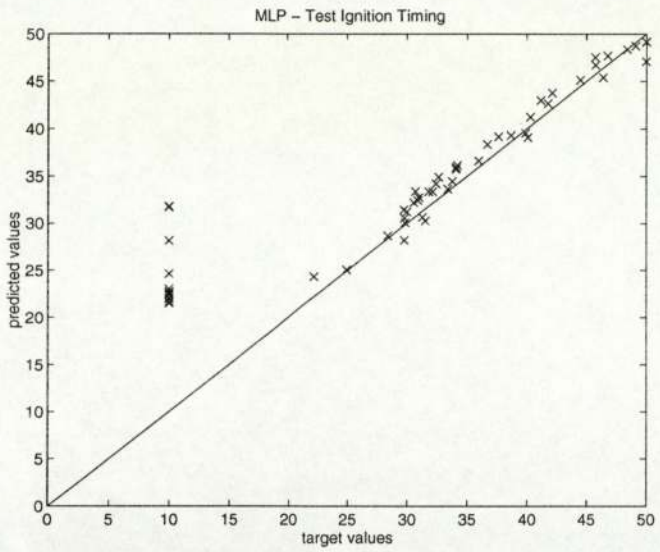


Figure 3.67: Committe predictive error surface for the ignition timing.

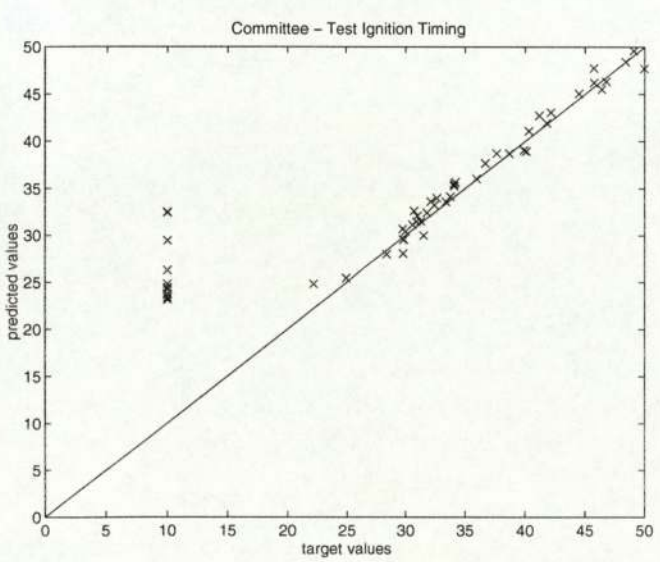


Figure 3.68: Committe predictive error surface for the ignition timing.

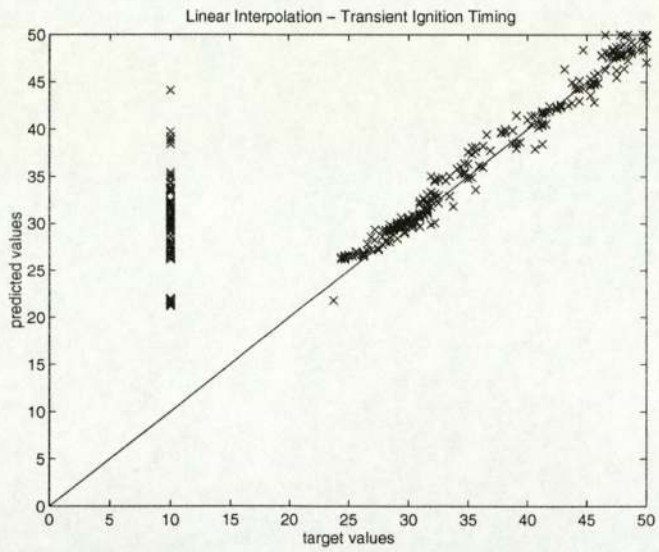


Figure 3.69: Committe predictive error surface for the ignition timing.

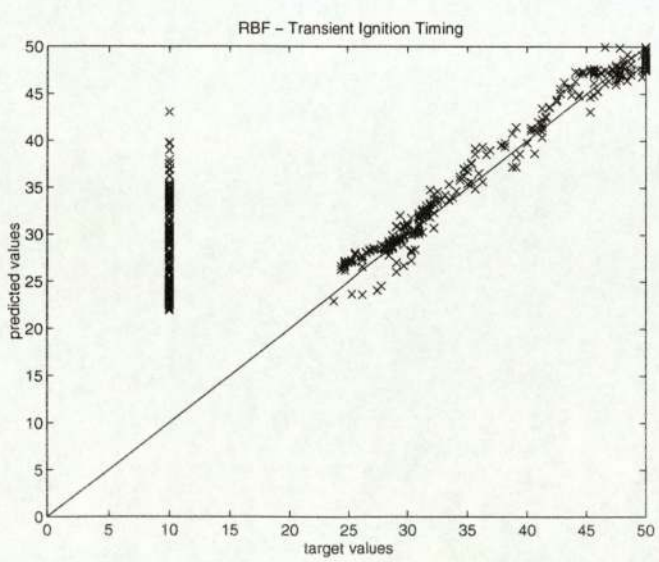


Figure 3.70: Committe predictive error surface for the ignition timing.

