Nonlinear system identification of a gantry crane process using neural networks

van den Beemt, B.J.M.

Award date:
1992

6011

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

NONLINEAR SYSTEM IDENTIFICATION OF A GANTRY CRANE PROCESS USING NEURAL NETWORKS

By B.J.M. van den Beemt

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

I would like to thank the following people for their support:

Udo Bartzke
Remi Zorge
Wim Beckers
Ad Damen
Siep Weiland
Heinz Falkus

Beemt, B.J.M. van den. Nonlinear system identification of a gantry crane process using neural networks

M.Sc. Thesis, Measurement and Control Section ER, Electrical Engineering, Eindhoven University of Technology, The Netherlands, December 1992.

ABSTRACT

The process under study is a pilot model of a gantry crane. In this process a trolley can move along a rail, driven by a servosystem. A pendulum has been attached to the trolley. This SISO process can be controlled with a Measurement And Control System (MACS), installed on a computer. A multilayered neural network has been trained with the static backpropagation learn algorithm to model the dynamics of this practical system.

There is an integration in the position signal of the trolley, because of the speed controlled servosystem. It turns out that it is difficult to train a neural network in open loop. By taking the velocity of the trolley as output, we skip the integration, and we derive a neural network model with a better output error performance than a linear model.

The gantry crane process can be extended to a MIMO process by replacing the pendulum by an hoisting system. For this system a mechanical nonlinear model has been derived and simulated. On the basis of a linearized model an LQG-controller has been designed to stabilize the process.

A multilayered neural network (MIMO) has been trained with a static backpropagation algorithm to identify the simulated process. The output error performance of this model is very poor, but can be explained.

CONTENTS

1. INTRODUCTION

Identification is the problem of finding a good model of a process with the same input-output behaviour. This model is obtained by some method, based on input-output datasets, with some criterion function. The datasets are obtained from measurements and experiments. The identified model will be used to simulate the process and to design a controller off-line.

Linear models can only be used under limited conditions and special assumptions since most physical processes usually behave nonlinearly. If the nonlinearities are small, a nonlinear process can be linearized around its working point and a linear model can be estimated to design the controller. In many cases however, a nonlinear dynamic model is needed.

Neural networks have recently been used as an alternative for classical nonlinear models. With neural networks we can approximate any nonlinear function, provided that this function has certain properties such as continuity and finiteness. Using neural networks in system identification, is quite a different situation, because systems are dynamic processes instead of static functions. In system identification, the neural network is fed with past inputs and outputs of the real system to obtain a dynamic model.

This year the research on neural networks in system identification has been started in the Measurement and Control Section of the Eindhoven University of Technology. In this thesis, neural networks are used to identify a practical nonlinear system. It is clear that a practical system is more difficult to identify, because both input and output noise are included, the system is not ideal and assumptions do not have to be true. The system under study is a simplified pilot model of a gantry crane, with fixed cable length. This process is referred to as a single input, single output (SISO) system. The multiple input, multiple output (MIMO) system, with hoisting facility, is simulated and identified by training a neural network.

In section 2 a mechanical description of the MIMO system is given, with as special case the SISO system. In section 3 neural networks used to identify this system is discussed. In section 4 a linear identification method following Y.C.Zhu is described. In section 5 an experiment is designed for the practical SISO system and both linear and neural identification results are presented. Section 6 gives the results of the MIMO simulated process, which is stabilized using an LQG-controller. This simulated MIMO process is identified with the use of a neural network model. In section 7 some conclusions and recommendations are given.

2. THE GANTRY CRANE PROCESS

In this chapter a description of the nonlinear system under study, the gantry crane process, is given. Attention is paid to the plant description, the mechanical equations and the sensors and actuators. A distinction is made between the system with hoisting (MIMO case) and the system without hoisting (SISO case).

2.1. Plant description

First a description of the MIMO gantry crane process is given (section 2.1.1) then a simplification will be made to obtain the actual SISO system (section 2.1.2).

2.1.1. MIMO

The gantry crane is used to move large parts and assemblies from one location to others on a factory floor. A cable is attached to the load to be moved which is then hoisted several feet in the air. See figure 2.1.

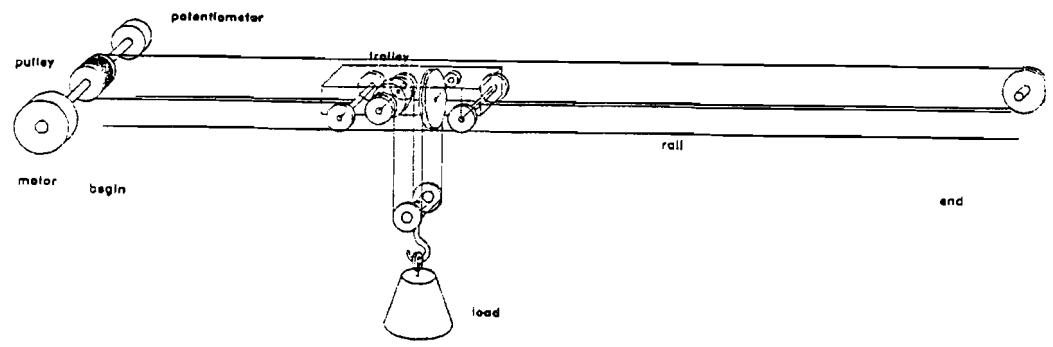


Fig.2.1. Gantry crane with hoisting

The control problem to be considered here is to transfer the load from its current position to a target position, without any load swing. The control system is responsible for controlling the horizontal motion of the crane and the vertical motion of the load such that:

- The load is moved to a new site specified by given co-ordinates.
- The motion of the load is well damped.
- The closed loop controlled system can cope with variable load mass and variable cable length.

This system is a multiple input, multiple output (MIMO) system. In figure 2.2 the block scheme of the controlled MIMO system with its inputs and outputs is given.

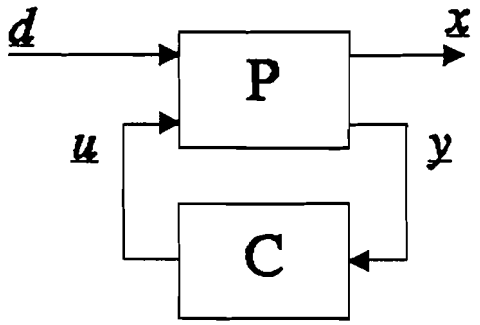


Fig.2.2. Controlled system

Block "P" denotes the plant and block "C" denotes the controller. The two inputs $\underline{u} \in \mathbf{R}^2$ of the system are the input voltages of the servosystems, proportional to the torques of the DC-motor drives on the crane and on the trolley. The output vector $\underline{x} \in \mathbf{R}^2$ of the system represents the co-ordinates of the load. The co-ordinates of the load are dependent on the measured variables denoted by the output vector $\underline{y} \in \mathbf{R}^3$ which is fed back into the controller. There are three measured variables, namely the position of the trolley x_t, the length of the grab cable L and the angle θ caused by swinging of the load. The vector d on the input of the process denotes the disturbance vector. The load mass is supposed to be constant at first instance.

2.1.2. SISO

The hoisting cable is replaced by a pendulum of fixed length. We refer to this system as the single input, single output (SISO) process. A scaled pilot model of the SISO gantry crane process is available in our group. The block scheme of the whole plant is given in figure 2.3.

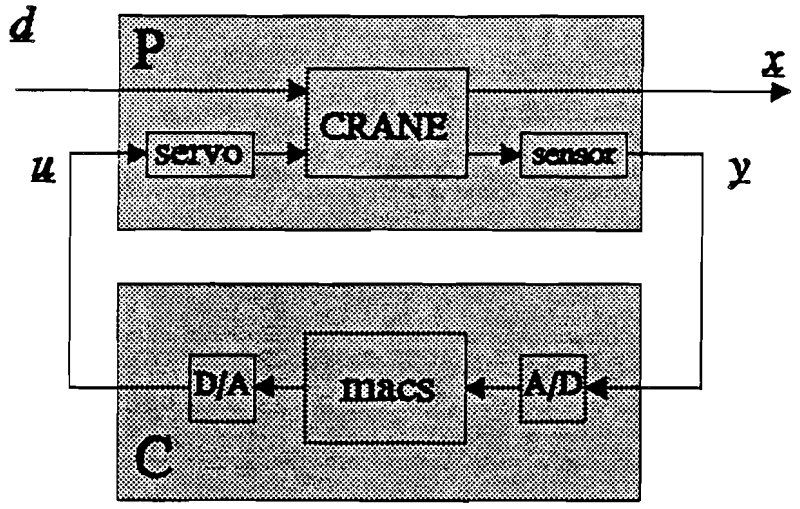


Fig.2.3. Block scheme of plant

The input signal $u \in [-1,1]$ of the system is obtained from the Digital Analog Converter (DAC) of the computer. This signal is proportional to the driving force on the trolley. The output x of the system is the x-position of the load. Like in the MIMO case, this signal depends on the measured variables, denoted by the vector $\underline{y}$. The measured variables are the position of the trolley $x_t \in [0,170]$ and the angle deviation from zero of the pendulum $\theta \in [-.5\pi,.5\pi]$. These measured signals are recorded by an Analog Digital Converter (ADC) of the computer. Some of the disturbances $\underline{d}$ working on the SISO system are mentioned below:

- The vibration of the drag cord.
- The mechanical trembling of the system (construction disturbances).
- The movement of the pendulum in lateral direction.
- The tilting of the trolley in lateral direction.
- The accuracy of the measured signals.

2.2. Mechanical equations

In this section a mechanical description of the MIMO system is given. The nonlinear equations of motion are used to simulate the MIMO system (see chapter 6). It is shown, that these equations reduce to SISO equations when the hoisting terms are set to zero.

Some remarks can be made on the derived equations:

1. The friction force on the trolley is modelled as a damping term of the trolley, with a constant damping factor d_t. This means that the nonlinear behaviour of the friction force in the trajectory of stopping and starting is not taken into account. Also the change of the friction when the speed becomes very high, is not taken into account.

2. The damping of the load-swing is considered to take place in the bearing. This damping is considered to be a small, positive, constant value. It works opposite to the acceleration of the load and it is important for identification purposes (otherwise there are two complex poles on the unit circle in the discrete time domain, which is very hard to identify).

3. In the MIMO system, the length L will be measured with the help of the amount of fetched-in cable. The real length L is dependent on the transfer ratio of the pulley. When there is just one noose in the cable, this transfer ratio is 1/2, see figure 2.4.

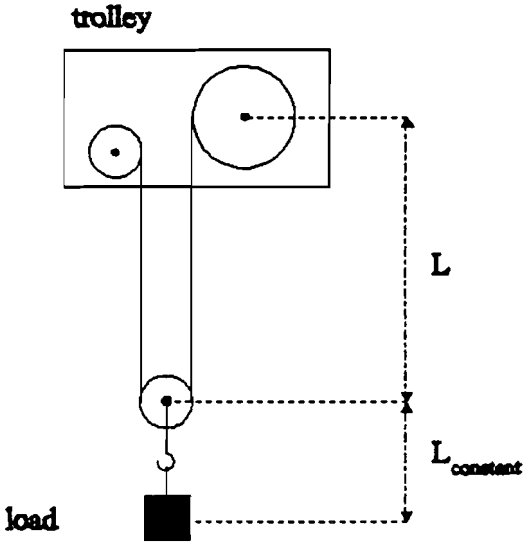


Fig.2.4. Transfer ratio pulley

4. The MIMO system can be seen as a SISO system with variable cable length. See figure 2.5.

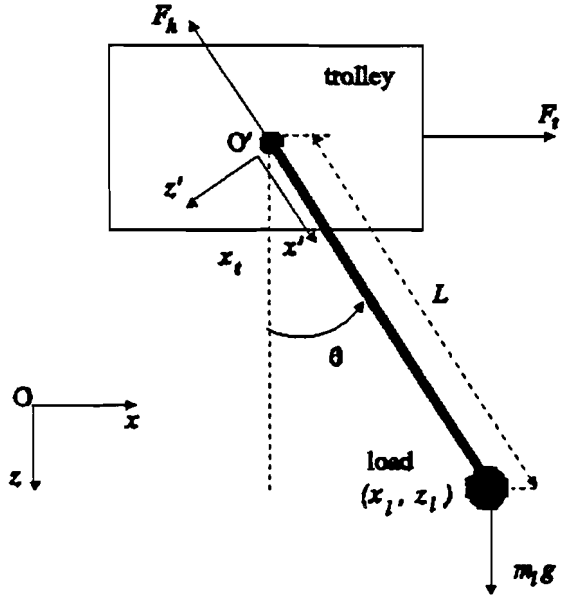


Fig.2.5. Schematic MIMO crane system and parameters

In this figure also the orientation of the angle and the co-ordinates is given. The roll-movement of the cable over the wheel of the pulley, which can influence the angle measurement, is neglected by concentrating the physical dimensions of the pulley in one point O'.

5. The total mass of the load m_l consists of a constant part, namely the pulley and the cable, and of a variable part, the load itself. In the equations this distinction is not taken into account. The mass m_l is considered to be constant in one simulation run. The load and pulley and cable mass are concentrated in one point.

6. The hoisting force F_h can have a value in one of the following categories:

1. $F_h > m_l g\cos\theta + F_{cf}$
2. $0 \leq F_h < m_l g\cos\theta + F_{cf}$
3. $F_h = m_l g\cos\theta + F_{cf}$
4. $F_h < 0$

Where F_{cf} is the centrifugal force. In situation one the hoisting force becomes larger than the gravity force and the centrifugal force; this means that hoisting occurs. In this case the cable length decreases. In situation two the hoisting force becomes smaller than the sum of the gravity force and the centrifugal force. The load is let down and the cable length increases. In situation three the load is halted. Situation four cannot occur in reality, because the cable can't be pushed downwards. In this case, the same effect occurs as if F_h is zero.

7. The initial conditions and bounds of variables are discussed in more detail in chapter 6.

We would like to derive a set of nonlinear differential equations where the three outputs of the system x_t, L and θ are expressed in the two inputs F_t and F_h and in the other known variables. First, the equations of motion are derived with the Newton laws. Then the unknown variables are eliminated and the differential equations are determined.

The co-ordinates of the load are governed by the following equations:

$$x_l = x_t + L\sin\theta \tag{2.1}$$

$$z_l = L\cos\theta \tag{2.2}$$

The length L is a function of time. L reflects only the part of the grab cable that can be changed in length. See also figure 2.4.

The force balance for the motion of the trolley is:

$$\text{x-direction:} \quad \sum F = m_t\ddot{x}_t + d_t\dot{x}_t = F_t + T\sin\theta \tag{2.3}$$

with $d_t > 0$: damping constant trolley
T : tension grab cable

The force balance for the motion of the load is:

$$\text{x-direction:} \quad \sum F = m_l\ddot{x}_l + d_\theta L\dot{\theta}\sin\theta = T\sin\theta \tag{2.4}$$

with $d_\theta > 0$ damping angle swing

$$\text{z-direction:} \quad \sum F = m_l\ddot{z}_l + d_\theta L\dot{\theta}\cos\theta = T\cos\theta - m_l g \tag{2.5}$$

The equation of motion with respect to hoisting can be found in the co-ordinate system $\Sigma' = O'x'z'$, see figure 2.5. This co-ordinate system is moving and rotating with respect to the inertial co-ordinate system $\Sigma = Oxz$. Because the motion of Σ' is not constant with respect to Σ, an extra force is introduced. The rotation introduces a second extra force, namely the centrifugal force F_{cf}, in the equation.

$$\begin{aligned} \sum F &= m_l\left(\ddot{L}+\ddot{x}_l\sin\theta\right) + d_l\left(\dot{L}+\dot{x}_l\sin\theta\right) = \\ &= -F_h + F_{cf} + F_z\cos\theta = \\ &= -F_h + m_l\dot{\theta}^2 L + m_l g\cos\theta \end{aligned} \tag{2.6}$$

where $d_l > 0$: damping in pulley

So far we have found six basic equations with six unknowns (x_l, z_l, $\dot{x}_l$, L, θ and T).
By double differentiating formulas (2.1) and (2.2), eliminating the tension of the grab cable T and the co-ordinates x_l and z_l, we obtain the following nonlinear equations of motion for the MIMO system (see appendix A):
Equation of motion for the position x_t:

$$\ddot{x}_t = \frac{1}{m_t}\left[F_t + F_h\sin\theta + (d_l\sin^2\theta - d_t)\dot{x}_t + d_l\dot{L}\sin\theta \right] \tag{2.7}$$

Equation of motion for the angle θ:

$$\begin{aligned} \ddot{\theta} = -d_\theta\dot{\theta} + \frac{1}{Lm_t}\Big[& -m_l g\sin\theta - 2m_l\dot{L}\dot{\theta} - F_t\cos\theta - \\ & - (F_h + d_l\dot{L})\sin\theta\cos\theta - (d_l\sin^2\theta - d_t)\dot{x}_t\cos\theta \Big] \end{aligned} \tag{2.8}$$

Equation of motion for the length of the grab cable L:

$$\ddot{L} = \frac{1}{m_l m_t}\Big[-m_l F_l \sin\theta - (m_t + m_l \sin^2\theta)(F_h + d_l \dot{L} + d_l \dot{x}_l \sin\theta) + \\ + m_l d_l \dot{x}_l \sin\theta + m_l m_t \dot{\theta}^2 L + m_l m_t g \cos\theta \Big] \tag{2.9}$$

So we are left with the equations (2.7), (2.8) and (2.9) in the unknowns x_t, L and θ.

In the SISO case, the length of the grab cable is fixed. This means that terms with the hoisting force and derivatives of the cable length are zero. The equations (2.7), (2.8) and (2.9) reduce to the SISO dynamic equations of motion (see also Appendix A) [4]:

$$\ddot{x}_t = \frac{F_t + m_l g \sin\theta\cos\theta + m_l \dot{\theta}^2 L \sin\theta - d_l \dot{x}_t}{m_t + m_l \sin^2\theta} \tag{2.10}$$

$$\ddot{\theta} = -d_\theta \dot{\theta} + \frac{-F_t \cos\theta - m_l \dot{\theta}^2 L \sin\theta\cos\theta - (m_t + m_l) g \sin\theta + d_l \dot{x}_t \cos\theta}{L\left(m_t + m_l \sin^2\theta\right)} \tag{2.11}$$

With these nonlinear equations of motion, the nonlinear behaviour of the process can be simulated with SIMULINK. This will be done in chapter 6 for the MIMO case.

2.3. MACS

To be able to measure the behaviour of processes without the need to built special measurement systems, IPCOS[1] developed the program MACS. MACS stands for Measurement And Control System. The MACS program consists of routines, which take care of the interface between the computer and measurement card (*Keithly Metrabyte™*). The routines have been written in language C. Together with this measurement card, the MACS program forms a measurement system. There are fourteen input channels available to measure sensor signals and two output channels to excite and control the process. For this measurement system we need at least an 80386-computer with a VGA-card.

The excitation signals can be generated by MATLAB and the measured signals can also be loaded into MATLAB. For control purposes it is also possible to compute control signals with the help of the measured signals. The user has to program this herself.

The gantry crane process is not directly coupled with the measurement system. To protect the input of the computer, a galvanic separation has been installed. This separation consists of input and output modules. The input module is used to read a voltage or current from the sensors. The input range of this module has to be in agreement with the voltage range of the signals to be measured. The input module under consideration has a range between 0 volt and 10 volts, unipolar.

The voltages are with the help of a multiplexer and a 'sample and hold' digitized by the 12-bits AD-converter of the DAS-16 card (*Keithly Metrabyte™*). Each signal has a range of $2^{12} = 4096$ steps independent of the input range. The DA-converters have the same resolution.

2.4. Offset correction input

The input signal of the system coming from the DA-converter is an unipolar signal, limited between 0 volt and 5 volts. We need positive and negative excitation signals to be able to drive the motor in both positive and negative direction. Therefore we substract 2.5 volts from the DA-converter signal by an offset correction circuit, see figure 2.6.

[1] Integrated Production COntrol Systems (IPCOS b.v.)

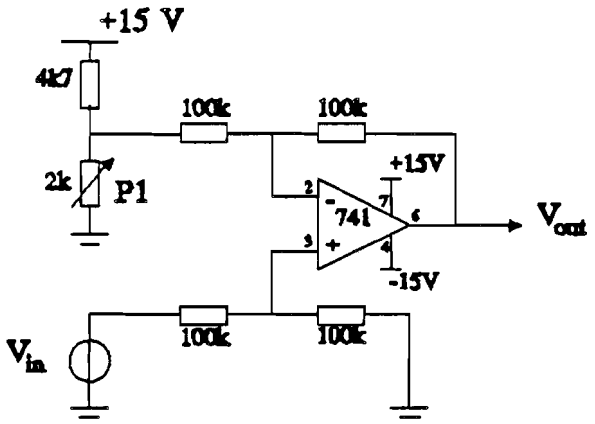


Fig.2.6. Offset correction circuit

Potentiometer P_1 is adjusted such that the output V_0 is exactly zero when the input voltage is proportional to a DAC value of 4096/2=2048 (12-bits DAC).

