Neural nets in an acoustic application

Moonen, F.J.

Award date:
1992

6009

EINDHOVEN UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING
Measurement and Control

NEURAL NETS IN AN ACOUSTIC APPLICATION

By F.J. Moonen

M.Sc. Thesis
Carried out from :April 1992 to December 1992.
Commissioned by :prof.dr.ir. A.C.P.M. Backx
Under supervision of :dr.ir. A.A.H. Damen
dr. S. Weiland
Date :10 December 1992

ABSTRACT

For reproduction of low frequent sound, loudspeakers with large physical dimensions are necessary. The voluminous boxes can be avoided if existing panels, excited with a small transducer, are used for the reproduction of the bass. Such systems however, produce a lot of linear as well as nonlinear distortion.
One way to overcome this problem is using an active controller which can compensate for these distortions.

Controller design will be based on an analytic model of the process and this modelling covers the main part of this report. The process which has to be modelled is a prototype transducer, designed to produce a lot of nonlinear distortion. The transducer has been considered as a siso process. Simo identification has not been included.

Two ARMA models have been obtained (an ARMA model is a linear model), using two completely different identification methods.

To model nonlinear behaviour, a multi layered neural net has been used. Several methods, in equation error configuration as well as output error configuration, have been applied to tune the parameters of the neural net. Validation of the obtained nonlinear model, in the time domain as well as in the frequency domain, has been carried out.

It turned out that the algorithm, (static) backpropagation, commonly used for tuning the parameters of the neural net, is not satisfying if a badly damped system has to be modelled in output error configuration.

An other algorithm, dynamic backpropagation, has been implemented and has been applied. This algorithm performs much better. It turned out however, that a combination of a steepest descent as well as a line search algorithm, has to be applied in order to obtain a good model.

Other important conclusions:

- A linear model performs quite well if the acoustic process has to be simulated.
- Improvement can be obtained, if an artificial neural net is applied.
- A lot of data samples are necessary, in order to tune the parameters of the neural net in order to identify a badly damped system (for the acoustic process, 5000 samples have been used).
- The nonlinear model performs well in the time domain as well as in the frequency domain.
- A lot of parameters in the net are redundant. To increase the convergence rate it is recommendable to determine the redundant parameters of the net.

CONTENTS

1 INTRODUCTION

Problem

For reproduction of sound, transducers with a large dynamic range are needed. Today most loudspeakers meet this requirement to a great extend. For the reproduction of bass however, there are two topics which can be improved. In general an acoustic system will have large physical dimensions and the distortion at low frequencies is relatively high. One possibility to solve the first problem, is placing a small transducer on any existing panel. This transducer is built up from a coil and a free moving magnet. The magnet and the panel can be moved by activating the coil. The main advantage of this kind of system is the use of a existing large panel, which avoids the use of another voluminous box.
The most important disadvantage of such a system is the production of linear as well as nonlinear distortion.

Objective

The objective is to design an active controller which can compensate for the distortions. If such a controller can be designed, the two most important problems concerning loudspeaker systems, are solved (large dimensions and relatively high distortion at low frequencies).
The controller design will be based on a model. In this report, attention will be mainly focused on how to obtain a good model for controller design.

Procedure

The problem has been tackled in two ways.
First a linear model will be obtained. Based on this model a linear controller can be designed. This controller can compensate for linear distortions.
Secondly a nonlinear model has been obtained. There are a lot of model sets which can be used to model nonlinear behaviour. In this report an artificial neural net has been used to model the process. Based on this model a nonlinear controller can be obtained (this part has not been included in this report).

2 DEFINITIONS

Identification

First definitions are given, used in the rest of this report. At the first stage we want to obtain a good model which can be used for controller design. Before we can start identifying a model, we have to define what we mean by "good". In other words we need to specify the criteria that we want to optimize.
A model is a good representation of the process if this model can approximate the process output only using the process inputs. We call the model optimal if the sum squared error between the process output and the model output, generated using validation data, is approximately the same as the noise power from the obtained process data. The performance of a model is therefore defined as:

$$J = \frac{\sum_{k=1}^{N} \left[y_{model}(k) - y_{process}(k) \right]^2}{\sum_{k=1}^{N} \left[y_{process}(k) \right]^2} \qquad [2A]$$

During optimization of the model parameters the loss function V is minimized. V is defined as:

$$V = \frac{1}{N} \sum_{k=1}^{N} \left[y_{model}(k) - y_{process}(k) \right]^2 \qquad [2B]$$

N = *number of samples used for estimation.*
k = *sample moment.*

The signals $y_{process}$ and y_{model} have been depicted in the next picture:

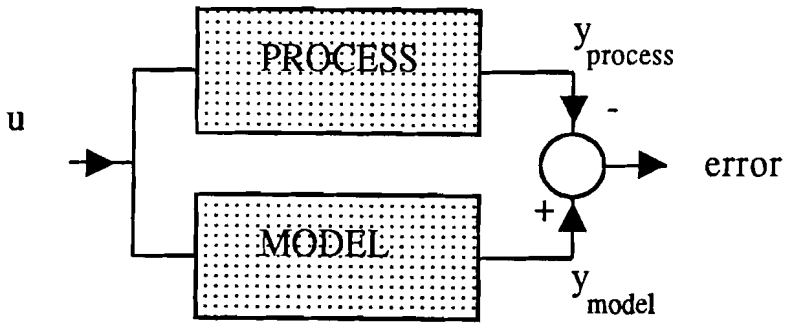


fig. 2.1 definition of performance

Model identification is split up in two parts; estimation and validation. The data set used for the validation is not used for estimation and vice versa. Both validation and estimation can be performed in two ways. These methods are defined as follows:

Estimation : - Prediction Error Method (PEM).
- Output Error Method (OEM).

Validation : - Prediction properties
- Simulation properties

For the prediction error method the model output at sample k is calculated using the process input at sample k, plus previous process inputs and previous *process* outputs. For the output error method (simulation) the model output at sample k is calculated using the process input at sample k plus previous process inputs and previous *model* outputs. At first instance it seems that the difference can be neglected, but the performance of the models, obtained from the different methods can differ significantly!
The picture below elucidates the different configurations:

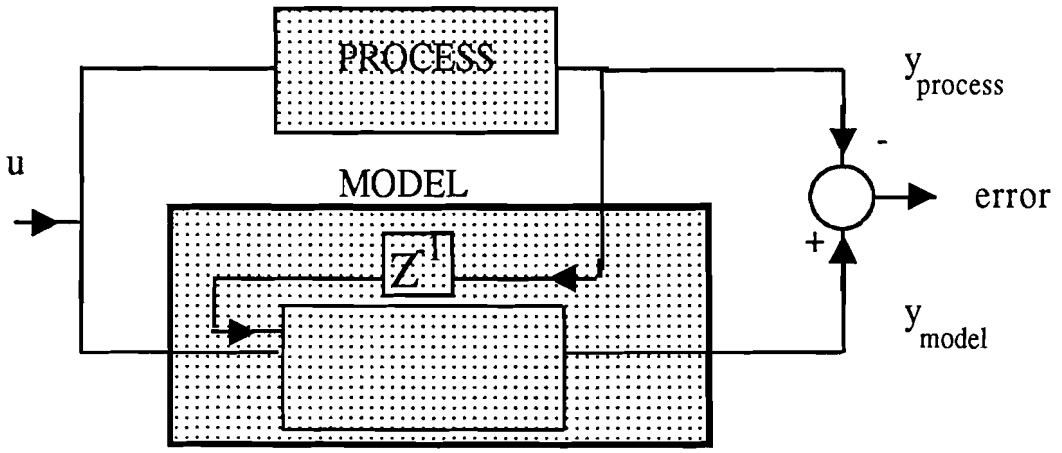


fig 2.2 configuration used for PEM

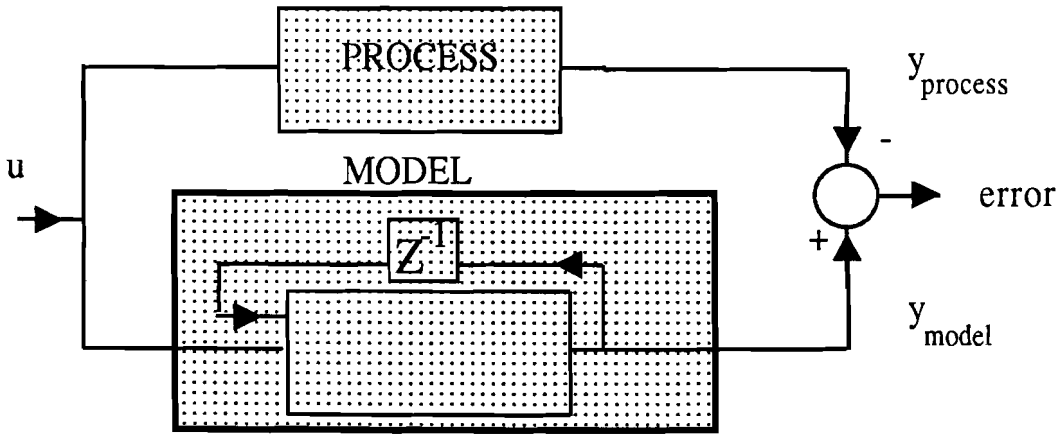


fig 2.3 configuration used for OEM

Controller design will be based on the estimated model only (Process data is not used at this stage). Therefore only simulation is an useful criterium for the purpose of validation of the obtained model. So to get good results one has to use the output error method to optimize model parameters. The advantage of PEM, however, is the fact that the optimal solution is easy to find. (This is the reason why a lot of publications, concerning neural nets, are based on PEM only). In the case of noiseless model identification and if the process can be fitted exactly by the model, PEM will indeed suffice (if numerical errors can be neglected). For real processes, however, it will be shown that OEM is of more importance. In this report examples are included to emphasize this.

Control

A transducer does not have linear distortion if all the frequencies in the frequency band of interest are equally amplified. No nonlinear distortion occurs, if the transducer, excited with only one frequency component, does not generate other frequencies in the output

signal.
The performance of the controller is related to the ability to suppress linear as well as nonlinear distortion. This performance is defined as :

$$J_c = \frac{\sum_{k=1}^{N}\left[y_{desired}(k) - y_{system}(k)\right]^2}{\sum_{k=1}^{N}\left[y_{desired}(k)\right]^2} \qquad [2C]$$

During optimization the loss function V_c is minimized. V_c is defined as:

$$V_c = \frac{1}{N}\sum_{k=1}^{N}\left[y_{system} - y_{desired}\right]^2 \qquad [2D]$$

The output variables, $y_{desired}$ and y_{system} are defined in the next picture.

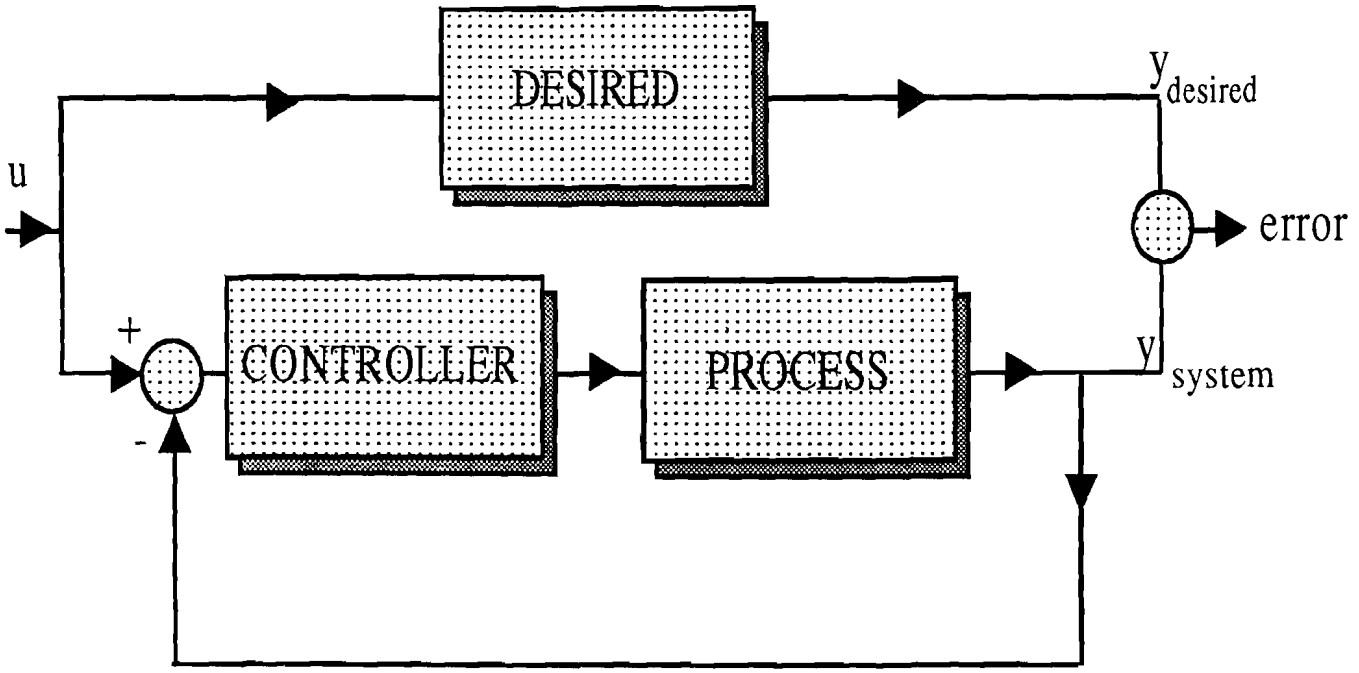


fig 2.3 controller configuration

3 SET UP

Proces description

The process, described by (Velden[19]) is an acoustic transducer with only one input and three outputs.
The input is the voltage, U_{in}, applied to the coil
The outputs are:
- electrical current through the coil of the transducer
- the acceleration of the board
- sound measured at one meter away from the board

later on refered to as the current, accelaration and sound.

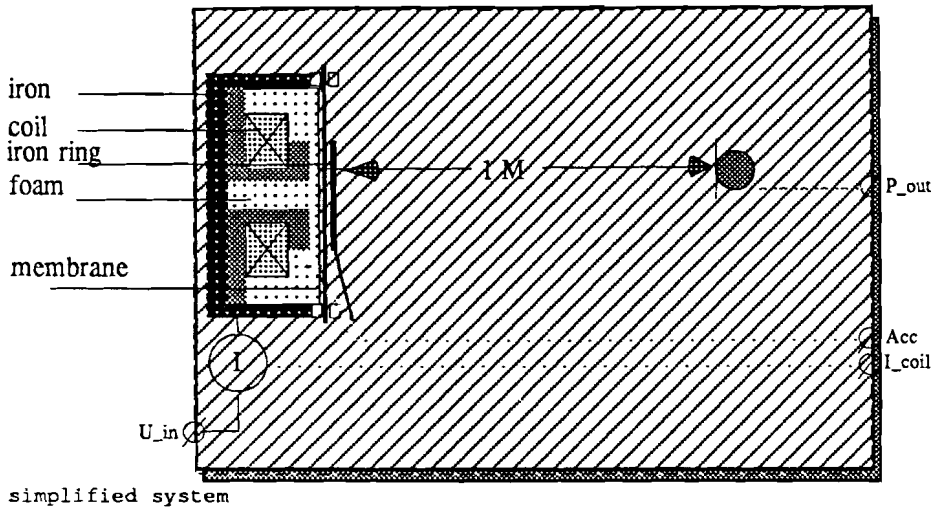


fig. 3.1 simplified system

The objective is letting the measured output at one meter track the input U_{in}. Hoping the current and the acceleration supply enough information concerning the sound at one meter, we can use these two process outputs as inputs for the feedback controller (final configuration has been given in a following section).
The analysis shows that the current and the sound are hardly correlated. Therefore, it was proposed to use only the acceleration for the input of the controller (The sound itself can only be used to control the lowest frequencies, due to the inherent delay time). Because the sound contains less higher harmonics than the acceleration, one can assume that the sound contains no higher harmonics if the accelaration is controlled optimal. Therefore, in this report only a siso system has been considered with input U_{in} (the applied voltage) and output the measured acceleration.

Data acquisition

Preliminary measurements

Using a sweep signal the frequency band of interest as well as the higher harmonics can be obtained, as has been shown in fig 3.2a

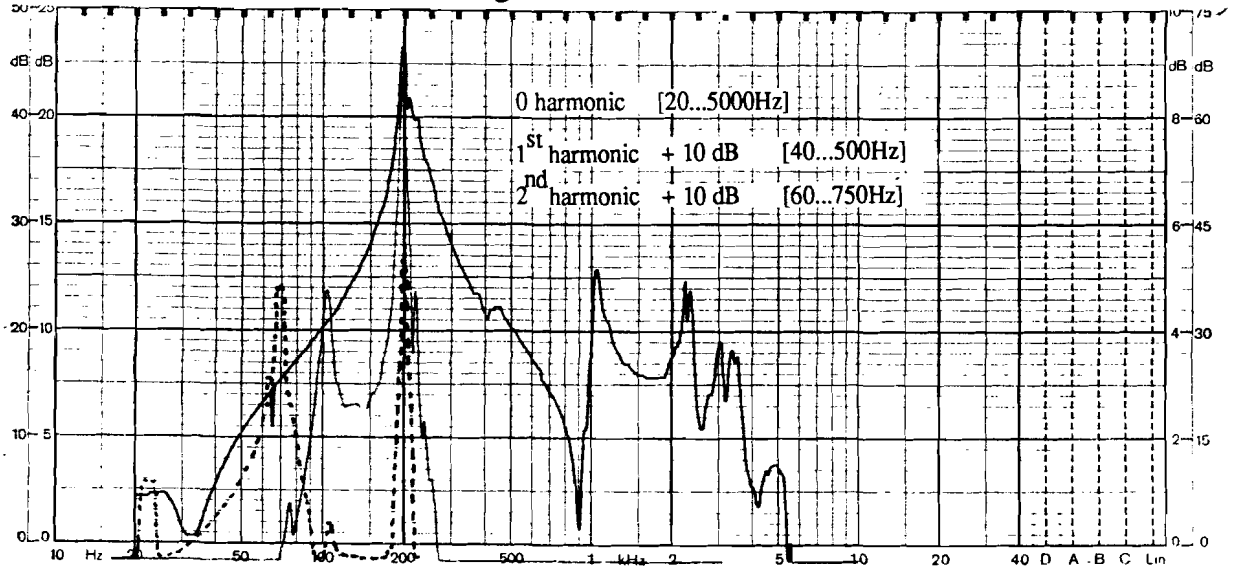


fig. 3.2a Preliminary measurements of the acceleration

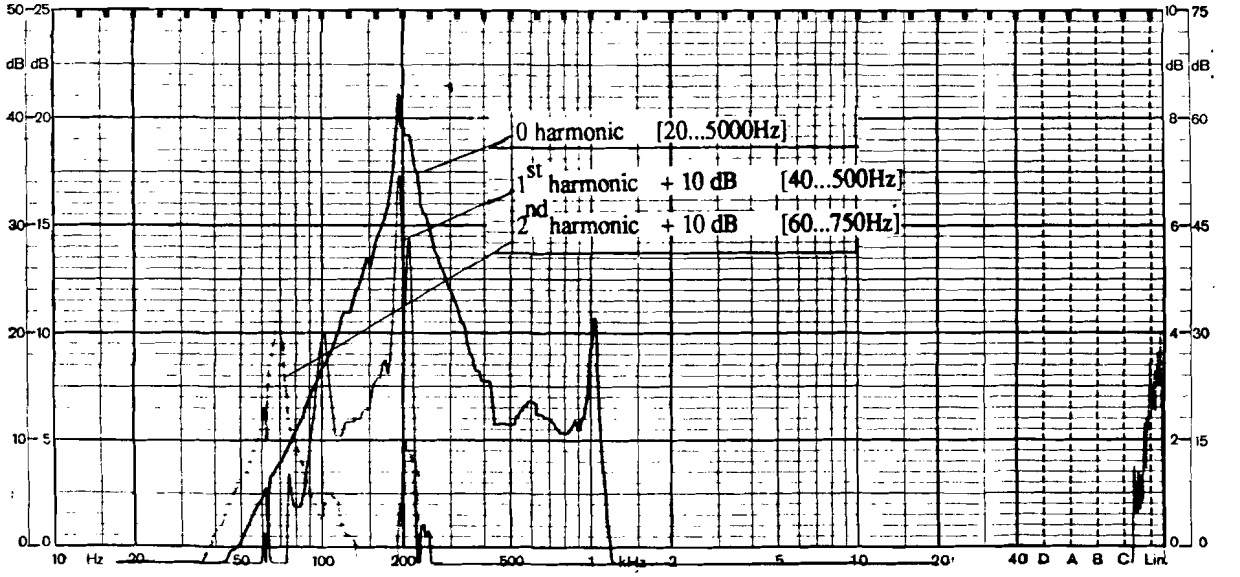
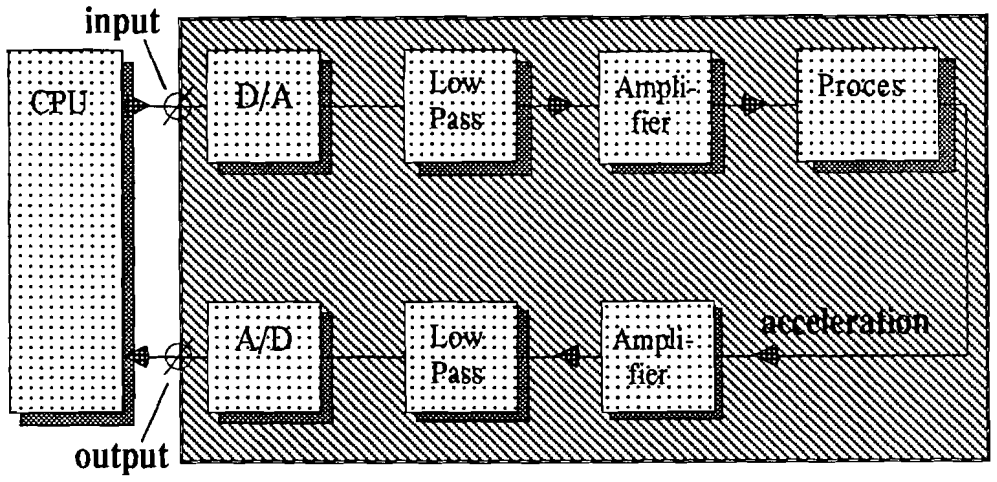


fig. 3.2b Preliminary measurements of the sound

Because the sound contains no frequencies beyond 1200 Hz, the bandwidth of interest has been choosen to be 1200 Hz. According to this a sampling frequency of 3000 Hz has been chosen.

To prevent aliasing, two filters have been included (bandwidth:1200 Hz). For data acquisition the overall system, together with the sample and hold circuits, has been depicted in figure 3.3.



The data sequence, used to excite the system, is generated by the cpu. The response has been saved. It turned out that apart from scaling, offset- and delay correction no further data processing was necessary.

Data preprocessing

To gain insight in the system, some correlations have been calculated.

