Chapter 6

Rolling Mill & Robot Actuator Modelling Examples

To show the methods in use for practical applications, the local model networks and learning hierarchies of models networks are applied to real problems, with data measured from physical systems.

They are used to produce a predictor which can model the roll gap in a rolling mill based on data sampled from a real mill. The Local Model net produced the best results in terms of accuracy, and generalisation ability, compared to multi-layer perceptrons, MARS, linear models and RBF networks. The interpretability of the trained network is discussed and visualised.

The methods are applied to a second example, a robot actuator modelling problem.

6.1 Rolling Mill Problem Description

To demonstrate the practical applicability of the modelling structures, methodology and algorithms developed in this thesis, the methods were applied to the task of modelling a rolling mill from real data. Rolling mills, are examples of complex nonlinear processes, where a wide variety of physical effects play a significant role. Despite the obvious usefulness of rolling mills, and long practical experience with their operation, the physical models developed for them tend to be highly complex (too complex for on-line use), and because of the uncertainties in industrial processes the models tend to be of little practical use. In practice, the control algorithms used in real mills tend to be simple enough to be adapted by the operators, and are often developed in a very heuristic manner. This suggested that the use of learning methods to better model the poorly understood characteristics of the rolling process would be a productive path to follow.

The mill investigated is shown in Figure 6.1, and is a four-high single stand aluminium rolling mill. A coil is placed on the left and the material is threaded into the roll bite, where the material is reduced and exits to be collected again in a coil on the other side.

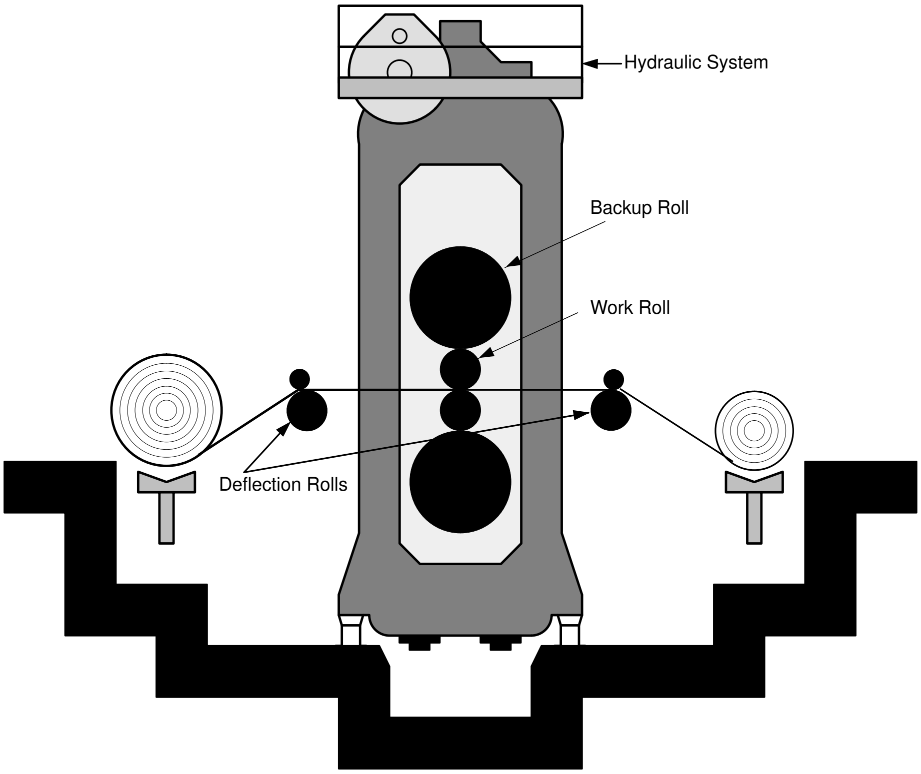


Figure 6.1: Single stand of a rolling mill.

The strip moves with a velocity v_1 and enters the roll gap with a thickness h_1. It exits with a thickness h_2 and a velocity v_2, preserving the mass-flow relationship,

$$h_1 v_1 = h_2 v_2. \tag{6.1}$$

The strip is deformed by two cylindrical rolls which apply a force f_w to the material to produce a flat strip of a pre-determined thickness, as shown in Figure 6.2.

The roll gap is varied by exerting force on the backup roll bearing. This force is transferred to the centre of the backuproll, to the centre of the work roll and then to the strip. Flat strip is produced when the force on both sides is equivalent. Roll gap position is measured on both sides and is averaged to give the *roll gap* variable s_0. The setting for the roll gap is a function of strip input thickness and the elastic deformation of the surrounding mill housing and roll bearings (shown in Figure 6.2 as springs).

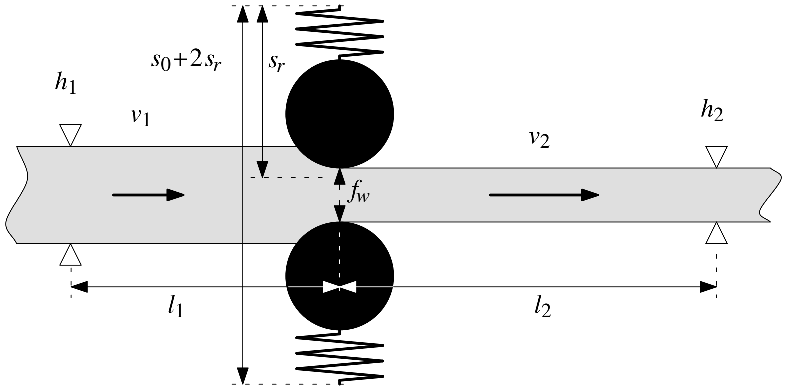


Figure 6.2: Roll bite

6.1.1 Nonlinearities in the rolling process

The rolling process is nonlinear in a number of ways. The deformation characteristics of the rolled material, the work rolls and the surrounding mill housing are nonlinear. The material undergoes elastic deformation, then plastic deformation, followed by elastic relaxation. This, combined with the elastic deformation of the work rolls and mill housing, is a process which is very hard to describe with simple differential equations. Further complexity is added by the effect of variations in temperature on the process (materials change their properties, the rolls swell). Lubrication effects further confuse the situation, and are very difficult to model from first principles.

6.1.2 Measurement noise and disturbances

Many of the disturbances in the modelling process are obviously closely related to the complex, unmeasured and poorly understood processes described above. As can be seen in Figure 6.2, the sensors for the measurement of h_2 are a distance l_2 away from the roll bite, leading to a dead time between actuation and sensing which varies with operating speed v_1. The sensors are usually X-ray gauges measuring the thicknesses h_1 and h_2, and these measurements tend to be very noisy. The rolling force can be measured either directly (the more accurate way)by load cells, or indirectly by using pressure transducers from the hydraulic cylinders and converting this quantity to force. The methods used in this application are the *indirect* methods. The strip velocities are calculated from the deflection rolls, and the work roll velocity is calculated using the tacheometer of the drive system. In each case the measured quantity is the angular velocity, which when combined with the roll diameter can give the actual roll speed. Due to the complex nature of the disturbances and the poorly understood interactions, it did not seem promising to build explicit disturbance models.

6.1.3 Modelling goals

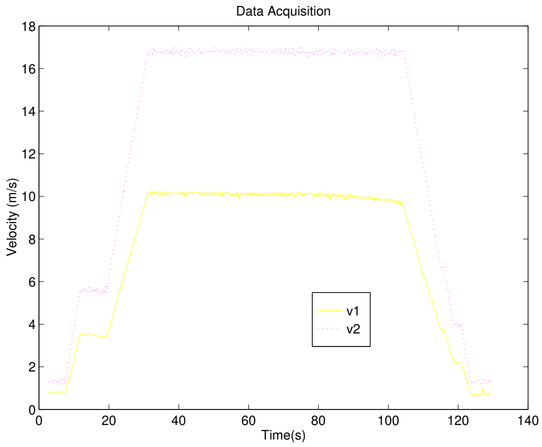
The goals of the learning task were therefore to try to produce a model of the system which reliably represented the data, and which could be used in a model-based control algorithm. The variable of interest is dh_2, a measure of the deviation from the reference output thickness. At present, the acceleration and deceleration phases are poorly modelled and controlled, so an improvement in model accuracy in those phases would be a major contribution to making the rolling process more economic by avoiding wastage caused by roll material which does not meet the accuracy specifications. In the general case it is obviously desirable to have a system which can cope with different types of metal and different reduction schedules. One possibility is to hand segment the training data into different training sets for different types of metal, then to bring them together committee-style using a classification network to decide which model matches the current strip best. Another possibility is to amalgamate a variety of independently trained models, to try to produce an 'average' general model. The method used in this thesis, however, concentrates on producing a single model from a training set composed of a variety of training runs, from a single type of strip.

Important areas of the input space

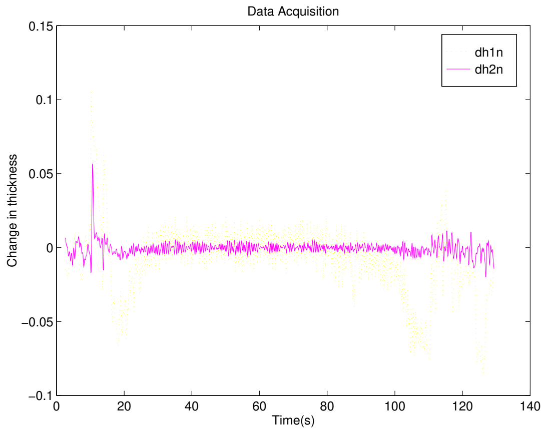
In many modelling applications there are given areas of the input space which are especially important to model accurately, because the model-based system needs particular accuracy in a given situation, bandwidth or region. In the rolling process, the long periods of relative stability, where the velocity does not change, are obviously important, as despite the fact that the modelling here is easier, this phase is where the mill produces most of its product. The acceleration and deceleration phases are, however, important because it is much more difficult to achieve an accurate model, so more data is needed to produce robust accurate representations of the physical process. The trade-off between the two areas is one for the engineer to decide on. (A further possibility is of course, to treat the three areas separately, by segmenting the training data, and associating a local model with each phase of the operation)

6.2 Rolling Mill Training and Test Data

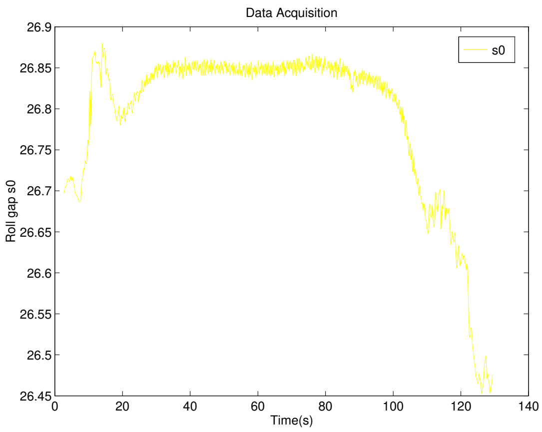
We have described the complex nature of the problem, which makes it clear that there is little hope of a clean, uniform, conflict free training set. Each run of a rolling mill is going to produce slightly different results, many of the reasons for which cannot be directly measured (e.g. temperature differences, different roll materials, possibly different machine settings). To give an impression of the distribution of the data in the input state-space, the correlation between various inputs is shown in Figure 6.4. To improve the numerical robustness of the learning process, the data was normalised so that each variable was distributed between 0 and 1. The normalised data are shown in Figure 6.5.



