In: Proceedings IFAC Symposium on Artificial Intelligence in Real-Time Control 1997
AIRTC'97 (Kuala Lumpur, Malaysia), Herb E. Rauch (editor)

Performance Optimisation of Learning Feed Forward Control

Wubbe J.R. Velthuis, Theo J.A. de Vries, Job van Amerongen

EL–BSC–RT, University of Twente, P.O.Box 217, 7500 AE Enschede, The Netherlands
e-mail: icontrol@el.utwente.nl; web: http://www.rt.el.utwente.nl/icontrol/

Abstract

The performance of sub-optimal feedback controllers can be improved in several ways. In this paper a learning control strategy is considered. The learning control system consists of the feedback and a feed forward controller. The feed forward controller is implemented as a neural network that is trained during control in order to minimise the tracking error. The type of neural network is a single layer network, in which B-spline basis functions are used to store the input-output mapping. The distribution of the B-splines on the domain of the input(s) is of influence on the performance of the learning controller. Until recently, the basis functions were distributed by rule of thumb. In this paper fuzzy clustering techniques are used to obtain the distribution in a systematic way. In experiments the learning controller has been used to control a linear motor. Also when the B-splines are chosen by rule of thumb, the learning controller was able to improve the performance of the feedback controller considerably. The tracking error could be reduced further by determining the distribution of the basis functions using fuzzy clustering.

Keywords: intelligent control, neural control, adaptation, B-spline networks, fuzzy clustering

1 Introduction

Feedback control is often applied to improve the dynamical behaviour of electromechanical systems. Feedback controllers are usually designed on the basis of a process model. When only an approximate process model is available, a sub-optimal performance may result. Several methods can be used to improve the performance of feedback controllers. *Commonly*, a more detailed process model is derived. However, this method has two disadvantages.

- For complex processes, obtaining a quantitative model that is suited for the controller design might be difficult and time consuming. In some cases, e.g. in case of process uncertainties, proper identification is even impossible.
- The feedback controller has to provide for both a high performance and robust stability. This usually results in a trade off between the two properties. A feedback controller that has a high performance often does not feature a robust stability. Small variations in process parameters may destabilise the system.

The latter problem can be overcome by creating separate means for obtaining high performance and robust stability. In this paper this is done by applying a feed forward controller in addition to the feedback controller. The feed forward controller is intended to generate a steering signal that makes the output of the process track the reference path, and thus obtain a high tracking performance. Therefore the feed forward controller has to compensate for the process dynamics and reproducible disturbances. Random disturbances such as noise are compensated for by the feedback controller. The feedback controller can be designed for robust stability only because the performance is obtained by the feed forward controller.

In order to be able to design an accurate feed forward controller, the process dynamics need to be known. As stated before, obtaining a detailed quantitative model requires a lot of effort and time and sometimes might even be impossible. Therefore, in this paper, the feed forward controller is not designed on the basis of a process model, but learns to control the process during operation. This control concept is known as Learning Feed Forward Control (LFFC) [1][2][3].

The feed forward controller is a neural network (NN) which is implemented as a B-spline network (BSN) [4]. In a BSN the input-output mapping is stored using basis functions (B-splines). Until recently, the distribution of the B-splines over the domain of the inputs was done by rule of thumb. Manual tuning was needed in order to optimise the tracking performance. In this paper fuzzy clustering techniques are used to obtain the B-spline distribution in a systematic and automated way.

In section 2 the BSN and LFFC will be dealt with in more detail. The optimisation technique that is used to determine the B-splines distribution is presented in section 3. In section 4 experiments will be shown. Finally some conclusions are drawn in section 5.

2 Learning Feed Forward Control

For a large class of processes an LFFC (figure 1) is able to track both repetitive and random paths accurately.

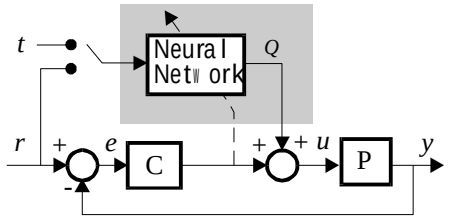

Figure 1: Learning Feed Forward Control

In case of repetitive paths the periodic *time* can be used as input of the feed forward controller. In case random paths have to be tracked the *reference path* has to be used. For sake of simplicity the application considered in this paper is to follow a repetitive path.

