USC–SIPI REPORT #267

Two-Pass Orthogonal Least Squares Algorithm to Train and Reduce Fuzzy Logic Squares

by

Jörg Hohensohn and Jerry M. Mendel

August 1994

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

Abstract

Fuzzy logic systems (FLS's) can be designed using training data (i.e., given numerical input/output pairs) and supervised learning algorithms. FLS's can be viewed as a linear combination of nonlinear fuzzy basis functions (FBF's), and methods of linear optimization like orthogonal least squares (OLS) can be applied to tune the parameters of the FLS. The generated FLS should be as small as possible while fulfilling the required task of function approximation, so as to reduce computation when the designed FLS is applied to new data. OLS can be used to select a subset of most significant FBF's out of an initial pool set up by the training data. The drawback to OLS is that the resulting system still contains information from all M initial rules, derived from the training points, even though only the most important M_S rules have been established by OLS. This is due to a normalization of the FBF's, and leads to excessive computation times during further processing. Our solution is to construct new FBF's out of the reduced rulebase and to run OLS a second time. The resulting system not only is of reduced computational complexity, but is of very similar behavior to the unreduced system. The second run of OLS can be applied to a larger set of training data, which significantly improves the precision. We illustrate our two-pass OLS algorithm for prediction of the Mackey-Glass chaotic time series. Extensive simulations are given in which we look at tradeoffs among the design parameters of a FLS for this application.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Fuzzy logic systems (FLS's) can be efficiently designed when linguistic information is available to generate fuzzy rules. Such rules may also be generated from available sample data, namely, desired input-output-pairs. The goal of automated learning for FLS's is to generate a system from given training data. This system should have a reasonable size, best possible precision to model the training data, and, make good extrapolations for input data not included in training. An advantage of FLS's over feedforward neural networks is that linguistic information may be combined with numerical data. We focus on FLS's with n inputs, one output, singleton fuzzification, Gaussian membership functions, and height method of defuzzification. These FLS's are proven to be universal approximators in [7]. If the parameters in the antecedent membership functions of a FLS are fixed, the FLS becomes a linear combination of nonlinear fuzzy basis functions (FBF's). Classical linear regression, like the Gram-Schmidt orthogonal-least-squares (OLS) algorithm, can be used to select significant FBF's from the training data and to determine the weights (the parameters in the consequent membership functions) which appear in the FBF expansion. This is done in [7], but its algorithm has a disadvantage which, as we show in Chapter 2, makes it difficult to use for practical implementations. The FLS in [7] is not reduced in complexity by selecting a subset of FBF's, because each FBF is normalized by all membership functions (MBF's) of the rules which define the original FBF's. An application of OLS for two-layer feedforward neuronal networks [2] does not suffer from this problem, because their radial basis functions are not normalized.

A slightly modified OLS algorithm, which is descibed in Chapter 3, is able to work just with the reduced number of M_S rules, instead of the original M rules. This allows us to use OLS in a second pass to generate a new reduced fuzzy system. The set of training data for this second pass may even be larger than the one used in the first, which improves the results a lot. The first pass of OLS requires most of the computation; but, it can be made with a smaller data set, which speeds computation time.

In Chapter 4 we apply our new design procedure to the prediction of the Mackey-Glass chaotic time series.

Chapter 2

OLS as applied to FLSs

The key idea in [7] is to view a FLS as a *fuzzy basis function expansion.* The FBF's are derived from the rules of the FLS's fuzzy associative memory (FAM) bank. A system with M rules contains M FBF's:

$$p_j(\underline{x}) = \frac{\prod_{i=1}^{n} \mu_{F_i^j}(x_i)}{\sum_{j=1}^{M} \prod_{i=1}^{n} \mu_{F_i^j}(x_i)}, \quad j = 1, 2, \ldots, M \tag{2.1}$$

with the antecedent Gaussian membership functions

$$\mu_{F_i^j}(x_i) = \exp\left[-\frac{1}{2}\left(\frac{x_i - \bar{x}_i^j}{\sigma_i^j}\right)^2\right]. \tag{2.2}$$

Other MBF's are also possible. We will call the parameters $\bar{x}_i^j$ and σ_i^j $(i = 1, 2. \ldots, n)$ the set of antecedent parameters for the jth rule.

A FBF can be viewed as providing the relative degree for which the jth rule fires. The numerator gives the degree to which a particular rule fires (the product is used as an AND-operator); the denominator of the FBF gives the sum of the degrees for all rules. Due to the normalization of the denominator, each FBF uses the whole FAM-bank.

The FLS using height method of defuzzification can be written as the following FBF expansion

$$f(\underline{x}) = \sum_{j=1}^{M} p_j(\underline{x})\theta_j \tag{2.3}$$

which is a weighted sum of our nonlinear FBF's. The OLS algorithm is capable of selecting a subset of $M_S < M$ basis function vectors and computing a weight vector $\underline{\theta} = [\theta_1, \cdots, \theta_{M_S}]^T$ to approximate a given training vector $\underline{d}$. In matrix form, it calculates a vector $\underline{\theta}$ to make the product $P\underline{\theta}$ as close to $\underline{d}$ as possible, where $\underline{d} = [d(1), \cdots, d(N)]^T$, $P = [\underline{p}_1, \cdots, \underline{p}_M]$ with $\underline{p}_i = [p_i(1), \cdots, p_i(N)]^T$, $\underline{\theta} = [\theta_1, \cdots, \theta_{M_S}]^T$, and N is the number of training samples. M is initially set equal to N to determine the regressor vectors $\underline{p}_j$. The elements of this vector are the values of the related FBF, computed for each input vector of the training data. These regressors are the input for the OLS algorithm, which then selects a subset of M_S of the M regressor vectors and computes the weights $\theta_1, \cdots, \theta_{M_S}$ for (2.3).

Using only the selected FBF's with the resulting weights gives a system that models the training data quite well and is able to generalize for data not included in training; however, it is incorrect to believe that the FLS has been totally reduced, because the denominator of each FBF in (2.1) contains all MF's, including those of rules belonging to the $M - M_S$ non-selected FBFs. At this point, it is impossible to totally discard the non-selected FBF's; hence, the present FLS can't be reduced to fewer than M rules, even though it contains fewer than M FBF's. To calculate the output value of the FLS for an input vector $\underline{x}$, $N \times n$ MF's have to be computed, regardless of how many FBF's are selected by the OLS algorithm. If we don't reduce the amount of computation during later processing, why should we apply the OLS algorithm at all? We could keep all the M FBF's and have a very precise system. To do this is not reasonable, if, as in signal processing applications, the amount of training data is large. We want a way to really reduce the system.

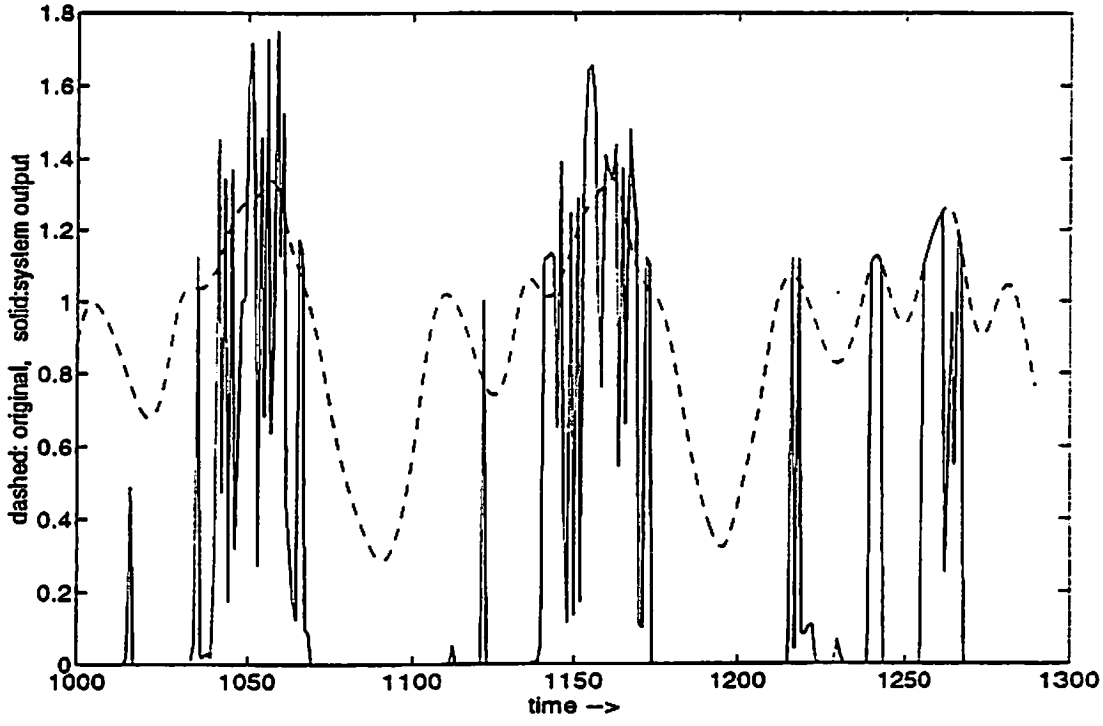

Figure 2.1: Poor system behaviour caused by too small σ, dashed: training data, solid: FLS output

Chapter 3

Reducing the FLS using OLS

Our approach is to use the OLS algorithm two times. The first application of the OLS algorithm is almost as described in [7], and picks the M_S most significant FBFs out of $M = N$ initially established FBFs. The second application of the OLS algorithm is described in detail below.