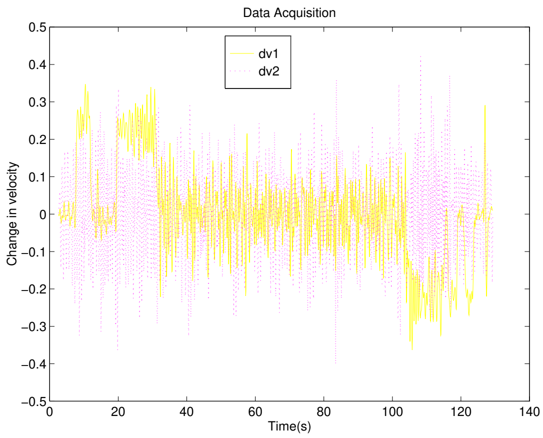
(a) Mill Velocity (Input/Output)



(b) Strip Thickness (Input/Output)

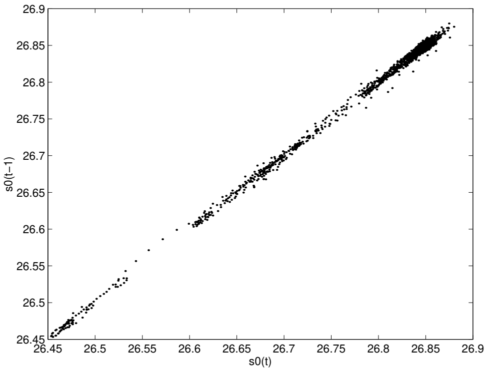


(c) Roll Gap

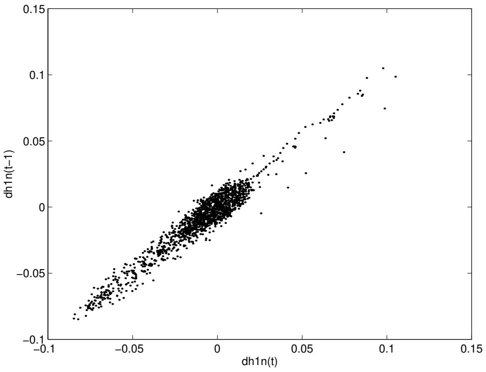


(d) Velocity derivatives

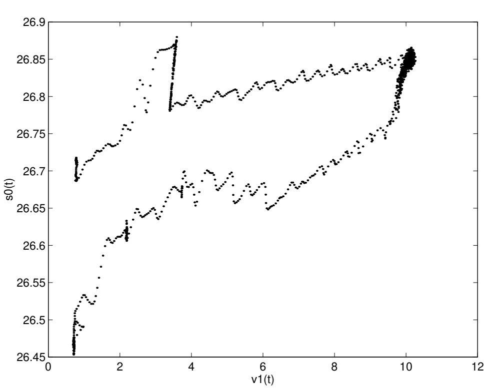
Figure 6.3: Observed data from mill



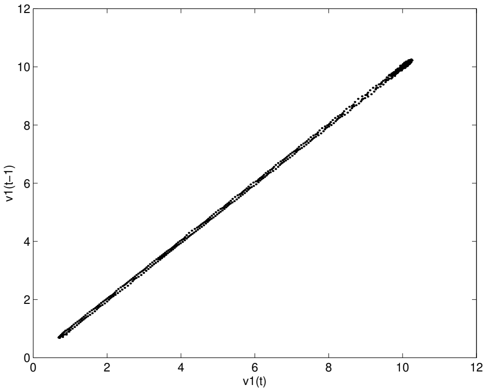
(a) Roll gap (s_0) correlation



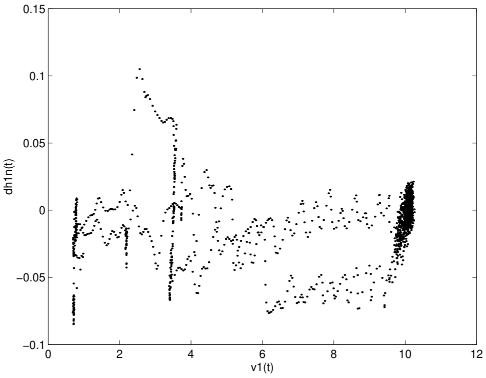
(b) Input thickness (dh_1n) correlation



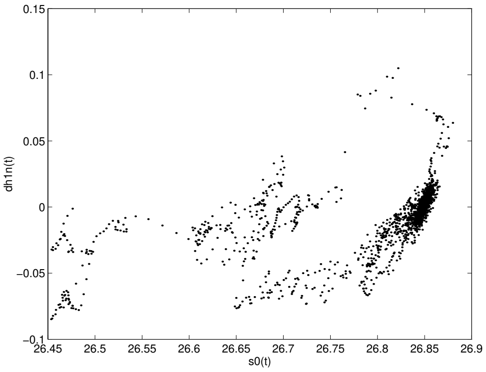
(c) Speed against Roll gap correlation



(d) Input velocity (v_1) correlation



(e) Input thickness (dh_1n) against input velocity (v_1) correlation



(f) Roll gap against Input thickness correlation

Figure 6.4: Observed data from mill – state portraits for a single run

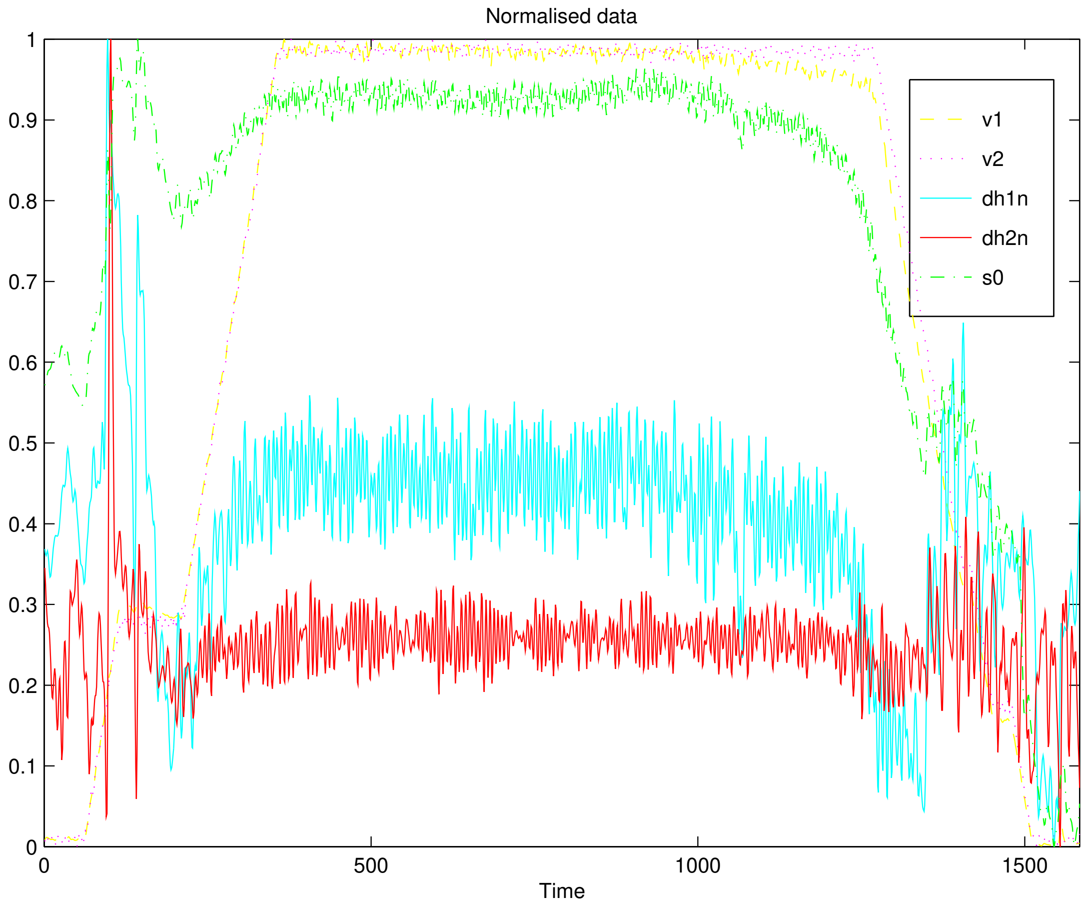


Figure 6.5: Normalised training data

Notation

dh_1	input thickness deviation
dh_2	output thickness deviation
dh_1n	input thickness deviation (time aligned)
dh_2n	output thickness deviation (time aligned)
v_1	strip input velocity
v_2	strip output velocity
s_0	roll gap
f_w	roll force

6.2.1 Pre-processing used

Filtering and time alignment

The data used for modelling is low-pass filtered and down-sampled to a more appropriate frequency for the modelling task. The filtering was done offline using a 32-order FIR filter and the MATLAB *filtfilt( )* routine, which results in zero-phase distortion after passing through the data set in both forward and backward directions.

One of the main difficulties in modelling aspects of rolling mills is that delays in sensor measurements are inevitable, due to the distance of the thickness gauges from the roll bite. These delays are also velocity dependent, so the delay for the forward sensors is

$$t_{d1} = \frac{v_1}{l_1}, \tag{6.2}$$

where v_1 is the input velocity, and l_1 is the distance from the roll bite, as shown in Figure 6.2. To make the modelling task more straightforward, the data was pre-processed so that it would be referenced to the roll bite, such that the new thickness dh_1n is given by

$$dh_1n(t) = dh_1(t - t_{d1}), \tag{6.3}$$

where at the sampling rates used, t_{d1} varies from eight sample delays at low speed, to 2 sample delays at top speed. The output thickness is analogously,

$$dh_2n(t) = dh_2(t + t_{d1}), \tag{6.4}$$

where $t_{d2} = \frac{v_2}{l_2}$. This time shifting was validated by checking correlations between the thickness variation and the roll force signal, confirming that the time-shift was accurate to within one sample.

Sampling rate

The mill has a variety of subsystems with different bandwidths–the hydraulic system has a bandwidth of 15-20Hz, the roll eccentricity effects are 12-15 Hz, but the most important effects are dominated by the slowest subsystems. From experience, the sampling time needed to cope with the change in response from the moment of sensing a disturbance, actuating the hydraulics and moving the rolls is between 1-2Hz. This is what was used in the design of the filtering and sampling algorithms.

The original data was sampled at 100Hz. Using the old rule of thumb that the sampling rate should be around 8-10 times the closed loop bandwidth (Ljung, 1987), the data was downsampled to a frequency of $f_s = 12.5$ Hz.

6.2.2 Planning the experiment

The problems of acquiring the data in many industrial processes are non-trivial, involving a great deal of time and money. In this case the data existed already, having been taken from a real plant under normal operating conditions. We had no influence over the experiments performed, and new experiments were deemed too costly to be worthwhile. Four different runs, each similar to that shown earlier were used for training, and the data selected from the combined data sets. The sets used to produce the training data are shown in Figure 6.6. The data are normalised to be within the limits defined by the maxima found in training and test sets. The test data used to validate the trained model were taken from five different runs with the same type of strip. The sets used for the validation are shown in Figure 6.7.

Open questions

The methods used to prepare the training data all seem to be practical measures aimed at a concrete engineering solution, but there are still many open questions. How many data do we require to be confident that our model is general enough? How will the model react to different materials, or reduction schedules? Will hot mills, where the input thickness is more variable need more runs?

6.3 Rolling Mill Modelling Results

6.3.1 Modelling specifications

Model structure and order

The local model structures used were linear models based on delayed inputs (equivalent to an FIR filter structure).

$$f_i(\boldsymbol{\psi}(t)) = \boldsymbol{\psi}(t)\boldsymbol{\theta}_i, \tag{6.5}$$