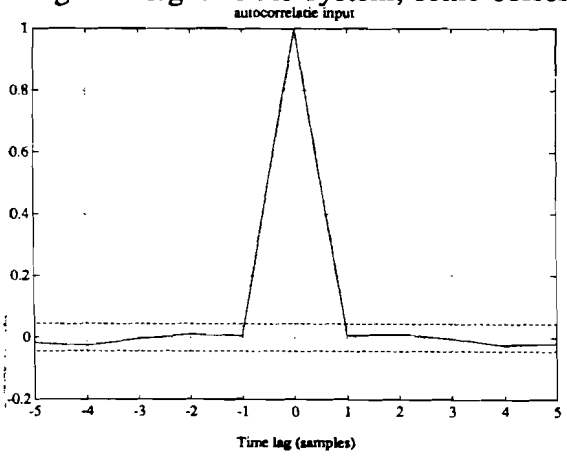


fig. 3.4 autocorrelation of input

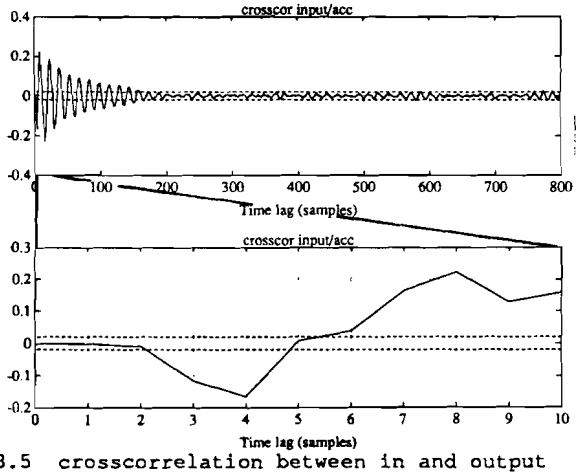


fig. 3.5 crosscorrelation between in and output

From fig. 3.4. it is evident that the input is a white noise sequence. Because the input is white the cross correlation between input and output is an approximation for the impulse response. From fig. 3.5 two conclusions can be drawn.

- The delay between input and the acceleration is 2 samples.
- The system is badly damped. The impulse length, using a sampling frequency of 3000 Hz is 200 samples.

The first conclusion can be used to apply delay correction. The second conclusion can be used to stipulate the number of samples used for the estimation. Usually one takes a data set of about ten times the impulse length. In this case 10 * 200 = 2000 samples should be preferable. The obtained data set consists of 10.000 pairs of input and output samples. For model estimation this is amply sufficient.
Another 2000 samples will be used for validation.

The input u, as well as the output y (=acceleration), are offset corrected and scaled such that the adjusted input output pairs $(u_s(k),y_s(k))$ satisfy:

$$|u_s(k)| \leq 1 \quad \forall\, k \in \{1..N\}$$
$$|y_s(k)| \leq 1 \quad \forall\, k \in \{1..N\}$$
$$\sum_{k=1}^{N} u_s(k) = 0$$
$$\sum_{k=1}^{N} y_s(k) = 0$$

$u_s(k)$ = *input after scaling*
$y_s(k)$ = *output: (acceleration) after scaling*
k = *sample time*
N = *length of data set used for estimation*

this is achieved by putting:

$$u_s(k) := \frac{u(k) - \overline{u}}{|u(k)|}$$
$$y_s(k) := \frac{y(k) - \overline{y}}{|y(k)|}$$

where:

$$\overline{u} = \frac{1}{N}\sum_{k=1}^{N} u(k)$$
$$\overline{y} = \frac{1}{N}\sum_{k=1}^{N} y(k)$$

In the remainder of this report $u_s(k)$ and $y_s(k)$ are for simplicity denoted as u(k) and y(k).

Choosing the configuration

Using the configuration depicted in fig. 3.3 (i.e. using the data set "input" and "output" in fig. 3.3) the process with the filters (= DA convertor, amplifier, and low pass filter) will be identified. If only the process has to be identified it is better to use the configuration depicted in fig. 3.6.

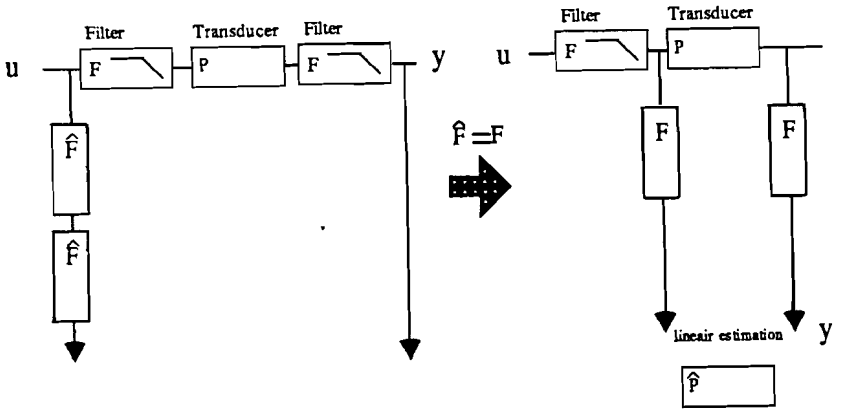


fig. 3.6 Alternative configuration used for identification

The filter dynamics can be identified by short circuiting the process in fig. (3.3.)This identification can be done quite accurately because hardly any disturbance is involved.

By filtering the input data with the estimated filter parameters the obtained model is an estimation of the process only, but weighted in frequency domain by the filter F. Estimation of only the process dynamics is usefull if an analogue controller has to be designed (this controller is implemented without the AD and DA convertors)
The model obtained using this method is only useful if an analogous controller has to be implemented. In the stage of development a digital controller will be designed, and implemented in the CPU. In a digital environment the filters are necessary, therefore the filters have to be estimated as well. So prefiltering of input data will not be used at this stage. The signals "input" and "output" depicted in fig 3.3 will be used for model estimation.
To indicate that the filter dynamics indeed can be estimated quite accurately, a simulation of the estimated filter has been included (see fig 3.7). An eight order has been used to estimate the filter dynamics. A uniformly distributed white noise sequence has been used to create a estimation set. The model has been identified using the function 'OE' (an OEM optimization) of the identification toolbox of MATLAB. The performance turned out to be 2%.

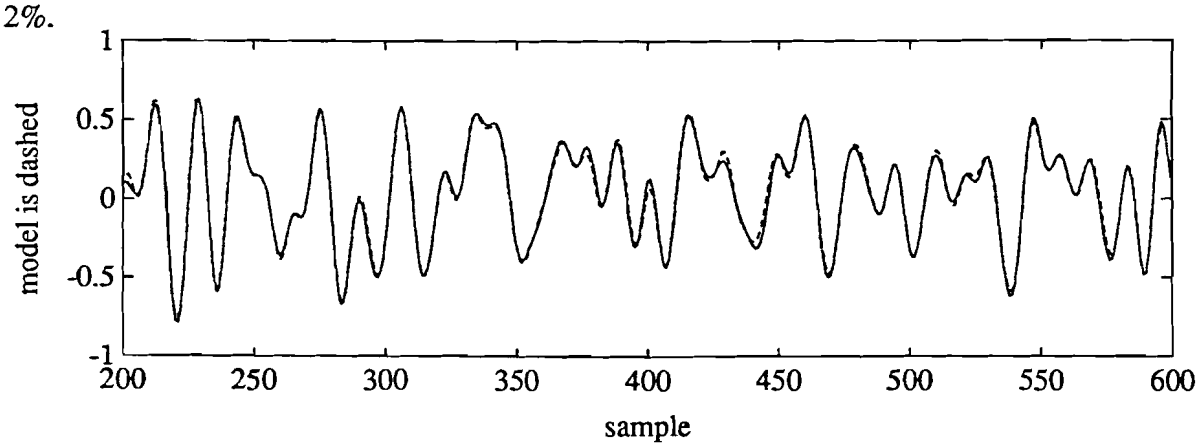


fig. 3.7 Simulation of estimated filter. Solid line is the real filter output.

4 LINEAR ESTIMATION

Two methods have been applied to obtain a linear model. The first method starts with a high order FIR (Finite Impulse Response) model. The output of a FIR model can be written as:

$$\hat{y}(k) = b_0u(k) + b_1u(k-1) + \ldots + b_fu(k-m).$$

where f : number of FIR parameters.

The advantage is that the optimal parameters of the FIR model, using OEM, can be calculated analytically. The disadvantage is that this model contains a lot of parameters. The second method starts with a high order ARMA model, which is optimized using PEM. The model output of an ARMA model can be written as:

$$\hat{y}(k) = b_0u(k) + b_1u(k-1) + \ldots + b_mu(k-m) - a_1y(k-1) - a_2y(k-2) - \ldots - a_ny(k-n).$$

where $m<=n$.

b_i : Moving Average parameters

a_i : Auto Regressive parameters

The advantage is that less parameters are needed for this initial estimate. The disadvantage is that this model is obtained using PEM.

After a brief description a comparison between two methods has been given.
It is not said that one method is superior. Just two completely different methods has been applied.

First method: optimization in time domain

First a high order finite impulse model is estimated. The optimal solution can be calculated analytically. After obtaining these parameters model reduction is applied using Hankel singular value decomposition. The reduced order model, obtained after the model reduction, can be used as an initial estimate for the direct estimation, an OEM identification. In the last step of the algorithm the real data set is used again. The method, described in (Backx[3]), has been depicted in the next figure.

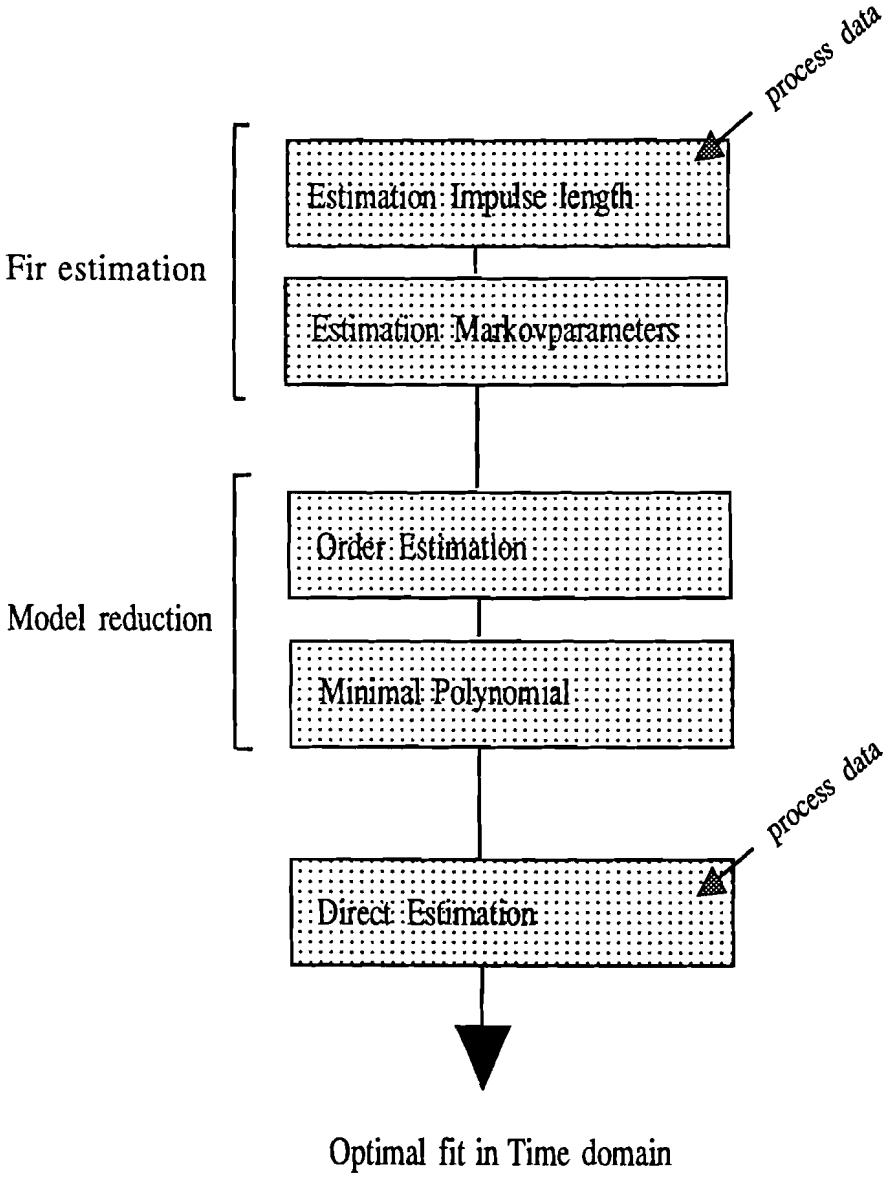


fig. 4.1 Optimization in time domain

In this case, estimation of a badly damped system, problems can occur if not enough memory space is available. Calculating a high order FIR model (=estimation of MARKOV parameters in fig 4.1), a huge matrix has to be filled.
The size of this matrix is N * n. Where:

N = number of samples used for estimation
n = length of impulse response estimated from cross correlation.

It turned out that the impulse response length has to be decreased. One method to do this is by prestabilization of the process using constant gain feedback of the acceleration. This method has not been applied, just the suggestion is made here. To get some idea what constant gain feed back of the acceleration will bring about, an approximation of the main process dynamics has been given.
A DC input does not give any responce (no net force on the membrane, no acceleration, so no output) therefore a zero in z=1. The membrane in combination with the coil will behave as a damped mass spring system. This can reperesented by a complex pole pair.

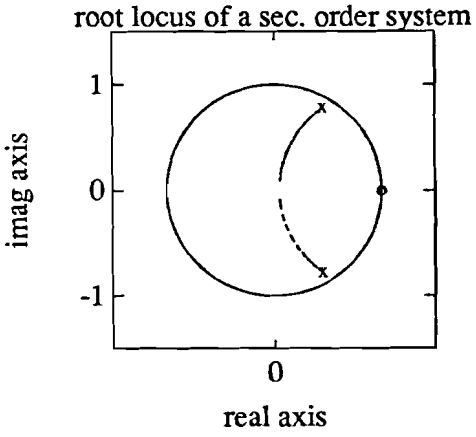


fig 4.2 Rootlocus of a second order model

The impulse length can easily be reduced by simple constant gain feedback of the acceleration. The rootlocus of this simplified system has been plotted in fig 4.2
Because at this point no model is available, a suitable feedback gain for the real system can only be determined by trial and error. This second order approach is just useful to get some idea what constant feedback would bring about for this system (At the end of this chapter the rootlocus of the obtained model has been included).

In this case another method has been applied to decrease the impulse length. Reduction has been obtained after downsampling. Of course the data has to be filtered to prevent aliasing effects. In this case a fourth order low pass Butterworth filter, with bandwidth equal to 400 Hz, has been used. Caused by this filtering, the sampling frequency can be decreased from 3000 Hz to 1000 Hz. Due to the data reduction the impulse length has been decreased to 60 samples. In this set up, a good 3th order model, which performes good on the reduced dataset, has been obtained (using the method depicted in fig 4.1).

Because frequencies beyond 400 Hz are quenched, the model is not optimal with respect to the original data. This model can be improved, performing two steps. First, to converse the obtained model, based on a sampling frequency of 1000 Hz to another model with the same impulse response based on a sampling frequency of 3000 Hz. (See appendix 1)

Second, high frequent dynamics can be estimated, using the frequency plot in fig 3.2. As can be seen from this plot the system has another resonance frequency, located at 1000 Hz. This can be represented by an extra complex pole pair. The output for 900 Hz is quenched. This can be represented by a complex zero pair. By adding this pole- and zero pair, like has been depicted in fig. 4.3, a better model can be obtained.

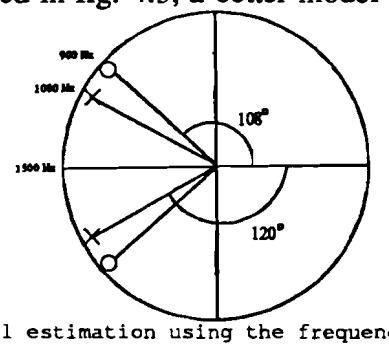


fig. 4.3 Obtaining initial estimation using the frequency plot

This is just a method to obtain an initial estimate which can be used to start the procedure for computing the optimal parmaters. (Using OEM) After performing direct estimation (see fig 4.1) a model of moderate quality, based on the original data set, has been obtained. The procedure to obtain the optimal parameters has been described in Backx[3]. A simulation of the final 5th order model output, together with the real output has been plotted in the next picture (3th order model obtained after data reduction, a 5th order model after adding two poles).

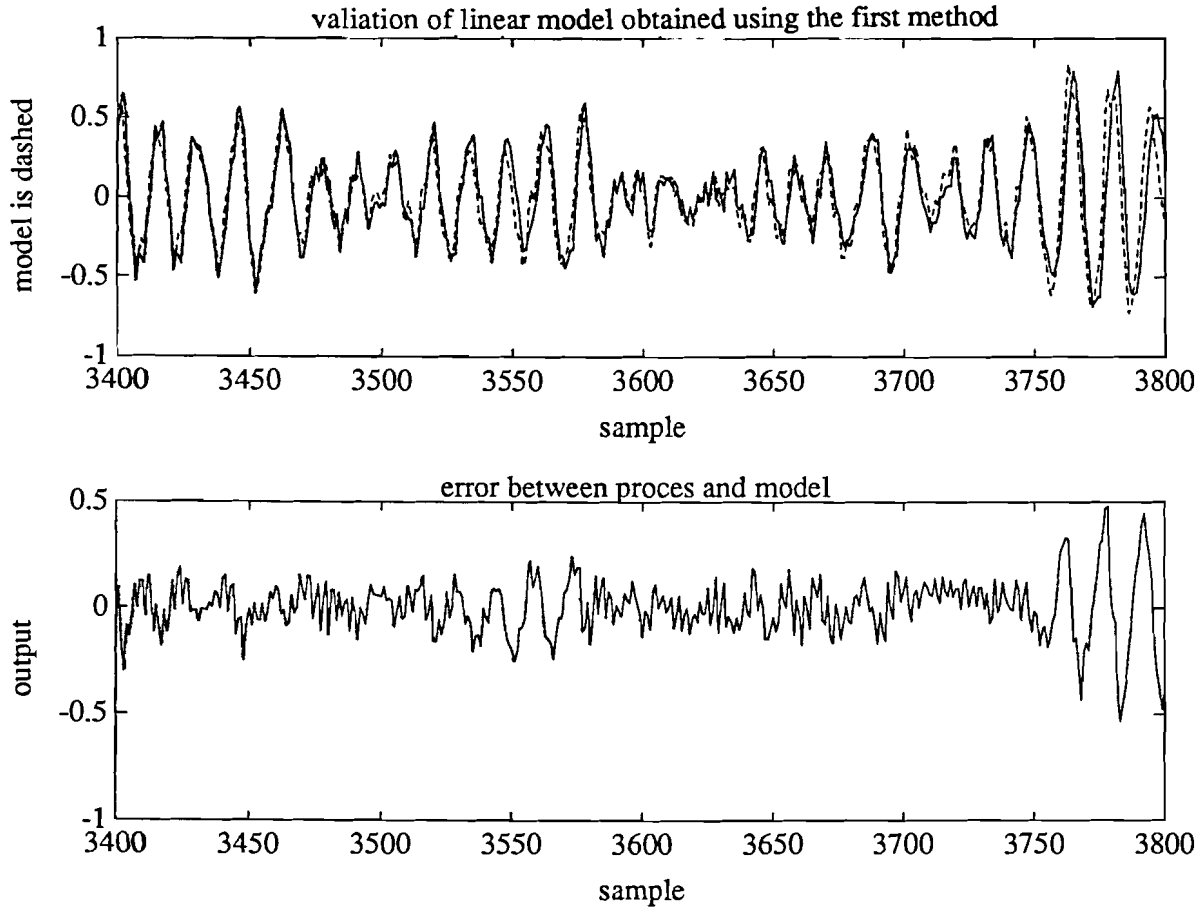


fig 4.4 Simulation of model obtained using method 1

In case of hardware limitations a tricky solution as reported above and in order to identify a badly damped system, has been applied. For such systems it is better to use the second method, which is independent of the impulse length and which is described next.

Second method

The method is proposed by Zhu[24].
As by impulse response estimation first a high order model using PEM is obtained. As opposed to the first method, however, the parameters are not used as an initial estimate for OEM, but just used to filter the data. In the last step an output error alike method is applied, starting with random initial parameter values. To get some insight in the method a brief description has been included.

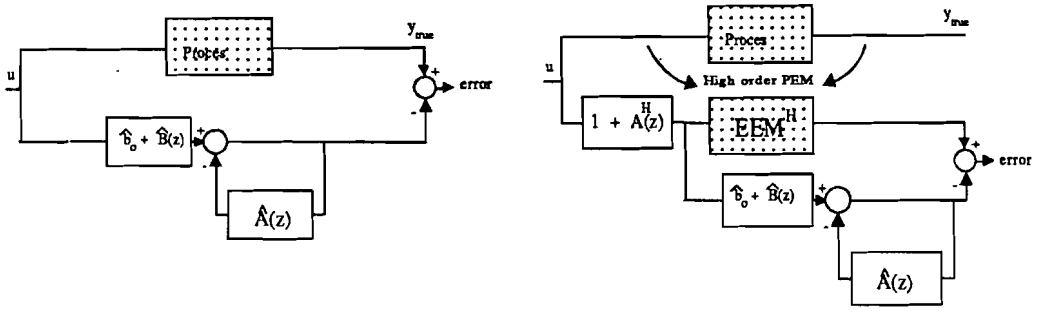


fig 4.5 OEM with and without prefiltering

$$
\begin{aligned}
H &= \textit{order of the high order model} \\
A^H &= \textit{estimated high order AR parameters} \\
b_o^H + B^H &= \textit{estimated high order MA parameters} \\
\hat{A} &= \textit{estimated low order AR parameters} \\
\hat{b}_o + \hat{B} &= \textit{estimated low order MA parameters} \\
EEM^H &= \frac{b_0^H + B^H(z)}{A^H(z)}
\end{aligned}
$$

At first a high order ARMA model is estimated, using PEM. These parameters are used to create a new data set. Based on this set a low order model is estimated.

Applying OEM (see fig. 4.5) the loss function [4A] is minimized. Applying the method proposed by Zhu [24] the loss function [4B] is minimized.

$G_t(e^{j\omega})$ = *transfer of real process.*
$G^H(e^{j\omega})$ = *transfer of high order model, obtained using PEM.*
$\hat{G}(e^{j\omega})$ = *transfer of final model, obtained after "OEM".*
$\phi_{uu}(\omega)$ = *spectral power density of input.*
V1 = *Loss function optimized using the first method.*
V2 = *Loss function optimized using the second method.*

$$V1 = \frac{1}{\pi}\int_{-\pi}^{\pi} | G_t(e^{j\omega}) - \hat{G}(e^{-j\omega}) | \phi_{uu}(\omega)\, d\omega \qquad [4A]$$

$$V2 = \frac{1}{\pi}\int_{-\pi}^{\pi} | G^H(e^{j\omega}) - \hat{G}(e^{j\omega}) | * \phi_{uu}(\omega) | A^H(e^{j\omega}) + 1 |^2 d\omega \qquad [4B]$$

$| A^H(e^{j\omega}) + 1 |^2$ = *weighting of loss function*

Both criteria are minimized using the function 'OEM' of the identification toolbox of MATLAB.
It has been proven in Zhu[24] that under certain assumptions (a.o. $\Phi_{uu}(\omega)$ is constant) a high order ARMA model converges to the real transfer function as:

$N \rightarrow \infty$
$H \rightarrow \infty$
$H \ll N$

where N : the number of data samples used for estimation.
H : the order of the high order model obtained using PEM.