Suppose we are given N training data pairs: $\{\underline{x}(t), d(t)\}$, $t = 1, 2, \ldots, N$. Before training, we initially choose and compute the parameters of the M antecedent FBF's as follows:

$$\bar{x}_i^j = x_i(j), \quad \sigma_i^j = \max[(x_i(j)) - \min(x_i(j))] \frac{n}{M_S},$$
$$i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, N. \tag{3.1}$$

Note that some of our experiments that used the algorithm in [7] for σ_i^j (which differs from (3.1) in that n/M_S is replaced by $1/M_S$) showed that the resulting FBF's degenerated to very narrow functions which do not approximate functions very well (Fig. 2.1). The above algorithm for the choice of σ_i^j is only heuristic. In general the choice of σ_i^j depends on the application. The final choice for σ_i^j can make a big difference in the approximation results. We have more to say about this in Chapter 4.

The number of FBF's, M, initially equals the number of training samples N. For every FBF we have to compute a vector $\underline{p}_i$, whose elements are given by (2.1). OLS is run, but after its completion we keep only the selected FBF's, together with their entries in the FAM-Bank and the respective antecedent parameters ($\bar{x}_i$, σ_i). As stated above, this changes the saved FBF's. The are now only normalized by $\sum_{j=1}^{M_S} \prod_{i=1}^{n} \mu_{F^j}(x_i)$. When plotted, we have observed that the new FBF's look quite similar to the original ones; basically, their ranges change, whereas their shapes remain the same. We use this first application of OLS to reduce the number of FBF's; we do not use it to choose the final θ-parameters in (2.3).

The task now is to find the θ-parameters for the given rules, so as to model the training data as best as possible. We use OLS a second time to do this; but, we modify the algorithm in [7], because it requires as many FBF's as training data points. We determine $\underline{\theta}$ for only M_S FBF's, but want to optimize it for all N given training samples. Fortunately, the original algorithm, as described in [2], lets us do this. Our second run of OLS is much faster than the first run, because usually $M_S \ll M$. Note, also, that there is no need to use the same training samples in the second application of OLS as in the first application of OLS. The training set may even be much larger than the first training set without costing too much computation. We use this approach in Chapter 4.

The complete set of equations associated with the OLS algorithm are stated next, for convenience to the reader:

- At the first step of the OLS algorithm, the first basis vector $\underline{w}_1$ is chosen. For $1 \leq i \leq M$ compute:

$$\underline{w}_1^{(i)} = \underline{p}_i, \quad g_1^{(i)} = \frac{(\underline{w}_1^{(i)})^T \underline{d}}{(\underline{w}_1^{(i)})^T \underline{w}_1^{(i)}}, \tag{3.2}$$

  and the error-reduction ratio

$$[err]_1^{(i)} = (g_1^{(i)})^2 \frac{(\underline{w}_1^{(i)})^T \underline{w}_1^{(i)}}{\underline{d}^T \underline{d}}. \tag{3.3}$$

Find the FBF with the largest error reduction ratio

$$[err]_1^{(i_1)} = \max([err]_1^{(i)}), \quad 1 \le i \le M \tag{3.4}$$

and select the first basis vector $\underline{w}_1$ and the first element g_1 of the LS solution vector

$$\underline{w}_1 = \underline{w}_1^{(i_1)} = \underline{p}_{i_1}, \quad g_1 = g_1^{(i_1)}. \tag{3.5}$$

- At the k'th step, where $2 \le k \le M_S$, for $1 \le i \le M$, $i \ne i_1, \ldots, i \ne i_{k-1}$, compute

$$\alpha_{jk}^{(i)} = \frac{\underline{w}_j^T \underline{p}_i}{\underline{w}_j^T \underline{w}_j}, \quad 1 \le j < k \tag{3.6}$$

$$\underline{w}_k^{(i)} = \underline{p}_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} \underline{w}_j, \quad g_k^{(i)} = \frac{(\underline{w}_k^{(i)})^T \underline{d}}{(\underline{w}_k^{(i)})^T \underline{w}_k^{(i)}}, \tag{3.7}$$

and

$$[err]_k^{(i)} = (g_k^{(i)})^2 \frac{(\underline{w}_k^{(i)})^T \underline{w}_k^{(i)}}{\underline{d}^T \underline{d}}. \tag{3.8}$$

Again, find the FBF out of the remaining ones with the largest error reduction ratio

$$[err]_k^{(i_1)} = \max([err]_k^{(i)}), \quad 1 \le i \le M, \quad i \ne i_1, \ldots, \quad i \ne i_{k-1} \tag{3.9}$$

and select the next orthogonalized basis vector $\underline{w}_k$ and the kth element of the LS solution vector g_k

$$\underline{w}_k = \underline{w}_k^{(i_1)} = \underline{p}_{i_k}, \quad g_k = g_k^{(i_k)}. \tag{3.10}$$

- Solve the triangular system

$$A\underline{\theta} = \underline{g}, \tag{3.11}$$

where

$$A = \begin{bmatrix} 1 & \alpha_{12}^{(i_2)} & \alpha_{13}^{(i_3)} & \cdots & \alpha_{1M_S}^{(i_{M_S})} \\ 0 & 1 & \alpha_{23}^{(i_3)} & \cdots & \alpha_{2M_S}^{(i_{M_S})} \\ 0 & 0 & \ddots & & \vdots \\ \vdots & & & & \\ & & & 1 & \alpha_{M_S-1,M_S}^{(i_{M_S})} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \tag{3.12}$$

and

$$\underline{g} = \begin{pmatrix} g_1 \\ \vdots \\ g_{M_S} \end{pmatrix}, \quad \underline{\theta} = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_{M_S} \end{pmatrix}. \tag{3.13}$$

The solution $\underline{\theta}$ can be found through back-substitution or by brute-force $\underline{\theta} = A^{-1}\underline{g}$. The matrix A can be inverted without numerical problems because it is ideally conditioned, all eigenvalues are the same.

The final FBF expansion is

$$f(\underline{x}) = \sum_{j=1}^{M_S} p_{i_j}(\underline{x})\theta_j. \tag{3.14}$$

Our two-step OLS procedure to reduce and optimize a FLS is:

- *Set $N = N_1$ and establish $M = N_1$ initial FBF's*

  Keep track of the antecedent parameters $(\bar{x}_i, \sigma_i)$ which describe the appropriate rule. Compute $N \times 1$ regressor vectors $\underline{p}_i$ for all antecedents, the elements of $\underline{p}_i$ are the FBF's computed for every given input vector $\underline{x}(t)$.

- *Run the OLS algorithm to select M_S significant rules out of $M = N_1$*

  At this point, only the M_S rules are of interest, and we only store their antecedent parameters. We do not solve the triangular system in (3.11) in this first application of OLS.

  Remark: It is not necessary to fix (i.e. prespecify) M_S. As in [2], we can terminate the first application of OLS at the M_Sth step when

$$1 - \sum_{j=1}^{M_S} [err]_j < \rho, \tag{3.15}$$

  where ρ is a prespecified tolerance and $0 < \rho < 1$.

- *Discard everything but the selected sets of antecedent parameters*

  Now we set $M = M_S$, to reduce the number of retained rules; hence, we need the regressor vectors $\underline{p}_i$ for only the M_S rules. The number of p_i vectors is reduced, but not their dimension, which is still $N_1 \times 1$.

- *Run OLS for a second time to obtain the θ-parameters*

  OLS now uses all the M_S FBF's; hence, it does not have to find the FBF with the largest error reduction ratio. It merely orthogonalizes the modified set of M_S FBF's by using Equations (3.2), (3.6), and (3.7), and then determines $\underline{\theta}$ from (3.11) - (3.13). In general, this second run is computationally much less extensive than the first run, because we do not need to choose the best subset of M_S FBF's out of M FBF's.

  Remark: It is not necessary to run OLS the second time using the same set of training data as the first run, i.e. we can choose $N = N_2 \neq N_1$. The training set may even be much bigger (i.e., $N_2 \ll N_1$), which can improve the resulting system a lot and may offset using a small training set on the first run.