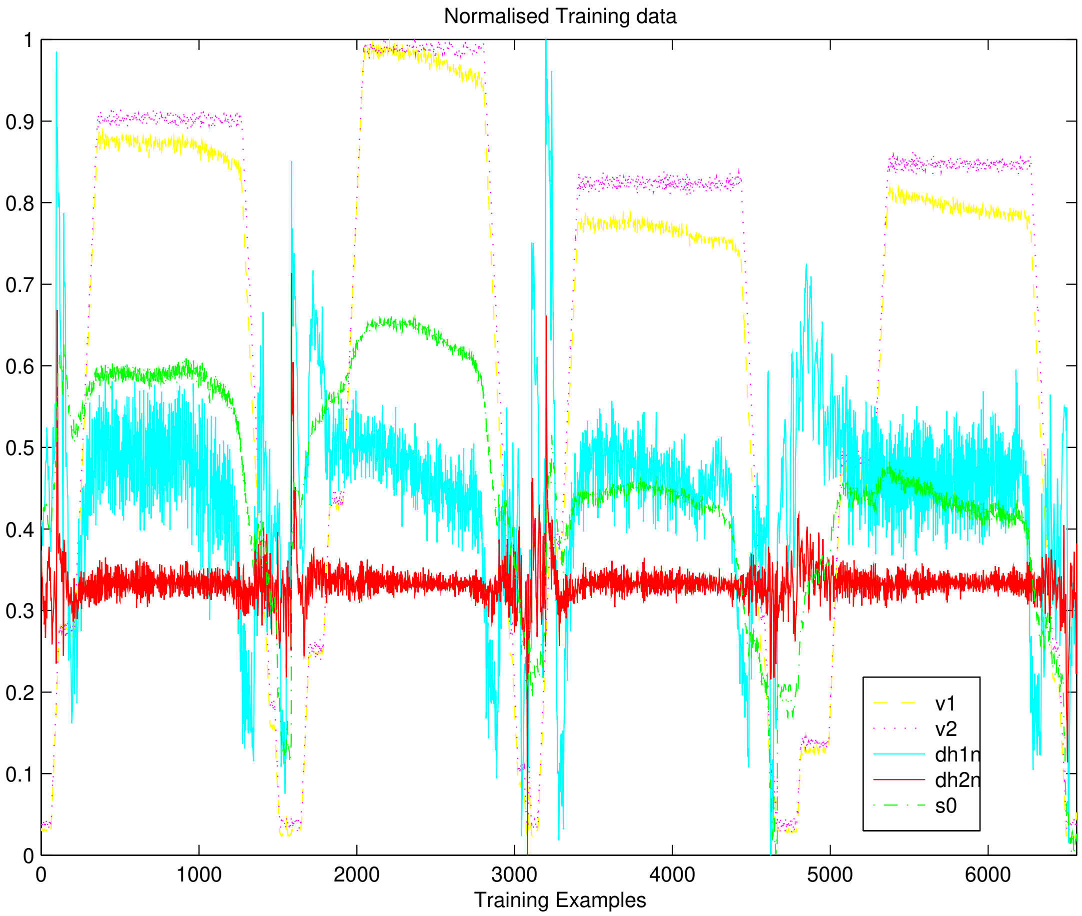


Figure 6.6: Combined training sets

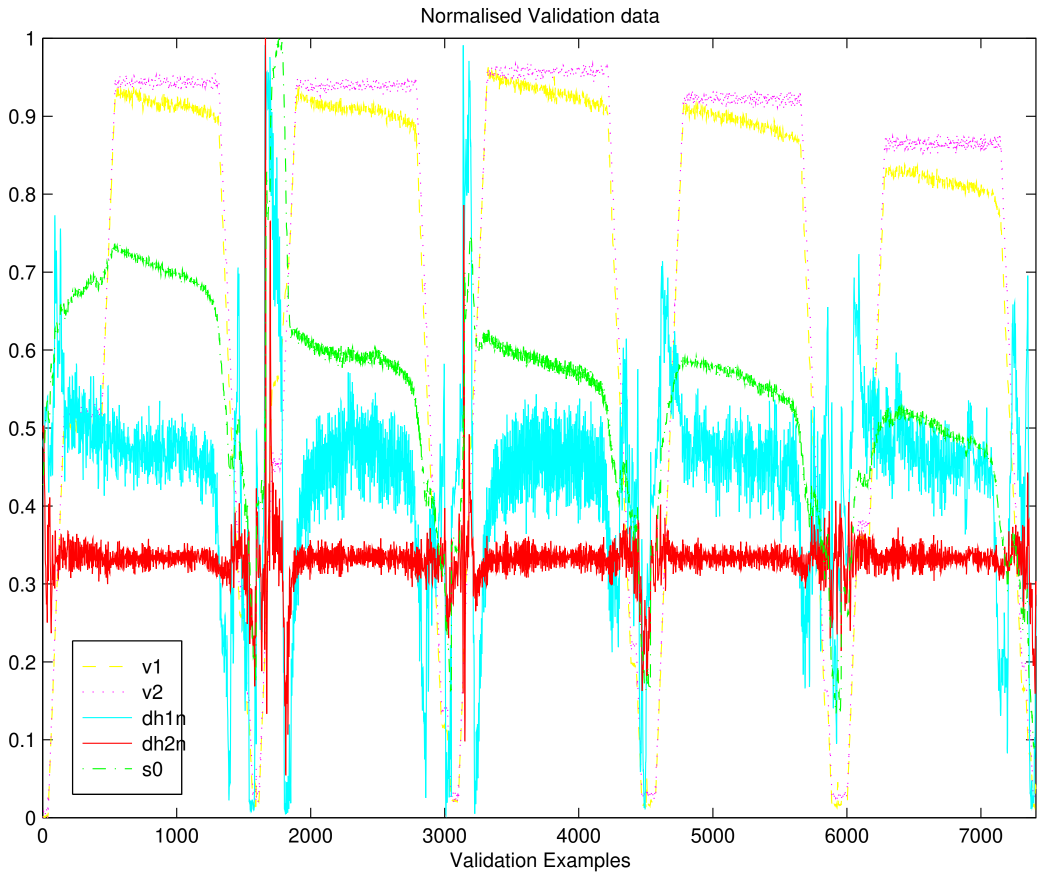


Figure 6.7: Combined Validation sets from 5 different strips of the same material. x-axis indicates time, y-axis indicates the normalised value of the various variables.

where

$$\boldsymbol{\psi}(t) = [v_1(t-1), s_o(t-1), dh_1n(t-1), \ldots, v_1(t-n_u), s_o(t-n_u), dh_1n(t-n_u), 1] \quad (6.6)$$

The previous measured output thickness could not be used for control purposes, because of the dead time in the output measurement process. The order n_u of the processes involved was not clear in advance, so models with a variety of dynamic orders were used. In general, models with an order of around $n_u = 7$ performed best. It is not clear whether this is due to high order dynamics or low order dynamics with the effect of unknown dead times.

Selection of the cost function for optimisation

The cost function used will depend on the noise on the training data and the relative importance of the area of the input space. In this case we simply applied the quadratic cost function, as the active learning and constructive algorithm would automatically devote more resources to the more complex acceleration and deceleration phases.

Parameter selection for learning algorithm

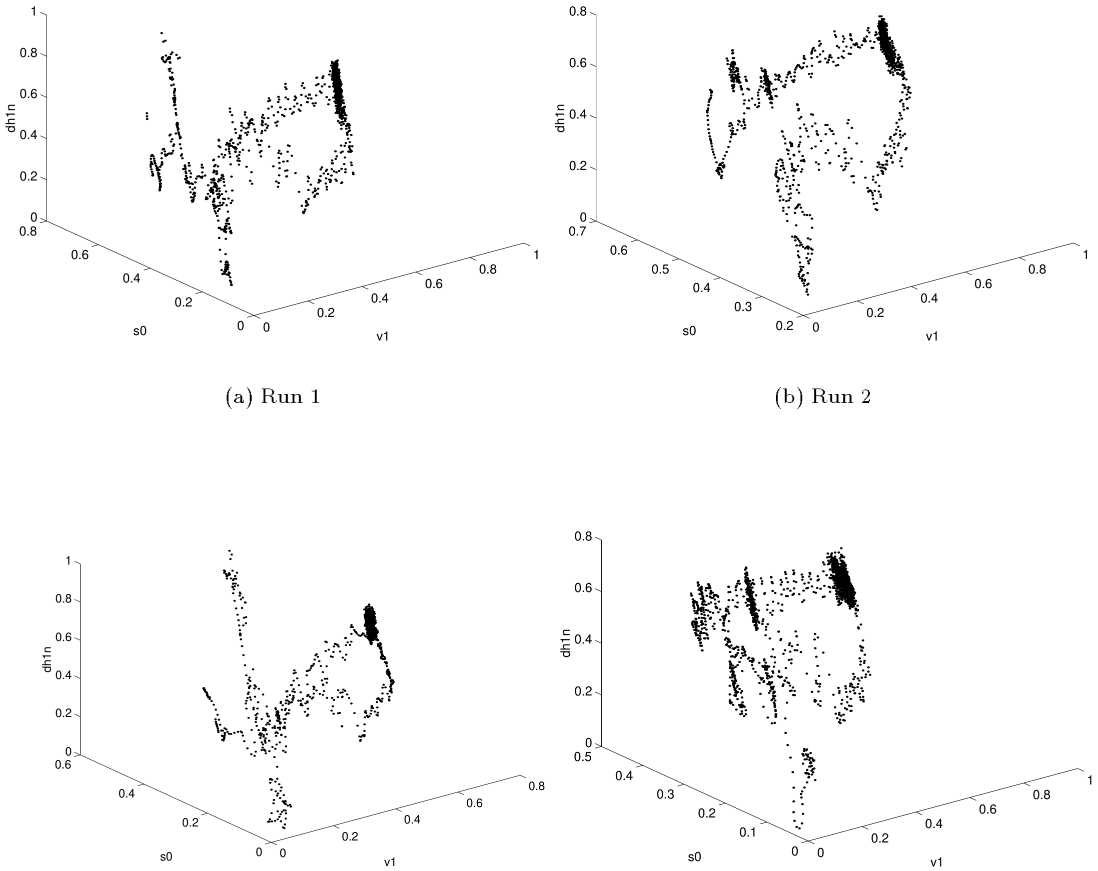
Ideally, the need to fiddle with the parameters of the learning algorithm should be kept to a minimum. In practice, this can prove to be highly important for the modelling process. The results in this chapter with the local model nets were obtained using the following parameter settings: $\sigma_{max} = 0.2\sqrt{n_{\tilde{\phi}}}, \lambda = 0.6, n_{res} = 3, \gamma = 0.5\sqrt{n_{\tilde{\phi}}}, N_{max} = 1000, N_{min} = 10n_{\psi}, n_{cutoff} = 4$. The distance metrics used for the basis functions were axis-orthogonal ellipses. The basis functions in all nets were normalised.

Reducing the operating space

In order to allow a more straightforward and robust model structure, the dimension of the space in which the operating regimes are placed was reduced. The operating point $\tilde{\boldsymbol{\phi}}$ is defined by the inputs s_0, v_1 and dh_1n, so these are the variables used to place the basis function centres. The local models inputs $\boldsymbol{\psi}$ are composed of s_0, v_1 and dh_1n, delayed n_u times, where n_u is the order of the tapped delay line on the inputs. Other variables were not found to contribute significantly to a reduction in the model error.

As mentioned in Section 2.5.4, in some applications it makes sense to form a rough partition of the input space initially, as seems logical to the development engineer. The results in this chapter did not use this technique, although it may prove useful in future work (e.g. to form an explicit partition of the acceleration and deceleration phases from the normal operating speed phase).

The distribution of the data points in the basis function space (this can also be viewed as the state-space in which the current operating point $\tilde{\boldsymbol{\phi}}$ is defined) is shown in Figure 6.8. The four runs which make up the training set are shown here.



(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure 6.8: Training data distribution in operating point space over the four runs. See Figure 6.4 for 2-dimensional slices fo the space.

Use of cross-validation to estimate robustness

The results given in this chapter are all one-step-ahead prediction results. As noted earlier, models of dynamic systems should normally be validated by applying the exogenous inputs and letting the system run, feeding back the model states as opposed to the measured states. For the rolling mill, however, the model is not autoregressive as the dead time involved in the sensor feedback would mean that the data would not be available for use in a control algorithm. The one-step-ahead prediction is therefore the ultimate off-line test in this case. The potential problems involved in on-line application of the model for control purposes are described later.

6.3.2 Benchmark algorithms

Linear modelling results

To provide a straightforward benchmark for the local model nets, we trained a simple non-autoregressive linear model (as in equation (6.5)) using the same data as the LMN. The results were surprisingly good for such a simple structure, as shown in Figure 6.9.

Figure 6.10 shows a cross-validation plot of the linear modelling results, showing that they are consistent over the different folds, as would be expected for such a simple model.

MLP modelling results

To provide a benchmark with the more traditional neural networks methods, a multi-layer perceptron was applied to the problem. A network with a single hidden layer of 30 neurons with sigmoidal activation functions was used. The output unit was a linear one. The training was carried out over 8000 runs through the training set of 6497 patterns, presented in a random order. The straightforward back-propagation algorithm was used with a learning rate of 0.1. (Other settings were tried, but had little effect on the outcome.) The modelling accuracy was found to be not as good as the MRC algorithm's, and training duration was several days, as opposed to minutes for the other methods. An interesting comparison is with the simple linear model, which proved to be more accurate than the potentially powerful MLP. The reasons for the MLP's poor performance were not investigated in detail, but the problem proably lies in the inefficient learning algorithm, back-propagation.

MARS results

As a benchmark with the methods developed in the statistical community the MARS structure identification algorithm was used. The results again, were not as good as the linear, or Local Model or LHM methods, as shown in Figure 6.12. This is probably due to the axis-orthogonal nature of the partitioning of the input space, combined with the lack of local linear models in the standard setup.

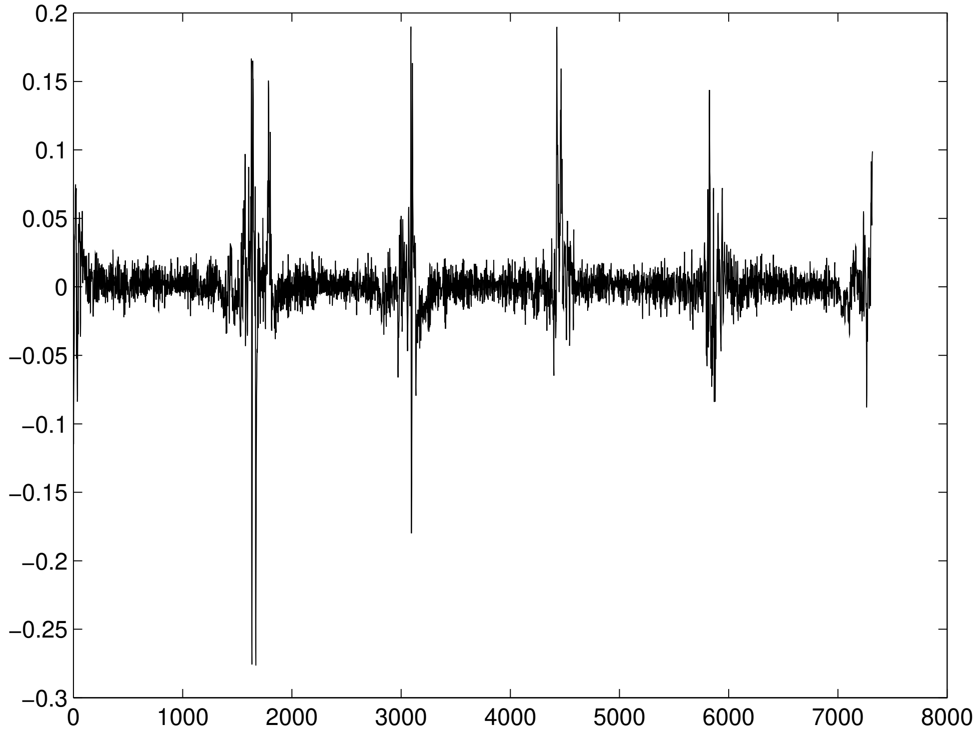


Figure 6.9: Mill model residuals for linear model on the validation data. Mean squared error = 0.00049. The plot shows the error residuals on the combined thickness modelling results from 5 strips used in the validation runs shown in Figure 6.7. The x-axis is time. y-axis indicates magnitude of the residual. Note the large errors in the acceleration and deceleration phase.