2.5. Sensors

The SISO system with a stiff pendulum of fixed length attached to the trolley has been described in section 2.1. It is mentioned there, that the x-position of the load can be calculated from two signals, namely the x-position of the trolley and the angle-swing of the pendulum. These sensors and also the protection switches are shortly described in this section.

2.5.1. Position trolley

The position of the trolley on the rail is measured by a potentiometer coupled on the axis of the DC-motor. The potentiometer (*Bourns*™) has a value of 10 kΩ with a resolution of 3% and a nonlinearity of 15%. The position sensor produces an output between 0 volt (begin of the rail) and 10 volts (end of the rail), proportional to the position of the trolley on the rail. The rail is about 1.7 meter long, which yields:

$$x_{t,m} = 1.7 \frac{V_{out}}{10} \quad (m)$$

where: $x_{t,m}$ = measured position trolley

V_{out} = output voltage potmeter

2.5.2. Angle pendulum

The pendulum has been attached to the axis of the angle sensor. The sensor has been attached to the trolley. The angle of the pendulum swing is measured with this sensor. The sensor is a 1kΩ precision potentiometer (*Green*™) which can be turned infinitely through its zero point. The resolution is 10^{-3} % with a nonlinearity of 1%. The output of the sensor is between 0 volt and 10 volts, dependent on the angle swing. Effectively, the output is between 2.34 volts and 6.84 volts, because of the maximum angles reachable. These values are determined by experience, see figure 2.7.

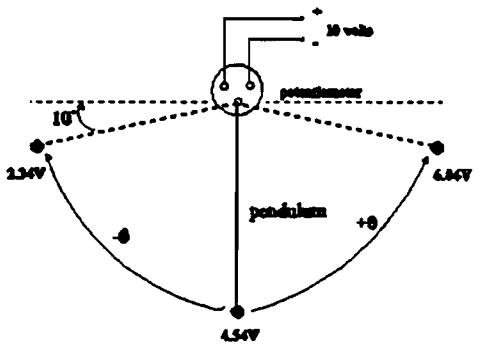


Fig.2.7. Angle measurement

When the pendulum hangs straight down, the angle is calibrated for $\theta=0$. In this example $V_{out}=4.54$ volts agree with $\theta=0$.

2.5.3. Protection switches

Protection switches are installed at the beginning and the end of the rail. When the trolley reaches one of those switches, the knob will be pushed by the little wheel of the switch and contact is made between point 1 and 4 (see figure 2.8). For one direction of rotation the motor is stopped. In appendix B the connection diagram of the switches is given.

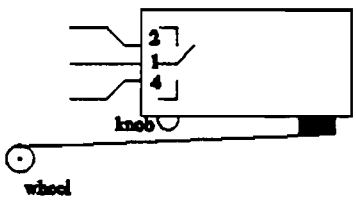


Fig.2.8. Protection switch

2.6. The Servosystem

The Servosystem consists of three components: the Servo-amplifier, the motor and the tachogenerator. The Servo-amplifier is an *Hauser Elektronik G.M.B.H.*™ transistor amplifier of type SV-042a. Tachometer feedback is used to control the number of revolutions of the motor. The connection diagram of the Servosystem is given in appendix B.

The question is: how does the Servosystem influence the dynamics of the process? First the relation between the input signal of the servomotor and the force on the trolley is given. Second, the equation of motion of the trolley is derived, where the moment of inertia of the servomotor is taken into account.

The torque of the DC-motor is proportional to the input voltage of the Servo-amplifier:

$$T = CV_{in}$$

with T = torque
V_{in} = input voltage
C = constant (2.12)

The force on the trolley now is:

$$F_t = \frac{T}{r} = \frac{C}{r}V_{in} \tag{2.13}$$

with r : radius of the pulley

The torque of the DC-motor <u>without load</u> holds:

$$T = J_m\ddot{\phi} + D_m\dot{\phi}$$

with T : torque
J_m : inertia of motor
D_m : damping
ϕ : angle of rotation (2.14)

The damping term D_m can be established by the tachogenerator. With equation (2.13) and

$$r\ddot{\phi} = \ddot{x}_t$$
$$r\dot{\phi} = \dot{x}_t$$

we can rewrite equation (2.14) as:

$$F_t = \frac{J_m}{r^2}\ddot{x}_t + \frac{D_m}{r^2}\dot{x}_t \tag{2.15}$$

If we consider the system with the trolley as load, equation (2.3) is expanded by an extra term:

$$F_t + T\sin\theta = \left[m_t + \frac{J_m}{r^2}\right]\ddot{x}_t + \left[d_t + \frac{D_m}{r^2}\right]\dot{x}_t = M\ddot{x}_t + D\dot{x}_t \tag{2.16}$$

with M = totalmass
D = total damping
T = tension in grab cable

3. NEURAL NETWORKS

Many types of neural networks are known. In this chapter a short introduction on neural networks is given and the class of multilayered neural networks and the backpropagation learning algorithm are discussed [1,2,3,8].

3.1. Multilayered neural networks

A neural network is a graph of processing elements or neurons. The structure of a processing element is given in figure 3.1.

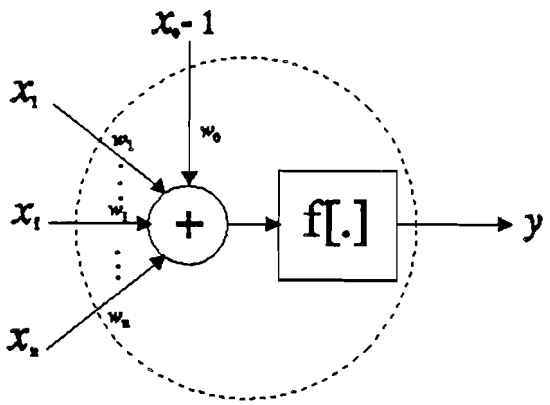


Fig.3.1. Processing element

The processing element first calculates the weighted sum of input signals x_i from the previous layer and adds a bias term to it. The weigths are denoted by w_i. The bias value x_0 is equal to a constant unity processing element at the previous layer. The output y is generated by passing the sum of the weighted inputs through an activating function f[.]. This function can be a linear, ramp, step or sigmoid function. Sigmoid functions are of primary interest, because they exhibit a linear, nonlinear and saturation behaviour, which is useful in nonlinear system identification. The relation between the input vector $\underline{x}$ and the output scalar y of a processing element can be described by formula (3.1) [1].

$$y = \mathrm{f}\left[\sum_{i=1}^{n}(x_i.w_i) + x_0.w_0\right] \tag{3.1}$$

A general multilayer feedforward neural network can be depicted as in figure 3.2. In this type of neural networks, the information propagates only in one direction, from the input layer to the output layer, as indicated by the arrows. Each processing element, as

described above, is denoted by a circle. The output of each processing element is connected to all the processing elements in the first "upstream" layer.

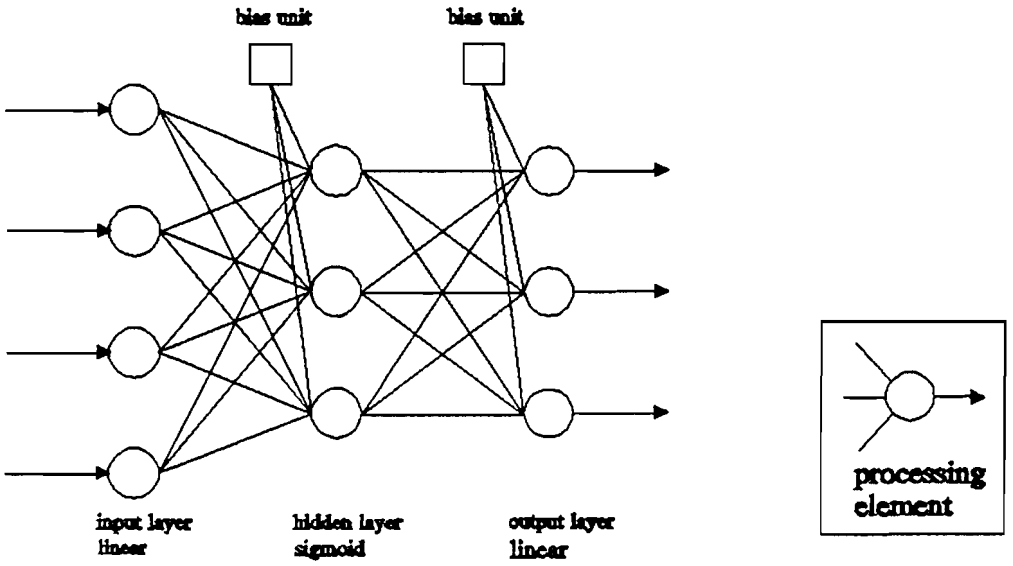


Fig.3.2. Multilayered feedforward neural network

The input nodes of the input layer distribute the inputs to the first hidden layer. So the input layer can be considered to have processing elements with unity weights on the input and a linear activating function, which means that f[x]=x. The hidden layers consist of processing elements with sigmoidal activating functions. In our study, we use a linear activating function in the output layer to operate over a wide range of output values and to prevent from restrictions on the bounds of the sigmoidal functions. In spite of this property, scaling of the output is recommended for numerical reasons.

The structure of a multilayered neural network is indicated by [2]:

$$N_{i_0 i_1 \ldots i_{L-1} i_L}$$

where:
- L number of layers (without input layer)
- $L-1$ number of hidden layers
- i_0 number of inputs
- i_L number of outputs
- $i_1 \ldots i_{L-1}$ number of processing elements in respectively layer 1 ... L-1

So for instance the neural network in figure 3.2 is indicated by $N_{4\,3\,3}$. The number of parameters of the neural network is equal to the number of weighting factors and depends on the size of the network, the number of inputs and the number of outputs. The number of parameters Q can be derived from the formula (3.2) [1]:

$$Q = \sum_{l=1}^{L} (i_{l-1} + 1) i_l \qquad (3.2)$$

Increasing the number of nodes or hidden layers will drastically increase the number of parameters. The problem of finding a sufficient network structure of minimum complexity and maximum performance is not solved yet.

3.2. Dynamics in neural networks

In the application of system identification, a feedforward multilayered neural network is fed with past inputs and past outputs of the real system to predict the future system output. The number of past inputs and outputs that are fed back depends on the order of the system. This configuration is referred to as the Equation Error method. With this Equation Error method, we find a good prediction model. But for a good controller we like to have a good simulation model, found by minimizing the output error.

This can be arranged in the neural network by replacing the past system outputs by the past outputs of the network. In this Output Error identification method, the structure of the neural network results in a network with delayed recurrent connections from its output neurons to its own input neurons. A neural network with such recurrent connections is referred to as a time-lag recurrent network [3]. Both Equation Error model and Output Error model are illustrated in figure 3.3. In this figure Δ indicates a tapped delay line:

$$(\Delta u)(t) := [u(t), u(t-1), \ldots, u(t-n)]^T \quad \text{for some } n > 0$$

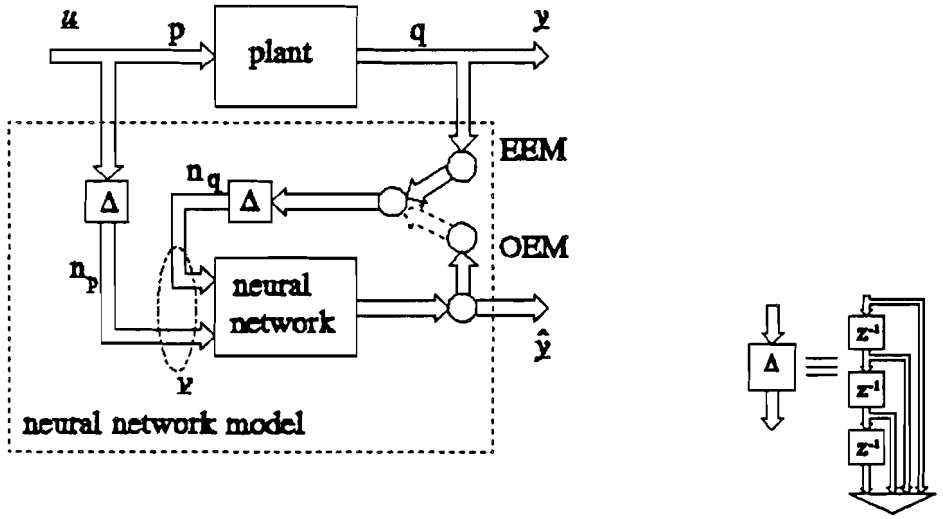


Fig.3.3. The Output Error and Equation Error method

In the application of the Output Error (OE) method, the problem of divergence occurs. To avoid this problem, we need an initial estimate of the weighting factors near by the global minimum. We are going to derive this initial estimate by first estimating with the Equation Error (EE) method. We hope that this will give an initial estimate sufficiently close to the global minimum.

3.3. Backpropagation algorithm

The backpropagation algorithm is used to train a multilayered (feedforward) neural network to obtain suitable weighting factors. For this, the following criterion function J has to be minimized:

$$J = \frac{1}{N_b} \sum_{k=1}^{N_b} \sum_{p=1}^{i_L} \tfrac{1}{2}[\hat{y}_p(k) - y_p(k)]^2$$

where: $\hat{y}_p(k)$: model output p
$y_p(k)$: system output p
N_b : batch size
i_L : number of outputs (3.3)

The batch size N_b is the number of training patterns over which the gradient is calculated. Every N_b samples the parameters of the neural network are adjusted. The backpropagation involves two steps [1,2]:

(1) Forward step
The input training pattern is propagated forwards through the network and the output values of each processing element is computed.

(2) Backward step
The output values of the processing elements in the output layer of the neural network are compared with the target values. These error signals are propagated backwards through the network and the error signals for each processing element are derived. Based on this error signal each weighting factor w_i is adjusted according to the steepest descent method.

Let θ be one parameter of the neural network and α the step size. The adjustment of this parameter reads:

$$\theta_{n+1} = \theta_n - \alpha \frac{\partial J}{\partial \theta}\bigg|_{\theta=\theta_n} \tag{3.4}$$

Where the gradient of J with respect to θ is given by:

$$\frac{\partial J}{\partial \theta} = \frac{1}{N_b} \sum_{k=1}^{N_b} \sum_{p=1}^{i_L} [\hat{y}_p(k) - y_p(k)] \frac{d\hat{y}_p(k)}{d\theta} \tag{3.5}$$

If the neural network is regarded as a function N[.] of its inputs $\underline{v}$ (see figure 3.3), and if we define for one output component

$$\hat{y}_p = N_p[\underline{v}]$$

there holds:

$$\begin{aligned} &\textit{EEM}: \quad \frac{d\hat{y}_p}{d\theta} = \frac{\partial N_p[\underline{v}]}{\partial \theta} && \text{static} \\ &\textit{OEM}: \quad \frac{d\hat{y}_p}{d\theta} = \frac{\partial N_p[\underline{v}]}{\partial \theta} + \frac{\partial N_p[\underline{v}]}{\partial \underline{v}^T} \cdot \frac{d\underline{v}}{d\theta} && \text{dynamic} \end{aligned} \tag{3.6}$$

In the backpropagation algorithm the real gradient of the EE-model is calculated following the first formula in (3.6). In the OE estimation a second term appears in the formula. This term occurs because of the feedback of the past outputs of the model. If the ouput is not highly dependent on its past inputs, this term will be very small. In this case, the OE gradient can be approximated by the EE gradient.

If the learning patterns contain low frequencies, the estimated output $\hat{y}(k)$ is highly dependent on the past outputs $\hat{y}(k-1)$, $\hat{y}(k-2)$,... . In this case we may expect the second term in the second equation of formula (3.6) to be substantial. This equation is also referred to as "dynamic backpropagation" [2].

In the next section this is illustrated by two simulation examples.

3.4. Influence of pole locations

Consider a system with two complex conjugate poles. If the poles are well damped, the amplitude of the impulse response will die out in a short time. However, if the poles are not damped, the amplitude of the impulse response will remain constant. In the gantry crane process, the swinging of the pendulum causes two not well damped complex poles. If the input u is the input of the servomotor and the output y is the x-position of the load, the oscillation frequency of the pendulum is dominant in the output with respect to other system dynamics. The influence of the past is relatively large with respect to the influence of the input on a certain moment.

The question arises if the static backpropagation algorithm is able to train a neural network sufficiently to identify such a system. To illustrate the influence of the pole locations on the quality of a neural network model, two examples are given. Both

examples are simulations without noise and with a random, uniformly distributed white input in the interval [-1,1].

Example 1. Well damped system.

Consider the transfer function of a second order linear process

$$H(z) = \frac{0.5z^{-1}}{1-z^{-1}+0.5z^{-2}}$$

with stable poles in $0.5 \pm 0.5j$. This process is identified by the neural network with only static backpropagation. This means that the Output Error gradient is approximated by the gradient of the Equation Error method, see formula (3.6). The parameters are given in table 3.1. The learn parameters are defined in section 3.3.

Table 3.1. Parameters example 1.

Neural Network structure	$N_{5\,20\,1}$	Activating function	$\frac{1}{1+e^{-x}}$
Batch size	1	OE-performance	0.0019
Step size	.01	Number of iterations	30

The five inputs of the neural network are *u(k), u(k-1), u(k-2), y(k-1), y(k-2).*
The output error performance is defined as the sum squared error divided by the sum squared signal over all samples *N*:

$$\text{OE-performance:} \quad \frac{\sum_{k=1}^{N} [y(k)-\hat{y}(k)]^2}{\sum_{k=1}^{N} [y(k)]^2} \tag{3.7}$$

In the first plot of figure 3.4 the process and the simulation with the neural network model can not be distinguished. In the second plot the error signal is shown. It is clear from this figure that the error is very small. The peak in the error signal at the first sample moment is a result of the initial states. It can be concluded that the neural network is able to estimate the dynamics of a well damped process (noisefree case), by only approximating the real gradient.

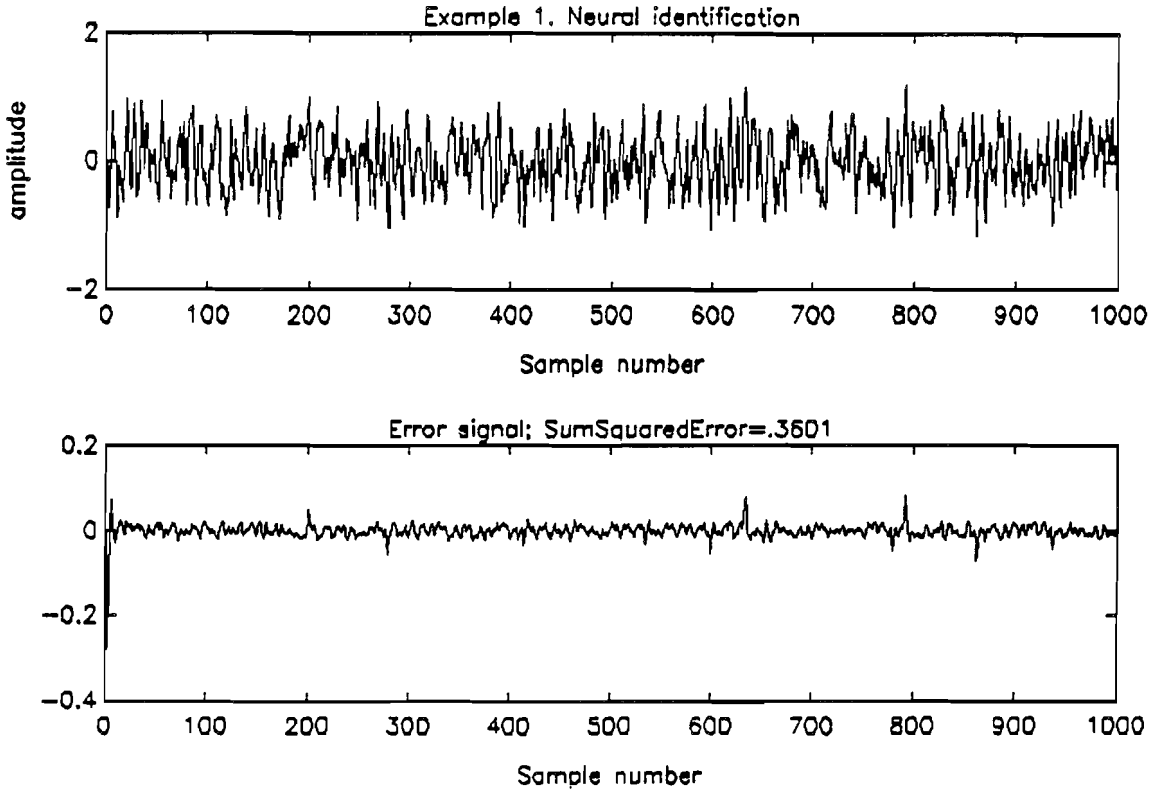


Fig.3.4. Well-damped system

Example 2. Badly damped system

The transfer function of the process

$$H(z) = \frac{0.97z^{-1}}{1-1.8z^{-1}+.97z^{-2}}$$

has stable poles at $0.9\pm0.4j$. Therefore this process is badly damped. The neural parameters are given in table 3.2.