During operation the input-output mapping of the NN is adapted in such way that the tracking error decreases. The adaptation is done on the basis of the output of the feedback controller. This signal is assumed to minimize the tracking error. Therefore applying the feedback signal in feed forward should result in a smaller tracking error. The adaptation of the feed forward signal is given by:

$$u_Q^{n+1}(t) = u_Q^n(t) + \gamma u_C^n(t) \qquad (1)$$

In which n is the number of times the path has been tracked and γ is the learning rate which indicates how strongly the feed forward signal should be changed (typically $0 < \gamma \leq 1$).

The proposed control system has to operate in a real-time environment. Therefore the NN should have the following properties:

- It has to be computationally cheap. Within one sampling period both the output of the neural network has to be calculated and the weights have to be adapted. In mechatronic applications high sampling frequencies are needed that are not attainable in case of computationally expensive NN's.
- Since learning takes place on-line, the NN should be able to learn fast. Slow learning results in a long training period in which the performance of the controller is sub-optimal. Furthermore slow learning could prevent the learning controller from adapting fast enough to varying process parameters.

An example of a NN that is both computationally cheap and is able to learn fast is a B-spline network.

In a BSN the i/o mapping is stored using basis functions, also known as B-splines. B-splines are piecewise polynomials, also known as B-splines. A B-spline of order n consists of piecewise polynomials of order n-1 (figure 2a). The B-splines are distributed on the input domain of the controller in such way that at each input the sum of the heights of the B-splines is equal to 1 (figure 2b).

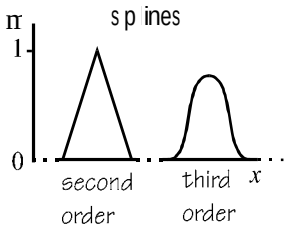

Figure 2a: B-splines

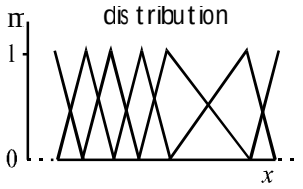

Figure 2b: Distribution

The output of a BSN (figure 3) is a weighted sum of the B-spline evaluations.

$$y(x) = \sum_{i=0}^{N} w_i \mu_i(x) \qquad (2)$$

In which $y(x)$ is the output of the BSN, $\mu_i(x)$ the height of the i-th basis function at input x and w_i the weight of the basis function.

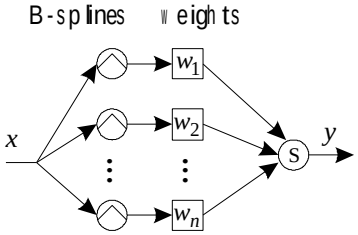

Figure 3: B-spline network

Training the network, or in other words changing the i/o mapping such that it becomes equal to the desired i/o mapping, is done by adapting the weights. The weights are adapted according to the following learning rule:

$$\Delta w_i = \gamma \mu_i(x)\, e(x) \qquad (3)$$

In which, γ is the learning rate ($0 < \gamma \leq 1$) and $e(x)$ is the mapping error of the network at input x.

In this research we use 2nd order BSN's. The basisfunctions consist of piecewise 1st order polynomials. Because the sum of the heights of the B-splines has to be equal to 1, the positions of the B-splines will be such that the position of the top of one B-spline coincides with the begin and end of 2 other B-splines (figure 4).

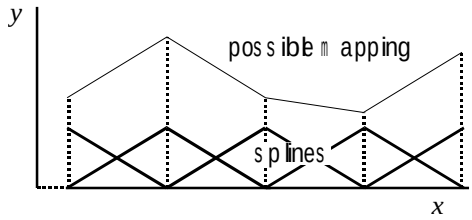

Figure 4: I/O Mapping

The positions of the tops of the B-splines are called knots. It is easily seen that the output of a 2nd order BSN is a linear interpolation of the mapping valueas at the knots.

In LFFC the BSN is used as a function approximation. It can be easily seen that the accuracy of the approximation depends on the number of B-splines that are used. A larger number of B-splines will result in a smaller approximation error. A simple example is given in figure 5, where a sinusoidal signal is approximated using both few as well as many B-splines.