We now have a completely reduced fuzzy system, one based on M_S rules. This system can be easily used to compute an output value for a new input vector $\underline{x}$. The computation time is proportional to $M_S \times n$ MF's. Compared to the method in [7], this represents a real-time computation reduction of M_S/N, which can be quite substantial.

Chapter 4

Prediction of Chaotic Time Series

Let $x(k), k = 1, 2, \ldots$ be a time series. The problem of time-series prediction is: given a window of n past measurements, $x(k-n+1), x(k-n+2), \ldots, x(k)$, determine a future value $x(k+l)$, where n and l are fixed positive integers.

$$x(k-n+1), x(k-n+2), \ldots, x(k) \longrightarrow x(k+l) \tag{4.1}$$

In our work we have chosen $l = 1$, i.e., we have developed fuzzy systems to perform prediction of time series one step into the future.

The OLS methods require a training set of data; hence, we assume that $x(1), x(2), \ldots, x(P)$ are given. The size of the window of past measurements, n, must be fixed ahead of time. This corresponds to the dimension of the system's input. Clearly, $n < P$. The set of training samples contains $P - n$ elements in the form of $\underline{x}_i = col[\text{input:desired output}]$. More specifically, $\underline{x}_1 = col[x(1), \ldots, x(n) : x(n+1)]$, $\underline{x}_2 = col[x(2), \ldots, x(n+1) : x(n+2)]$, $\ldots$, $\underline{x}_{P-n} = col[x(P-n), \ldots, x(P-1) : x(P)]$.

4.1 The Mackey-Glass Chaotic Time Series

The time series we shall apply our predictors to is the *Mackey-Glass chaotic time series*. It has become a benchmark problem for time-series prediction. Chaotic time-series are generated from deterministic nonlinear systems and are sufficiently complicated so that they appear to be random; however, because they are deterministic in nature, chaotic time series are not random. The Mackey-Glass series is generated from the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{4.2}$$

It is meant to be a crude description of the regulatory mechanism for human blood cell production [6]. The delay τ is the time that cells need to mature. It can be increased by certain diseases, causing aperiodic (chaotic) fluctuations. The solution of the differential equation is a continuous signal, which we discretize to a time series at integer units. The delay τ affects the fractal dimension of the attractor; in general, the higher the τ, the higher the fractal dimension. There are four important regions for τ:

0	$\leq$	τ	$\leq$	4.53	: stationary
4.53	$<$	τ	$\leq$	13.3	: stable limit cycle
13.3	$<$	τ	$\leq$	16.8	: period doubling, bifurcation sequence
16.8	$<$	τ			: chaotic

For further details, see [3]. When $\tau > 16.8$, this system exhibits chaotic behaviour, with some limit cycles interspersed. In our experiments we have chosen values around $\tau = 30$, for which it is chaotic. Figures 4.1 to 4.4 illustrate the series for periodic ($\tau = 15$) as well as chaotic ($\tau = 30$) cases. Both time and phase plots are shown.

There is no closed-form solution for this equation, and it can only be approximated, using numerical integration methods. A characteristic feature of chaotic systems is the exponential growth of disturbances.

A small change (error) of a state is amplified exponentially over time. Finally, the disturbance becomes as big as the signal and the systems behaviour is permanently changed. This causes a problem with approximate solutions; they can stay close to the true trajectory only for a limited time. The unavoidable errors finally cause divergence. Another cause for errors is that the Mackey-Glass equation is continuous and has to be discretized for a numerical integration.

We tried different approaches to solve the equation. Integrations using Euler and 4th-order Runge-Kutta methods with different stepsizes were tested. Runge-Kutta requires a lot more computation, and the retarded argument has to be interpolated (we used cubic spline-interpolation). We got best results with simple Euler integration and a stepsize of 1/5000. The resulting trajectories are converging for about 1600 units in time, after that the errors due to finite precision and discretization cause divergence. Although the signal is different from the theoretical solution then, the system's dynamics are still very close to the original. The data is still useful for training and application of our FLS's.

4.2 Training and Results

Figures 4.5 to 4.9 show prediction results for fuzzy systems with $M_S = 50$ rules and $n = 10$-dimensional input data. All results where obtained by training in the first pass of OLS for a section of 301 data points, when $1000 \leq t \leq 1300$, yielding 291 training vectors. Unless otherwise specified, the second pass of OLS was performed with the same data. Training was done in 3 different ways:

(1) Only one-pass OLS was performed, including the computation of the θ-parameters; Figs. 4.5 and 4.6 show the performance of the resulting FBF expansion. The first 291 points of the graph show the behaviour for the training data. the rest is the real prediction.

(2) Our two-pass OLS algorithm was applied. Figures 4.7 and 4.8 show the performance of the reduced system, where again the first 291 points of the graph show the behaviour for the training data. the rest is the real prediction. The results are comparable, but computation time for the second approach is much faster than for the first approach. Note that the error for the reduced system in Fig. 4.8 is slightly larger than for the one-pass output in 4.8. This is typical because due to the renormalization the shapes of the FBF's change slightly when the least significant ones are removed. The solution becomes suboptimal. We now show how to compensate for this.

(3) The amount of training data for the second pass of OLS was increased to the region $1000 \leq t \leq 2000$. yielding 991 training samples. Also, the standard deviations σ_i^j of the MF's were increased to $\sigma_i^j = 0.3$ from the value obtained from (3.1); $\sigma_i^j = 0.8$ was found by trial and error to give best results. Very small values of σ_i^j gave very jagged performance, as depicted in Fig. 2.1. As σ_i^j values were increased, better and better predictions were obtained. Then, as σ_i^j values became too large, predictions once again deteriorated, although this time the predictions were not jagged; instead they became too smoothed out. The second-pass of OLS training gives a FLS with outstanding performance. When plotted in the same graph, original time-series and its prediction are indistiguishable.

4.3 Choice of Parameters

In a FLS, there are some free parameters to choose: the number of rules (FBF's) M_S in the final FLS, the number of inputs. and the standard deviations σ_1 and σ_2 of the Gaussian membership functions for the first and second pass of OLS training, respectively. The number of free parameters depends on the application. In our example of time-series prediction we are free to choose the number of inputs n (in general, this is not the case; n is usually not a free parameter). On the other hand, in this application all inputs use the same data, but at different points in time; therefore all MBF's have the same common σ. In general this is not the case, i.e., the inputs might have completely different meanings and ranges, in which case each input has its own σ, which increases the number of parameters. The amount of training data for each pass can also be viewed as a design parameter. In an experiment we used different 301 points for the first-pass training data of 100 FLS and found that approximately 1/10th of them show poor out-of-sample performance because the training data by chance wasn't rich enough. For all later experiments therefore the amount of training data was increased to 501 points. We did not use more because there is a tradeoff between reliability of training

and first-pass training time, the latter growing quadratic with the amount of trainig data. Figure 4.10 shows the training data leading to the worst performance as an example. In the first-pass training region [1000,1300], from where the rules are picked, the series is by chance almost periodic. The rules get optimized to model these dynamics. After one-pass training the modelling for this area works fine, but out-of-sample performance expectedly isn't good. A second pass training with the whole shown area can't improve this, since the rules are fixed. The quality of the predictor gets visible after the second pass, a bad system shows poor performance even on its second pass training data and can be rejected by a suitable algorithm at that point.

We performed a lot of experiments to find out what the optimum design parameters are for the the Mackey-Glass predictor, to get a baseline of the possible quality of predictors generated by our algorithm. In most experiments, one or two design parameters are varied while the others are held fixed. Our measure for the performance is the RMS prediction error. To get more meaningful results, we perform Monte-Carlo simulations, average across 10-50 realizations. A realization is the output of a FLS; they differ by the random initial conditions of the Mackey-Glass series they were trained for. The most trivial algorithm for a prediction of a time series is the *hold-method*, which uses the last observed value as the prediction for the next future value, e.g., like predicting today's temperature as tomorrow's. If the time series changes slowly, this is quite adequate. The error of this method is proportional to the derivative of the time series. A more sophisticated predictor must outperform this benchmark to be useful; therefore we compare our results to the hold-method.

We now discuss in more detail the effect of the parameters.

- Number of rules M_S in the final FLS: The number of rules in the final FLS establishes how many features it can recognize. Too few wouldn't be enough for a system description; too many lead to a jittery system. There is a dependency to the number of inputs, the more inputs are used, the more versalite are the states the FLS can "see". To handle these properly, more rules are required. In an experiment we used 10 inputs and varied M_S. Figure 4.11 shows the results, the performance for $20 \leq M_S \leq 60$ is quite stable. Our FLS is very robust to a wide range of values for M_S; hence, above a certain value, M_S is not a critical parameter.

- Number of inputs n to the FLS: The window of past observations is the input vector for the FLS. If the window is too small, the FLS can't recognize the system state. With increasing window size the number of rules should be increased as well, to compensate for the greater variety. This dependency of M_S and n is visible in Fig. 4.12. It is also visible that the region of low error becomes wider for larger systems, meaning that those are less critical to size. The diagonal of the lowest errors is approximately a straight line described by the relation

$$n = 2 + 0.2 * M_S. \tag{4.3}$$

  Along this line the error decreases with growing system size, as expected.