Table 3.2. Parameters example 2.

Neural Network structure	$N_{5\,20\,1}$	Activating function	$\frac{1}{1+e^{-x}}$
Batch size	1	OE-performance	1.082
Step size	.001	Number of iterations	91

The step size is chosen smaller than in example 1, because of divergence problems with higher step sizes. The number of iterations is higher than in example 1. Even though the

prediction performance converges, the output error performance does not decrease further but stays above 1. So the zero output would actually be better! The estimation (dashed line) in comparison with the real output (solid line) is shown in figure 3.5.

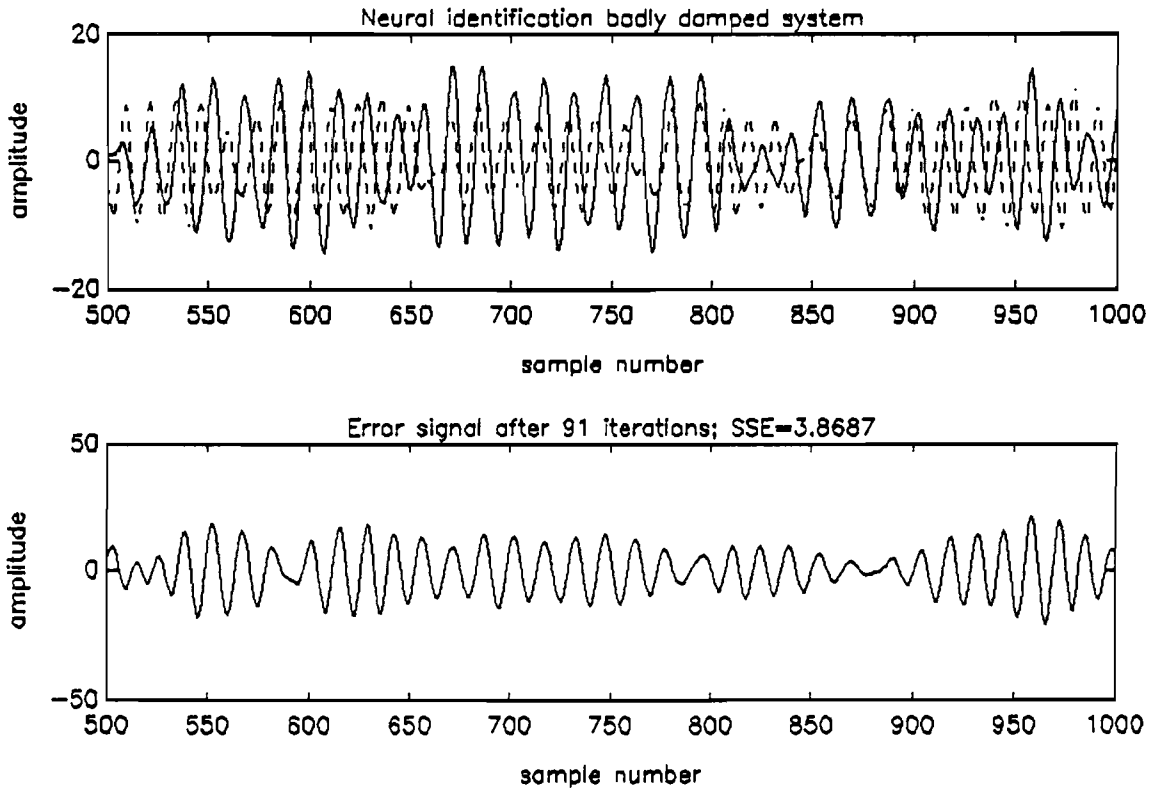


Fig.3.5. Badly damped system

Thus it is shown that it is more difficult to identify badly damped systems with the neural network trained by static backpropagation. We have to calculate the real gradient by means of dynamic backpropagation of the output error model of formula (3.6) in this case to find a simulation model with minimized performance. The initial estimate of the Equation Error model may not be sufficiently close to the real minimum in this case.

4. LINEAR MODEL

In this chapter a linear identification method is presented following the method of Y.C.Zhu [6]. This method is a simple and reliable method to identify badly damped systems. The basic steps of this method consist of high order model estimation and subsequent model reduction.

Assume that the true process is described by:

$$y(k) = G^0(z^{-1})u(k) + H^0(z^{-1})e(k) \tag{4.1}$$

where $G^0(z^{-1})$ and $H^0(z^{-1})$ are the true transfer operators of the process and the disturbance filter respectively. The disturbance e(k) is supposed to be white, Gaussian noise. See also figure 4.1.

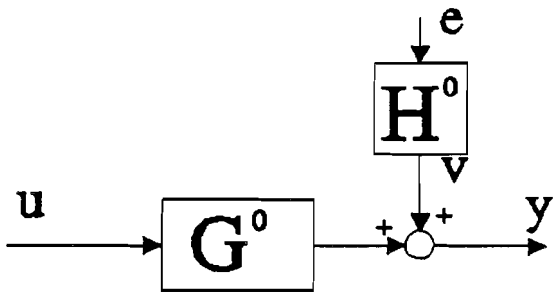


Fig.4.1. True process

We can parametrize this process by an high order ARX model, e.g. n=20-30 if n denotes the model order. Then the model has the form:

$$\hat{A}^h(z^{-1})y(k) = \hat{B}^h(z^{-1})u(k) + e(k) \tag{4.2}$$

and

$$\begin{aligned} \hat{G}^h(z^{-1}) &= \frac{\hat{B}^h(z^{-1})}{\hat{A}^h(z^{-1})} \\ \hat{H}^h(z^{-1}) &= \frac{1}{\hat{A}^h(z^{-1})} \end{aligned} \tag{4.3}$$

Where $\hat{A}^h(z^{-1})$ and $\hat{B}^h(z^{-1})$ are the parameters of the high order model and $\hat{G}^h(z^{-1})$ and $\hat{H}^h(z^{-1})$ are the transfer operators of the estimated model and estimated disturbance filter. As the number of samples $N\to\infty$ and the order of the $n\to\infty$, then the transfer operators of the high order model will converge to the real process.

The high order ARX model can be viewed as the noisy observations of the true process. We apply the maximum likelihood method to these observations to find the low order model. Denote $\hat{G}^l(z^{-1})$ and $\hat{H}^l(z^{-1})$ as the reduced order process model and disturbance model respectively, then the asymptotic maximum likelihood loss function of the process model is [6]:

$$V_1 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \hat{G}^h(e^{j\omega}) - \hat{G}^l(e^{j\omega}) \right|^2 \frac{\phi_u(\omega)}{\left| H^0(e^{j\omega}) \right|^2} d\omega \qquad (4.4)$$

with $\phi_u(\omega)$ = spectrum of the input

and the loss function of the disturbance model is:

$$V_2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \hat{H}^h(e^{j\omega}) - \hat{H}^l(e^{j\omega}) \right|^2 \frac{1}{\left| H^0(e^{j\omega}) \right|^2} d\omega \qquad (4.5)$$

The low order model is found as follows. First the input u is filtered by the inverse noise filter $|\hat{A}^h|$. Then the high order model is simulated to generate the input/output data. The low order model is found by estimating with the OE procedure of the MATLAB TOOL-BOX. The procedure is reflected in figure 4.2.

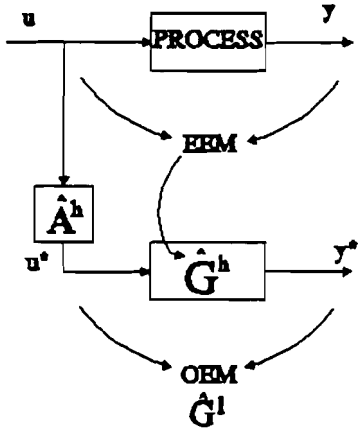


Fig.4.2. Estimation procedure

In this method we look at the identification problem as *approximate modelling*. Hence we do *not* intend to find the 'right' model structure, instead we want to find a model structure such that the reduced order model will have smallest errors in the low frequency range. The consequence of this philosophy is that the selected order can be either lower or higher than the 'true' order of the process. In this selection rule, the order to be selected is related to the signal-to-noise ratio, the experiment time and the order of the high order model [6].

In chapter 5 this identification method will be performed.

5. CRANE PROCESS IDENTIFICATION (SISO)

The experimental design and the identification model are discussed. The results of linear identification will be presented and compared to the nonlinear identification results. Some conclusions are given.

5.1. Experiment design

In this section the system parameters are determined. The experiment to derive the identification dataset of the SISO gantry crane process is described.

5.1.1. Determining system parameters

From the step response experiment, we can determine the oscillation frequency and the damping factor of the pendulum. Figure 5.1 shows the angle of the pendulum when the input signal is a step of 1 volt.

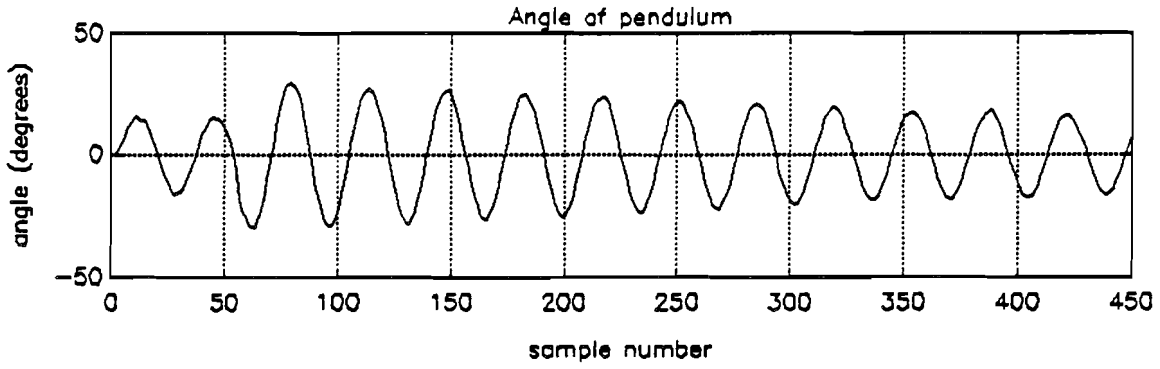


Fig.5.1. Step response angle

After 50 samples (2.5 seconds) the trolley reaches the end switch of the rail and is forced to stop immediately. The pendulum is excited at that point with an extra impulse and oscillates a long time with small damping. In 340 samples (17 seconds) the signal has 10 periods. This corresponds to an oscillation frequency of 0.59Hz.

In 7 periods, the amplitude of the signal decreases with 10 degrees. If we assume constant damping of the pendulum, the rate of decay is 0.02 rad/sec. The largest time constant is then approximately 80 seconds in theory, when the maximum angle of the pendulum is 0.5π. With a sampling frequency of 20Hz, this corresponds to 1570 samples. So in the worst case it can be expected that the influence of the initial conditions has died out after 1570 samples! In the cross correlation between the input signal and the angle of the pendulum, see figure 5.2, it is shown that the amplitude of the impulse response lies in the 95% reliability interval after 1570 samples, with the remark that the dominant

frequency is still in the signal and that the input signal has been filtered and is not white.

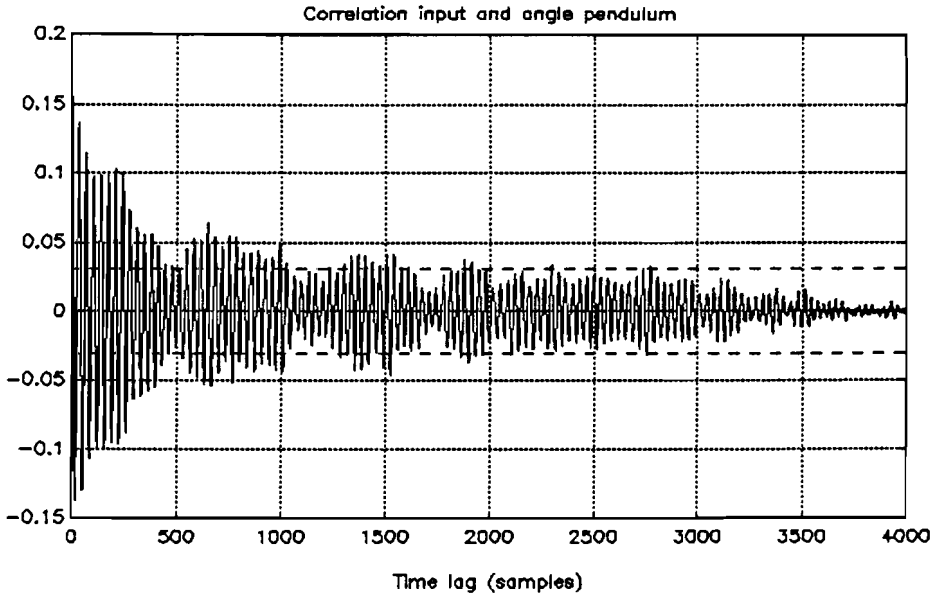


Fig.5.2. Cross correlation

The oscillation frequency of a physical pendulum is given by:

$$f_{pend} = \frac{1}{2\pi}\sqrt{\frac{3g}{2l}} \qquad (5.1)$$

where l = length pendulum (1.10 m SISO)

g = constant of gravity (9.8 m/s^2)

The theoretic value is 0.59 Hz. This is in accordance with the value determined from the step response. In the MIMO situation, this value can vary between 0.43Hz $<= f_{pend} <=$ 1.9Hz because the pendulum length can vary between 0.1 metres and 2 metres.

5.1.2. Sampling frequency

Nyquist states that the sampling frequency has to be at least twice the maximum system frequency, to be able to reconstruct the signal without aliasing. For identification purposes a sampling frequency of about ten times the maximum frequency will do. Because the maximum frequency of the system is the oscillation frequency of the pendulum, this value is maximal 0.59Hz (SISO) and 1.9Hz (MIMO). So a sampling frequency of 20 Hz is amply sufficient.

5.1.3. Bandwidth

The bandwidth of the process can be determined by putting a white noise signal on the input and observing the spectrum of the output. If beyond a certain frequency f_{bw} the amplitude of the spectrum is below some threshold level, say 1% of the amplitude of the spectrum at low frequencies, we may assume that the process has no interesting dynamics above f_{bw}. Of course, the sampling frequency for bandwidth determination should be rather high, just to ensure that all relevant dynamics are captured.

An experiment is done with a random, uniformly distributed input between -2 and 2 volts. The spectrum of the position of the trolley (the output) with 256 samples in the positive band, is shown in figure 5.3. The bandwidth is determined at 2Hz. The amplitude of the signal is then beneath the 10^2 level.

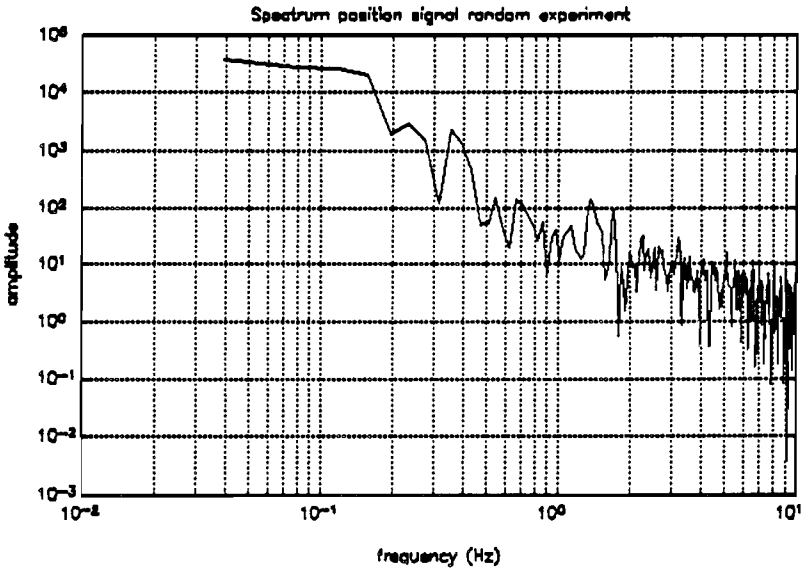


Fig.5.3. Spectrum output random experiment

The amplitude of the low frequencies is very high, because of the integrating term. At 0.59Hz we would expect a peak in the spectrum, because of the influence of the swinging of the pendulum. Two explanations can be given. First, the influence of the swinging of the pendulum on the trolley is relatively small. Second, the number of samples over which the spectrum has been computed is too small.

5.1.4. Input signal

The input signal to the system needs to be a random signal, because otherwise the neural network tunes to the input signal in stead of the input-output relation of the system [8]. A PRBNS signal for example, often used in linear identification, is not suitable as training set for a neural network, because this signal has only two levels. For training we need a whole range of input values. A white noise signal is suitable as training set, but this signal will cause mechanical resonance of the practical system, because of the high frequencies in it. So we have been using a filtered white noise signal. The highest

relevant frequency of the system is the oscillation frequency of the pendulum. The numerical value is determined to be 0.59 Hz (see section 5.1.1). The white noise signal is filtered with a low pass Butterworth filter, with a cut off frequency of 3 Hz. This input signal is rich enough to excite all the relevant modes of the system without causing mechanical resonance, and it is suitable for identification with neural networks. The physical meaning of the input signal is the input to the servomotor, which will cause a force on the trolley.

The input signal to the servomotor is bounded between -1 and 1 volt. The reason for this is the effect of integration of the signal: the trolley will drift to one of the end switches of the rail. When the trolley reaches the end of the rail, the data is not useful anymore for identification. The higher the speed, the earlier this happens.

5.1.5. Time delays

A rough estimation of the time delays of the system can be found by looking at the cross correlation between the input signal and the position of the trolley, see figure 5.4. The position signal is detrended, peak-shaved and offset corrected. The input signal is filtered white noise, as described in section 5.1.4, and the sampling rate is 0.05 seconds.

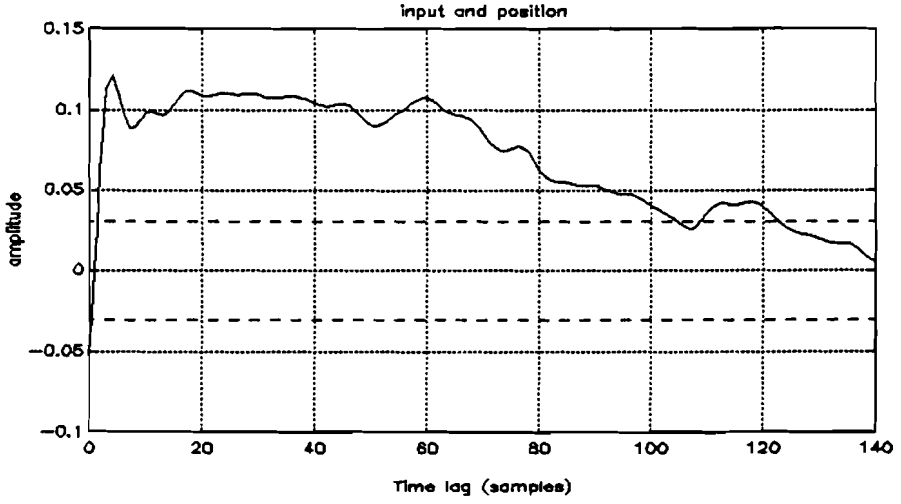


Fig.5.4. Cross correlation

It is clear from this figure, that there exists at most one time delay in the system, because the impulse response starts at almost zero time-lag. The amplitude of the cross correlation is not very high. This implies that the signal-to-noise ratio is poor. Higher amplitudes of the input signal will improve the signal-to-noise ratio, but this will cause the practical problem that the trolley reaches the end of the rail. The solution to this is to stabilize the process first by feedback control. The impulse response of an integrator is a step signal. Because of the integrator in the position signal, the cross correlation should be a step. But the position signal is filtered by detrending and this makes the correlation function decay. The linear technique of pre-processing the data is only used to get insight into the time delays of the system. Of course, pre-processing is not allowed in nonlinear identification.

5.2. Identification

In the first sub section an introduction is given to the general objective of system identification for the process. The next subsections show the neural identification results for different identification schemes.