The errors of the estimated transfer function have an asymptotical normal distribution. The variance of the errors at a given frequency is proportional to the signal to noise ratio at that frequency. The asymptotic variance is proportional to $\frac{1}{1 + A^H(e^{j\omega})}$.

By filtering conform configuration depicted in fig 4.5 the loss function is weighted by $1 + A^H(e^{j\omega})$. This is proportional to the inverse of the asymptotic variance. Therefore a minimum variance estimation is obtained.

To explain the advantage of optimizing the weighted loss function V2 instead of V1, some frequency plots have been included (see fig 4.6). In the first plot the spectrum of $G^H(e^{j\omega})$ - $G_t(e^{j\omega})$ has been plotted. In the second plot the spectrum of the weighting filter A^H has been plotted. As can be seen the frequencies where the high order has small variance (error) will have large weight in the estimation criteria. Therefore a minimum variance estimation can be obtained.

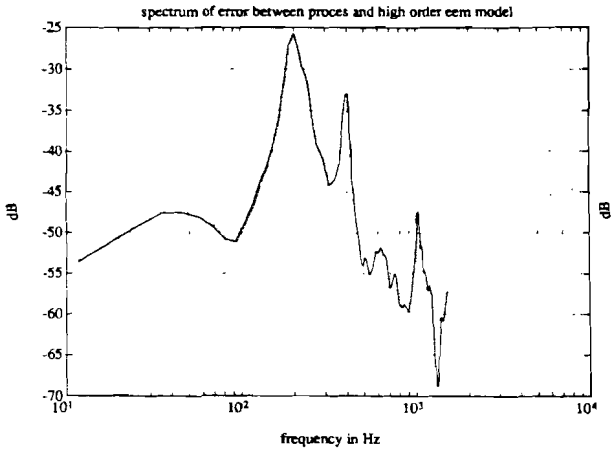


fig 4.6a Spectrum of error

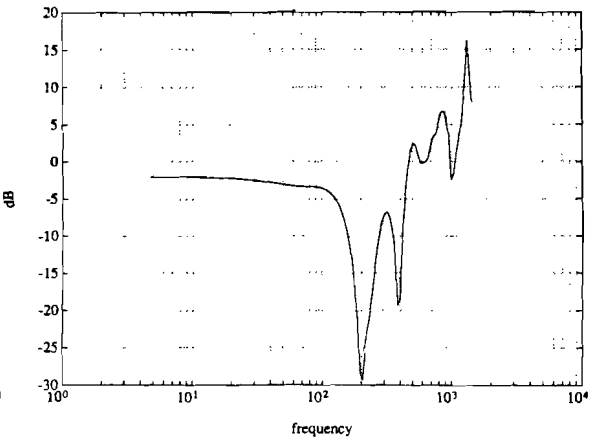
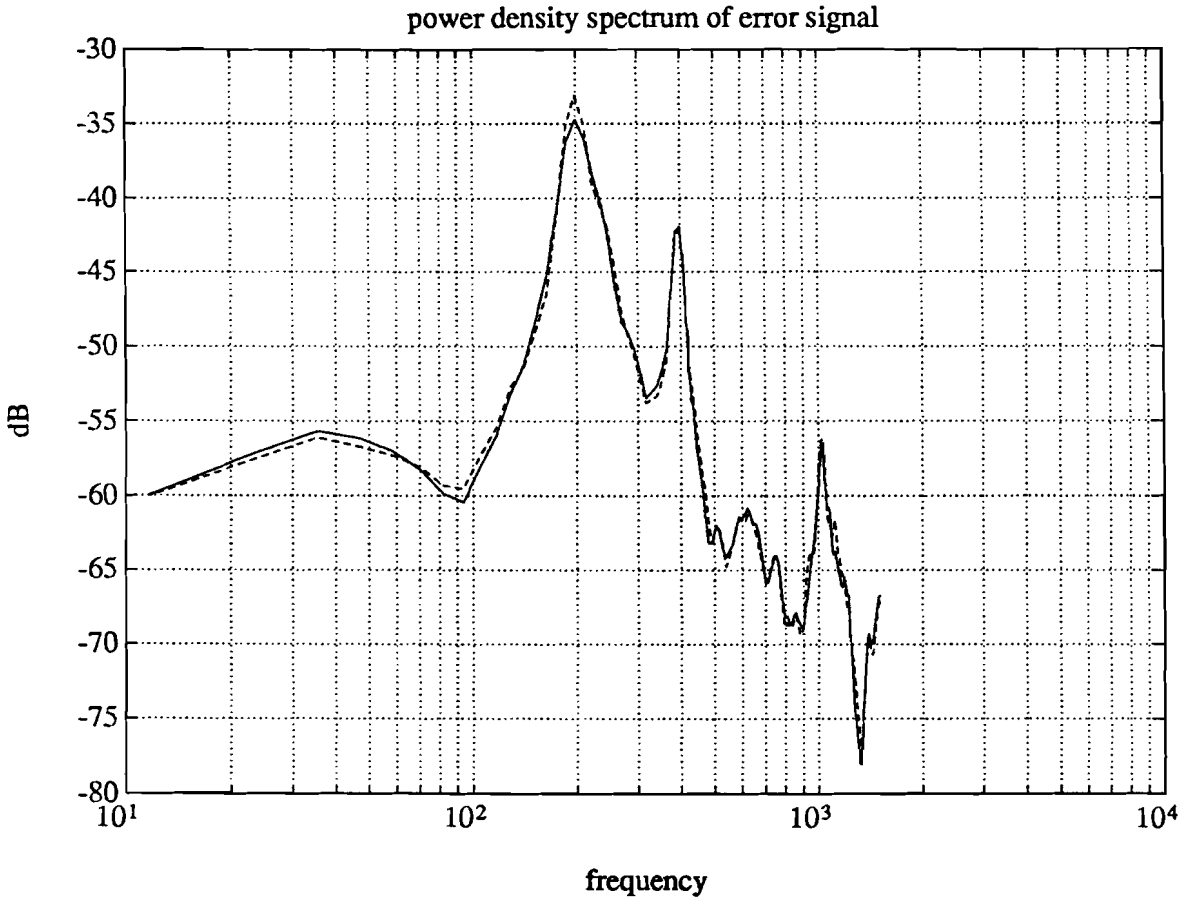


fig 4.6b Spectrum of $1+A^H$

In the next picture the spectral power density of the error between the high order model and the real process output (solid line) and the error between the low order method and the real process output (dashed) has been plotted. According to [4B] the error of the final low order model will be smaller than the error of the high order model. In this particular case, this does not hold. Probably this is due to the fact that the mismatch is caused by non linearities, not by additive disturbances on the output. The proof in (Zhu[24]) is only valid for linear system identification, therefore no improvement with respect to the high order model can be obtained.



dashed: error between real process and 25th order model
solid: error between real proces and 8th order model
fig 4.7

Comparison of two methods.

The most significant difference between the methods occurs in the last step of the algorithm, performing OEM. In the first method the optimal FIR parameters are estimated. as an initial estimation for the OEM, in this step the real process data is used. In the second method PEM is used to design a filter, which is used to create a data set. using this data set an output error alike method is applied. The initial parameters for this estimation are chosen randomly.

In this application the second method turned out to perform better (performance J obtained using the first method is 9 %, performance of the model obtained using the second method is 4.8%). This is probably caused by hardware limitations, occuring using the first method. Having more memory space available, it is to be expected that the model obtained using the first method can be improved. Further research in the difference between the two methods is beyond the scope of this report. Just two methods are proposed to get a linear estimation of a process.

Order estimation

The order has been estimated by applying the second method for several orders. The performance for different models has been depicted in next picture.

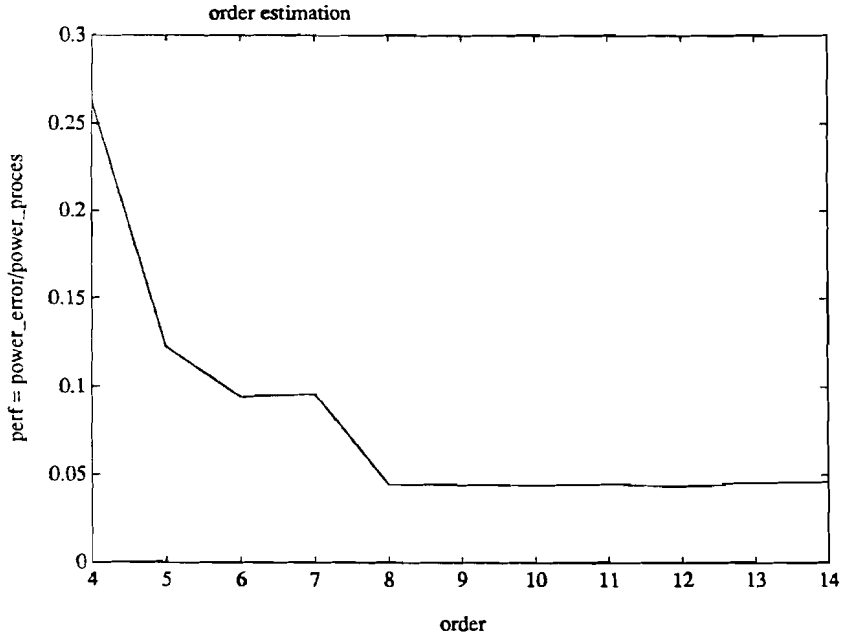


fig 4.8 Performance for different model orders.

Taking the model order to be higher than 8, no significant improvement can be obtained. Therefore an eight order linear model has been taken for the linear simulation of the process.

Results of the linear estimation

Both configurations as depicted in fig. 3.3 and in fig. 3.6 has been used for model estimation, using the second method. As can be seen in fig. 4.9 the spectrum of the estimated model, obtained using the first configuration (fig. 3.3) shows resemblance with the spectrum obtained after the preliminary measurements. This is to be expected, because the preliminary measurements have been executed without using the filters (analogue measurements). So the transfer of only the process is obtained during these measurements. By prefiltering the input data with the estimated filter parameters, only the approximate dynamics of the process will be identified.

The estimated model transfer, obtained using configuration 3.6, is comparable with the power spectrum density of the sampled output signal (input signal is white).

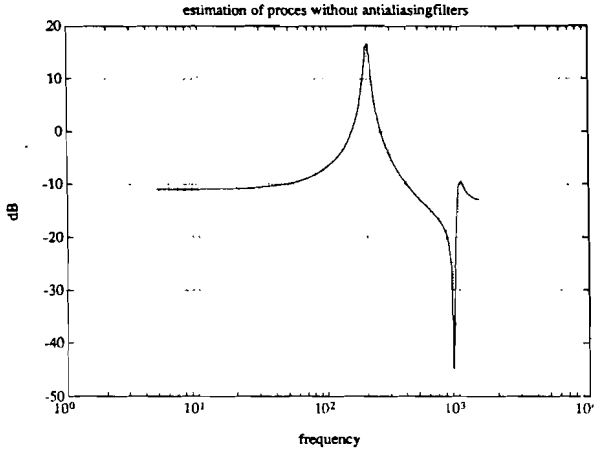


fig 4.9a Estimation of process

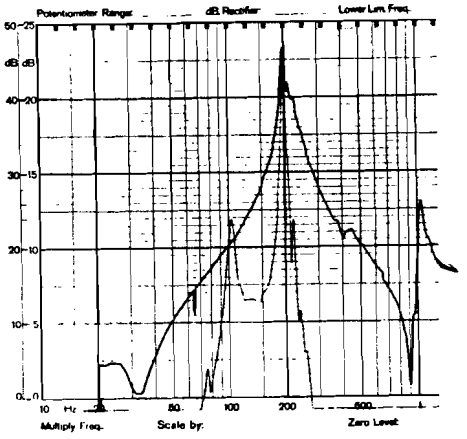
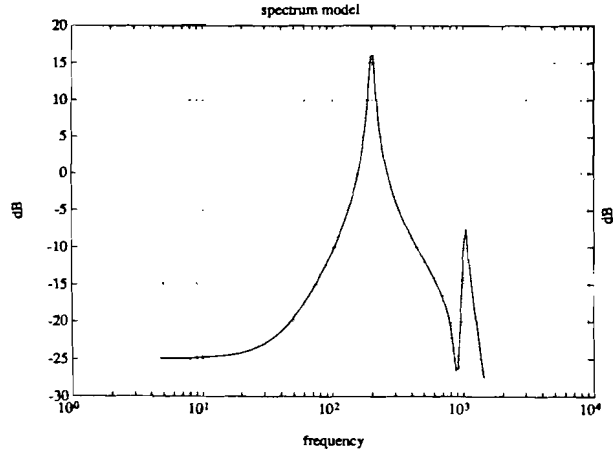


fig4.9b Measurement obtained after analoge measurement



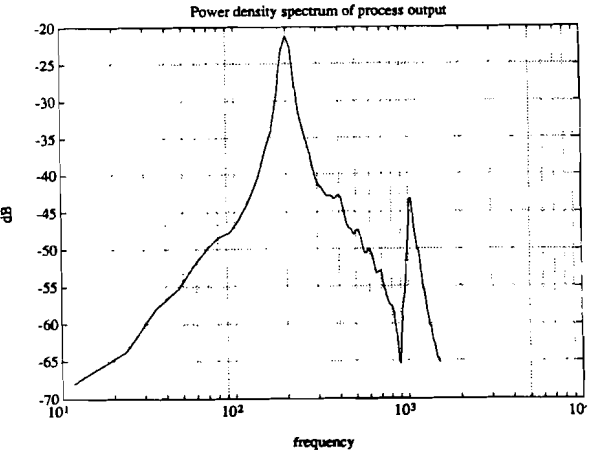


fig 4.10a Estimation of process plus filters fig 4.10b Power density spectrum of output

As can be seen in (fig. 4.9) the highest frequencies of the real process have been estimated correctly if the input data is prefiltered. Due to the filters the frequencie2s beyond 1200 Hz are deamplified. This is recognizable in both plots depicted in fig 4.10 From the previous pictures one can conclude that the required linear models, for linear digital, as well as for linear analog controller design are available.

For the model obtained using the configuration depicted in (fig. 3.3) a validation in the time domain has been included. Validation of the model obtained using the configuration from fig. 3.6 is not possible because process data in this set up is not available (analogue measurement).
A simulation of the 8^{th} order model, obtained using the second method, has been included.

The simulated-(dashed) and the process output(solid) have both been plotted in the next picture. One can assume that the mismatch is mainly caused by the nonlinearities, because of the low disturbances. Comparing this simulation with the simulation obtained using the first method, it can be seen that the second method performs slightly better

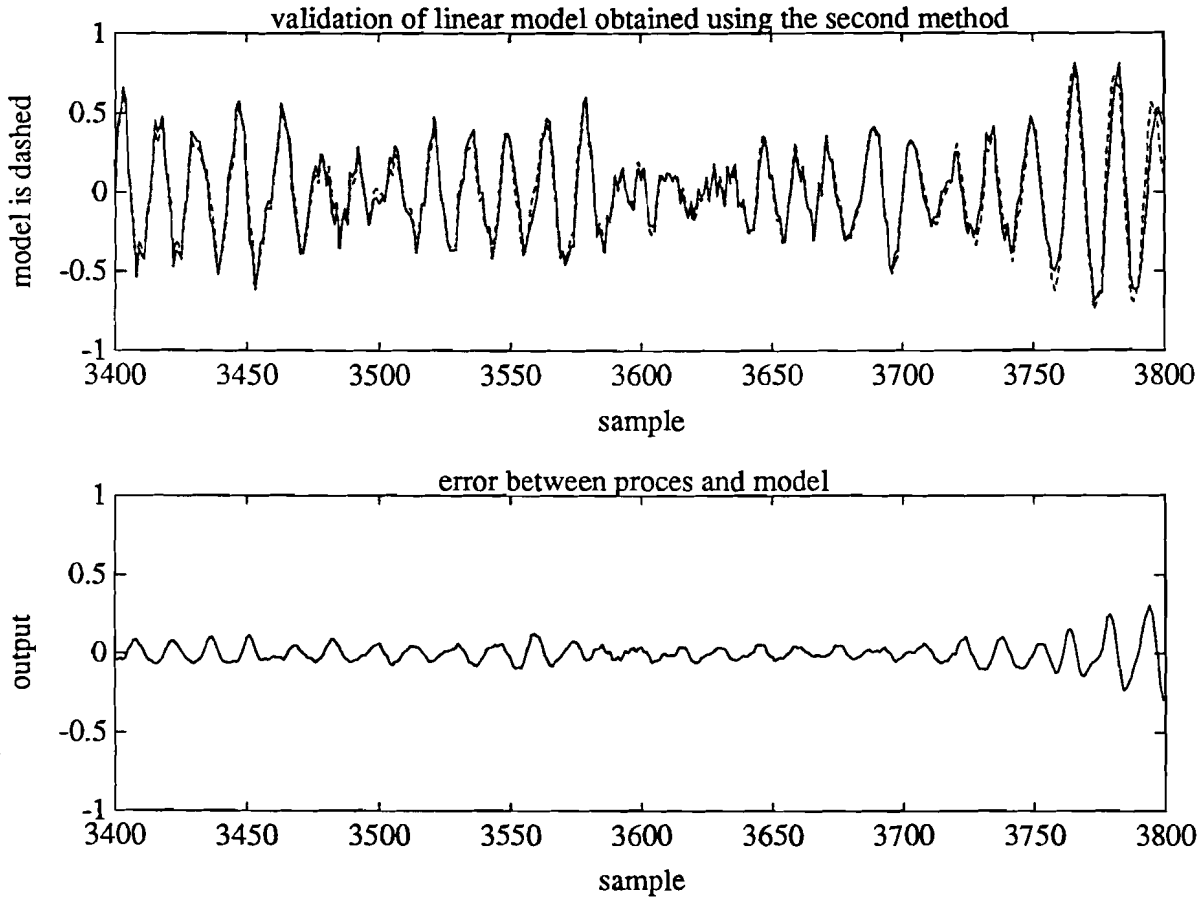
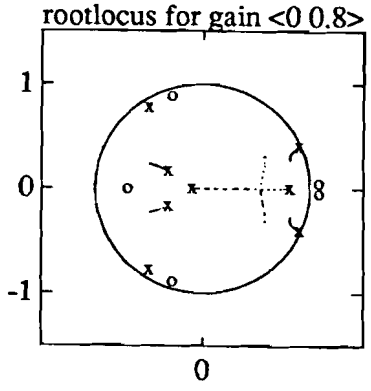


fig 4.11 Simulation of model obtained using method 2

Prestabilization for purpose of FIR parameters estimation.

On page 15 has been suggested to decrease the impulse length by constant gain feedback of the acceleration. Because a good linear model is available now, one can predict what constant gain feed back of the acceleration will bring about.
In the next picture, the root locus of the identified process for different constant gain feedback of the acceleration has been plotted. As can be seen, the gain can not be chosen arbitrarily large. According to the linearized model only a minor decrease of the impulse length can be obtained using constant gain feedback. Taking the gain to big results in instabilities of the controlled system.
The optimal gain with respect to the smallest impulse length, which can be obtained using constant gain feedback, is 0.8 (for this gain the dominant poles are far away from the unit circle, by increasing the gain the poles move towards the unit circle again).



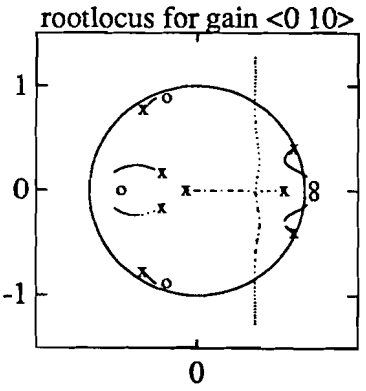


fig 4.12 rootlocus of linear model

5 PROCESS IDENTIFICATION USING ARTIFICIAL NEURAL NETS

Data preprocessing

Choosing an optimal sampling frequency and an input signal is important for creating an estimation set, because filtering and downsampling afterwards are not permitted. This can be explained easily: for linear processes there holds that every frequency component ω_k in the output can only be caused by the same frequency ω_k in the input. So filtering ω_k out of the spectrum of the input signals implies that no frequency component ω_k will be present in the output.
For non linear systems however this does not hold. Every frequency component in the output signal is composed, at least in principle, of a whole frequency range in the input signal. So filtering afterwards is not permitted.
On the other hand offset correction (O1) and scaling (A1) is permitted. Model estimation has been executed using the scheme depicted in fig (5.1).

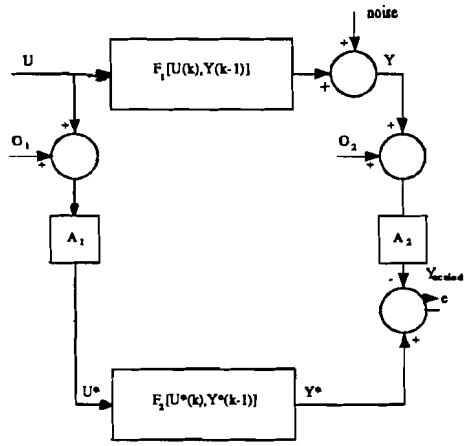


fig 5.1 Configuration used for estimation.

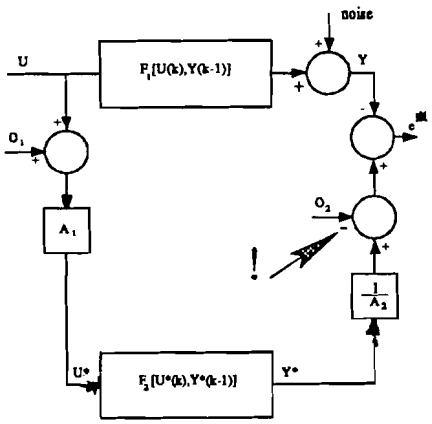


fig 5.2 Configuration used for simulation.

The parameters of the model have been chosen to minimize the error e= y^* - y_{scaled} in quadratic sense. (See fig 5.1) Because of the scaling factors and the non linear behaviour of the plant the estimated mapping (from u^* to y^*) is different from the real mapping (from u to y). Despite of the fact that the mapping is not estimated correctly, the model can be used for simulation purposes, using the configuration depicted in fig(5.2).