  The numerical data for this plot is given in Table 4.1.

  Fig. 4.13 shows the results of an experiment where the number of rules was held fixed, while varying the number of inputs. The minimum is at $n = 11$ for our $M_S = 50$ rules; beyond $n = 11$ the performance decreases slowly.

  When the number of inputs is set to one, the resulting system performs very similar to the hold-method. This is not surprising, because one input cannot sense any trend and therefore the FLS learns to repeat the last value. Figure 4.14 shows how the system output lags behind the true series when $n = 1$; just like the hold-method.

- Shape of the antecedent membership functions for the first and second pass, in our case the standard deviation of the Gaussian MBF's σ: We found that a major reason for good performance of the two-pass method is the independence of σ_1 and σ_2. We get better results if σ_2 is larger than σ_1. Figure 4.15 shows results of Monte-Carlo simulations in which both values were varied. It depicts the average RMS prediction error for 50 realizations. The minimum was found at $\sigma_1 = 0.05$ and $\sigma_2 = 0.7$. From the surface plot it can be seen that σ_1 is less critical to choose (it must not be too small) than σ_2.

The numerical data for this plot is given in Table 4.2.

In another experiment we tested how robust the generated FLS's are to changes in the system's dynamics. We applied FLS's that were trained for $\tau = 30$ to Mackey-Glass time series with different (randomly chosen) τ. Results are shown in Fig. 4.16. Surprisingly, the error is still low compared with the hold-method (which is about 0.03, way above our plot), even for significantly changed τ.

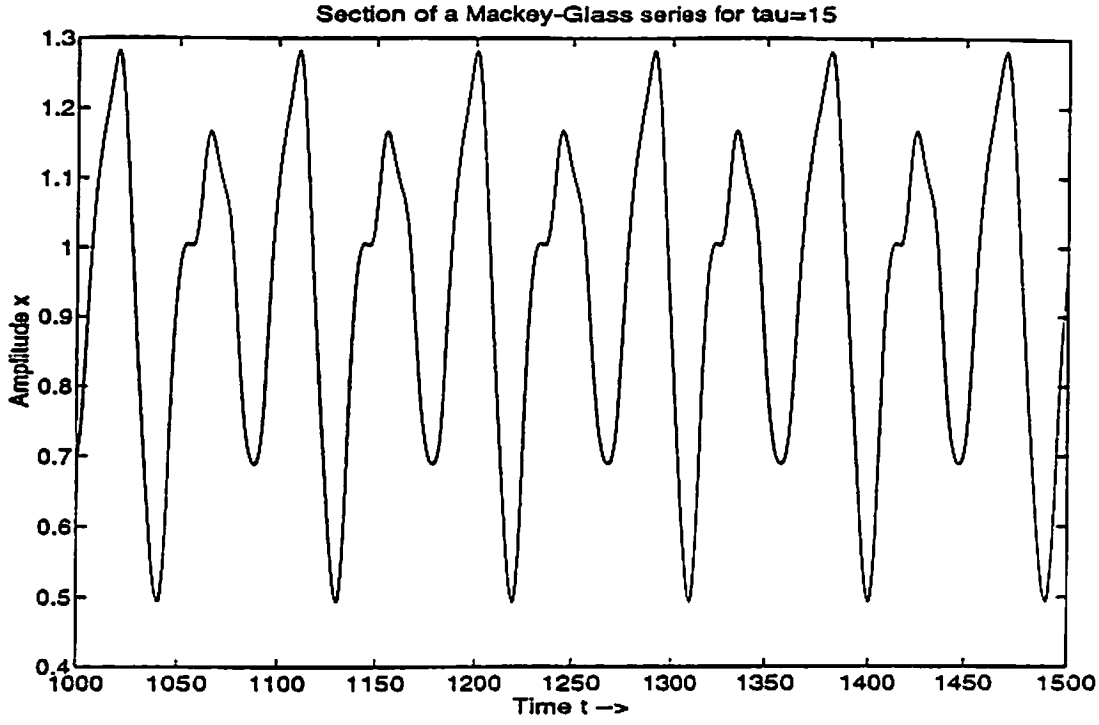

Figure 4.1: Time series for $\tau = 15$ (periodic case)

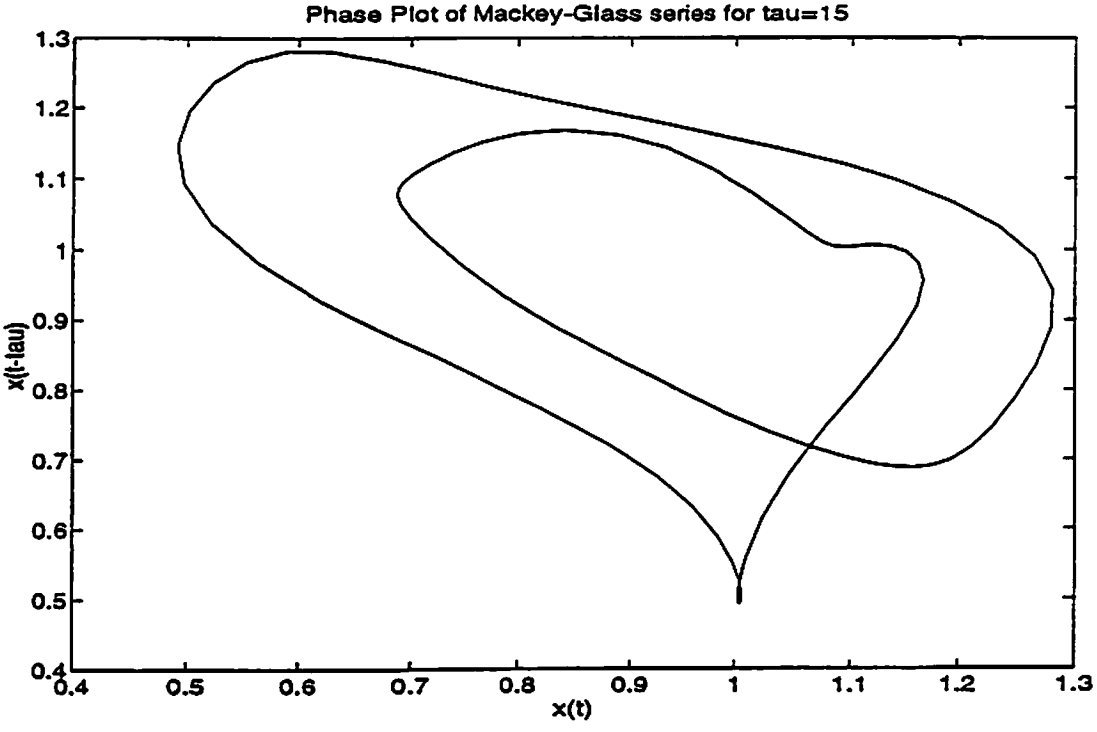

Figure 4.2: Phase plot of the limit cycle for $\tau = 15$

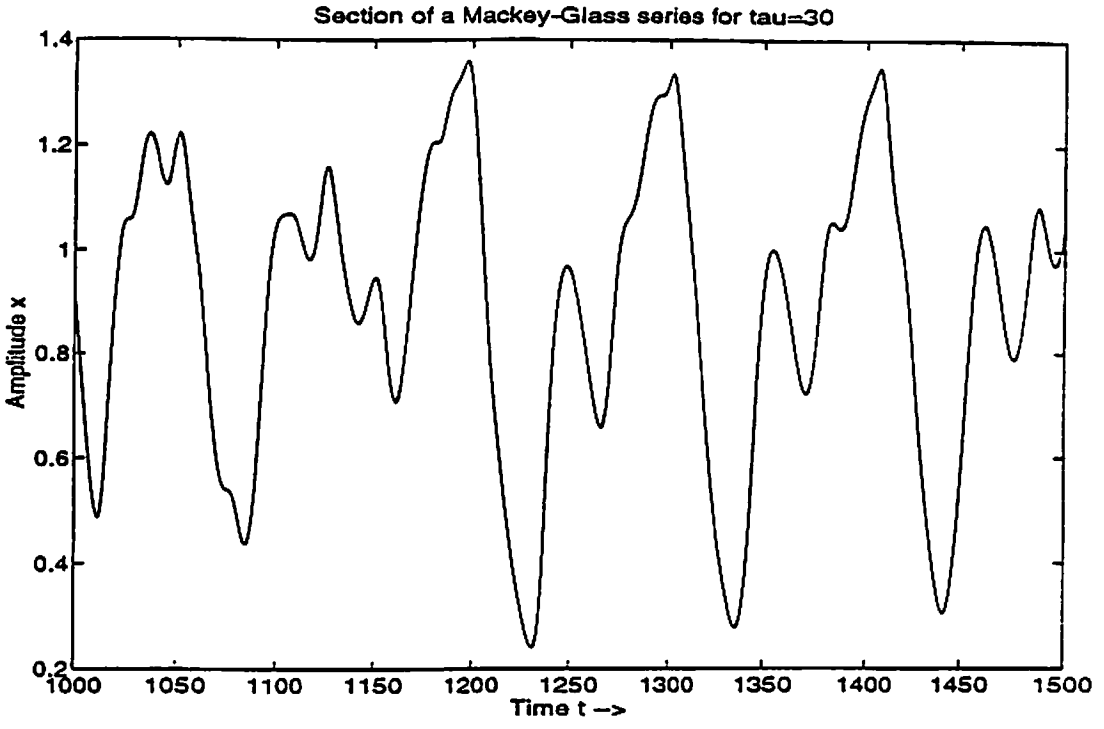

Figure 4.3: Time series for $\tau = 30$ (chaotic case)