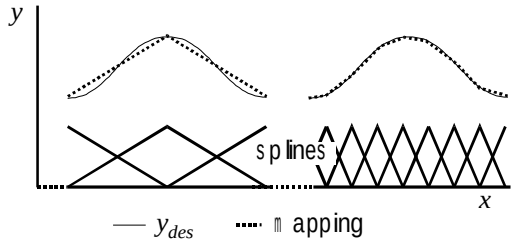

Figure 5: Few and many B-splines

Not only the number of B-splines but also their position is of influence on the accuracy of the approximation. This can be seen in the following example, where again as sinusoidal signal is approximated using 2 B-spline distributions. In this example the width of the B-splines in both distributions is identical but the B-splines are shifted over a distance equal to a quarter of their width.

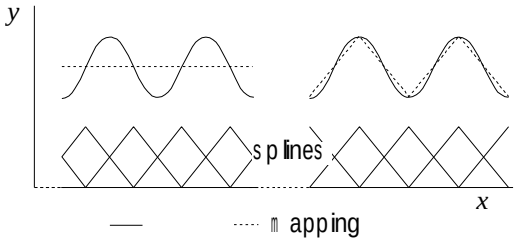

Figure 6: Badly and well placed knots

In this situation the accuracy of the approximation is highly influenced by the position of the B-splines.

From these 2 examples, it can be concluded that when a BSN is used for accurate approximation, the distribution of the B-splines should be designed on the basis of the signal that is to be approximated.

3 B-spline Distribution Optimisation

The BSN is to learn the steering signal that would result in perfect tracking. To obtain the smallest possible tracking error, the B-spline distribution should be chosen on the basis of the optimal steering signal. Due to the fact that we only have an approximate process model, this signal cannot be calculated in advance. However, using an iterative procedure the best approximation of the optimal steering signal and the corresponding B-spline distribution can be obtained.

Firstly, an initial B-spline distribution is chosen. This B-spline distribution bears no relation with the mapping to be learned. As a consequence the feed forward signal that is obtained after learning is a first approximation of the optimal feed forward signal and not the optimal feed forward signal itself. The approximation can be improved by improving the B-spline distribution, which requires a more accurate estimation. This might seem a vicious circle. However, the output of the feedback controller indicates how the feed forward signal can be improved. We are able to give a more accurate estimation of the optimal feed forward signal by adding the feed back signal to the feed forward signal that is obtained after learning. The more accurate approximation is denoted by u_Q'

$$u_Q' = u_Q + u_C \qquad (4)$$

The B-spline distribution corresponding to the optimal approximation will be determined using this signal in the following iterative way:

1. Choose an initial spline distribution.
2. Train the LFFC until convergence.
3. When the RMS tracking error over one trail has not decreased, go to step 6.
4. Determine the best spline distribution for u_Q'.
5. Go to step 2.
6. End.

In step 4, where the new spline distribution is determined, both the number of B-splines and their position can be adapted.

Number of B-splines: Because the accuracy of the approximation increases if a larger number of splines is used, one would intuitively choose a distribution that has a large number of splines. However, the following difficulties may arise. *Firstly*, a large amount of computer memory is needed to store the feed forward signal. *Secondly*, the rate of convergence of the LFFC decreases as the width of the B-splines becomes smaller. Especially in case of processes with varying process parameters fast learning is required, which might not be able using too many B-splines.

Finally, and most importantly, research on the stability of LFFC [2] showed that a large number of B-splines may result in instability of the learning mechanism. Instability occurs if the transfer function from the output of the NN to the learning signal has a positive real part. In this situation a positive/negative adaptation of the weights, and therefore also of the output, in run n induces an even larger positive/negative adaptation in run n+1. Typically this is the case at high frequencies.

To avoid instability a BSN should be used that is able to learn the low frequencies, for which the learning mechanism is stable and not the high unstable frequencies. The frequency of the signals that can be accurately approximated is determined by the width of the B-splines. The smaller the width the higher the frequencies the BSN is able to learn. Choosing a large amount of B-splines enables the BSN to learn the high unstable frequencies and instability of the learning mechanism results. The minimum width of B-splines, and therefore also the maximum number of B-splines, can be determined on the basis of an approximate process model [5].

It can be concluded that in order to obtain stable learning there is a maximum to the number of B-splines that can be used. Because in this situation the position of the B-splines has a large influence on the accuracy of the approximation (figure 5), the position of the B-splines should be determined carefully.

Position of the B-Splines: The position of the splines is obtained from u_Q', using a fuzzy clustering (FC) algorithm [6][7]. FC is a generic term for algorithms that partition data in fuzzy sets, also known as clusters, in such way that similar data points, are gathered in the same cluster. From these clusters local linear approximations of the data can be derived. Subsequently these local linear approximations can be transformed in a BSN. The procedure of obtaining B-splines will now be explained thoroughly on the basis of a simple example.

