Floating platform control, based on black and white neural states

van Doorn, Henk C.K.

Award date:
1997

Eindhoven University of Technology
Department of Electrical Engineering
Measurement and Control Group

Floating platform control, based on black and white neural states

by H.C.K. van Doorn

M.Sc. Thesis
carried out from september 1996 to august 1997
commissioned by prof.dr.ir. P.P.J. van den Bosch
under supervision of dr.ir. A.A.H. Damen
date: 27 august 1997

We are such stuff as dreams are made on, and our little lives are rounded with a sleep.

- William Shakespeare

Abstract

The floating platform is a mechanical construction consisting of a triangular body with three floats attached to it. This platform lies in a squared tub filled with water. On this platform, a crane has been mounted, which acts as a disturbance on the system. With the help of three servo systems it is possible to change the height of the three pillars, which is the connection between the platform and the floats.

The main goal of this thesis work is to develop a controller which should meet our requirements. Generally this means that the controller should be able to suppress the wave disturbances, acting on the platform as quick as possible.

To estimate a proper nonlinear model for using it in a model based controller, and for developing such a model based controller, we tried this by use of neural networks. These networks are able to capture nonlinear behaviour. We used a state-space structure, implemented in a multy layer perceptron. Singular value decompositions and loss functions showed us that a state dimension of seven was the best choice.

The estimation results were satisfying, although the model predictions were biased. This bias is caused by standing and/or reflected waves and behave chaotic. This means only in a small horizon after excitation these waves are correlated with the input signals, strongly dependent on initial conditions.

We tried to include these waves in the model by means of a dynamic nonlinear filter, for short horizon predictions. Singular value decomposition of the output error, showed us that the waves can be modelled by a fourth order model. Two methods have been used: one structure with fixed output error, fed back to the inputs and one structure with variable output error, which should be minimized also. After some stability problems, the first method gave us good results, the bias disappeared almost totally. The second method has not produced any good results yet, due to some problems with the gradient calculation in the quasi-Newton optimization procedure.

Finally we can conclude that neural networks applied in a dynamic state-space structure are very usefull for modelling nonlinear dynamical systems.

Acknowledgements

During a year I completed my thesis work in the Control and Measurement group of the department of Electrical Engineering, Eindhoven University of Technology. In this year I have been involved in several different activities. These activities have, especially in the beginning of yhis year, put a strain on my thesis work. Therefore, first of all, I would like to express my gratitude to my coach, dr.ir. Ad Damen for his patience and explanations during my thesis work. These explanations helped me to get insight in many aspects of the problem. In addition I would like to thank prof. dr. ir. Paul van den Bosch for offering me this great opportunity to finish my study in his group.

Furthermore, I like to thank my family for their support, my friends and inmates for their companionship and my house mates for having such a great time in Eindhoven. Also I like to thank all active members of the IEEE Student Branch Eindhoven, making my board year, but most of all, my thesis work a success.

Henk van Doorn

Contents

List of used symbols

X_i	servo inputs
H_a	average height of platform [m]
θ_y	rotation of platform around y-axis
θ_x	rotation of platform around x-axis
H_i	pillar heights for i=1,2,3
Δ_i	average wave height around float *i*
L	horizontal distance between centre of pillar and centre of platform
L_o	horizontal distance between sensors and centre of platform
F_g	gravity force [N]
F_b	buoyancy force [N]
A	surface [m^2]
τ	torque [Nm]
J	inertia
F_{ci}	forces induced by crane actions [N]
$\underline{x}$	input vector for neural network
θ	weight vector
w	weight
d	bias
u	summation of weighted inputs
y	output of activation function
ϕ	activation function
J	cost function
$\underline{d}$	desired output vector for neural network
∇	gradient
M	inverse approximation of Hessian
H	Hessian
f	dynamic neural network, state space structure
h	output map
Σ	nonlinear state space model parametrization
Γ	Jacobian matrix
Φ	correlation function
$\underline{x}^1$	known part of state vector
$\underline{x}^0$	unknown part of state vector

Chapter 1

Introduction

At the Measurement and Control group of the Department of Electrical Engineering at the Eindhoven University of Technology a continuous research for identification and control of processes takes place. The newly developed identification and control theories are tested on laboratory processes.

One of these processes is the floating platform. This platform consists of a triangle frame, floating on three adjustable floats placed in a large tub, filled with water. The vertical distances between the frame and each float are controllable by servo systems. In the centre of the triangle frame a rotatable crane will be mounted which can pick up, move along the arm of the crane and rotate a load. At several places on the platform height sensors are mounted for two different kind of purposes.
The main measurement is used to measure the absolute height. Therefore the height of the platform to the waterlevel in a bucket fixed to the bottom of the tub is measured. This level is considered to be stationary and not influenced by the waves in the tub. The other kind of measurement is used for measuring the average wave height around the floats. This will change quickly by wave disturbances and is proportional to the buoyancy force acting on the float. This will give us the opportunity to predict the vertical movement of the float caused by waves.

Due to the waves in the tub, introduced by the movements of the floats, and to the movements of the crane the platform won't stay in a strictly horizontal position. The goal is to design a controller which keeps the platform horizontal at an average height. Several years of study have been spent to the floating platform now, leading to the conclusion that the platform is a nonlinear Multiple Input Multiple Output (MIMO) process. This nonlinearity is mainly caused by reflected and/or static waves and these are hard to model due to their chaotic behaviour. The attempts to model the process so far were based on linear identification and control techniques. The problem with such models is that they can't predict these nonlinear wave disturbances. This causes the model based controllers, developed so far, to give only marginal results.

In this thesis we will describe our approach to model and control the platform by use of nonlinear techniques, which should be able to predict the nonlinear behaviour of the platform movements and the wave disturbances. We did this by use of neural networks. These networks, whose design was inspired by biological neural networks, are able to capture any kind of (smooth) nonlinear predictive behaviour. Several structures are available. We used the most commonly used structure, the multi layer perceptron (MLP). We followed the path taken by Jozef Mazak ([MAZ96]), who used a nonlinear copy of the Linear Quadratic Regulator theory. Here the model, the Kalman gain

and the feedback matrix are replaced by neural networks.

This thesis is divided into eight chapters. After the introduction, in chapter two the platform and the used tools will be described. Chapter three will show us a basic description of neural networks, their training tools and their practical use in control theory. Chapter four describes the way(s) we identify the process and the results are discussed here. In chapter five the development of the observer is written and practical problems which occured during this stage are discussed. In chapter six the design of a controller is described. Unfortunately, due to a lack of time we haven't finished this stage. Finally, in chapter seven the results of this thesis work are discussed and some recommendations for further improvement will be given.

Chapter 2

Description of used tools

2.1 The floating platform

The platform is a stiff construction, consisting of a frame, a crane and three identical floats, which are placed in a tub (figure 2.1). The body of the platform has an equilateral triangle shape with side lengths of 0.97 [m], see figure 2.2. The platform leans on three floats, which are placed in the corners at a distance L (= 0.45 [m]) from the platform centre. The floats are made of tempex and have diameters of 0.46 [m] and heights of 0.25 [m]. Around the floats sensors are placed to measure the wave heights. In static situation, about 2/3 of the float is under the water line. In the middle of each side, opposite of the floats, arms are stretched out, which are holding the height sensors. The platform is affected by two kinds of forces: The gravity force and the upward or buoyancy force. The gravity force attacks on the middle of the platform (the centre of gravity, Mp) and on each float (Mf) and is pointed towards the earth.

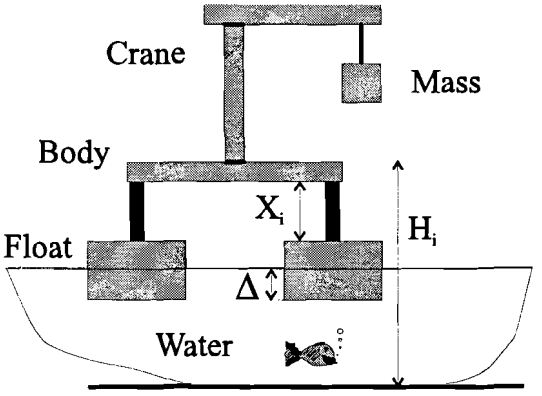

Figure 2.1: *Front view of the platform with the crane.*

The buoyancy force is effected by the water and are exerted, to the bottom of the floats. According to Archimedes, this force is equal to the displacement of the water. If a float is moving, there is also a damping and inertia force.The damping force and inertia force are always directed to the opposite of respectively the movement and the acceleration of the float. The

stiffness of the platform causes that if one upward force is changing, for example by a wave, the platform tries to rotate around a horizontal axis through the center of gravity. Then the depth of the other two floats will change too.

When a wave passes a float, it will lift and lower the float. It introduces alternating larger and smaller upward forces which disturb the equilibrium of the platform. By controlling the distances X_i between the platform and the floats, the influence of the disturbances can be reduced.

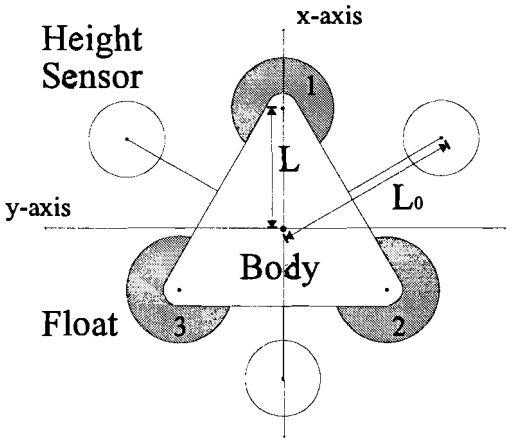

Figure 2.2: *Top view on the platform.*

As discussed in the introduction, the platform can be described in two ways:

- a MIMO model describing the absolute heights H_i of the platform above the floats.

- three SISO models describing the platform's average height H_a, its rotation around the negative y-axis θ_y and its rotation around the positive x-axis θ_x.

Both models are described in Cartesian coordinates. Which one we are going to use depends on the identification process. Therefore both derivations will be made:

$$H_a(t) = \frac{1}{3}(H_1(t) + H_2(t) + H_3(t)) \tag{2.1a}$$

$$\sin(\theta_y(t)) \approx \theta_y(t) = \frac{2}{3L}\left(H_1(t) - \frac{1}{2}H_2(t) - \frac{1}{2}H_3(t)\right) \tag{2.1b}$$

$$\sin(\theta_x(t)) \approx \theta_x(t) = \frac{\sqrt{3}}{3L}(H_2(t) - H_3(t)) \tag{2.1c}$$

and

$$H_1(t) = H_a(t) + L\sin\theta_y(t) \quad \textbf{(2.2a)}$$

$$H_2(t) = H_a(t) - \frac{L}{2}\sin\theta_y(t) + \frac{L}{2}\sqrt{(3)}\sin\theta_x(t) \quad \textbf{(2.2b)}$$

$$H_3(t) = H_a(t) - \frac{L}{2}\sin\theta_y(t) - \frac{L}{2}\sqrt{(3)}\sin\theta_x(t) \quad \textbf{(2.2c)}$$

The transformation in equations 2.1a-c will be called Ψ further on in this thesis, and the transformation in equations 2.2a-c will be called Ψ^{-1}. The dynamics of the platform are related to the dynamics of the floats. It depends on the buoyancy forces of the floats. These forces are disturbed by waves.

Normally, the position of the platform is defined by the float heights H_i. This is not the best form to present the position of the platform to the user; the position according to the average height and two perpendicular, horizontal axes is better. It is much easier to visualise and better to understand during communication with a user. The effect of the first set of transformation on the physical model of the sytem is that it transforms it into three separate SISO models. This is shown in [HEU95] and [MOR94]. Understanding why this happens is easy: changing the average height will not lead to a rotation. So, changing H_a will not affect θ_x and θ_y. This is also true for θ_x and θ_y. In practice this will not entirely be true. There will be a wave coupling, so when varying one input, the other two outputs will be excited a little by wave disturbances.

2.2 *Static and dynamic forces working on the platform*

The two main mechanisms, working on the platform are floating and tilting. The floating mechanism works on the floats of the platform. This is an interaction between gravity and buoyancy forces. The body of the platform and the crane on top of the platform cause the tilting behaviour.

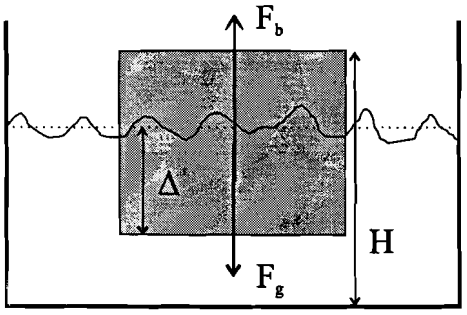

Figure 2.3: *Forces acting on one float in equilibrium.*

When the platform is moved out of position, the moment of inertia of the crane will make the platform swing back and forth like a pendulum.

In static situation the sum of the forces working on the floats, which are the buoyancy force and

the gravity force, is equal to zero. The gravity force F_g acting on one float is equal to a third part of the gravity force of the total platform, due to the symmetric building. Here we do not consider the influence of the jib. The buoyancy force F_b is proportional to the volume part of the float that is beneath the water line (Archimedes). In figure 2.3 this is depicted. The top of the float is at a distance H from the bottom of the tub. The average height of the water around the float is Δ. We can calculate Δ as follows:

$$F_g = F_b \quad \Leftrightarrow \quad mg = \rho g A_{float} \Delta \quad \Rightarrow \quad \Delta = \frac{m}{\rho A_{float}}$$

with:

- m = a third part of the mass of the total platform + mass float
- g = gravitation constant (= 9.81 [m/s^2])
- $A_{float}\Delta$ = volume part of the float beneath the water line
- ρ = mass density of water

In dynamic situation, as the platform moves straight in a vertical direction, two other forces have to be added to the equation above [HEU94]: a force F_i which represents the inertia of the water being moved and a force F_d which stands for the damping force caused by the friction between float and water:

$$F_g - F_b - F_d - F_i = ma \quad \Leftrightarrow \quad mg - \rho g \Delta A_{float} - D_{float} \Delta' - \alpha \Delta'' = m \Delta''$$

with D_f a damping constant, $\alpha\Delta'$ the inertia being moved. So the damping force $F_d = D_f\Delta'$, is proportional to the velocity of the float movement and is always pointed opposite to the direction of movement of the float. The inertial force $F_i = \alpha\Delta'$ which accounts for the water, being pushed by the floats. In the case of the platform, α is larger than the mass of the floats. This "pushing" of the floats, together with the friction between the water and the floats together determine the interaction between the platform and the water.

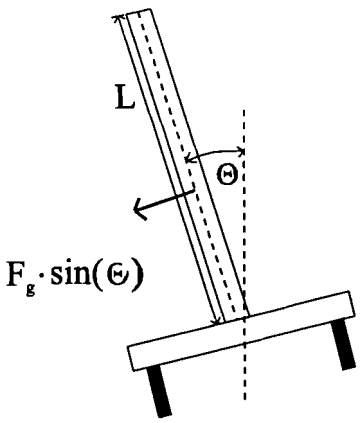

Figure 2.4: *Tilting of the platform.*

The tilting of the platform, with means there is a rotation angle, not equal to zero, around one or both axes, causes in static situation a torque working on the platform with, especially, the crane. This torque is proportional with the sinus of the rotation angle. In equilibrium the sum of the torques working on the platform and crane will be zero. When the platform is tilted, gravity and buoyancy acting on a float will not be equal.
In dynamic situation the sum of torques:

$$\sum \tau = J\theta''$$

will not be zero. It will be dependent on the netto inertia (J) and the angular acceleration (θ'').

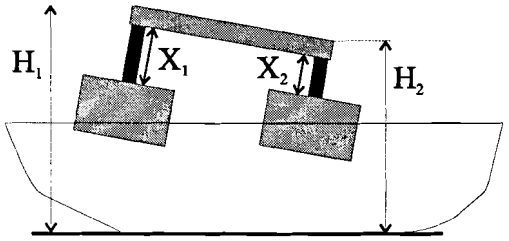

Figure 2.5: *Changed inertia.*

2.3 A transformation of the measured heights to the outputs

The height of the platform in relation to the earth is measured at three points WS_{4i} (i=1,2,3), which are the distances between the platform body and the mean water line. The distance from the height sensor to the platform centre is called L_o (= 0.68 [m]). Since for further research we are only interested in the float heights H_i (i=1,2,3), a linear transformation is needed to calculate the float heights (see also figure 2.2):

$$\begin{pmatrix} H_1(t) \\ H_2(t) \\ H_3(t) \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 1-2\frac{L}{L_0} & 1+\frac{L}{L_0} & 1+\frac{L}{L_0} \\ 1+\frac{L}{L_0} & 1-2\frac{L}{L_0} & 1+\frac{L}{L_0} \\ 1+\frac{L}{L_0} & 1+\frac{L}{L_0} & 1-2\frac{L}{L_0} \end{pmatrix} \begin{pmatrix} WS_{41}(t) \\ WS_{42}(t) \\ WS_{43}(t) \end{pmatrix}$$

With WS_{4i} the sensor signals. This transformation is derived by [HEU94].
The heights H_i are measured perpendicular to the platform (fig. 2.6). It is obvious that this kind of measurement doesn't present the real distance between platform and water surface, because of the tilting of the platform.
Therefore a transformation is derived by [MOR95] to calculate the real platform heights, and is as follows:

$$H_{ri} = \frac{3L}{\sqrt{4(H_1^2+H_2^2+H_3^2)-4(H_1H_2+H_1H_2+H_2H_3)+9L^2}} H_i$$

for i=1,2,3. In this equation, H_{ri} are the corrected heights.

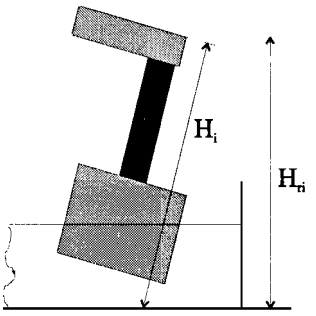

Figure 2.6: *Real and measured platform heights.*

2.4 Inputs and outputs

To control the platform, the pillar heights H_i (i=1,2,3) can be manipulated, by servo systems. These servo systems are driven by a control signal U_i which are the control inputs of the proces. The servo systems will be denoted by H_{si}
The outputs can be divided in three groups:

- The height measurements (H_i).
- The wave height measurements (Δ_i)
- The distance between the platform and the floats (X_i).

By use of the transformation Ψ and Ψ^{-1} described in equations 2.1 and 2.2 we can also use the independent inputs and outputs based on average height and two perpendicular axes. In figure 2.7 this all is schematically drawn.

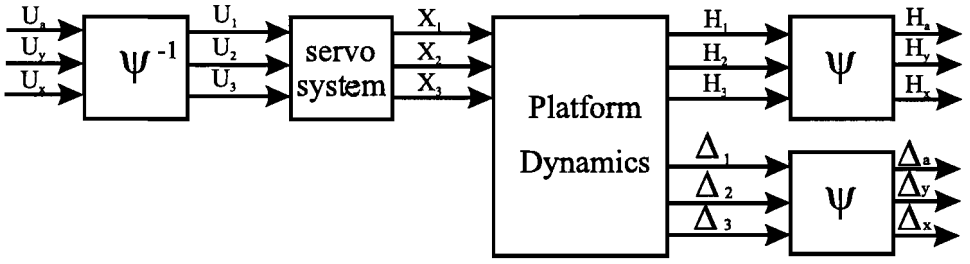

Figure 2.7: *Overview of inputs and outputs.*

Using the transformation Ψ means that we will use the three independent SISO transferfunctions, containing the average height, rotation around the x-axis and rotation around the y-axis. According to [HEU94] these SISO systems will all have a physical order of two, because they can be considered as mass spring systems. Using the transfer functions X_i to H_i (i=1,2,3) we will get

an order of six for i=2 and 3, due to the fact that the the SISO system can be considered as a diagonalization of the MIMO system. But for output H_1 an order of 4 is derived, because by exciting X_1 we do not excite the rotation H_y , due to the symmetry (see also figure 2.1).

2.5 The crane.

The crane, which is mounted on the platform, consists of a horizontal tube, placed on a rotable vertical tube. On top of the jib, a small car can drive from the hook of the vertical tube towards the end of the jib. The vertical tube can rotate by a harmonic drive. The servo amplifiers for this harmonic drive are placed outside the vertical tube to improve the balance of the platform: they are acting as a contra weight. This is depicred in figure 2.8.

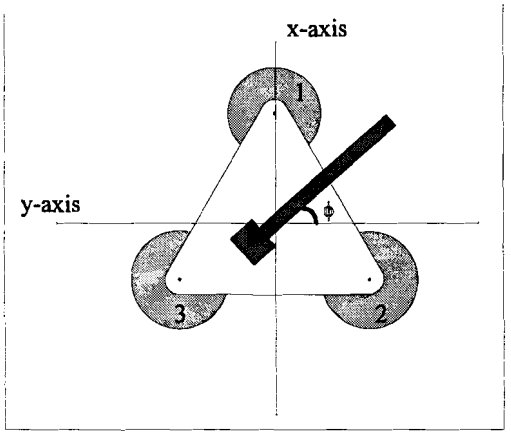

Figure 2.8: *Platform with crane.*

The forces acting on the platform due to the crane, can be transformed in three independent forces, in the same way as the outputs of the MIMO model, by use of the transformation Ψ:

$$F_{ca} = F_{c1}(t) + F_{c2}(t) + F_{c3}(t) = -gM_l$$

$$F_{cx}(t) = F_{c2}(t) - F_{c3}(t) = \frac{-2}{\sqrt{3}} \frac{L_l(t)}{L} g M_l \cos\phi(t)$$

$$F_{cy}(t) = F_{c1}(t) - \frac{1}{2}F_{c2}(t) - \frac{1}{2}F_{c3}(t) = -\frac{L_l(t)}{L} g M_l \sin\phi(t)$$

with ϕ the angular velocity of the crane. This velocity is assumed to be very low. Therefore the centrifugal force is neglected. It is clear that the inertia's J_x and J_y which will occur during tilting of the platform, will be influenced by the position of the trolley and the weight of the mass. If the process is not controlled, the platform will lean over in the direction of the load mass. Without a load mass the platform is almost balanced, but not entirely; it leans a bit over in the direction of the jib. So the centre of gravity is not entirely in the middle of the platform. If the jib is positioned above a float, this edge of the platform will be 1.5 cm lower then the other two edges. This means an angle of 0.9 degrees.

2.6 The servo systems

The distance between the floats and the platform can be varied by a spindle and a motor. The rotational speed of the motor is controlled by a servo amplifier with an input range from -10 Volt until 10 Volt. The rotational speed of the motor is transformed into a translation by the spindle. The resulting pillar height X_i is measured by an LVDT. This measurement tool can be considered as almost linear. In figure 2.9 the output voltage of the LVDT at pillar 1 is plotted as function of the distance between the float and the platform.

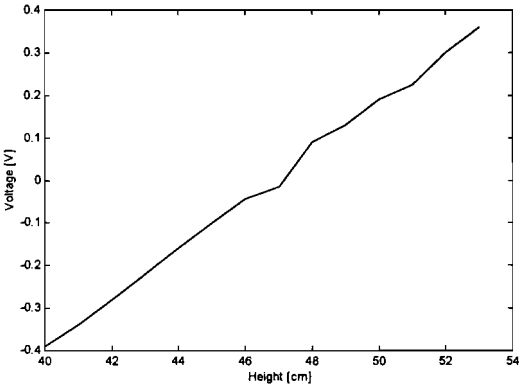

Figure 2.9: *The output (voltage) as function of the input (height) of the LVDT at float one.*

The total construction is depicted in figure 2.10:

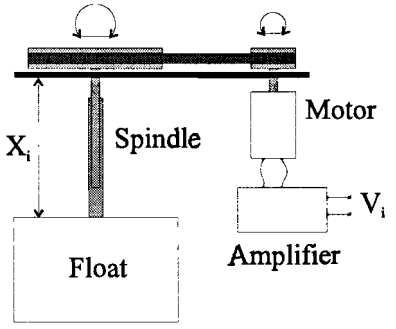

Figure 2.10: *The servo system in combination with the floats.*

The distance X_i can be varied between 16 and 38 cm. A servo system can be appointed by a pure integrator, so $H(s)=K_s/s$. If position feedback is used, the total transfer function becomes:

$$H_{total} = \frac{K_{ff}K_s}{K_{ff}K_{fb}K_s + s}$$

with K_{ff} as feedforward gain, K_{fb} as feedback gain and K_s the servo gain.

K_s was measured and was 0.27 [m/V]. K_{ff} and K_{fb} have to be defined externally, for example by realtime Simulink options.

2.7 The sensor system

The distances of the platform with respect to the water level were originally measured by three small strips, length 40 cm, which were placed opposite of the floats, as was depicted in figure 2.2. The principle of this measurement depends on the conductivity between the two copper layers on each strip, which increases (almost) proportional with de depth of the strip in the water. If there is a dc voltage between the two layers of a strip, then the layers will be polarized, and electrolyse will take place on that part of the strip that is beneath the water line. The result is that there will arise an oxide layer on top of the copper layer, which influences the sensitivity of the measurement. Therefore an ac voltage was used, this will avoid polarization.
These sensors showed to be time varying during the experiments worked out for this thesis. This was caused by the fact that after a short time the conductivity of the sensors changed by the dissolving of metal particles in the water and by oxidation of the copper. This gives them a varying sensitivity as function of the position on the copper surface. For this reason the sensors had to be calibrated several times every day. Further, the sensors were strongly nonlinear. This had to be compensated by a look up table. During the measurements the sensors were compared with a new type of sensor, based on pressure: the Motorola MPX2010 on chip temperature compensated, silicon, pressure sensor. These were prepaired on top of a metal pipe of approximately 60 cm. and positioned at the platform. Comparing the measurements, big differences occured between these two types of sensors. It was caused by the fact that the conductivity sensors were not stiff enough; white noise exitations, used for data aquisition, were causing a sweeping effect. This resulted in the occurence of higher harmonics in the sensor signal, causing a highly disturbed measurement signal. It's obvious that such kind of sensors couldn't be used to get a reliable measurement.
Therefore the new type of sensors were installed. These consist of a small pressure sensor, mounted on top of a pipe with an inside diameter of 2 mm. and a lenght of 60 cm. (figure 2.13) These sensors were tested by [RIS97] and were showing a very accurate behaviour. In figure 2.11 the hysterese diagram of a sensor is depicted. As follows from this picture we can conclude that the behaviour is very linear in static usage.

For dynamic situation we have to look at Bernoulli's equation:

$$\rho g h + \frac{1}{2}\rho v^2 + P = \text{constant}$$

With ρ the mass-density of water, g the gravitation accelaration, h the height, v the velocity of the water and P the pressure at a certain position of the sensor. The second term at the left says that dependent from the vertical velocity a measurement error will occur due to this nonlinear term. However, experiments with different frequencies all showed a measurement error wich was negligible. Another point, to take in account, is the dynamic behaviour of the sensor. This can be considered as a second order system with complex poles, because it acts like a mass spring system. The mass is the water column witnin the pipe, the spring is the damping of the air column in the pipe. The time constants measured were approximately 30 ms.

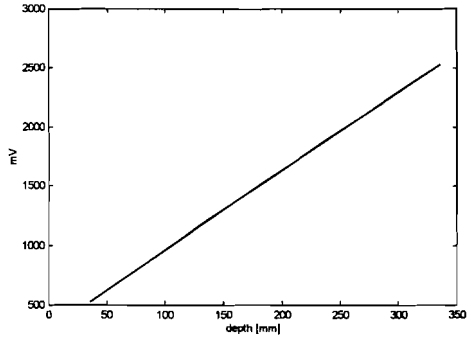

Figure 2.11: *Hysterese diagram of pressure sensor.*

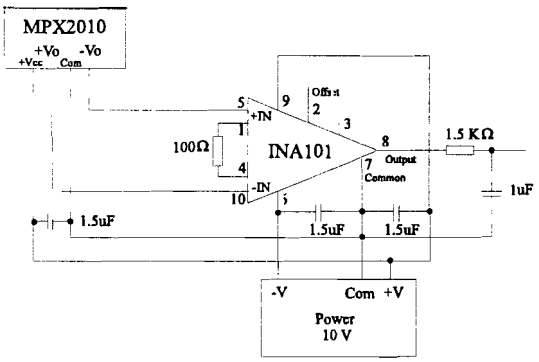

Figure 2.12: *Electronic scheme used to transform the sensor outputs to usefull signals.*

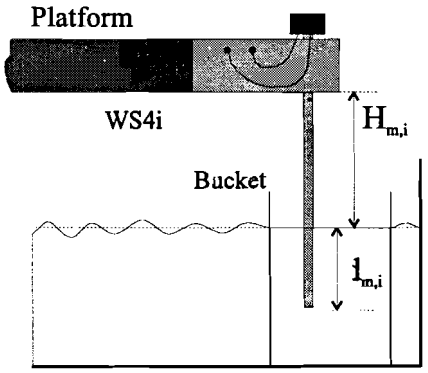

Figure 2.13: *Heights sensor assembled at the platform.*

To predict the movements of the platform due to the waves, the wave heights around each float were measured. This prediction should increase the performance of the platform control. Imagine a situation wherein the float/platform distance at one edge is too small. The platform will lean over in the direction of this float. The controller will start an action to increase this distance. But at the same time a wave will reach the float and will lift it. The controller should take this upward movement into account, and decrease its output value for this float.
The wave height sensors are made of a sheet of epoxy material with a thin layer of copper on top of it. A part of the copper is etched away, this way a specific pattern is formed. This is depicted in the picture beneath. The pattern is made out of two parts. Between these parts the resistance is measured.