Whenever F_2 is determined so as to minimize $\|e\|_{L_2}$ in the configuration of fig. 5.1, the resulting F_2 also minimizes $\|e^*\|_{L_2}$ in schema of fig 5.2. Proof:

$$\underline{y}(k-1) = [y(k-1)\ y(k-2)\ \ldots\ y(k-n)]^T$$
$$\underline{u}(k) = [u(k)\ u(k-1)\ \ldots\ u(k-n)]^T$$
$$n = \text{model order obtained linear estimation}$$
$$J^* = \text{performance obtained after simulation.}$$
$$J^* = \sum_{k=1}^{N}\left[\frac{F_2\{\underline{u}(k)^*, \underline{y}^*(k-1)\}}{A_2} - O_2 - F_1\{\underline{u}(k), \underline{y}(k-1)\}\right]^2$$

$$J^* = \sum_{k=1}^{N}\left[\frac{F_2\{\underline{u}(k)^*, \underline{y}^*(k-1)\}}{A_2} - [F_1\{\underline{u}(k), \underline{y}(k-1)\} + O_2]\right]^2 \quad [5A]$$

$$A_2^2 * J^* = \sum_{k=1}^{N}[F_2\{\underline{u}(k)^*, \underline{y}^*(k-1)\} - A_2 * [F_1\{\underline{u}(k), \underline{y}(k-1)\} + O_2]]^2$$

$$\Rightarrow A_2^2 * J^* = J = \sum_{k=1}^{N}[y^*(k) - y_{scaled}(k)]^2$$

J = *Performance obtained after estimation.*
If J is optimal then J^* is also optimal. *QED*

the constants O_1, O_2, A_1 and A_2 have been depicted in fig. 5.1 and fig.5.2.

Neural nets

A neural net is a mapping form n inputs to m outputs. The parameters of the neural net are adjusted such that the error between outputs of the neural net and a given output signal is minimized in quadratic sense.
formally stated:

A neural net is a function, parameterized by a parameter vector $\underline{\theta} \in \mathbb{R}^{m^o}$.

given the data u(k),y_t(k) we wish to determine an optimal $\underline{\theta}_{opt} \in \mathbb{R}^{m^o}$ such that:

$$V(\theta) = \frac{1}{N}\sum_{k=1}^{N}[(f_\theta(u))(k) - y_t(k)]^2 \text{ is minimized. Where:}$$

$(F_\theta(u))(k)$ = *output of the neural net at moment k, based on* $\underline{\theta}$ *and the input u* [5B]
$= \hat{y}(k)$
$\underline{\theta}$ = *parameters of neural net*
k = *sample moment*

For model identification a miltilayered neural net has been used. Such neural nets will be described as follows:

$$NN^{L}_{i_0\, i_1\, \ldots\, i_L}$$

where: L := number of layers (input layer is excluded)
i(0):= number of inputs
i(1):= number of processing elements in the first hidden layer
i(L):= number of outputs

An comprehensive review of Artificial neural net has been given in Pijnenburg[4].

Optimization of the parameters

The algorithm to optimize the stated loss function V is a gradient search algorithm. The adjustments of the parameter values, can be realized using a steepest descent algorithm. Applied to tune the parameters of the multilayered neural net, this algorithm is called "back propagation algorithm".(=BPA) The name and the implementation has been described at the end of this chapter.

$$\underline{\theta}^0 = \underline{\theta}^{initial}$$

$$\underline{\theta}^i = \underline{\theta}^{i-1} - \alpha \frac{dV}{d\underline{\theta}^{i-1}}$$

$\underline{\theta}^i$ = *parameter vector after* i^{th} *iteration*
θ_p = p^{th} *element of the vector* $\underline{\theta}$
α = *a scalar influencing the stepsize*
$\frac{dV}{d\underline{\theta}}$ = *vector. Every element is the derivative on the errorspace*

$$8 \quad = \left[\frac{dV}{d\theta_1}\ \frac{dV}{d\theta_2}\ \cdots\ \frac{dV}{d\theta_{m^\circ}}\right]^T \qquad [5C]$$

m° = *number of parameters*

$$V = \frac{1}{N}\sum_{k=1}^{N} [y_p(k) - \hat{y}(k)]^2$$

one can speed up the algorithm by adjusting the parameters after L samples instead of N. where

$$L = \frac{1}{k}N \qquad \{k \in \mathbb{N} \mid k \leq N\}$$

= *Batch factor*

Situations were it is not recommendable to choose L (later on refered to as "batchfactor") smaller than N will be discussed in section "batch factor"
The algorithm can be visualized easily if only 1 parameter has to be adjusted. Suppose the process which has to be estimated is:

```
y[u] = 1*tanh(2*u + 0)                                    [5D]
```

The process (or mapping or fuction) can be fitted exactly using a 3-layered neural net with one node in the hidden layer (processingfunction of the node is tanh(.)).

.) = tanh(.)
y = a*tanh(w*u + p) + o
) = [p w a o]

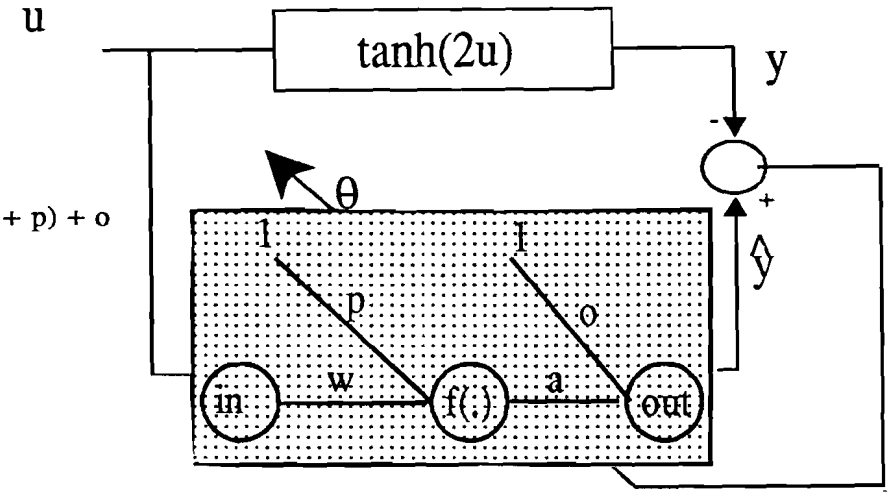


fig 5.3 Model to model adjustment using one hidden layer.

The output of the neural net is described by $\hat{y} = a \cdot f(w \cdot u + p \cdot bias) + o \cdot offset$
The parameter vector, describing the net equals $\theta = [p\ w\ a\ o]^T$
Assume the parameters w, a and o are known yet, so only p has to be adjusted using a steepest descent algorithm. For different values of p the criterium [5B] which has to be minimized can be calculated. The values are plotted in fig. (5.4).
$\underline{\theta}_t=[p_t\ w_t\ a_t\ o_t]^T = [0\ 2\ 1\ 0]^T$

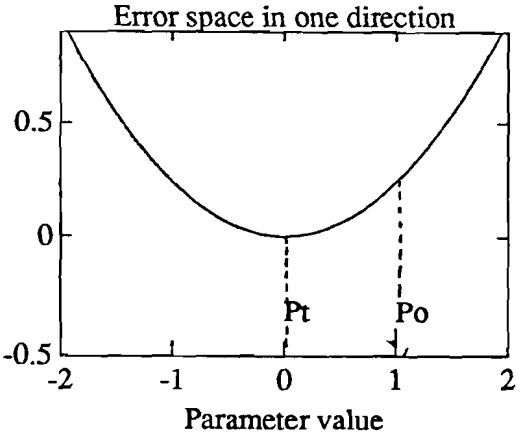


fig 5.4a Performance.

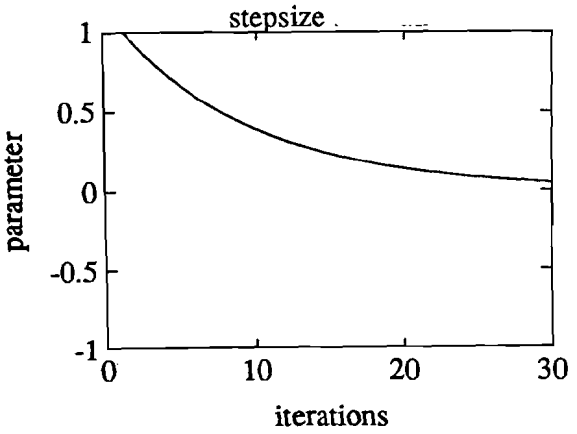


fig 5.4b Iterations

We want tot minimize V so the optimal p equals p_t.
Starting at p_0 the optimal p_t can be found easily using algorithm [5C].
For convex errorsurfaces a steepest descent algorithm easily manages to find the optimum. The error surface concerning the optimization of the neural net parameters is far from convex. Therefore optimization can cause some problems. One can effect the convergence of the learning algorithm by means of:

- the variable α.
- proper choise of initial weights.
- varying batchfactor L (see [5C]).

(L equals the number of samples used for the calculation of the gradient)

Stepsize.

The stepsize α can be chosen within a certain interval. Choosing α to small results in slow convergence, choosing α too big however, can result in divergence or oscillations

For quadratic error surfaces one can calculate the exact interval for which the algorithm will converge. Assume the errorsurface can be described according to the following formula:

$$
\begin{aligned}
V &= a\theta^2 + b\theta \Rightarrow \textit{minimum at } \theta_t = \frac{-b}{2a} \quad (a>0) \\
\theta^0 &= 0 \\
&\textit{applying the steepest descent} \\
&\textit{algorithm results in:} \\
\theta^{I+1} &= \theta^I - \alpha\frac{dV}{d\theta^I} \\
&= \theta^I(1-2\alpha a) - \alpha b \\
\theta^I &= -\alpha b\sum_{k=1}^{I-1}(1-2\alpha a)^k - \alpha b \\
&\textit{convergent if } |1-2\alpha a| < 1 \\
&\Rightarrow 0 < \alpha < \frac{1}{a}
\end{aligned}
\qquad [5E]
$$

If the error surface is quadratic the optimal parameter value can be calculated in one iteration step using:

$$
\theta_{n+1} = \theta_n - \left[\frac{d^2V}{d\theta_n^2}\right]^{-1}\frac{dV}{d\theta_n}
$$

$$
\textit{In this example } \alpha_{opt} = \frac{1}{2a}
$$

The error surface concerning the neural net parameters is far from convex, so the optimal α can not be calculated using the previous descrbed algorithm. A correct α has to be defined by trial and error. An α in the interval <0.001 0.8> turned out to perform sufficient.

Batchfactor

Another parameter which can influence the convergence rate is the batchfactor. (L in [5C]). Adjusting this parameter one can obtain a trade off between noise sensitivity and calculation time. The higher the batchsize the less the noise sensitivity. Because the noise is supposed to be random and independent one can assume that the average noise contribution to the calculated gradient becomes smaller as the length of the dataset increases. The disadvantage of a high batchsize is the time necessary to perform the calculations which have to be made to adjust the parameters.

In choosing the batchsize L, one has also to take into account the kind of input signals. When all the input signals of the net are uniformly distributed over a certain interval, a batchsize of one would not give any problems. When however an inputsignal for example has a couple of dominant frequencies the choice of the batchsize is important.

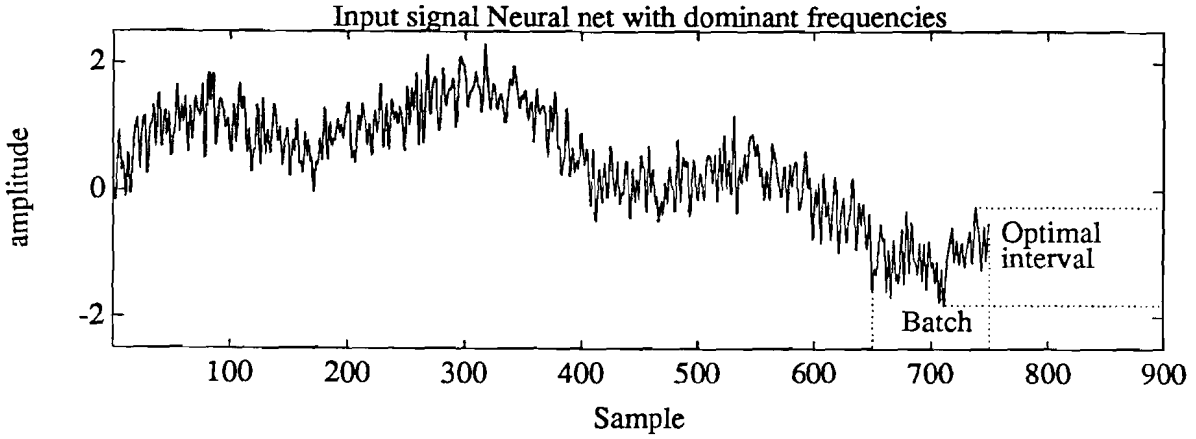


fig 5.5 Input signal with dominant frequencies.

When the batchsize is chosen to be 100, only the last part of the input set is, after optimization, fitted in an optimal sense. So the performance of the model will only be optimal when that input is excited within the optimal interval indicated in the previous picture. So the richness of the signal and the power of the disturbance determines the minimal batchfactor L. For the inputsignal depicted in fig 5.5 it is recomendable to take L=N because the input is not rich enough.

Initial weights

The last item which can be used to influence the convergence rate is the choice of initial weights. A neural net is subdivided in several identical subsections. One subsection has been given in the next picture.

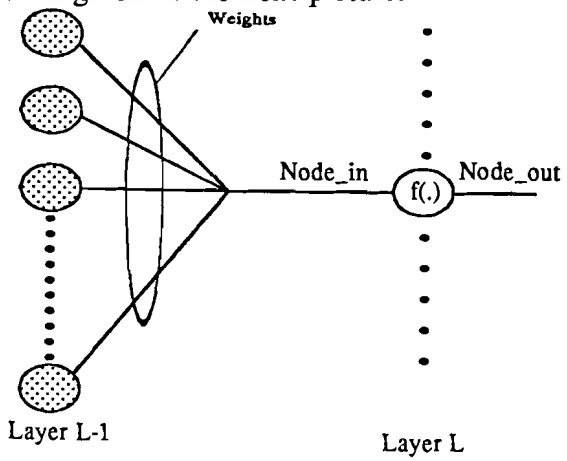


fig 5.6 Subsction of multilayered neural net.

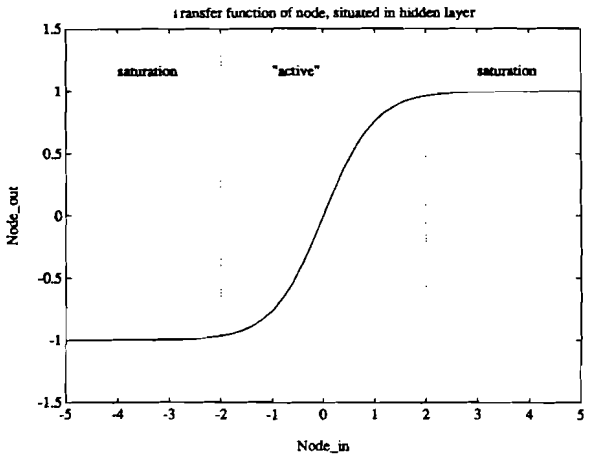


fig 5.7 Transferfunction.

It is important to choose the initial weights such that for the given input vectors at layer L-1, "node_in" never is in the interval for which the transfer function saturates. (no saturation for -2 < node_in < 2) If not the error surface in the parameter space near to the initial weights is to be expected very flat at the first iteration. This is due to the fact that, once in saturation, "node_out" is almost insensitive for minor changes of the parameters. Using a tanh(.) as a transfer function one can calculate, conform [5F], the interval where in the initial weights have to be chosen to prevent flat error surfaces at the first iteration.
(The parameters has to be chosen randomly)

$$Bias = 1$$

$$Assume\ saturation\ starts\ at\ 2$$

$$Maximum\ interval\ for\ init\ W\ to\ prevent\ saturation:$$

$$W_{init} \in [-Max, Max]\ with$$

$$Max = \frac{2}{Input_{max} * \#Inputs + bias}$$

$$Input_{max} = Max \| Input\ Amplitude \| \qquad [5F]$$

$$\#Inputs = Number\ of\ inputs\ of\ the\ net$$

$$Using\ more\ hidden\ layers\ one\ has\ to\ take$$
$$for\ the\ second,\ the\ third,..hidden\ layer:$$

$$Max = \frac{2}{\#Nodes_{prev} + bias}$$

$$\#Nodes_{prev} = Number\ of\ nodes\ in\ previous\ layer$$

If these precautions are not taken it is hard to find the optimum using BPA. This has been illustrated in the next example.
The same error surface as in fig(5.5), has been calculated again but now for a broader range of P. As can be seen in the next picture the gradient in sections 1 and 3 is very small.

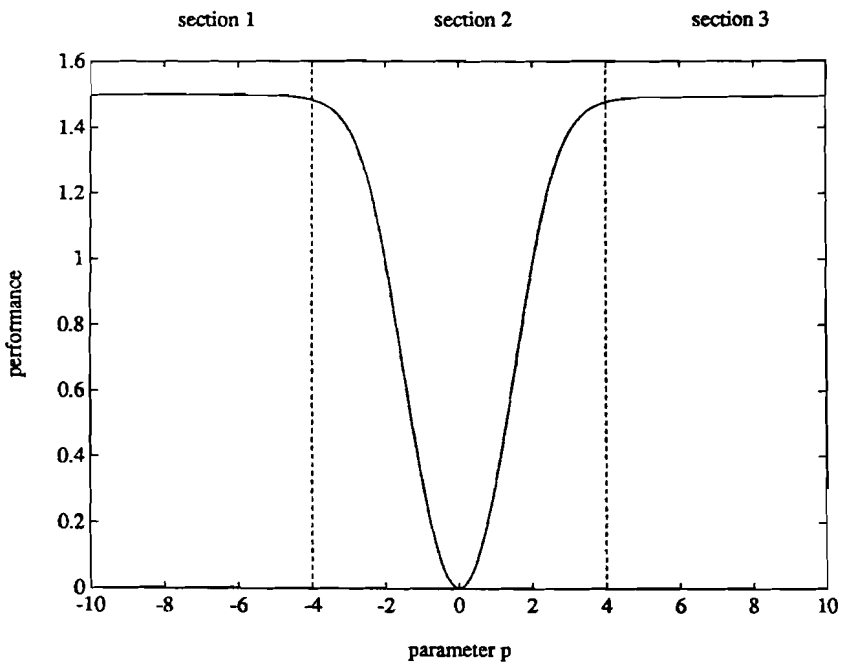


fig 5.9 Error surface in one direction.

When the initial parameter is situated in these sections convergence to the optimal value is hard to accomplish. Because the gradient is almost zero one has to choose a large α to increase convergence rate. When P eventually reaches the disjunction between 2 sections the parameter skips section 2 because the stepsize at that point is far too big. The flat regions, or plateaus, are characteristic for error surfaces concerning the parameterspace of a neural net.

This example only illustrates the problem which occurs if the initial weights are too big. Flat error surfaces or plateaus also can occur at the learning stage. This is a problem which occurs if the parameters of a neural net have to be optimized. By choosing small initial weights one is only sure that at first iteration no flat errorsurface occurs.

Illustrating the phenomenon "plateau", a two dimensional error surface has been plotted as a function of the parameters P and W. As can be seen the error surface, for such simple system, is not convex at all. Looking at this picture it seems impossible to minimize this type of error criterium, using BPA. However taking the precautions with respect to the initial values, the algorithm indeed manages to find the solution.

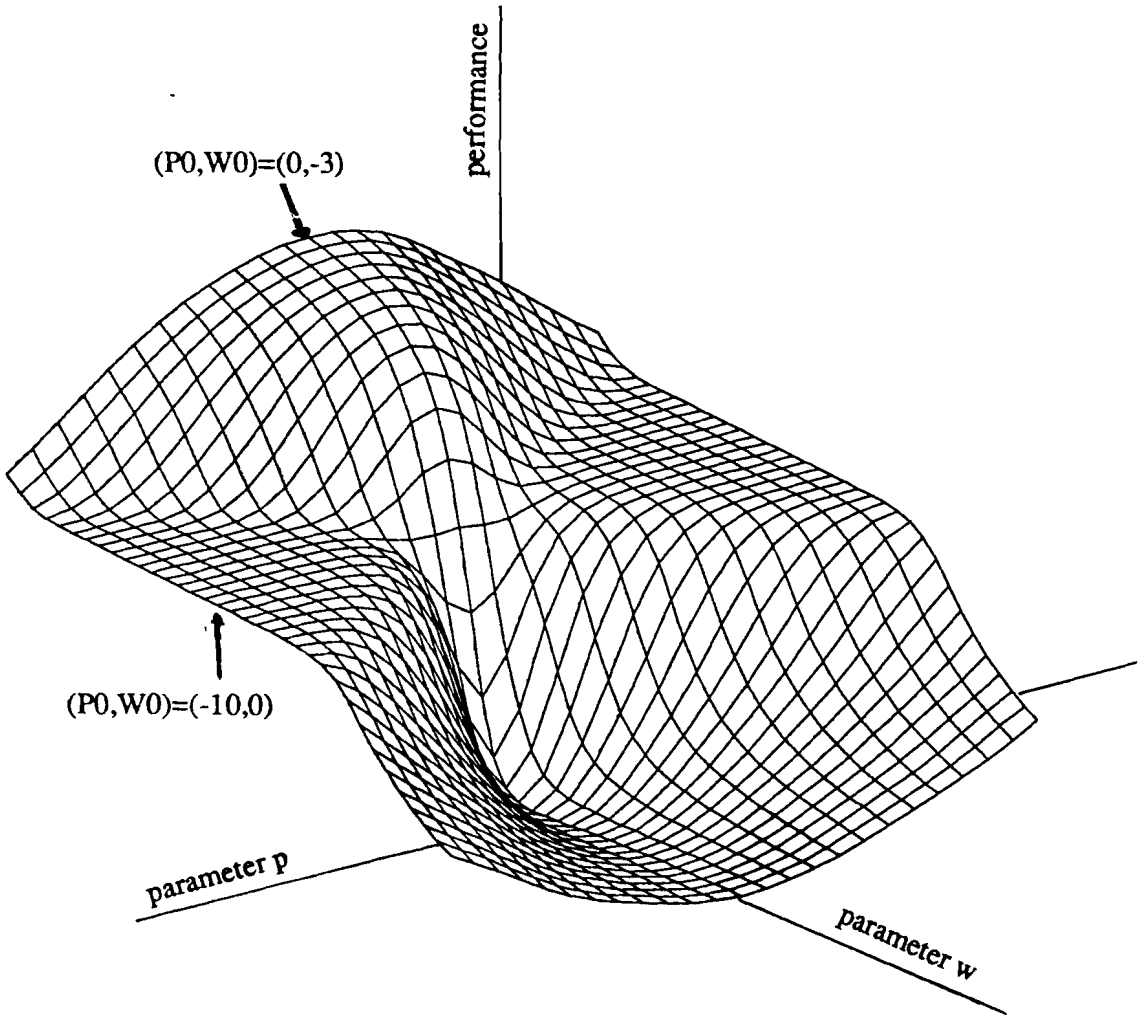


fig 5.10 Error surface in two directions.

From simulation experience it is evident that by choosing the initial weights small enough and choosing α not too big, the algorithm usually manage to find an optimum.