Step 1: Partition the data in fuzzy sets

The FC-algorithm that is used to obtain the clusters is the Gustafson-Kessel algorithm [8]. This algorithm is chosen because it can detect clusters of different shapes and orientations. In figure 7 the GK-algorithm has been applied to a number of data points. The clusters that have been obtained are hyperellipsoids, which are represented by their level curves.

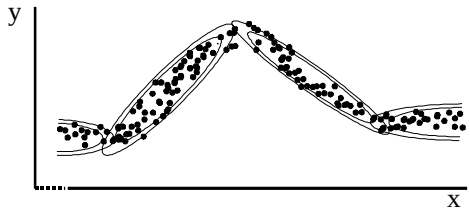

Figure 7:Data & Clusters

Step 2: Obtain local linear approximations

Now that the data has been partitioned, the clusters will be used to generate a fuzzy model. The type of fuzzy model is a Takagi-Sugeno (TS) model [8], which consists of rules of the following form.

$$\text{IF } x = A_i \text{ THEN } y = a_i x + b_i \qquad (5)$$

In which x is the input variable, y the output variable A_i is the fuzzy antecedent set and a_i, b_i are the parameters of the affine model. The output of the TS model consists of smoothed local linear approximations of the data.

The fuzzy antecedent sets A_i can be obtained by projecting the clusters on the input axis. The parameters a_i, b_i can be derived from the position and the orientation of the clusters.

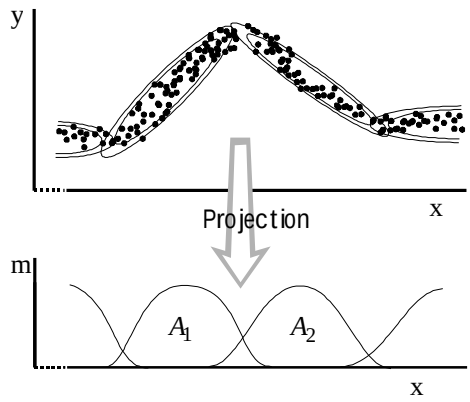

Figure 8: Data & Clusters

Step 3: Transform the TS model into a BSN

Similar to the BSN, the output of a TS-model is a local linear approximation of the data. However, in both methods the output is obtained in a different way. In case of a BSN the output is the result of a linear interpolation between two weighted splines. Switching between linear approximations occurs at the tops of the B-splines. In the TS-model the output is calculated using linear functions. In this case gradually switching between linear approximations occurs at the intersection of two fuzzy antecedent sets.

It can be seen that a TS-model can be transformed into a BSN in the following way. The knots should placed at the intersection of the fuzzy antecedent sets, while the

weights of the BSN should equal the output of the TS-model at the intersections (figure 9).

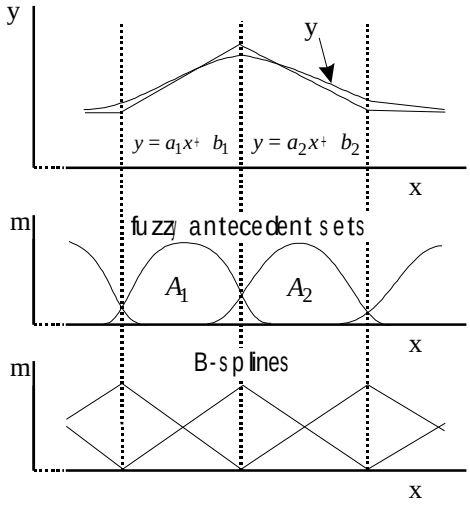

Figure 9: Determination of the B-splines

After FC has been applied, the distance between each pair of knots should be checked to see if it is not smaller than half the minimum B-spline width. This prevents instability in the learning mechanism.

4 Experiments

The presented theory is validated by applying LFFC, including FC, to a Linear Motion Motor System (LiMMS). The LiMMS is designed by Philips to perform linear motions for applications such as scanning and pick-and-place tasks. The motor configuration consists of a base plate on which permanent magnets are placed, and a translator, which contains the coils (figure 10). The thrust force is generated by applying a 3 phase current to the coils.