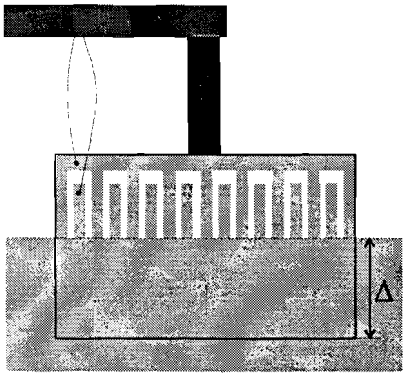

Figure 2.14: *Wave height sensor.*

This resistance increases/decreases almost proportional as the average wave height around the float decreases/increases. But due to the varying sensitivity on the sensor surface, caused by oxidation, the behaviour of the sensor is nonlinear and very time varying. To use the outputs of this sensor system, the output values had to be transformed to linear values. This has been done with look up tables, making use of linear interpolation. But after one hour after callibration the results were not reliable anymore and they had to be callibrated again. This was very time consuming (2 a 3 hours). Here too the oxidation caused problems.

2.8 The used software and hardware

The software for training the neural networks as will be explained in chapters 3 and 4, was written by Jozef Mazak ([MAZ96]) in the programming language C. It can be divided into three parts. At first the program *ID_08F.C*. This was developed for estimating a nonlinear model. For developing an observer the program *OBS_08E.C* was used. At the final stage, the development of a controller was done with the program *CNT_08F.C*. This software could only be used at a DEC ALPHA, because the software needs some mathematical routines from the NAG Fortran Library, which was only reachable from this computer.

For data acquisition the digital signal processor (DSP) from dSPACE was used, in combination with a Pentium computer, because of its realtime SIMULINK possibilities. Further, of course, MATLAB was used for all data preparations.

2.9 References

[HEU94]: Heuvel C.J.J. van den
Modeling of a floating platform.
TUE, graduation report, 1994.

[JAN90]: Janssen J.F.R.
Identificatie en regeling van een drijvend platform.
TUE, graduation report, 1990.

[REI96]: Reinierkens M.M.H.J.
Identification and control of a floating platform.
TUE, graduation report, 1996.

[RIS97]: Risseeuw P.
Waterniveaumeting met behulp van druksensoren.
TUE, practical training report, 1997.

[MAZ96]: Mazak J.
Transition control based on grey, neural states.
TUE, pHd thesis, 1996.

[MOR95]: Mortel, H. van de
Modeling of a floating platform.
TUE, graduation report,1995.

Chapter 3

On Neural Networks

3.1 Basics of Neural Networks

The biological neuron is the basis upon which neural networks are based. Each neuron is composed of a body (or soma), an axon and a large number of dendrites. The dendrites form a very fine brush around the body of the neuron. The axon can be seen as a very fine, thin tube which splits into branches terminating in little endbulbs almost touching the dendrites of other cells. The small gap between such an endbulp and a dendrite of another cell is called a synapse. The axon of a single neuron forms synaptic connections with many other neurons. Figure 3.1 shows the components of a biological neuron. Impulses propagate down the axon of the neuron and impinge upon the synapses,

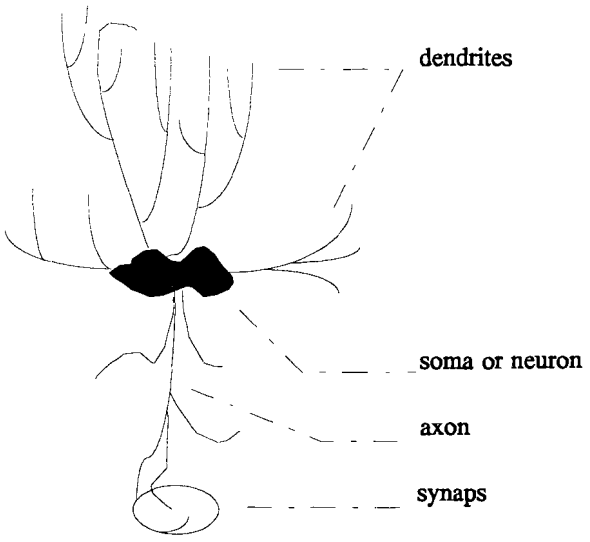

Figure 3.1: *Elementary components of a biological neuron*

sending signals of various strengths down the dendrites of other neurons. The strenght of the neuron is determined by the efficiency of the synaptic transmission. A neuron will send an impulse down its axon if sufficient signals from other neurons impinge upon its dendrites in a short periode of time. A biological neuron sends an impulse down its axon, if the exitation level

exceeds a critical level, the treshhold of the neuron. In artificial intelligence, simple models of such a neuron are used. The body of an artificial neuron is often presented by a weighted sum of the input signals, followed by an activation function, which determines the output of the neuron. Such a neuron is called a McCulloch-Pitts neuron (fig 3.2).

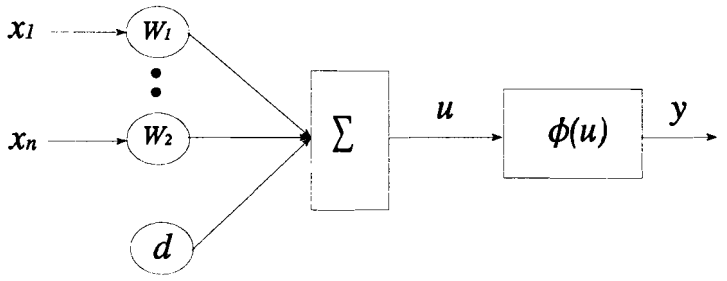

Figure 3.2: *The McCullogh-Pitts neuron.*

with:

x = input vector
θ = vector of weights (w_1, w_2,,.... , w_n)^T from layer l-1 to a node in layer l
d = offset
u = summation of weighted inputs
ϕ= activation function

The output y_i^l of such a neuron, with i the output number and l the layer number, is then calculated by:

$$u_i^l = \sum_{j=1}^{N_{N,l-1}} \theta_{w_{ij}}^l x_j^{l-1} + d_i^l \quad , \qquad y_i^l = \phi\left(u_i^l\right)$$

With $N_{N,l-1}$ the number of nodes in the previous layer and l the layer index.
The most commonly used activation functions are plotted in the picture beneath:

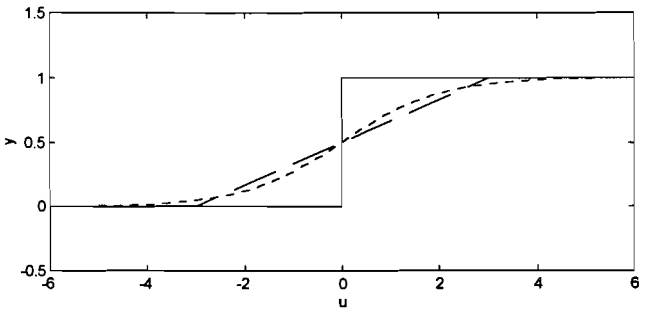

Figure 3.3: *Three of the most commonly used activation functions: Step-threshold, piecewise linear (dashed) and sigmoid (dotted).*

The sigmoïd function,

$$\phi(u) = \frac{1}{1+e^{-u}}$$

is mostly used, because it exhibits an asymptotic behaviour as well as a smooth nonlinear behaviour, which is important for the back-propagation algorithm, as will be explained later. In reality it is known that real neurons, located in different parts of the brain, behave in different ways, ranging from Gaussian-like for visual needs to sigmoidal for ocular motor needs. It is usual the case for artificial neural networks, however, that only one type of nonlinearity is used in the network, linking it with the fact that the network is suitable for only one specific task.

3.2 Benefits of a neural network

Some specific benefits of neural networks will be discussed here.

Structure: The power of a neural network is in the first place derived from its parallel structure. This makes short computation times possible, all nodes in a layer in such a network can be executed in one time, only if more processors are available.

Nonlinearity: A neural network is a nonlinear system, which can learn nonlinear input/output mappings.

Knowledge: The knowledge captured in a neural network is stored in its network weights. During the learning phase these weights are adjusted to get a correct input/output mapping. This is the main difference with other artificial techniques; they use rules instead of weights. Therefore these rule based systems are called symbolic AI systems and neural networks are called subsymbolic systems.

Fault tolerance: A neural network acts fault tolerant. If a connection link in the network is damaged or falsified, generally the network will not provide for useless information, but will only show a degradation of its performance

3.3 The Multi Layer Perceptron

When several neurons are connected together in a particular way, like in figure 3.4, we have composed a multi-layer perceptron neural network. The network consists of a number of layers; an input layer to collect the inputs, an output layer and some hidden layers between them. A multi-layer perceptron requires a pattern (a set of data) to be presented as inputs to the input layer. The outputs from this layer are then fed as weighted inputs, to the first hidden layer, and subsequently the outputs from the first layer are fed, as weighted inputs to the second hidden layer. This process continues until the output layer is reached. So, each node in a layer merges the weighted inputs of the previous layer, so the network will map an input space (of dimension n) to an output space Y (of dimension m) in a nonlinear way. This kind of network is called a

feedforward network, because the information flows from one side to the other. According to Funahashi (1988) [NAR90], every continuous function can be approximated with such a multi-layer perceptron, containing one hidden layer, and sigmoid activation functions. The output layer may even be a linear function. The synaptic strength of one neuron i towards another neuron j is determined by an interconnection weight w_{ij}. The combination of weights and the activation function determines the operation of the network. [KRIJ93] specifies a neural network by three features:

- **Network topology**: this is the way the nodes in a network are interconnected.
- **Control algorithm**: executes the network and provides for the I/O.
- **Learning rule**: provides for the modification of the network weights.

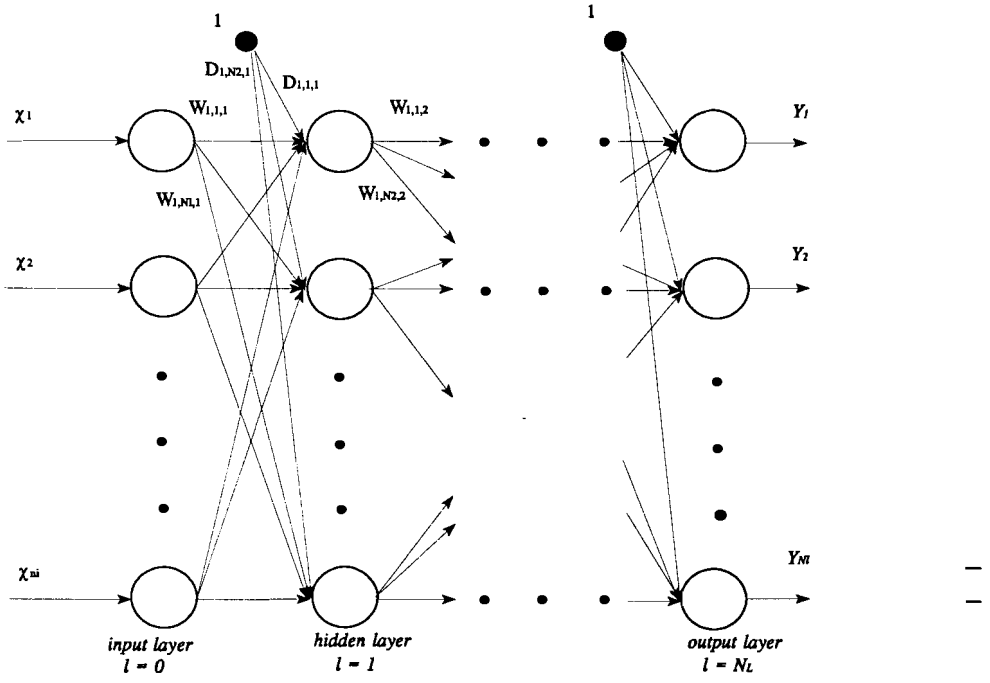

Figure 3.4: *Multi-layer perceptron.*

Network Topology

A neural network consists of a number of interconnected nodes, which mostly are organized in layers; an input layer, an output layer and a number of hidden layers, as depicted in fig 3.4.
Data is presented to the network via the input layer, and the network presents its results to the environment via the output layer. The hidden layers are used for storing information. In the depicted network the information flows from left to right; no recurrency is in the network. The nodes can also be connected in a particular way that

a feedback structure arises (e.g. Hopfield network).

Control Algorithm

The reaction of the network to an input is directed by a control algorithm and is dependent upon the network topology. In a feedforward network, where the information flows from input layer to output layer, the layers are executed sequentially. In case of a feedback network, the information reverberates around the network, under control of the algorithm. The communication between the network and the environment is also handled by the control algorithm.

Learning rules

To achieve an adequate mapping, the weights in the network have to be varied until the input/output relation shows a satisfying behaviour. Complex learning laws are used. Learning laws determine how the network adjusts its weights by using some error function The kind of learning is determined by the characteristics of the network desired. During the learning fase, the user provides the network with a number of input patterns and the network adjusts its weights until the output is equal or close to the desired output.

For multi-layer perceptrons, questions about the structure will arise. Some choices have to be made in relation to the number of hidden layers and the number of neurons in each layer. A lack of neurons will show bad performance, too many neurons will lead to an overparametrised network, which means it will take much computational time and energy to find a global minimum. Because no proper algorithm or formula exists, they have to be appointed in a heuristic way.
After this, some values have to be adjudgest to the network weights, so that the network will capture the desired input/output relation, between certain error bounds. Several supervised iterative and recursive training algorithms are available, in which a cost function is minimalized, according to a gradient calculation of the network weight vector. Generally, the cost function is chosen as:

$$J(\Theta) = \frac{1}{2N}\sum_{k=1}^{N} \varepsilon(k)^T \varepsilon(k) = \frac{1}{2N}\sum_{k=1}^{N} (\underline{d}(k) - \underline{\hat{y}}^p(k))^T (\underline{d}(k) - \underline{\hat{y}}^p)$$

where $\underline{y}$ is the network output and d is the desired output. N is the number of samples on which this criterion is minimized and p is the number of patterns.
Clearly, if we have N_N weights in a network, we are dealing with an N_N-dimensional optimization problem. This is a nonlinear problem, so finding the minimum will be quite difficult, because of the nonconvex landscape. The problem is to avoid the local minima to reach the global minimum. Some stochastic optimization methods have the possibility to climb out of a local minimum. To find the global minimum, the optimization function has to pass the local minima, without getting stuck in it. This problem is relatively easy to solve. But in practice the problem becomes more complex. For example, a network with 6 inputs, one hidden layer with 8 nodes and an output layer with 5 nodes is a 101 dimensional problem. This is very complex, and can't be solved explicitely. Some heuristic techniques have been developed, but they are no guaranty to find the real global minimum. Most of the learning algorithms are based on gradient computations; the gradient shows the sensitivity of the cost function as function of the network weights.

The direction where the gradient is most sensitive is chosen to decrease the costfunction;

$$\Theta(j+1) = \Theta(j) - \eta \nabla J(\Theta)$$

where ∇ is the gradient of the cost function with respect to the weights and η is the is the stepsize for every iteration.
Some methods to optimize the multi-layered perceptron network weights are shown beneath.

Back propagation

We will consider a network with one input layer, one output layer and certain number of hidden layers. First the output error of a input pattern x is propagated back through the network towards the inputs. Then the network weights are adjusted according to gradient calculations. The pattern will be propagated forward through the network and the new, upgraded error can be computed. We define an input vector:

$$\underline{x} = (x_1,...,x_N)^T$$

a hidden vector:

$$\underline{h} = (h_1,....,h_L)^T \quad , \quad h_l = \phi\left(\sum_{n=1}^{N} w_{ln} x_n\right)$$

and an output vector:

$$\underline{y} = (y_1,....,y_M)^T \quad , \quad y_m = \phi\left(\sum_{l=1}^{L} w_{ml} h_l\right)$$

The system error:

$$J(\Theta) = \frac{1}{2N}\sum_{P}\sum_{k}(\underline{d}(k) - \underline{\hat{y}}^p(k))^T(\underline{d}(k) - \underline{\hat{y}}^p) = \sum_{p} E^p \, , \qquad E^p = \frac{1}{2}\sum_{k}(y_k^p - d_k^p)^2$$

with:

p = number of patterns
m = number of outputs
l = number of inputs
d = desired output

can be minimised by adjusting the network weights in the opposite direction of the gradient:

$$w_{ml}(n+1) := w_{ml}(n) - \eta \frac{\partial E^p}{\partial w_{ml}}$$

with

$$\frac{\partial E^p}{\partial w_{ml}} = \frac{\partial E^p}{\partial u_m^p} \frac{\partial u_m^p}{\partial w_{ml}} = \frac{\partial E^p}{\partial y_m^p} \frac{\partial y_m^p}{\partial u_m^p} \frac{\partial u_m^p}{\partial w_{ml}}$$

and

$$\frac{\partial E^p}{\partial w_{ml}} = \frac{1}{2} \sum_m (y_k^p - d_m^p) \cdot \frac{\partial y_m^p}{\partial w_{ml}} = \frac{1}{2} \sum_m (y_m^p - d_m^p) \cdot \phi'(u_m^p) \cdot h_l^p$$

and η as the stepsize. This leads to

$$w_{ml}(n+1) := w_{ml}(n) - \eta \frac{1}{2} \sum_m (y_m^p - u_m^p) \cdot \phi'(u_m^p) \cdot h_l^p$$

The input variable *h* can be the output of an input node in case of a network with no hidden layer, or the output of a hidden node in case of the presence of a hidden layer. Clearly error back propagation only works if ϕ is a differentiable function. If the activation function is a sigmoid, then:

$$\phi(u) = \frac{1}{1+e^{-u}} \Rightarrow \phi'(u) = \frac{e^{-u}}{(1+e^{-u})^2} = \phi(u)(1-\phi(u))$$

The local gradient for an output node is computed by:

$$\delta_m^p = \frac{-\partial E^p}{\partial u_m^p} = \frac{-\partial E^p}{\partial y_m^p} \cdot \frac{\partial y_m^p}{\partial u_m^p} = (d_m^p - y_m^p) \cdot \phi'(u_m^p)$$

So, with

$$u_m^p = \sum_{l=1}^{L} w_{ml} h_l^p \Rightarrow \frac{\partial u_m^p}{\partial w_{ml}} = h_l^p \Rightarrow -\frac{\partial E^p}{\partial w_{ml}} = \delta_m^p \cdot h_l^p$$

this becomes:

$$w_{ml}(t+1) := w_{ml}(t) + \eta \delta_m^p \cdot h_l^p$$

Because of the choise of a sigmoid activation function this can be written as:

$$w_{ml}(t+1) := w_{ml}(t) + \eta (d_m^p - y_m^p) \cdot y_m^p (1 - y_m^p) \cdot h_l^p$$

We can apply this also to the *l*-th hidden layer. The weights in this layer can be updated by:

$$w_{ln}(t+1) := w_{ln}(t) - \eta \frac{\partial E^p}{\partial w_{ln}}$$

The error sensitivity of the output l of the hidden layer is computed by:

$$\frac{\partial E^p}{\partial w_{ln}} = \frac{\partial E^p}{\partial u_l^p} \frac{\partial u_l^p}{\partial w_{ln}} \qquad \textbf{(3.19)}$$

For the second term in equation 3.19, we can say:

$$u_l^p = \sum_{n=1}^{N} w_{ln} x_n^p \quad \Rightarrow \quad \frac{\partial u_p^l}{\partial w_{ln}} = h_l^p$$

And for the first:

$$\frac{-\partial E^p}{\partial u_l^p} = \frac{-\partial E^p}{\partial h_l^p} \cdot \frac{\partial h_l^p}{\partial u_l^p}$$

The latter can be further simplified by:

$$\frac{\partial h_l^p}{\partial u_l^p} = \phi(u_l^p)$$

and

$$\frac{-\partial E^p}{\partial h_l^p} = \sum_{m=1}^{M} \frac{-\partial E^p}{\partial u_m^p} \cdot \frac{\partial u_m^p}{\partial h_l^p}$$

If we combine equation 1 with:

$$u_m^p = \sum_{l=1}^{L} w_{ml} h_l^p \quad \Rightarrow \quad \frac{\partial u_m^p}{\partial h_l^p} = w_{ml}$$

Then this makes:

$$-\frac{\partial E^p}{\partial h_l^p} = \sum_{m=1}^{M} \delta_m^p w_{ml}$$

This leads to a gradient for a node in the l-th layer:

$$\delta_l^p = -\frac{\partial E^p}{\partial u_l^p} = \sum_{m=1}^{M} \delta_m^p w_{ml} \cdot \phi'(u_l^p)$$

So equation 3.19 can be described into

$$-\frac{\partial E^p}{\partial w_{ln}} = \sum_{m=1}^{M} \delta_m^p w_{ml} \cdot \phi'(u_l^p) \cdot h_n^p$$

and the weights between the layers L and M are updated by

$$w_{ln}(t+1) := w_{ln}(t) + \eta\, \delta_l^p \cdot h_n^p = w_{ln}(t) + \eta \sum_{m=1}^{M} \delta_m^p w_{ml} \cdot h_l^p (1 - h_l^p) \cdot h_n^p$$

This method can also be applied to networks with more hidden layers by adjusting the weights in every layer similar to the method pointed out above. This method is called error back propagation. The main draw backs of using this algorithm is the poor scaling behaviour: an increase of the network size results in an exponential growth of the computation time, its slow convergence and its property to get stuck in a local minimum. Therefore this method is nowadays hardly used in software implementations. In future it will only be used in hardware implementations, because for this it's very difficult to implement more complexer algorithms. Some of these other, more complex methods will be discribed here.

Quasi Newton Search

Quasi Newton acts as follows:

$$\Theta(j+1) = \Theta(j) - \eta\, M(\Theta(j)) \nabla J(\Theta(j))$$

Where M is a positive definite approximation of the inverse Hessian H:

$$H = \left[\frac{\delta^2 J(\theta)}{\delta w \delta w^T} \right]$$

The general difference between back propagation and quasi-Newton is the rate of convergence: as backpropagation converges linearly, quasi-Newton converges quadratically. This makes quasi-Newton only usefull if the values of the parameters are close to the global minimum. Otherwise, due to the high convergence speed, it will get stuck in a local minimum.
If M is chosen to be the identity matrix, the method above becomes the back propagation method.

Simulated annealing

It should be pointed out that the deterministic methods above try to find the local minima and choose the best one. But, if no analytical function is known, it will not be clear if all the minima are found. Better results will be achieved by use of stochastic methods. Such a stochastic method is Simulated Annealing
At first the error E[k]=E(w[k]) is computed. Then Simulated Annealing generates random (Gaussian) points along each coordinate direction, $r_i[k] \in N(0,\sigma^2[k])$, where each coordinate stands

for a network weight. If the cost function decreases for the new point, the point is accepted. Otherwise the error, computed by:

$$\Delta E[k] = E(\underline{w}[k] + \underline{r}[k]) - E[k]$$

is used to determine the acceptance of a new point. This probability of acceptance can be calculated, for example by:

$$P_r = \frac{1}{1 + \exp(-\Delta E[k]/T[k])}$$

If this probability is larger then a certain value, then the point will be accepted, otherwise not. If accepted, then $w[k+1]=w[k]+r[k]$ and $E[k+1]=E(w[k]+r[k])$. So points can be accepted which cause an increase of the error function. The probability of accepting such points decreases in time because of the decrement of the temperature. This means that the higher the temperature, the bigger the chance of escaping from a local minimum. For the decrement of the temperature several methods can be used, like:

$$T(k) = \frac{T_0}{\ln(k)}$$

The program is stopped after a certain number of iterations or if $E[k] < \varepsilon$

During this thesis work Simulated Annealing and quasi-Newton were both used. At first Simulated Annealing was used to make a (rough) approximation of the system and to avoid the local minima with high cost function values. This takes only short computing time due to the stochastic way of searching. After that a procedure with quasi-Newton was started for optimizing the problem. This algorithm takes very much computing time per iteration. This is caused by the fact that the algorithm has to compute the first and second derivatives after every iteration for evaluation of the cost function.

3.4 Some important parameters during the learning phase

Training the network requires, dependent on the trainings method values for some specific training parameters. For example using simulated annealing we have to choose the initial temperature, the noise variance and so on. But there are also two parameters which occur in (almost) every trainigs method. We will discuss them below:

Generally, training the network requires a set of data which has to be split in two parts. The first part is used for learning: the network should capture the relation between the inputs and outputs in the learning set, by adjusting its weights. A rule of thumb says that the lenght of this data set should be at least ten times the number of weights in the network. Otherwise the network will not learn the desired input/output relation, but tries to copy the data set. Therefore the second part of the data set is used: with this set the performance of the network can be validated.

The learning phase happens in a number of iterations. It's important that there are enough iterations during this phase. The netto output error will decrease after every iteration, until after a certain number of iterations no more decrement will be found. It's often wise not to take too much iterations, in spite of the still decreasing cost function. The danger is that generalization will

occur: the network tries to find certain properties in the learning set (for example noise) which are typical for the learning set only. The performance while using the learning set will decrease. This is shown in figure. After N iterations this phenomenom occurs.

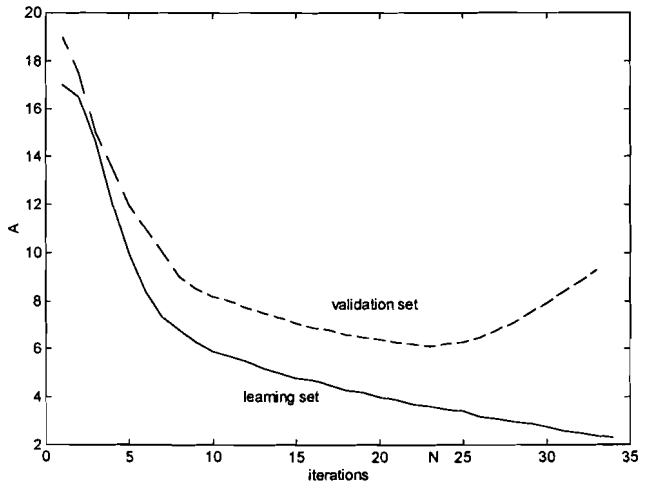

Figure 3.6: *Determination of generalization point N.*

An important parameter in these deterministic learing algorithms is the step size η. This should be chosen wisely, as depicted in figure 3.7 In case of a too small η, convergence will be very slow and optimization of the cost function will take much computing time. On the other hand, if η is chosen too big, no convergence will take place and the network optimization will become unstable, in the sense of Lyapunov.

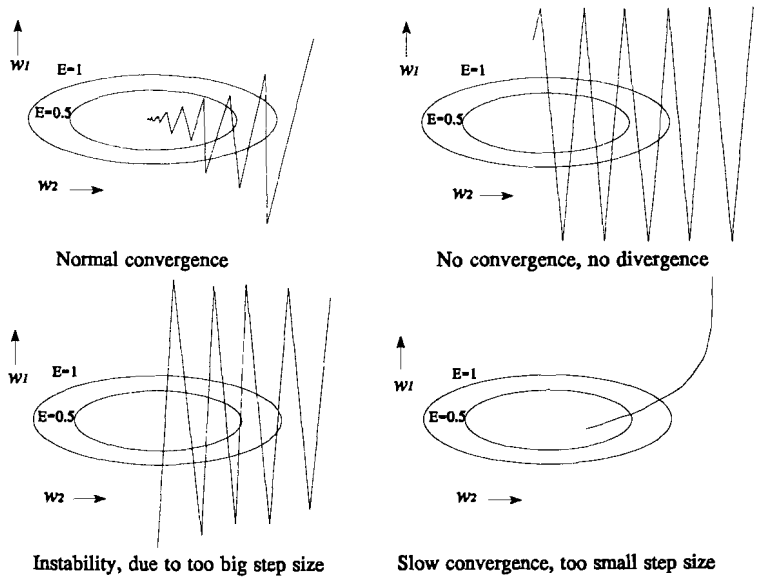

Figure 3.7: *Choices for the rate of convergence.*

No rules are available for chosing the optimal value for η. Therefore it has to be fixed by experiments or by prior knowledge.

A third number to take into account is the network size. If this size is taken too small certain characteristics will not or poorly be modelled and bad performance will occur. On the other hand it should not be taken too large otherwise the computation time becomes too big. The large network size could also cause a too large degree of freedom. Overparametrization will occur with the chance of generalization, as mentioned earlier.

3.5 Dynamic neural networks

In control applications mostly we are dealing with dynamical systems. Therefore we can use recurrent feedforward neural networks. In this structure the outputs are fed back to the inputs, usually through a number of delays (fig 3.8) to capture the dynamic properties of a system.