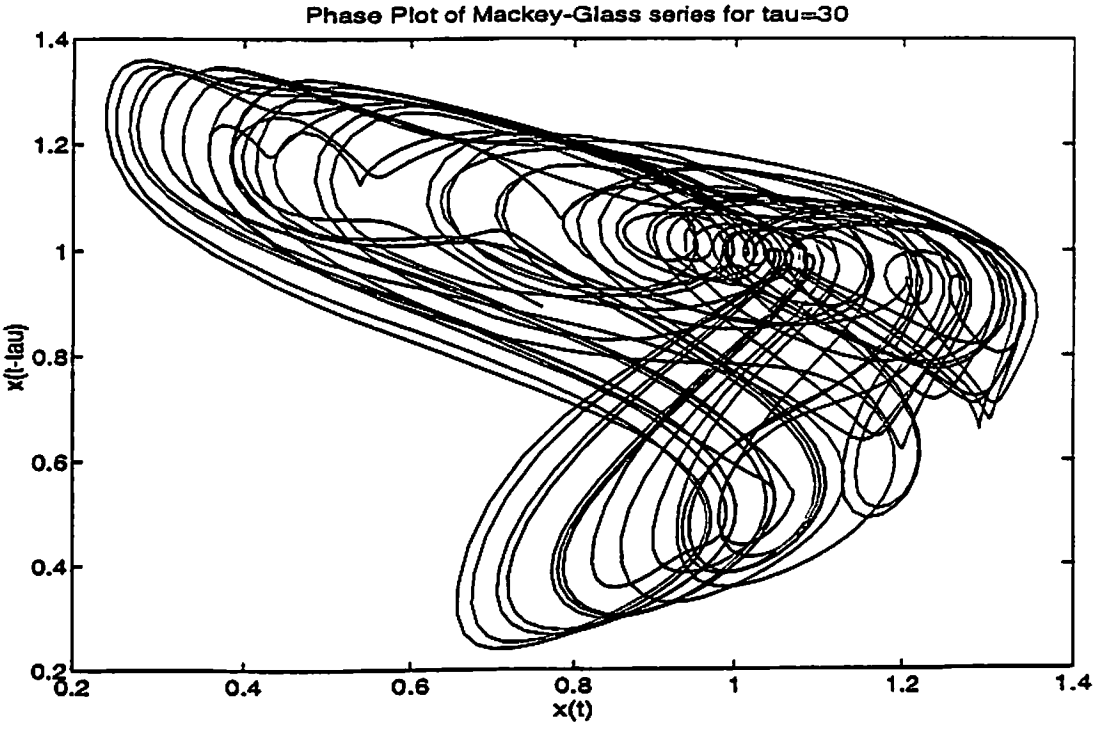

Figure 4.4: Phase plot of the chaotic attractor for $\tau = 30$

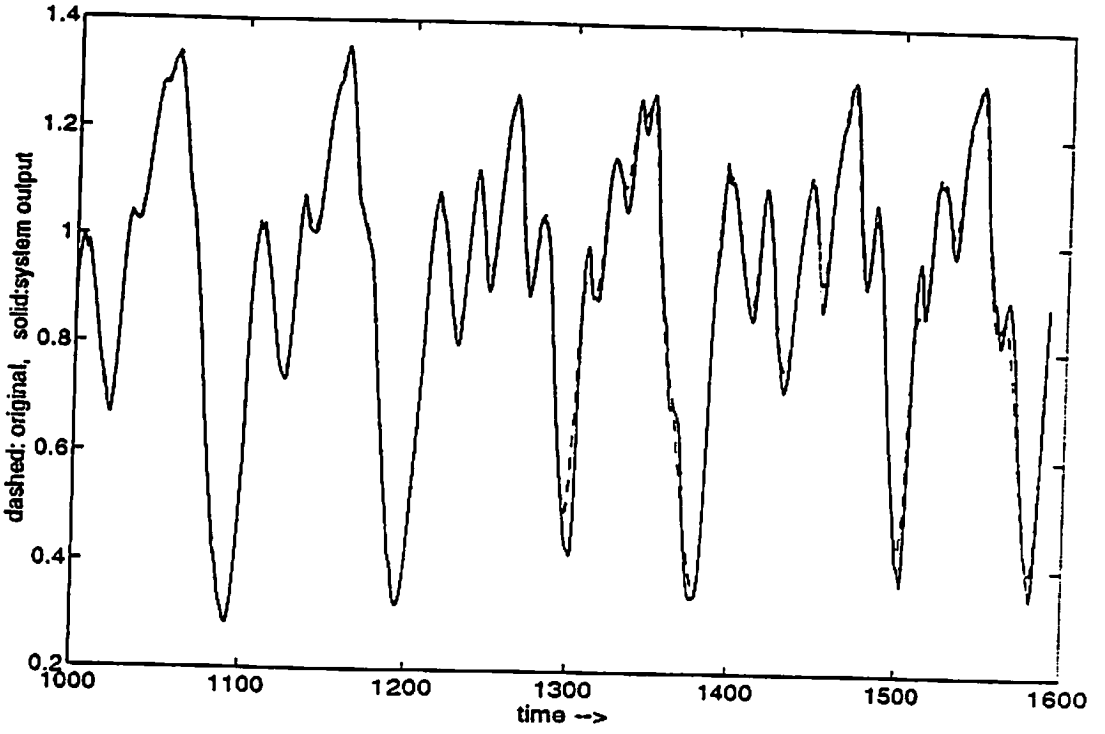

Figure 4.5: FBF expansion (one-pass OLS) dashed: desired output, solid: FBF expansion output