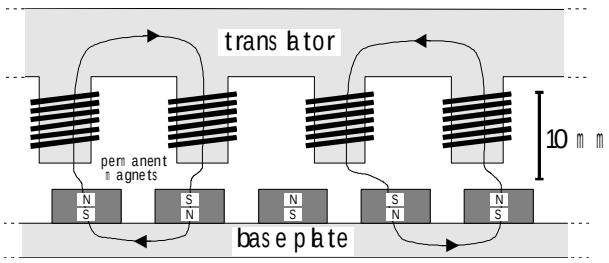

Figure 10: Linear Motor Motion System

In order to optimise the efficiency of the motor, iron cores have been placed in the coils. The iron cores have a magnetic interaction with the permanent magnets in the base plate. A force results which tries to move the translator in detent, stable positions. This phenomenon is generally known as cogging. Cogging occurs both during motion and in a zero-current situation. The influence of cogging on the translator could be modelled if the exact position and the magnetic properties of the permanent magnets would be known. However, to reduce manufacturing costs the following choices have been made:

- The permanent magnets have a not too tight magnetic tolerance.
- There is a not too tight tolerance on the position and orientation of the permanent magnets.

Other disturbances that the LiMMS suffers from are:

- A reluctant force, which is caused by the varying self-inductance of the coils during motion.
- Unknown friction in the ball bearings.

Due to these facts it is difficult to obtain an accurate model of the LiMMS which could be used for the design of a feedback controller. In order to meet the tracking accuracy that is needed in some applications, an LFFC is used for control.

In the following experiments, the LiMMS is to repeatedly track the reference path that is shown in figure 11.

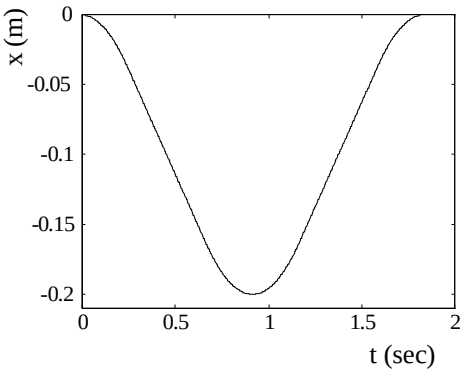

Figure 11: Reference Path

The feedback controller that is used to control the LiMMS is a PID controller of which the parameters have been obtained using auto-tuning techniques. The output of the feedback controller and the resulting tracking error are given in figure 12 and 13.

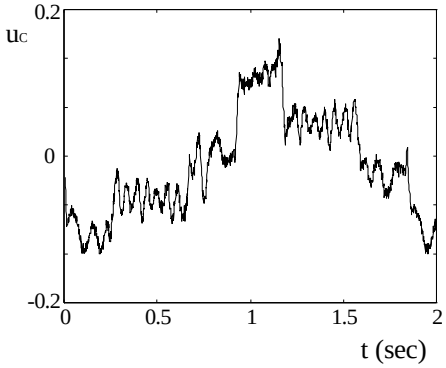

Figure 12: Output of the feedback controller

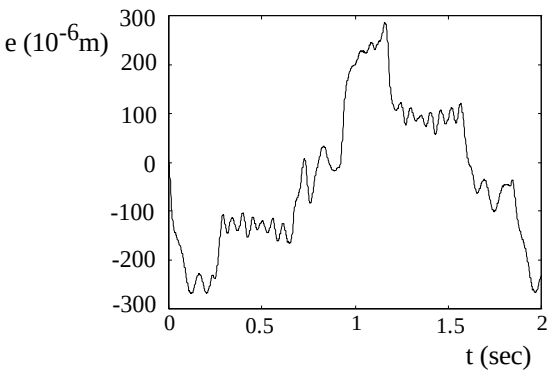

Figure 13: Tracking error of the feedback controller

Now that the feedback controller has been designed, an initial B-spline distribution for the LFFC should be chosen. The number of B-splines is determined on the basis of the output of the feedback controller. Using the fact that the BSN uses 1st order polynomials to approximate the signal, a total number of 75 B-splines is chosen by rule of thumb. The B-splines are distributed uniformly over the domain of input. The tracking error that remains after learning can be seen in figure 14. (Note the difference in scale compared to figure 11)