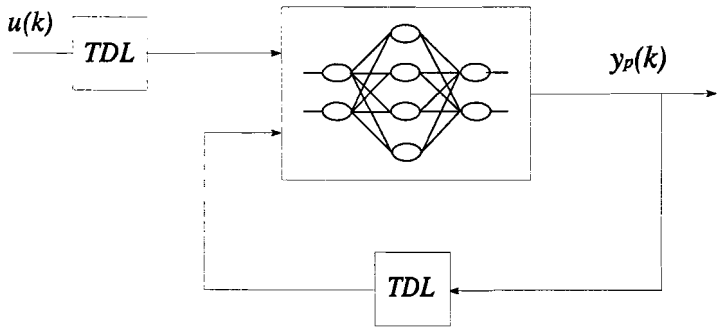

Figure 3.8: *Modelling of a dynamic process with a neural network and a dynamic feedback with tapped delay lines (TDL).*

This means that calculation of the needed gradients will become more complex, the optimization procedure will become dependent of past gradients: The aim is still to determine the derivatives $\partial y(k)/\partial\theta$. Because of the dynamic properties of such a system, $\partial y(k)/\partial\theta$ is the solution of a difference equation, i.e., $\delta y(k)/\delta\theta$ is affected by its own past values:

$$\frac{\partial y}{\partial \theta} = \frac{\partial N[v]}{\partial v} z^{-1} \frac{\partial y}{\partial \theta} + \frac{\partial N[v]}{\partial \theta}$$

with v the input vector of the neural network and $\partial N[v]/\partial v$ and $\partial N[v]/\partial\theta$ are the Jacobian matrix and vector.
If the delay block from output to input in figure 3.8 is replaced by a discrete linear transfer function $H(z)$, we are talking about hybrid modeling. Then the linear transfer function describes the nonlinear proces behaviour in a certain working point, the neural network adds the nonlinear behaviour to it. This method of identification is described in [NAR90] and [NAR91].

3.6. Neural control

A lot of research has been done in getting nonlinear (neural) control techniques. Some examples are ([KRIJ93]):

- Neural direct inverse control
- Neural model reference control
- Adaptive neural control
- Neural internal model control
- Neural model-based predictive control

Here, the approach to achieve a satisfying control of the platform is carried out with a model based strategy, in which the model serves as a simulator of the proces dynamics. Due to the nonlinearities, the modeling will be carried out with use of neural networks. The proposed method here is studied and succesfully used by [MAZ96]. The general idea behind this method is to use

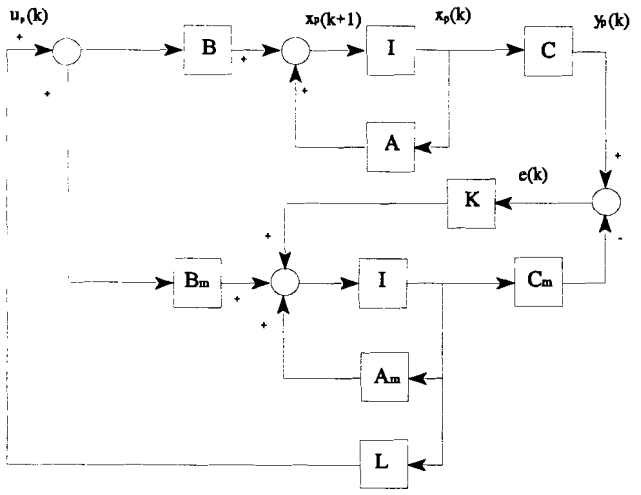

Figure 3.9: *LQG control, with K = Kalman gain and L = linear feedback matrix.*

a 'nonlinear copy' of the well known LQG-structure (fig 3.9), with which [BOU91] and [REIJ96] achieved already adequate performances at the floating platform. Therefore a nonlinear model will be estimated, and is expected to show more accuracy with respect to the linear models mentioned earlier. With this model improvement we can possibly achieve better robustness and better performance of the closed loop. The LQG-strategy uses a model of the process, to estimate the states $\underline{x}$, a so called Kalman-gain to estimate the state disturbances $\underline{\varepsilon}$ for adding them to the estimated states $\underline{x}$ and a linear feedback gain L to control the states of the real proces.

In the nonlinear case, the model, the Kalman-gain and the state feedback have been replaced by

neural networks to capture and control the nonlinear dynamics of the system (fig 3.10).
The neural state space model contains a well estimated nonlinear state space description of the process, where the state vector $\underline{x}$ is divided into two parts, a well known white part $\underline{x}_l$ according to priori knowledge of the proces, and an unknown black part $\underline{x}_0$. The outputs of the proces are described by a nonlinear matrix h. The neural filter is a nonlinear filter/predictor, which replaces the linear Kalman-gain in the LQG-structure, to estimate/predict the nonlinear state disturbances e. The state vector $\underline{x}$ is fed into the neural controller, which is a nonlinear controller, eventually with some integrating elements, to get a final tracking error equal to zero. This way of controlling a nonlinear system has shown great performance according to [Maz96], who used it for modeling and controlling a polymerization reactor.

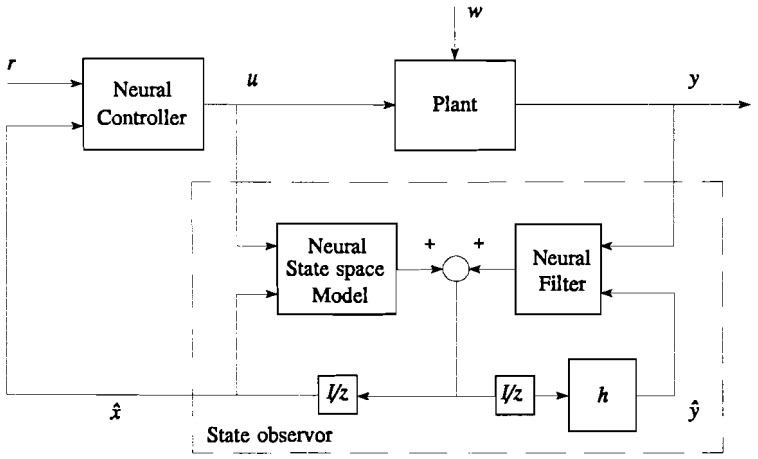

Figure 3.10: *Neural control structure for state feedback, containing a neural state-space model, a neural filter and a neural controller.*

Thus, the strategy for the controller synthesis can be divided in three stages:

1. Using measured process data we estimate a nonlinear a priori knowledge-based dynamic state-space simulation model of the proces which captures the control relevant dynamic relations between manipulated variables and controlled variables. The a priori knowledge is brought into the model by means of combination of known analytical part with a black-box neural network part. This allows us to give a physical meaning to a part of the estimated state vector.

2. At the second stage we compute a static nonlinear filter gain to build op a state observer reconstructing the proces in a disturbed environment. An important point to stress here is that the filter gain is going to be computed for a fixed model preserving its simulation capabilities.

3. At the last stage we compute a nonlinear, in principle static, state feedback controller.

The feedback gain is optimized such that the tracking errors are minimized.
There's also a possibility to add some integrating actions to the controller, to achieve final tracking errors equal to zero.

With respect to this proposed strategy, we will have to deal with a complex nonlinear dynamic system. Finding a proper weight vector θ becomes more difficult by increasing the number of states and the number of nodes (and layers). The modeling part will be a dynamic optimization problem, the filter gain and controller part are static, pattern recognition problems.

3.7 References

[KRIJ93]: Krijgsman A.J.
Dictaat van het college Kennisgestuurde Regelsystemen.
Delft University of technology, college dictate course 1995/1996, 1995.

[BIL92]: Billings S.A., Jamaluddin H.B. and Chen S.
Properties of neural networks with applications to modelling nonlinear dynamical systems.
International Journal of Control, Vol. 55, No. 1, pp 193-224, 1992.

[CHE90]: Chen S., Billings S.A. and Grant P.M.
Nonlinear system identification using neural networks.
International Journal of Control, Vol. 51, No 6, pp. 1191-1214, 1990.

[CHE92]: Chen S. and Billings S.A.
Neural networks for nonlinear dynamic system modelling and identification.
International Journal of Control, Vol. 56, No. 2, pp. 319-346.

[HUN92]: Hunt K.J., Sbarbaro D., Zbikowski R. and Gawthrop P.J.
Neural networks for control systems - A survey.
Automatica, Vol. 28, No. 6, pp. 1083-1112, Pergamon Press Ltd. U.K., 1992.

[NAR90]: Narendra K.S. and Parthasarathy K.
Identification and control of dynamical systems using neural networks.
IEEE Transactions on Neural Networks, Vol. 1, No. 1., pp. 4-17, March 1990.

[NAR91]: Narendra K.S. and Parthasarathy K.
Gradient methods for the optimization of dynamical systems containing neural networks.
IEEE Transactions on Neural Networks, Vol. 2, No. 2, pp. 252-262, March 1991.

[LEV93]: Levin A.U. and Narendra K.S.
Control of nonlinear dynamical systems using neural networks: controllability and stabilization.
IEEE Transactions on Neural Networks, Vol. 4, No. 2, pp. 192-206, March 1993.

[LEV93]: Levin A.U. and Narendra K.S.
Control of nonlinear dynamical systems using neural networks- part II : observability, identification and control.

IEEE Transactions on Neural Networks, Vol. 7, No. 1, pp. 30-42, January 1996.

[MAZ96]: Mazak J.
Transition control based on grey, neural states.
TUE, Phd. thesis, 1996.

Chapter 4

Identification of the process

4.1 Basics of identification.

In general control applications, a mathematical model of the system is derived to design a proper control scheme, which should meet all requirements given by a user. To get a mathematical representation of a nonlinear process, identification of the proces is necessary. A model of a system is expressed as an operator , which maps an input space U to an output space Y. In [NAR90] the objective of identification is given as :" characterize the class $\wp$ to which P belongs. Given a class $\wp$ and the fact that $P \in \wp$ the problem of identification is to determine a class $\hat{\wp} \subset \wp$, and an element $\hat{P} \in \hat{\wp}$ so that $\hat{P}$ approximates P (in the way it was desired)". It is important that the operater P is defined by the specified input-output pairs. The choice of the class of identification models $\hat{\wp}$, as well as the method to determine $\hat{P}$ depends on a variety of factors which are related to the accuracy desired as well as to the a priori information that is available concerning the system to be identified.

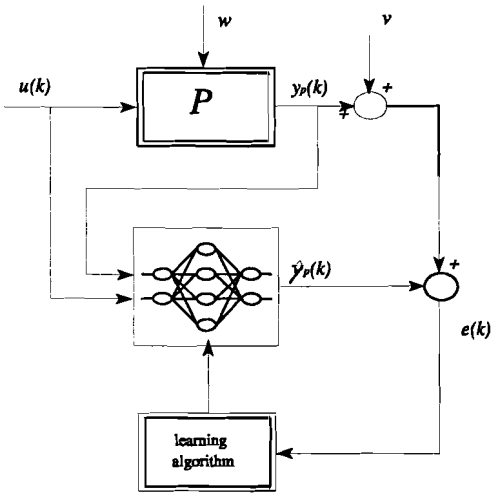

Figure 4.1: *General scheme for neural identification.*

In many cases, the learning is performed by learning input-output mappings: given an input vector $u(k)$ a corresponding output vector $\hat{y}(k)$ should be produced by a network. If this is not the case,

the output error e ($=y(k)-\hat{y}(k)$) has to be minimized iteratively by a learning algorithm. In figure 4.1 the general identification structure is depicted. The objective of the model is:

$$\|\hat{y}_p - y_p\| = \|\hat{P}(u) - P(u)\| \leq \epsilon, \qquad u \in U$$

for a desired limit value ε and a suitable norm on the outputspace. Here, $P(u) = y$ denotes the (approximated) output of the identification model. So, the estimation error will be $\hat{y}$-y $\triangleq$e. Since we are dealing with a nonlinear system, and want to create a nonlinear model of it, we have to make a nonlinear approximation of the system. According to the Theorem of Stone-Weierstrass ([LEV93]) it can be stated that feedforward neural networks with one hidden layer are able to contain every input-output mapping P. So, if the static behaviour of a system is implemented in a neural network, and the dynamic behaviour is added externally to the network with tapped delay lines, we have a powerful tool to capture the nonlinear dynamic behaviour of a system.

4.2 Proposed identification strategy, based on grey states

As mentioned before, the model to be estimated should have a state-space structure, to make the process states visible for the controller. This means the next parametrization:

$$\sum: \qquad \begin{aligned} \hat{x}(k+1) &= \hat{f}[\hat{x}(k), u(k), \theta_{\hat{f}}] \\ y(k) &= h[\hat{x}(k), u(k)] \end{aligned}$$

where h can be either a fixed output map or an estimated nonlinear (neural) model. We will just consider h as a fixed output map here. By using a priori information about the proces, like the process structure and order, we can partly elaborate the proces structure. Consequently, the total dynamical model must contain a white box part to reflect the a prioiri knowledge and a black box part to model the complementary process dynamics. This means that the white part will have a physical meaning, in contrast with the black part. The partitioning will be:

$$\underline{x} = \begin{pmatrix} x^0 \\ x^1 \end{pmatrix}$$

where x^0 are hidden state components, representing a black box part of the model and x^1 are known state components, representing the a priori knowledge about the system dynamics. In our situation for MIMO-modelling x^1 will contain the three pillar heights H_1, H_2 and H_3, and the float wave heights Δ_1, Δ_2 and Δ_3.. For SISO-modelling x^1 wil contain the independent SISO outputs H_a, θ_y and θ_x and the SISO description of the float heights Δ_a, Δ_y and Δ_x . So, we can define a grey box state space model as:

1. that the process output map h is being known exactly and fixed in the model
2. that a certain part of the true system map f can also be known and the model state map is then a combination of an analytically known part and an unknown part with no physical meaning.
3. that the state vector is devided in a white part and a black part.

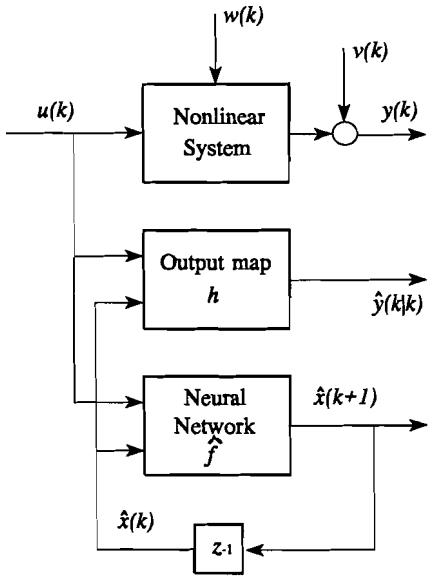

Figure 4.2: *State-space structure.*

The state space model is then defined as follows:

$$\hat{x}(k+1) = \hat{f}[\hat{x}(k), u(k), \theta_{\hat{f}}]$$

$$y(k) = h[\hat{x}^1(k), u(k)]$$

This is shown in the figure above. *W(k)* represents the state disturbances, *v(k)* the measurement noise.
For using gradient-oriented optimization routines with dynamic feedback systems like quasi-Newton, the derivatives of the system have to be calculated, as explained in section 3.5.
Let $\Gamma_x^f(k)$ be the Jacobian matrix of the neural network *f* given by:

$$\Gamma_{\hat{x}}^{\hat{f}}(k) = \frac{\partial \hat{f}[\hat{x}(k), u(k), \theta_{\hat{f}}]}{\partial \hat{x}(k)}$$

and $\Gamma_x^h(k)$ be the Jacobian matrix of the fixed output map *h*:

$$\Gamma_{\hat{x}}^{h}(k) = \frac{\partial h[\hat{x}(k), u(k)]}{\partial \hat{x}(k)}$$

Then the gradient of the error function with respect to the weights in the neural network *f* is given by:

$$\frac{\partial J(\theta_{\hat{f}})}{\partial \theta_{\hat{f}}} = -\frac{1}{N}\sum_{k=1}^{N} (y(k) - \hat{y}(k))^T \left(\Gamma_{\hat{x}}^{h}(k) \frac{\partial \hat{x}(k)}{\partial \theta_{\hat{f}}} \right)$$

where $w_{f,i}$ are components of the neural network weight vector θ_f. The partial derivative of $x(k)$ with respect to weight $w_{f,i}$ has to be computed recursively by:

$$\frac{\partial \hat{x}(k)}{\partial \theta_{\hat{f}}} = \frac{\partial \hat{f}[\hat{x}(k-1), u(k-1), \theta_{\hat{f}}]}{\partial \theta_{\hat{f}}} + \Gamma_{\hat{x}}^{\hat{f}}(k-1) \frac{\partial \hat{x}(k-1)}{\partial \theta_{\hat{f}}}$$

The first tem in the last expression is being evaluated by the backpropagation algorithm, as explained in chapter 3. The same algorithm is used to evaluate the Jacobian matrix $\Gamma_x^f(k-1)$ which is necessary for evaluation of the second term. The last formula is a recurrent relation which has to be evaluated recursively in time.

4.3 Identification preliminary

In this section will be explained how to get an adequate nonlinear model of the dynamic system, with use of neural networks. The identification procedure can be divided into three parts:
- preparation
- estimation
- validation

Preparation

In the first fase, a priori information will be collected, like (non-) linearity, bandwidth, delays, indication of the model order etc. With this information an excitation signal can be designed and a data set can be collected.

Model estimation

In the second fase, the trainings software is used to estimate a satisfying set of weights and biases for a dynamic recurrent MLP neural network, which is a state-space simulation model of the process.

Model validation

For getting an well fitting model, validation is needed to judge whether a model is usefull or not. Therefore, residuals of the estimation should be unpredictable from past inputs and past outputs; no correlation between prediction errors and combinations of (non-)linear past inputs and outputs should occur. Because we are dealing with a nonlinear system, we have to judge in two directions:

1. nonlinearities approximation
2. system dynamics approximation

Billings and Voon [BIL86] recommend the following analyses:

1. Plot of residues and visual judgement. Check for zero mean values of the residues.
2. Plots of residual spectra. If the noise is strictly additive to the output, it should show

a flat spectrum of residuals.

3. Nonlinear correlation tests.

[MAZ96] recommends that the estimation is well done if the (relative) norm of the learning phase and validation phase are almost of the same order. For this we can compute an index

$$\zeta = \frac{\sum_{k=1}^{N} \|y(k) - \hat{y}(k, \theta_{\hat{f}})\|_2^2}{\sum_{k=1}^{N} \|y(k)\|_2^2} \cdot 100\% \qquad \textbf{(4.10)}$$

relating the approximation error magnitudes to the magnitudes of the real proces output.
The last option mentioned above, is according to Billings and Voon ([BIL86]), and is based on nonlinear correlation tests. If the noise can be considered as additive at the output, the usual correlation tests ϕ_{ee} and ϕ_{ue} are adequate to make a decision about the model performance. But if this is not the case, the tests mentioned are not enough to judge.

Therefore the next five nonlinear correlation tests are recommended[1]:

$$\Phi_{\xi\xi}(\tau) = \delta(\tau)$$

$$\Phi_{u\xi}(\tau) = 0 \qquad \forall\tau$$

$$\Phi_{\xi\xi u}(\tau) = E[\xi(t)\xi(t-1-\tau)u(t-1-\tau)] = 0 \qquad \tau \geq 0$$

$$\Phi_{u^{2\prime}\xi^2}(\tau) = E[(u^2(t) - \overline{u^2}(t))\xi^2(t-\tau)] = 0 \qquad \forall\tau$$

$$\Phi_{u^{2\prime}\xi}(\tau) = E[(u^2(t) - \overline{u^2}(t))\xi(t-\tau)] = 0 \qquad \forall\tau$$

These tests are recommended for nonlinear systems whereby the noise can not be considered as additive to the output. If the system is linear, this does not create any additional problems because by superposition the noise can be translated to the output. But when the system is nonlinear internal noise can induce cross product terms or nonlinear expansions between input, output and noise. If this occurs, the induced term will be highly correlated with the input such that the model will be biased. These cross term products cannot be discovered by the regular correlation tests. In theory, the tests above cannot do this either for every situation, but in practice they have shown great performance. See the next equation:

[1] u' means : the mean value of u(t).

$$y(t) = H_u[u(t)] + H_{ue}[u(t), e(t)]$$

If the last term at the right side is detected, which represents the cross term products, the noise can not be considered as additive at the output. This cross correlation $\phi_{uu'\xi\xi}(\tau)$ detects both terms in the equation above. Sometimes $u^2(t)$ and $\xi^2(t)$ are small and the correlation $\phi_{uu\xi\xi}(\tau)$ may also be small even though $\xi(t)$ is correlated by $u(t)$. Therefore [BIL86] advises to include $\phi_{uu'\xi}(\tau)$ and $\phi_{u\xi}(\tau)$ in the validity checks. The table beneath shows their possibilities to detect nonlinear correlations in the output noise.
All the tests above are based on single dimensional correlation functions, which, for sampled input and output signals, are calculated according to the formulae

$$\hat{\Phi}_{x'y'}(k) = \frac{\frac{1}{N}\sum_{t=1}^{N-k}(x(t)-\bar{x})(y(t+k)-\bar{y})}{\sqrt{(\Phi_{x'x'}(0)\,\Phi_{y'y'}(0))}}$$

$\phi_{uu\xi\xi}(\tau)$	$\phi_{uu\xi}(\tau)$	$\phi_{u\xi}(\tau)$	Comments
$= 0$	$= 0$	$= 0$	Model unbiased. Noise additive
$\neq 0$	$= 0$	$= 0$	Internal noise term $\alpha u^k(t)e^l(t)$ where k = even or odd, l odd omitted from the model if odd-order moments of $e(t)$ are zero
$\neq 0$	$\neq 0$	$\neq 0$	Internal noise and/or wrong model order
$\neq 0$	$\neq 0$	$= 0$	Even power of $u(t)$ and/or internal noise $\alpha u^k(t)e^l(t)$ k, l = even omitted from the model if odd order moments of $u(t)$ are zero.
$\neq 0$	$= 0$	$\neq 0$	Odd power of $u(t)$ and/or internal noise $u^k(t)e^l(t)$ k = odd, l = even omitted from the model iff odd-order moments of $u(t)$ are zero.

Table 4.1: *Comments on nonlinear correlation tests*

This normalization ensures that they lie in the range:

$$-1 \leq \hat{\Phi}_{x'y'} \leq 1$$

If the computed correlations lie outside a certain confidence interval, this indicates that the correlation between the variables is significant. If N is large, the standard deviation of the

correlation estimates is $1/\sqrt{N}$, the 95% confidence limits are therefore approximately $\pm 1.96/\sqrt{N}$. These tests have been tested according to the examples given by [BIL86]. The results were satisfying.

4.4 Preparation

For designing experiments by which we can identify the process some a priori information of the process is needed. The bandwidth of the process can be determined by experiments. The highest relevant will be used to determine the sampling frequency (f_s) of the process and the clock frequency (f_c) of the uniform white noise sequences, which will excite the servo systems. The lowest relevant frequency will be used to determine the length of the data set. This length should be at least five times the largest time constant of the system. By taking several step experiments, with different amplitudes, the linearity of the system is examined. An experiment is made where one float is excited by some step inputs with different initial values (37.5, 36, 34.5, 33 and 31.5 cm) and same end values, as depicted in figure 4.3 for 450 time steps[2], with a time instance of 2 seconds. The endvalue is chosen in an area where the static equilibrium of the platform will be during the identification experiment. The plot shows that the overshoot is depending on the initial value, this is caused by the nonlinearity of the system. .

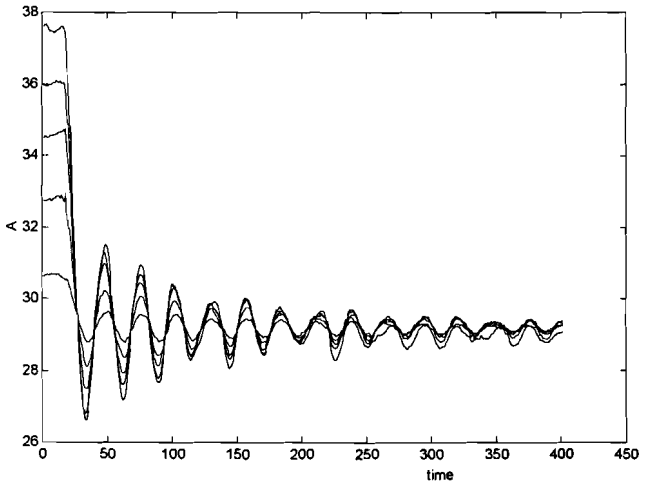

Figure 4.3: *Several step experiments for testing nonlinearity and determining the slowest time constant of the proces. Every time step is equal to 0.1 second.*

Further, at certain moments, the amplitude peak is larger than peaks at previous moments, for example after 23 seconds. This is mainly caused by waves, reflected by the edges of the tub. To get an impression of the bandwidth of the proces, two experiments have to be carried out. The largest time constant can be determined by step responses, like in figure 4.3. The output reaches

---

[2] *In this thesis every time step will be equal to 0.1 second, according to the sample frequency of 10 Hz. The amplitudes are scaled, and therefore not direct related with the real heights.*

the 67% margin within 20 seconds. But the wave reflections cause a temporary increase of the response, so for safety we better take a value of 30 seconds. From this plot the main wave time constant can be determined; it's approximately 3 seconds, which means a frequency of about 0.3 Hz. This plot was dicussed and the main question was whether the results in this picture were reproducible or not. The results depicted are dependent on the position of the platform in the tub. A slight change in the position will probably cause other wave effects due to chaotic behaviour of the waves. This means they are not stochastic but very dependent on the initial position of the platform. After the moment that the 67% margin has been reached, the wave effects are just very little related to the servo inputs U_i, which is typical for chaos.
To determine the smallest relevant time constant, the process is excited by a white noise signal. A process is said to have a bandwidth f_b, where f_b is the highest frequency of the noise signal that passes the proces and has not decreased by a decay factor to a certain level. Usually this decay factor is chosen to be 1%, which means that the noise power has decreased by 40 dB.

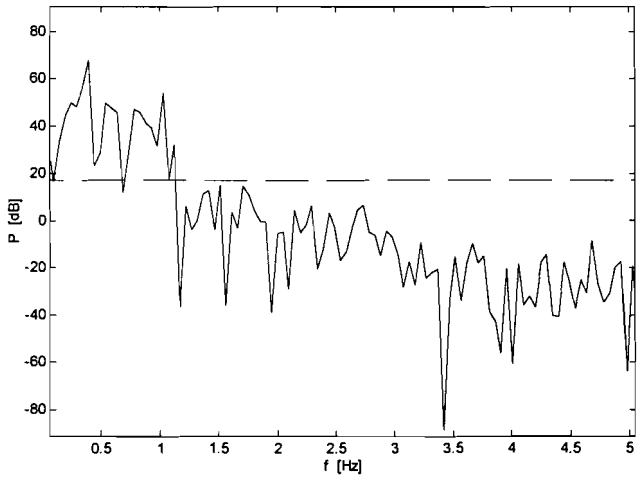

Figure *4.4: Power spectrum of output* H_1.

The Nyquist frequency should be at least f_b, to capture all the relevant system dynamics. This frequency can be determined from figure 4.4; it's approximately 1.2 Hz. So, the frequency area of the system dynamics lies between 0.05 and 1.2 Hz. This means that the system has become slower by mounting the crane on the platform and increasing the size of the floats; before the modifications the bandwidth was from 0.07 Hz to 1.5 Hz.
An other point to investigate is to examine whether there exists a delay between input and output or not; the system may not respond immediately at the input signal, there may be some delay. If there's a delay in the system, it can be determinated by an impulse response, or by a cross correlation between input and output:

$$\Phi_{uy} = E(u[n]y[\tau - n])$$

Because the input signal is white noise, the cross correlation between input and output is a rough estimation of the impulse response. In fig. 4.5 the cross correlation between X_1 and H_1 is plotted. As can be seen from this plot, no time delay exists.

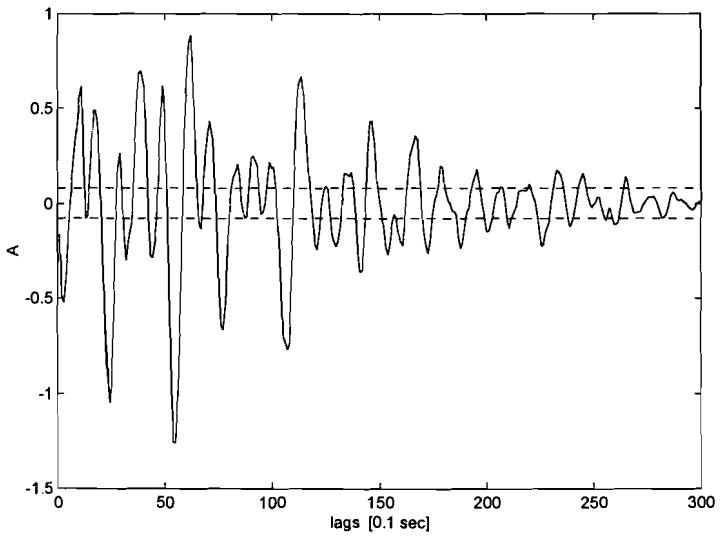

Figure 4.5: *Cross correlation between input X_j and output H_j.*

This was also the same for the other input/output relations. After approximately 25 seconds the output stays within the confidence interval. This means that from this point the output is hardly correlated with the input. Now the waves are static or reflected and behave chaotic.