5.2.1. Introduction

The general objective is to find a neural network model, that describes the input-output behaviour of the process. Therefore we need to
- Define the input and output of the process.
- Design an experiment to acquire the dataset.
- Define the criterium function which has to be minimized.

To define the input and output of the process, we start from the signals that are really available:
- the input to the servosystem
- measured position of trolley
- measured angle deviation from zero point of pendulum.

The x-position of the load has to be controlled (section 2.1). This signal is dependent on the measured variables. Now we can define the output of the process in two ways:
- first compute the x-position of the load out of the two measured signals and identify the indirectly obtained x-position of the load.
- identify both transfers from the input to the position of the trolley and to the angle deviation of the pendulum.

The experiment to acquire the dataset has been described in section 5.1. This means that we have done an experiment with an uniformly distributed white noise input signal, filtered with a 3Hz low pass filter and with an amplitude in the interval of [-1,1]. The sampling frequency is 20Hz.

The error criterium which we want to minimize is defined by formula (3.7) in section 3.4. The learn algorithm of the neural network is described in section 3.3.

In the following subsections, different identification schemes and identification results are presented. In the figures the real outputs are indicated by solid lines and the estimated outputs by dashed lines.

5.2.2. Identification scheme 1

Consider the following identification scheme:

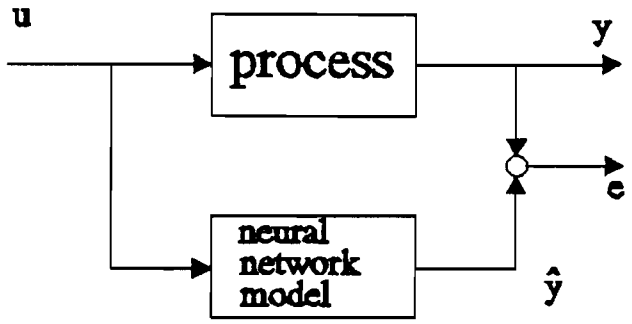


Fig.5.5. Identification scheme 1

Where u denotes the input to the servomotor and y the measured output. The estimated output is denoted by $\hat{y}$. There are two measured outputs:

- the position of the trolley
- the angle deviation of the pendulum

We will try to identify the transfer between the input u and both measured outputs (second possibility as described in section 5.2.1).

Transfer from input to position of trolley

The position of the trolley is not controlled by some feedback system. Because of the servosystem we derive a position signal with an integration term. This means that with a white input signal, the trolley will drift to one of the end switches and a very low frequent trend will be in the data. We saw already that low frequencies may be a problem in identification with neural networks (see remark end of section 3.3). Using the scheme of figure 5.5, the parameters of table 5.1 and the static backpropagation learn algorithm, we find the result of figure 5.6.

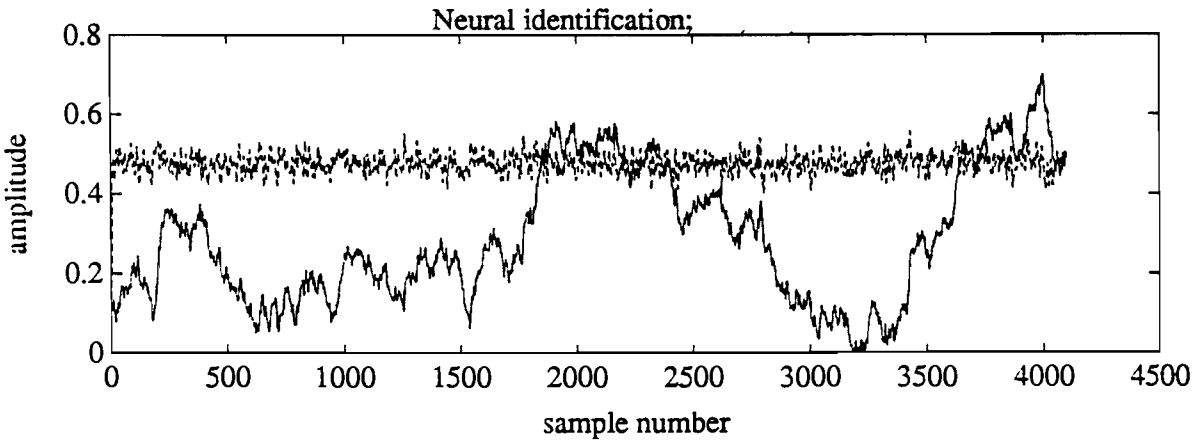


Fig.5.6. Simulation neural network model of postion trolley

Table 5.1. Parameters estimation position

Neural Network structure	$N_{5\,15\,1}$	Activating function	$\frac{1}{1+e^{-x}}$
Batch size	1	OE-performance	bad
Step size	.01	Number of iterations	50

The high frequent behaviour seems to be estimated but the low frequencies are not estimated at all. In fact, the number of inputs should be 9 (*u(k),u(k-1),u(k-2),u(k-3),u(k-4),y(k-1),y(k-2),y(k-3),y(k-4)*) instead of 5, because we expect a fourth order system. But when the influence of the swinging of the pendulum is not too large, and this is the case, a second order model has to approximate the system. We can conclude that the process with data in this form can not be estimated in this way. So we have to find alternatives.

In linear identification a commonly used method is detrending of the data. The very low frequent trend is then filtered out. The term "trend" can be confusing, because we have to deal with a real trend as a result of the drift voltages in the servosystem, which is a disturbance, and a trend as a result of the integration of white noise. The detrending is done in the output of the dataset. The neural network is trained by means of static backpropagation to identify the system from input to detrended output. Because of the filtering the model order will increase. This is not taken into account in the following estimation. The parameters are the same as in table 5.1 and the estimation is shown in figure 5.7.

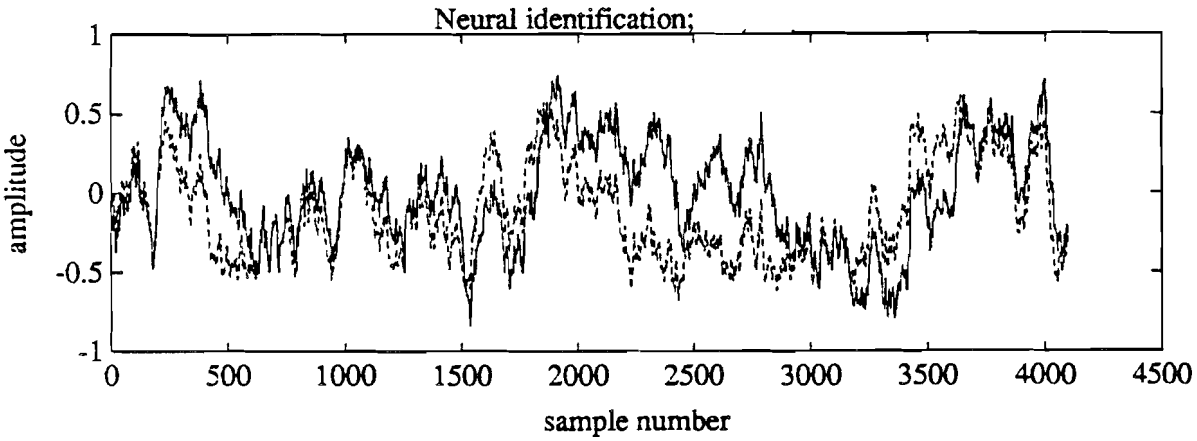


Fig. 5.7. Simulation neural network model after detrending position

The result is better than without detrending. The result would even be better with more iterations and an higher model order. But detrending is not allowed, because the "trend" is mainly caused by the integration and not by the disturbance. We don't know the effect

of detrending on the nonlinear data and so we don't know what model we estimate. So we decided not to go further this way.

Transfer from input to angle

The second transfer we would like to identify is from the input of the servomotor to the angle deviation of the pendulum. The neural network estimation is shown in figure 5.8 and the learn parameters are given in table 5.2.

Table 5.2. Parameters estimation angle

Neural Network structure	$N_{11\ 15\ 1}$	Activating function	$\frac{1}{1+e^{-x}}$
Batch size	1	OE-performance	bad
Step size	.2	Number of iterations	3

The inputs of the neural network are of fifth order, that is *u(k), u(k-1),...,u(k-5),y(k-1),y(k-2),...,y(k-5)*. This high order model is chosen after trying with lower order models.

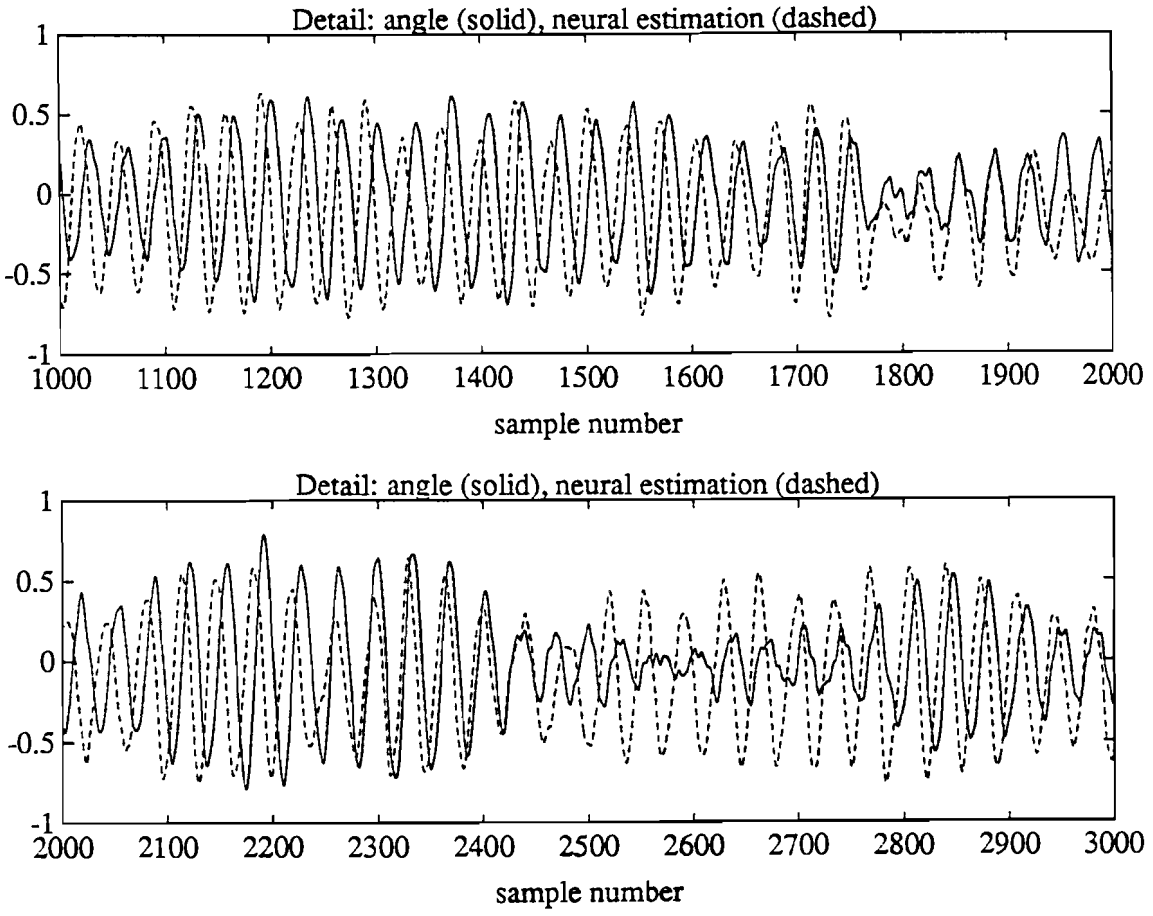


Fig.5.8. Simulation neural network model of angle

We see that the frequency (after only three iterations!) is estimated very well, but we see that the output error performance is poor because of a phase shift. The "zeros" of the neural network are not located very well. This result is to be expected because the system is now badly damped. In section 3.4 we saw already that the neural network can't even be trained for a simulation model with badly damped poles, by only using the static backpropagation algorithm.

5.2.3. Identification scheme 2

Another possibility is, to make the integrator in the position signal of the trolley explicit, see formula (2.10). This can be done by identifying the transfer between the input signal and the velocity of the trolley. But this signal is not available, so we differentiate the position signal. This scheme is shown in figure 5.9.

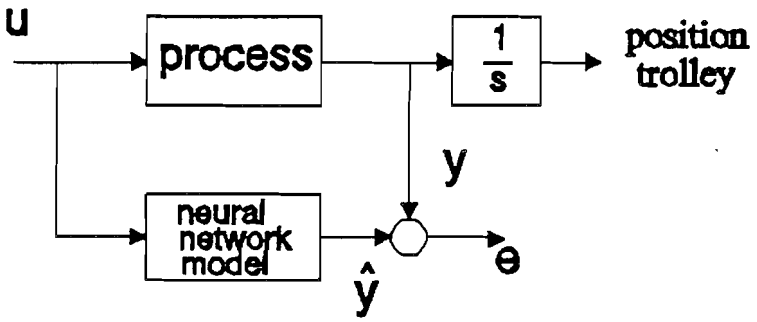


Fig.5.9. Explicit integrator

A part of the dataset of 600 samples without spikes is chosen for identification. The inputs to the neural network are the same as in the previous subsections. The step size is adjusted by hand, when the performance function didn't decrease further. The parameters are given in table 5.3. A remark should be made on the number of inputs. In fact the system is a fourth order system. When making the integrator explicit we are left with a third order system, which means that we should use 7 inputs in stead of 5. But in spite of this, the output error performance is very good.

Table 5.3. Parameters explicit integrator

Neural Network structure	$N_{5\ 15\ 1}$	Activating function	$\frac{1}{1+e^{-x}}$
Batch size	1	OE-performance	.0249
Step size	adjusted	Number of iterations	800

Figure 5.10 shows a simulation with the derived neural network model on the training set. The error signal is also given. On the validation set, see figure 5.11, the output error performance is a little bit worse, namely 0.0379.

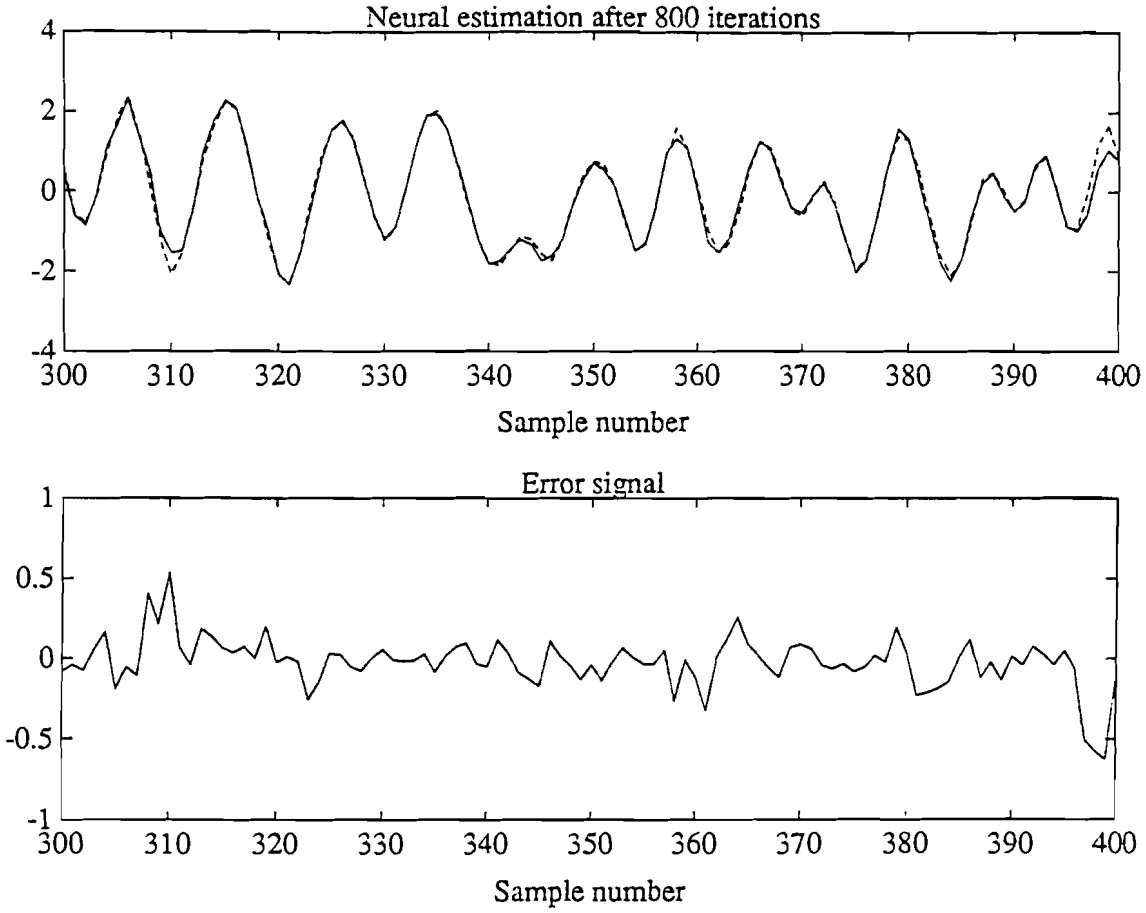


Fig.5.10. Simulation with neural network on differentiated position trolley

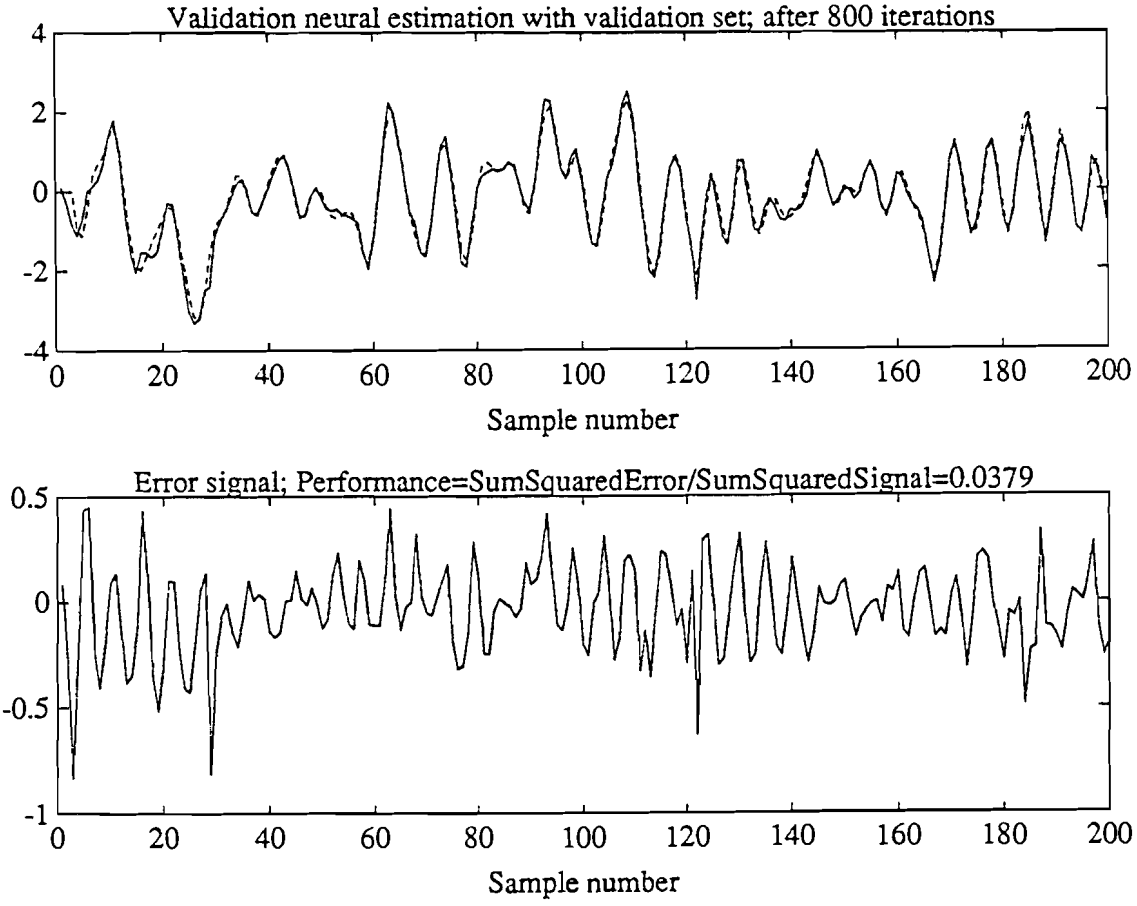


Fig.5.11. Validation with neural network on differentiated position trolley

Some remarks can be made:
- The neural network model is not optimal, because the step size is adjusted by hand and the prediction error (which in fact is minimized) still converge.
- The data seems to be rather linear, the influence of the swinging of the pendulum is little.