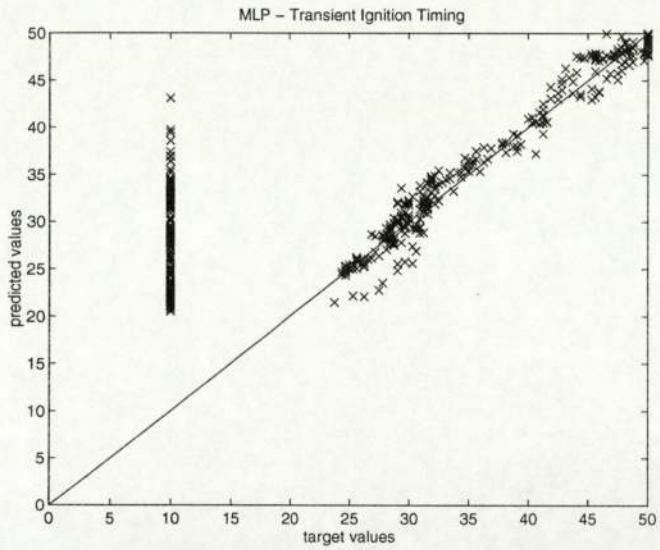


Figure 3.71: Committe predictive error surface for the ignition timing.

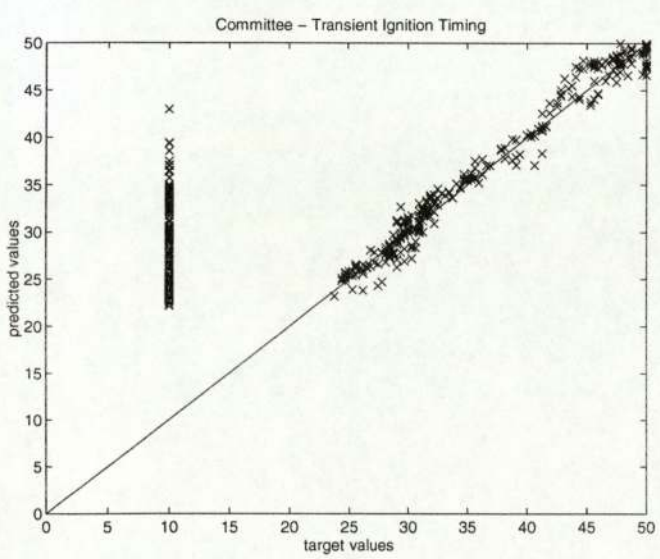


Figure 3.72: Committe predictive error surface for the ignition timing.

3.8 Identified problems

One major problem has been faced so far - data availability.

The data provided was very sparse. Moreover, there are some regions in the mappings which are accessed more often than the others. This can be examined by modelling the input data density. The prior knowledge about the input data density can be incorporated in the error function during training. Next, one could use this prior knowledge and gather more data in regions where the input data density is higher. Obtaining a lot of data in more "important" regions could lower the error bars associated with those input areas [2].

There is one more problem that will occur later - the curse of dimensionality. After moving to higher dimensions the models will require a very large quantity of data in order to adequately cover the input space.

Another problem that is closely related to the above-mentioned ones is a lack of a proper test set. Since the number of training examples was very small, proper testing the neural networks on an independent set could not be done. Therefore, only one method - the leave-one out cross-validation (appendix D) was used to estimate the proper model order complexity and model performance.

3.9 Future directions

The further work will be going in two parallel directions.

First, developed software components will be applied to high-dimensional data that is going to be collected from the prototype engines.

The work will therefore be very useful for the collaborating industry - Sagem Ltd., since they will be able to replace existing system with neural networks, which will result in producing better mappings.

Second, the optimum air to fuel ratios and ignition timings can possibly be further optimised using a custom-designed cost function, e.g. a linear combination of the

engine characteristics.

This will possibly improve the mappings by eliminating possible human errors and and help to achieve the following long-term objectives:

- reduce of the on-chip memory consumption
- maximise torque output
- minimise fuel consumption
- maintaining low level of exhaust gas emissions
- optimise a car drive-ability.

Third, when applying mathematical models to industrial problems, the system designer rarely has enough data points for a variety of reasons (e.g. high cost of obtaining data). Without having a sufficient number of data points, networks cannot generalise and extrapolate well on the domain boundaries.