Orders of the servo systems

In the ideal situation, a servo system can be represented by an integrator, or by a first order system if feedback is used. We have checked this by an identification of the dynamic behaviour of the servo systems. For this identification the inputs were U_i, which are the input voltages for the servo systems and the outputs were the measured voltages from the LVDT's, which gives us an indication about the distance between the float and the platform. In the plots below the validation of a second and a third order estimation are depicted. As follows from these pictures, a second order representation of the servo system isn't very accurate. The third order estimation follows the real value very acceptable. These poles can probably be divided into one feedback pole of the servo system, one mechanical pole and one electrical pole caused by the resistance and the coil of the motor.

This high order makes it quite difficult to see the servo's and the platform dynamics as two separate systems during the identification procedure. The software for trainig the dynamic neural network $\hat{f}$, which should capture the system dynamics, is not developed for hybrid identification. Therefore, from now on, the platform, including the servos will be considered as one black box. This means for identification of the process that U_i will be the inputs and H_i will be the outputs

For further usage of the servo systems we fed back the system by a gain, such that the lowest cuttoff frequency of the system lays at approximately 10 Hz.

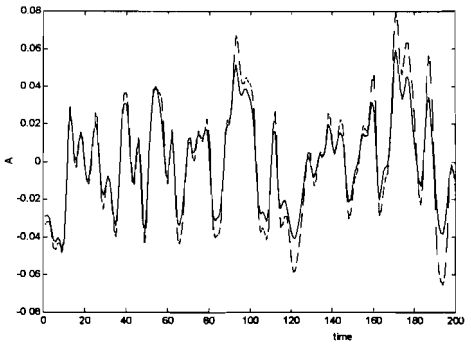

Figure 4.6a : *Second order output error validation. Every time step is 0.1 second.*

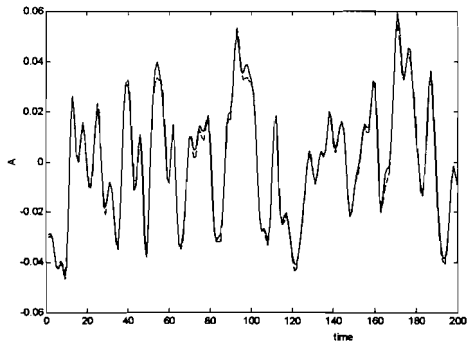

Figure 4.6b : *Third order output error validation.*

In the figure beneath a schematical overview of the frequencies and bandwidths for the identification is plotted.

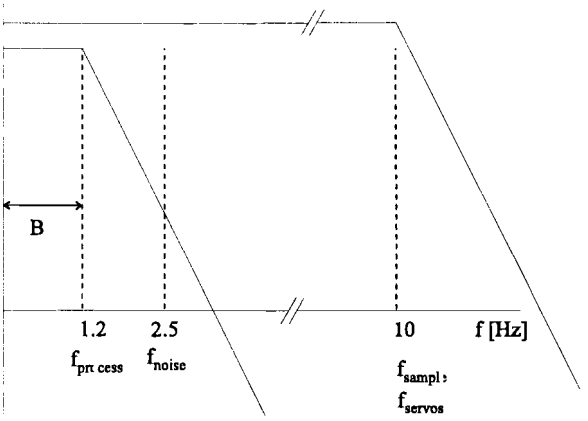

Figure 4.7: *Schematical overview of important frequencies during identification.*

As we can see, the influence of the servo systems in the process bandwidth is negligible and can be considered as constant.

Rough order estimation

To get an prior idea of the order of the number of poles, needed to model the dynamics of the system, is by means of the Hankel matrix and singular value decomposition. The order as calculated this way can be used as an indication for the modeling. Later on, the validity of the model order should be determined more accurately by the cost function.
The order can be calculated as follows:

An impulse response of the system is used to get an impression. The servo systems are included

in the total system. In figure 4.8 the impulse response U_1 to H_1 is depicted. Further, for the SISO-identification the impulse responses of the average height H_a and the rotation angles θ_y and θ_x are also measured.

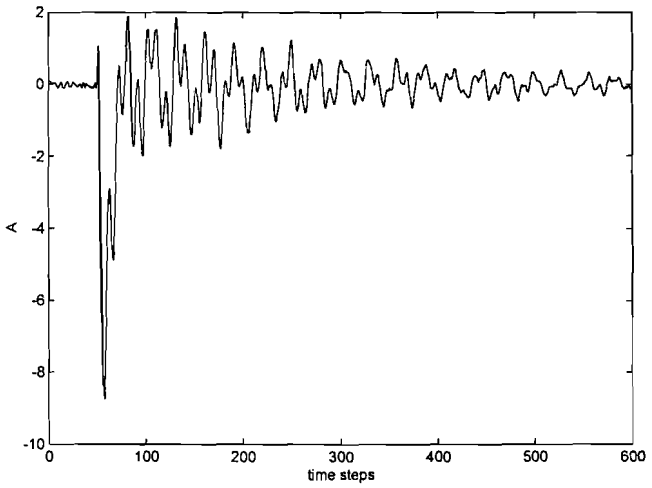

Figure 4.8: *Impulse response from U_1 to H_1.*

The responses are taken in a small amplitude area, to minimize the nonlinearities, but not too small so the response can be distinguished from the noise. To obtain the impression, consider the impulse response as a linear proces which can be perfectly modeled. The output of this proces, *y[n]*, is disturbed (superimposed) with Gaussian white noise, which together form the measured impulse response *h[n]*:

$$h[n] = y[n] + n_o[n]$$

The impulse response is of length N. N is chosen so that it was obvious that the information for $n>N$ was not related with the inputs anymore. According to figures 4.3 and 4.5 this results in an impulse response of 25 seconds. So N represents 250 data points with a sample frequency of 10 Hz. This means the impulse response can be perfectly modeled with an ARMA model of order N/2. The model impulse response can now be expressed in the Auto Regressive (AR) parameters a_p and the impulse response *h[n]*:

$$y[n] + n_o[n] = \begin{cases} h[n] & 0<n<\frac{N}{2}-1 \\ -\sum_{p=\frac{N}{2}}^{N-1} a_p h[n] = h[n] & \frac{N}{2} \le n \le N-1 \\ 0 & n>N \end{cases}$$

If the noise level isn't too high,the latter expression can be solved perfectly to obtain a_p.
If the ARMA model is not chosen equal to N/2, the expression can be solved in least square

sense. The energy transfer from past inputs to future outputs can be investigated in order to find out how many AR parameters are needed. The future outputs are entries in a vector:

$$\underline{y}_f[n] = \left(y\left[\frac{N}{2}\right] \quad y\left[\frac{N}{2}+1\right] \quad \cdots \quad y[N-1] \right)^T$$

The from the past inputs to the future outputs transferred energy E_t can be written as:

$$E_t = \sum_{n=\frac{N}{2}}^{N-1} y_f^2[n] = \text{trace}\left(\underline{y}_f^T \underline{y}_f\right)$$

where $y_f[n]$ is the n^{th} diagonal element.
The Hankel matrix is given by:

$$\mathbf{H} = \begin{pmatrix} h[1] & h[2] & \cdots & h\left[\frac{N}{2}-1\right] \\ h[2] & h[3] & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ h\left[\frac{N}{2}-1\right] & h\left[\frac{N}{2}\right] & \cdots & h[N-1] \end{pmatrix}$$

H can be separated in a Hankel matrix due to model $y[k]$, and a Hankel matrix due to the additive noise $n[k]$:

$$\mathbf{H} = \mathbf{H}_y + \mathbf{H}_n$$

The output power E_t is now given by the following expression:

$$\text{trace}\left(\underline{\mathbf{y}}_f^T \underline{\mathbf{y}}_f\right) = \text{trace}\left(\underline{\mathbf{u}}_p^T \mathbf{H}^T \mathbf{H} \underline{\mathbf{u}}_p\right)$$

$$\Leftrightarrow \text{trace}\left(\underline{\mathbf{y}}_f^T \underline{\mathbf{y}}_f\right) = \text{trace}\left(\underline{\mathbf{u}}_p^T (\mathbf{H}_y + \mathbf{H}_n)^T (\mathbf{H}_y + \mathbf{H}_n) \underline{\mathbf{u}}_p\right) \qquad (4.27)$$

with:

$$\underline{\mathbf{u}}_p = \left(u[1] \; u[2] \; \cdots \; u\left[\frac{N}{2}-1\right] \right)^T$$

The formulas above express how long energy passes through the system by the excitation $\mathbf{u}_p$, which remains in the system formed by **H**. Using singular value decomposition, the Hankel matrix can be written as a product of three new matrices, the unitary matrices **U** and **V**, and the diagonal matrix **S** with nonnegative elements in decreasing order so that:

$$\mathbf{H} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T$$

This can be applied to equation (4.27):

$$\text{trace}\left(\underline{\mathbf{y}}_f^T\,\underline{\mathbf{y}}_f\right) = \text{trace}\left(\underline{\mathbf{u}}_p^T(\mathbf{USV}^T)^T(\mathbf{USV}^T)\underline{\mathbf{u}}_p\right)$$

$$\Leftrightarrow \text{trace}\left(\underline{\mathbf{u}}_p^T\mathbf{VS}^T\mathbf{U}^T\mathbf{USV}^T\underline{\mathbf{u}}_p\right)$$

$$\Leftrightarrow \text{trace}\left(\underline{\mathbf{u}}_p^T\mathbf{VSISV}^T\underline{\mathbf{u}}_p\right)$$

$$\Leftrightarrow \text{trace}\left(\underline{\mathbf{u}}_p^T\mathbf{VS}^2\mathbf{V}^T\underline{\mathbf{u}}_p\right)$$

The diagonal matrix $\mathbf{S}^2$ is given by:

$$\mathbf{S}^2 = \mathbf{S}_y^2 + \sigma_n^2\mathbf{I} + 2\mathbf{S}_y\mathbf{S}_n\cos(\chi)$$

In this expression only the first P samples of $\mathbf{S}_y^2$, with P the true order, are unequal to zero. However, these P parameters cannot be retrieved from $\mathbf{S}^2$, because they are biased with the noise energy σ_n^2 and a disturbance caused by the cross products of the true system and the imposed noise: $2\mathbf{S}_y\mathbf{S}_n \cos(\chi)$. For not too high noise levels, most of the energy from past inputs to future outputs is transferred over the true singular values. This can be applied to the impulse responses. The result are plotted in figures 4.9-12. The orders can now be estimated and are given by the number of squared singular values at which these values approach the noise level. Normally a sharp decrease to a certain level (the noise level) can be observed. This has been done with the four mentioned impulse responses.

For the MIMO-approach with U_1 to H_1 the singular values plot, depicted in figure 4.9, are decreasing at three clear edges, namely at three, five and seven. The latter was also chosen by [REIJ96] and [HEU94] during their estimation phase. However, the order derived by physical modeling by [HEU94] was four (see section 2.4). This is derived by the fact that output H_1 is dependent on a translation (H_a) and a rotation (θ_x or θ_y), according to figure 2.2 The orders five and seven probably mean that some poles of the servo system will be included, like explained on page 51.
For the SISO-approach the three singular value plots are shown in figures 4.10-12. The singular values of U_a to H_a are decreasing after order two and order six. The physical model is of order two (see section 2.4 and [HEU94]). Some poles of the three servo systems will be included. This means that the servo systems don't have the same pole positions. The rotation around the y-axis shows that the system has an order two or four. The singular value plot of the rotation around the x-axis shows more edges: at orders two, five and eight. The physical model is of order two. The reason for the fact that the plots for θ_y and θ_x differ is probably caused by the nonsymmetric placement of the platform in the tub: according to picture 2.2 output θ_y is only dependent on movements of two servo systems, θ_x depends on three servo systems.

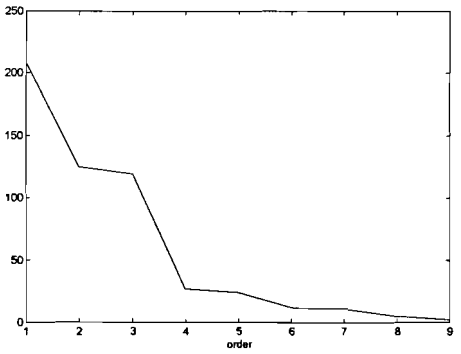

Figure 4.9: *Singular values of U_l to H_l.*

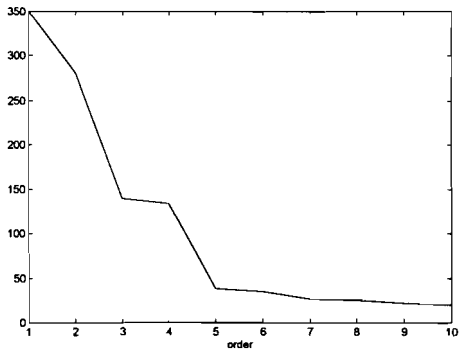

Figure 4.10: *Singular values of U_a to H_a.*

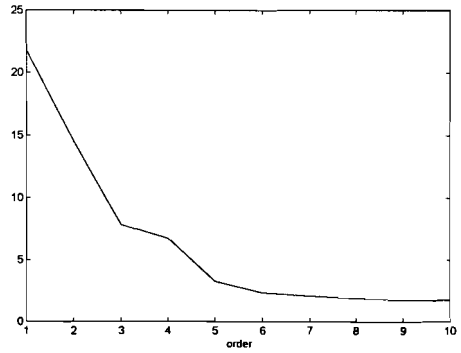

Figure 4.11: *Singular values of U_y to θ_y.*

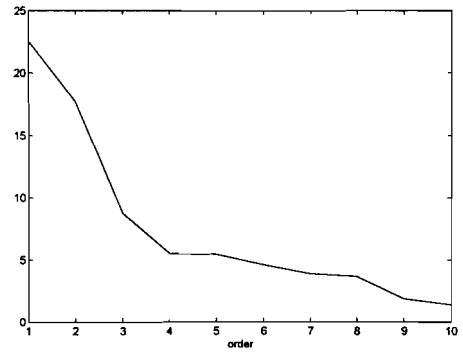

Figure 4.12: *Singular values of U_x to θ_x.*

Input signals and data set

To create a useful data set for estimating the process dynamics and non-linearities, the process was excited with a uniformly distributed, random white noise sequence, low pass filtered with a fifth order butterworth filter, with a cut-off frequency $f_f = f_c$, where f_c is the clock frequency of the input signal (= 2.5 Hz). This doesn't effect the range and phase of the output signal in the process bandwidth. This input signal is almost smooth in the area of the process bandwidth, which is 1.2 Hz. The periodicity in the frequency domain is due to the block shape of the noise signal in the time domain; the Fourier transform of a block is Sinc-function. Therefore, where f_s is a mutiple of f_c, the power spectral density will be zero. The range of the input signal was chosen as big as possible, which means that the platform was excited almost touching the buckets and the borders of the tub. The noise is chosen to be uniformly distributed, to excite the process nonlinearities persistently. The data was gathered by a sample frequency of 100 Hz, which was decimated to 10 Hz after collection. The experiment time was 240 seconds, so 2400 samples where available.

The scheme in figure 4.15 shows the Simulink structure that was implemented in the DSP for the real time data acquisition. The outputs of the process are explained in chapter two. K_{1a}, K_{2a} and K_{3a} are the feedback gains for the servo systems which are called K_{fb} in section 2.6.

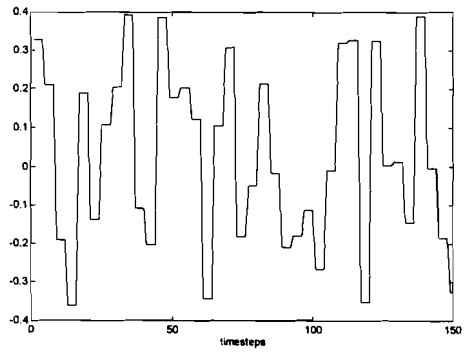

Figure *4.13: Uniformly distributed white noise for inputs U_i, one time step is 0.1 second.*

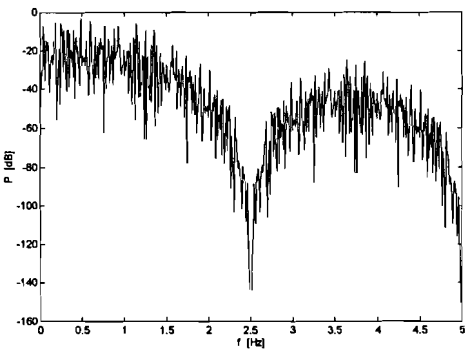

Figure 4.14*: Power spectrum of the input signal.*

K_1, K_2 and K_3 are the forward gains for the servo systems, which are called K_{ff} in section 2.6. The Butterworth filters are used for bandlimiting the white noise. The look-up tables are containing the calibration values for both the height and float sensors. The transformation box 'subsystems' contains the transformations described in section 2.1.

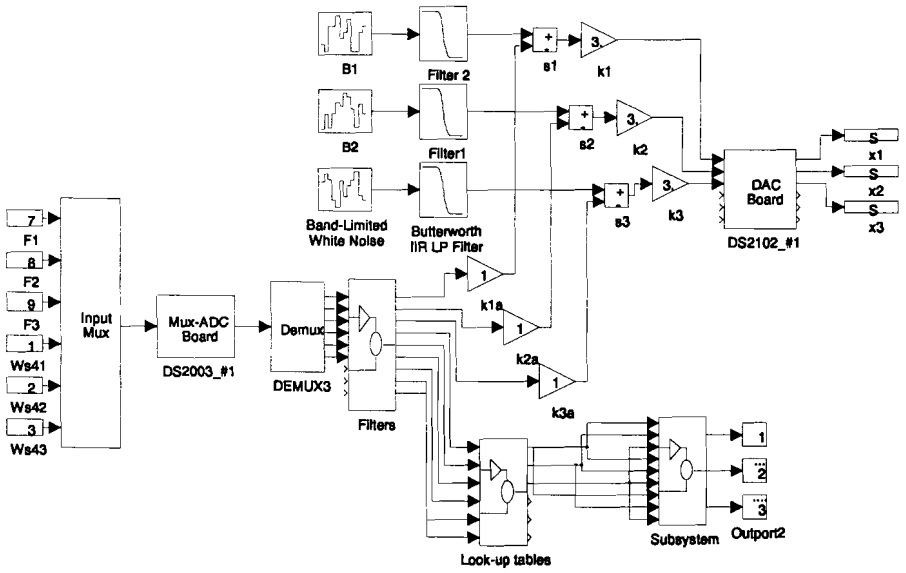

Figure 4.15: *Simulink block scheme for data acquisition.*

The signals F_i and $Ws4_i$ are respectively the measurement signal of the LVDT's which are indicating the heights and the measurements signals of the pressure sensors.
After aquisition the data had to be prepared for estimation. The offset was removed, and the data was scaled. Further, the data was filtered with a fourth order Butterworth filter with a cut-off frequency of approximately 4 Hz. The first and last few samples were not used, which resulted in a data set of 2071 samples. 1500 samples are used for estimation, 571 for validation.

4.5 The validation model in Simulink

For validation of the results we made a model in Simulink, like depicted beneath. Because the weight vectors were formatted in another way by the estimation software than the format in Simulink, a conversion program has been written. This is shown in appendix B.

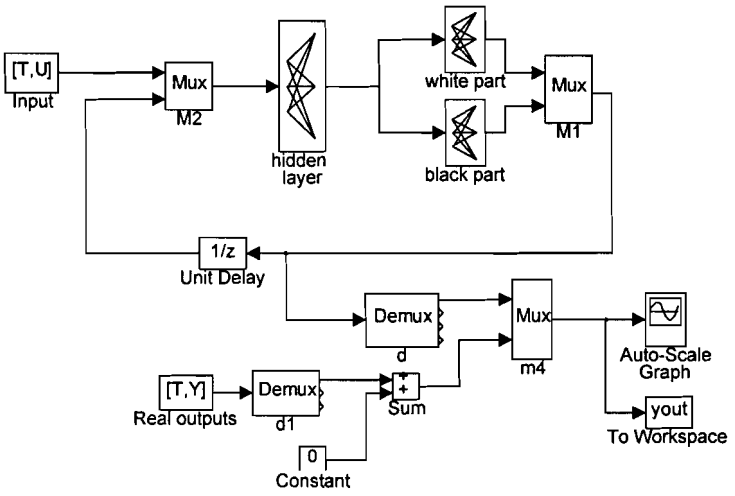

Figure 4.16: *Simulink block scheme for simulation.*

The blocks [T,U] and [T,Y] represent the input and output data for validation. The blocks 'hidden layer' and 'black part' contain layers with sigmoid activation functions, the block 'white part' contains linear nodes.

4.6 Estimation of the MIMO-model

The platform can be identified in several ways, based on the matrix descriptions in chapter 2. At first the MIMO-description will be used. In this method the servo inputs X_1, X_2 and X_3 are the inputs of the process and the heights H_1, H_2 and H_3 and the wave heights Δ_1 , Δ_2 and Δ_3 are the outputs of the process. The white part of the state vector will be equal to the outputs:

$$\underline{y} = \underline{x}^1 = (H_1 \;\; H_2 \;\; H_3 \;\; \Delta_1 \;\; \Delta_2 \;\; \Delta_3)^T$$

This means that the output map h will be the identity matrix. For the estimation software we have to implement the Jacobian matrix of the output map h. Generally this Jacobian is calculated as in equation 4.36. Due to the fact that the output map was chosen as the identity matrix, this results in a Jacobian which is again an identity matrix.
The first estimate attempts were done with a neural network with one hidden layer with 8-10 nodes and 2-3 hidden states. This seemed to be a reasonable choice. A lot of runs were made by the Alpha station, but almost every time the program was interrupted by an arithmetic trap. This trap was caused by a too large condition number of the Hessian matrix, used in the quasi-Newton

optimization routine. The idea for the large value of this ratio was that one or more nodes got stuck in their saturation area. The condition number was about 5e12, which indicates the ratio

$$\begin{pmatrix} \frac{\partial f_1(\underline{x})}{\partial \underline{x}} \\ \vdots \\ \frac{\partial f_n(\underline{x})}{\partial \underline{x}} \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(\underline{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\underline{x})}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(\underline{x})}{\partial x_1} & \cdots & \frac{\partial f_n(\underline{x})}{\partial x_m} \end{pmatrix} \tag{4.36}$$

between the largest and the smallest singular value of this matrix. This number is also more or less the machine accuracy. This means that the largest gradients became too small, because of:

$$\lim_{u\to\infty} \frac{\partial s(u)}{\partial u} = \lim_{u\to\infty} \left(\frac{1}{1+e^{-u}}\right)\left(1-\frac{1}{1+e^{-u}}\right) = 0$$

Some measures in the software were taken (if-then statements) to specify upper and lower bounds for the values of u. This avoids too big differences between the largest and the smallest singular value of the Hessian. This worked out, the traps didn't occure anymore. Another idea to avoid such problems (which is not implemented so far) is to use a second cost function for the weight values, which should be kept minimal. In this way large weight values which cause too much saturation will be avoided.

After this, the results were very poor. This was caused by the strange dynamic behaviour of the wave height sensors. The algorithm didn't succeed in finding a proper fitting of these signals. Especially the fitting of the signal Δ_2 showed some bad performance. Probably the representation of this output in the dataset was disturbed. This could be caused by some failures in the electronic circuit or by oxidation of some particular areas on the sensor surface. Therefore, the outputs were MISO-identified; one by one using a dataset with three inputs and one output and six states, to see what was going wrong. Table 4.2 shows the experiment data.

Experiment type	Multiple Input Single Output estimation (MISO)
Goal	Checking if all outputs can be identified well
Data length	1500 samples
Input signals	ZMWN, uniform distributed, cut off freq = 2.5 Hz
Network structure	3 inputs, 1 hidden layer with 6 nodes, 1 white state, 5 black states
Number of active weights	102
Number of SA iterations	20*N
Number of QN iterations	20*N

Table 4.2: *Experiment data of the MISO experiments.*

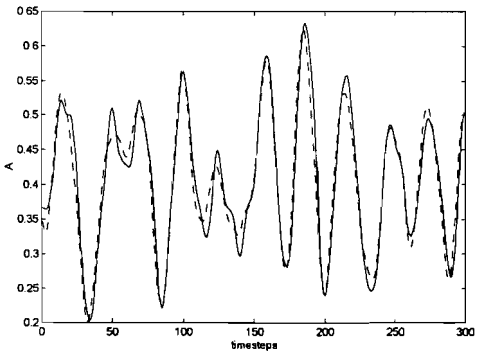

Figure 4.17: *MISO experiment with output H_1, real output is solid, validation is dotted.*

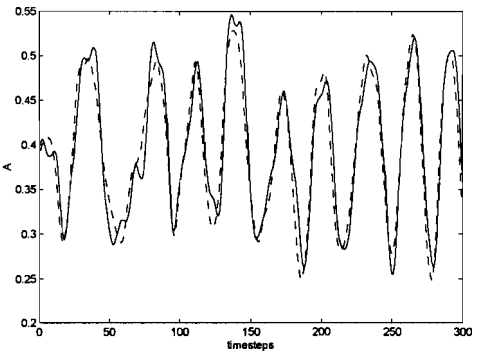

Figure 4.18: *MISO experiment with output H_2.*

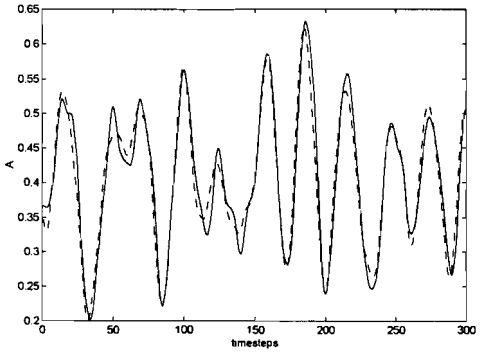

Figure 4.19: *MISO experiment with output H_3*

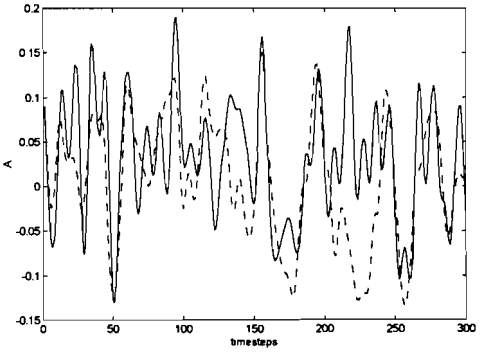

Figure 4.20: *MISO experiment with output Δ_1.*

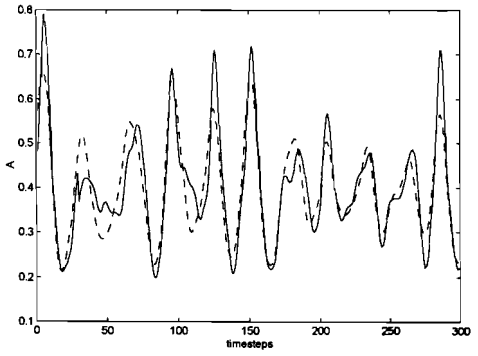

Figure 4.21: *MISO experiment with output Δ_3.*

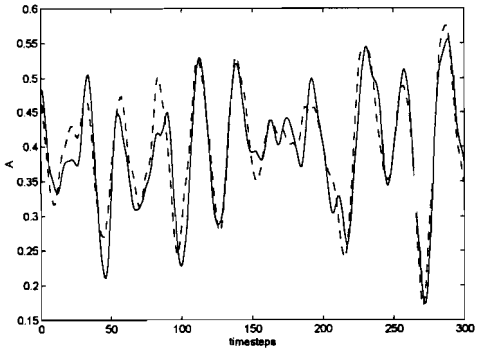

Figure 4.22: *MISO experiment with output Δ_2.*

The results are shown in figure 4.17-22. The identification of the signals H_1, H_2 and H_3 were very good: the error, as calculated by equation 4.10 was about 5e-5. The outputs Δ_1 and Δ_3 were not identified as well as the outputs H_i, but the performance was reasonable. The output Δ_2 was identified very poorly, this could not be used to design a controller. This bad performance of the wave height sensor identification was obviously caused by big high frequent peaks on the measurement signals. These peaks were partly removed by a sort of peak shaving algorithm: at first an upper bound and a lower bound were definied. Then all peaks outside these bounds were removed. Finally the signal was filtered again to get a smooth behaviour. But no improvement was shown. The usefulness of these measurements has been discussed and finally it was decided not to use them anymore.

So, now only the outputs H_i are considered as the white part of the state vector. The dimension of the black part was determined iteratively, with as default values the dimensions derived by the Hankel singular value decomposition. The loss function is plotted in figure 4.23. Two orders can be determined from this plot, namely five and seven.