For illustration purpose some simulations have been included. It is a model to model adjustment of a static process which can exactly be modelled by the neural net. The model is described by:

y=2tanh(5u-5) + 2tanh(-10u-3) + tanh(3u+2) + 1.15 (shown in fig. 5.11)

The real neural net parameters are uniquely defined by

$P_{true} = [-5 \ -3 \ 2]^T$

$W_{true} = [\ 5 \ -10 \ 3]^T$

$A_{true} = [\ 2 \ \ 2 \ \ 1]^T$

$O_{true} = 1.15$

The estimation has been executed using the following setup

input: Uniformly distributed white noise (1000 samples).
Amplitude in the interval <-1.5 1.5>

α : 0.1

L : 1

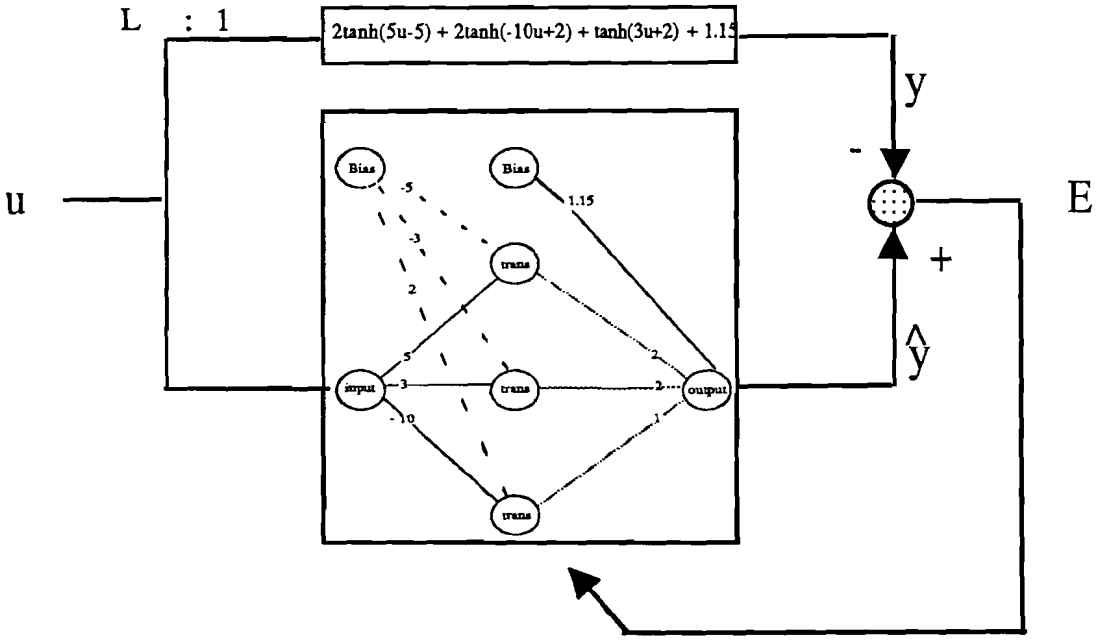


fig 5.11 Static model to model adjustment.

Because the exact parameters are known one can investigate whether the algorithm converges to the true value.

Three simulations have been carried out.

1: noiseless with small initial weights
2: noiseless with large initial weights
3: additive white noise with small initial weights

In the first experiment the parameters converged to the true values within 20 iterations.
In the second experiment the parameters converged tot a biased value, probably the error surface was very flat.
In the last experiment uniformly distributed white noise has been added (signal to noise ratio= 12dB)
Despite the bad signal to noise ratio the net was still capable estimating the function quite well. (At the last simulation the batchsize has been chosen to be the length of the dataset).

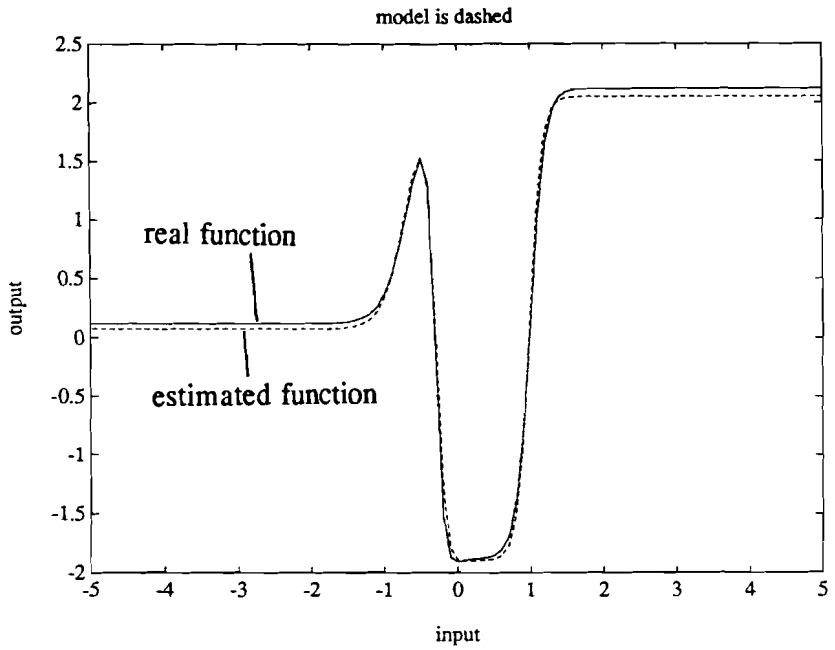


fig 5.12 Estimated function based on noisy data.

Conclusively could be stated for static model estimation:

- The examined static function can easily be estimated using neural nets with a steepest decent algorithm.
- Initial weights have to be chosen such, that the transfer functions are not saturated in the first iteration.
- Using large batchsizes it is possible to do estimations based on noisy outputs.
- The batchfactor L can be chosen smaller than N if the inputs of the net are "rich enough"

Modelling dynamic systems

In the previous chapter has been explained how a static function can be represented by a multi layered neural network.
The neural net gives the relation between the input and the output of the net.
Using a neural net for modelling dynamic systems the number of inputs of the net should be extended so as to include delayed (or past) values of inputs and outputs. Fig 5.13 shows how to configurate signals so as to introduce dynamics in the static net.

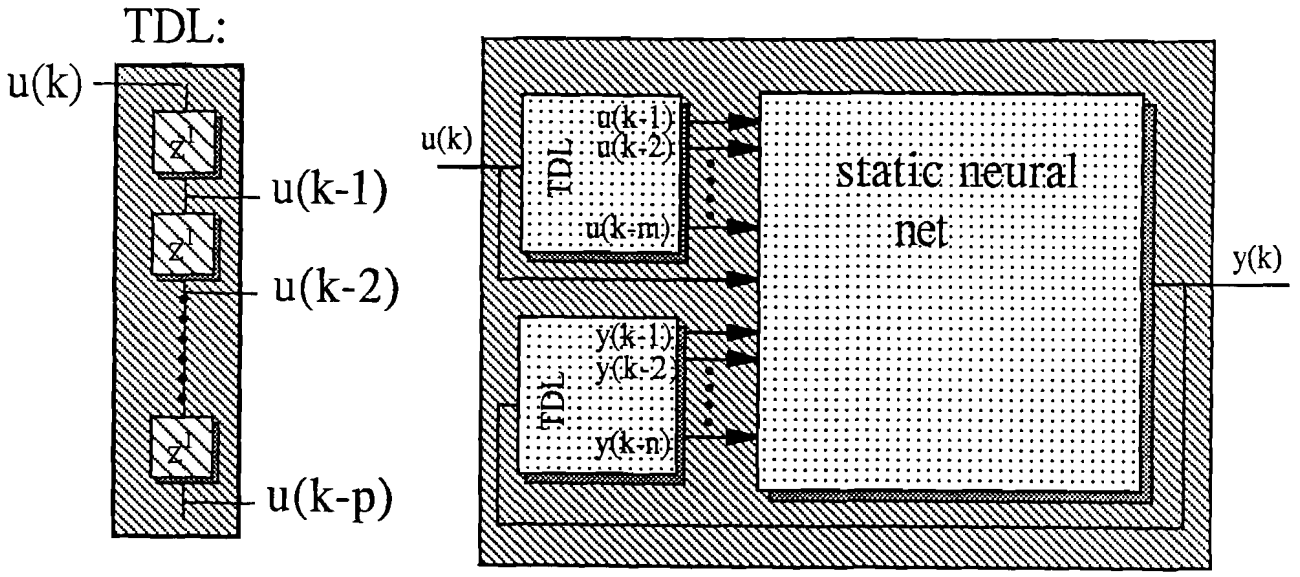


fig 5.13 Dynamical neural net.

The static neural net is just a mapping from inputs to outputs.
For dynamical siso systems the description is given by:

$$y(k) = F(\underline{y}(k-1), \underline{u}(k), \underline{\theta})$$
$$\underline{y}(k-1) = [y(k-1)\ y(k-2)\ y(k-3)\ \ldots\ y(k-n)\ ]^T$$
$$\underline{u}(k) = [u(k)\ u(k-1)\ u(k-2)\ \ldots\ u(k-p)\ ]^T$$

$y(k)$ = *output of neural net at sample moment k* 5G
$u(k)$ = *input of neural net at sample moment k*
n = *order of model*
$n \geq m$
$\underline{\theta}$ = *parameter vector* $\in \mathbb{R}^{m^*}$
denoting the vector of weights of the neural net

For dynamical mimo systems u(k) and y(k) are vectors. Usually one takes p equals n. n is the estimated order of the system acquired from linear estimation.

Parameter estimation

Like optimization of parameters for linear model estimation one can distinguish two methods for non linear model estimation. Prediction error method and output error method. As by linear estimation the optimum according to the first method is more easy to find. However the problem can not be solved analytically, so the parameters have to be adjusted using an iterative algorithm.

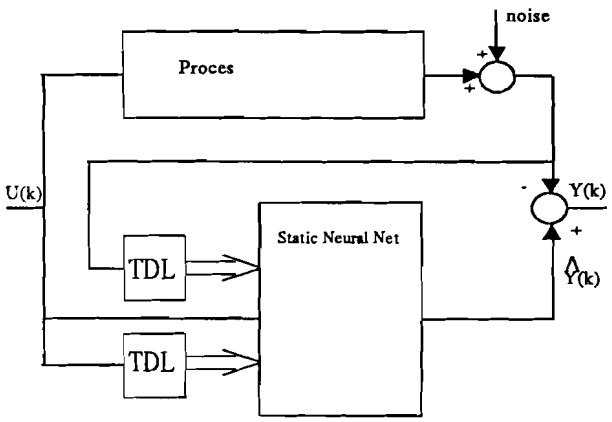


fig 5.14 Configuration using PEM.

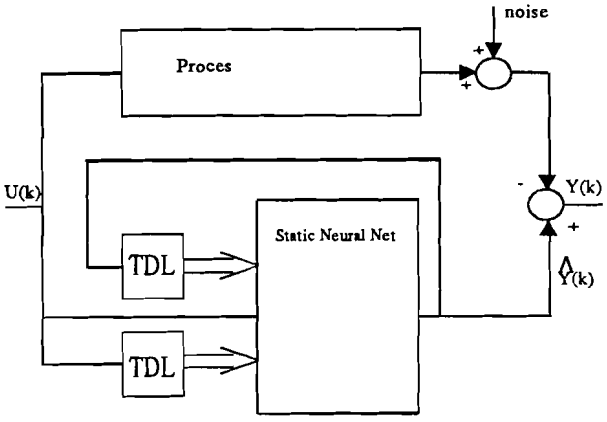


fig 5.15 Configuration using OEM

One algorithm which can solve the problem is the steepest descent algorithm. The algorithm has been explained in the previous section. For dynamical systems there are problems which have to be solved. We want to minimize the criterium [5B]. The steepest decent algorithm is also valid for dynamical systems.

$$V = \frac{1}{2N}\sum_{k=1}^{N} [\hat{y}(k) - y_t(k)]^2$$

$$= \frac{1}{2N}\sum_{k=1}^{N} [e(k)]^2$$

$$\frac{dV}{d\theta} = \frac{1}{N}\sum_{k=1}^{N}\left[e(k)\frac{d\hat{y}(k)}{d\theta}\right]$$

The problem is the calculation of the gradient $d\hat{y}/d\theta$.

Distinguishing the two methods the model output can be calculated according to the following formulas:

$$\begin{aligned} PEM:&\quad \hat{y}(k) = F(\underline{y}_t(k-1), \underline{u}(k), \underline{\theta}) \\ OEM:&\quad \hat{y}(k) = F(\underline{\hat{y}}(k-1), \underline{u}(k), \underline{\theta}) \end{aligned} \qquad [5H]$$

The gradients for the different configurations has to be calculated according to:

$$PEM:\quad \frac{d\hat{y}(k)}{d\theta_i} = \frac{\partial F(\cdot)}{\partial \underline{y}_t^T(k-1)} \frac{d\underline{y}_t(k-1)}{d\theta_i} + \frac{\partial F(\cdot)}{\partial \underline{u}^T(k)} \frac{d\underline{u}(k)}{d\theta_i} + \frac{\partial F(\cdot)}{\partial \underline{\theta}^T} \frac{d\underline{\theta}}{d\theta_i} \qquad 5I$$

$$OEM\quad \frac{d\hat{y}(k)}{d\theta_i} = \frac{\partial F(\cdot)}{\partial \underline{\hat{y}}^T(k-1)} \frac{d\underline{\hat{y}}(k-1)}{d\theta_i} + \frac{\partial F(\cdot)}{\partial \underline{u}^T(k)} \frac{d\underline{u}(k)}{d\theta_i} + \frac{\partial F(\cdot)}{\partial \underline{\theta}^T} \frac{d\underline{\theta}}{d\theta_i} \qquad [5J]$$

$\hat{y}(k)$ = *output of neural net at sample moment k*
$y_t(k)$ = *Proces output at sample moment k*

Last term in both [5I] and [5J] can be easily calculated. The second term, for both methods, is zero because the inputs of the net can be chosen independent form the parameters.

The first term in [5I] is zero because the process output is independent of the neural net. So for PEM the gradient can be calculated using the next formula.

$$\frac{d\hat{y}(k)}{d\theta_i} = \frac{\partial F(\cdot)}{\partial \theta_i} \qquad [5K]$$

Using OEM the first term in [5J] is not zero. Perturbation of a parameter results in a change of the output. Because this output is fed back to the net it can have a large contribution to the gradient.
If this term is negelected, the algorithm is refered as "static back propagation". For a lot of processes this does not give any problems. For badly damped systems however, it is found that static back propagation can not be used because the contribution of the first term is very substantial. In that case is the dependence on the previous y(k-t) as important as the as the dependence on u(k) via the network. For such processes the next formula should be used (using this formula the real gradient is obtained):

$$\frac{d\hat{y}(k)}{d\theta_p^i} = \frac{\partial F(\cdot)}{\partial \underline{\hat{y}}^T(k-1)} W(z^{-1}) \frac{d\hat{y}(k)}{d\theta_p^i} + \frac{\partial F(\cdot)}{\partial \theta_p^i}$$

$$\begin{aligned} W(z^{-1}) &= \textit{delay operator} \\ &= \left[z^{-1}\ z^{-2}\ z^{-3}\ \ldots\ z^{-n}\right]^T \\ n &= \textit{the estimated process order} \end{aligned} \qquad [5L]$$

$$\frac{d\hat{y}(k)}{d\theta_p^i} = \sum_{t=1}^{n} \frac{\partial F(.)}{\partial \hat{y}(k-t)} \frac{d\hat{y}(k-t)}{d\theta_p^i} + \frac{\partial F(.)}{\partial \theta_p^i}$$

Analytical calculation of gradient.

In formula [5L] can be seen that $d\hat{y}/d\theta^i$ is a function of the previous gradients based on different input patterns, but calculated with the same parameter vector $\underline{\theta}^i$. The calculated gradient at k=0 is wrong because the previous gradients are not known. However it is to be expected that, as k increases, the calculated gradient gradually converges to the correct value.

Assume that for k=skip (see 5M) the calculated gradient has converged to the real value. If skip can be determined (and after redefining the performance) the gradient $dV/d\theta^i$ can be calculated precisely.

$$V = \frac{1}{2}\frac{1}{N-skip}\sum_{K=skip}^{N}\left[y_{model}(k) - y_{process}(k)\right]^2$$

$$\frac{dV}{d\theta^I} = \sum_{k=skip}^{L} e(k)\frac{dy(k)}{d\theta^I} \qquad [5M]$$

L = number of samples used to calculate the gradient $\frac{dV}{d\theta^I}$.

N = number of data samples used for estimation

(It is difficult to determine "skip". A suggestion, purely based on intuition, is to take skip equal to the estimated impulse length.)

An important remark has to be made. If the parameters are adjusted according to a batch factor (L) smaller than the number of data samples (N), one has to calculate the gradient $dy/d\theta$ again for all previous samples. However if the stepsize α is small enough the effect is negligible.

Implementation

$$WL_{t,\underline{s}} = [WL_{t,0}\ WL_{t,1}\ \ldots\ WL_{t,i(L-1)}]$$

WL = matrix which contains the parameters in between row L and row L-1

$$\underline{xl}_r = [xl_1\ xl_2\ \ldots\ xl_{i(l-1)}]$$

$$l = 0, 1, \ldots l_{max}$$

l_{max} = number of layers

$i(l)$ = number of nodes in layer l

xl_i = output signal of i^{th} node in layer l

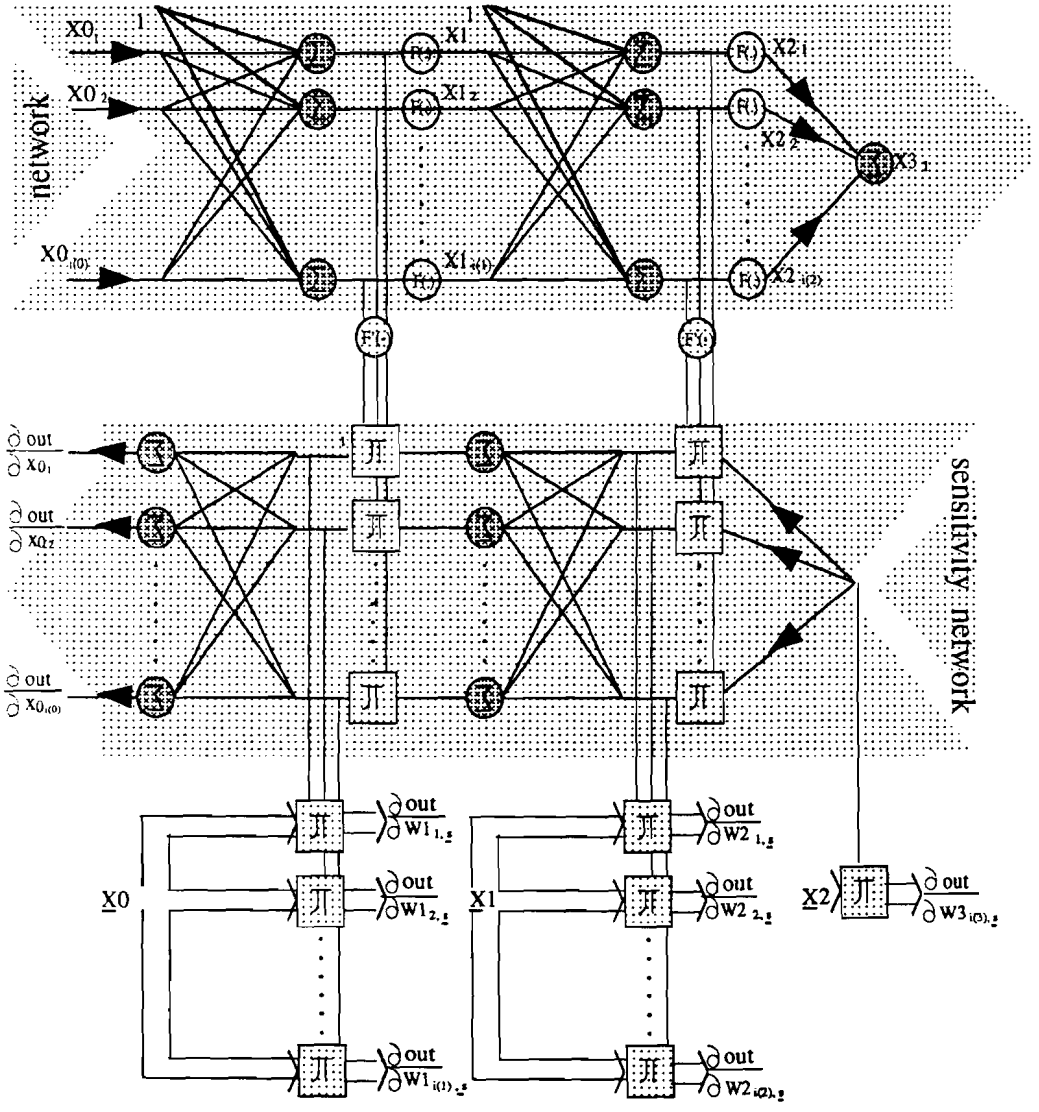


fig 5.8. Sensitivity network

Up till now nothing has been said about how the partial derivatives can be obtained. They can be calculated using a configuration similar to the multilayered neural net. For every sample moment k the input pattern is distributed through the net. If the output is calculated, the partial derivatives are calculated by propagating backwards through a 'sensitivity net' depicted in the fig 5.8.

In Narendra [12] a sensitivity net has been given which can be used to calculate $\partial V/\partial\theta$. In that picture the error e(k) is fed back into the input of the 'sensitivity net. The error is back propagated through the net (therefore the steepest descent algorithm, applied to tune the neural net parameters, is called "back propagation").
The output of that net equals $e^*d\hat{y}/d\theta$.
The sensitivity net in fig 5.8. can be used to calculate $\partial\hat{y}(k)/\partial\theta$ and $\partial\hat{y}(k)/\partial\hat{y}(k-i)$ where i>=1. (As can be seen the error e(k) is not fed back in this configuration)

Numerical calculation of gradient
Another way to calculate the gradients is using numerical approximation.
The advantage is: - simple algorithm, easy to implement
disadvantage: - a lot of calculations have to be made.

For every parameter out of the parameter vector $\underline{\theta}$, dV has to be calculated. This has been schematically depicted in the next picture.

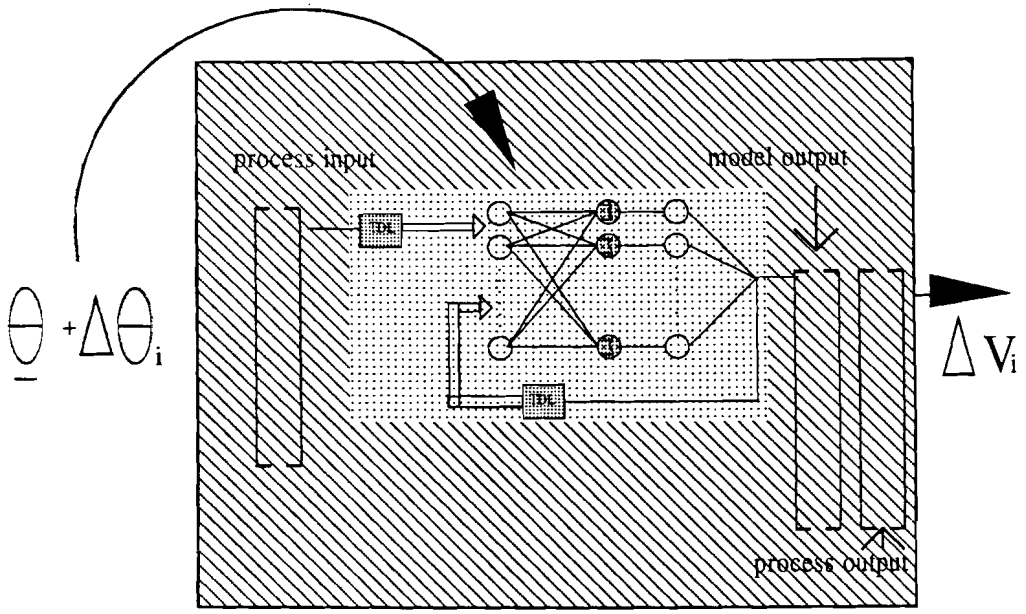


fig 5.9 numerical gradient approximation.