In such cases it is necessary to incorporate prior knowledge into a neural network. This can be done by manually imposing some boundary conditions on the network solutions. One way of resolving this problem may be to incorporate the first-derivative conditions within selected points that lie on the edges of a data set.

The above-described experiments were conducted for a fully warmed up engine running in a steady state. The inputs were the engine load and speed.

In the next stages the following mappings will be realized:

- steady state - warming

  inputs:

  - speed
  - load
  - coolant temperature

4. Conclusions

The preliminary experiments carried out for the air to fuel ratio and ignition timing data show that non-linear models, such as the Radial Basis Function Networks, Multilayer Perceptrons and the committees of these networks significantly outperform standard linear regression methods.

The models used in the experiments had two inputs: the engine load and speed. The outputs were the optimum air to fuel ratio and ignition timing.

Among the non-linear models, the committee of networks has a low training error (comparable with other models) but the associated error bars are the lowest. Its generalisation capabilities are comparable, in some regions even better, than those of the single RBF and MLP.

These initial studies were very useful in testing the software tools and neural networks developed so far on real data provided by Sagem Ltd.

The neural network approach is feasible and provides a new and more efficient way to construct non-linear mappings of engine parameters.

The literature survey conducted also confirms the feasibility of using neural networks in replacing the current look-up tables.

- transient state - warmed up

  inputs:

  - speed
  - load
  - coolant temperature
  - throttle change rate

- transient state - warming

  inputs:

  - speed
  - load
  - coolant temperature
  - throttle change rate.

Bibliography

[1] D.S. Broomhead, D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks", Complex Systems, vol 2 (1988), pp. 321-355

[2] C. M. Bishop, "Neural Networks for Pattern Recognition", Oxford University Press (1995)

[3] C. K. I. Williams, C. Qazaz, C. M. Bishop and H. Zhu, "On the relationship between Bayesian error bars and the input data density", Proceedings of the Fourth International Conference on Artificial Neural Networks, Cambridge, U.K. (1995)

[4] B. J. Wythoff, "Back-propagation neural networks, a tutorial", Chemometrics and Intelligent Laboratory Systems, vol. 18, 115 (1993)

[5] S. S. Haykin, "Neural networks, a comprehensive foundation", Maxwell Macmillan (1994)

[6] S. Morita, "Optimization control for combustion parameters of petrol engines using neural networks - in the case of on-line control", J. of Vehicle Design, vol. 14, no. 5/6, 552 (1993)

[7] S. Leonhardt, H. Gao, V. Kecman, "Real Time Supervision of Diesel Engine Injection with RBF-based Neural Networks", Proceedings of the American Control Conference, Seattle, Washington, (1995)

[8] P. O'Reilly, S. Thompson, "A neural network air-fuel estimator", Proceedings of the IEE Control Conference, (1994)

[9] G. V. Puskorius, L. A. Feldkamp, "Automotive Engine Idle Speed Control with Recurrent Neural Networks", Proceedings of the American Control Conference, San

Francisco, California, (1993)

[10] S. P. Stevens, T. H. Ma, "Experimental data processing techniques to map the performance of a spark ignition engine", Proceedings of the Institution of Mechanical Engineers, vol. 209, 297 (1995)

[11] C. K. I. Williams, C. E. Ramussen, "Gaussian Processes for Regression", Advances in Neural Information Processing Systems, vol. 8, (1996)

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, "Numerical Recipes in C", Cambridge University Press (1992)

[13] P. Coad, J. Nicola, "Object Oriented Programming", Yourdon Press (1993)

A. Software tools

In the first phase of the project, software tools were created that have successfully served as bases for a further project development.

The language selected for the work was the object-oriented C++ in a 64-bit SGI IRIX 6.02 environment. Figure 33 shows the main class hierarchy.

At the top there is a generic base class "Model" which describes basic properties of any given multiple-input, multiple-output (MIMO) system. Also, it has capabilities of storing a data set consisting of input and/or output vector pairs, which can be used either for training or for evaluating in derived systems.

Apart from this, the class "model" contains such useful functions as

- training the models with steepest gradient descent, conjugate gradients and a modified genetic algorithm
- checking training complexities
- cross-validating with leave-one out method
- preparing a detailed cross-validation profile for a given model complexity
- saving and restoring models from a disk file.

All these functions are virtual and can easily be overridden in derived classes, which allows the entire class hierarchy to be very flexible and modified without spending too much time on this task. Another advantage of such an approach is extensive code reuse.

From the base class "Model" we can derive a class "Linear", which is a standard linear regression model, "Fuzzy Model" and two further base classes: "RBF" and "MLP".

The class "RBF" contains a generic two-stage tuning procedure for all radial basis function networks: first centres of the basis functions are randomly chosen from an input space, next linear mixing coefficients are calculated using pseudo-inverse of the design matrix.