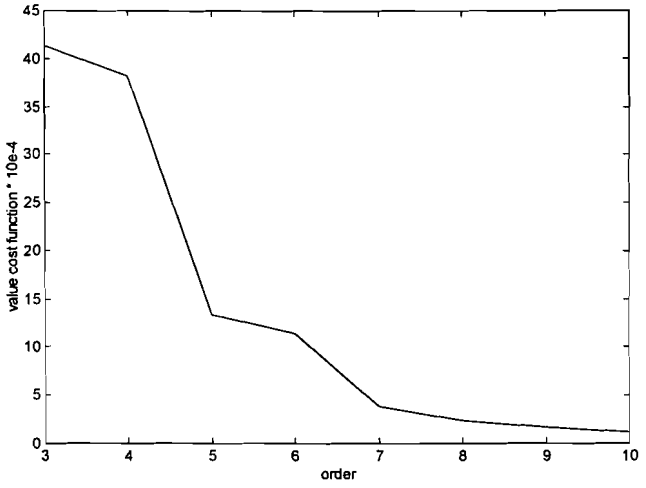

Figure 4.23: *Loss function.*

Compared with the singular value decomposition in section 4.6, there we found orders for the entire state vector of five and seven. So the orders found in practice are the same as found with the Hankel singular value composition. According to the loss function it is obvious that the orders five and seven will give the best results. This has been checked by looking at the power spectrum of the residu . This has been done for both fifth and a seventh order model. The results are plotted in figures 4.24-29. In the power spectral density of the residuals of the fifth order estimation can be seen that the main power of the residuals lie between 0.3 and 0.4 Hz. Comparing this with the power spectrum of the outputs as depicted in figure 4.4, we see that the resonance peak at 1.1 Hz is modelled well. But the peak at 0.38 Hz shows us that not all the dynamics in the neighbourhood of this peak are modelled. The main power in the residuals can be explained by static and reflected waves generated by the floats.

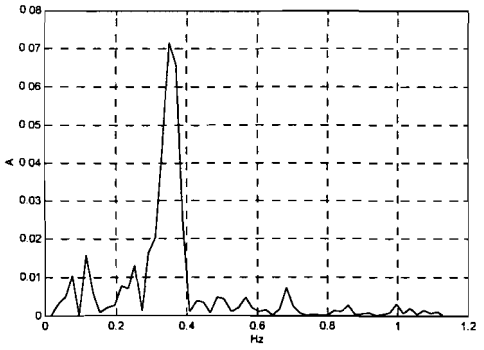

Figure 4.24: *Power spectra of residue of output H_1 with fifth order estimation.*

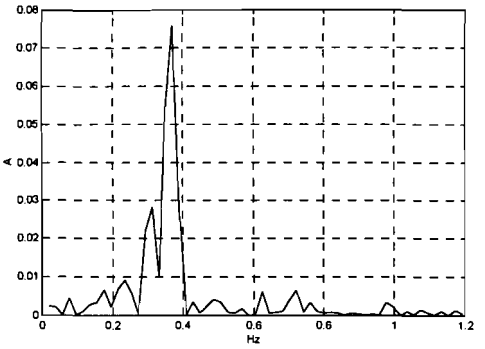

Figure 4.25: *Power spectra of residue of output H_2 with fifth order estimation.*

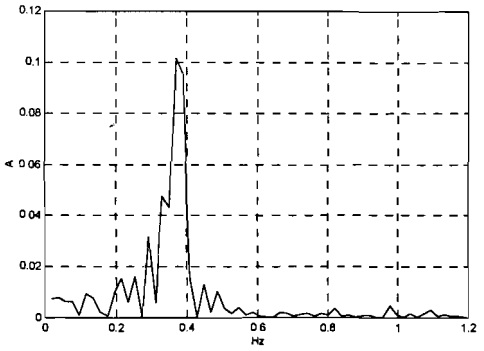

Figure 4.26: *Power spectra of residue of output H_3 with fifth order estimation.*

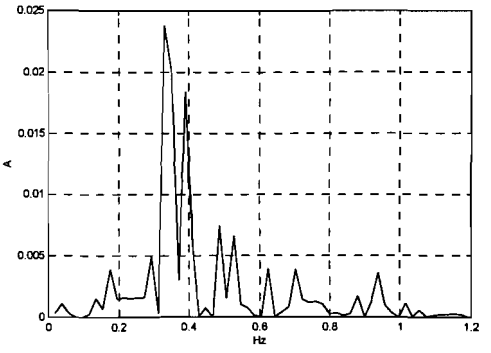

Figure 4.27: *Power spectra of residue of output H_1 with seventh order estimation.*

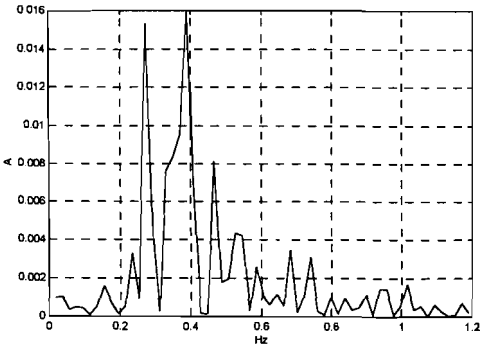

Figure 4.28: *Power spectra of residue of output H_2 with seventh order estimation.*

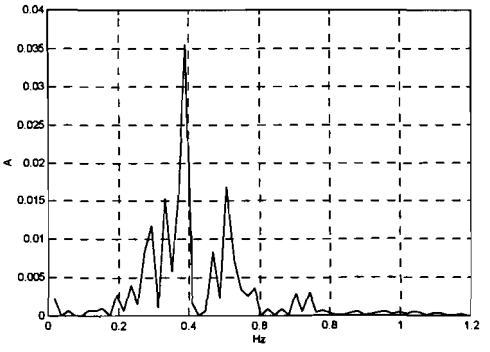

Figure 4.29: *Power spectra of residue of output H_3 with seventh order estimation.*

A wave front, caused by the movements of a float runs away from the float and returns, after reflected partly by the perforated screen, and smashes on the floats. In the power spectra density plots of the seventh order estimation can be seen that the peaks at 0.38 Hz are supressed quite a lot. Only output H_3 shows a considerable peak. It is remarkable that the main peak, which was at 0.38 Hz with the fifth order estimation, is now shifted towards 0.33 Hz, which was already there at the fifth order model.

This means that the increasing order leads to modelling of the dynamics around 0.38 Hz. Increasing the order further didn't lead to better performance. Some components in the waves couldn't be modelled. Perhaps this can be explained by the wild behaviour of the platform in the tub during the data aquisition. The horizontal movement of every float was about 10 centimeters around its centre of rest. So the distance between tub and floats can be varying for 20 centimeters which causes different time constants for the reflected waves. Such a time varying behaviour is hard to model. The fact that the performances for the three different floats differ is (probably) caused by the initial placement of the floats. Another data set probably would give other values for the residue.

The definitive results were gathered with a network with one hidden layer with four to nine nodes. More hidden layers didn't lead to further improvement. For every network complexity the optimization was started from six random initial points.
Experiment data is shown in table 4.3.

Experiment type	MIMO identification
Goal	Estimating a proper MIMO model
Data length	1500 samples
Input signals	ZMWN, uniform distributed, cut off freq = 2.5 Hz
Network structure	3 inputs, 1 hidden layer with 4-9 nodes, 3 white states, 4 black states
Number of SA iterations	20*N
Number of QN iterations	20*N

Table 4.3: *Experiment data for MIMO estimation.*

The results with the validation set were poor. This was mainly caused by over training (generalization). Probably the program tried to fit the static or reflected waves in the learning set. Reducing the number of iterations in the quasi-Newton procedure to $20N_\theta$ led to an marginal increase for the learning set cost function, but the results for the validation set became much better. The ratio between the learning and validation cost function was about 1.5. This couldn't be reduced further. For safety we changed the learning set and the validation set. But the results stayed the same. These results are plotted in table 2, the underlined result was taken as the final model. This means a network with one hidden layer with eight nodes. It is clear that a network with 7 or 9 nodes in the hidden layer will also give good performance.
After the experiments the results were discussed. A remarkable effect was discovered, which was not found in the literature studied so far.

N_N N_θ	J_e (10^{-4})	J_v (10^{-4})	N_N N_θ	J_e (10^{-4})	J_v (10^{-4})
4	3.4569	3.8239	7	1.1699	2.1043
	2.7993	3.3103		1.3105	2.2828
79	2.8191	3.1659	133	1.2850	2.7831
	2.9000	3.5399		1.2681	2.2250
	2.9726	3.5279		1.2309	2.2308
	2.9931	3.4754		1.5796	2.7235
5	2.2211	3.0879	8		
	2.3328	3.4149		1.1271	2.1636
97	2.0720	2.7626	151	1.1171	2.1292
	2.1951	3.0186		1.1677	2.0516
	1.9887	2.8054		1.2547	2.1799
	2.0617	2.7627		1.1553	2.2400
6	1.6468	2.3847	9	0.9849	2.2074
	1.4747	2.2733		1.0955	2.3257
115	1.6653	2.8491	169	1.0121	2.0351
	1.5190	2.4560		0.9448	2.1040
	1.7854	2.9419		1.0939	1.9612
	1.8824	2.6491		1.1305	2.3215

Table 4.3: *Results of the identification.*

This was the fact that the number of nodes in the hidden layer of a neural network network should be at least the number of process states if no pure delays are involved. This can be explained: Let A be the linearized matrix of a neural network f with p nodes and n states. Then the rank of A should be equal to p. If $p<n$, then the matrix A will not be square, the rank of A will be less than n, contradicting the fact that we deal with n nondelaying states. So the results on the left side in table 4.2 are not representing the results we wanted. This explaines the less good results of the left column. Probably during the estimation not all states have been used.
The validation of the results are shown in figures 4.30-32. The visual validation looks quite good, but its clear that the model is biased. In figures 4.33-35 the errors of the estimation are plotted. A clear wave pattern can be recognized. The main harmonic in the error signal seems to have a main frequency of 0.38 Hz. This is in relation to the results of the power spectral density plots and coincides with the main wave frequency which could not be modelled.
The correlation tests are plotted in figures 4.34-39. The auto correlation of the residu ξ_1 can not be considered as white noise; it shows coloured noise with a clear wave pattern. This means that dynamic information is hidden in this residue. This is trivial because the spectral density of this residue should be equal to figure 4.28 because of:

$$P_{xx} = \frac{1}{2\pi}\int_{-\infty}^{\infty}\phi_{xx}(\tau)\,e^{-j\omega\tau}d\tau$$

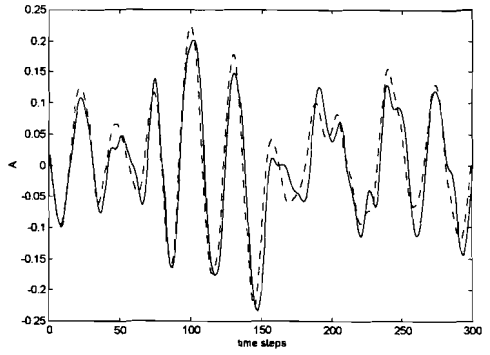

Figure 4.30: *Validation output H_1.*

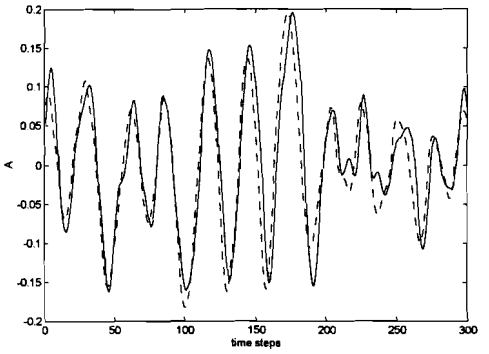

Figure 4.31: *Validation output H_2.*

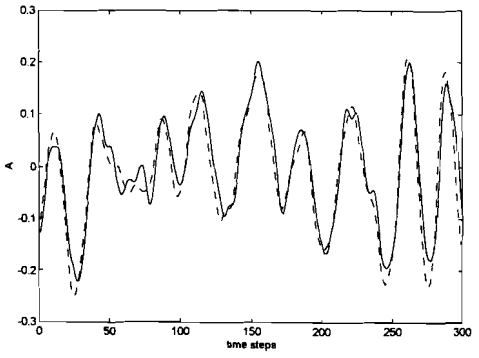

Figure 4.32: *Validation output H_3.*

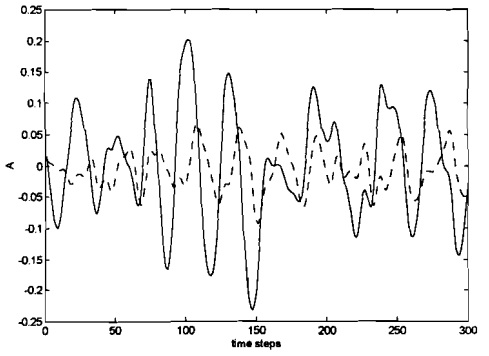

Figure 4.33: *Output H_1 and validation residue (dotted).*

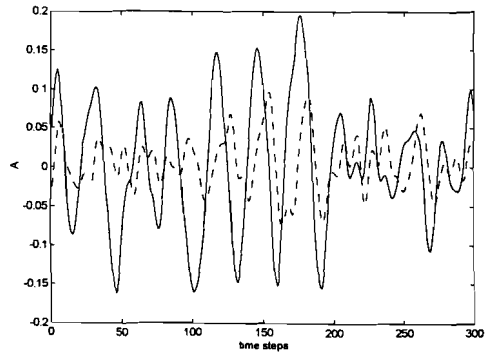

Figure 4.34: *Output H_2 and validation residue (dotted).*

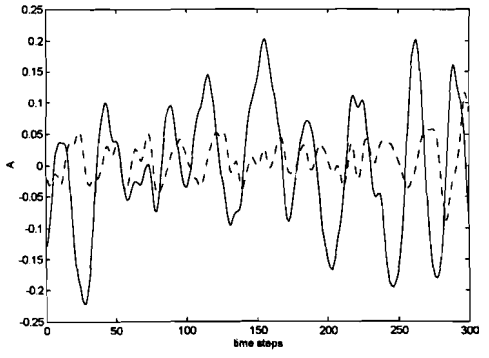

Figure 4.35: *Output H_3 and validation residue (dotted).*

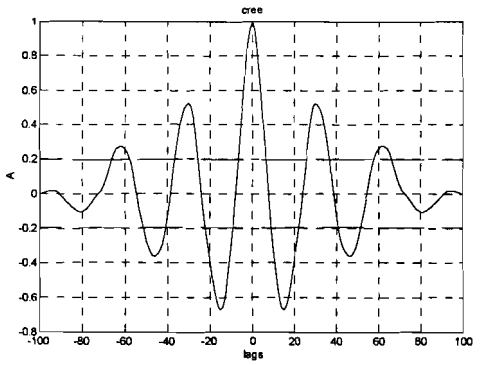

Figure 4.36: *Auto correlation of residue* ξ_1.

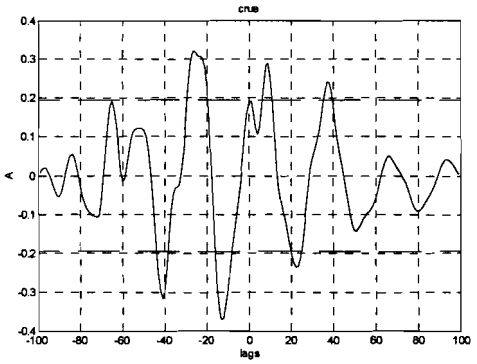

Figure 4.37: *Cross correlation of* X_1 *and* ξ_1.

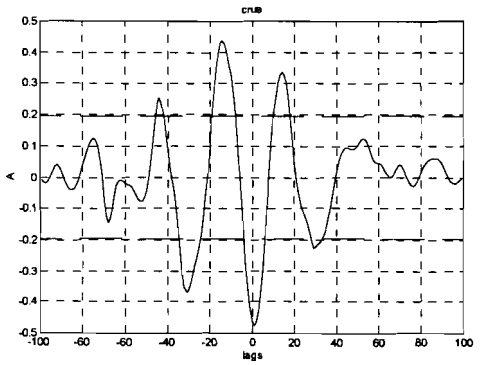

Figure 4.38: Cross correlation *of* X_2 *and* ξ_1.

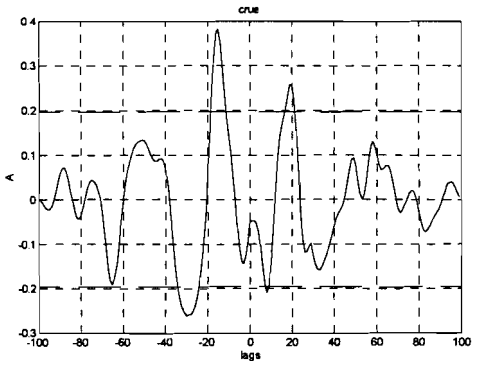

Figure 4.39: *Cross correlation* X_3 *and* ξ_1.

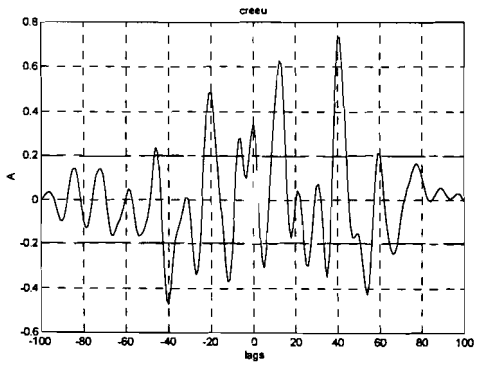

Figure 4.40: $\phi_{\xi(\xi u)}(\tau)$

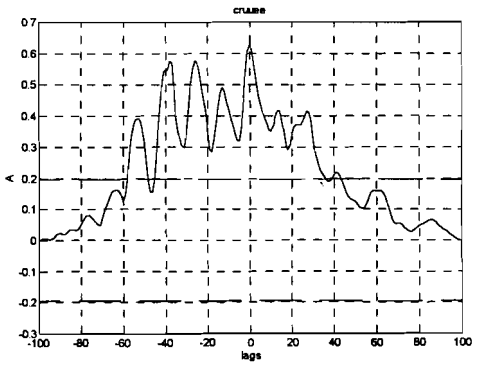

Figure 4.41: $\phi_{uu'\xi\xi}(\tau)$

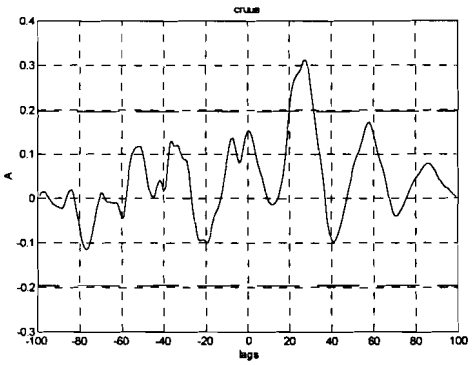

Figure 4.42: $\phi_{uu'\xi}(\tau)$

This is due to the fact that, as mentioned earlier, not all the wave effects are modelled. This figure denotes the wave frequency of approximately 0.4 Hz too. The output H_1 is also correlated with X_2 and X_3, in the same ratio as with X_1. This was also with the other outputs, as shown in appendix B. The cross correlations $\phi_{u\xi}(\tau)$, $\phi_{\xi(\xi u)}(\tau)$, $\phi_{uu'\xi\xi}(\tau)$ and $\phi_{uu'\xi}(\tau)$ are plotted in figures 4.36 and 4.37. We can extract two conclusions from these tests. At first we can conclude that there is internal noise in the model; the noise can not be considered as additive at the output and so there will be cross term products. Secondly, its obvious that the model is biased. This can be explained by the not modelled wave parts. These parts appear as coloured noise at the outputs of the neural network f and their feedback cause that the model is a biased predictor. [BIL92] recommends to increase or decrease the number of states, but we know that we can not further improve the results this way. Therefore we could include a submodel to our total model which can predict the static waves and/or the reflected waves. But these wave effects are chaotic and cannot predicted over a large horizon due to the strong dependency of the initial values of the float placement and movement. It has to be mentioned that looking at the validation set it seems more disturbed by nonstructural wave parts than the learning set. In chapter 5 we will try to predict these wave parts by a nonlinear dynamic Kalman-filter, for short horizons.

4.7 Results of identification, using the transformation Ψ.

After this, we tried to estimate a model using the complete structure as depicted in figure 2.7. Now the inputs for the neural network f are the average height U_a, and the aimed rotations over the x- and y-axis U_y and U_x. The transformation Ψ will be implemented in the output map h:

$$\begin{pmatrix} H_a(t) \\ \sin\theta_y(t) \\ \sin\theta_x(t) \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 1 & 1 & 1 \\ \frac{1}{L} & \frac{-1}{2L} & \frac{-1}{2L} \\ 0 & \frac{\sqrt{3}}{L} & \frac{-\sqrt{3}}{L} \end{pmatrix} \begin{pmatrix} H_1(t) \\ H_2(t) \\ H_3(t) \end{pmatrix}$$

So, by moving the sinus terms to the right expression, the output vector will be:

$$\underline{y} = (H_a \;\; \theta_y \;\; \theta_x)^T$$

The state vector will be divided in a known white part and an unknown black part.
The state vector of the neural network *f* will contain a white part. This white part will be:

$$\underline{x}^1 = (H_1 \;\; H_2 \;\; H_3)^T$$

For the gradient calculation of the output map during the dynamic optimization of the neural network $f(x,u)$, we need the Jacobian of *h*. If we make use of the next relation:

$$\frac{\partial \arcsin(\theta/a)}{\partial \theta} = \frac{1}{\sqrt{a^2-\theta^2}}$$

With *a* a constant. Then the Jacobian representation of *h* is:

$$\begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{\sqrt{(\frac{3L}{2})^2-(\xi_1)^2}} & \frac{1}{\sqrt{(-3L)^2-(\xi_2)^2}} & \frac{1}{\sqrt{(-3L)^2-(\xi_2)^2}} \\ 0 & \frac{1}{\sqrt{(\frac{L}{\sqrt{3}})^2-(\xi_3)^2}} & \frac{1}{\sqrt{(\frac{L}{\sqrt{3}})^2-(-\xi_3)^2}} \end{pmatrix}$$

with:

$$\xi_1 = H_1(t) - \frac{1}{2}H_2(t) - \frac{1}{2}H_3(t)$$

$$\xi_2 = -2*H_1(t) + H_2(t) + H_3(t)$$

$$\xi_3 = H_2(t) - H_3(t)$$

This is implemented in the estimation software[3]. This means we will estimate a MIMO model which, in linearized situation and with no wave disturbances, should be a diagonalization of the results in the previous section according to three independent outputs described in section 2.1. In

[3] It has to be mentioned that in the range of edges (maximum of 8 degrees) also the linear approximation can be used because of the slight difference in this area (<1%).

practice this will not realisable, because the wave coupling and static waves will cause interaction between the independent assumed inputs and outputs.
The identification procedure was started with one hidden layer and the number of nodes in this layer was first chosen to be nine, to get a view at the loss function for dimensioning the state vector. The dimension of the state vector was chosen in the range from four to nine. The total loss function is plotted beneath.

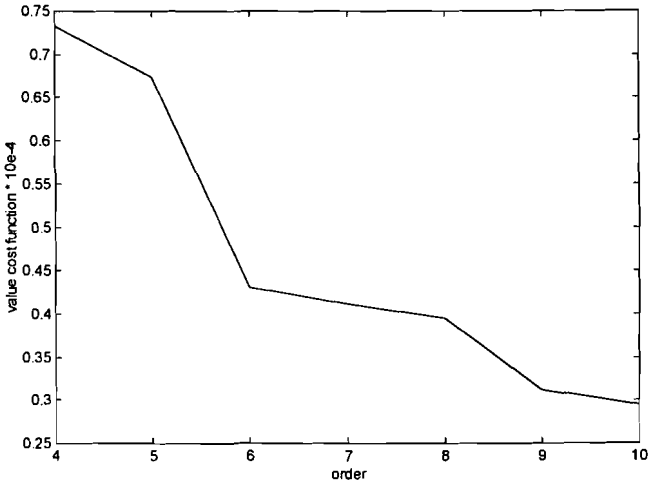

Figure 4.43: *Loss function of MIMO modelling with independent inputs and outputs.*

The sharp decrease which occurs for a sixth order model is conform to the physical modelling, which says that this order is due to three mass spring systems of second order, as explained in section 2.1. However, a state dimension of nine shows an even sharper decrease, probaly caused by modelling of structural wave parts. So, we choose a final state vector dimension of nine for the estimation procedure. We varied the number of nodes in the hidden layer from four to nine. In this procedure every network complexity was optimized with four random initial points. The experiment data is given by table 4.4. The results were quite good. Especially the rotation angle θ_x was identified very well. The results are shown in table 4.5. The dashes mean that the optimization got stuck in a local minimum. The validation results are plotted in figures 4.44-56.

Experiment type	MIMO estimation
Goal	Derive MIMO model with independent inputs and outputs
Data length	1500 samples
Input signals	ZMWN, uniform distributed, cut off freq = 2.5 Hz
Network structure	3 inputs, 1 hidden layer with variable number of nodes, 3 white states, 6 black states
Number of SA iterations	20*N
Number of QN iterations	20*N

Table 4.4: *Experiment data.*

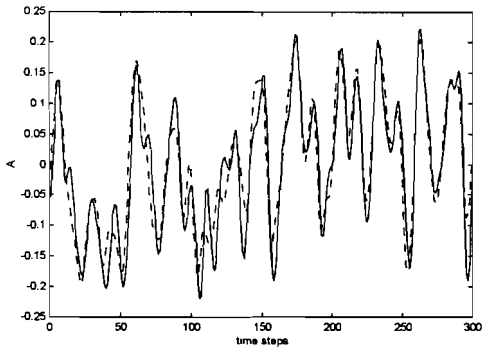

Figure 4.44: *Real (solid) and estimated (dotted) output H_a.*

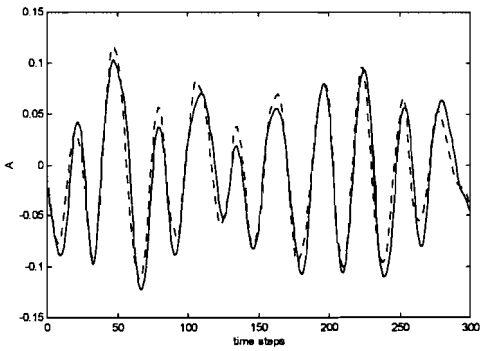

Figure 4.45: *Real (solid) and estimated (dotted) output θ_y.*

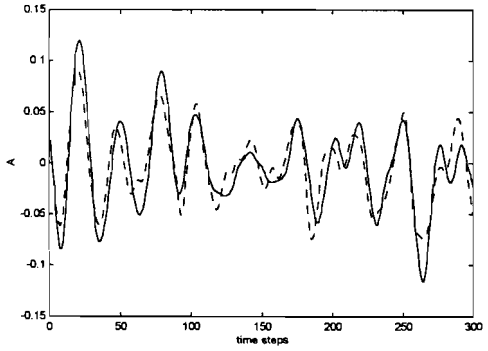

Figure 4.46: *Real (solid) and estimated (dotted) output θ_x.*

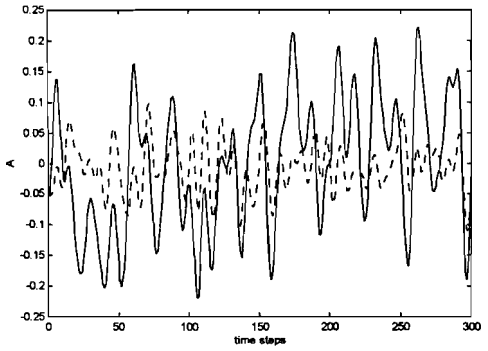

Figure 4.47: *Real output H_a (solid) and estimation error (dotted).*

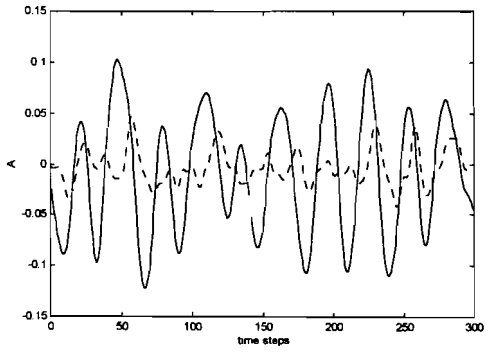

Figure 4.48: *Real output θ_y (solid) and estimation error (dotted).*

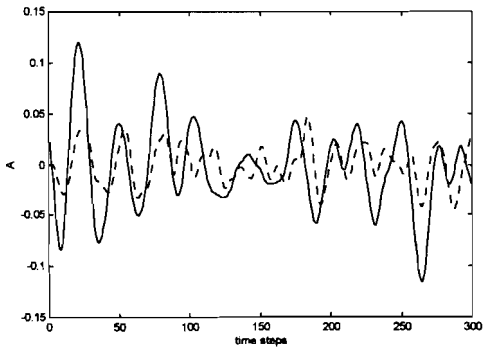

Figure 4.49: *Real output θ_x (solid) and estimation error (dotted).*

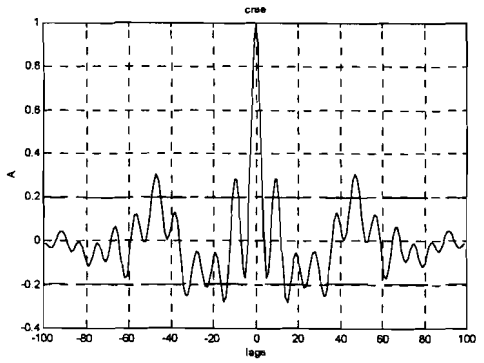

Figure 4.50: *Auto correlation of residue of output H_a.*

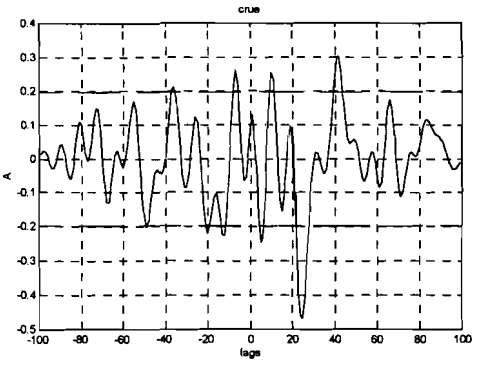

Figure 4.51: *Cross correlation between input X_a and output H_a.*

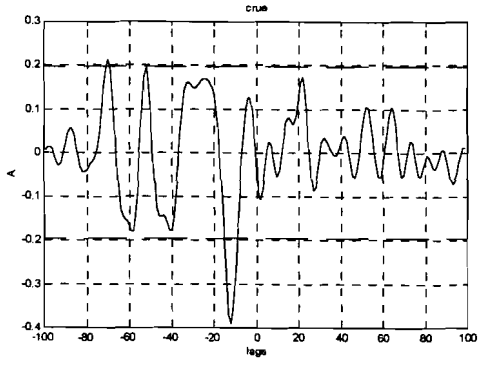

Figure 4.52: *Cross correlation between input X_y and output H_a.*

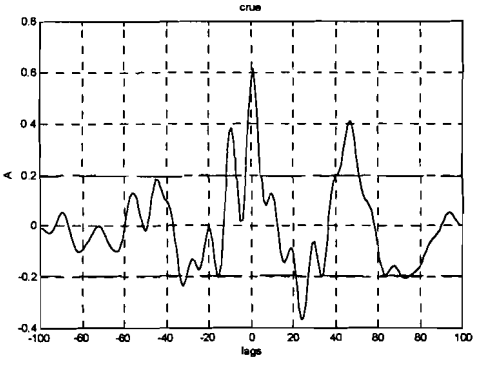

Figure 4.53: *Cross correlation between input X_x and output H_a.*

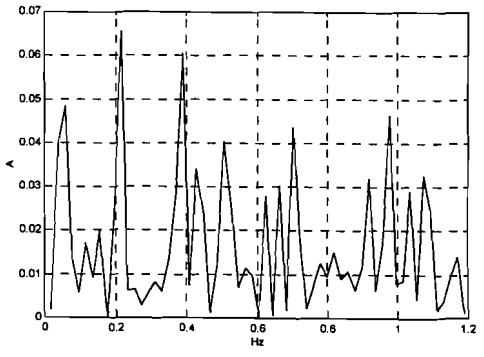

Figure 4.54: *Power spectral density of residue of output H_a.*

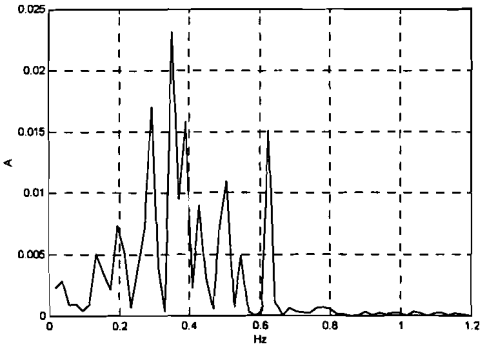

Figure 4.55: *Power spectral density of residue of output θ_y.*

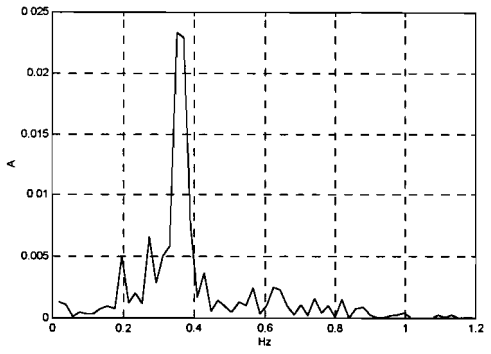

Figure 4.56: *Power spectral density of residue of output* θ_x.