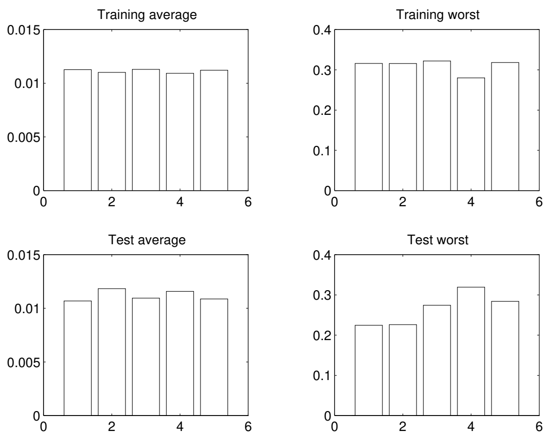


Figure 6.10: Cross-validation results for linear model

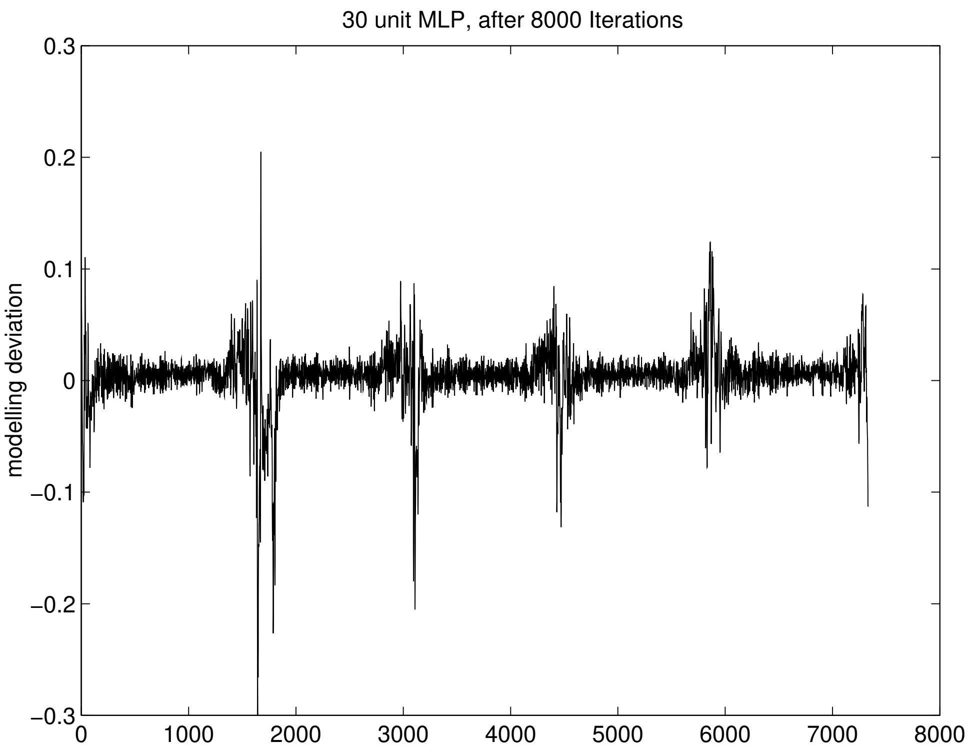


Figure 6.11: Mill model residuals for MLP with 30 units on the validation data. Mean squared error = 0.00065.

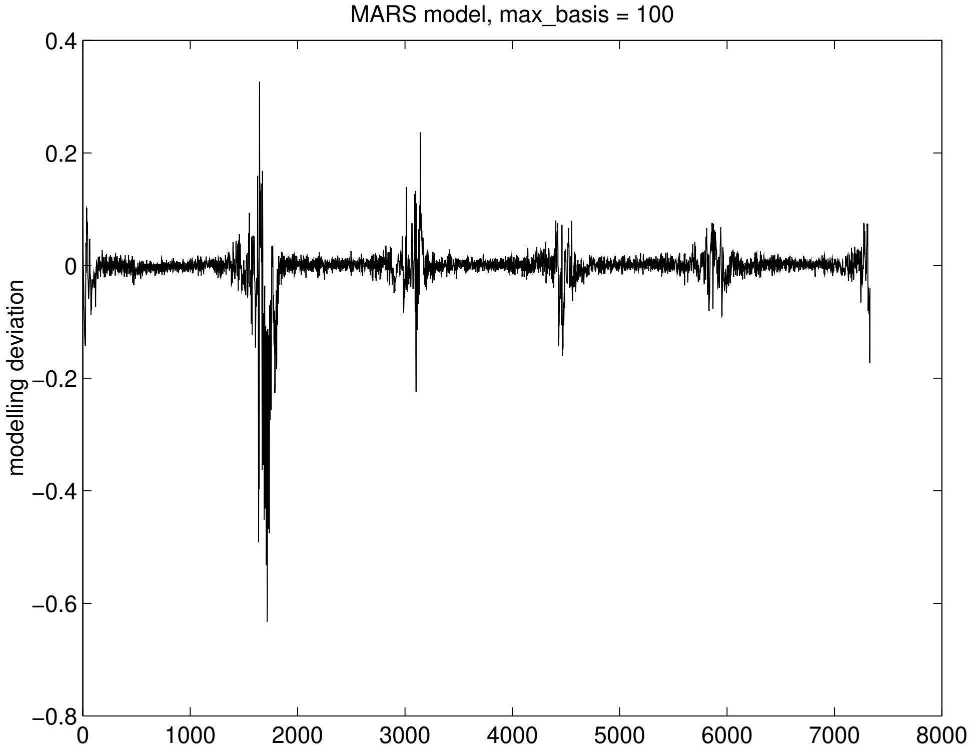
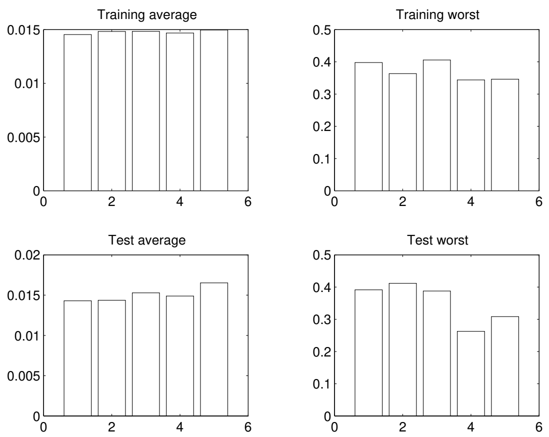
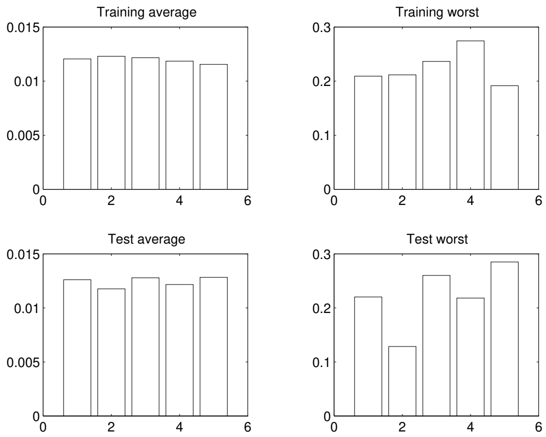


Figure 6.12: Mill model residuals for MARS on the validation data. Mean squared error = 0.0017. Large errors especially in the acceleration and deceleration phase. Model even worse than a linear model.



(a) Cross-validation results for MARS with maximum of 20 basis functions and $n_u = 7$



(b) Cross-validation results for MARS with maximum of 100 basis functions and $n_u = 7$

Figure 6.13: Cross-validation results for MARS on mill data. Reasonably robust with a cross-validation on the training data, but performed badly when face with data from a totally different run.

6.3.3 Local model net results

Local learning

The error results for a LMN with 21 models, trained locally are shown in Figure 6.14

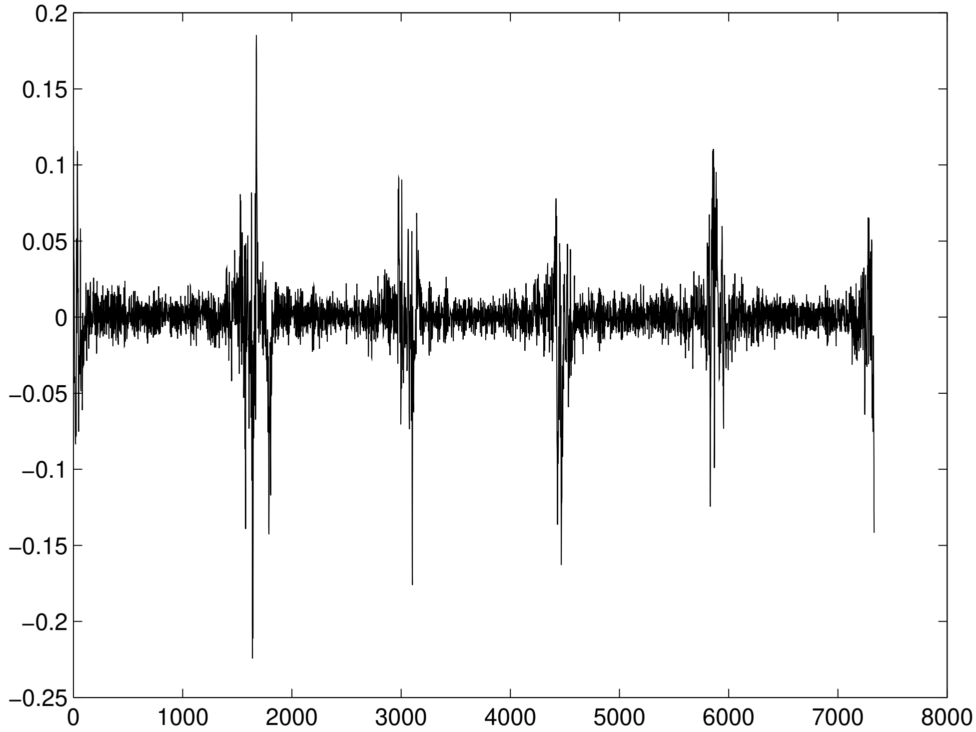


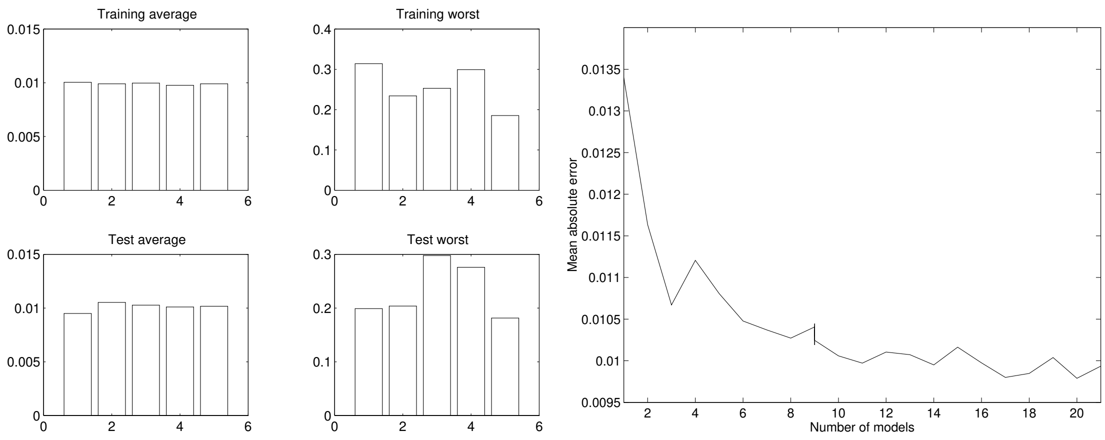
Figure 6.14: Mill model residuals for LMN (local training) on the validation data. Mean squared error = 0.00040.

The mean absolute errors on the training set at each stage of the local learning model construction are shown in Figure 6.15(b). The cross-validation results are given in Figure 6.15(a), and indicate that the approximation process is relatively robust.

Other parameter settings for learning were used, for example, letting the training continue for a further resolution level, which lead to the smallest training residuals, with an average absolute error of 0.85

Global learning

The error results for a LMN with 23 models, trained globally are shown in Figure 6.14. The global method used was SVD, where singular values which were a factor of 10^4 smaller than the largest singular value were zeroed. Note the poorer performance in the acceleration and deceleration phases, compared to the locally trained model.