Actual instances of RBF networks are created using two derived classes: "Gauss" and "Spline". In the first class, Gaussian basis functions in the form of

$$\Phi_j(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x} - \mu_j||^2}{2\sigma_j^2}\right) \tag{A.1}$$

are being used. In the latter case, spline functions

$$\Phi_j(\mathbf{x}) = z^2 \log(z), \text{ where } z = ||\mathbf{x} - \mu_j|| \tag{A.2}$$

have been applied.

The next branch in the class hierarchy starts at the "MLP" class. This class contains training and support functions for feed-forward multilayer perceptron networks. From "MLP" two instances of MLP networks can be derived: back-propagation networks with both linear ("LinMLP") and non-linear ("NonlinMLP") outputs. The first one is used for function approximation problems and creating multi-dimensional mappings, whereas the latter can be applied to a variety o fcalssification tasks.

Non-linearity i sintroduced to the models by using sigmoidal hidden and output nodes; activation of neurons is defined to be

$$f(x) = \frac{1}{1 + \exp(-x)}. \tag{A.3}$$

Next, the class encapsulating the Gaussian Processes models has been implemented. The class is derived from the "Model" and the realised model is trained using gradient-descent based methods.

The "Fuzzy Model" class, derived from a generic model, is another tool that can be applied to solving non-linear problems (e.g. fuzzy control). It uses a fast one-pass

"table look-up" method to infer fuzzy logic rules from the training data. Triangular membership functions (a class "Fuzzy") are used for both inputs and outputs. The centre of gravity defuzzyfier is being used to obtain crisp outputs from fuzzy sets.

Fuzzy logic models are widely used in fuzzy control - modelling highly non-linear control surfaces. Their advanteges are speed of learning and linguistic interpretation of the fuzzy rules.

There have been made some analogies between neural systems and fuzzy models which enable further training the models using neural networks techniques. These structures are called "neuro-fuzzy" and such hybrid approach allows the production of fine-tuned controllers. Also, genetic algorithms have been recently used to find optimum sets of fuzzy rules and membership functions [11].

Design of all above classes used "generalisation-specialisation" method for designing objects and inheritance links between them.
Further procedure that can be used is called "whole-part", according to Coad/Yourdon object-oriented methodology [12].

The "whole-part" method has been applied to designing a class "Expert Net" - a committee of linear, RBF and MPL models.

In the figure 33 we can see that the committee network has three major components: one linear model a number of different RBF and MLP neural networks.
The training procedure i scarried out independently for each component. The outputs of the expert network are taken as the weighted average of outputs from component models. At the moment all weights are equal but later on they can be adjusted according to different model characteristics (e.g. validation set error).

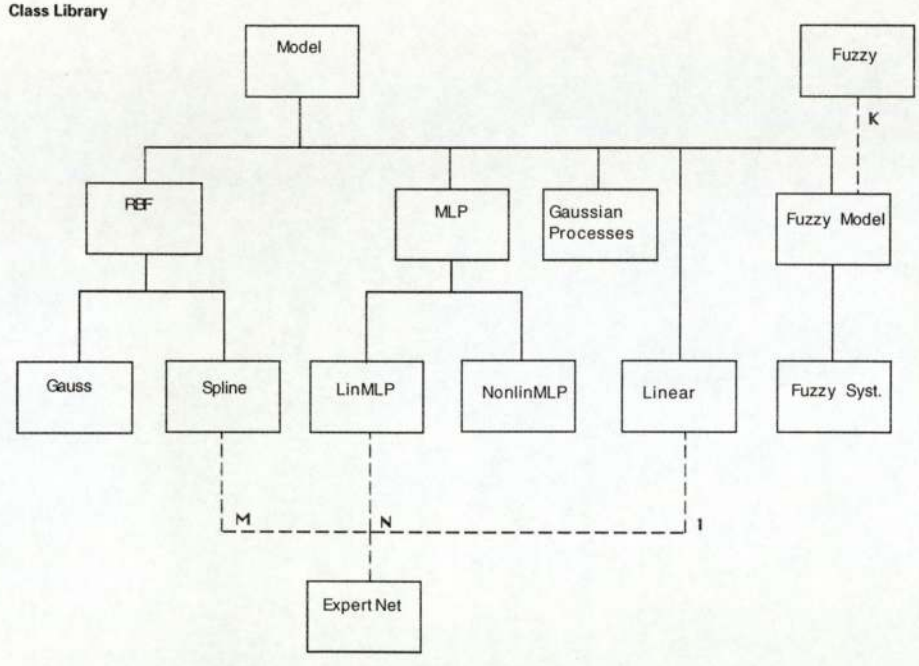


Figure A.1: C++ class hierarchy