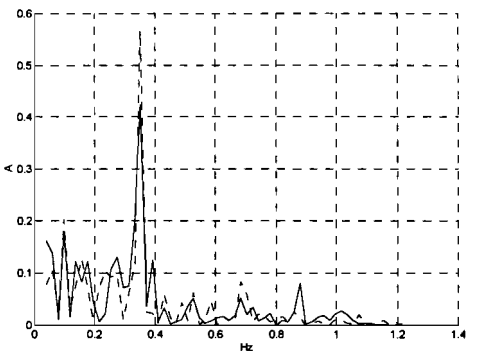

Figure 4.57: *Power spectral density of real output (solid) and estimated.*

The correlation tests for the rotation axes are plotted in appendix C. The approximation accuracy for H_a, θ_y and θ_x are respectively 8.56%, 2.87% and 9.08%. It was remarkable that estimated networks with eight or nine nodes in the hidden layer showed some oscillating behaviour. We cannot explain this. The power spectral of the residues of the residues show a main peak at 0.38 Hz, like the MIMO-model. The spectra of the average height residue shows more peaks. This can be explained that the peak around 0.35 Hz is modelled well (fig 4.57). So what's left can be considered as noise, probably caused by reflected and standing waves with different time delays, caused by the different distances between the floats and the tubs. This means we can expect for each of the three floats two perpendicular standing waves. These will cause six different peaks in the frequency domain.

N_N N_θ	J_e (10^{-4})	J_v (10^{-4})	N_N N_θ	J_e (10^{-4})	J_v (10^{-4})
4	3.4569	3.8239	7	1.1699	2.1043
	2.7993	3.3103		1.3105	2.2828
97	2.8191	3.1659	163	1.2850	2.7831
	2.9000	3.5399		1.2681	2.2255
5	2.2211	3.0879	8	-	-
	2.3328	3.4149		1.1271	2.1636
119	2.0720	2.7626	185	1.1171	2.1292
	2.195	3.0134		1.1677	2.051
6	1.3468	2.0147	9	0.9849	2.2074
	1.4747	2.2733		1.0955	2.3257
141	1.6653	2.8491	207	1.0121	2.0351
	-	-		-	-

Table 4.5: *Results.*

For real SISO-estimation with three independent inputs and outputs, the next procedure can be followed: Try to estimate three separate SISO models, with simple neural networks and small

state vector dimensions, according to figures 4.10-4.12. After estimation take a more complex network with a state vector dimension equal to the sum of the dimensions of the separate SISO state vectors. Implement the weights in the new , more complex weight vector and give the interconnection weights a random default value in the linear area. Start the optimization again. The estimation program will now try to optimize the interconnection weights, which are representing the wave coupling. This idea was not brought in practice during this thesis work, due to a lack of time.

4.6 Estimation of a linear model for a reference

To proof the increase of performance while using a nonlinear model, also a linear model was estimated as a reference. The same software was used, but now the sigmoid activation functions were replaced by linear functions. We maintained the same network complexity as found to be the optimal one in section 4.4. This means a network with one hidden layer with eight nodes, and a state vector dimension of seven. This can be described by:

$$\hat{x}(k+1) = \hat{A}\hat{x}(k) + \hat{B}u(k)$$

$$\hat{y}(k) = \hat{C}\hat{x}(k)$$

In figure 4.58 the structure is plotted.

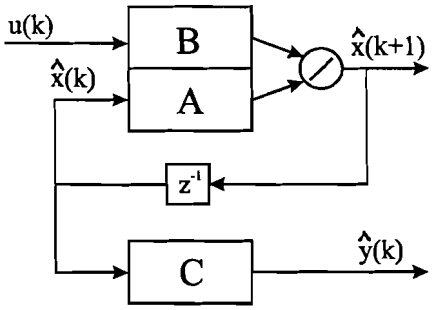

Figure 4.58: *Linear state space identification set-up: seen as a neural network with only linear nodes.*

The results are quite remarkable. The performance of the linear state-space model is almost as good as the nonlinear model. We found an optimal value for the cost function of the validation set of approximately 2.48e-4. The validation of the nonlinear model gaves us 2.06e-4.

In the table 4.6 the experiment data is given. For the nonlinear model we found an approximation accuracy of about 12,0%, for the linear model we found 14.0%. So clearly we haven't won much. The validation results are plotted in figures 4.59-67.

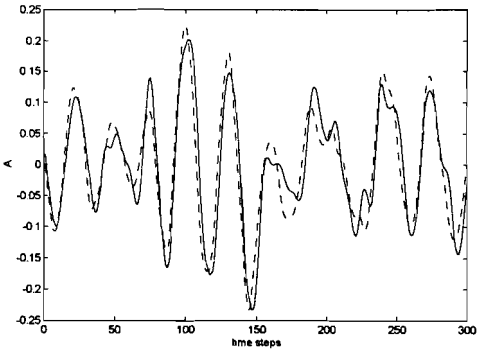

Figure 4.59: *Linear estimation of output H_1.*

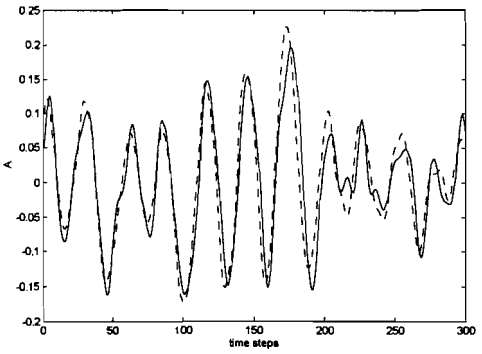

Figure 4.60: *Linear estimation of output H_2.*

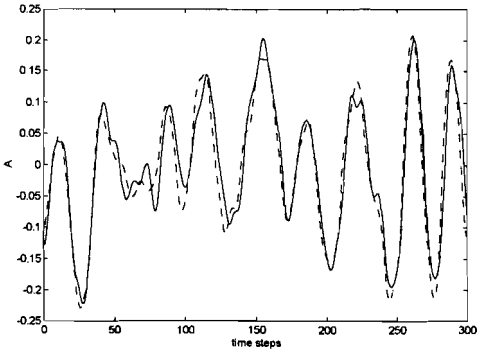

Figure 4.61: *Linear estimation of output H_3.*

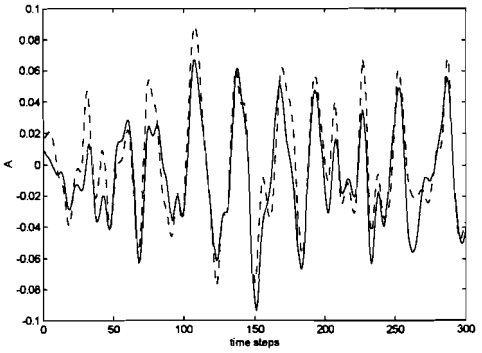

Figure 4.62: *Error of nonlinear and linear(dotted) estimation of output H_1.*

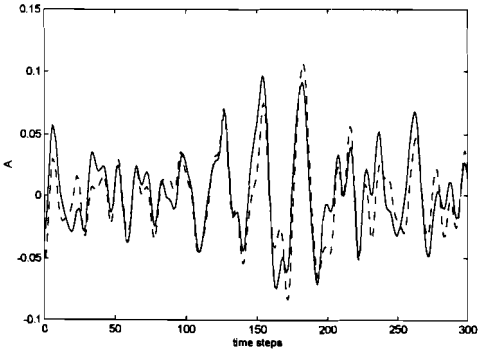

Figure 4.63: *Error of nonlinear and linear (dotted) estimation of output H_2.*

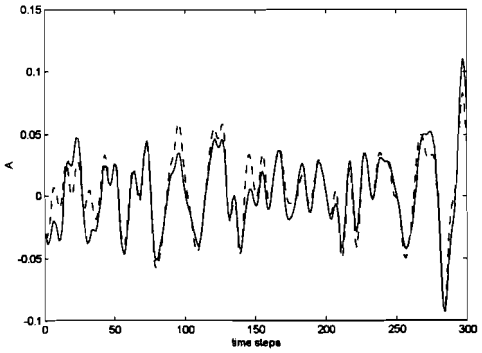

Figure 4.64: *Error of nonlinear and linear (dotted) estimation of output H_3.*

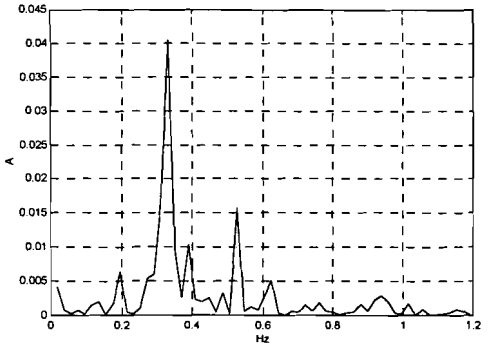

Figure 4.65: *Power spectral density of residue of output H_1.*

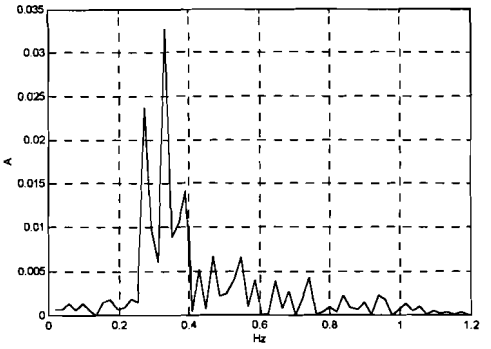

Figure 4.66: *Power spectral density of residue of output H_2.*

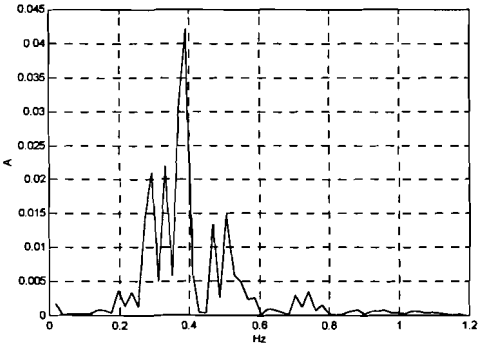

Figure 4.67: *Power spectral density of residue of output H_3.*

Experiment type	Linear model estimation
Goal	Estimating a linear model as a reference for the nonlinear identification.
Data length	1500 samples
Input signals	ZMWN, uniform distributed, cut off freq = 2.5 Hz
Network structure	3 inputs, 1 hidden layer with 18 nodes, 3 white states, 4 black states
Number of active weights	151
Number of SA iterations	20*N
Number of QN iterations	20*N

Table 4.6: *Experiment data for linear estimation.*

The performance of output H_1 has become a bit worse, but the other two outputs have become a bit better. The peaks at 0.38 Hz are still there. Output H_3 now shows a new peak at 0.3 Hz

and at 0.5 Hz. This again can be related to the initial placement in the tub. The validation plots are depicted in figures 4.58-60. Further, the errors of the linear and the nonlinear estimation are plotted in figures 4.61-63. We can hardly see any difference. It seems that the nonstructural wave effects are the main nonlinearity in the system. So we can conclude that we didn't achieve our goal: better estimation performance by use of neural networks.

4.9 Concluding remarks

Estimating a model of the platform dynamics is quite difficult because of the nonstructural wave parts, which can't be modelled this way. The structural part of the waves can be captured in the model, if the model order is chosen wisely. We tried to do this at three different ways: with a nonlinear MIMO-model, with a nonlinear MIMO model existing of combined SISO-models and with a linear MIMO-model. All gave good results.

The fact that the linear MIMO-model showed almost the same performance as the nonlinear equivalent, tells us that the platform dynamics together with the structural wave parts can be considered as almost linear. The nonstructural wave parts must be considered as nonlinear. This is proven by the nonlinear correlation tests. An attempt can be made to capture these nonstructural parts by the nonlinear filter gain, as explained in section 3.6. This will be quite time consuming, because the software is not prepared yet to face the problem. The original software is developed for predicting Gaussian white noise, and here we're dealing with coloured noise. So the filter gain has to be dynamic. This will be described in the next chapter.

4.10 References

[HEU94]: Heuvel C.J.J. van den
Modeling of a floating platform.
TUE, graduation report, 1994.

[REI96]: Reinierkens M.M.H.J.
Identification and control of a floating platform.
TUE, graduation report, 1996.

[LEV93]: Levin A.U. and Narendra K.S.
Control of nonlinear dynamical systems using neural networks: controllability and stabilization.
IEEE Transactions on Neural Networks, Vol. 4, No. 2, pp. 192-206, March 1993.

[LEV93]: Levin A.U. and Narendra K.S.
Control of nonlinear dynamical systems using neural networks- part II : observability, identification and control.
IEEE Transactions on Neural Networks, Vol. 7, No. 1, pp. 30-42, January 1996.

[MAZ96]: Mazak J.
Transition control based on grey, neural states.
TUE, Phd. thesis, 1996.

[DER96]: Derksen J.P.A.M.
IRR modeling of acoustic impulse responses.
TUE, graduation report, 1996.

[BAC94]: Backx A.C.P.M. and Boom A.J.W. van den.
Dictaat van het college toegepaste systeem analyse, deel II.
TUE, college dictate, 1994.

[BIL86]: Billings S.A. and Voon W.S.F.
Correlation based model validity tests for non-linear models.
International Journal of Control, Vol. 44, pp. 235-244, 1986.

[BIL92]: Billings S.A., Jamaluddin H.B. and Chen S.
Properties of neural networks with applications to modelling nonlinear dynamical systems.
International Journal of Control, Vol. 55, No. 1, pp 193-224, 1992.

Chapter 5

Designing a full state observer

5.1 Basics of state estimation.

In the last sections we mentioned that the idea for controlling the process was based on nonlinear state feedback control. The structured state space model as designed and computed in section 4 will be used in a model based controller. This model will provide for a simulation of the true process states $\underline{x}$, which were not available in the real process. As the true process states become subject of some state disturbances w, there will be a difference between the process output y and the simulated output $\hat{y}$. Then reliable full state feedback can't be guaranteed. Therefore, some correction for x has to be made. This will be shown for a linear (-ized) case of the process.

From LQG- theory (see also figure 3.9) we know that :

$$\underline{x}(k+1) = A\underline{x}(k) + B\underline{u}(k) \tag{5.1}$$

and

$$\hat{\underline{x}}(k+1) = A\hat{\underline{x}}(k) + B\underline{u}(k) + L(y - \hat{y}) \tag{5.2}$$

With L the Kalman-gain. By substracting equation 5.2 from equation 5.1 we get an expression for the state disturbance:

$$\tilde{\underline{x}}(k+1) = \underline{x}(k+1) - \hat{\underline{x}}(k+1) = A\tilde{\underline{x}}(k) - L(y - \hat{y}) = (A - LC)\tilde{\underline{x}}(k)$$

Since we are not able to measure the real state error $\tilde{\underline{x}}$, an approximation can be made by taking $Cx=y$. This means that we are considering the output disturbance y, which is a weighted version of the white state disturbances. So, increasing the Kalman gain L leads to faster eigen values of the matrix (A-LC) and the performance of the state disturbance suppression will increase too. In our nonlinear design method, we will replace the Kalman-gain L by a neural network NN_g which will estimate the state disturbance e(k) given all disturbance measurements up to time index k, like depicted in fig. 5.1. Now we can describe the nonlinear equivalents of the equations 5.1 and 5.2:

$$\underline{x}(k+1) = f(\underline{x}(k),u(k))$$

$$\hat{\underline{x}}(k+1) = \hat{f}(\hat{\underline{x}}(k),u(k),\theta) + NN_g(y - h(\hat{\underline{x}}))$$

This last equation can be written as:

$$\hat{\underline{x}}(k+1) = \hat{f}(\underline{x}(k) + \tilde{\underline{x}}(k), u(k), \theta_f) + NN_g(y - h(\underline{x}(k) + \tilde{\underline{x}}(k))$$

The state estimate is denoted by $x(k|k)$. Prediction involves estimation of the state at some future time $k+l$, $l>0$ and the corresponding estimate is then denoted by $\hat{x}(k+l|k)$. In this thesis l is taken equal to one.

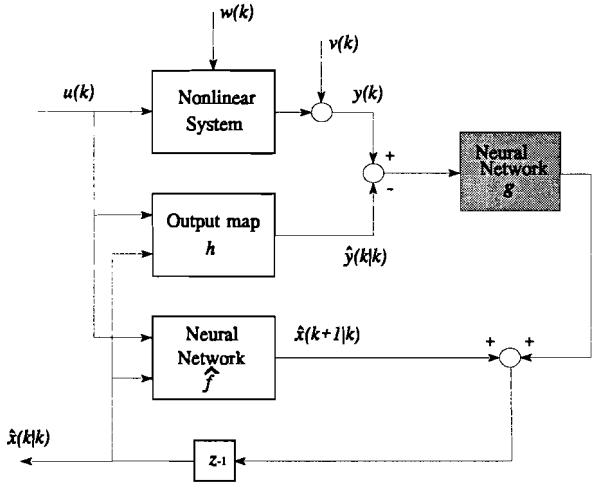

Figure 5.1: *A common single stage state estimator diagram.*

This is often called a single-stage ahead predictor. An optimal single-stage ahead predictor is required to minimize an estimation error criterion

$$J = \text{trace}\, E(\tilde{x}(k+1|k)\tilde{x}(k+1|k)^T)$$

But we have to estimate the state disturbances from the outputs. Then the criterion becomes:

$$J = \text{trace}\, E(h(\tilde{x}(k+1|k))\, h(\tilde{x}(k+1|k))^T) = \text{trace}\, E(\tilde{y}(k+1|k)\tilde{y}(k+1|k)^T)$$

For developing the optimal filter gain, we were involved in the next problem. Linear quadratic regulator theory and the derivated nonlinear control theory used here are based on state disturbances which can be considered as zero mean white noise. In our case the disturbances, caused by static waves, have to be considered as coloured (chaotic) noise. If we want to include these disturbances in our model for prediction, then this makes the proposed structure as depicted in figure 5.1 useless, because the prediction has to be based on past values of the error signal due to the dynamic behaviour of the waves. Therefore we have to add some dynamics to our neural network to predict the influences of the waves. Generally it is known that static waves have a second or higher order behaviour, (pairs of two conjugate badly damped complex poles) and they only occur if they are reflected by a refection angle of 180 degrees. Because we have a square tub these static waves can occur in two opposite directions.

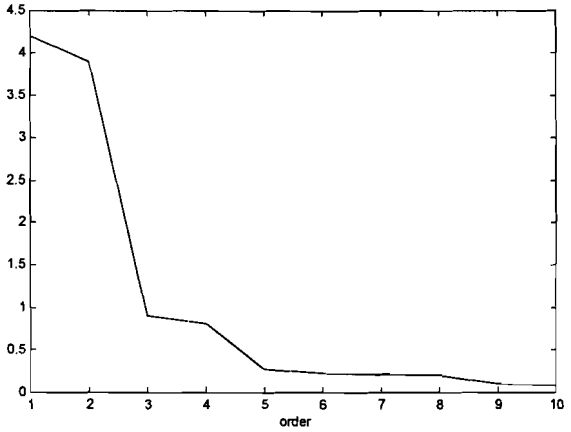

Figure 5.2: *Singular values of the wave disturbances.*

This means we can expect a disturbance order of four or higher. This is tested by a Hankel singular value decomposition of the residu between the simulated impulse response and the real impulse response from X_1 to H_1. The plot shows the expected edges, namely at two and four. Further we can see a small decrease after eight singular values.
For designing an observer that can predict these dynamic state disturbances, we followed the next strategy. At first we took the neural network as described in section 4.7 as a model for our process. Then we enlarged the network by adding a number of nodes to the hidden layer and four extra states x_e for modelling the waves. Now the new weights should help to minimize the output error by adjusting them in the way that the additional nodes and weighted connections in the network take care of the disturbances. The weights of the estimated model will stay fixed during the optimization procedure. The structure is depicted in figure 5.4.

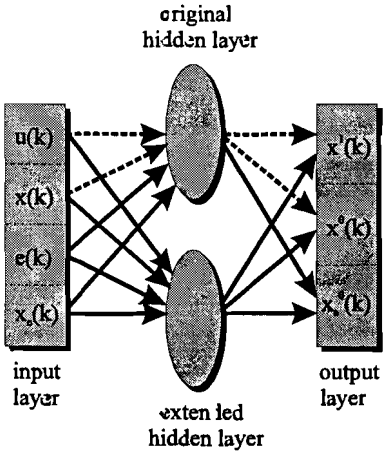

Figure 5.3: *General structure for adding and training additional nodes to the hidden and output layer.*

Where $e(k)=\hat{y}(k)-y(k)$. Now we used also the three residues of the original network for inputs of the extended network. These errors are connected directly back to the network input layer, without any delay. By minimizing the cost function this way it is possible to take the state disturbances, caused by waves into account. For the feedback error $e(k)$ we have two possibilities. At first we can take for $e(k)$ the error between the real outputs and the predicted output of the original model. A second possibility is to take for the error the difference between the real output and the output of the extended network. Now also the feedback error $e(k)$ will be minimized.

The new, more complex network caused that we had to gather a new data set. The number of weights will increase from 151 to 388 using 5 additional nodes in the hidden layer, from which 237 weights have to be optimized starting form small random initial points. This means that the new data set should have a length of at least 2370 data points, maintaining the rule of thumb mentioned in chapter 3, which tells us that the number of data points should be at least ten times the number of the weights. The data acquisition is done like described in section 4.3.
The experiment was 10 minutes, which means a length of 6000 data points. We used 4100 points to estimate the extended model and 1800 points for validation.

For the sofware this meant that it had to be changed drastically. Much effort was needed to do this, but finally it all worked out. Unfortunately we did not succeed in reprogramming the Simulated Annealing procedure in the software. Therefore, instead of this, we used another stochastic optimization routine, which is described in [MAZ96], pp. 40-41 and is called 'controlled random search'. Before we will discuss the results, we will give a short description of chaos.

5.2 Chaos

Chaos is the term used to descibe the apparently complex behaviour of what we consider as simple, well behaving, nonlinear systems. Chaotic behaviour looks erratic and almost random. This behaviour, however, arises in very simple systems (those with only a few degrees of freedom), which are almost free of noise. In fact these systems are rather essentially deterministic. To predict the deterministic future behaviour we need to know the initial conditions of the system exactly. The importance of the system being nonlinear is concerned with the example of a parameter that describes the system. Changing the parameter in a linear system will result in a change in the system's frequency and/or amplitude but the qualitive nature of the system stays the same. Parameter changes in nonlinear systems can lead to sudden changes in the system behaviour. Some of these sudden changes in system behaviour can give rise to complex behaviour called chaos. The term chaos is used to describe the strange time behaviour of a system which is aperiodic and apparently random. One of the most important properties of a chaotic system is the sensitivity of the time evolution of the system to the initial conditions. This makes the future prediction of the system very difficult. The smallest perturbation in initial condition which is very likely to occur in systems with finite accuracy like computer models, can give rise to totally different long term behaviour. This will be shown by one of the most famous chaotic systems, known as the population model. This system describes the rise and fall of a population, for example of animals, in time.

The system is described by:

$$x(k+1) = r\,x(k)(1 - x(k))$$

If the control parameter r is chosen equal or bigger than approximately 3.7, then the sytem becomes chaotic. This is shown in figures 5.4 and 5.5.

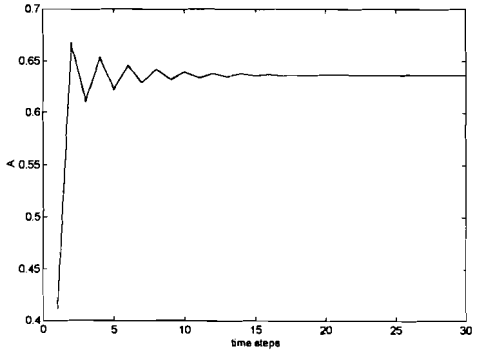

Figure 5.4. *Behaviour of the population model with $r=2.75$, with initial value $x(0)=0.410$ (solid) and $x(0)=0.415$ (dotted).*

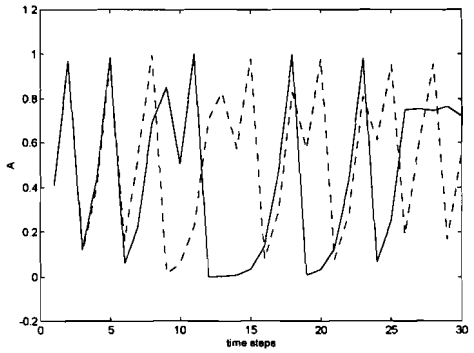

Figure 5.5: *Behaviour of population model with $r=4$ and initial values $x(0)=0.410$ (solid) and $x(0)=0.415$ (dotted). Chaotic behaviour occurs.*

For respectively $r=2.75$ and $r=4$. For the initial condition r is taken to be .41. For $r=2.75$ the population grows to a certain steady endvalue. But for $r=4$ the population behaviour becomes chaotic. At $t=12$ the population has almost died out. In the same figure the dotted line represents the system behaviour after a small change in the initial condition; from $x(0)=0.410$ to $x(0)=0.415$. So this slight change causes a big difference in future behaviour. This makes it very difficult to make accurate predictions for long horizons, for a system with chaotic influences. Nevertheless, a neural network should be able to predict over small horizons ([BIJ96]).