(a) Cross-validation results for Local Model Net construction. The results are relatively consistent.

(b) Local Learning Error Curve. The glitch in the graph indicates that a model was removed at this stage, due to a lack of training data.

Figure 6.15: Cross-validation results for net construction, and training error development during a single construction run.

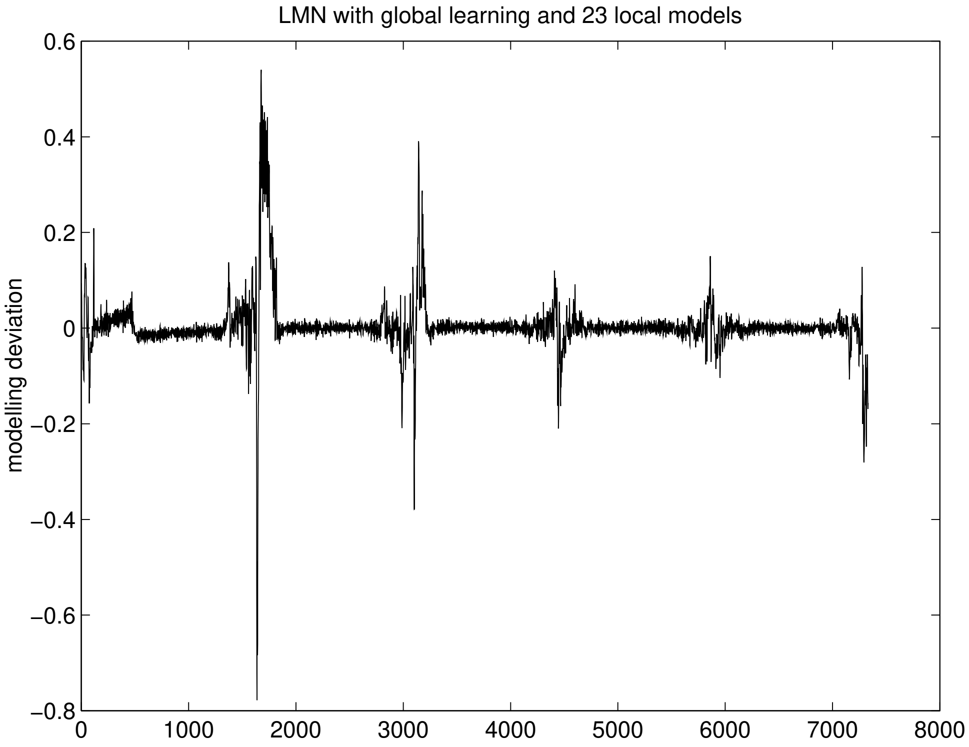


Figure 6.16: Mill model residuals for LMN (global training) on the validation data. Mean squared error = 0.0036.

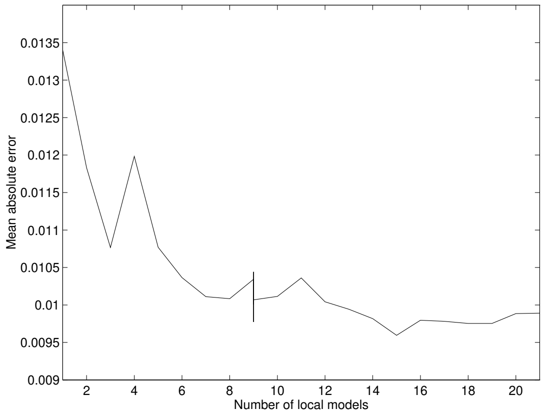
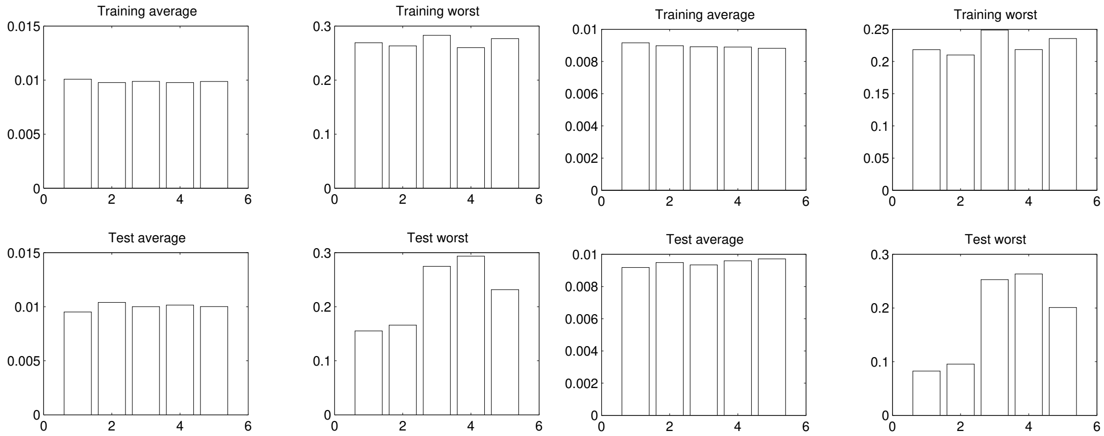


Figure 6.17: Global Learning Error Curve. Global optimisation performs well on training set, but poorly on validation set. Glitch in curve indicates pruning of models.

To compare the robustness of the local and global parameter estimation, independently of the structure identification, the structure which produced the residuals in Figure 6.14 was fixed, and a cross-validation experiment was carried out, where only the local model parameters were adapted, both locally and globally. The errors are, in general, slightly better in the global case. How does this fit in with the residual plots in Figures 6.14 and 6.16, where the local results are clearly better? The difference is that the validation results from the cross-validation done here use data points extracted from the rolling runs used for training, whereas the residuals are plotted on validation data from completely fresh runs, which provide a better test of the modelling framework.

Summary of results on validation data

Table 6.1 summarises the modelling results tested on the validation data. The Local Model nets consistently provided the highest level of accuracy, producing both the smallest average errors and smallest maximum errors. Surprisingly, the simple Linear model performed better than MARS and Multi-Layer Perceptrons. This is not to say that an MLP could not be trained to produce better results, but it does indicate the lack of robustness in the widely used learning algorithms, such as back-propagation, which have trouble identifying a linear system.



(a) Cross-validation results for a fixed structure Local Model Net with local learning of parameters.

(b) Cross-validation results for a fixed structure Local Model Net with global learning of parameters.

Figure 6.18: Cross-validation Comparison between local and global learning. In both cases the parameter estimation process seems to produce robust results. However, the residuals on completely new validation data (shown in Figures 6.14 and 6.16 show that local learning produced better generalisation ability).

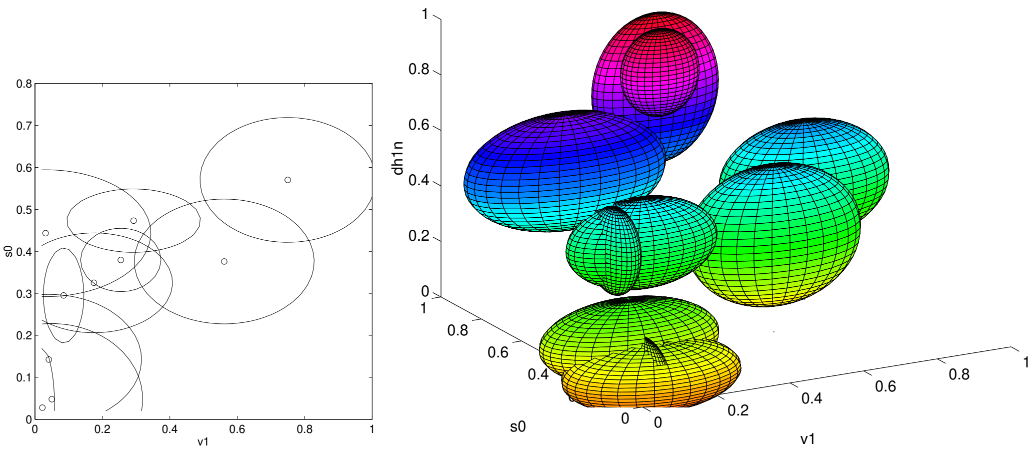
Model Type	Mean abs error	Mean squared error	Max abs error
Linear model	0.0116	0.00049	0.28
MARS (19 BF's)	0.0168	0.0014	0.45
MARS (85 BF's)	0.0162	0.0017	0.63
RBF (39 Basis Functions)	0.0185	0.0017	0.72
MLP (30 Units)	0.0144	0.00065	0.29
LMN (gl, axis-orth)	0.0235	0.0036	0.78
LMN (ll,RBF, n_u =2)	0.0151	0.00068	0.32
LMN (ll, no active)	0.0118	0.00059	0.32
LMN (ll,RBF, n_u =5)	0.0115	0.00045	0.26
LMN (ll, ellipse)	0.0116	0.00044	0.227
LMN (ll,RBF)	0.0111	0.00046	0.264
LMN (local learning, axis-orth)	0.0110	0.00040	0.22

Table 6.1: Summary of mill modelling results on validation run. gl – global learning, ll – local learning, ellipse – full ellipsoidal distance metric, axis-orth – axis-orthogonal ellipsoidal distance metric.

6.3.4 Interpreting the trained models

Can the final machine learned model be interpreted to give the human engineer a better understanding of the system in question? This is a question which has often been ignored in the literature on learning systems, but one which is very important in industrial situations. There will usually be a trade-off between flexibility and interpretability, which will depend on their relative importance for a given application.

The positions of the basis functions can be visualised in a three dimensional plot, as shown in Figure 6.19. This gives us some insight into the location of complexity in the problem's state space. The hyper-ellipsoids in the figure correspond to the scales of the distance metrics of the basis functions. The constructive algorithm seemed to develop models with the complexity in the 'intuitively correct' areas of the input space. The acceleration and deceleration phases are deemed the most complex, and these were covered to a greater extent than the constant velocity area, which in general was covered by only one or two models. The resulting models were also reasonably small, the overfitting protection limiting the detrimental effects of sparse data in the complex areas of the input space – the model structures found tended to have less than 20 local models, despite training sets of over 6000 examples.



(a) BF Contour plot

(b) 3-D plot of Basis functions for roll mill model. The basis functions are shown here in the unnormalised form, basically as ellipsoids representing the volume equivalent to that of the contour plot at 0.5.

Figure 6.19: Visualisation of the mill model operating regimes.

A further aid to understanding the model is to go through the validation runs examining the model output and actual output, while viewing the current position $\tilde{\phi}$ in the operating space.

The top plot in Figure 6.20 shows the model and actual output, while the plot below shows the run for the entire training set in the operating space, with the portion being examined above is marked as the darker area.

The use of such visualisation tools lets the developer examine the areas of the operating space responsible for large model deviations, to try and determine possible inadequacies in the model structure (e.g. too few local models in a particular region, ill-suited local model structures, etc.) with the aid of visualisation tools such as shown in Figure 6.19. These tools are obviously limited to any three given dimensions, but can still provide useful insight in many cases. Further development of such tools would allow the user to select certain areas of the inputs space and find the nearest local models, so that their structures and parameters can be investigated, or gain more insight about the model workings. In general, as noted in (Johansen and Foss, 1994a), there should always be room for the human user to intervene in the modelling process. To give a more detailed feeling for the accuracy of the model, three

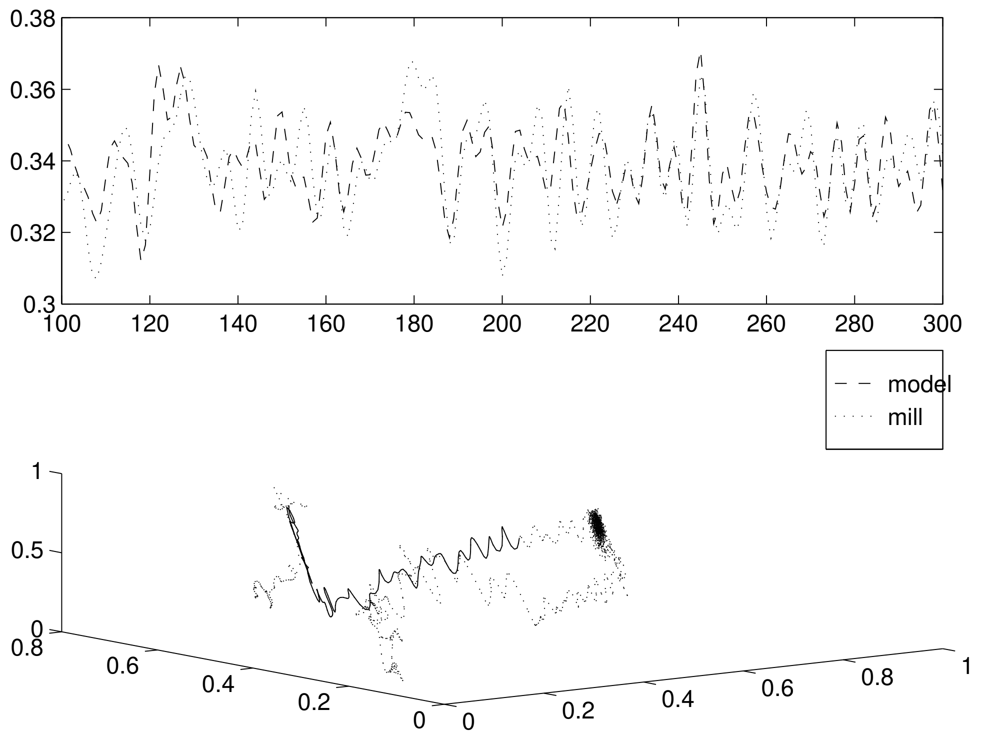


Figure 6.20: Model and Real Mill output, with the related area of the input space. The visualisation tool allows you to examine the model residuals, while viewing the current position in the operating space below (the subset of the data corresponding to the model plot above is highlighted). More powerful tools would allow the use such plots in conjunction with images such as Figure 6.19, where the local model closest to the data shown could be 'clicked' on and the structure and parameters viewed.

areas of the test data set are plotted in Figures 6.21 to 6.23.