So for every θ all the samples have to be calculated.
Some different methods, used in two configurations, have been described. In the next section an overview has been given.

Overview of the algorithms

Optimizing parameters of a neural net, 4 situations can be distinguished.

Dependent of the configuration OEM or PEM, two kinds of loss functions are minimized:

DIFFERENT LOSS FUNCTIONS

$$V_{pem} = \frac{1}{2N}\sum_{k=1}^{N}\left[y_{model}\{\underline{y}_{process}(k-1),\underline{u}(k),\underline{\theta}\} - y_{process}(k)\right]^2$$
$$V_{oem} = \frac{1}{2N}\sum_{k=1}^{N}\left[y_{model}\{\underline{y}_{model}(k-1),\underline{u}(k),\underline{\theta}\} - y_{process}(k)\right]^2 \qquad [5N]$$

There are two methods to calculate the gradient.

DIFFERENT CALCULATIONS OF GRADIENT

$$static: \quad \frac{dy(k)}{d\theta} = \frac{\partial y(k)}{\partial \theta}$$
$$dynamic: \quad \frac{dy(k)}{d\theta} = \frac{\partial y(k)}{\partial \theta} + \sum_{t=1}^{n}\frac{\partial y(k)}{\partial y(k-t)}\frac{dy(k-t)}{d\theta} \qquad [5O]$$
$$y(k) = output\ of\ neural\ net\ at\ sample\ k$$

The first method is easy to apply, but it is in OEM configuration, an approximation of the real gradient. To apply the second method, a dynamic formula has to be solved. Therefore it is more complicated but, the real gradient is likely to obtained.

Static gradient calculation in OEM configuration is just an approximation!

An overview of all different methods has been given in the next table.

ALGORITHM	CRITERIA	
	V_{pem}	V_{oem}
Static	PEM	OEM_s
Dynamic	PEM_d=PEM	OEM_d

In the PEM configuration, no model feedback is present, the term $\frac{dy(k-t)}{d\theta}$ in formula [5O] is zero. Therefore no distinction between PEM_d and PEM_s can be made.

Using numerical approximation like is depicted in fig 5.9 an OEM method is applied. In the remainder of the report this method has been depicted as OEM_n.

To show the influence (on the calculated gradient) of using OEM_s in stead of OEM_d, a simple example is included.
Assume a linear first order system:

$$y(k) = b_o u(k) + a_1 y(k-1)$$

if we wish to determine the gradient $\frac{dy(k)}{da_1}$ *can use:*

$$\frac{dy(k)}{da_1} = a_1 \frac{dy(k-1)}{da_1} + y(k-1)$$

Since this equation is valid for all k
we can recursively deduce that:

$$\frac{dy(k)}{da_1} = [a_1]^n \frac{dy(k-n)}{da_1} + \sum_{i=0}^{n-1} [a_1]^i y(k-i-1)$$

As we are dealing with stable systems so $|a_1| < 1$ *and so*

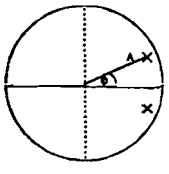
$$\lim_{n \to \infty} a^n = 0$$

we therefore obtain:

$$\frac{dy(k)}{da_1} = \sum_{i=0}^{n-1} a^i y(k-i-1) \qquad [5P]$$

Using OEM_s the gradient dy/da_1 is calculated as y(k-1). It is obvious that the contribution of the previous outputs y(k-i) on the gradient can be very substantial for badly damped systems. For non-linear systems similar expression can be derived. This expression will be more complicated, but the conclusion remains the same.

According to the previous analysis there are problems to be expected if a badly damped system has to be estimated. Because the process under test has 2 poles in the proximity of the unit circle it has first been examined whether a neural net can manage to estimate a badly damped second order system.
Consider the pole configuration of the picture below.



$$p_{12} = A * e^{(-j\phi)}$$
$$\phi = 23°$$
$$A \in [0{,}8..1\rangle \qquad [5Q]$$

For different values of "A", PEM as well as OEM_s has been applied.
To estimate the output of the system, a uniformly distributed white noise input has been applied.

The parameters of a NN_{351}^{3} has been adjusted in three steps:

1:- PEM, batchfactor=1, α=0.1, 10 iterations
1:- PEM, batchfactor=N, α=0.01, 20 iterations
1:- OEM_s, batchfactor=N, α=0.01, 20 iterations

The results has been depicted in the next table:

A	J_{PEM} [%]	J_{OEMs} [%]
0.8	0.01	0.2
0.9	0.01	0.4
0.95	0.01	0.8
0.98	0.01	2.8
0.99	0.01	8.0
0.999	0.01	130.0

Estimation of a second order process using simple back propagation
table 5.1

As can be seen from the previous table, the optimum, according to PEM, can be easily found for all "A". This is to be expected because the gradient is calculated correctly. Using OEM_s, problems occur when A is getting too big. This is caused by the neglected term in [5K]. Due to this approximation the calculated gradient is much smaller than the real gradient.

An illustration has been given for A=0,98. For one parameter of the net, which is adapted to model this process the error surface has been plotted. The gradient, calculated using OEM_s turned out to be 2.5 e-4. In fig 5.16 the performance has been calculated for different parameter values. From this plot an approximation of the real gradient can be obtained.

$$\frac{dV}{d\theta} \approx \frac{\Delta V}{\Delta\theta} = \frac{0.01}{0.0038} = 2.6$$

So the real gradient is much bigger than the calculated gradient. Such plots can be made for all the parameters. The conclusion is the same: the calculated gradient is much smaller than the real gradient!

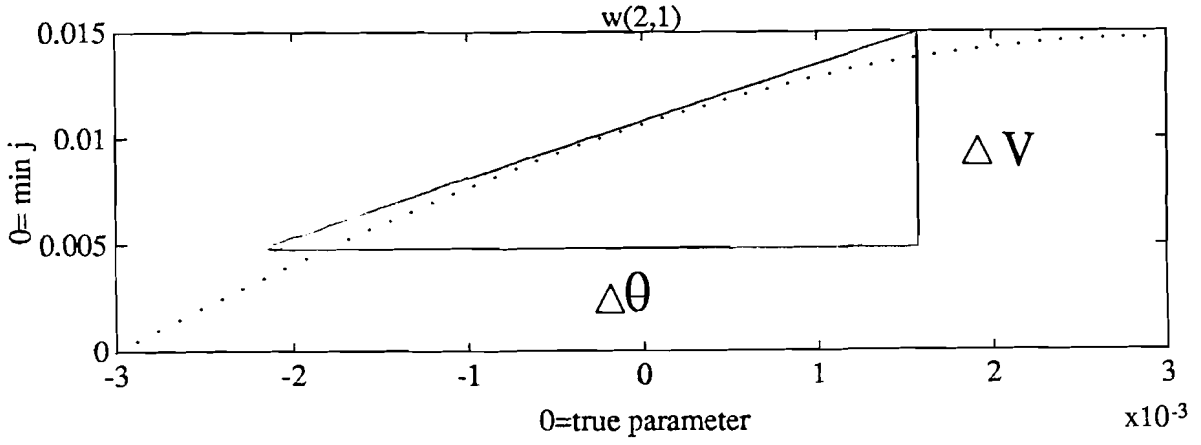


fig 5.16 Errorsurface.
Calculated gradient is 2.5E-4. Real gradient 2.6

0 on the X axis stands for the real parameter value after the last iteration.
0 on the Y axis minimum performance calculated in the plot.

Conclusion:
OEM_s does not give good results for badly damped systems.

6 NONLINEAR IDENTIFICATION OF AN ACOUSTIC PROCESS

In section 5 several algorithms have been discussed. In this section these algortihms have been applied in order to identify an acoustic system, described in section 3. A NN^3_{17171} has been used. The number of the delayed inputs for the net is equal to the estimated process order, obtained after linear identification. An eight order linear model has been obtained, therefore a neural net with 17 inputs is necessary (current input, eight delayed inputs and eight delayed outpus). The number of hidden layers, and -nodes, have to be chosen by trial. In this experiment a neural net with one hidden layer with 17 nodes has been chosen The difference between PEM, OEM_s and OEM_d has been given.

Identification

As could be expected from the previous experiments the prediction properties of a model based on PEM is indeed sufficiently accurate. However a simulation of the same model is very poor.

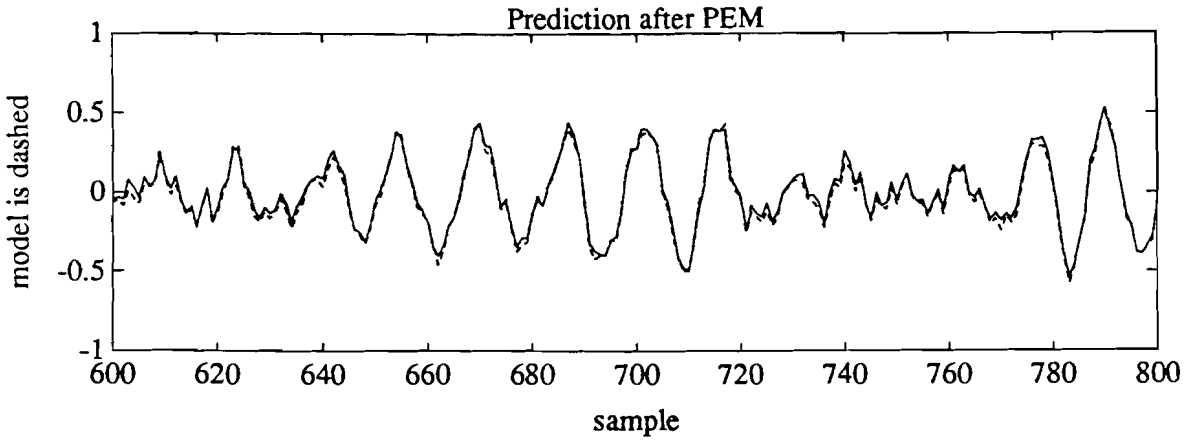


fig 6.1 prediction after PEM has been applied

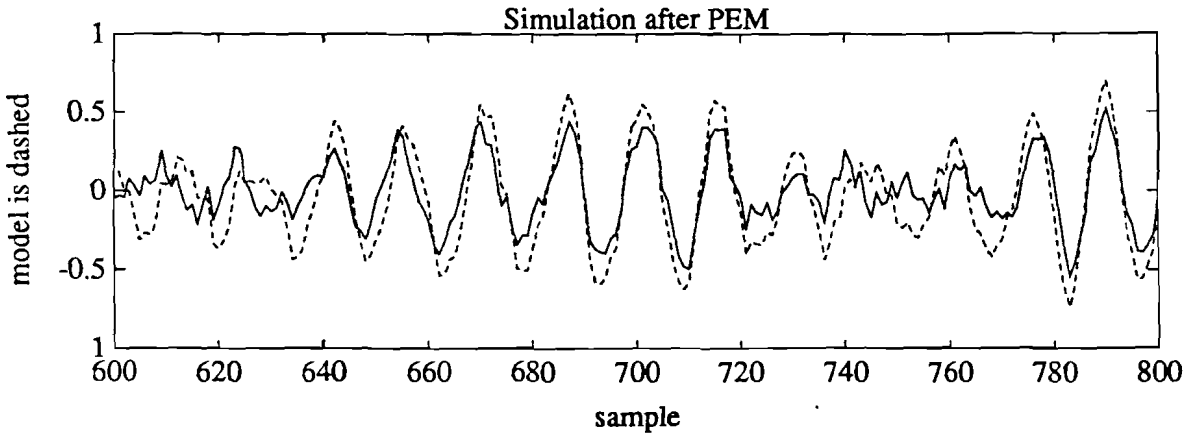


fig. 6.2 Simulation with prediction model

So here the importance of including OEM for this process has been emphasized. Applying OEM_s, an improvement can be obtained but because the gradient is not calculated correctly convergence is very slow.
Applying OEM_d the gradient is calculated correctly. Therefore the convergence rate increases tremendously (Also the obtained performance turned out to be better).
In the next picture for both static as well as for the dynamic algorithms the performance of the NN^3_{17171} as a function of the iterations have been plotted.

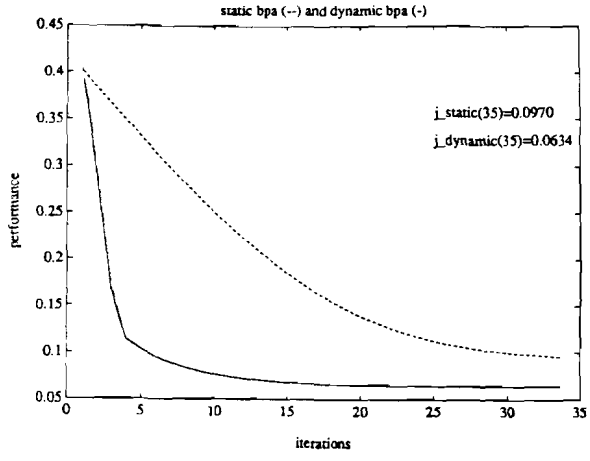


fig 6.3 Performance as a function of the iterations, for OEM_d and OEM_s

The results for as well the linear identification as the non linear identification have been depicted in the next table.

Linear 8th order model using Zhu[4]	J based on estimation set: 4.8%
NN^3_{17171} using OEM_s	J based on estimation set: 9.7 %
NN^3_{17171} using OEM_d	J based on estimation set: 6.3 %

All models have been estimated using the same dataset. (2000 samples input is uniformly distributed white noise. The amplitude of the input as well as the output is scaled within the interval <-1 1>.

Although the results using OEM_d are promising the performance of the obtained neural model is still worse than the linear model.
This is due to the fact that although the calculation of the gradient is quite complicated, the algorithm is very simple. More advanced algorithms, with respect to speed and robustness, are available. Those algorithms approximate the gradient numerically. Of course this approximation is not necessary because the gradients can be calculated using OEM_d. However because, at this stage, we just want to know whether modelling can be realized using neural nets, no effort is put in combing two methods yet.
The final model is obtained using the "Optimization Toolbox" of Matlab.
Using OEM_n with the same dataset, a performance of 4.7 % has been obtained. Using a longer data set of 5000 samples, improvement has been obtained (performance 2.7%)
The validation of the obtained model has been described in the section "validation". First has been described when the iterative search has to be stopped. A suggestion for structure estimation has been included.

Choosing number of iterations

The number of iterations need to find a good simulation model has to be determined. If the iterative search is continued untill no futher cnvergence can be obtained, the redundant parameters are tuned according to the specific disturbance sequence of the learning set. This model is optimal with respect to the estimation set. If another data set is used, the distributions are caused by a completely different noise sequence. The redundant parameters are not adjusted with respect to these disturbances. So the performance is worse.
A pragmatic solution to solve this problem is to calculate the performance of the model after each iteration using a validation set. The iterations must be stopped if no improvement of the performance, based on the validatiuon set, can be obtained. (see fig 6.4)
The final neural model of the acoustic process has been obtained using this method.

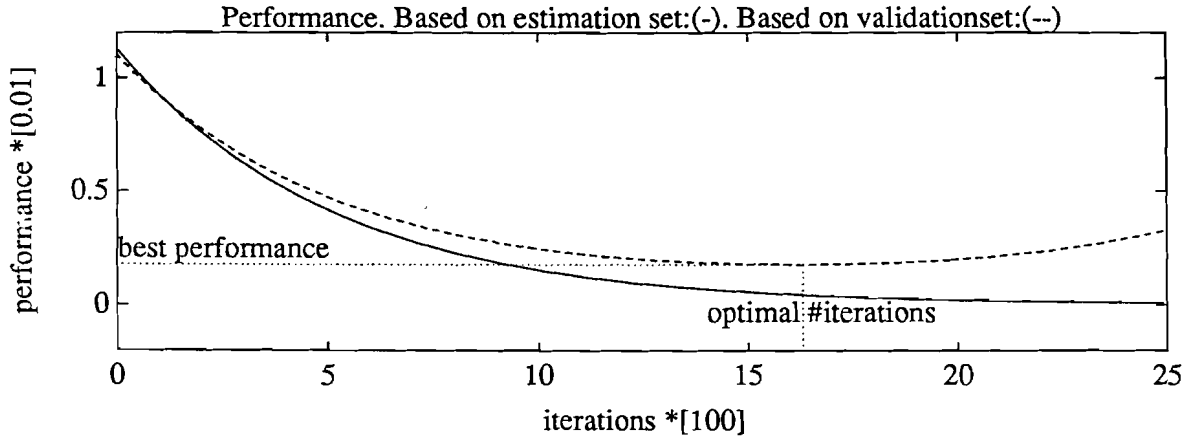


fig 6.4 Choosing the optimal number of iterations

This is not the optimal solution. Because of the contribution of the redundant parameters, the relevant parameters can not be tuned to the optimal value. It is better to remove the redundant parameters. A suggestion to obtain a neural net with less parameters has been included.

Suggestion to choose structure

Linear models do have 2 degrees of freedom in choosing model structure i.e. the number of AR parameters and the number of MA parameters. For multi layered neural nets there are 4 degrees of freedom i.e.:

the number of delayed inputs
the number of delayed outputs
the number of nodes in a hidden layer
the number of hidden layers.

Funashi[8] stated that for any $\varepsilon > 0$ there exist an integer i_1 such that a $NN^3_{I_0 I_1 I_2}$ can be tuned such that :

$$\max|F_t(\cdot) - \hat{F}(\cdot)| < \varepsilon$$

where:

$F_t(\cdot)$: *function which has to be identified*

$\hat{F}(\cdot)$: *Mapping from input to output described by the NN*

Therefore it seems logical to take only one hidden layer. This leaves only 3 degrees of freedom. The estimated order, determined using linear methods, can also be used for nonlinear modelling.
So only the number of hidden nodes has to be determined. If the process can be fitted exactly using X parameters, every extra parameter can be chosen to compensate, on the average for one disturbance sample. Adding one extra parameter the loss function will on the average decrease a factor 1/N (N = number of samples). The minimal number of nodes, necessary to model the process can be determined if first a net with only one processing element is tuned. A better simulation model can be obtained, if after adding one node (=processing element) the improvement is more than dX/N (performance based on estimation set) where:

dX: the number of the extra parameters after adding one hidden node
N : the number of data samples used for estimation

If the improvement is less than dX/N no nodes must be added anymore.

From theoretical point of view this method can be used. From practical point of view one should take into account the following:

- This method is time consuming for high order mimo system
- BPA is a simple algorithm, you never know if the global minimum is found

Therefore a pragmatic solution is just to try a number of hidden nodes. If the loss function, based on the validation set is approximately the same as the estimated power of the disturbance one can rely on the obtained model.

Validation

The NN^3_{17171} has been obtained using OEM_n. The model has been validated in the frequency domain as well as the time domain.

Validation in the frequency domain.

At the first stage, for creating a data set, both digital as well as analogue measurements have been carried out.
The digital data set has been used for parameter estimation and validation in the time domain. The analogue measurements can be used for validation in the frequency domain

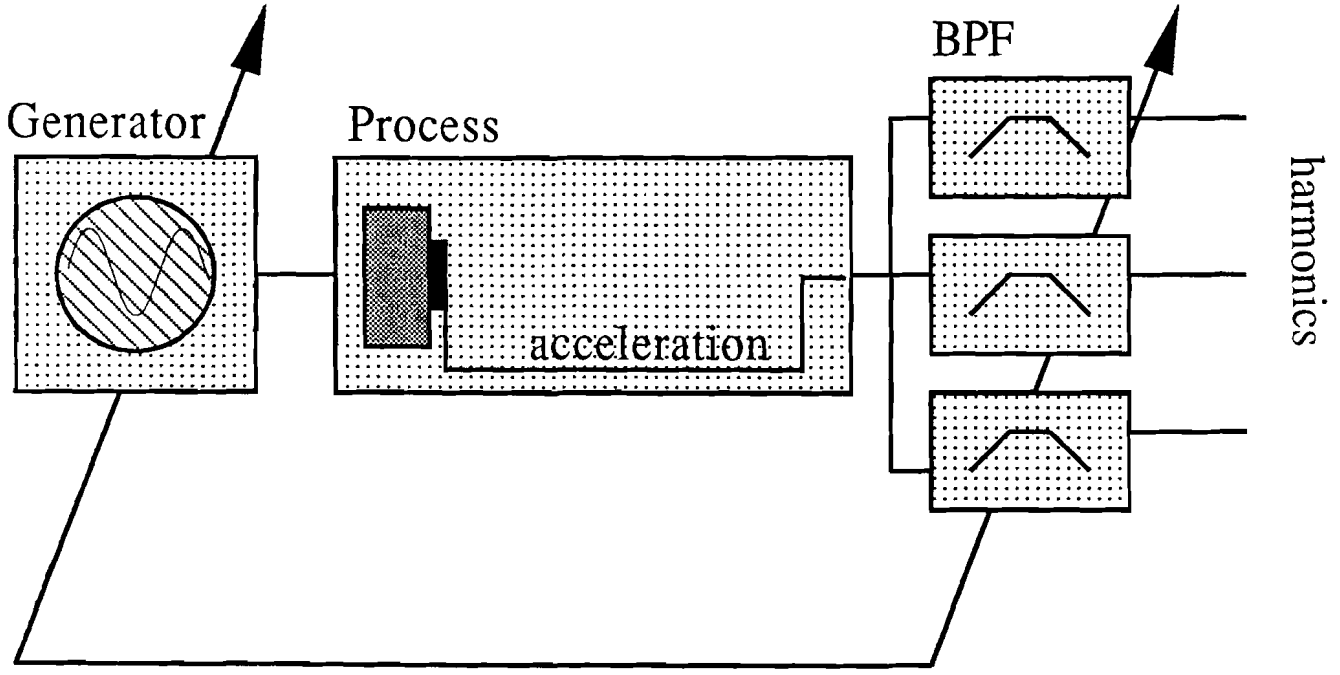


fig 6.5 analogue measurement of higher harmonics

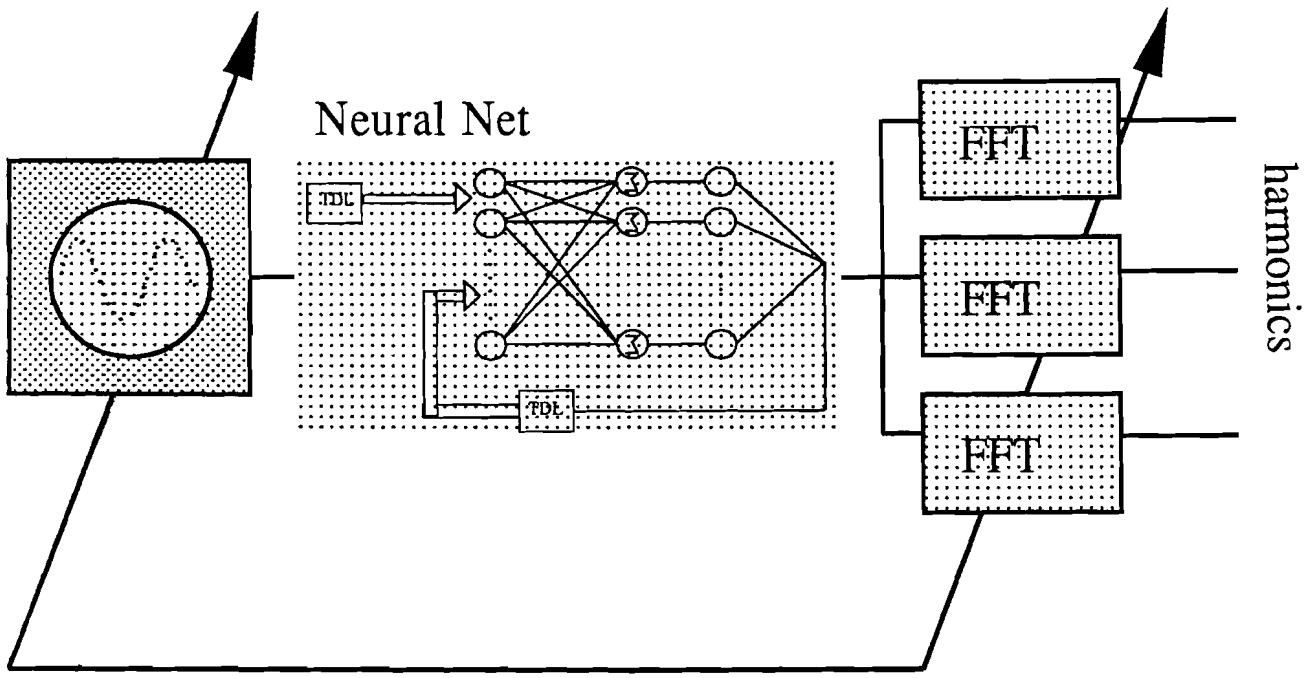


fig 6.6 digital measurement of higher harmonics generated by the model

During analogue measurements the process is excited by a sinus function for which the frequency increases slowly. The output is measured using band pass filters which are adjusted according to the current frequencies (see fig 6.5).