5.3 Results of the extended modeling with fixed output error e(k)

For this procedure the error will be fed back to the inputs and will be the difference between the real output $\hat{y}(k)$ and the original network $y(k)$. This output error will stay fixed, because the weights of the original network will also stay fixed, like mentioned before.
The optimization procedure was started with small random initial points. The number of iterations was about $30N$ for the controlled random search routine followed by $8N$ iterations for the quasi-Newton routine. The optimization time was quite large, about 15 hours for every attempt. The experiment data is shown in table 5.1
At first sight, the results for the extended network showed much better performance than with the original network during validation with the white noise exitations. The performance was really improved in relation with the results of chapter 4.

During the validation of these experiments we observed some peaks on the first predicted values.

These peaks were damped after approximately 30 seconds and occured on both the learning set and testing set.

Experiment type	Extended model estimation
Goal	Estimation with a model extension which should recognize predictive, chaotic behaviour in the residue, with feedback of fixed output error
Data length	4100 samples
Input signals	ZMWN, uniform distributed, cut off frequency = 2.5 Hz
Network structure	6 inputs, 1 hidden layer with 13 nodes, 3 white states, 4 original black states, 4 extended black states
Number of active weights	237
Number of SA iterations	5000
Number of QN iterations	15*N

Table 5.1: *Experiment data for estimation with a model extension.*

An example of this jittering is shown in figure 5.6. The first idea about this was that it could be caused by a higher harmonic due to a nonlinear input/output relation, like the sigmoid activation function in our case. This phenomenon still occured every time after several attempts and it disappeared after short time.

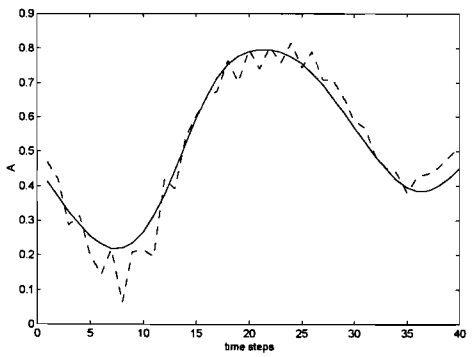

Figure 5.6: *Noise on the prediction.*

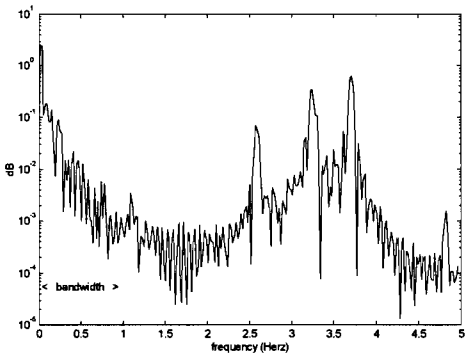

Figure 5.7: *Internal noise power spectrum after exitations with filtered step responses.*

To check the stability of the estimated model, we excited the model with filtered step responses in different areas of the range the outputs reached during estimation. This experiment told us that the system becomes meta stable after more successive steps. In figure 5.7 a power spectrum with the oscillation frequensie(s) are plotted. We should say 'a power spectrum', because other modelling attempts gave slight shifts for the noise peaks. An example of these filtered step responses is plotted in figure 5.8. These frequencies occur already if the process is excited with filtered steps with a cutoff frequency far lower than the noise frequencies. This means the network generates these harmonic oscillations by itself. The oscillations will be fed back to the inputs and

keep themselves alive. This will also occur during the estimation phase. The optimization program considers them as signals which should be modelled. So, for proper modelling, we have to avoid the feedback of these harmonic signals during the optimization phase. At first we tried to filter the states of a 'disturbed' model, but this didn't lead to a proper model. This is caused by the fact that due to the nonlinearity this internal noise cannot be considered as additive. It is generated by the nonlinear model and influences the estimation performance during the optimization in de process bandwidth too. The model simply expects the feedback of these noise signals. In figure 5.9 the result of this filtering is shown.

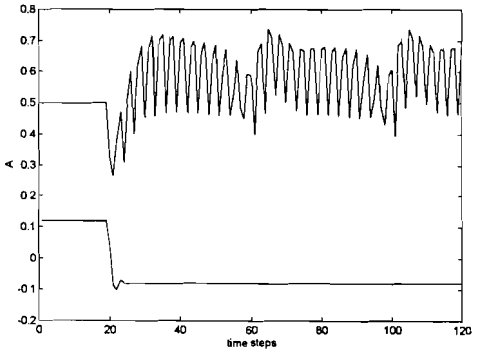

Figure 5.8: *Oscillations, occuring with filtered step inputs.*

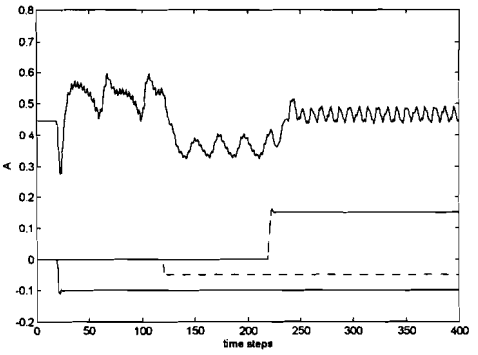

Figure 5.9: *Filtering the states, before feedback. Three steps are generated at the three different inputs*

After the third step, an almost harmonic signal appears, with a main frequency of approximately 1.2 Hz. So, for proper modelling we have to avoid the feedback of this state noise already during the learning phase. Therefore, we have two possibilities:

- Reducing the nonlinearities in the model, and hope that the noise will reduce too.
- Avoiding the feedback of the noise

The first method seems to be most elegant, the second is more practical. We tried them both. We tried the next options:

- Removing some connections between the original and extended network layers, to reduce the nonlinear feedback in the system.

- Estimation of a model with filtered state feedback. The noise will be attenuated before using it as input for the model. Now it will influence the optimization performance less.

- Using only linear network layers for the extended models, to reduce the nonlinearity in the system. Now only the original nonlinear nodes will have to take care for the process nonlinearity.

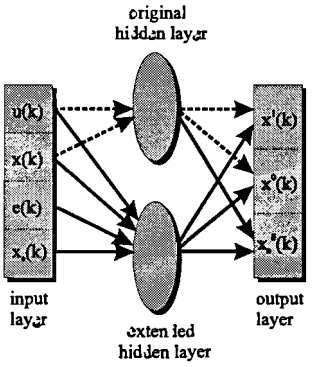

Figure 5.10: *Extended network structure with no influence of extended inputs to the original hidden layer part. Original connections are dotted.*

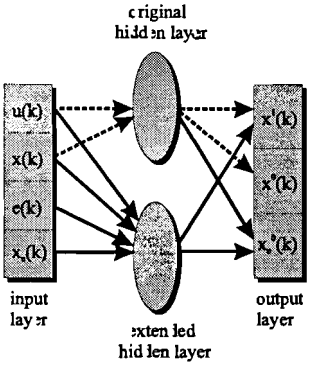

Figure 5.11: *Extended network structure with no direct influence of extended inputs and extended hidden layer to the original layers.*

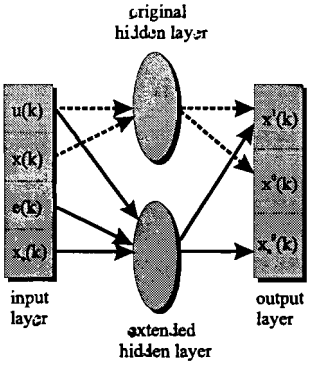

Figure 5.12: *Extended network structure, by which the extension is purely additive.*

Changing the network structure

By changing the network structure we mean: removing the most critical connections between the original network and the extended network, and we can only hope to reduce the internal noise this way.
We start with avoiding the influence of the inputs and states of the extended model (figure 5.10). This means the nonlinearities in the original hidden layer will not be influenced by other signals than those at which the network was trained on. It is possible that from the start values of the optimization, the model is acting in an area which was not encountered (enough) before and due to the fixed biases from the original layer it will not be possible to escape from this area.

After this, we removed the connection between the extended hidden layer and the black states of the original model, so only the outputs will be influenced by the additional nodes (figure 5.11). But again no improvement was shown.

The last connection we could remove were those between the original network inputs and feedback states and the extended hidden layer. Now the extended model can be considered as additive to the original model (figure 5.12). This also did not lead to any improvement

Filtering the states

By lack of succes with the previous method, we tried to estimate the model with filtering the states. So the disturbance will be removed from the feedback signals, before they enter the model again. At this way we will try to avoid the influence of the disturbance with respect to the model parameters.

In section 3.5 we saw that for a dynamic feedback network applies:

$$x(k+1) = f(v(k)) = f[u(k), W(z)x(k)]$$

With $\underline{v}(k)$ the complete input vector, consisting of the input vector $\underline{u}(k)$ and the state feedback vector $\underline{x}(k)$. We can replace $\underline{v}(k)$ by $\underline{x}(k)$, because the input vector $\underline{u}(k)$ is not dependent of the network weights. Taken the first derivative of this equation yields:

$$\frac{\partial x(k)}{\partial \theta} = \Gamma_{\hat{x}}^{\hat{x}} \frac{\partial \hat{x}(k-1)}{\partial \theta} + \frac{\partial \hat{f}(\hat{x}(k-1))}{\partial \theta}$$

This new approach causes that we have to consider the gradient computation algorithm again. Now we have to take the filter in account as it will influence the error backpropagation, which reminds us of the fact that the term on the left is a solution of a difference equation. According to [NAR90], if the first term on the right is small compared with the second term, we can consider the problem as a static optimization problem. But because of the large number of states we are not sure if we can expect this. Therefore we have to use dynamic gradient computation.

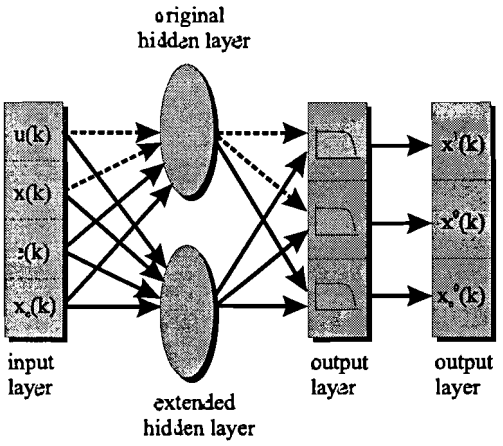

Figure 5.13: *Filtering the states.*

Let $\Gamma_x^f(k)$ be the Jacobian matrix of the neural network f given by:

$$\Gamma_{\hat{x}}^{\hat{f}}(k) = \frac{\partial \hat{f}[\hat{x}(k), u(k), \theta_{\hat{f}}]}{\partial \hat{x}(k)}$$

and $\Gamma_x^h(k)$ be the Jacobian matrix of the fixed output map h:

$$\Gamma_{\hat{x}}^{h}(k) = \frac{\partial h[\hat{x}(k), u(k)]}{\partial \hat{x}(k)}$$

Then the gradient of the error function with respect to the weights in the neural network f is given by:

$$\frac{\partial J(\theta_{\hat{f}})}{\partial \theta_{\hat{f}}} = -\frac{1}{N}\sum_{k=1}^{N} (y(k) - \hat{y}(k))^T \left(\Gamma_{\hat{x}}^{h}(k) \frac{\partial \hat{x}(k)}{\partial \theta_{\hat{f}}} \right)$$

The partial derivative of $x(k)$ with respect to the weight vector has to be computed recursively by:

$$\frac{\partial \hat{x}(k)}{\partial \theta_{\hat{f}}} = \frac{\partial \hat{f}[\hat{x}(k-1), u(k-1), \theta_{\hat{f}}]}{\partial \theta_{\hat{f}}} + \Gamma_{\hat{x}}^{\hat{f}}(k-1) W(z) \frac{\partial \hat{x}(k-1)}{\partial \theta_{\hat{f}}}$$

The first term in the last expression is being evaluated by the backpropagation algorithm, as explained in chapter 3. The same algorithm is used to evaluate the Jacobian matrix $\Gamma_x^f(k-1)$ which is necessary for evaluation of the second term. This formula is a recurrent relation which has to be evaluated recursively in time. This has been implemented in the estimation software.

The filter we will use has to satisfy some requirements. At first it should attenuate the disturbance enough almost without attenuating the training signal. Secondly, almost no fase shift may take place in the process bandwidth. After some research we concluded that an elliptic filter meets our

requirements well. We designed a filter of second order with a cutoff frequency of 1.5 Hz, 0.2 dB of ripple in the passband and 50 dB of ripple on the stopband. This means a transfer function of:

$$H(z) = \frac{0.2320z^2 + 0.4591z + 0.2320}{z^2 - 0.2991z + 0.2438}$$

This filter has a phase shift of 12 degrees at the important frequency around 0.4 Hz. The damping is already 4 dB (=12.5) at 2.5 Hz and 20 dB (= 27) at the main noise peak at 0.37 Hz.

The bodeplots and the pole-zero map of the filter are plotted beneath.

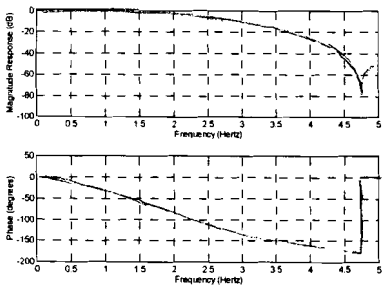

Figure 5.14: *Bode magnitude and phase diagram of elliptic filter.*

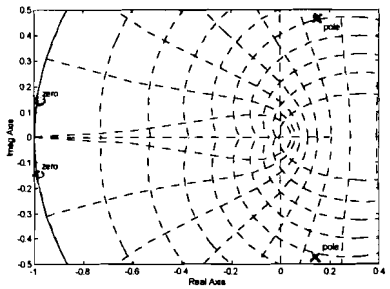

Figure 5.15: *Pole zero map of elliptic filter.*

It is clear that by using this kind of filter we introduce fase shifts in the passbands, which are not marginal. The results for this of experiment, with the experiment data of table 5.4, are plotted in figures 5.16-18.

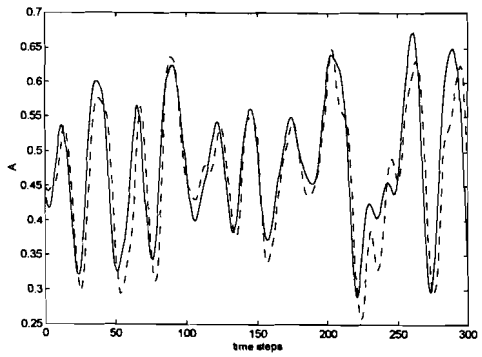

Figure 5.16: *Validation of output H_1, with state filtering during optimization.*

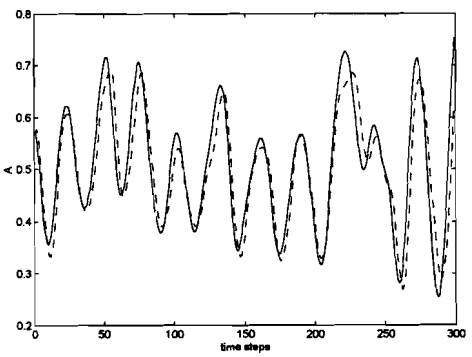

Figure 5.17: *Validation of output H_2, with state filtering during optimization.*

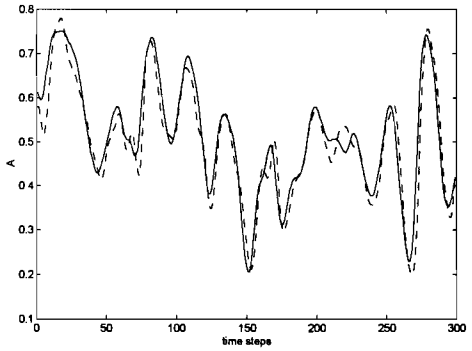

Figure 5.18: *Validation of output H_3, with state filtering during optimization.*

The results are satisfying, but not really outstanding. It is not clear if such a combination of a model and a filter is usefull for internal model control, because at this stage we do not know the influence of the time delay in the passband. But there are some ways to improve this result. At first we can make use of interpolation. By adding more data points in the data set with a zero value, and filter the data set by a special interpolation filter, it is possible to filter more accurately. Due to a lack of time and due to the fact that we knew at this stage the results described in the next section, we haven't done this. However, for experiments of the same kind as described here, with also the same kind of problems, it is recommendable to try this.

Estimation of an extended model with linear nodes in the additional layers.

The last option we tried was estimation with just linear nodes in the network extension. The nonlinearity we need for modelling the chaotic behaviour should be taken into account by the sigmoid activation functions in the original network. This means a hidden layer with 8 sigmoidal nodes and 5 linear nodes.
This time the optimization was successful. The noise still apppeared on the signal, but strongly attenuated. It didn't lead to oscillations anymore. The validation of the outputs are shown in figures 5.22-24. The cost function for the estimation was reduced from 2.0516e-4 towards 2.5516e-5. Now the cost functions for the estimation set and the validation set were almost the same, so the ratio of 1.5 as described in chapter 4 disappeared.
The approximation accuracy for the validation set was reduced from 12.0% to 1.06 %. This means that the extended network recognizes some predictive behaviour in the residues of the outputs. Still at certain moments the prediction is biased, but less than before. It is also visible that output H_2 is fitted better than the other two outputs.
The residues are plotted in figures 5.25-27, the power spectral densities of the residues in figures 5.28-30. It is clear that the performance is increased. Finally, the correlations of the outputs are shown in figures 5.31-33. The residues of the outputs H_1 and H_3 can almost be considered as white noise.

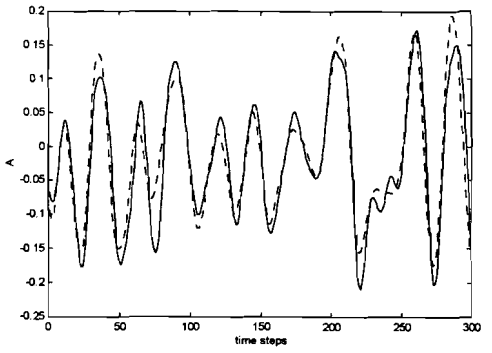

Figure 5.19: *Real (solid) and validated (dotted) output H_1 with original model.*

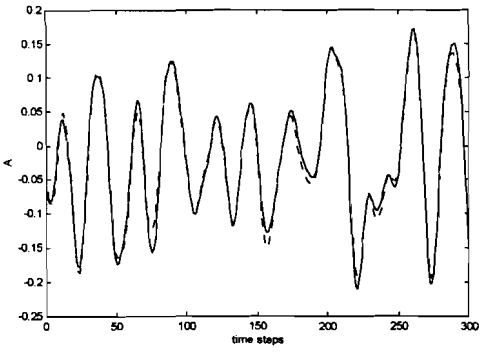

Figure 5.20: *Real (solid) and validated (dotted) output H_1 with extended model.*

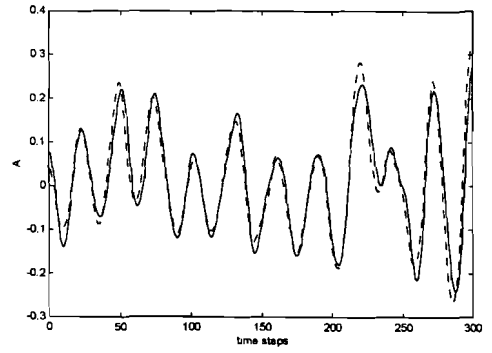

Figure 5.21: *Real (solid) and validated (dotted) output H_2 with original model.*

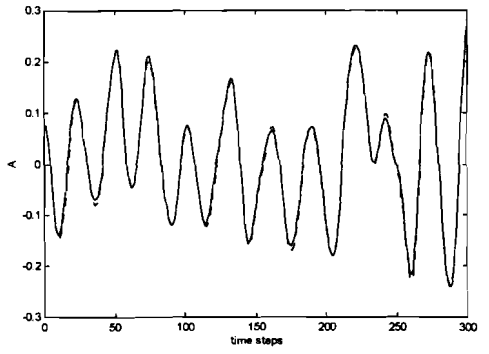

Figure 5.22: *Real (solid) and validated (dotted) output H_2 with extended model.*

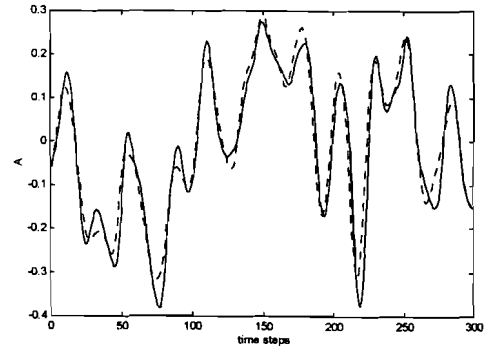

Figure 5.23: *Real (solid) and validated (dotted) output H_3 with original model.*

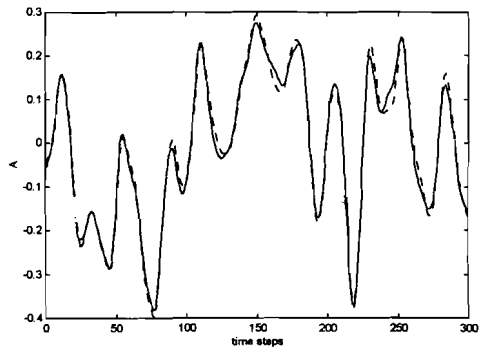

Figure 5.24: *Real (solid) and validated (dotted) output H_3 with extended model.*

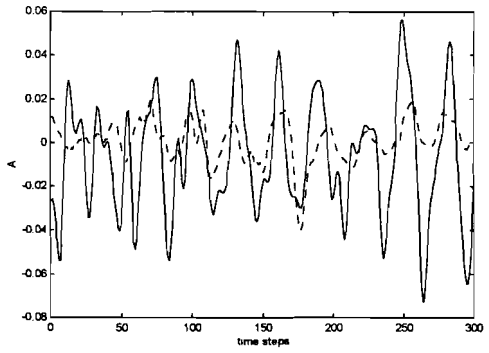

Figure 5.25: *Validation error output H_1 for original (solid) and extended model.*

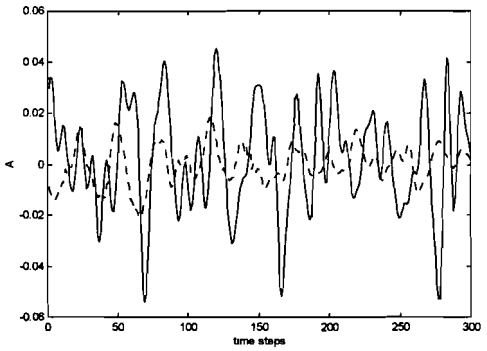

Figure 5.26: *Validation error output H_2 for original (solid) and extended model.*

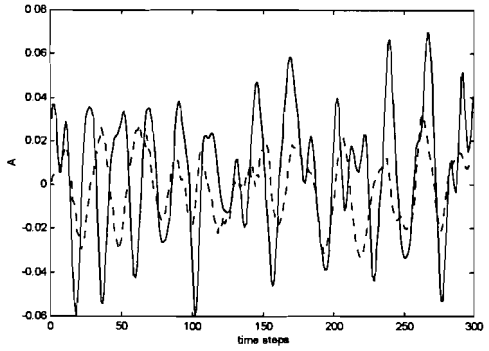

Figure 5.27: *Validation error output H_3 for original (solid) and extended model.*

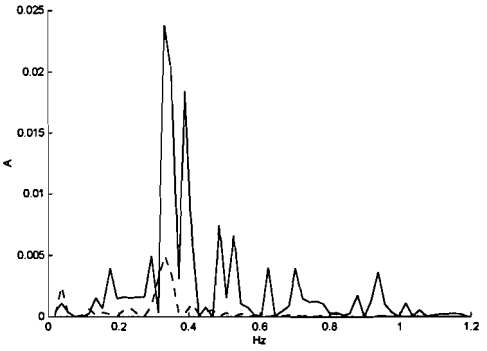

Figure 5.28: *Power spectral density residue H_1 for original (solid) and extended model.*

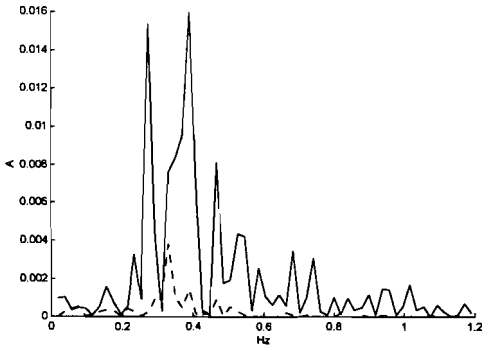

Figure 5.29: *Power spectral density residue H_2 for original (solid) and extended model.*

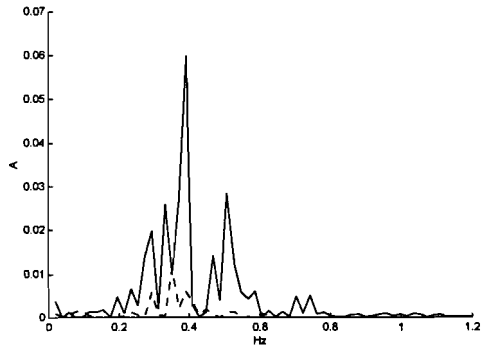

Figure 5.30: *Power spectral density residue H_2 for original (solid) and extended model.*

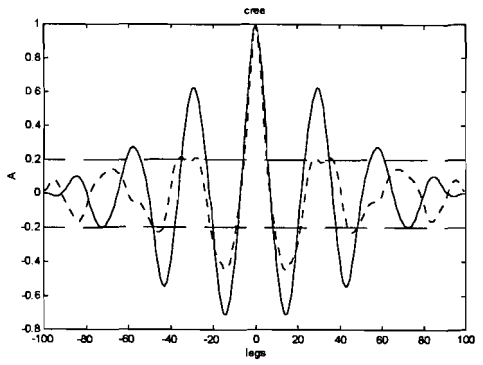

Figure 5.31: *Auto correlation residue of H_1 for original (solid) and extended model.*

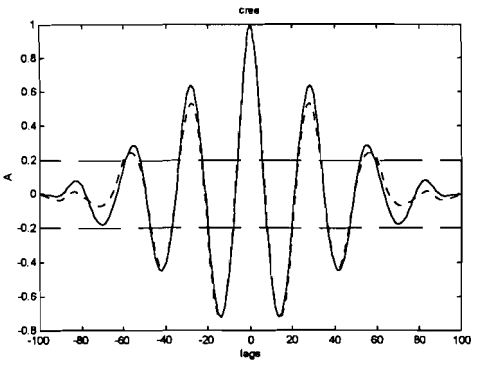

Figure 5.32: *Auto correlation residue of H_2 for original (solid) and extended model.*

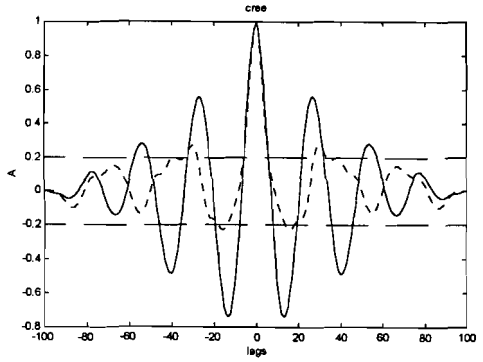

Figure 5.33: *Auto correlation residue of H_3 for original (solid) and extended model.*

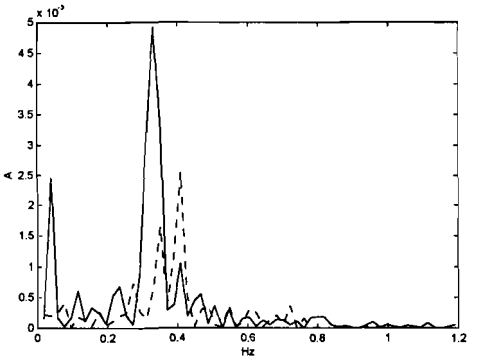

Figure 5.34: *Power spectral density of residue H_1 of extended model. Validation with filtering the input signal (dotted) and without.*

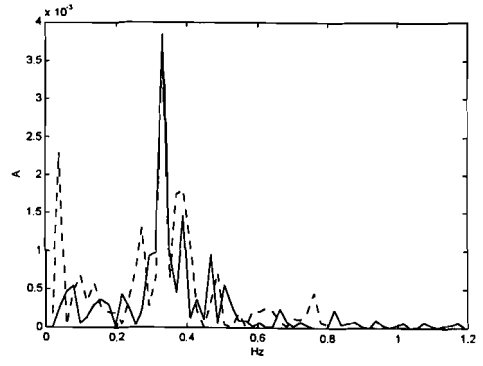

Figure 5.35: *Power spectral density of residue output H_2 of extended model. Validation with filtering the input signal (dotted) and without.*

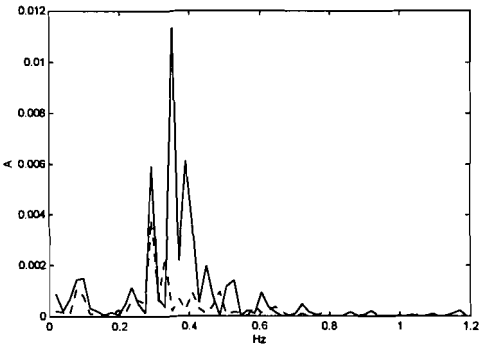

Figure 5.36: *Power spectral density of residue output H_3 of extended model. Validation with filtering the input signal (dotted) and without.*

The tests with the filtered step inputs are shown in figure 5.37.