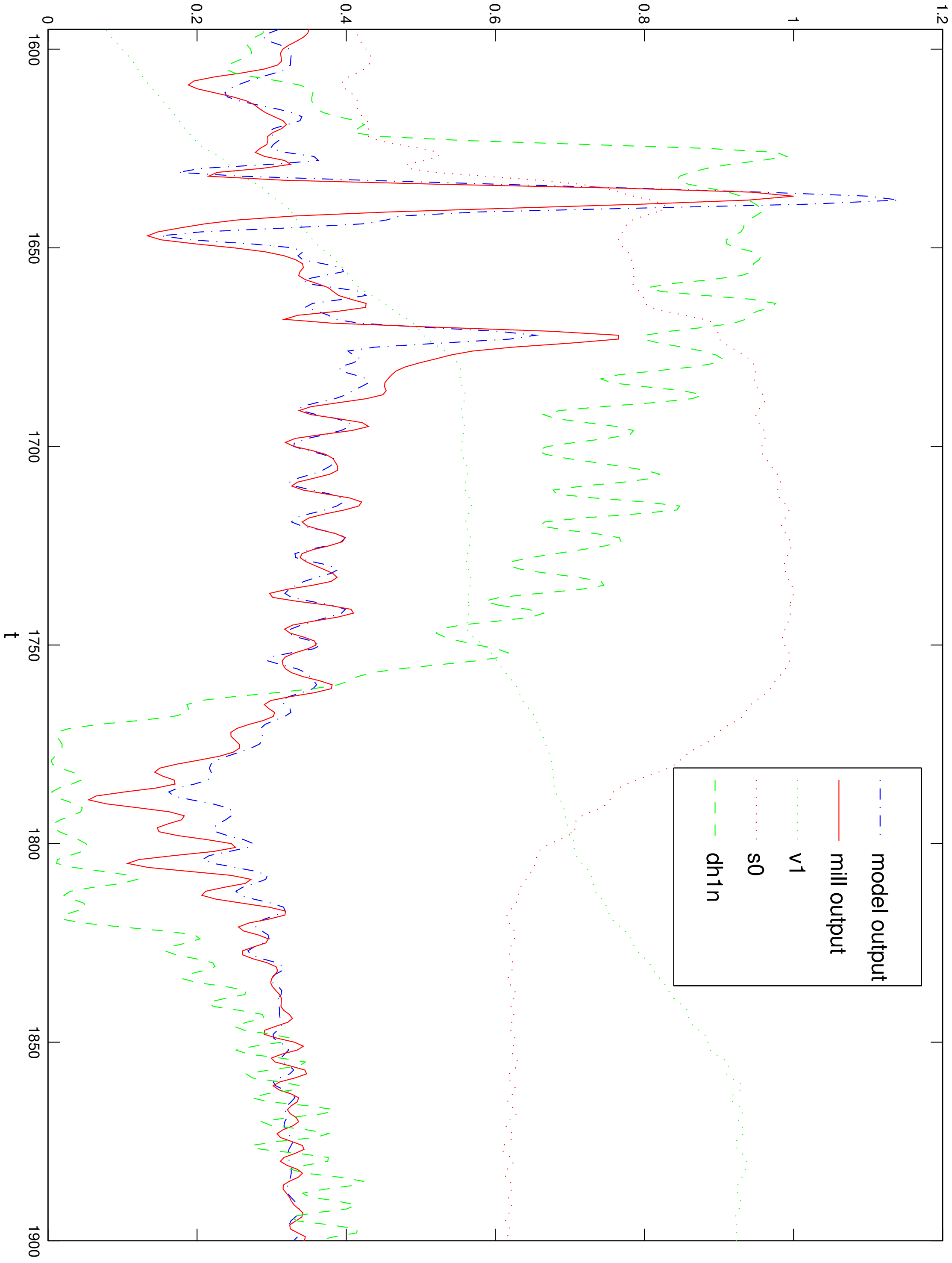


Figure 6.21: Detailed validation modelling runs with LMN and local learning. LMN model in acceleration phase. Despite the changes in the inputs, the model performs well, although it goes off track between 1760 and 1820.

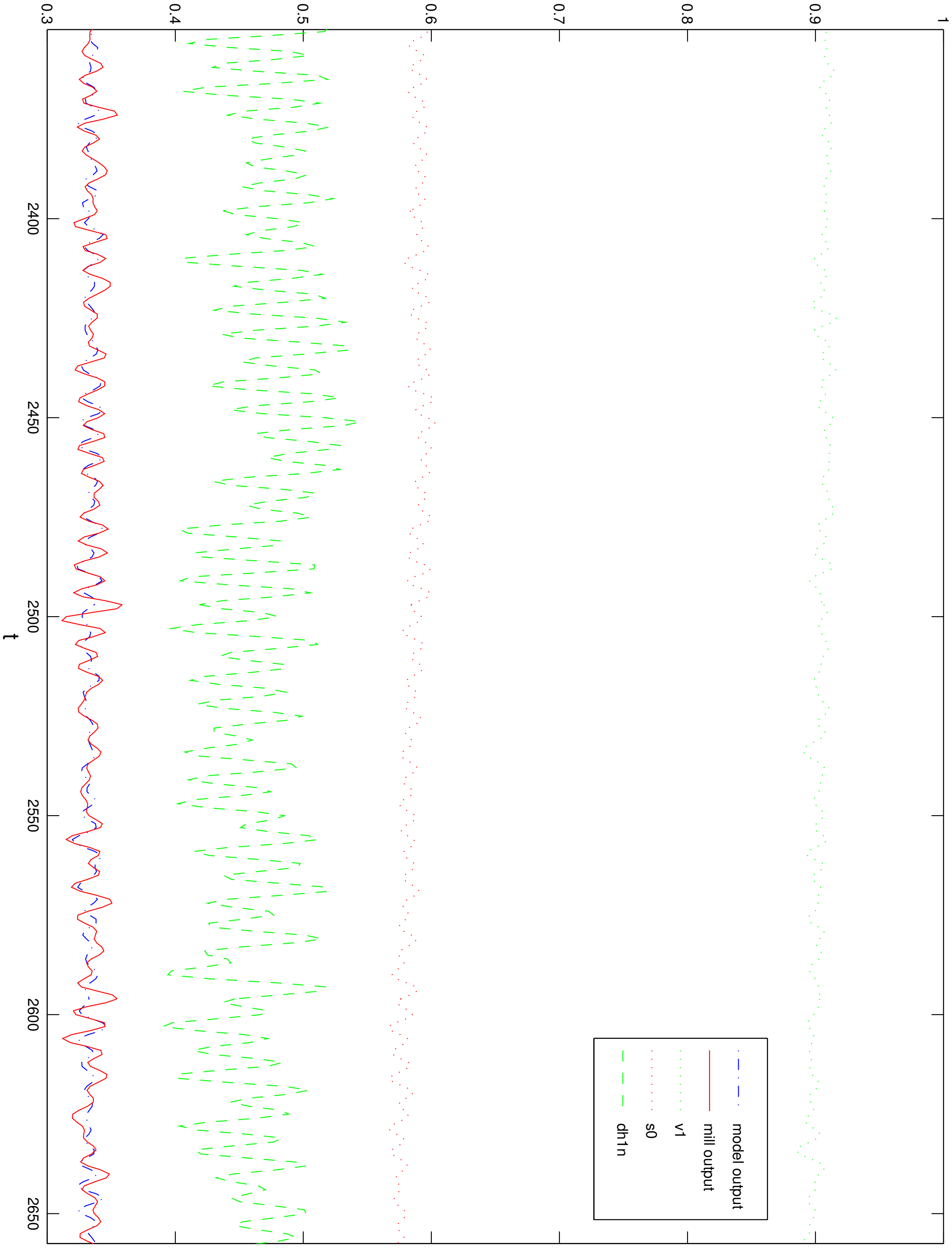


Figure 6.22: Detailed validation runs with LMN and local learning, in constant velocity phase.

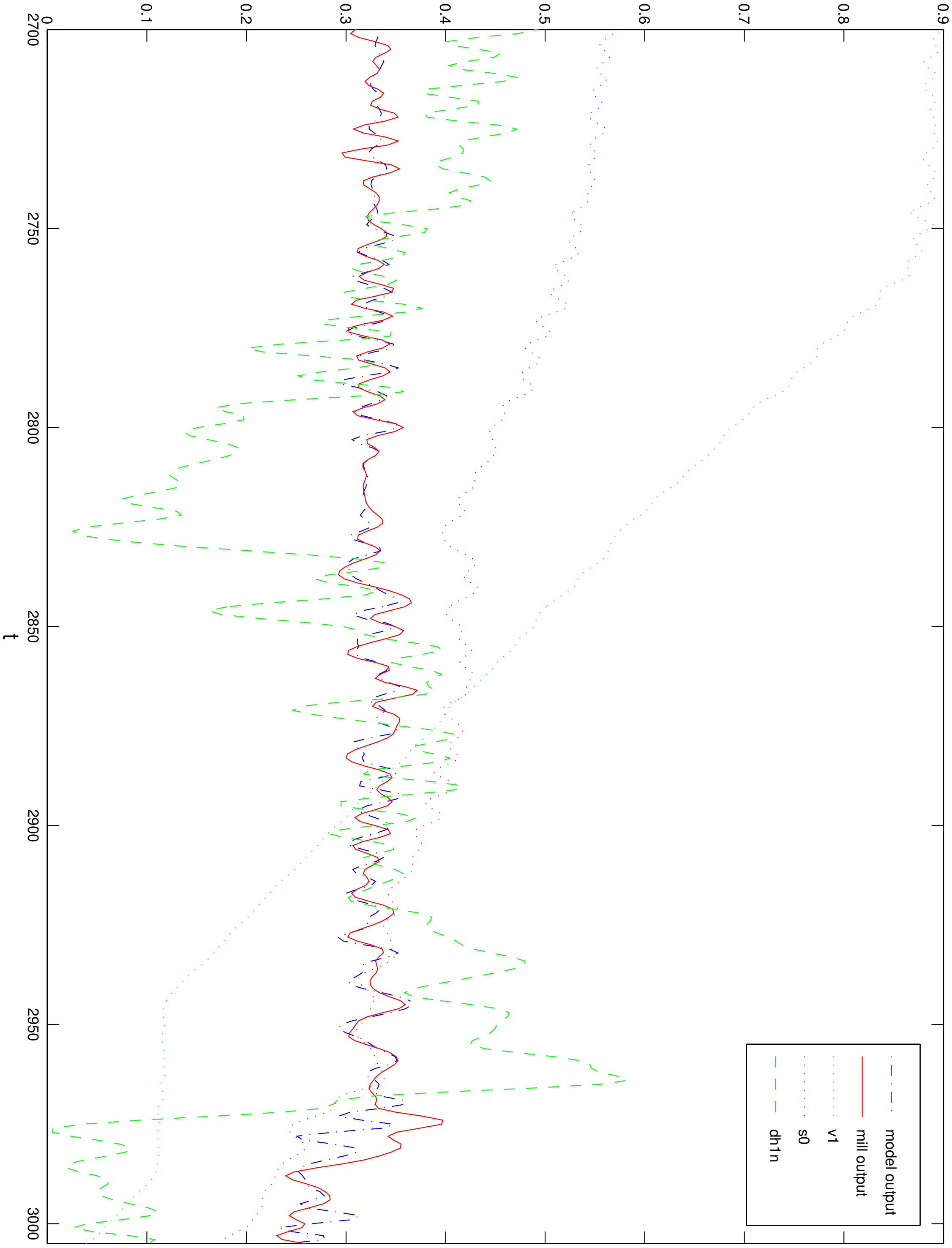


Figure 6.23: Detailed validation modelling runs with LMN and local learning. LMN model in deceleration phase. Again model does well despite the changing state of the physical system.