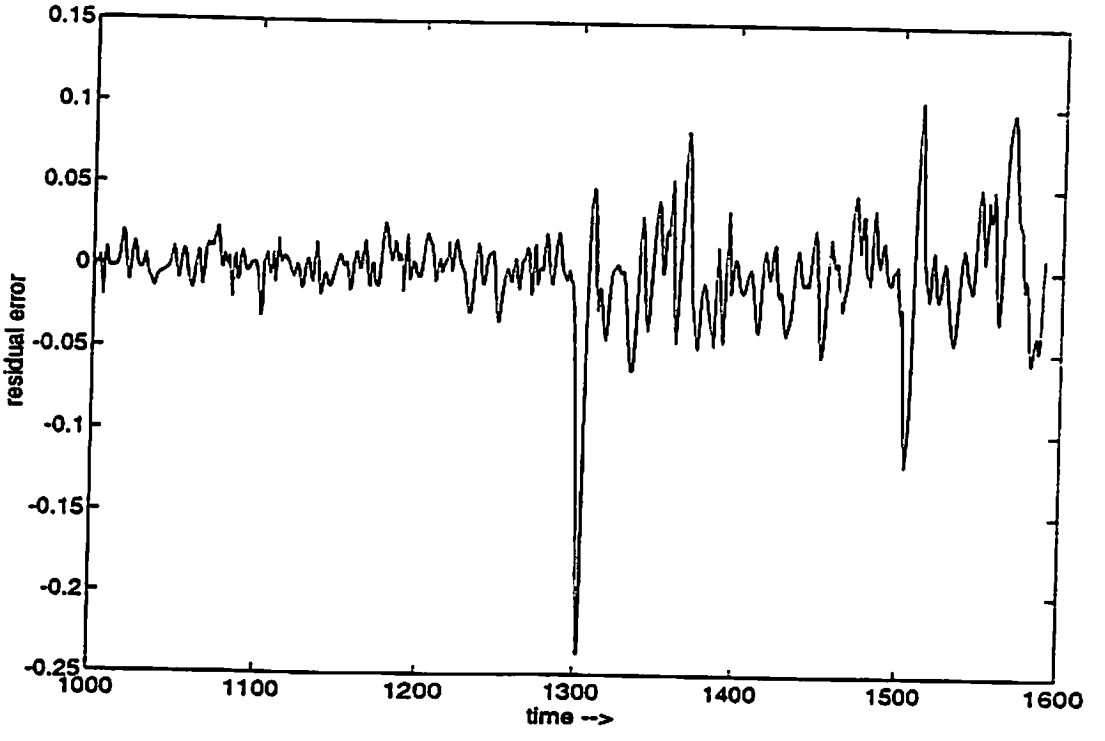

Figure 4.6: Residual error for FBF expansion

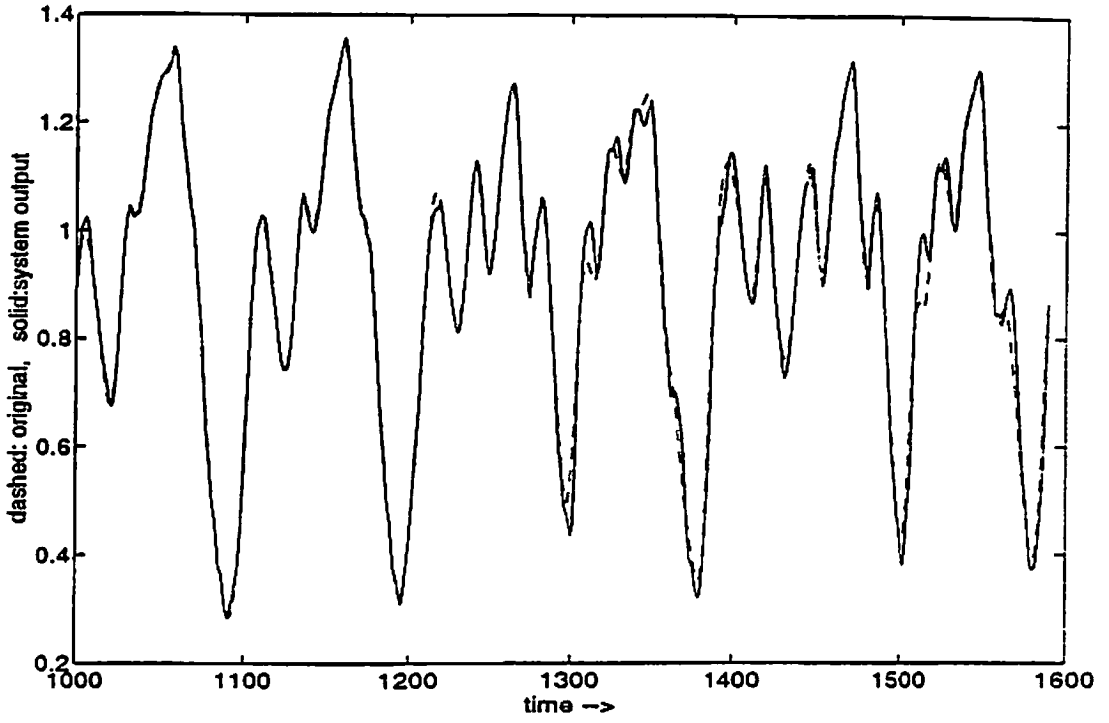

Figure 4.7: Reduced fuzzy system (two-pass OLS) dashed: desired output, solid: FBF expansion output