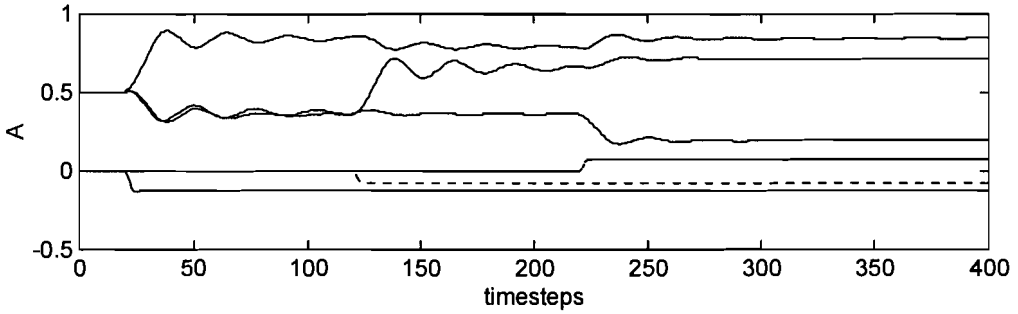

Figure 5.37: *Reaction of outputs to multiple, filtered stepresponses.*

During the experiments done so far it became clear that not the entire frequency peak at approximately 0.38 Hz can be removed this way. An idea to decrease this peak further is to put more weight on it. This can easily be done by filtering the input signals in the data set. We did this by use of a fourth order Butterworth bandpass filter with a pass area from 0.25 Hz to 0.45 Hz. The vector was chosen random with the elements in the linear area of the sigmoid function, estimation with the final weights of the normal optimization caused the network to become oscillating again.

In figure 5.34-36 the power spectral density plots of the residues are plotted for the optimization with and without filtering. It is clear that the model performance is improved just a bit this way, now the total error is decreased from 1.06 % to .80 %.

5.4 Results of modelling with the extended network and variable output error e(k)

The goal of this experiment was feed back of the output error between the real outputs and the prediction of the extended network, and to try to minimize this error. The results obtained were not really satisfying. This was caused by an improper gradient calculation algorithm. We didn't succeed to improve this during our thesis work.

5.5 References

[END87] Enden, A.W.M. van den and N.A.M. Verhoeckx
Digitale signaalbewerking
Overberg, The Netherlands, Delta Press B.V. 1987, 379 p., First print, ISBN 90 6674 722 6

[KRA93] Krauss T.P., L. Shure and J.N. Little
Signal Processing Toolbox
The Matworks, Inc.

Chapter 6

Neural state control

6.1 Basics of state control

In LQG-theory, the states of the system are fed back with a linear matrix K to the inputs (see fig. 3.9-10). The fact that not all these true states are measurable in our situation, we estimated an observor in the previous sections. Now we can use these simulated states to predict the control actions. From LQG-theory we know that:

$$x(k+1) = Ax(k) + B(u(k) - K\hat{x}(k))$$

For the input *u(k)* (fig. 3.9) we can say:

$$u(k) = r(k) - K\hat{x}(k)$$

So, if we consider the setpoint vector *r(k)* containing just zeros, e.g. the setpoints are zero, the inputs of the process will be the process states multiplied by feedback gains, $u(k) = -Kx(k)$. The optimal choice for such a linear feedback matrix can be derived by solving the Ricatti equations. For nonlinear plants we have to follow another traject:

Let a sampled version of the controlled nonlinear dynamic system Σ_s be given by:

$$x(k+1] = f[x(k), u(k), w(k)]$$

$$y(k) = h[x^1(k), u(k)] + v(k)$$

Let the control system be described by the following nonlinear state-space dynamic system:

$$z(k+1) = a[z(k), x(k), r(k)]$$

$$u(k) = b[z(k), x(k), r(k)]$$

where *a* and *b* are smooth, a priori unknown nonlinear functions, *z(k)* is the state vector of the controller, *r(k)* is a reference signal and the controller output *u(k)* becomes the system input.

It includes the idea of adding integrators into the state feedback. In this case the function *a* could be defined as:

$$a[z(k),x(k),r(k)] = z(k)+(x(k)-r(k))$$

The structure is depicted in figure 6.1.

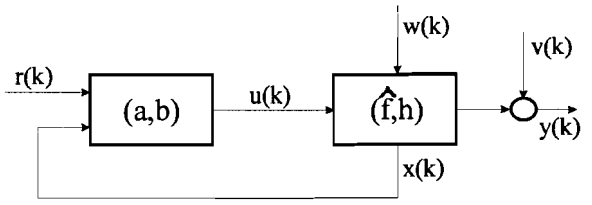

Figure 6.1: *Nonlinear state feedback.*

In the design of the controller we will assume that:

- The state of the process is being predicted by an observer.
- No exact model is available.
- The process is subject to process disturbances and measurement noise

The state-space observer has been built up of an estimated simulation model $\hat{f}$ and a model extension to predict wave disturbances, approximated by a neural network $\hat{g}$, like described in chapter 4 and 5. That is:

$$\hat{x}(k+1) = \hat{f}[\hat{x}(k+1),u(k),\theta_{\hat{f}}]+\hat{g}[y(k),\hat{y}(k),\theta_{\hat{g}}]$$

$$\hat{y}(k) = h[\hat{x}^1(k),u(k)]$$

Like in LQG-theory for controller estimation we have to define the weight matrices Q and R. In the semi-positive matrix Q the weights are for defining the priority of tracking the process outputs, in the positive definite matrix R we can weight the controller outputs. The ratio between Q and R defines the importance ratio between the tracking performance and the controller output. These requirements can be translated in the next equation:

$$J(t) = \sum_{k=t_0}^{t_f} \left(\|\hat{x}(k)-r(k)\|_Q^2 + \|u(k)\|_R^2 \right)$$

where the norms are weighted Euclidian vector norms, and the discrete time interval $\langle t_0,t_f \rangle$. If we want to make use of integral actions to get final tracking errors we have to consider the next criterion:

$$J(t) = \sum_{k=t_0}^{t_f} \left(\|\hat{x}(k)-r(k)\|_Q^2 + \|\Delta u(k)\|_R^2 \right)$$

where $\Delta u(k)=u(k)-u(k-1)$ and $\Delta u(t_0)=0$. The state vector and control vector constraints are most of the time considered as side constraints, constraining operating ranges of these variables. These

constraints are given by:

$$\underline{x} \leq x(k) \leq \bar{x}$$

$$\underline{u} \leq u(k) \leq \bar{u}$$

A consideration of the last two constraints is not only important for a proper process behaviour, but we have to have in our mind that the process model is nonlinear and is only valid within certain limited bounds. Therefore, these constraints can never be omitted.

Due to the nonlinear estimation our model is not bounded to one specific point at which the model is linearized. So we can cover a specific range, in which the model is valid. Generally this means that we can define our controller such that he can bring the process to another working point. This point has to be stable one. Therefore the controller has to follow some specific state trajectories. In our case, we are just interested in horizontal stabilization of the floating platform at one specific point, so the controller doesn't have to be trained at certain state trajectories. The only inputs we use for training the controller will be the wave disturbances, eventually caused by some crane hoist/rotation actions. State references have to be defined for state tracking.

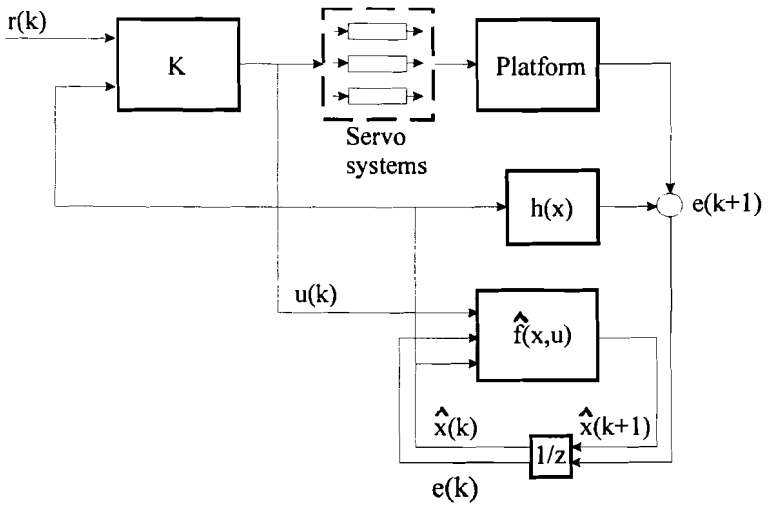

Figure 6.2: *General structure for neural floating platform control. Now the neural network f contains the original estimated network and the model extension as well.*

Also for the control actions references have to be defined. The adjustment for the Q and R matrices in our case is simplified because the the elements in both matrices have to be of the same value, the outputs are all even important. This is also true for the inputs. So just the ratio between Q and R will be important. With proper adjustment of these matrices a controller can be estimated.

We haven't done this anymore due to a lack of time. Just the software has been programmed for our specific case.

Chapter 7

Conclusions and recommendations

7.1 Conclusions

Estimation of a proper model of the dynamics of the floating platform led us to some problems. At first there was the problem with the sensor system. The copper strips used for the position measurement of the platform are not reliable. Some sweeping effect was discovered due to a lack of stiffness. Instead of these, pressure sensors can be used. These show a very accurate behaviour, both in static and dynamic situation. During the estimation phase, the reliability of the wave height sensors was discussed. The estimation software couldn't find a simulation behaviour, this was probably caused by severe disturbances due to electronic malfunctioning. These sensors should be replaced by another, more reliable sensor system with less time varying behaviour. For future experiments it is also recommendable to update the electronic circuits. Drift problems, probably caused by inferior circuit parts caused that much time was lost unnecessarily. After every drift, the measurements results became useless and the very time consuming work of calibration could be started over again.

Generally, the application of neural networks in the area of system identification has been proved to be successful. They are powerful in capturing and approximation of (nonlinear) system dynamics. Main drawback is the computational effort involved. Compared with linear estimation algorithms the growth is considerable. Another drawback is the difficulty of finding the global minimum or one that this approaches. The relation between the growth of number of nodes and growth of time needed to find such a mimimum is NP-difficult and therefore small network sizes are advantageous.

Considering the mean goal of this thesis, we can make the next conclusions.

The model estimation has been done at three different ways:

- A nonlinear Multiple Input, Multiple Output model.
- A combination of three Single Input, Single Output models
- A linear Multiple Input, Multiple Output model

A nonlinear MIMO model structure consisting of one hidden layer with eight nodes, and seven

states turned out to be the optimal choice. However, the differences in performance compared with the other methods were small. Model estimation done this way leads to a biased model. More attempts for decreasing the residue won't be successful because there is (almost) no relation between the inputs and the information left in the residue. This is caused by the fact that the residue contains the behaviour of the static waves in the tub. These waves look stochastic, but are in fact chaotic and hard to include in the model this way. According to previous work done at the platform, it is strongly advisable to predict these waves by a nonlinear filter. This makes a better (quicker) controller possible, because now the periodic movement of the static wave at the position of a float can be predicted for short horizons and doesn't have to lead to instability by quick servo actions.

By enlarging the network complexity with the original nodes staying fixed, this problem can be solved. Such an extension of an already existing neural network with extra nodes in the hidden and output layer led in our case to optimization problems. Higher harmonic signals were generated by the network and by feedback they were used to train it. So signals occur in the state space of the model, which were not defined and expected, and keep themselves alive. This influences also the process behaviour in the lower frequency part. For removing these signals the proposed methods mentioned in chapter 5 are advisable. Because no other example of this approach was found in literature, we can't give a cause for this phenomenon. Therefore further study on this problem is recommendable, because it can give an extra dimension to neural network estimation with predictable noise.

7.2 Recommendations

- Review the electronic circuits of the sensor systems, to guarantee no time varying properties for a long experiment period.

- A recommendation which will probably not be brought in practice, is the placement of the platform in a larger tub. Mounting the rather heavy crane on the platform caused that we had to enlarge the floats. In this situation the floats seem too big for white noise excitations; the floats are touching the tub borders and cause discontinuities in the data set, which can not be used anymore. Many attempts have to be made to gather a data set without this effect, by adapting the white noise magnitudes.

- In general: take longer data sets for nonlinear estimation to be sure that all areas in the nonlinear state space will be reached. Advisable would be to take the number of samples 30 or 40 times the number of parameters in the network.

- Add more dynamic/static a priori knowledge to the neural network, to let it become less black. Much of this work is described in [HEU94] and [MOR95]. It is possible by reprogramming the software to include these.

- Combine the estimation of a linear model as described in section 4.7 with the estimation of a nonlinear model extension as described in chapter 5. This way we'll give the non linearity properties direct to the model part where they are needed at most. Now we don't have to answer the question: does the addition of more connections to a node with already fixed

weights and bias improve or decrease the model performance by changing its nonlinear aspects.

- If a proper controller for the stabilization of the floating platform is estimated, it would be useful to take the properties of an additional (neural) feed forward controller into account. This controller can be used to compensate the disturbances caused by hoist actions or swinging of the jib of the crane. Therefore as inputs we can take the jib position, the position of the car on the jib and the mass which is hoisted. The first two can be measured on the platform interface, for measuring the mass a simple electronic circuit with some kind of load cell would be enough. This could be a rather complex neural controller, but improvement will already be there if just a static, roughly tuned controller will be implemented.

- A last recommendation concerns the software used for training the neural networks. This has been written in the C programming language and is very difficult to understand because no documentation is available. Sometimes it takes much time to understand the tricks used by the programmer. This is, of course, avoidable by documenting the code.

Appendix A: Algorithm for Simulated Annealing.

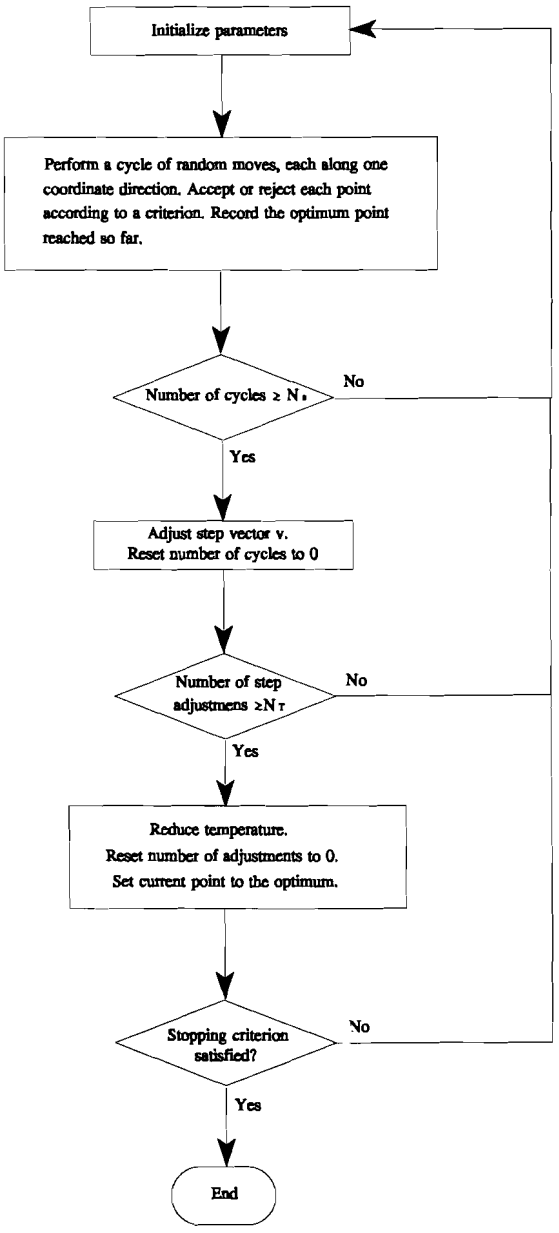

Figure A1: *Algorithm for simulated annealing.*

Appendix B: M-file for transformation of the weight vectors.

This file converts the weight file format, used by the VAX C software to the format used in Simulink.

```
% M-file "Genvec.m" for conversion of weight vector to Matlab format

clear;
% Extract weights from vax weight file id_w_*_*.dat
% for simulink model neuralsim
% clear work environment

f=fopen('g:\result\id_w_chb.dat','rb','vaxg'); % open weight vector file
gv=fread(f,'double');
fclose(f);

% load validation data
load g:\soft\totchaes.mat -ascii ;              % open data set
T=(.1:.1:120)';                                % define time vector

U=totchaes(2801:4000,1:6);                     % define inputs
Y=totchaes(2801:4000,7:9);                     %define outputs
IC=[.5 .5 .5 .5 .5 .5 .5 .5 .5 .5 .5];         % define initial values
ni=6;                                          % number of inputs
nh=13;                                         % number of nodes in hidden layer
nw=3;                                          % number of white states
nb=8;                                          % number of black states
no=nw+nb;                                      % total number of st ıtes
Ts=.1;                                         % define sample tim›


for i=1:nh,B1(i)=gv(1+(i-1)*(no+ni+1));        % fill bias vector B1
end

for i=1:nh                                     % fill weight vector W1
        for j=1:no+ni, w1((i-1)*(ni+no)+j)=gv(1+(i-1)*(no+ni+1)+j);
        end
```

```
end
for i=1:nh
        for j=1:no+ni, W1(i,j)=w1(j+(i-1)*(no+ni));
        end
end;

for i=1:nw, B2(i)=gv(nh*(no+ni+1)+1+(i-1)*(nh+1));
end

for i=1:nw
       for j=1:nh, w2((i-1)*(nh)+j)=gv(nh*(no+ni+1)+1+(i-1)*(nh+1)+j);
        end
end;
for i=1:nw
        for j=1:nh, W2(i,j)=w2(j+(i-1)*nh);
        end
end;


for i=1:nb,B3(i)=gv(nh*(no+ni+1)+nw*(nh+1)+1+(i-1)*(nh+1));
end

for i=1:nb
        for j=1:nh, w3((i-1)*nh+j)=gv(nh*(no+ni+1)+nw*(nh+1)+1+(i-1)*(nh+1)+j);
        end
end;
for i=1:nb
        for j=1:nh, W3(i,j)=w3(j+(i-1)*nh);
        end
end;

return;                 % return values for Simulink
end.
```

Appendix C: Further correlations from section 4.5

In this appendix the linear correlations for output H_2 and H_3 are shown.

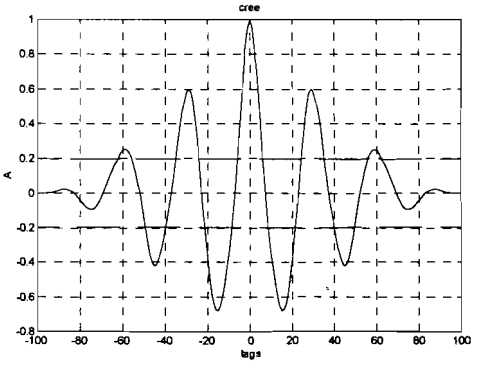

Figure B1: *Auto correlation output H_2.*

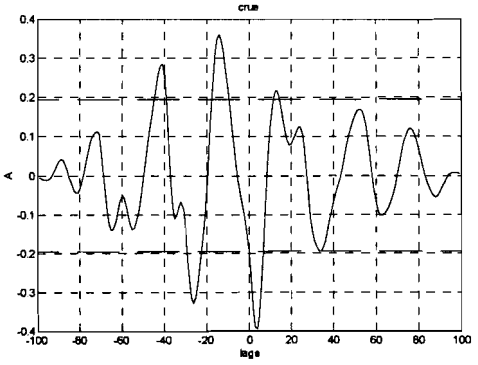

Figure B2: *Cross correlation between input X_1 and output H_2.*

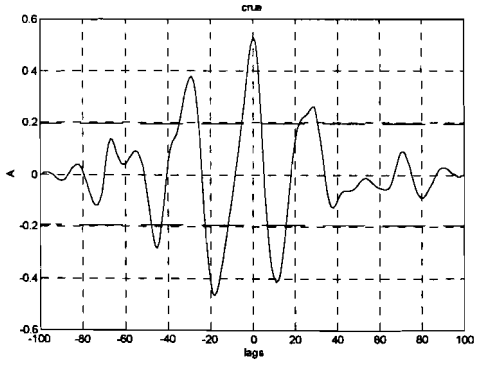

Figure B3: *Cross correlation between input X_2 and output H_2.*

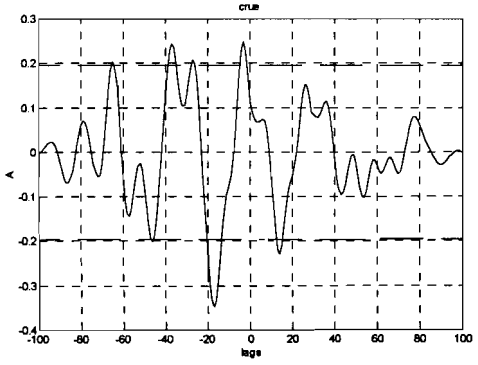

Figure B4: *Cross correlation between input X_3 and output H_2.*

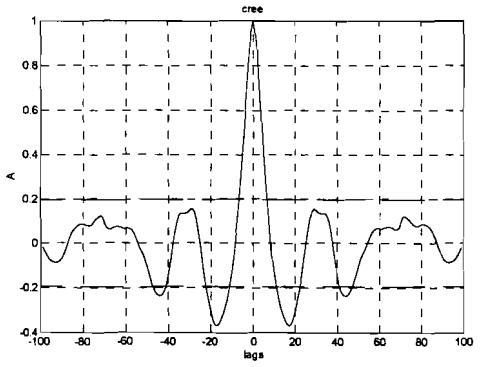

Figure B5: *Auto correlation output H_3.*

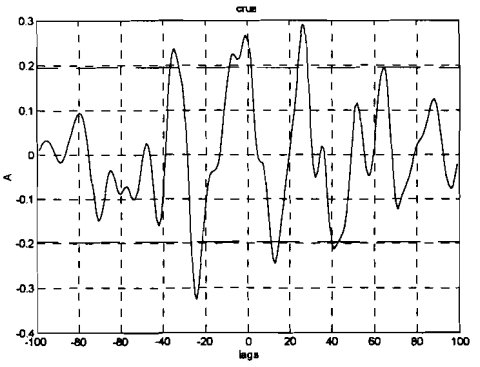

Figure B6: *Cross correlation between input X_1 and output H_2.*

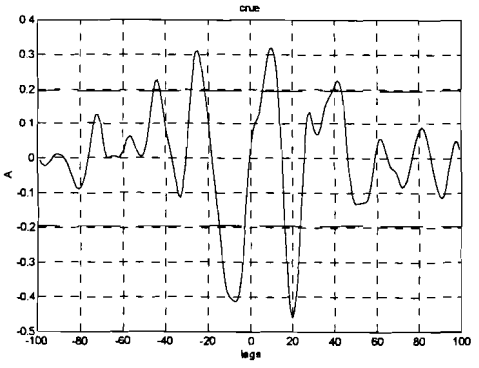

Figure B7: *Cross correlation between input X_2 and output H_3.*

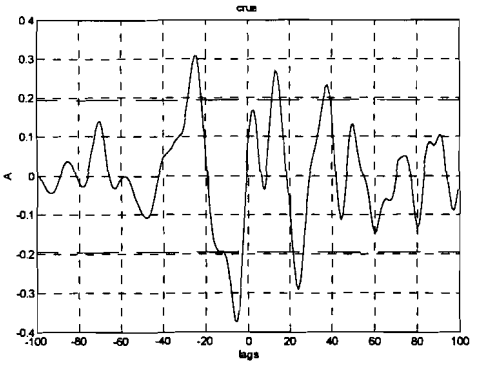

Figure B8: *Cross correlation between input X_3 and output H_2.*

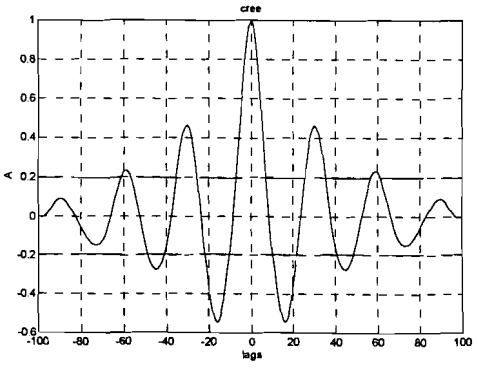

Figure B9: *Auto correlation of residue of output θ_y.*

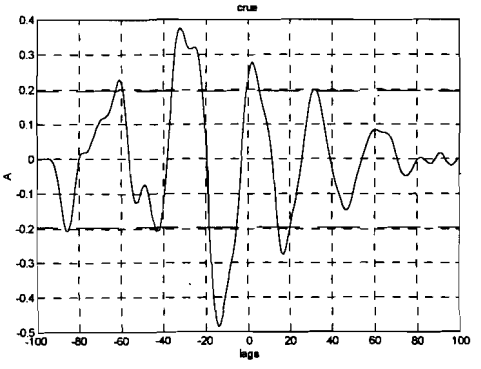

Figure B10: *Cross correlation between input X_a and output θ_y*

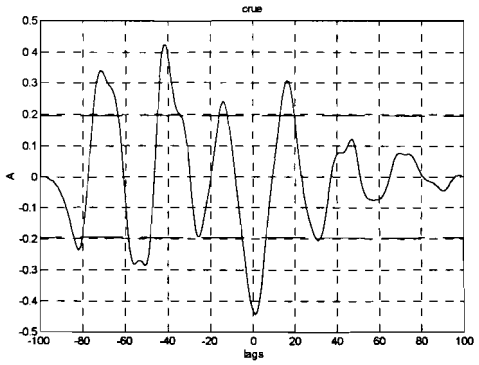

Figure B11: *Cross correlation between input X_y and output θ_y*

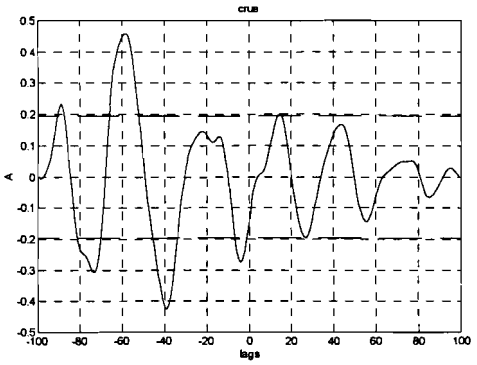

Figure B12: *Cross correlation between input X_x and output θ_y*

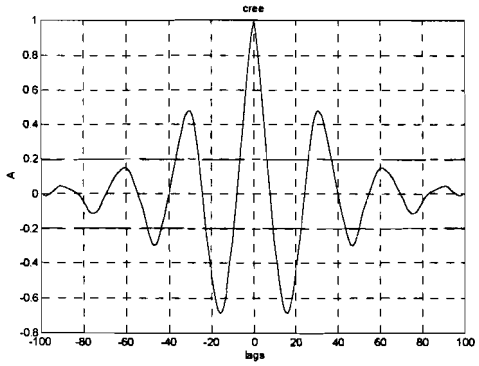

Figure B13: *Auto correlation of residue of output θ_x.*

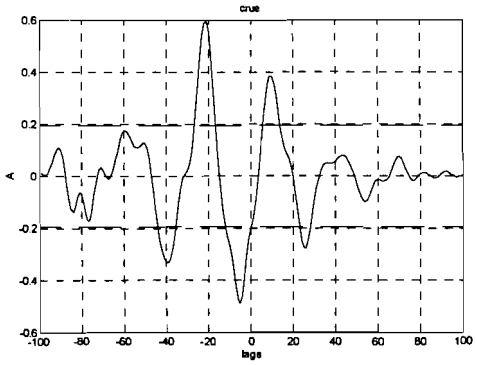

Figure B14: *Cross correlation between input X_a and output θ_x*

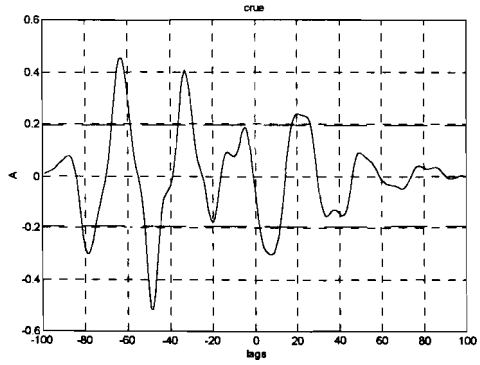

Figure B15: *Cross correlation between input X_y and output θ_x*

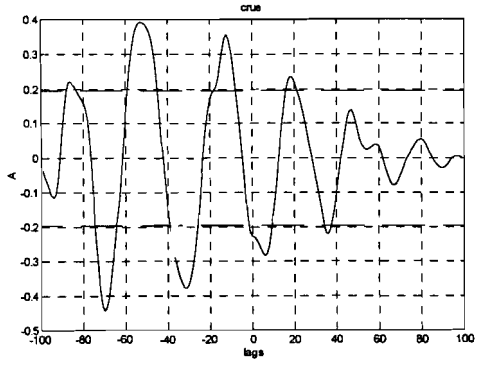

Figure B16: *Cross correlation between input X_x and output θ_x*