We compared the neural network estimations with second order linear estimations, following the method of Y.C.Zhu (see section 4). The OE-performance of the linear model on the training set is 0.052 and on the validation set 0.053. In this case, the neural network performs better than the linear model. See figure 5.12 for the linear model on the training set and figure 5.13 for the linear model on the validation set.

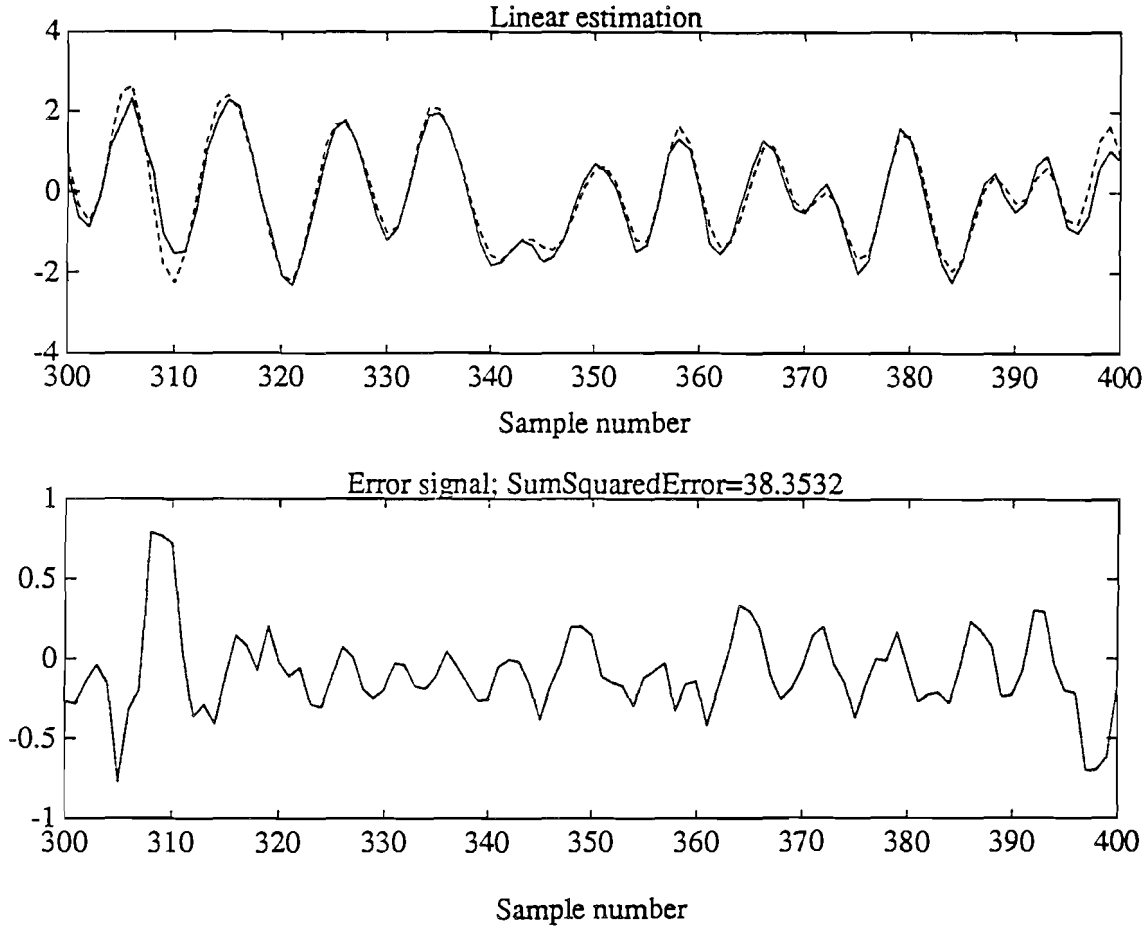


Fig.5.12. Simulation with the linear model on differentiated position of trolley

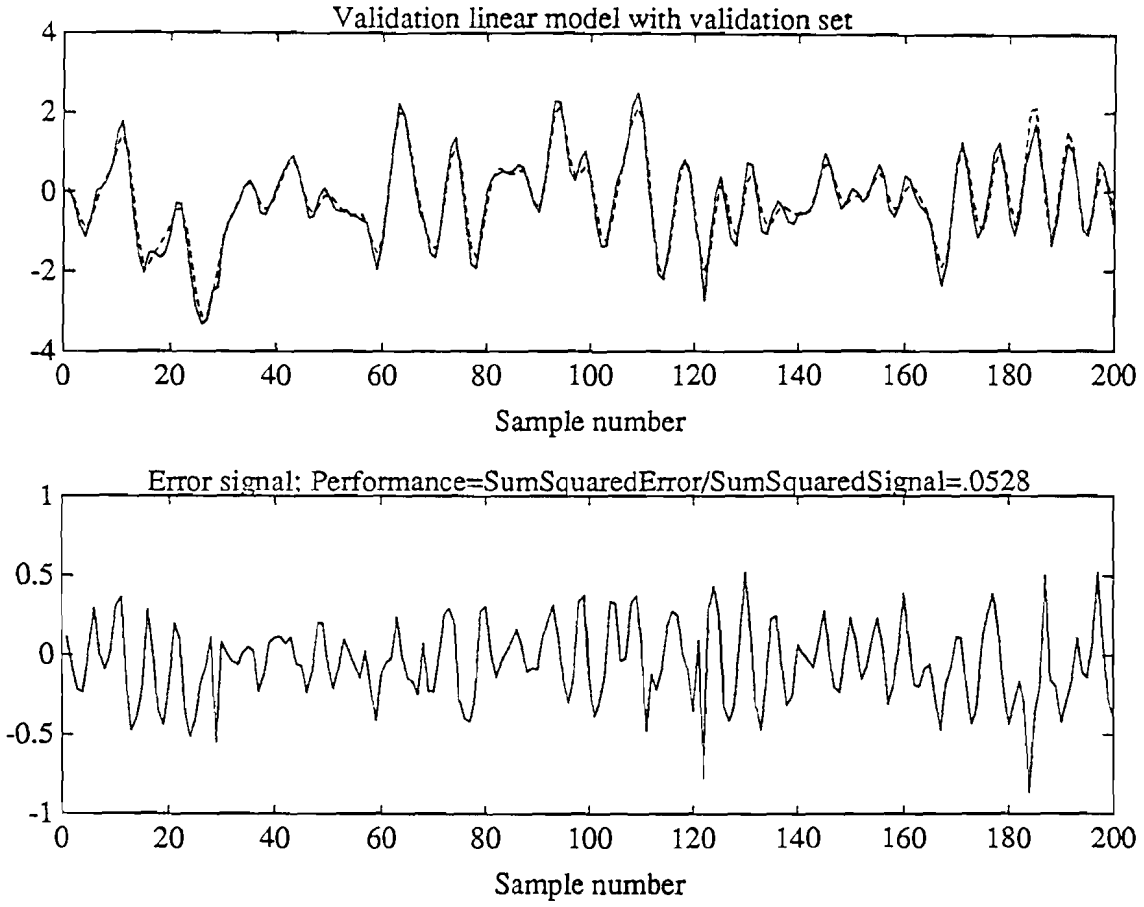


Fig.5.13. Validation linear model on differentiated position of trolley

5.2.4. Identification scheme 3

Because the final goal is to control the position of the load, we tried to identify the transfer between the input of the servomotor and the position of the load. If the measured position of the trolley is denoted by x_t and the measured angle deviation of the pendulum by θ, we can define the x-position of the trolley by:

$$x_l = x_t + L\sin\theta \qquad (5.2)$$

where L = cable length (1.1 m)

The integration in the position signal of the load x_l can be made explicit, but intern the feedback loop exists (see for example the feedback loop in appendix C4). Therefore, the order of the model doesn't decrease. We differentiate the data to get the velocity signal. Three identification examples are shown:

- identification with a neural network
- identification with a linear model following the method of Y.C.Zhu (section 4).
- identification with a neural network in parallel with the identified linear model.

Example 1. Neural network

The inputs to the neural network are *u(k), u(k-1), y(k-1), y(k-2)*. The estimation is done on the whole set of 4096 samples. The parameters are given in table 5.4.

Table 5.4. Parameters neural estimation

Neural Network structure	$N_{4\ 10\ 10\ 1}$	Activating function	$\frac{1}{1+e^{-x}}$
Batch size	1	OE-performance	.19
Step size	.1	Number of iterations	1

Some remarks can be made here.

- The number of memory locations by means of inputs to the neural network is insufficient to capture all the dynamics of the process.
- It is notable in table 5.4 that there is just one iteration. In fact more iterations did not improve the output error performance of the simulation.

See figure 5.14 for the simulation results. In this figure only a part of the estimation is shown.

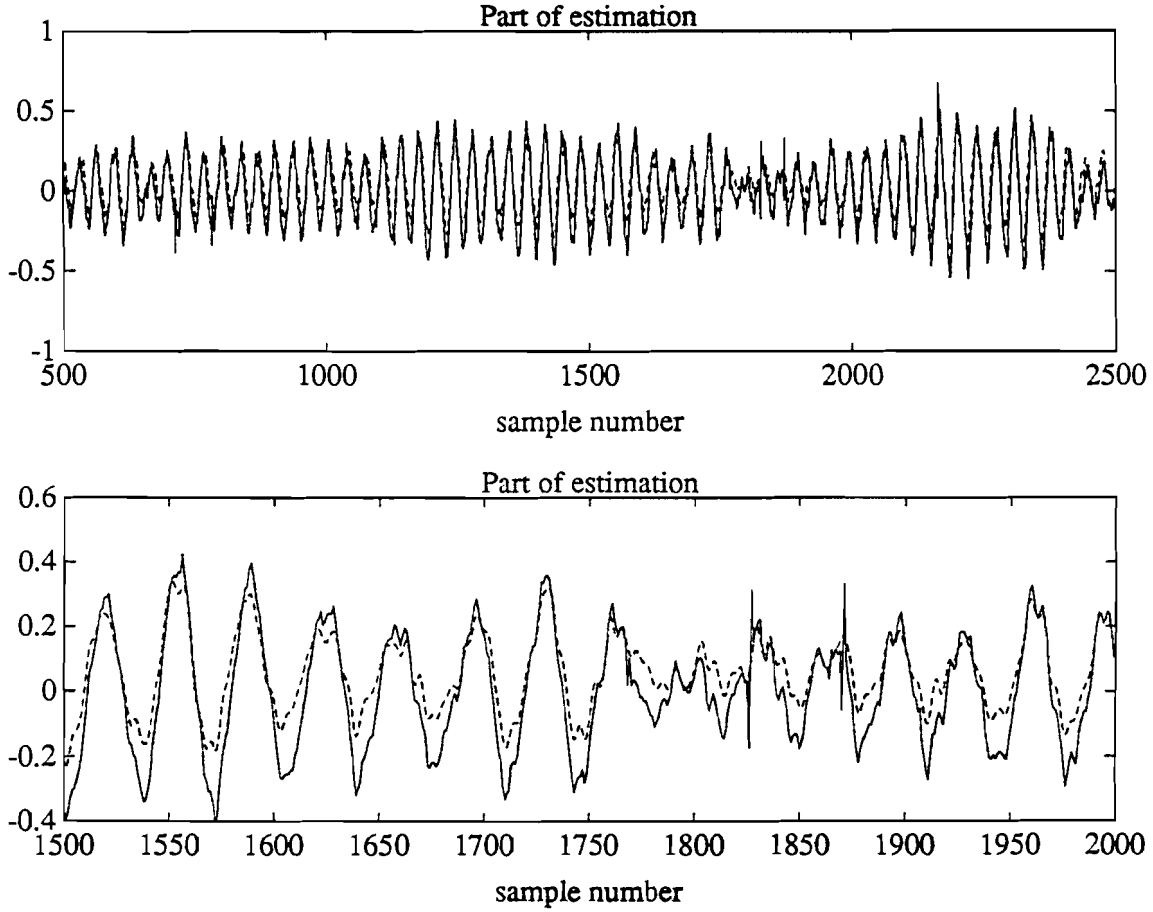


Fig.5.14. Simulation neural network of differentiated x-position load

Example 2. Linear estimation

We would like to compare the performance of the neural network model with a linear model. We choose the method of Y.C.Zhu, see chapter 4, to estimate that linear model. This method is simple and reliable even when the system is badly damped and the length of the dataset is short. Following this method we first estimate a 25th order ARX-model. Then the model reduction has been performed to find a lower order model. We would expect a fourth order model.

We start with the estimation of a third order model. When we compare the spectrum of the simulated output of the high order model with the simulated output of the third order model, see figure 5.15, we see that the important peak at the oscillation frequency of the pendulum is not estimated at all. The same result is found when we estimate a fourth order model. The sixth order model behaves poor in amplitude of the dominant frequency with respect to the seventh order model. When we compare the spectrum of the simulated

output of the high order model and of the seventh order model, we can't distinguish. See figure 5.16.

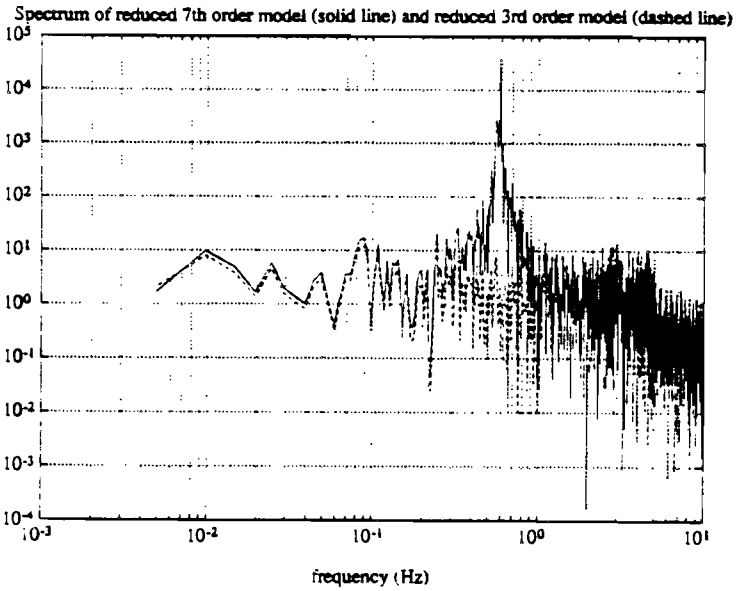


Fig.5.15. Spectrum 3rd order model

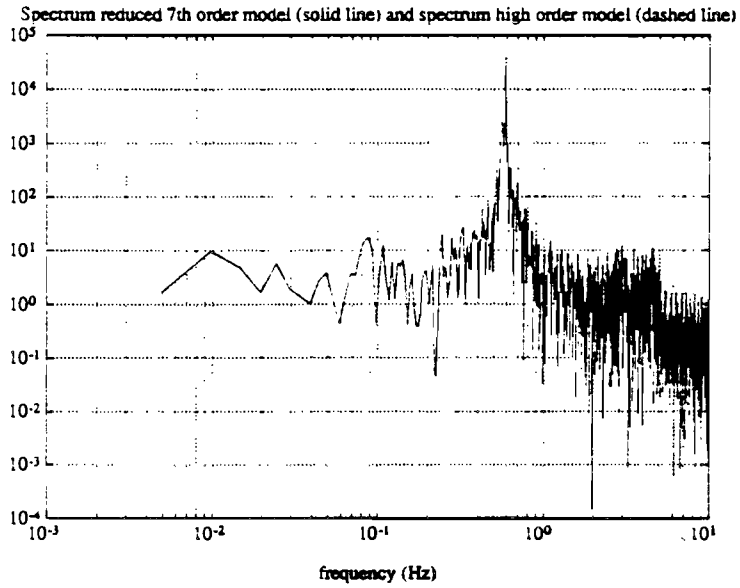


Fig.5.16. Spectrum 7th order model

So we choose the seventh order model. The order seems rather high, but with the remark of Y.C.Zhu (see end of chapter 4) we can explain. A representative part of the simulated output (dashed line) of this model is given in figure 5.17. The output error performance is 0.12, which is better than the neural network model.

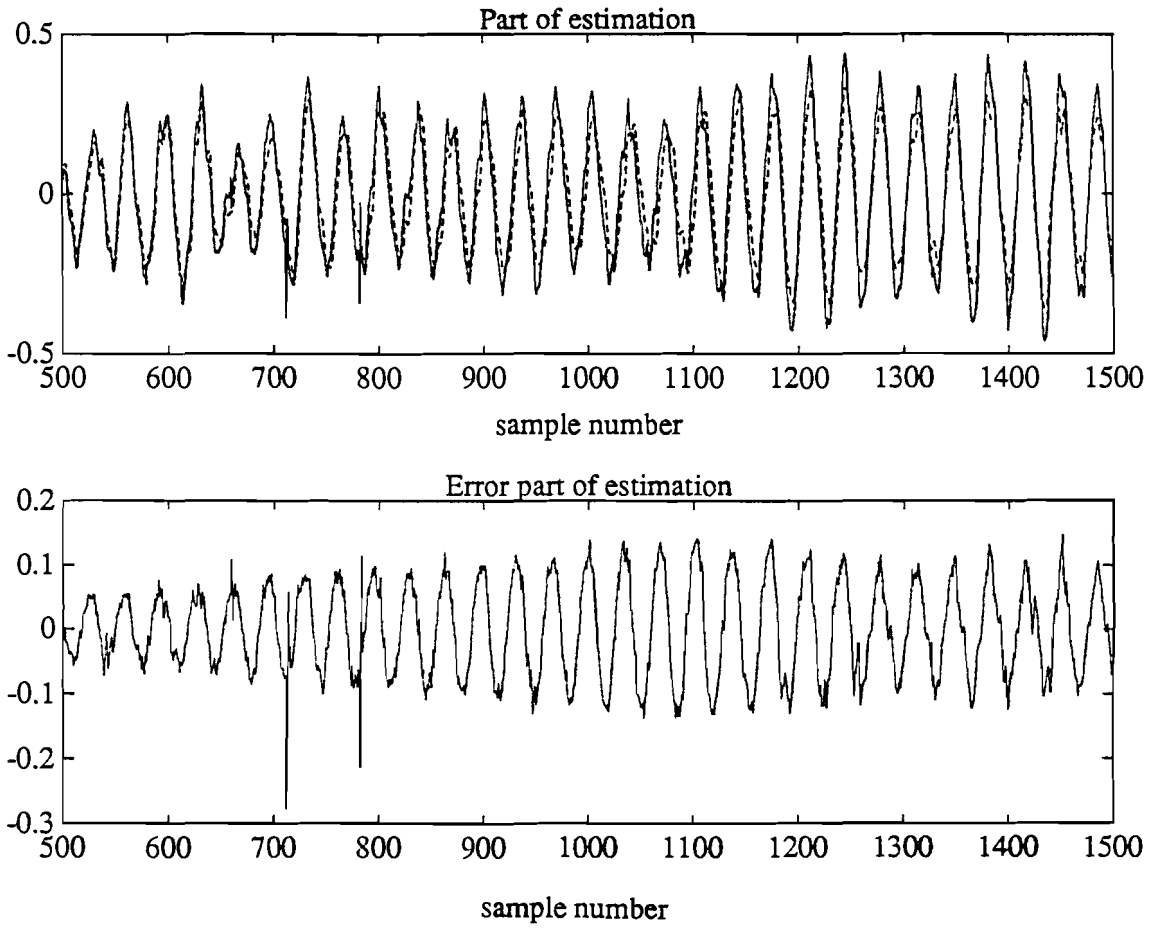


Fig.5.17. Simulation linear model of differentiated x-position load

Example 3. Neural network and linear model parallel

The 7th order linear model, derived in example 2, can be used parallel to the neural network. This identification scheme is shown in figure 5.18.

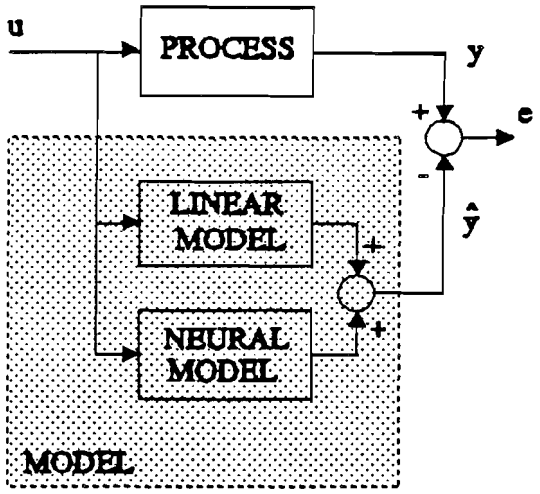


Fig.5.18. Neural network and linear model parallel

In the training procedure of the neural network we used the equation error method. This means that the past outputs of real process are fed back as inputs to the neural network. The parameters of the linear model are not changed during execution, they stay fixed. The idea behind this configuration is, that the neural network only has to estimate the nonlinear part of the data which may improve the output error performance of the linear model. But that this is not the case is shown in figure 5.19. The error with respect to the linear model is almost doubled, because of a phase shift in the estimated signal.