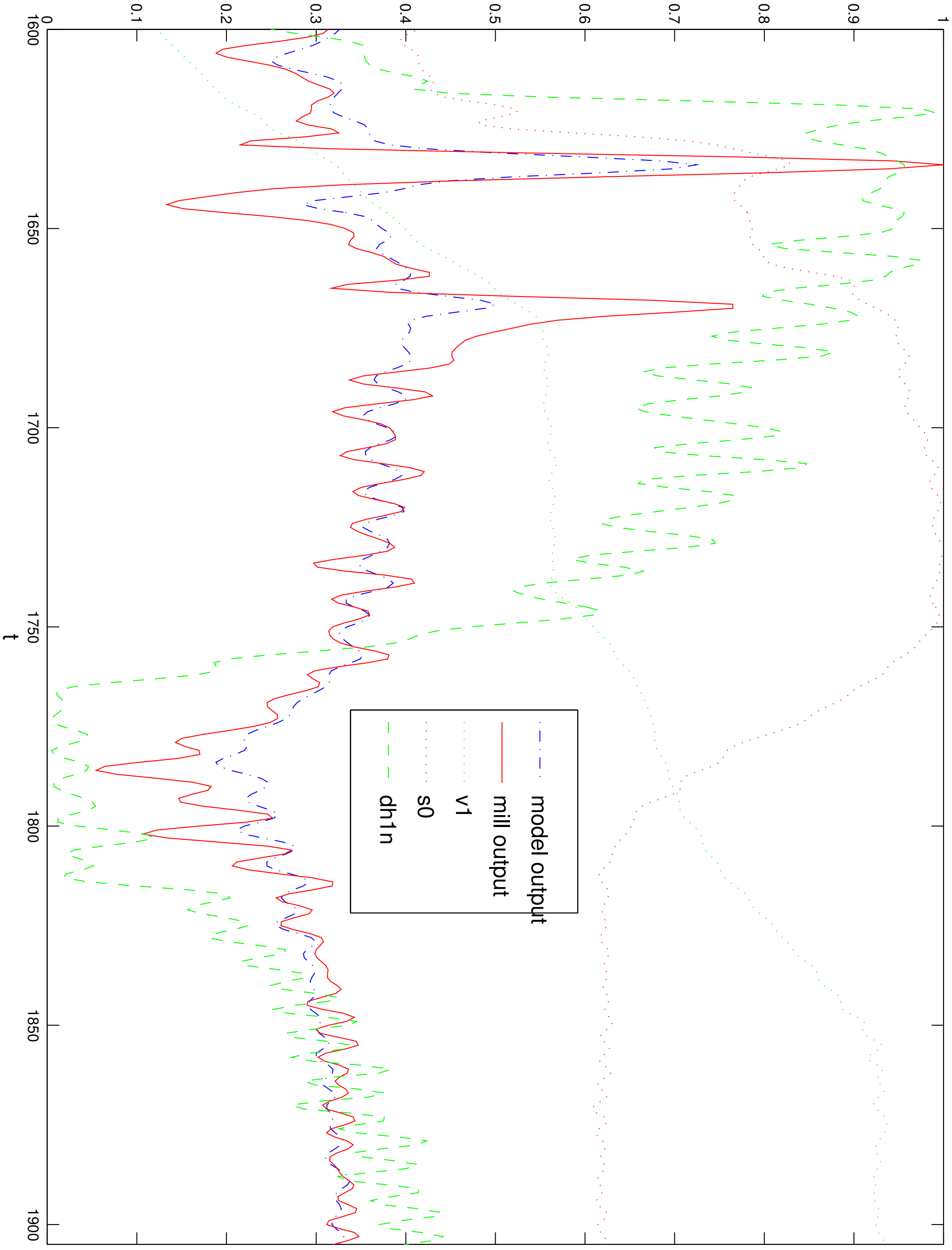
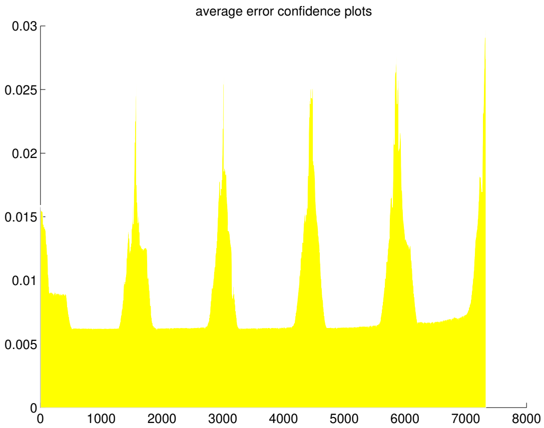


Figure 6.24: Detailed validation modelling runs with linear model. Compare with the LMN Model in Figure 6.21. The linear model does not manage to come as close to the actual output in the area of large output changes between 1600 and 1700. Nevertheless, the linear model could still prove to be useful in a real application because of its simplicity.

6.3.5 Analysing the local confidence limits

By examining the local error statistics (as defined in Section 3.2.1) for the local models the location of the worst errors can be deduced, possibly helping the engineer learn where the complexity in the real process is, or where insufficient experimental data exists. The test run is plotted below, where the error is shown, with the network's estimation of its own accuracy.

(a) Error statistics and LMN's weighted average error prediction on the training set used to derive the estimates.

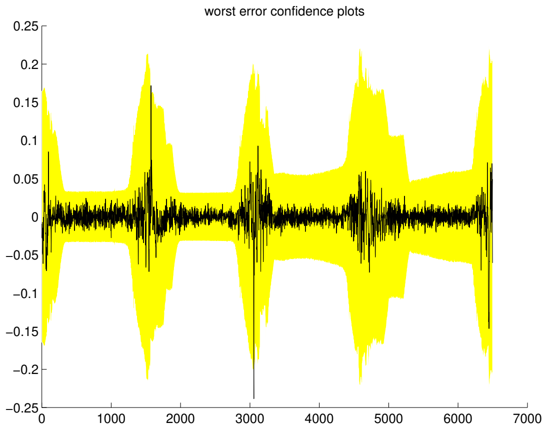


(b) Error statistics and LMN's weighted average error prediction on the validation set.

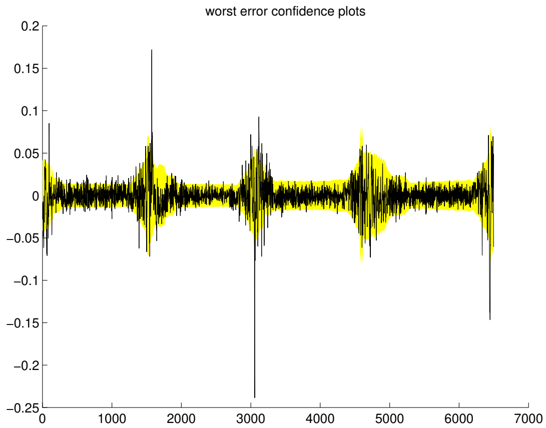
Figure 6.25: State-dependent average error statistics on the training and validation data

What happens when we close the loop?

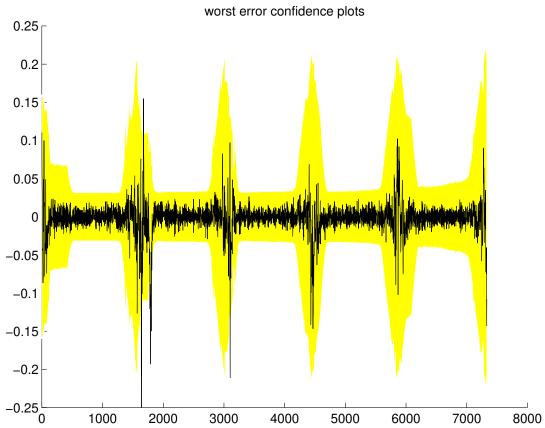
Despite the validation process, the real test of the model will only happen when it is used as it was intended in the real system. In this case, any false assumptions about the system, or disturbances on the training data will become immediately apparent. Once a controller is built which uses the model from the learning phase, it may also make the old model invalid. This is because the new, improved controller will impose a different type of stimulation on the process, which will cover different areas of the input space, and which may produce quite different responses from the process.



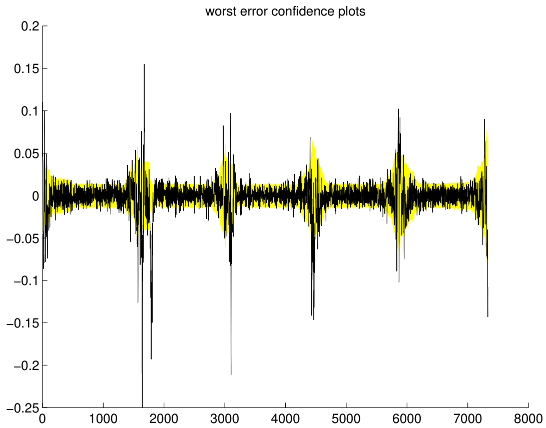
(a) Model residuals and LMN's local worst error prediction on the training set used to derive the estimates. The worst error from the local model's receptive field is taken (where basis function > 0.1), so the algorithm tends to overestimate the worst error.



(b) Model residuals and LMN's weighted worst error prediction on the training set used to derive the estimates. Because the worst error is weighted the algorithm tends to underestimate the maximum error



(c) Model residuals and LMN's local worst error prediction on the validation set.



(d) Model residuals and LMN's weighted worst error prediction on the validation set.

Figure 6.26: State-dependent worst error statistics on the training and validation data.

6.4 Robot Actuator Modelling

The robot application described in this section is based on data sampled from a physical system, supplied by Tom Kavli, SINTEF, Oslo. The application and the data sets have been described for modelling work in (Kavli, 1992) and in (Johansen and Foss, 1994b). A brief overview is given here:

Many industrial robot applications now demand high dynamic accuracy. Model based control schemes have the potential to improve performance, but the use of hydraulic manipulators has suffered due to the lack of good nonlinear models for the hydraulic components. Model based control schemes have been more successful on electric direct drive arms with low friction and linear actuators. The goal of the learning task is to form a model of the servo valve/actuator system of a hydraulic robot. The robot is an ABB Trallfa TR4000 Robot, specially designed for spray painting, where tracking accuracy over a desired trajectory is extremely important. The control signal u is described as a function of the joint position q, velocity $\dot{q}$ and acceleration $\ddot{q}$:

$$u = f\left(q, \dot{q}, \ddot{q}\right). \tag{6.7}$$

The nonlinearities are due to:

- the changing moment arm of the cylinder over the operating range,
- the nonlinear damping coefficient due to the quadratic flow/pressure relation for turbulent flow and,
- the changing pressure gain characteristics for the servo valve at different flow rates.

The data was sampled by logging the data at 100Hz, while the manipulator moved along a randomly generated path. The velocity and acceleration signals were calculated by low pass filtering the data and differentiating the joint positions. The linear effects in the system were subtracted from the data to emphasise the nonlinearity of the system. The training data consisted of 8000 training points and the test set had 1000 points.

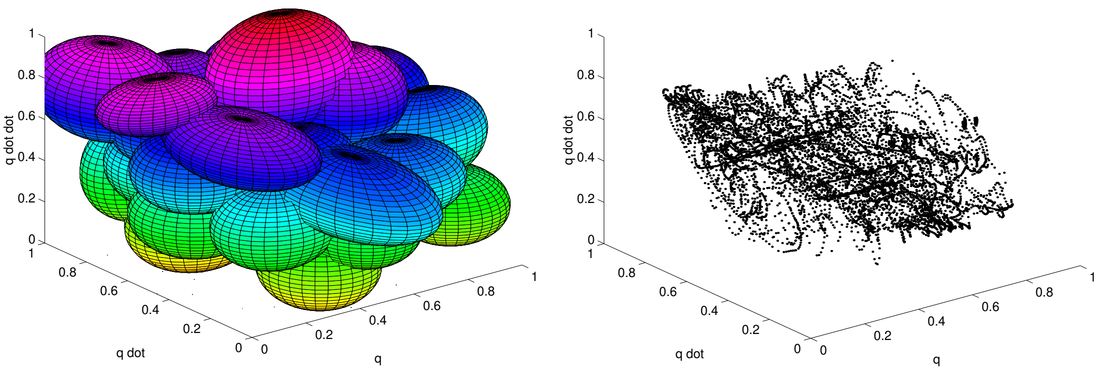
6.4.1 Experimental results

The ASMOD results * are taken from (Kavli, 1992). LSA (Local Search Algorithm) results + are from (Johansen and Foss, 1994b). The LMN results were obtained by using the algorithms described in this thesis, as well as the active selection algorithm to reduce the number of points in the training set (the training set was reduced to a maximum of 1200 points).

The MARS results are shown in Figure 6.30(a). The cross-validation results for the local model net with local learning are shown in Figure 6.29(a) is compared with the global learning case in Figure 6.29(b). Note the poor performance of local learning in the cross-validation results.

Model Type	Test Error (NRMS)
LSA+	17
MLP (3-20-1)	23
ASMOD*(Quadrat.)	15
ASMOD*(Linear)	17
RBF*	23
MARS	17
LMN (local learning)	18
LMN (global learning)	17
LHM (axis oblique)	18
LMN (local learning)	18
LMN (global learning)	17

Table 6.2: Robot modelling results on the test set. The only method to provide better results than the LMN was the ASMOD model with quadratic splines (and a large number of parameters at 512 basis functions). Local Modelling proved to lead to slightly worse results, and required a larger number of parameters, but produced 'safer' extrapolation to areas with little training data.