Determination of the higher harmonics of the estimated model is similar to the analogue measurements; excite the model with a sinus function and calculate the FFT components of the output according to the input frequency (see fig 6.6). The validation has been depicted in fig 6.7. (Take into consideration that the neural net models prefilters and A/D D/A converters while during analogue measurements only the real process is measured). Both frequency plots can be seen in fig 6.7.

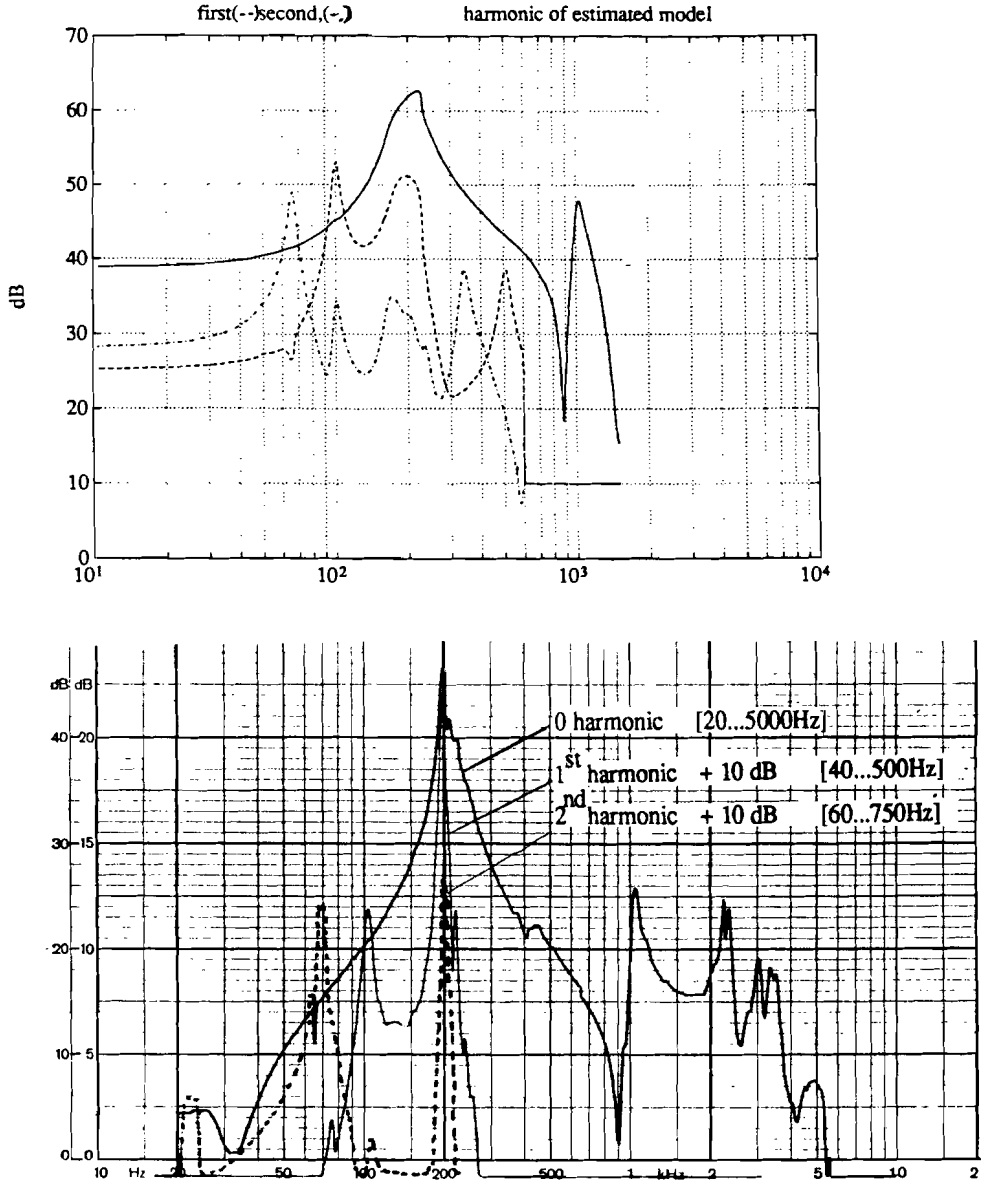


fig 6.7 validation in the frequency domain

As can be seen the lower frequencies are badly estimated. This is due to the fact that the lowest frequencies are badly represented in a data set. Therefore the errorsurface concerning the parameters which represent the lower frequencies is very flat.
Solution:
- taking a longer data set
- increase number of iterations

For frequencies higher than 100 Hz the model is a good representation of the process.

Validation in the time domain

The neural net model has been validated in the time domain. In fig 6.8 a part of the estimation set has been depicted. The performance based on the whole estimation set (5000 samples) is 2.9 %. In fig 6.9 a part of the validation set has been depicted. The performance based on the whole validation set (4999 samples) is 4.7%

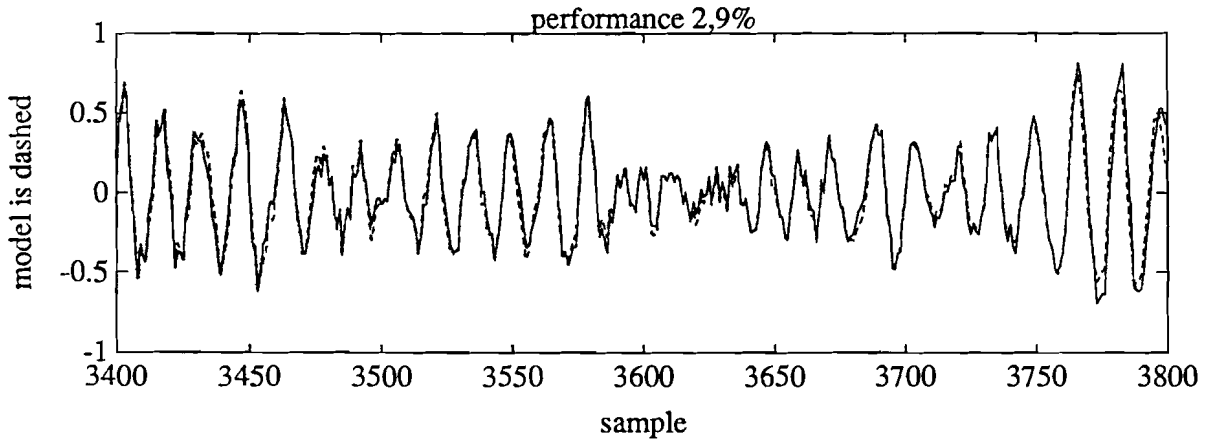


fig 6.8 Dashed is simulated model output
Solid is estimation set.

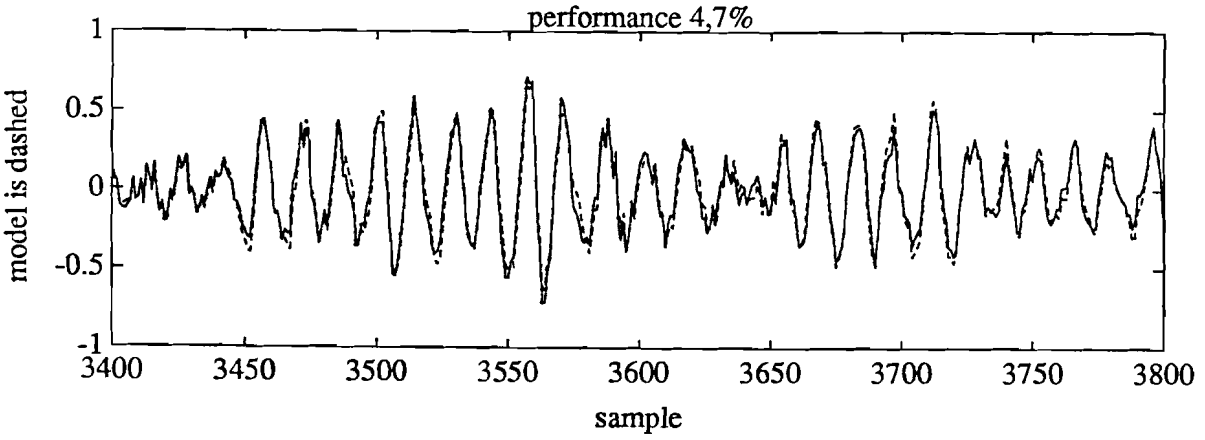


fig 6.9 Dashed is simulated model output
Solid is validation set.

From both validations one could conclude that a neural net is capable to simulate a badly damped system with complicated nonlinearities.

7 CONTROLLER DESIGN.

Linear controller design

As has been said in section 2, controller design can be divided into two parts: a linear and a nonlinear part. First a linear LQG controller has been designed based on the linear model.
A state space representation of the system has been given in fig 7.1.

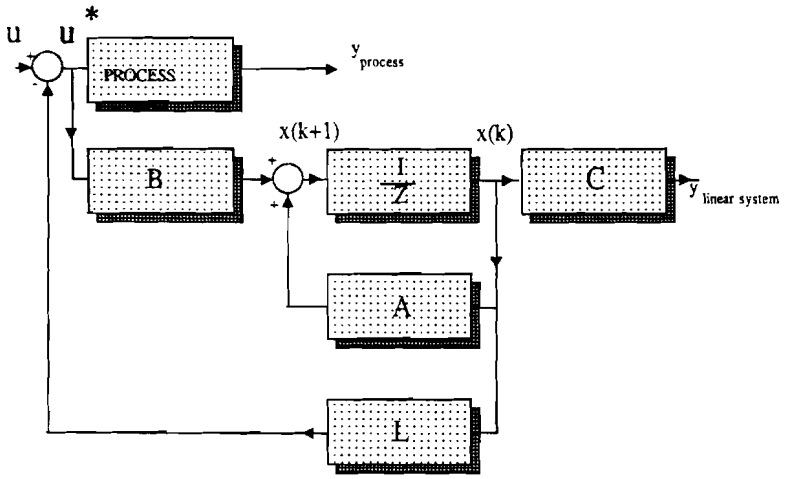


fig. 7.1 State space description with fullstate feedback

Full state feedback, brought about by L, is determined optimizing the following criteria (see [7A]).

$$\text{for SISO sytems:}$$

$$V_c = \sum_{k=1}^{\infty} [x^T(k) Q x(k) + u^T(k) R u(k)]$$

x = vector containing the estimated states of the process.
Q = matrix with weights for the states.
R = weight for input (scalar). [7A]

If $Q = C^T R_0 C \rightarrow$

$$V_c = \sum_{k=1}^{\infty} [y R_0 y + u(k) R u(k)]$$

R_0 = weight for the output (scalar).

L has to be designed such that a small response time of the controlled system is obtained. This can be realised by keeping the output y small. Taking Q to be $C^T R_0 C$ the output y is weighted.
Choosing R to be large results in low excitation of the inputsignal u*.
Choosing R_0 to be large results in small outputsignal, or short settlingtime. The fastest response time is obtained if the transfer function, of the controlled system, is flat. This can

be obtained by choosing a large R_o. However one has to take into account the input range. The system is modelled for signal amplitudes in between the interval <-1 1>. The input-signal u, and the vector L must be chosen such that the amplitude of u^* never reaches beyond the interval <-1, 1>. If not the model is (due to nonlinearities) not valid anymore .

Because no time was left solely an optimal L has been calculated. Not implementing a Kalman gain, results in a feed forward controller. This controller is easy to design and easy to implement.
The state space representation of the feed forward controller can be derived from fig. 7.1.

$$\textit{state space description for transfer } \frac{y_{lin.}}{u} =$$

$$\begin{aligned} x(k+1) &= (A - BL)x(k) + Bu(k) \\ y_{lin}(k) &= Cx(k) + Du(k) \qquad (D = 0) \end{aligned} \qquad [7B]$$

$$\textit{state space descripton for transfer } \frac{u^*}{u} =$$

$$\begin{aligned} x(k+1) &= (A - BL)x(k) + Bu(k) \\ u^*(k) &= -Lx(k) + u(k) \end{aligned}$$

A block scheme, according to the last formula has been given in the next figure.

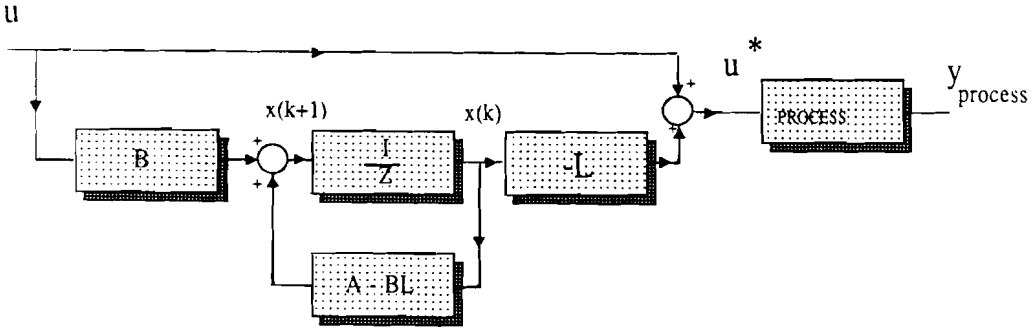


fig 7.2 Feed forward controller based on LQG

If u is in the interval <-1, 1> only the dominant poles can be removed a little bit away from the unit circle, without exciting the transducer beyond <-1, 1>. The next figure depicts the pole- zero plots of the linear- and the controlled model.

The controller has been designed using the following weights:

R=3
R_o=20

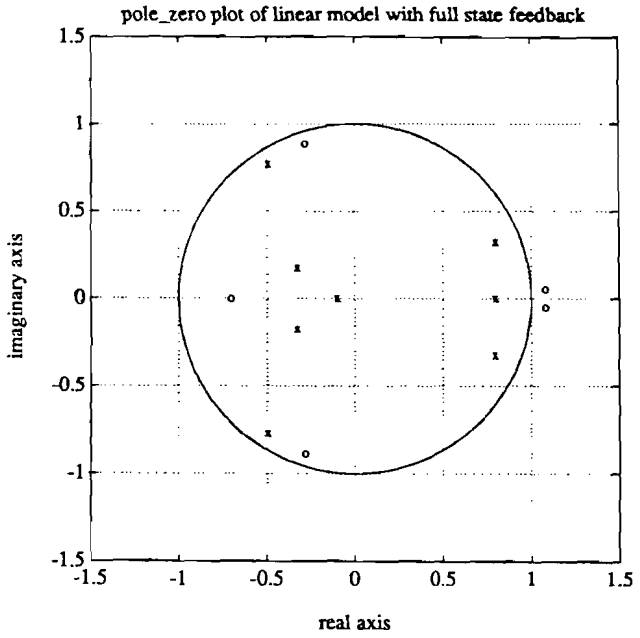


fig. 7.3

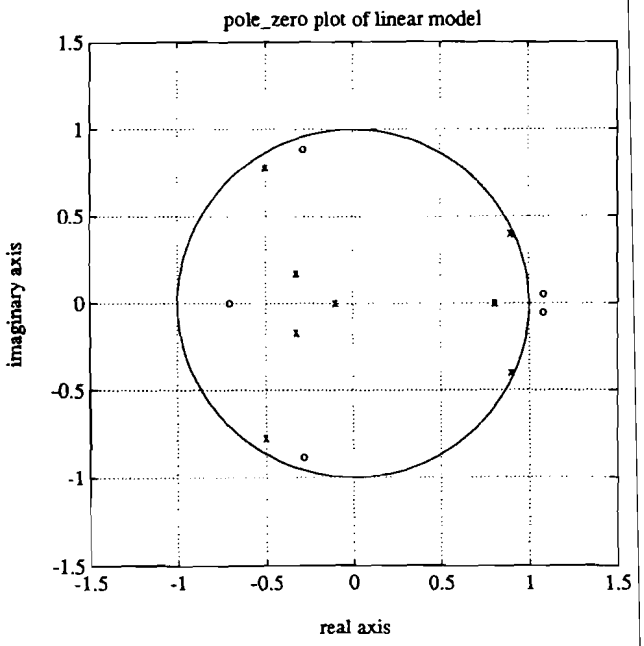


fig. 7.4

As can be seen in the pole-zero plots, some improvement has been obtained, but the transfer is not flat. For both the model as well as the model with fullstate feedback, the transfer has been plotted in fig. 7.5.

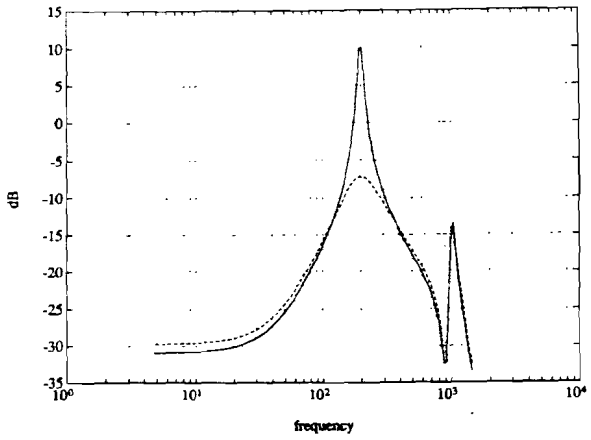


fig 7.5 Spectrum linear model (solid)
Spectrum linear model with full state feedback (dashed)

A flat spectrum can be obtained if all the zeros in the unitcircle are cancelled by poles. The number of zeros outside the unitcircle, determine the overall performance, because they can not be cancelled (cancelling these zeros results in an unstable controller).

Improvement, with respect to fig 7.5, can only be obtained if less weight is put on the input u. Taking care that the transducer is not "over excited", the input power must be decreased (decrease amplitude of u).
In the next figure the pole_zero plot and the transfer of the controlled system has been given,

based on weights:

$R = 1\ e\text{-}5.$

$R_o = 20.$

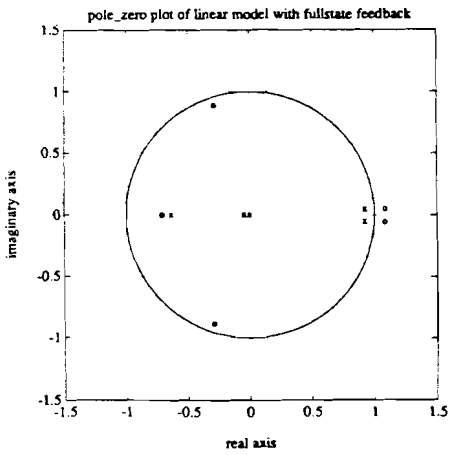


fig. 7.6

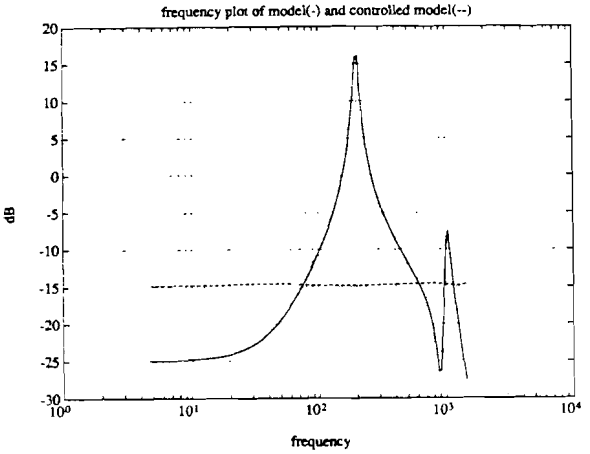


fig. 7.7

As can be seen all the frequencies below 70 Hz will be amplified, while input frequencies between 70-600 Hz will be quenched, to result in a flat spectrum of the overall system. Because hardly any penalty is on the input signal, u has to be decreased such that the amplitude is in between <-.3 .3>. Only then u^* is in between <-1 1>.

To increase the maximum interval of u, one can filter the lowest frequencies, inaudible to the human ear. The overall feedforward controller, connected to the real process, has been depicted in the next figure.

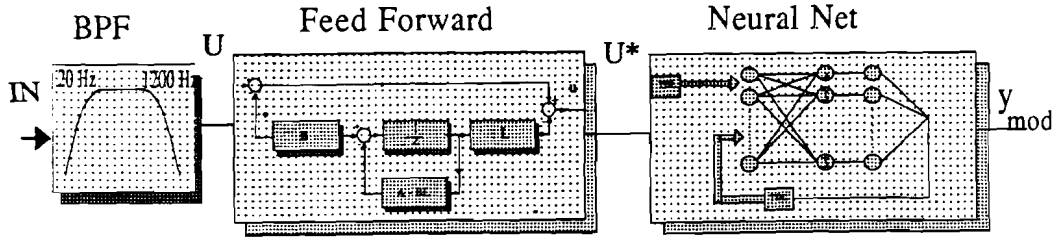
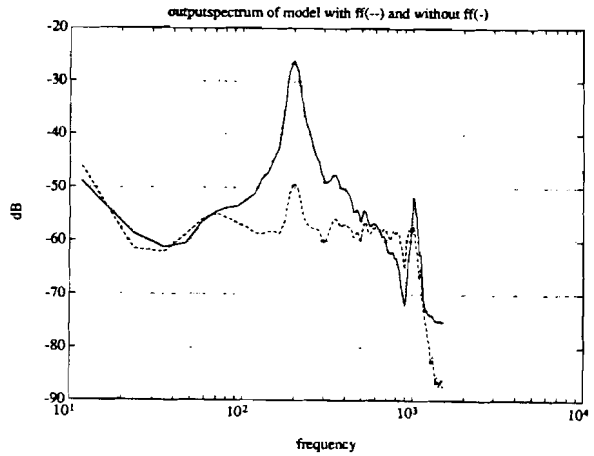


fig 7.8 Feedforward controller plus band pass filter.