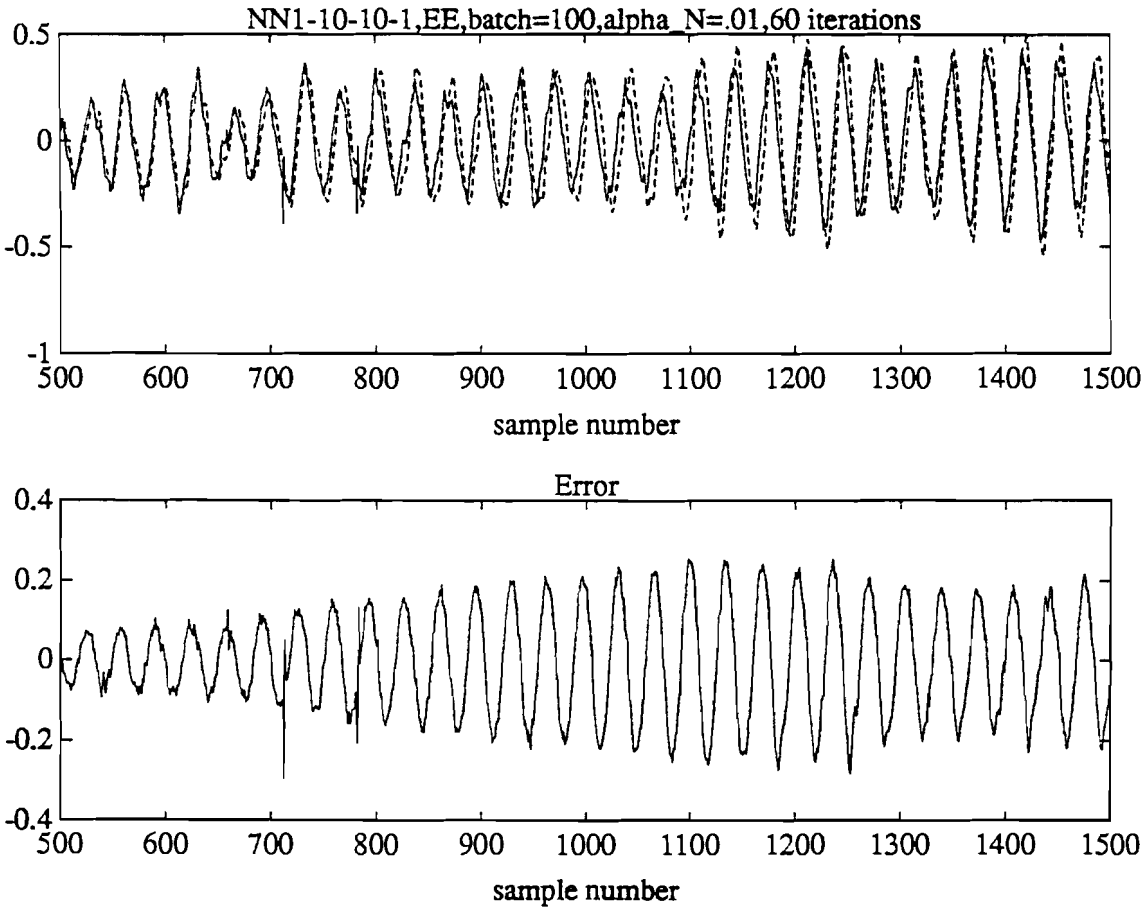


Fig.5.19. Simulation neural network parallel with the linear model

The inputs of the neural network are again *u(k), u(k-1), y(k-1), y(k-2)* and the parameters are given in table 5.5.

Table 5.5. Parameters neural and linear model parallel

Neural Network structure	$N_{4\ 10\ 10\ 1}$	Activating function	$\frac{1}{1+e^{-x}}$
Batch size	100	OE-performance	bad
Step size	.01	Number of iterations	60

The error signal of figure 5.19 is almost doubled with respect to the estimation of only

the linear model (see figure 5.17). This large error is caused by a phase shift in the estimated output. For linear models a phase shift indicates that the zeros of the model are not well placed or there are too few zeros to model the system. In the neural network the number of "zeros" depends on the number of past inputs. In this case the number of inputs to the neural network is insufficient to identify the system. A disadvantage of increasing the number of inputs is the explosion of the number of parameters that have to be estimated.

5.2.5. Conclusions

The integration in the position causes a low frequent trend in the data when a white input signal is used.

- The static backpropagation learn algorithm is not able to train the neural network sufficiently to identify the position of the trolley.
- Detrending of the data, commonly used in linear identification, is not a good solution since we would like to identify the nonlinear system.
- The integrator can be made explicit by estimating the velocity of the trolley. In this case, the output error performance of the neural network model is better than the linear model, for both training and validation set. With more iterations, an automatical adjustment of the step size (see the Neural Network Toolbox of MATLAB) and a larger dataset can also improve the performance of the neural network model.

- The angle deviation of the pendulum is difficult to estimate with the neural network, because of the small damping.
- Therefore the position of the load can not be modelled this way with a neural network model.

6. MIMO SIMULATION AND IDENTIFICATION

The MIMO crane process is simulated with SIMULINK. The implementation is described. An LQG-controller is designed for the linearized process to stabilize the process. The results of the neural identification with backpropagation training are given for a simple MIMO example and for the simulated MIMO crane process.

6.1. Implementation in SIMULINK

In future the MIMO version of the gantry crane process will be operational. Therefore it is nice to simulate this process and look how the neural network identification will perform. Simulated data has some nice properties. The data is noise free and it is possible to create a nice dataset, by choosing carefully the actual values of the variables and the input signals.

The MIMO process is an unstable process, because of the integration of the input signals (see chapter 2). The effect is a very low frequent trend in the data. If we want to identify with neural networks, we have to stabilize the process. This can be done by feeding back the states. It is necessary that the power of the external input $\underline{u}^*$ will be large in respect to the feedback signal $\underline{z}$, see figure 6.1. More power in the feedback signal, gives a more stable process, but the excitation of the external input will become less. This is a trade-off that has to be made by designing the LQG-controller (see section 6.2.3).

Since we would like to identify the system without feedback, the transfer function from the input vector $\underline{u}$ to the output vector $\underline{y}$ will be identified, see figure 6.1.

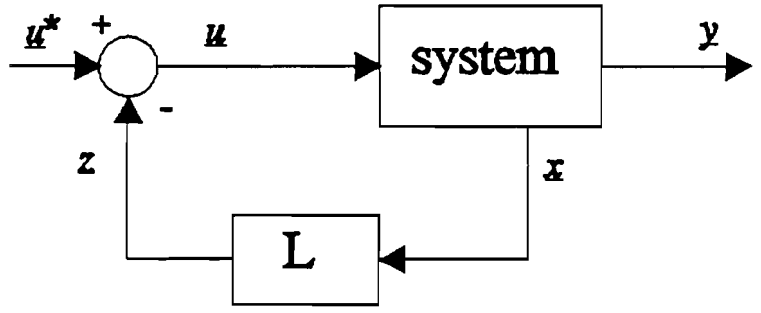


Fig.6.1. Identification scheme

In this figure the block "L" indicates the feedback matrix of the LQG-controller. THe implementation in SIMULINK is denoted by the block "system". In this scheme the real state vector $\underline{x}$ is fed back. This can be done because the simulation is noise free and the

states are all available. In practice we need an observer to estimate the states from the output. It is required that the system is observable and controllable (see section 6.2.2).

The description of the implementation, complete with the input and output variables and the used constants is given in Appendix C. The described implementation in Appendix C agrees with the formulas given in chapter 2. In this chapter 6, the used simulation model contains some errors. Also the numbers given in this chapter are based on this model. The main difference between the two models is, that some terms in the acceleration formula of the grab cable (2.9) are not included. For example the centrifugal force and the dependance on the pulling force. To test the implementation, different responses of the system are evaluated. A MIMO impulse response example of hoisting and pulling is given in figure 6.2 (used model in this chapter) and in 6.3 (model, described in Appendix C). An impulse response on the forces agree with a step on the velocities. In this example the impulse on the hoisting force is 5N and on the pulling force 10N. The damping on the trolley velocity is d_t=0.2 m/s and on the angle d_θ=0.02 rad/s. No damping is considered on the velocity of the grab cable.

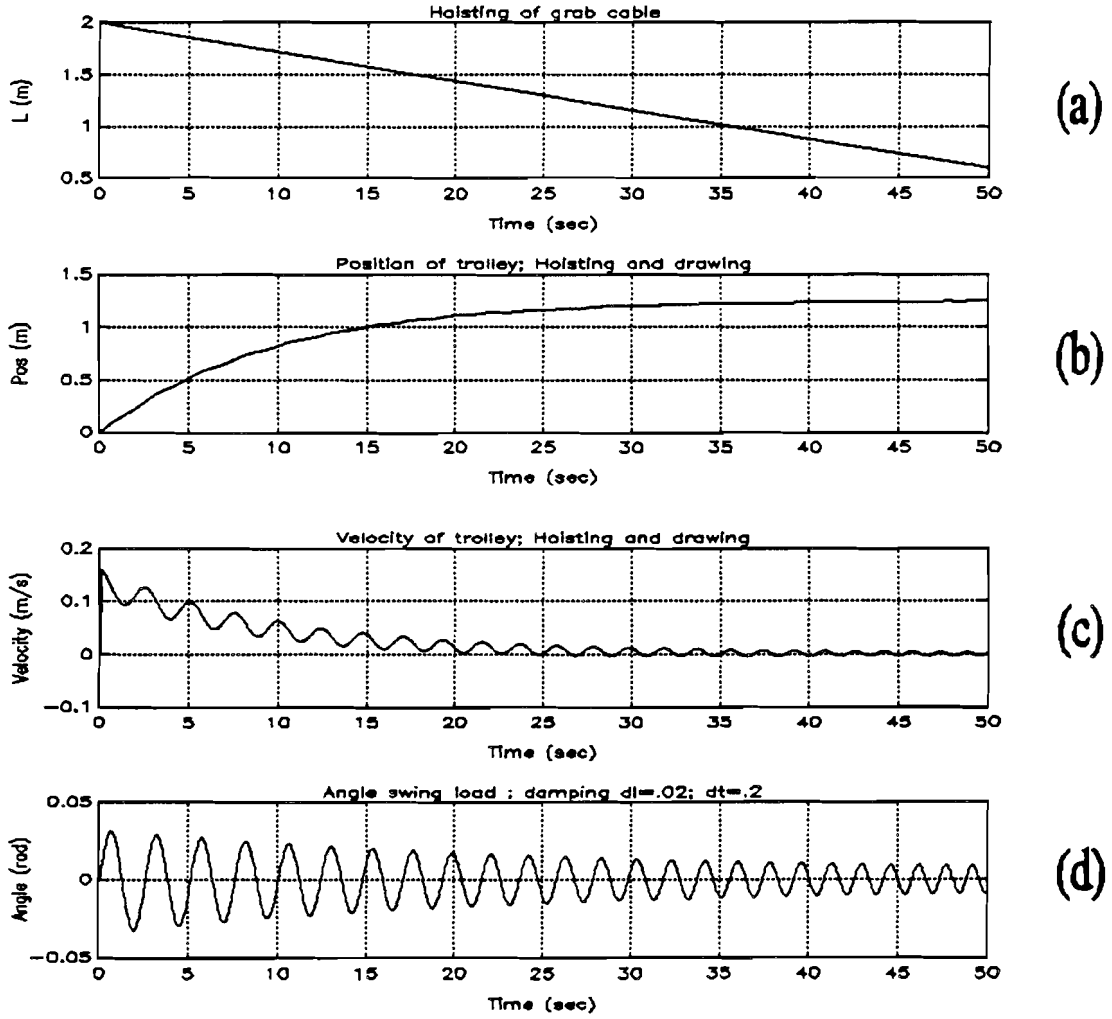


Fig.6.2. Impulse responses simulation model chapter 6

It is clear from figure 6.2.a that the length of the grab cable only depends on the hoisting force and the gravity force (the other terms are not included!). The angle of the load swing is well damped and the frequency is becoming higher because of the decrease of cable length (6.2.b). The velocity of the trolley decreases to zero, because of the damping term (6.2.c). Therefore the position of the trolley stabilizes at a certain level conform expectation (6.2.d).

In figure 6.3 the impulse responses with the simulation model following Appendix C is shown. The damping factors and the impulses on the forces are chosen the same. A small difference is the start value of the cable length, which starts at one metres. We see that after 32 seconds the cable length is increasing. The influence of the angle swing on the velocity of the trolley is larger than in figure 6.2. The angle doesn't decrease as fast as in figure 6.2 with this damping value.

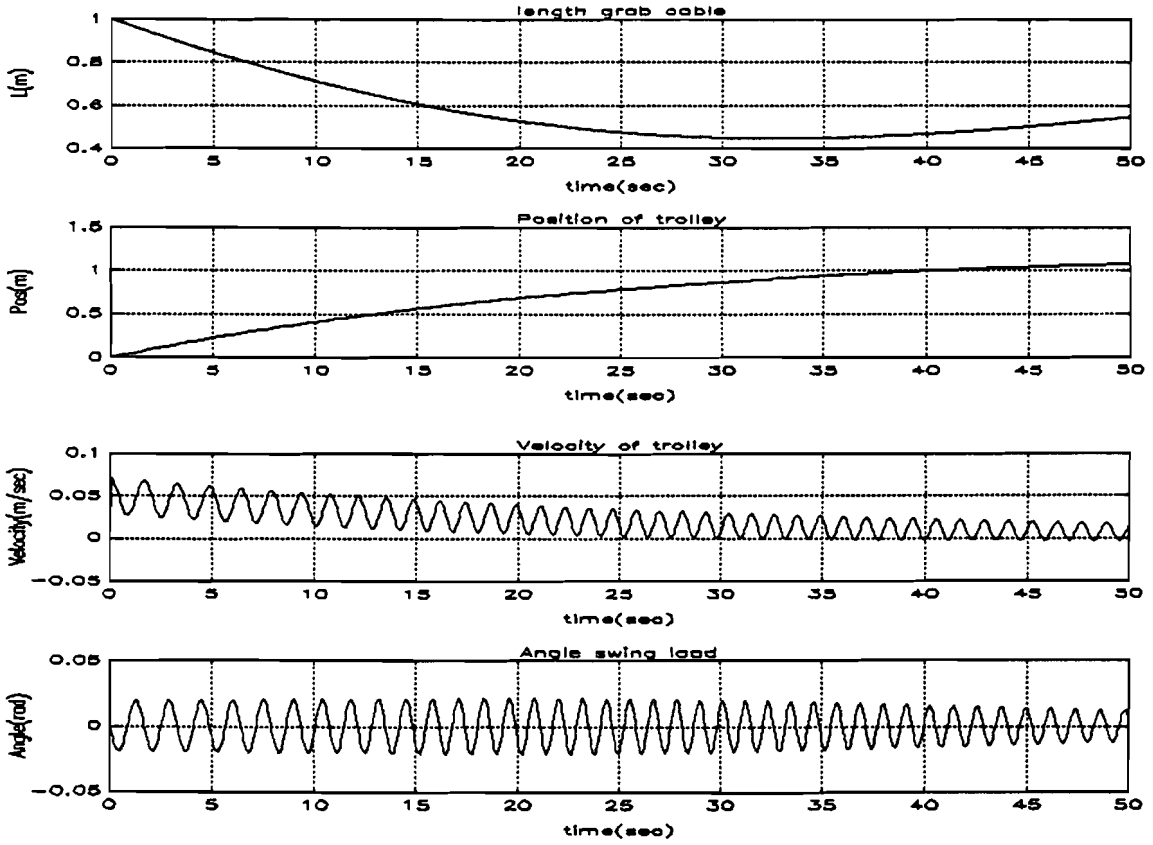


Fig.6.3. Impulse responses simulation model appendix C

6.2. Stabilization

The physical nonlinear equations of motion of the MIMO system are linearized about a working point (section 6.2.1). Observability and controllability is discussed (section 6.2.2). The LQG-controller to stabilize the simulation process is designed (section 6.2.3).

6.2.1. Linear state space description

LQG-control is a technique based on linear systems. When we linearize the system around a working point, the LQG-controller for this linearized system can be designed. If we use this LQG-controller in the nonlinear system, as described in section 6.1, we don't know how the nonlinearities will influence the stability. But if the nonlinearities are small, we may expect that the linearized model is a sufficient approximation of the real system to stabilize it.

Consider the following nonlinear realization of the MIMO process:

$$\dot{\underline{x}} = F(\underline{x},\underline{u},t)$$

$$\underline{x}(0) = \underline{x}_0$$

with states:

$$\begin{aligned} x_1 &= x_t \;,\; x_1(0)=0 & x_4 &= L \;,\; x_4(0)=0 \\ x_2 &= \dot{x}_t \;,\; x_2(0)=0 & x_5 &= \theta \;,\; x_5(0)=0 \\ x_3 &= L \;,\; x_3(0)=1 & x_6 &= \dot{\theta} \;,\; x_6(0)=0 \end{aligned} \qquad (6.1)$$

and inputs:

$$u_1 = F_t \;,\; u_1(0)=0 \qquad u_2 = F_h \;,\; u_2(0)=m_1 g$$

We have linearized the system around a working point by calculating the gradients with respect to all states. This is done for the nonlinear equations (2.7), (2.8) and (2.9) derived in section 2.2. The state vector of the working point $\underline{x}_0$ is chosen zero for all states, except the length of the grab cable which is choosen to be 1. The input vector $\underline{u}_0$ of the working point is found by substitution of the vector $\underline{x}_0$ in the equations.

The linear state space description holds:

$$\dot{\underline{x}} = A\underline{x} + B\underline{u}$$
$$\underline{y} = C\underline{x}$$

with:

$$A = \nabla_{\underline{x}} F|_{\underline{x}_0,\underline{u}_0} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{d_t}{m_t} & 0 & 0 & \frac{m_l g}{m_t} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{d_l}{m_l} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{d_t}{m_t} & 0 & 0 & -\frac{g}{m_t}(m_t+m_l) & 0 \end{bmatrix}$$

$$B = \nabla_{\underline{u}} F|_{\underline{x}_0,\underline{u}_0} = \begin{bmatrix} 0 & 0 \\ \frac{1}{m_t} & 0 \\ 0 & 0 \\ 0 & -\frac{1}{m_l} \\ 0 & 0 \\ -\frac{1}{m_t} & 0 \end{bmatrix} \qquad (6.2)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In this formula the initial values are substituted. For instance $L=L(0)=1$ and $F_h=F_h(0)=m_lg$.

6.2.2. Observability and controllability

Since we would like to stabilize the system by feeding back the states, we have to make sure that all states are completely controllable. This means that, for any initial state, the system can be brought to any final state in finite time by an appropriate control. When the controllability matrix

$$\Delta = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \qquad (6.3)$$

with: n = order of the system

has rank n, the system is controllable.

When we need an observer to estimate the states, the system must also be completely observable. Then the state vector at any time can be uniquely recovered from known inputs and outputs. Therefore the observability matrix

$$\Gamma = \begin{bmatrix} C & CA & CA^2 & \dots & CA^{n-1} \end{bmatrix} \qquad (6.4)$$

must have rank n.

It can be verified that the linearized sixth order system under study, described in section 6.2.1, is both observable and controllable.

6.2.3. LQG-controller

We can design a Linear Quadratic regulator for the linearized realization described before. This means that we can obtain a finite energy input as $\underline{u}(.)=L\underline{x}(.)$ such that:

$$\dot{\underline{x}}(t) = (A-BL)\underline{x}(t) \qquad (6.5)$$

By choosing L suitably, we can make $\underline{x}(.)$ decay to zero as fast as we wish. We should try to make a trade-off between the rate of decay of $\underline{x}(.)$, dependent on how negative the real parts of the eigenvalues of (A-BL) are, and the energy of the input. This is done by choosing $\underline{u}(.)$ to minimize [5, p.114]:

$$J = \int_{t_0}^{\infty} \frac{1}{2} \left[\underline{x}^T(t)Q\underline{x}(t) + \underline{u}^T(t)R\underline{u}(t) \right] dt \qquad (6.6)$$

By choosing Q and R we can give different weightings to the cost of control. Both Q and R are symmetric and positive definite. By solving the Algebraic Riccati Equation [5, p.118]:

$$0 = PA + A^TP - PBR^{-1}B^TP + Q \qquad (6.7)$$

we derive the unique positive definite solution P. With the solution matrix P we can compute the L-matrix:

$$L = R^{-1}B^{T}P \tag{6.8}$$

After trying some weighting matrices, we determined the following suitable Q- and R-matrices:

$$Q = \begin{bmatrix} 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.9}$$

With the help of the MATLAB CONTROL TOOLBOX the following L-matrix is derived:

$$L = \begin{bmatrix} 7.0711 & 8.5159 & 0 & 0 & 1.7415 & -0.6329 \\ 0 & 0 & -7.0711 & -4.7131 & 0 & 0 \end{bmatrix} \tag{6.10}$$

The eigenvalues in the s-plane for the three outputs are all stable:

$$E = \begin{Bmatrix} -1.5710 \pm 1.4986\,j \\ -0.8256 \pm 0.8514\,j \\ -0.3182 \pm 3.7384\,j \end{Bmatrix} \tag{6.11}$$

All states are available in the simulation model, so we don't need an observer. The states are fed back with the L-parameters (6.10). Now, the system is stabilized. With random, normal distributed white inputs we can obtain a dataset to use in neural identification. The actual values of the other parameters in the simulation are listed below.