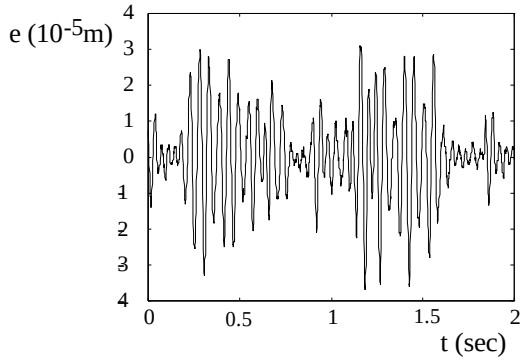

Figure 14: Tracking error using initial B-spline distribution

The reference path had to be presented 8 times before learning had converged and no further improvement could be obtained. The tracking error has been reduced by a factor 10 compared to the tracking error of the feedback controller. To obtain an even higher performance, the B-spline optimisation procedure, as described in section 3, is carried out. After the procedure has been performed twice, no further improvement of the tracking error could be obtained. In figure 15, the tracking error that resulted after optimisation is shown.

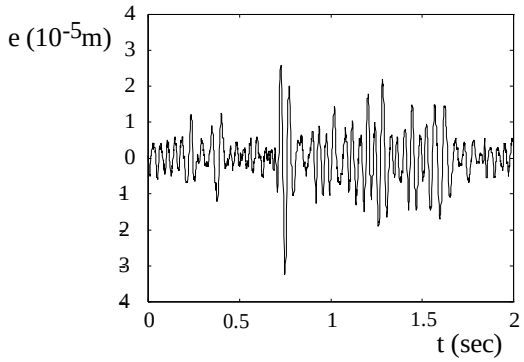

Figure 15: Tracking error using improved distribution

It can be seen that the over-all performance of the LFFC improved considerably by optimising the B-spline distribution.

5 Conclusions

Learning feed forward has been used to improve the performance of sub-optimal, robust feedback controllers. To what extent the LFFC is able to improve the performance depends on the distribution of the B-splines that are used to store the feed forward signal. Until recently the B-spline distribution was chosen by rule of thumb.

In this paper an iterative procedure that exploits fuzzy clustering techniques is proposed, which is able to determine the B-spline distribution such that the optimal feed forward signal can be best approximated. The iterative procedure consists of the following steps:

1. Choose an initial spline distribution.
2. Train the LFFC until convergence.
3. When the RMS tracking error over one trail has not decreased, end the iteration.
4. Using clustering techniques determine the new B-spline distribution and go to step 2.

In experiments LFFC was applied to a linear motor. Using a B-spline distribution that was determined by rule of thumb, he LFFC was able to reduce the tracking error of the feedback controller by a factor 10. Subsequently a B-spline distribution was determined using fuzzy clustering. The application of LFFC using the improved B-spline distribution showed a further decrease of the tracking error of almost 50%.

These results suggest that in the future, tuning of the B-splines in LFFC in order to optimise the performance, can be done in a systematic and automated way by using fuzzy clustering techniques. The fact that the fuzzy clustering algorithm is computationally expensive, is no problem since the optimisation procedure has to be done only once and can be carried out off-line.

6 References

[1] J.G. Starrenburg, W.T.C. van Luenen, W. Oelen and J. van Amerongen, "Learning feed forward controller for a mobile robot", Control Eng. Practise, Vol. 4, No. 9, pp. 1221-1230, 1996.

[2] W.J.R. Velthuis, T.J.A. de Vries and J. van Amerongen, Learning feed forward control of a flexible beam, Proc ISIC, Dearborn, MI, USA, pp. 103-108, 1996.

[3] G. Otten, T.J.A. de Vries and J. van Amerongen, "Linear motor motion control using a learning feed forward controller", IEEE ASME Tran. Mechatronics, to appear.

[4] M. Brown, and C. Harris, Neurofuzzy adaptive modelling and control, New York: Prentice Hall, 1994.

[5] W.J.R. Velthuis, P. Schaak, T.J.A. de Vries and J. van Amerongen, "Stability analysis of learning feed forward control:.

[6] R. Babuška, H.B. Verbruggen, Fuzzy set methods for local modelling and identification, in: Murray-Smith, R., and Johansen, T.A. (eds.): Multiple model approaches to non-linear modelling and control, London: Taylor & Francis, 1997.

[7] R. Babuška, Fuzzy Modelling and Identification, Ph.D. Thesis, University of Delft, Delft, The Netherlands, 1997.

[8] D.E. Gustafson and W.C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix", Proc IEEE CDC, San Diego, CA, USA, pp 761-766, 1979.