Filtering more lower frequencies, the input signal "in" can be chosen in a broader interval without "over exciting" the transducer. In the next table, the maximum amplitude for the inputsignal "in" has been given for several band pass filters.
(The lowest frequencies audible for human, is 20 Hz).

band pass	max "in"
0Hz - 1200Hz	0.33 Volt
20Hz - 1200Hz	0.40 Volt
50Hz - 1200Hz	0.43 Volt

The feedforward controller has been based on the linear process approximation. Of course, if the real process is connected the transfer is not as flat as has been depicted in fig 7.5. In the next picture two different power spectra have been plotted. The solid line depicts the spectrum of the output of the neural net if excited with a white noise sequence, filtered by a band pass filter (20 Hz 1200 Hz). The dashed line depicts the spectrum if apart from the band pass filter, the feedforward controller has been used.



Output spectrum of nn without feed forward controller(solid).
Output spectrum of nn with feed forward controller (dashed)
fig 7.9

As can be seen in fig 7.9 quite an improvement has been obtained. Adding the feedforward controller, the nonlinear controller is to be expected less complicated.

Nonlinear controller design

In this section a configuration has been depicted which can be used to obtain a controlled system using a neural net as a feed back controller. The objective is tracking the output generated by the feedforward controller in series with the linear model. A block scheme of the overall system has been given in the next picture.

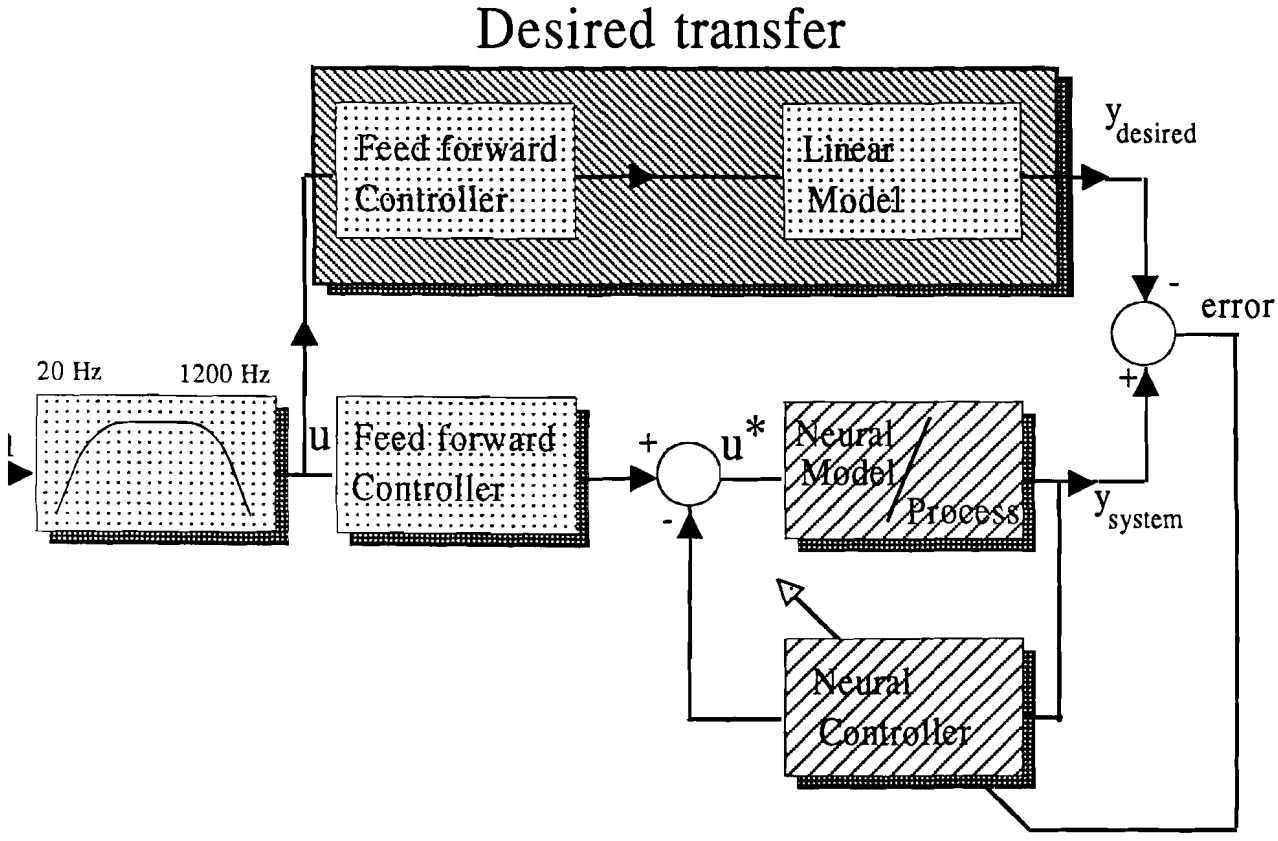


fig 7.10 block scheme of the overall system

The spectrum of the desired transfer is approximately flat in the frequency band of interest. Because the desired output is generated by linear models, no nonlinearities will be present. So if the controllers are able to simulate the desired output, $y_{desired}$, we achieve the objective, stated in the first section of this report (a flat transfer function, no nonlinearities).
Of course, numerous other desired transfers can be chosen, but the advantage of this configuration is the fact that the feedback controller becomes less complicated in case the linear model is more able to identify the process (if model = process the feed back controller can even be omitted).
The parameters of the controller can be optimized using OEM_d
For a somewhat simpler configuration the back propagation algorithm has been depicted in appendix 3.
If the process can be modelled exactly by the linear model, the feed back can be omitted.

8 CONCLUSIONS AND RECOMMENDATIONS

Conclusions

An acoustic process has been identified using linear as well as non linear models. Two methods have been applied to tune the parameters of a linear model. An artificial neural net has been applied in order to simulate nonlinear behaviour. Conclusively could be stated:

-Due to hardware limitations, it is difficult to identify a badly damped system using the method proposed by Backx[3].

-Linear identification method, proposed by Zhu[24] can be used to identify badly damped systems.

-An artificial multilayered neural net can be used to model the acoustic process. The model performs good in the time domain as well as in the frequency domain.

-It has been shown that the neural net is capable to estimate nonlinearities but:

> -Static Back Propagation is not sufficient if a badly damped system has to be identified using a neural net, as it does not yield the real gradient of an output error criterion.
>
> -Using Dynamic Back Propagation, an accurate approximation of the real gradient of an output error criterion is obtained.
>
> -Steepest decent algorithm has then to be combined with another algorithm (for example a line search algorithm) to get faster convergence.

Recommendations

-Optimization of the search algorithm, using the gradients calculated by dynamic back propagation.

-Parameter reduction. A lot of the parameters in the neural net are redundant. Due to these parameters the optimization is more difficult. Improvement can be obtained if those parameters are removed.

-Controller design using a neural net controller in close loop. The parameters can, in principle, be tuned using the dynamic back propagation algorithm. Because this software is available now, implementation of the algorithm, which can be used to tune the controller parameters, is not that difficult any more.

9 REFERENCES

[1] ANTSAKLIS, P.J.
Neural networks in control systems
IEEE Trans. on Syst. Mag.
April 1990 p.3-5

[2] BARTO, A.G., et al.
Neuronlike adaptive elements that can solve difficult learning control problems.
IEEE trans. on syst. man and cyb. vol. SMC-13(1983), no.5 1983 p.834-846

[3] BACKX, A.C.P.M.
Identification of an industrial process: A markov parameter approach.
Phd thesis, published in own control of the Eindhoven University of Technology.
1987

[4] BAVARIAN, B.
Introduction to neural networks for intelligent Control
IEEE control syst. mag.
April 1988 p.3-7

[5] BLEULER, H., et al.
Non lineair neural network control with application example.
INNC vol.1
1990 p.201-204

[6] CHEUNG, J.Y. and R.J. MULHOLLAND
Using a neural network as a feedback controller.
Proc. of the 32nd Midwest Symp. on Circuits and Systems
1990 p.752-755

[7] CHOUIKHA, M.F. and C. AISSI
Design & implementation of an intelligent controller using neural networks
Proc. of the 32nd Midwest Symp. on Circuits and Systems
1990 p.756-759

[8] FUNAHASHI, K.
On the approximate realization of continuous mapping by neural networks.
Neural Networks vol.2
1989 p.183-192

[9] HUNT, K.J. and D. SBARBAVO
Neural networks for nonlineair internal model control
IEE proc.-D vol. 138, no.5
1991 p.431-438

[10] ICHIKAWA, Y. and T. SAWA
Neural network application for direct feedback controllers.
IEEE trans. on Neural Networks, vol.3 no.2
1992 p.224-231

[11] IIGUNI, Y., et al.
A non lineair regulator design in the presence of systems uncertainties using multilayered neural networks.
IEEE trans. on Neural Networks, vol.2 no.4
1991 p.410-417

[12] NARENDRA, K.S. and K. PARTHASARATHY
Identification and control of dynamical systems using neural networks.
IEEE trans. on Neural Networks, vol.1, no.1
1990 p.4-27

[14] NARENDRA, K.S. and K. PARTHASARATHY
Gradient methods for the optimization of dynamical systems containing neural networks.
IEEE trans. on Neural Networks, vol.2, no.2
1991 p.552-262

[15] PIJNENBURG, J.L.C.M.
Non linear system identification using neural networks
M.Sc Thesis Eindhoven University of Technology.
April 1992

[12] NGUYEN, D.H. and B. WIDROW
Neural networks for self learning control systems.
IEEE Control Syst. Mag.
April 1990 p.18-23

[16] PSALTIS, D., et al.
Neural controllers
Proc. Int. Conf. on Neural Networks vol.4
1987 p.551-558

[17] PSALTIS, D., et al.
A multilayered neural network controller.
IEEE Control Syst. mag. April
1988 p.17-21

[18] SAERENS, M., SOQUET, A.
Neural controller based on back propagation
IEE proc-F vol.138, no.1
1991 p.55-62

[19] VELDEN, J.A.J.
The application of a system identification technique for modelling a non linear acoustic transducer.
M.Sc Thesis Eindhoven University of Technology.
June 1992

[20] WERBOS, P.J.
Backpropagation and neurol control
Proc. Int. Conf. on Neural Networks vol.1
1989 p.209-215

[21] WERBOS, P.J.
Neural Networks for control and system identification.
Proc. 28th Conf. on Decision and Control
1989 p.260-265

[22] WERBOS, P.J.
An overview of neural networks for control
IEEE Control Syst. Mag. Jan.
1991 p.40-41

[23] WILLIS, M.J., et al.
Artificial nerural networks in process engineering
IEE proc.-D vol.138, no.3
1991 p.256-266

[24] ZHU, Y.C., et al.
Multivariable Process Identification for Robust Control.
EUT Report 91-E-249.
Jan. 1991

APPENDIX 1

Usually neural nets are excited with a random input sequence. In this example, however, the time has been used as the input of the net. This example has been included to illustrate how a neural net can be used to model a arbitrary continuous function.

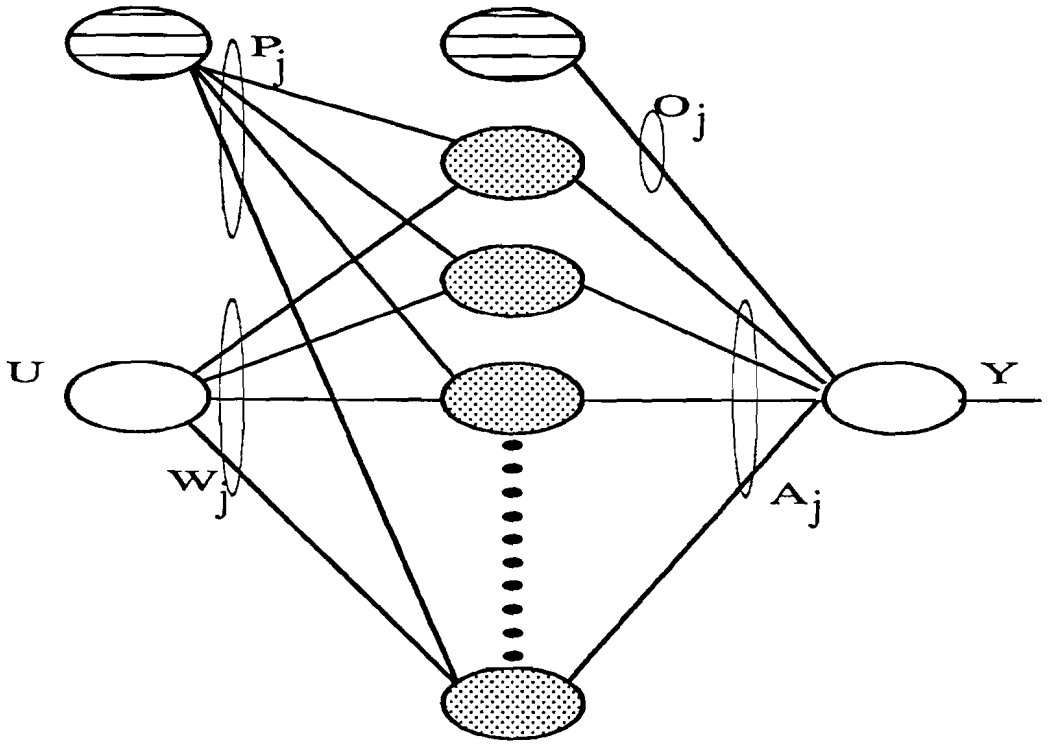


fig 1 Static neural net with one hidden layer.

The functional description of a 3 layered neural net which has been shown in the picture above is:

$$y(k) = \sum_{j=1}^{N_{nodes}} [A_j F(W_j * k + P_j] + O_j \qquad [C]$$

N_{nodes} = *number of hidden layers*
$F(.)$ = *transerfunction*: tanh(.)
k = *input of the net*
$y(k)$ = *output of the net*

It has been proven that a 3 layered neural net with an undetermined number of nodes can simulate, to any degree of accuracy, a continuous mapping from input to output. For neural nets with one input and only one output this can be easily seen. Referring to fig 1

we choose the parameters to be:

- $W_j = 2\pi j/N$.
- $P_j = \phi[j]$.
- $A_j = A[j]$.

Choosing a sine to be the transfer function, the following can be derived.

$$
\begin{aligned}
y[k] &= Offset + \sum_{j=1}^{\frac{1}{2}N-1} A[j]\sin\left(\frac{2\pi j}{N}k + \phi[j]\right) \\
&= Offset + 2\sum_{j=1}^{\frac{1}{2}N-1} G_r[j]\cos\left(\frac{2\pi j}{N}k\right) + 2\sum_{j=1}^{\frac{1}{2}N-1} G_i[j]\sin\left(\frac{2\pi j}{N}k\right) \\
&= \sum_{j=0}^{N} G[j]W_N^{-jk}
\end{aligned}
$$

[D]

$$
\begin{aligned}
N &= Even \\
G[j] &: Discrete\ Fourier\ Components \\
G_r[j] &: \Re\{G[j]\} \\
G_i[j] &: \Im\{G[j]\} \\
W_N^{-jk} &: e^{-j2\frac{\pi}{N}k}
\end{aligned}
$$

$$
\begin{aligned}
A[j] &= \sqrt{G_r[j]^2 + G_i[j]^2} \\
\phi[j] &= \arctan\left(\frac{G_i[j]}{G_r[j]}\right)
\end{aligned}
$$

So using a sine for the transfer function of the hidden nodes, the parameters in the neural net can be related directly to the discrete Fourier components. So a neural net with ½N-1 nodes in the hidden layer can be tuned in such a way that the N equal distributed points match exactly the Fourier coefficients of the signal in these points. Increasing N to a sufficient amount, results in exact fit of the whole function, also in between two consecutive points. In principle all kind of processing elements can be used for simulating a mapping from input to output. For neural net applications one usually takes a sigmoid function (a sigmoid function saturates at a certain fixed value. Therefore it is easier to model a damped function). It is not proven yet what function gives the best performance, but knowing that biological neurons have sigmoid like transfer functions, it seems quite logical to take these functions for artificial neural nets.

APPENDIX 2

Every transfer function $H(z^{-1}) = \frac{NUM(z)}{DEN(z)}$ can be written as :

$$\sum_{i=1}^{L} \frac{A_i^o z}{z - \lambda_i^o} + \sum_{i=1}^{K} \frac{B_i^o z + C_i^o}{z^2 - 2\sigma_i^o z + (\sigma_i^o)^2 + (\omega_i^o)^2}$$

L: number of real poles
K: number of complex pole pairs

The superscript ° indicates that the parameter is based on the old sample frequency T^o. The new frequency is T^n

For real poles $\forall i$:

$$\frac{A^o z}{z - \lambda^o} = \frac{A^o z}{z - e^{aT^o}}$$

this equation is valid if : $a = -\frac{1}{T^o}\ln(\lambda^o)$

"a" has to remain the same $\Rightarrow$ $\lambda^n = e^{-aT^n}$

$$= (\lambda^o)^{\frac{T_n}{T_o}}$$

For complex poles $\forall i$:

$$\frac{B^o z + C^o}{z^2 - 2\sigma^o z + (\sigma^o)^2 + (\omega^o)^2} = D\left[\frac{z + e^{-aT^o}\sin(fT^o) - e^{-aT^o}\cos(fT^o)}{z^2 - 2ze^{-aT^o}\cos(fT^o) + e^{-2aT^o}}\right]$$

this equation is valid if:

$$D = C^o e^{-aT^n}[(\sin(fT^n) - \cos(fT^n)]$$

$$a = -\frac{1}{2T^o}\ln((\sigma^o)^2 + (\omega^o)^2)$$

$$f = \frac{1}{T^o}\arccos(\sigma^o e^{aT_o})$$

where ;

σ^o : σ *base on old sample frequency*

ω^o : ω *based on old sample frequency*

T^o : *old sample frequency*

f : *frequency, has to be remain the same*

a : *damping, has to remain the same*

substituting T^n *and keeping "a" and "f" constant results in:*

$$B^n = B^o$$

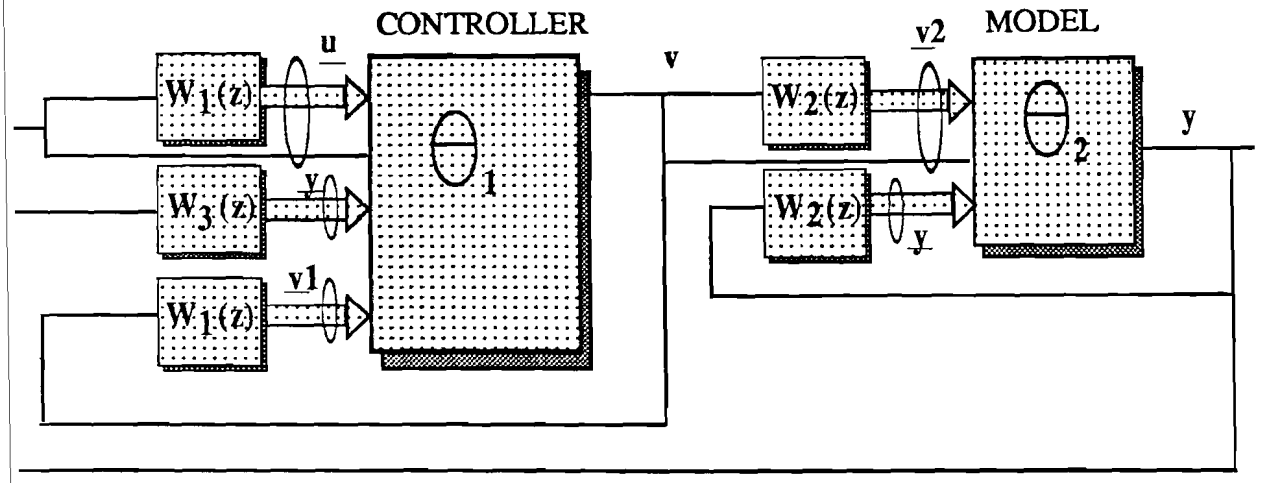
$$\sigma^n = e^{-aT^n}\cos(fT^n)$$

$$C^n = B^n[e^{-aT^n}\sin(fT^n) - e^{-aT^n}\cos(fT^n)]$$

$$(\omega^n)^2 = e^{-2aT^n} - (\sigma^n)^2$$

using $(\cdot)^n$ *the new transfer can be calculated*

APPENDIX 3



neural net in close loop configuration

$$v = N_1(\underline{u}, \underline{y}, \underline{v}, \underline{\theta}_1)$$
$$y = N_2(\underline{v}, \underline{y}, \underline{\theta})$$
$$W_1(z^{-1}) = \left[z^{-1}\ z^{-2}\ \ldots\ z^{-n_1}\right]^T$$
$$W_3(z^{-1}) = \left[z^{-1}\ z^{-2}\ \ldots\ z^{-n_3}\right]^T$$
$$n_1 = \textit{order of controller}$$
$$n_3 = \textit{number of delays fed back from output to controller}$$

$$[1]\quad \frac{dy}{d\theta_1} = \frac{dy}{dv}\frac{dv}{d\theta_1}$$

$$\frac{dv}{d\theta_1} = \frac{\partial N_1}{\partial \underline{y}^T}\frac{d\underline{y}}{d\theta_1} + \frac{\partial N_1}{\partial \underline{v}_1^T}\frac{d\underline{v}_1}{d\theta_1} + \frac{\partial N_1}{\partial \theta_1}$$

$$[2]\quad = \frac{\partial N_1}{\partial \underline{y}^T}W_3(z^{-1})\frac{dy}{d\theta_1} + \frac{\partial N_1}{\partial \underline{v}^T}W_1(z^{-1})\frac{dv}{d\theta_1} + \frac{\partial N_1}{\partial \theta_1}$$

substitute [2] *in* [1]:

$$\frac{dy}{d\theta_1} = \frac{dy}{dv}\left[\frac{\partial N_1}{\partial \underline{y}^T}W_3(z^{-1})\frac{dy}{d\theta_1} + \frac{\partial N_1}{\partial \underline{v}^T}W_1(z^{-1})\frac{dv}{d\theta_1} + \frac{\partial N_1}{\partial \theta_1}\right]$$

The partial derivatives can be calcuted using the sensitivity net, discussed in section 6.

The derivatives $\frac{dy}{dv}$, $\frac{dy}{d\theta_1}$ *and* $\frac{dv}{d\theta_1}$ must be calculated using dynamic BPA.