$m_1 = 1.5\ kg$	$L_{max} = 2\ m$
$m_t = 3.5\ kg$	$L_0 = 1\ m$
$d_1 = .01$	$T_s = 0.05\ s$
$d_t = 0.1$	$N = 2000\ samples$

If we call the feedback signal $\underline{z}$, see figure 6.1, then the power of $\underline{z}$ with respect to the input signal $\underline{u}^*$ of this dataset is acceptable:

driving: $$\frac{E\{z_t^2\}}{E\{F_t^2\}} = \frac{1.9004}{24.8467} = 0.0725 \approx 7\%$$

hoisting: $$\frac{E\{z_h^2\}}{E\{(F_h - m_t g)^2\}} = \frac{1.9984}{15.8158} = 0.1264 \approx 13\%$$

6.3. Neural identification

In this section a nonlinear MIMO simulation example is identified with the neural network. Then the results of neural identification on the stabilized MIMO data of the simulated gantry crane process are given.

6.3.1. Introduction

The identification with neural networks is done with the C-routine "MOD5MIMO.C", described in Appendix D. This routine is able to train the neural network with the static backpropagation algorithm as described in section 3.3. In fact we estimate an Equation Error model and use the trained network as an initial estimation for the Output Error estimation. This method is recommended by [1]. A remark on this method can be made. This Output Error estimation is not done with the real gradient because we are constraint by the static backpropagation. See also formula (3.6). As a result of this, the performance of the simulation will depend on how close the Equation Error model is to the real minimum.

6.3.2. Simulation example 1

Consider the following MIMO example:

$$\begin{aligned} y_1(k) &= \frac{0.5z^{-1}}{1-0.5z^{-1}+0.5z^{-2}}\, u_1(k) + \frac{1}{1-0.7z^{-1}}\, u_2(k) \\ y_2(k) &= \frac{1}{1-0.8z^{-1}}\, u_1(k) + \frac{0.4z^{-1}}{1-0.7z^{-1}+0.2z^{-2}}\, u_2(k) \end{aligned} \qquad (6.12)$$

This system has stable poles which are well damped. It is linear and the two outputs are independent of each other when the inputs are independent. Both inputs are random, normally distributed white signals with a standard deviation of 1. The simulated outputs contain no noise.

To identify this MIMO process a neural network is trained. The neural network has the structure $N_{8\ 15\ 2}$. The eight inputs of the neural network are $u_1(k)$, $u_2(k)$, $u_1(k-1)$, $u_2(k-1)$,

$y_1(k-1)$, $y_2(k-1)$, $y_1(k-2)$, $y_2(k-2)$. The two estimated outputs of the neural network are $y_1(k)$ and $y_2(k)$. The estimation is done with the Equation Error method (see section 3.2). The batch size is 1998, that is as long as the length of the data set (2000 samples) minus the maximum delay (2). After 100 iterations and a constant step size of 0.1, we obtain the following Output Error performance, calculated following equation (3.7):

OE-performance:	0.1152 (output 1)
	0.0714 (output 2)

The simulation with the neural network on the training set is shown for both outputs in figure 6.4.

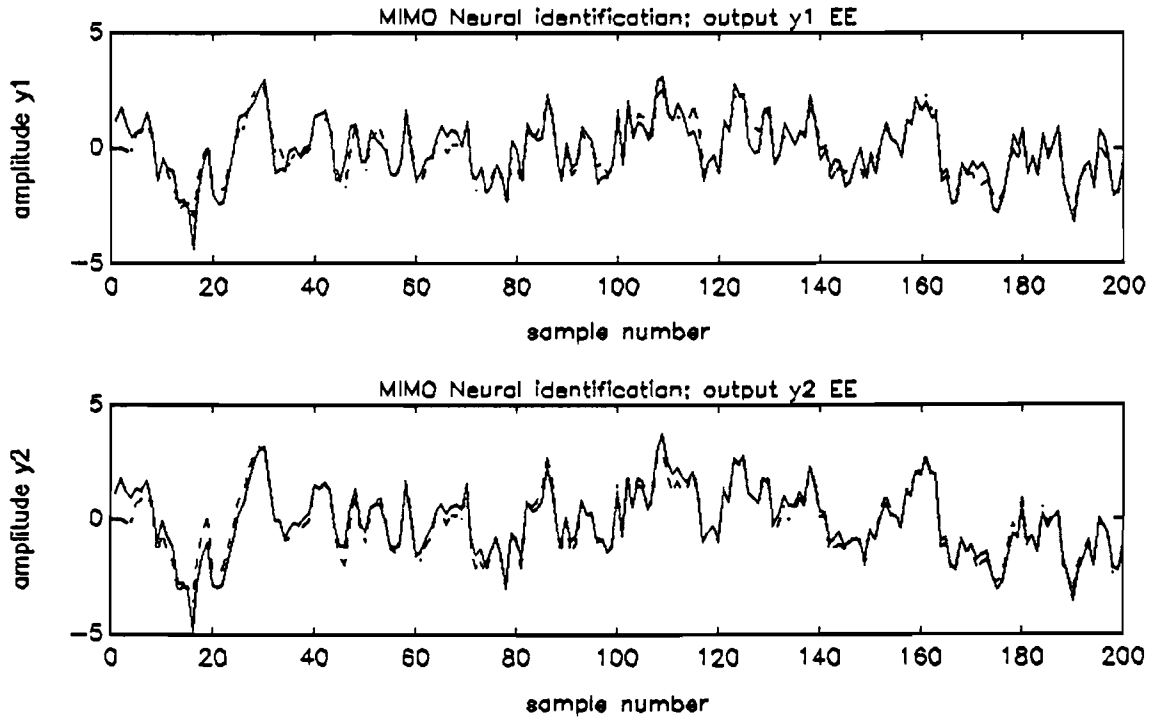


Fig.6.4. Simulation example 1

The solid lines indicate the outputs of the process, the dashed lines the neural estimations. With this example it is shown, that the neural network can be trained to identify a MIMO system. In this case the Equation Error model is a good approximation of the Output Error model, because there is no noise and the system is in the model set.

6.3.3. Simulation gantry crane without feedback

We first try to identify the system without feedback. This is hard (see also chapter 5)

because of the integration terms. So we decided to make the integration explicit and identify the process from the inputs (the pulling force F_t and the hoisting force F_h) to velocities $\dot{x}_t$, $\dot{L}$ and $\dot{\theta}$. These velocities are available in the MIMO simulation program, so we don't have to differentiate signals as in section 5. The inputs of the neural network are $u_1(k)$, $u_1(k-1)$, $u_2(k)$, $u_2(k-1)$, $y_1(k-1)$, $y_2(k-1)$ and $y_3(k-1)$. Thus the neural network has 7 inputs and 3 outputs. The estimation is done with the neural network structure $N_{7\ 15\ 3}$ and the step size is adjusted by hand. The simulation of the neural network estimation on the training set is given in figure 6.5.

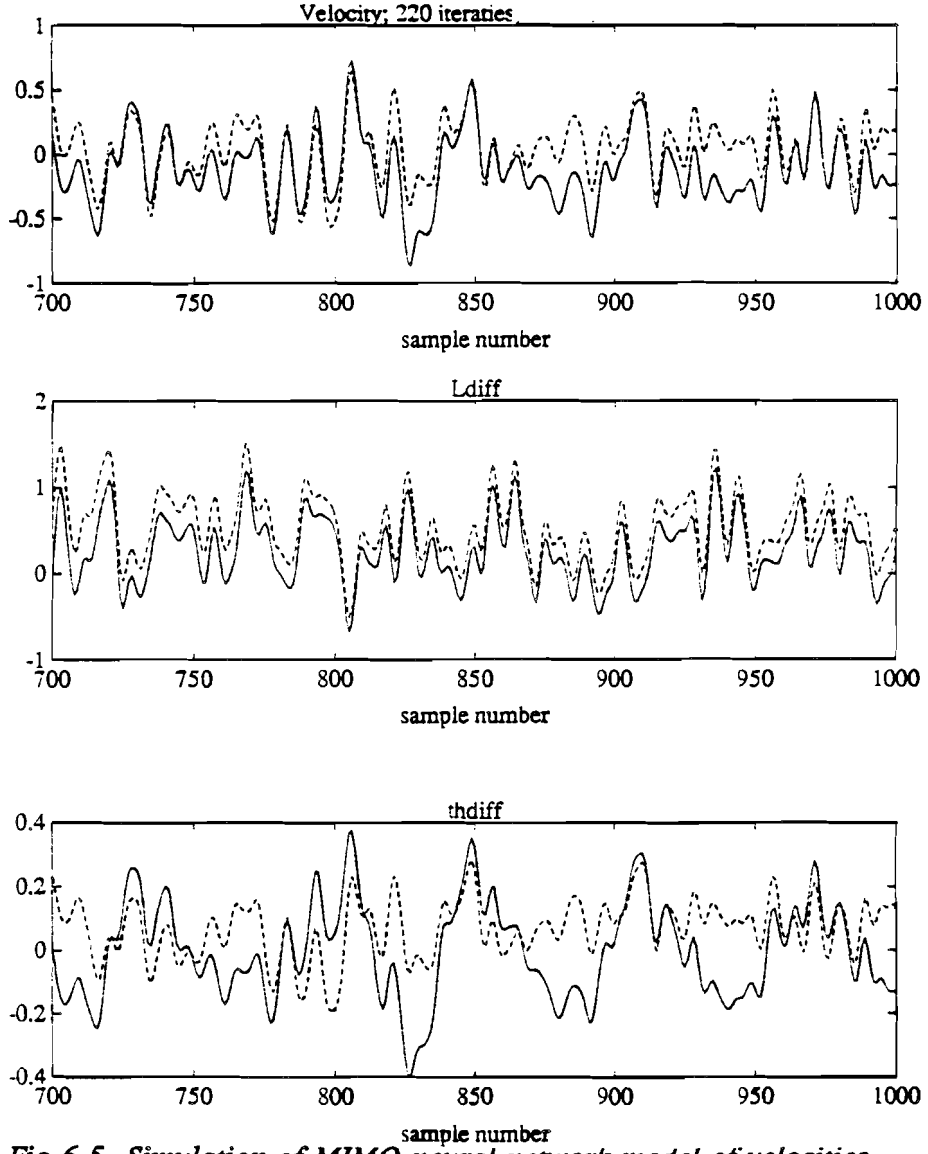


Fig.6.5. Simulation of MIMO neural network model of velocities

The dashed lines indicates the estimation and the solid lines the real outputs. It can be seen that the high frequent behaviour is almost the same as the real signal, but that the low frequent behaviour is bad, especially when we look at the third estimation of thetadot.

May be that more iterations, more inputs and automatical adjustment of step sizes can improve this estimation. But in fact this makes no sense, because

- in practice we are not able to measure all the velocities (such as thetadot)
- the cable lenth velocity and the velocity of the trolley are also dependent on the cable length L and the angle θ itself. In the configuration described in this section, that information is not explicitly available. So we decide to stabilize the system with a feedback of the states.

6.3.4. Simulation example gantry crane

The MIMO dataset, stabilized with feedback of the states (see section 5.2) is used to identify with the neural network model. The transfer function from inputvector $\underline{u}$ (forces minus feedback signals) to outputvector $\underline{y}$ (position, cable length, angle) will be identified, see figure 6.1. The dataset is normalized such that the signals have a standard deviation of 1. The batch size is taken 1998 (2000 samples minus the maximum delay). The neural network structure is $N_{10\ 15\ 3}$. Thus the number of parameters of the network is 213 (formula (3.2)). The ten inputs of the neural network are $u_1(k)$, $u_2(k)$, $u_1(k-1)$, $u_2(k-1)$, $y_1(k-1)$, $y_2(k-1)$, $y_3(k-1)$, $y_1(k-2)$, $y_2(k-2)$, $y_3(k-2)$. (The number of inputs is insufficient, we need more "zeros" in the neural network model). The inputs correspond to the driving force and the hoisting force and the outputs correspond to respectively the position of the trolley, the length of the grab cable and the angle deviation of the grab cable. The results of the estimation after 50 iterations and a step size of .005 is given in figure 6.6.

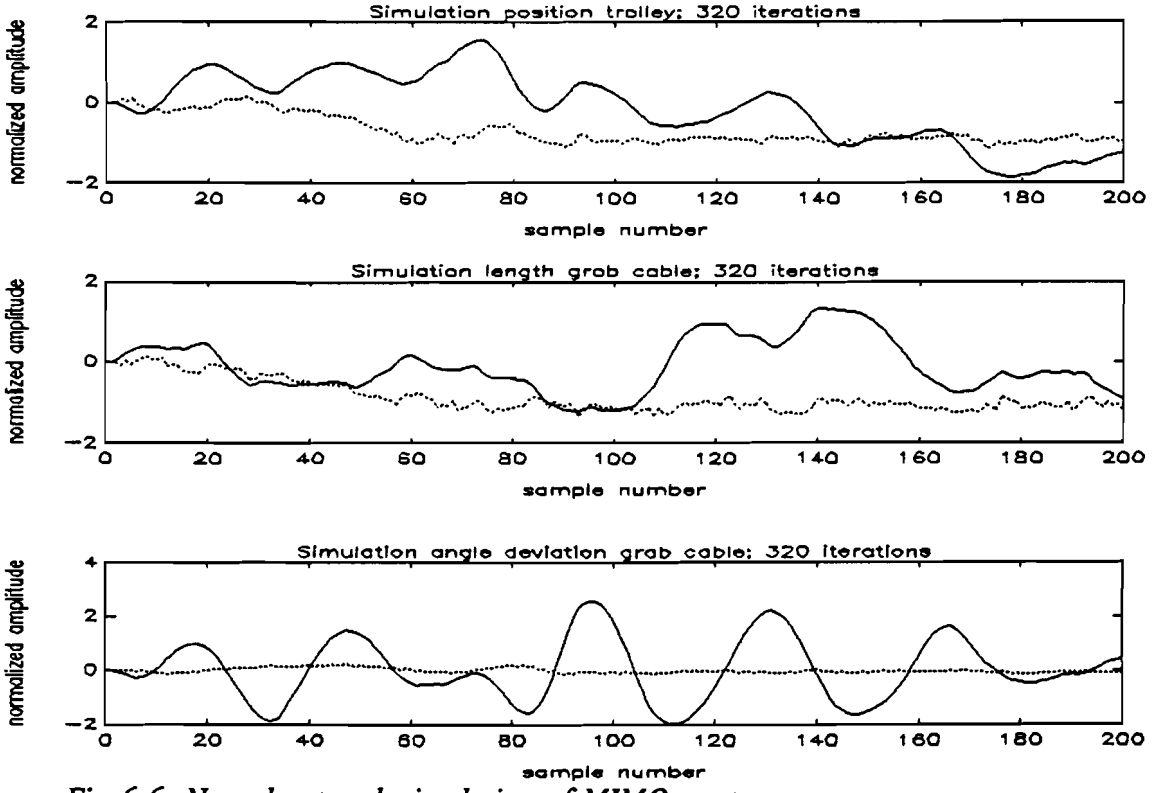


Fig.6.6. Neural network simulation of MIMO gantry crane process

As can be seen in the figure, the results are very poor. More iterations or other step sizes did not improve the Output Error performance.

With the NEURAL NETWORK TOOLBOX of MATLAB, we can also train the neural

network with the static backpropagation algorithm. The main advantage of this procedure is, that it adjusts the step size automatically. The step size increases exponentially and slowly as long as the performance of the estimation is decreasing. If the performance increases, the step size is taken half of the previous step size. In figure 6.7 the simualation results after 200 iterations are given of the estimation with the NEURAL NETWORK TOOLBOX. The network structure is also $N_{10\ 15\ 3}$ with the same inputs and outputs as above described. The result remains not satisfactory, although the step size is adjusted automatically.

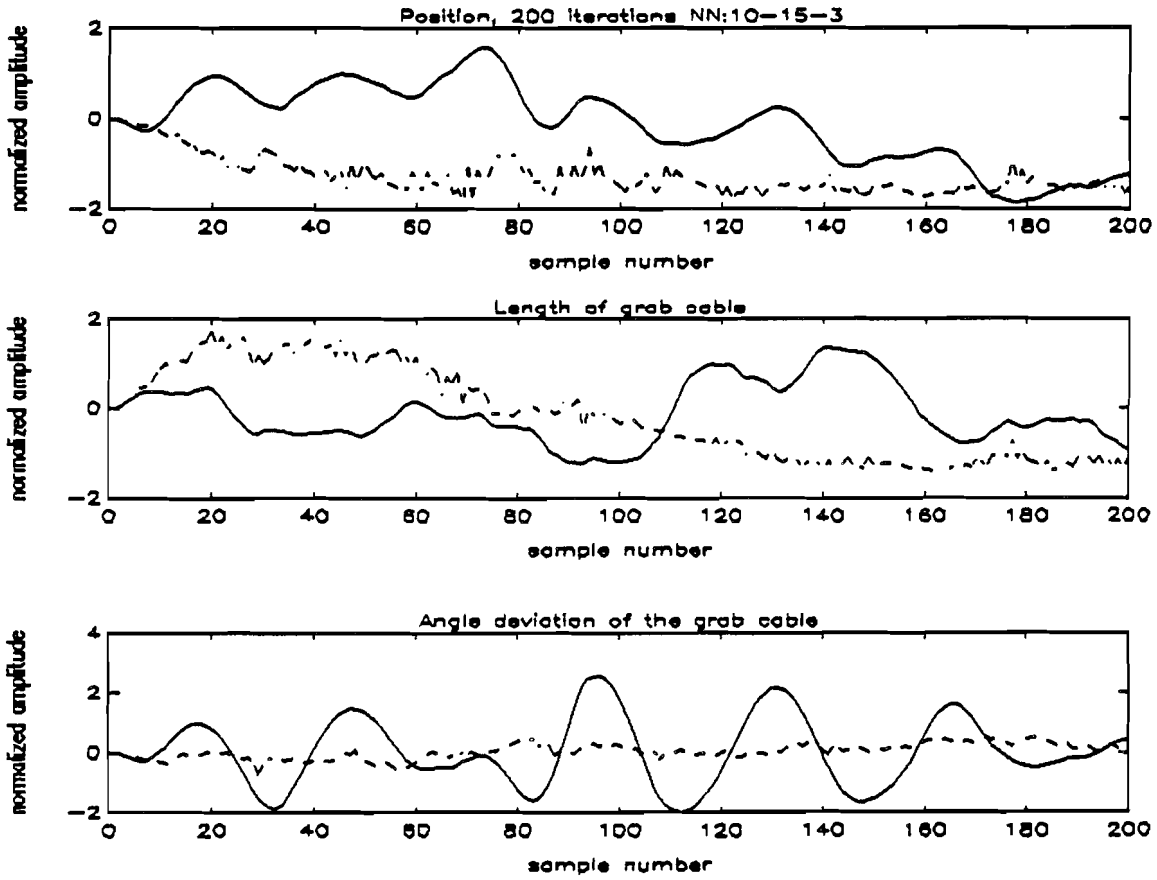


Fig.6.7. Neural network toolbox estimation

In figure 6.8 the prediction is given. It is almost exact.