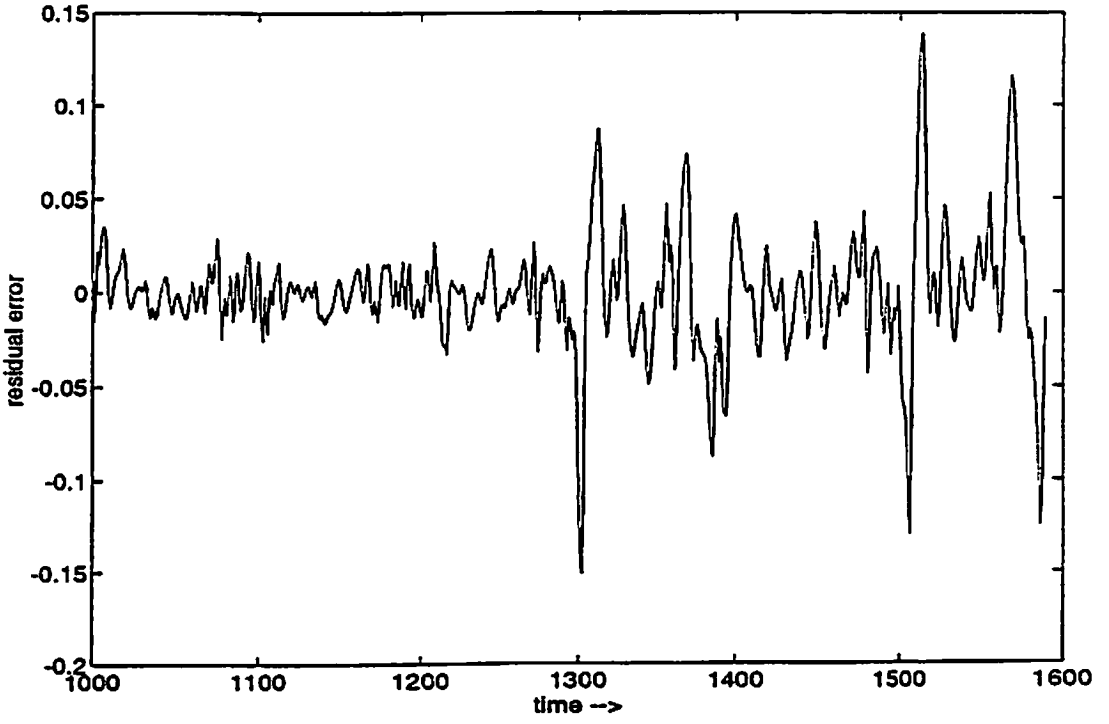

Figure 4.8: Residual error of reduced fuzzy system

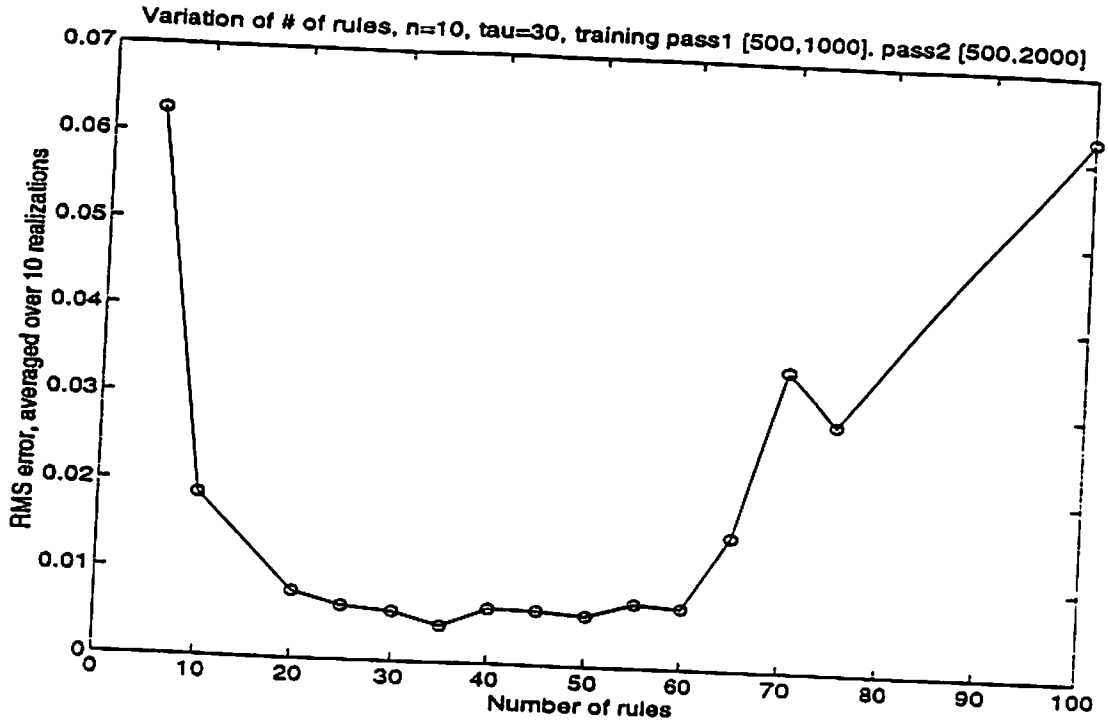

Figure 4.11: Variation of the number of rules

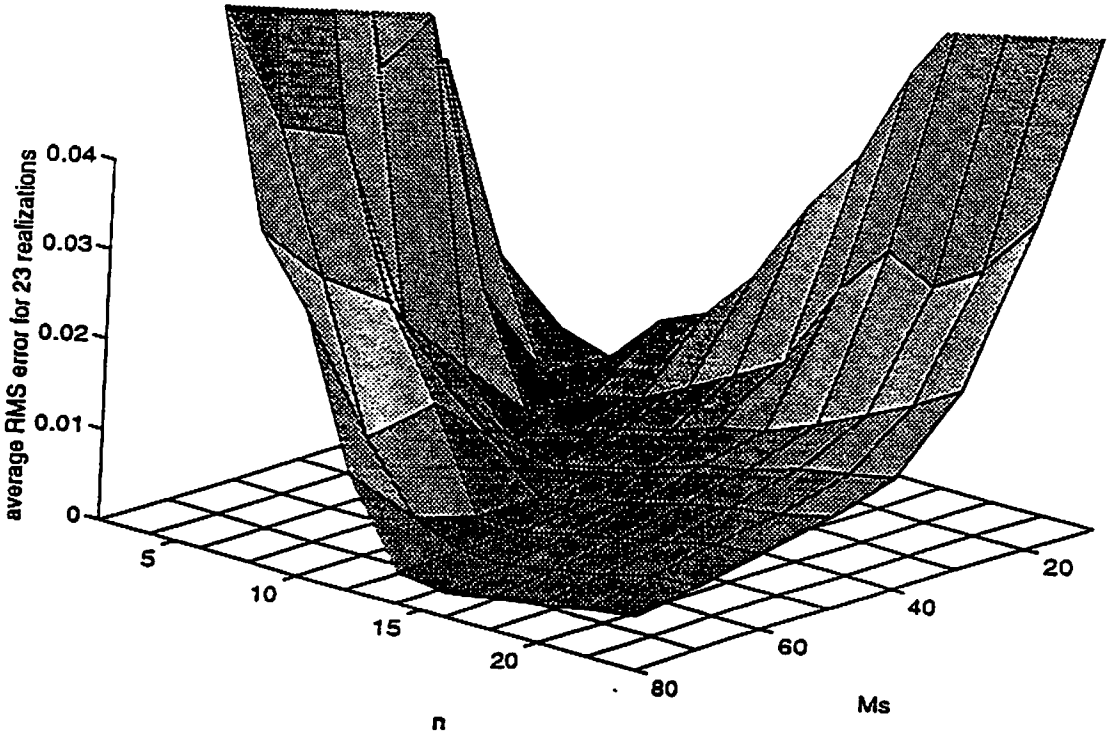

Figure 4.12: Variation of both number of inputs and rules

$n \backslash M_S$	10	20	30	40	50	60	70	80
2	0.0053	0.0158	0.0399	0.0414	0.0853	0.0948	0.1413	0.1219
4	0.0037	0.0039	0.0155	0.0485	0.0558	0.0819	0.1087	0.1099
6	0.0093	0.0032	0.0037	0.0107	0.0182	0.0415	0.0446	0.0615
8	0.0116	0.0044	0.0028	0.0047	0.0122	0.0243	0.0293	0.0369
10	0.0173	0.0071	0.0049	0.0030	0.0082	0.0145	0.0131	0.0297
12	0.0268	0.0100	0.0062	0.0045	0.0032	0.0036	0.0057	0.0129
14	0.0340	0.0131	0.0080	0.0058	0.0045	0.0033	0.0036	0.0034
16	0.0447	0.0229	0.0092	0.0076	0.0057	0.0045	0.0034	0.0026
18	0.0524	0.0287	0.0115	0.0087	0.0072	0.0057	0.0044	0.0036
20	0.0541	0.0262	0.0141	0.0103	0.0083	0.0067	0.0054	0.0043
22	0.0575	0.0295	0.0168	0.0114	0.0092	0.0080	0.0068	0.0054
24	0.0582	0.0365	0.0199	0.0125	0.0104	0.0085	0.0071	0.0061

Table 4.1: Average RMS error for variation of both n and M_S over 23 realizations

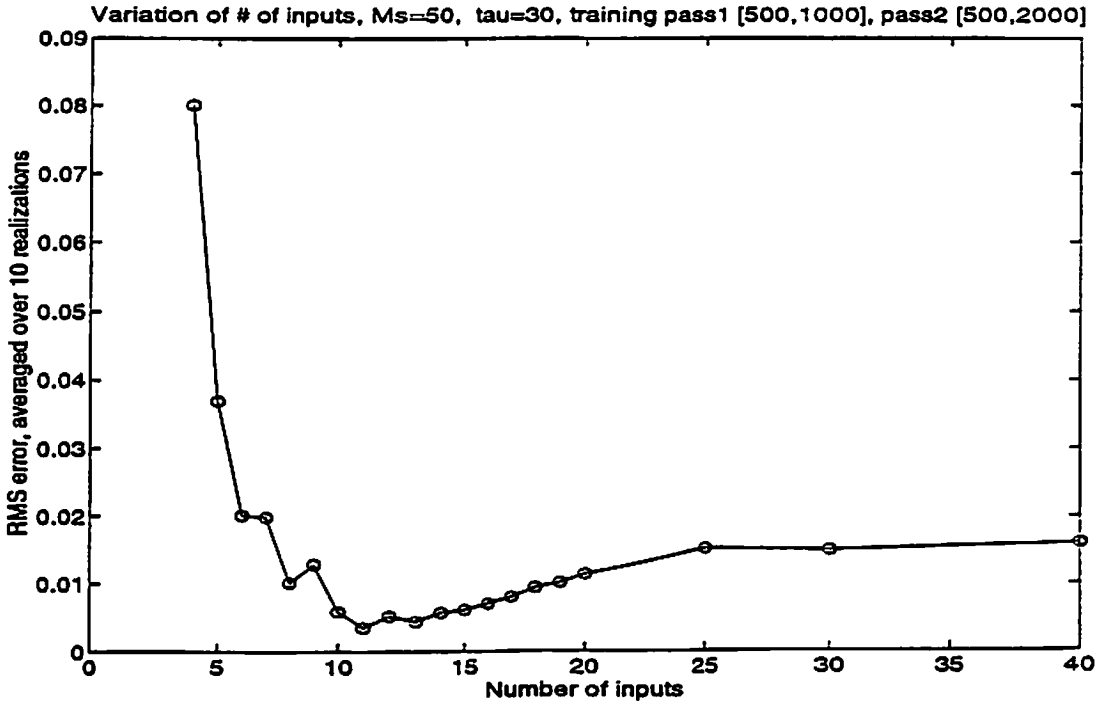

Figure 4.13: Variation of the number of inputs

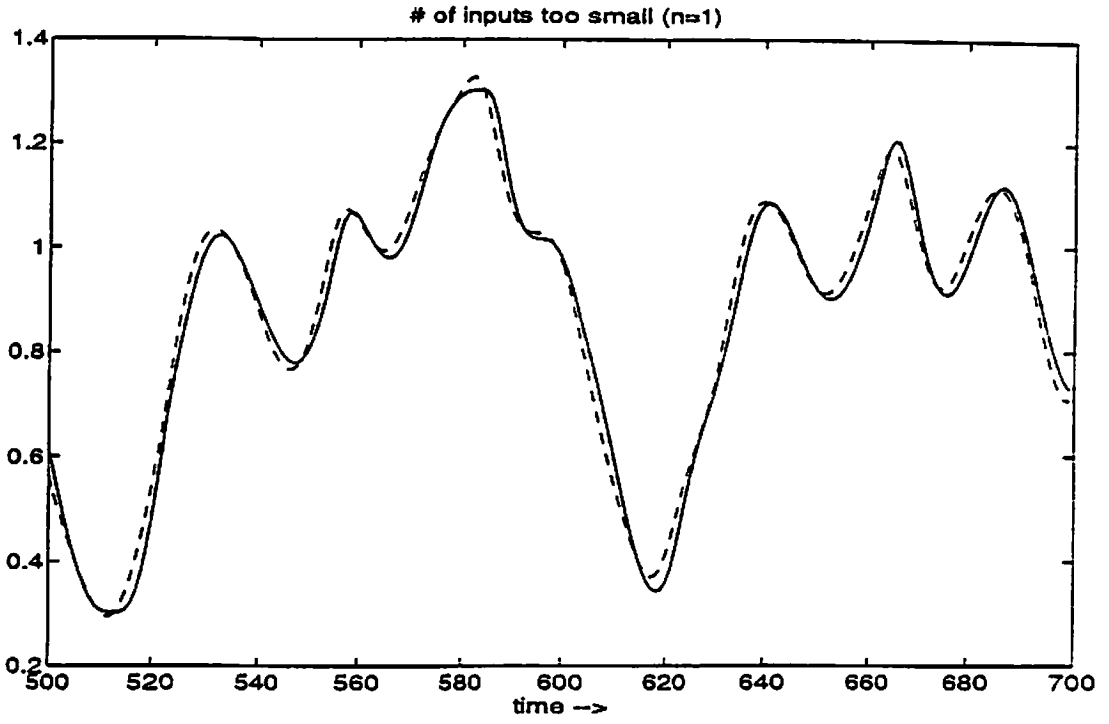

Figure 4.14: Performance with only one input; dashed: true series, solid: prediction