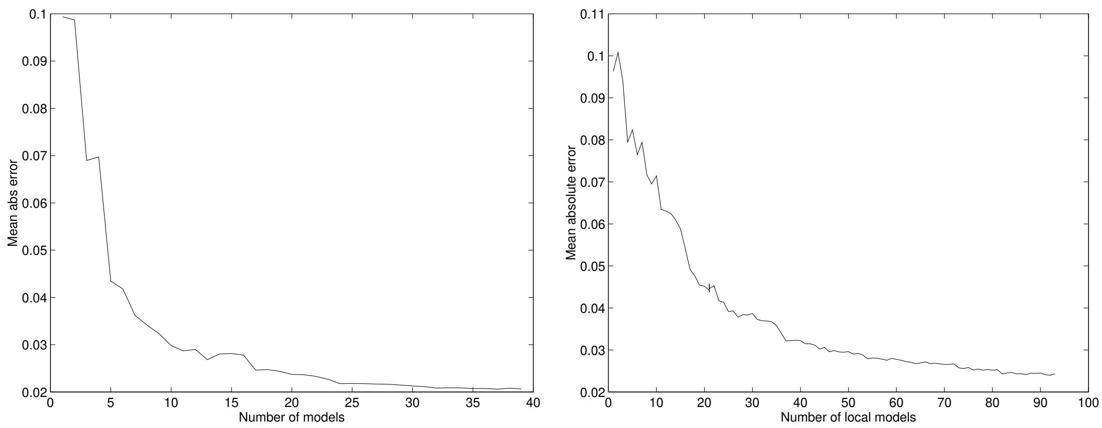
(a) LMN Basis Functions for the robot model

(b) Robot training data

Figure 6.27: Distribution of robot training data and local models' basis functions.

Only one of the runs produces a mean error similar to that of the global optimisation. This is, however, due to a weakness in the structure identification algorithm, which stops growing when the error stops decreasing. In global learning, the greater degree of freedom for the optimisation means that the additional local models have a more pronounced effect, and construction continues. By altering n_{res} to 5 for local learning produced the results shown in Figure 6.28(b), where a larger number of local models were built, but which achieved an improvement in accuracy to that close to the globally trained model.

The problems with ill-conditioning for global learning are less prevalent in this application, due to the low dimensionality, accompanied by large amounts of training data, and the apparent smoothness of the underlying process nonlinearity.



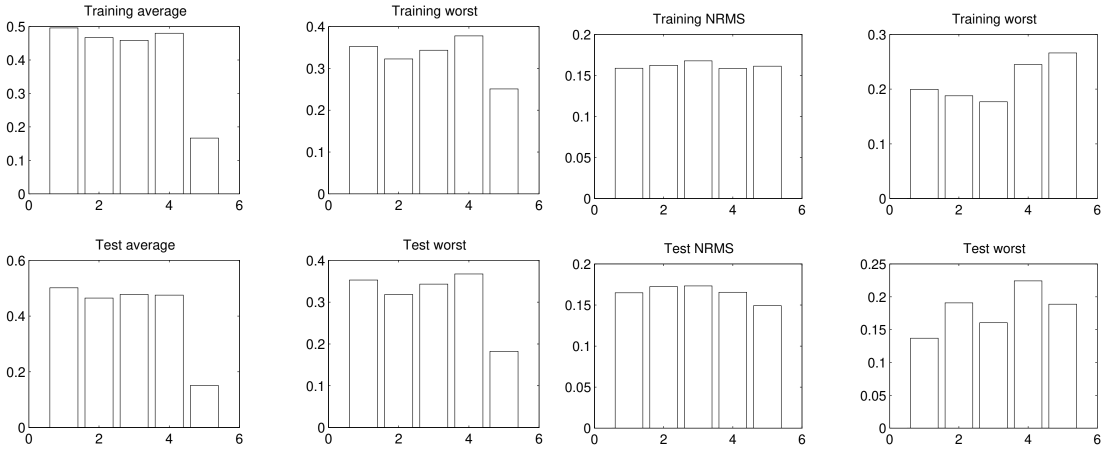
(a) LMN error curve with global learning

(b) LMN error curve with local learning

Figure 6.28: The progress of the average absolute error as new models are added to the robot modelling example. Note that local learning finishes earlier than global learning.

Visualisation of the system

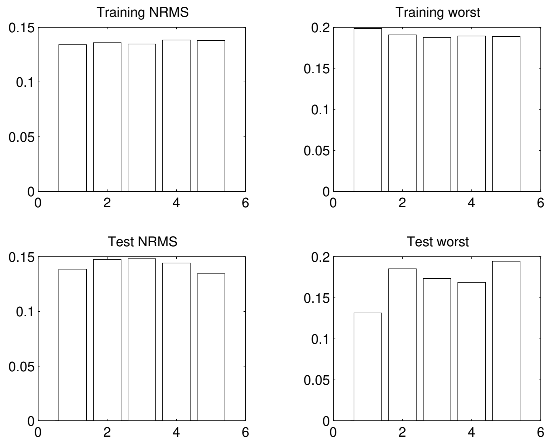
The following surfaces are representations of the input space as seen through different slices through the space. The corners of the plots are areas where the system had no training data, and are therefore unreliable.



(a) LMN results with local learning

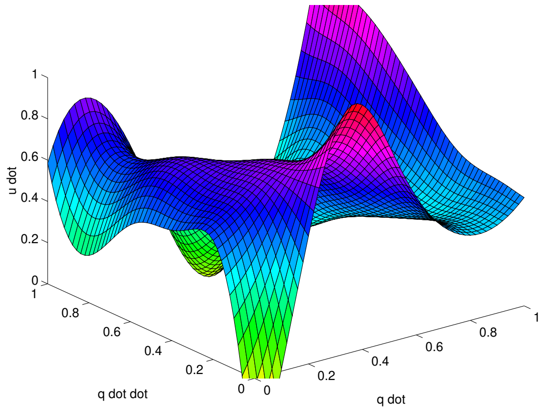
(b) LMN results with global learning

Figure 6.29: Cross-validation results for the Trallfa robot with Local Model Nets. Note the poor performance of local learning in all but one example. By increasing n_{res} to 5, the local learning could perform better, as shown in Figure 6.28(b)

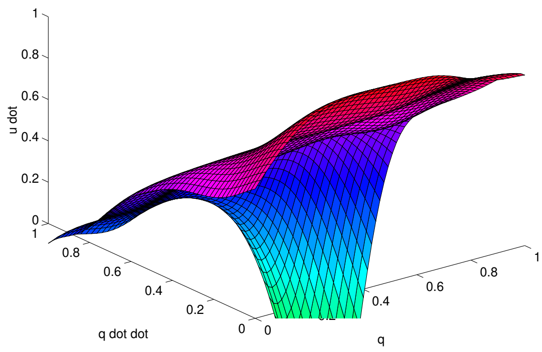


(a) MARS results with maximum number of basis functions set to 200

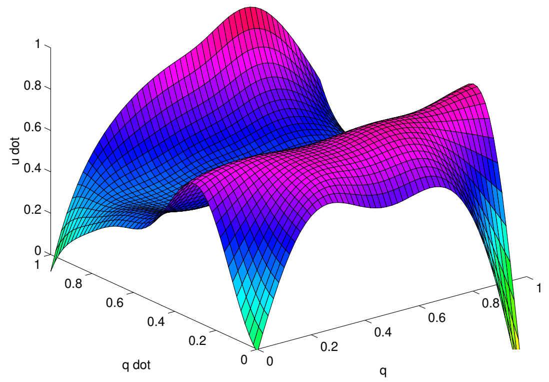
Figure 6.30: Modelling results for the Trallfa robot with MARS.



(a) Speed and acceleration, with position at 0.5.

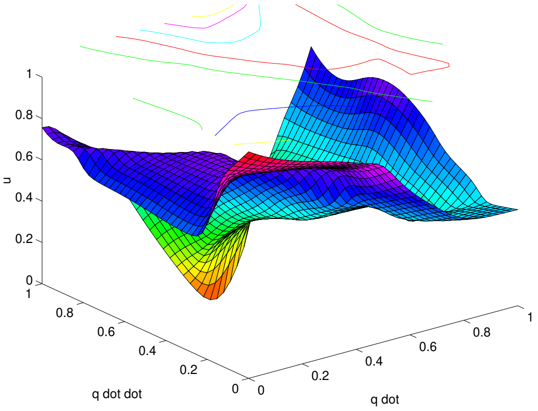


(b) Position and acceleration with speed at 0.5.

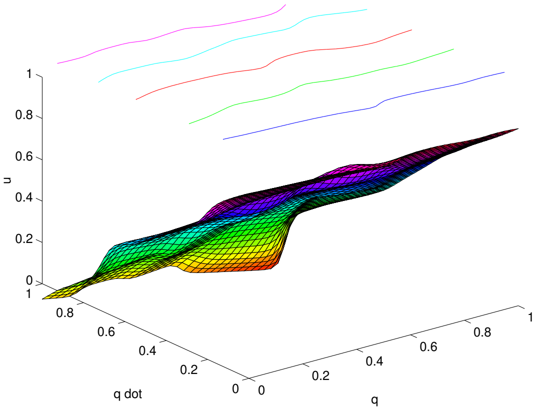


(c) Position and speed with acceleration at 0.5.

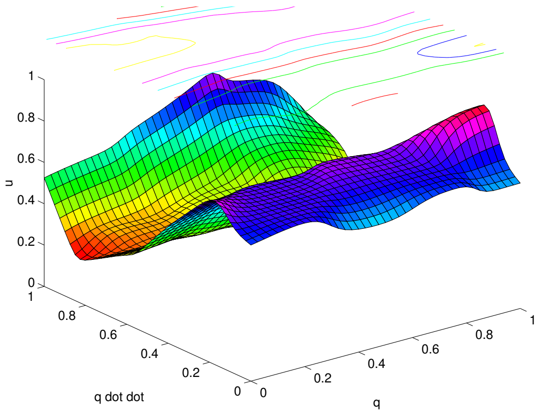
Figure 6.31: Output responses for slices through the robot actuator model. Note the way the model response surface flies off in the corners where data does not exist. This is one disadvantage of normalised basis functions, where the basis functions are supported through the whole input space, combined with global learning, which produces less robust models. Compare the plots here to those achieved with local learning in Figure 6.32.



(a) Speed and acceleration, with position at 0.5.



(b) Position and acceleration with speed at 0.5.



(c) Position and speed with acceleration at 0.5.

Figure 6.32: Output responses for slices through the robot actuator model using local learning. Compare the plots here to those achieved with local learning in Figure 6.31. The locally trained models provide a less dramatic form of extrapolation outside the populated area of the input space.

6.5 Conclusions

6.5.1 Rolling mill results

The rolling mill application provided an interesting environment for testing the local model network methodology and the constructive algorithms proposed in this thesis. To solve the problem meant dealing with the practical problems of data acquisition, pre-processing, and variable selection. In some ways though it is also not an ideal choice to show off the structure, due to the poor available understanding of the system, and the difficulty in performing experiments to acquire data, or to validate the usefulness of models produced. Despite the relative deficit of formalised *a priori* knowledge about the system, the local model network methods proved to be the most accurate, and most interpretable representations of the rolling mill process. The MARS algorithm was relatively fast, but did not have the desired accuracy, and the conventional MLP neural network was very slow during training, and produced low-accuracy models.

The validation phase used cross-validation techniques to test the robustness of the learning algorithms, as well as using validation sets from completely different runs. This showed up some interesting features in the experimental setup. The local learning techniques led to significantly more accurate models for new validation data than globally trained ones, even though they were found to be slightly less accurate on 5-fold cross-validation runs based on partitions of the training set. This seems to indicate that the local learning produced a more robust model of the mill, at the expense of accuracy on the training data.

The state-dependent error estimates proved, despite their simplicity, that they could capture the areas of the input space where the model is least accurate – the acceleration and deceleration phases. Other tools were used which helped interpret the model, including the visualisation of model residuals, accompanied by the position of the data in the operating space of the local model net's basis functions. This can also be combined with slices through the basis functions, in the form of either contour plots or ellipsoidal plots of basis functions in three dimensions.

6.5.2 Robot actuator results

The results obtained by the Local Model Net are comparable with the results quoted in the original work (Kavli, 1992) and also with those in (Johansen and Foss, 1994b). The problem is low dimensional, the nonlinearity seems to be fairly smooth, and there is an abundance of training data. This means that other model structures can also cope with the problem, and that while the local model nets perform well at the modelling task, the potential advantages of local model nets are not as evident as in other applications. A further interesting result from the experiment pointed out weaknesses in the structure identification algorithm working with local learning, where it often stops construction too early. Using a larger n_{res} lead to better results, but also needed a larger number of parameters.