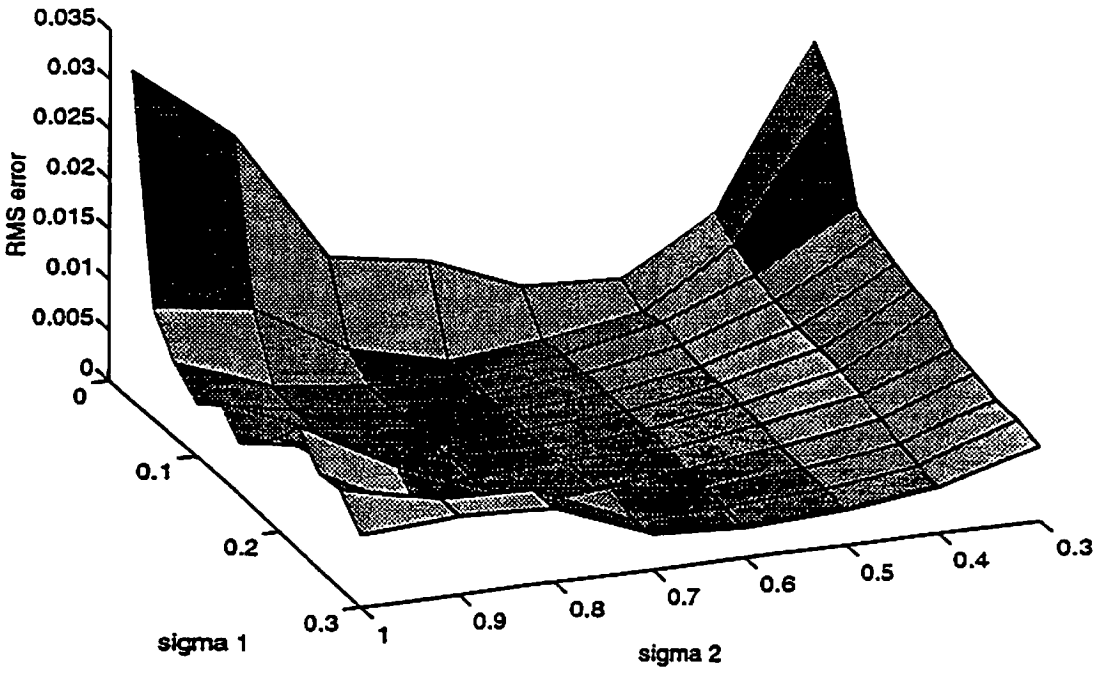

Figure 4.15: Variation of both σ_1 and σ_2

$\sigma_1 \backslash \sigma_2$	0.3000	0.4000	0.5000	0.6000	0.7000	0.8000	0.9000	1.0000
0.025	0.0256	0.0102	0.0051	0.0056	0.0096	0.0138	0.0262	0.0309
0.050	0.0236	0.0101	0.0046	0.0033	0.0021	0.0044	0.0086	0.0115
0.075	0.0145	0.0090	0.0055	0.0035	0.0027	0.0025	0.0039	0.0083
0.100	0.0131	0.0079	0.0051	0.0034	0.0025	0.0026	0.0035	0.0061
0.125	0.0129	0.0076	0.0049	0.0034	0.0027	0.0029	0.0037	0.0069
0.150	0.0122	0.0072	0.0049	0.0034	0.0027	0.0034	0.0038	0.0049
0.175	0.0119	0.0071	0.0048	0.0035	0.0029	0.0038	0.0053	0.0078
0.200	0.0100	0.0065	0.0047	0.0033	0.0032	0.0035	0.0036	0.0094
0.225	0.0092	0.0060	0.0044	0.0032	0.0032	0.0034	0.0062	0.0113
0.250	0.0084	0.0053	0.0040	0.0032	0.0041	0.0036	0.0054	0.0096
0.275	0.0084	0.0051	0.0039	0.0033	0.0028	0.0064	0.0074	0.0100
0.300	0.0079	0.0047	0.0036	0.0031	0.0040	0.0057	0.0072	0.0088

Table 4.2: Average RMS error for variation of both σ_1 and σ_2 over 50 realizations

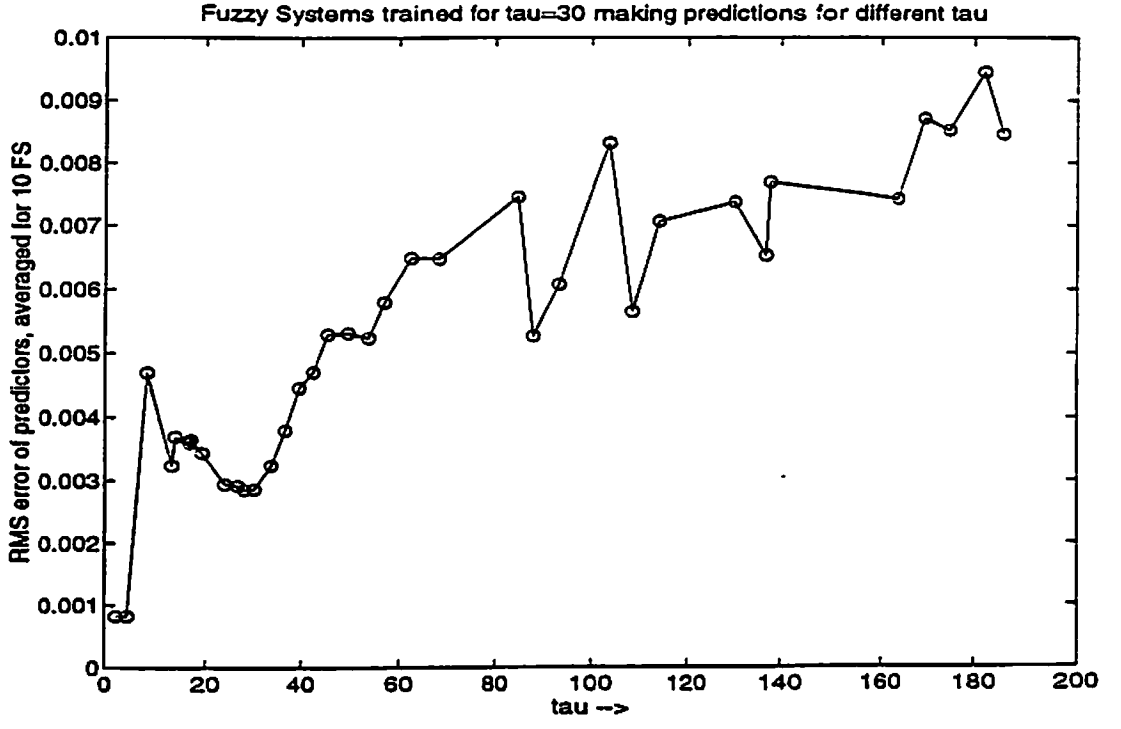

Figure 4.16: Application on different τ

Chapter 5

Conclusions

We have shown how to modify the OLS FLS training algorithm presented in [7], so that a substantial saving in running time is achieved. Our two-pass OLS method lets us use different training segments for each pass, leading to better generalization capabilities. The process of choosing an optimal set of rules is now made independent from the computation of the output centroids, giving more design flexibility. Our application of the two-pass OLS-trained FLS to prediction of the Mackey-Glass chaotic time series gives excellent results, even for changed system dynamics. Finally, we advocate comparing all new ways (FLS's, feedforward neural networks, radial basis function networks, etc.) to predict time series against the *hold method*.

Bibliography

[1] A. Björck, "Solving linear least squares problems by Gram-Schmidt orthogonalization", *Nordirsk Tidskrift Informations-Behandling* vol. 7, pp. 1-12, 1967

[2] S. Chen, C. F. N. Cowan and P. M. Grant "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks" *IEEE Transactions on Neural Networks* vol. 2, no. 2, pp. 302-309, March 1991

[3] J. D. Farmer "Chaotic Attractors of an Infinite-Dimensional Dynamical System" *Physica 4D* 1982 pp. 366-393

[4] J. Hohensohn and J. M. Mendel "Two-Pass Orthogonal Least Squares Algorithm to Train and Reduce Fuzzy Logic Systems" *Proceedings of the 3rd IEEE International Conference on Fuzzy Systems* pp. 696-700

[5] A. Lapedes and R. Farber "Nonlinear Signal Processing using Neural Networks: Prediction and System Modelling" *Los Alamos Report LA-UR-87-2662* July 1987

[6] M. C. Mackey and L. Glass "Oscillation and Chaos in Physiological Control Systems" *Science 197* 1977 pp. 287-289

[7] L.-X. Wang and J. M. Mendel "Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least-Squares Learning" *IEEE Transactions on Neural Networks* vol. 3, no. 5, pp. 807-813 September 1992