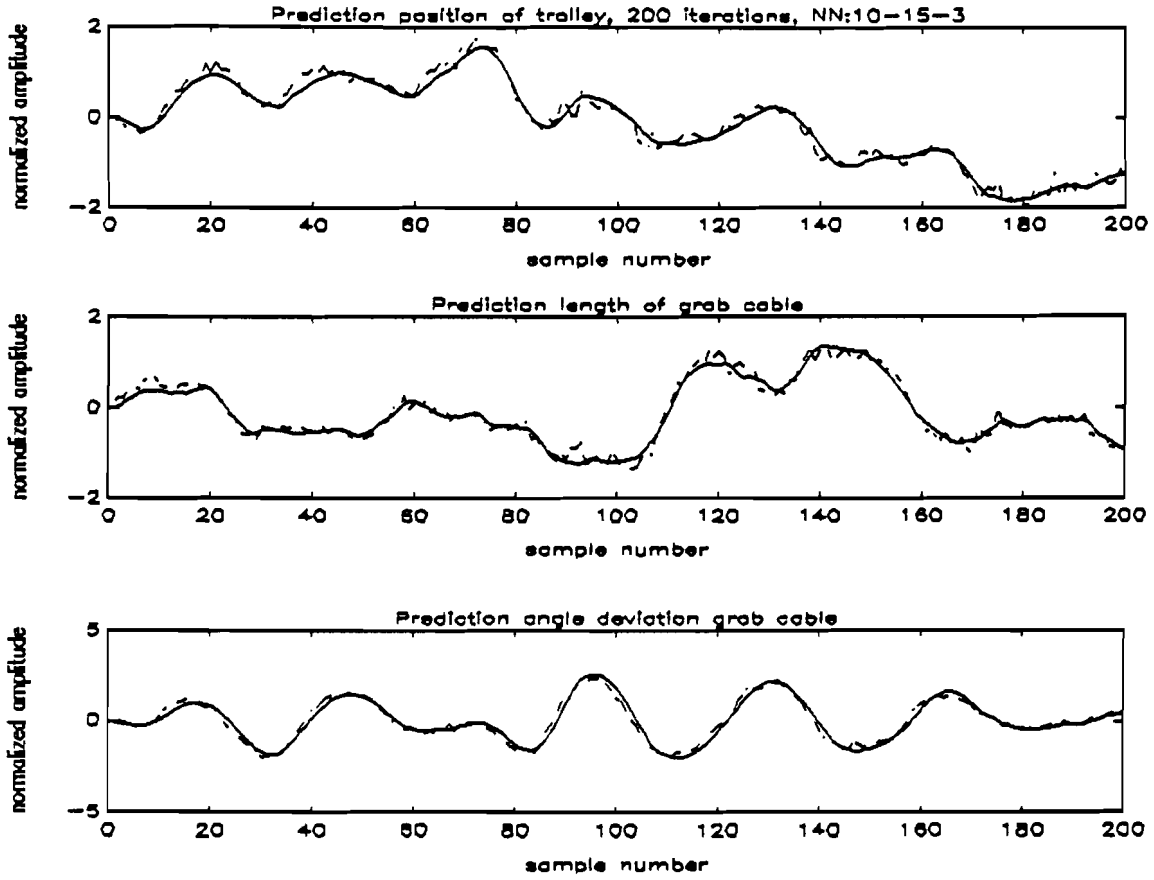


Fig.6.8. Neural network toolbox prediction

6.3.5. Conclusions

It is shown by a linear example that the static backpropagation algorithm is able to train the neural network to identify MIMO systems when there is no noise included and when the system is in the model set.

Estimation of the velocities of position, cable length and angle deviation is not a fine estimation scheme, because the velocities are dependent on the cable length and the angle itself. Therefore this information is needed as inputs to the neural network.

We stabilized the system with feed back of the states. The static backpropagation algorithm is not able to train the network properly for the output error model. The prediction error model performs very good.

The simulation model contains some errors with respect to the real MIMO crane process as described in section 2, but the difference is not too large.

7. CONCLUSIONS AND RECOMMENDATIONS

In our study of the gantry crane process we distinguish two situations:
- The SISO process which is a pilot model of a gantry crane. The input is the input of the servosystem and the output is the position of the trolley on the rail, influenced by the swinging of the pendulum.
- The MIMO process, with a pulling force on the trolley and an hoisting force on the load. This process is simulated on the basis of a mechanical description. The three outputs considered here are the position of the trolley on the rail, the length of the grab cable and the angle deviation caused by the swinging of the load.

The static backpropagation algorithm is used to train the neural network to identify the dynamics of the process in both SISO and MIMO case.

For the SISO process we found that
- The static backpropagation learn algorithm is not able to train the neural network suitably to identify the position of the trolley. The reason for this is the low frequent behaviour because of the integration of the input signal.
- The integration has been made explicit, by estimating the velocity of the trolley. In this case the output error performance of the neural network model is better than the output error performance of the linear model for both training and validation. With more iterations, automatical adjustment of the step size, just as in the Neural Network Toolbox of MATLAB and a larger dataset may improve the neural network model. The influence of the swinging of the pendulum on the position of the trolley is little, because in this estimation there are not enough memory locations available for the neural network to model this effect. Thus the performance can also be improved by taking an higher order of the model.
- The angle deviation of the pendulum is difficult to estimate, because of the small damping. Therefore, the position of the load can not be identified with the neural network model, trained with the static backpropagation algorithm.

Suggestions:
- Stabilize the process with a feedback controller. Then the length of the dataset will not be limited by mechanical conditions.
- Use another learn algorithm, for instance dynamic backpropagation.

In the MIMO simulation model we stabilized the process by an LQG-controller based on the linearized model. We found that
- Despite of the feedback, the static backpropagation algorithm is not sufficient to train the neural network model to find the global minimum of the criterion function. That is, so far the neural network model performs very poor.
- The prediction is rather good, as can be expected.

Suggestion:
- Use also in the MIMO situation another learn algorithm, for example dynamic backpropagation.

REFERENCES

[1] Pijnenburg, J.L.C.M.
Nonlinear system identification using neural networks
Department of Electrical Engineering, Measurement and Control Section,
Eindhoven University of Technology, 1992
M.Sc. Thesis

[2] Narendra, K.S. and K. Parthasarathy
Identification and control of dynamical systems using neural networks
IEEE Trans. on Neural Networks, vol.1 (1990), no.1, p.4-27

[3] Su, H.T. and T.J. McAvoy
Identification of chemical processes using recurrent networks
American Control Conference, vol.3 (1991), p.2314-

[4] Ven, H.H. van de
Time-optimal Control of a Crane
Department of Electrical Engineering, Measurement and Control Section,
Eindhoven University of Technology, 1983
EUT Report 83-E-135
ISBN 90-6144-135-8

[5] Damen, A.A.H. and A.J.W. van den Boom
Toegepaste systeemanalyse
Eindhoven University of Technology, 1988
Collegediktaat nr.5669, 5P280

[6] Zhu, Y.C., A.C.P.M. Backx and P. Eykhoff
Multivariable process identification for Robust Control
Department of Electrical Engineering, Measurement and Control Section,
Eindhoven University of Technology, 1991
EUT Report 91-E-249
ISBN 90-6144-249-4

[7] Vloet, P.J.C.
De identificatie van een drijvend platform
Department of Electrical Engineering, Measurement and Control Section,
Eindhoven University of Technology, 1990
M.Sc. Thesis

[8] Yuan Zhen-Dong
Insight into neural network models for nonlinear system identification
Internal report Linköping University, Sweden January 1991.

[9] Salminen, R., A. Martinen and J. Virkkunen
Adaptive pole placement control of a pilot crane
Helsinki University of Technology, p.137-142

APPENDIX A. Derivation mechanical equations

Variables used:

x_l	x-position load
z_l	z-position load
x_t	x-position trolley
L	length grab cable
θ	angle swing load
m_t	mass trolley
m_l	mass load
T	tension grab cable
F_t	pulling force on trolley
F_h	hoisting force load
d_t	damping factor trolley
d_θ	damping angle swing
d_l	damping in pulley
g	constant of gravity

Co-ordinates load:

$$x_l = x_t + L\sin\theta \tag{1}$$

$$z_l = L\cos\theta \tag{2}$$

Force balance trolley:

$$\sum F = m_t\ddot{x}_t + d_t\dot{x}_t = F_t + T\sin\theta \tag{3}$$

Force balance load:

$$\text{x-direction:} \quad \sum F = m_l\ddot{x}_l + m_l d_\theta L\dot{\theta}\cos\theta = -T\sin\theta \tag{4}$$

$$\text{z-direction:} \quad \sum F = m_l\ddot{z}_l - m_l d_\theta L\dot{\theta}\sin\theta = -T\cos\theta + m_l g \tag{5}$$

Equation of motion with respect to hoisting:

$$\begin{aligned}\sum F &= m_h\left(\ddot{L}+\ddot{x}_t\sin\theta\right) + d_h\left(\dot{L}+\dot{x}_t\sin\theta\right) = \\ &= -F_h + F_{cf} + F_z\cos\theta = \\ &= -F_h + m_t\dot{\theta}^2L + m_tg\cos\theta\end{aligned} \tag{6}$$

Double differentiating equation (1):

$$\dot{x}_t = \dot{x}_t+\dot{L}\sin\theta+L\dot{\theta}\cos\theta \tag{7}$$

$$\ddot{x}_t = \ddot{x}_t+\ddot{L}\sin\theta+2\dot{L}\dot{\theta}\cos\theta+L\ddot{\theta}\cos\theta-L\dot{\theta}^2\sin\theta \tag{8}$$

Double differentiating equation (2):

$$\dot{z}_t = \dot{L}\cos\theta-L\dot{\theta}\sin\theta \tag{9}$$

$$\ddot{z}_t = \ddot{L}\cos\theta-2\dot{L}\dot{\theta}\sin\theta-L\ddot{\theta}\sin\theta-L\dot{\theta}^2\cos\theta \tag{10}$$

Substitution of equation (8) in (4):

$$m_h\left(\ddot{x}_t+\ddot{L}\sin\theta+2\dot{L}\dot{\theta}\cos\theta+L\ddot{\theta}\cos\theta-L\dot{\theta}^2\sin\theta\right)-m_td_\theta L\dot{\theta}\cos\theta = -T\sin\theta \tag{11}$$

Substitution of equation (10) in (5):

$$m_t\left(\ddot{L}\cos\theta-2\dot{L}\dot{\theta}\sin\theta-L\ddot{\theta}\sin\theta-L\dot{\theta}^2\cos\theta\right)-m_td_\theta L\dot{\theta}\sin\theta = -T\cos\theta+m_tg \tag{12}$$

Elimination of T from equation (11) and (12):

$$\begin{aligned}&m_h\left(\ddot{x}_t\cos\theta+\ddot{L}\sin\theta\cos\theta+2\dot{L}\dot{\theta}\cos^2\theta+L\ddot{\theta}\cos^2\theta-L\dot{\theta}^2\sin\theta\cos\theta\right)+m_td_\theta L\dot{\theta}\cos^2\theta \\ &= m_t\left(\ddot{L}\sin\theta\cos\theta-2\dot{L}\dot{\theta}\sin^2\theta-L\ddot{\theta}\sin^2\theta-L\dot{\theta}^2\sin\theta\cos\theta\right)-m_tg\sin\theta-m_td_\theta L\dot{\theta}\sin^2\theta \quad \Rightarrow \\ &\Rightarrow \quad \ddot{\theta} = \frac{1}{L}\left(-g\sin\theta-\ddot{x}_t\cos\theta-2\dot{L}\dot{\theta}\right)-d_\theta\dot{\theta}\end{aligned} \tag{13}$$

Apart from the damping term this equation is exactly the same as found in [9].

Substitution of equation (11) in (3):

$$m_t\ddot{x}_t+d_t\dot{x}_t = F_t-m_t\left(\ddot{x}_t+\ddot{L}\sin\theta+2\dot{L}\dot{\theta}\cos\theta+L\ddot{\theta}\cos\theta-L\dot{\theta}^2\sin\theta\right)-m_td_\theta L\dot{\theta}\cos\theta \quad \Rightarrow$$

$$\Rightarrow \quad (m_t+m_l)\ddot{x}_t = F_t-d_t\dot{x}_t-m_t\ddot{L}\sin\theta-2m_t\dot{L}\dot{\theta}\cos\theta-m_tL\ddot{\theta}\cos\theta+m_tL\dot{\theta}^2\sin\theta-m_td_\theta L\dot{\theta}\cos\theta \tag{14}$$

Substitution of equation (6) and (13) in (14):

$$\begin{aligned}(m_t+m_l)\ddot{x}_t &= F_t-d_t\dot{x}_t-\left[\,-F_h+m_t\dot{\theta}^2L+m_tg\cos\theta-d_t(\dot{L}+\dot{x}_t\sin\theta)-m_t\ddot{x}_t\sin\theta\,\right]\sin\theta - \\ &- 2m_t\dot{L}\dot{\theta}\cos\theta-m_t\cos\theta\left[\,-g\sin\theta-\ddot{x}_t\cos\theta-2\dot{L}\dot{\theta}-d_\theta L\dot{\theta}\,\right]+m_tL\dot{\theta}^2\sin\theta-m_td_\theta L\dot{\theta}\cos\theta = \\ &= F_t-d_t\dot{x}_t+F_h\sin\theta+d_t(\dot{L}+\dot{x}_t\sin\theta)\sin\theta+m_t\ddot{x}_t\end{aligned} \tag{15}$$

$$\Rightarrow$$

$$\ddot{x}_t = \frac{1}{m_t}\left[\,F_t+F_h\sin\theta+(d_t\sin^2\theta-d_t)\dot{x}_t+d_t\dot{L}\sin\theta\,\right]$$

Substitution of equation (15) in (13):

$$\begin{aligned}\ddot{\theta} &= \frac{1}{L}\left[\,-g\sin\theta-2\dot{L}\dot{\theta}-\frac{1}{m_t}\left[\,F_t\cos\theta+F_h\sin\theta\cos\theta+(d_t\sin^2\theta-d_t)\dot{x}_t\cos\theta+d_t\dot{L}\sin\theta\cos\theta\,\right]\right]-d_\theta\dot{\theta} = \\ &= \frac{1}{Lm_t}\left[\,-g\sin\theta-2m_t\dot{L}\dot{\theta}-F_t\cos\theta-(F_h+d_t\dot{L})\sin\theta\cos\theta-(d_t\sin^2\theta-d_t)\dot{x}_t\cos\theta\,\right]-d_\theta\dot{\theta}\end{aligned} \tag{16}$$

Substitution of equation (15) in (6):

$$\begin{aligned}&m_t\ddot{L}+\frac{m_t}{m_t}\left[\,F_t+F_h\sin\theta+(-d_t+d_t\sin^2\theta)\dot{x}_t+d_t\dot{L}\sin\theta\,\right]\sin\theta+d_t(\dot{L}+\dot{x}_t\sin\theta) = \\ &= -F_h+m_t\dot{\theta}^2L+m_tg\cos\theta\end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned}m_tm_t\ddot{L} = &-m_tF_t\sin\theta-(m_t\sin^2\theta+m_t)(F_h+d_t\dot{L}+d_t\dot{x}_t\sin\theta)+ \\ &+m_td_t\dot{x}_t\sin\theta+m_tm_t\dot{\theta}^2L+m_tm_tg\cos\theta\end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned}\ddot{L} = \frac{1}{m_tm_t}\Big[\,&-m_tF_t\sin\theta-(m_t\sin^2\theta+m_t)(F_h+d_t\dot{L}+d_t\dot{x}_t\sin\theta)+ \\ &+m_tm_t\dot{\theta}^2L+m_td_t\dot{x}_t\sin\theta+m_tm_tg\cos\theta\,\Big]\end{aligned} \tag{17}$$

Equations (15), (16) and (17) are the nonlinear equations of motion for the MIMO system.
In the SISO system, the following holds:

$$L = L = 0 \tag{18}$$

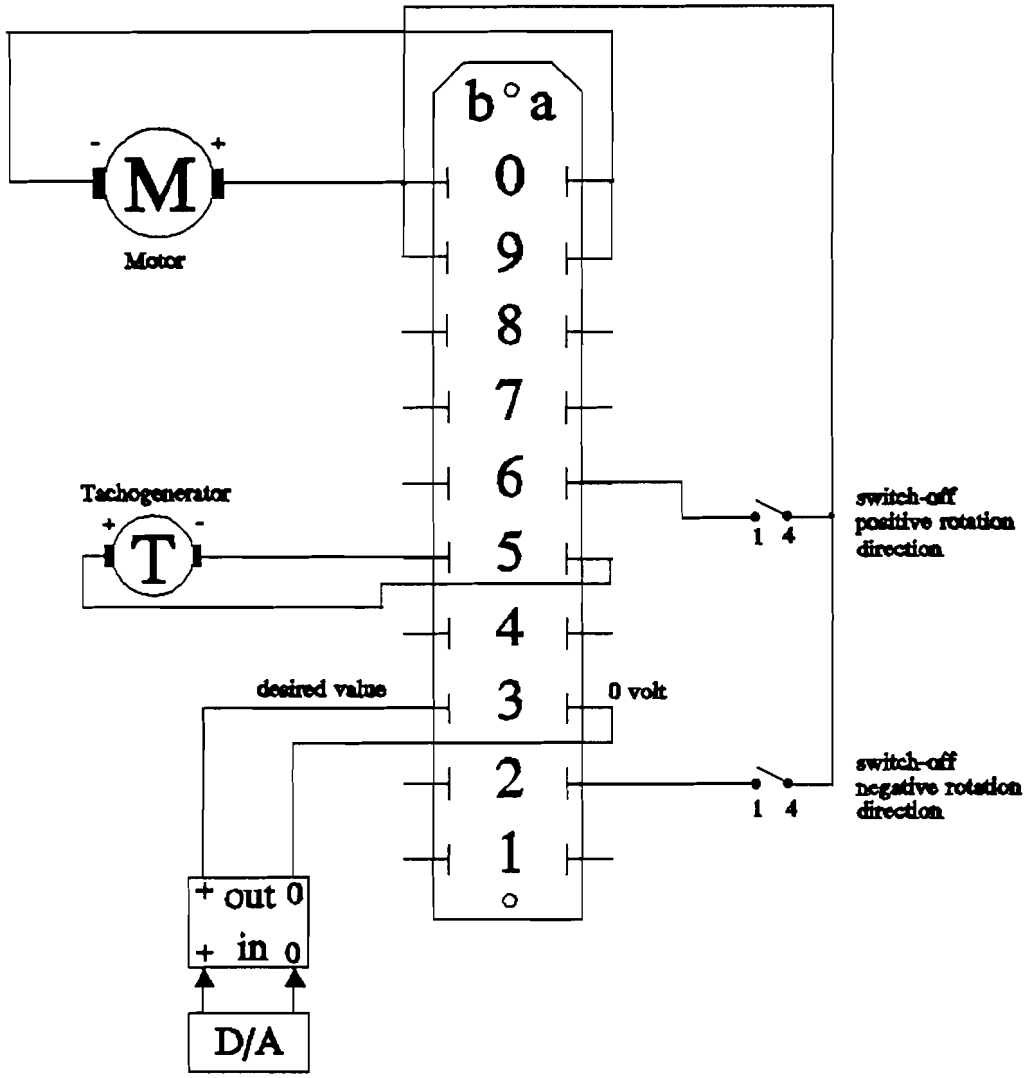
$$\ddot{x}_t = \frac{F_t - d_t\dot{x}_t + m_p g\sin\theta\cos\theta + m_p L\dot{\theta}^2\sin\theta}{m_t + m_p\sin^2\theta} \tag{19}$$

$$\ddot{\theta} = -d_\theta\dot{\theta} + \frac{-F_t\cos\theta - m_p L\dot{\theta}^2\sin\theta\cos\theta - (m_t + m_p)g\sin\theta + d_t\dot{x}_t\cos\theta}{L\left(m_t + m_p\sin^2\theta\right)} \tag{20}$$

Apart from the damping terms equations (19) and (20) are also found by Van de Ven [4] (the orientation of theta used here is different from that report!).

APPENDIX B. Connection diagram Servosystem

The servo amplifier (*Hauser Elektronik G.M.B.H.*™) is connected with the motor, the tachometer, the input (desired value) and two protection switches as follows. Datasheets can be found in [7].



APPENDIX C. Implementation in SIMULINK

The equations (2.7), (2.8) and (2.9) are implemented in SIMULINK in the continue time domain. This appendix is divided in five parts:

C1. Description of variables
C2. Realization subblock SUB1
C3. Realization subblock SUB2
C4. Overall block scheme of implementation

C1. Description of variables

Input variables:

	Description	**Offset**
F_t	driving force on trolley. 2 columns: first column: time vector second column: input samples	0
F_h	hoisting force on load. 2 columns: first column: time vector second column: input samples	$m_l g$

Output variables:

	Description	**offset**	**upper bound limited integrator**	**lower bound limited integrator**
x	position trolley	0	100	-100
xdot	velocity trolley	0	10	-10
L	length grab cable	$.5*L_{max}$	$.49*L_{max}$	$.49*L_{max}$
Ldot	velocity grab cable	0	10	-10
theta	angle deviation of grab cable	0	$.5\pi$	$-.5\pi$
thetadot	angle velocity	0	10	-10
u1	input signal system after feedback w.r.t. driving			

u2	input signal system after feedback w.r.t. hoisting			

Constants:

	Description	Simulation value	Dimension
ml	total mass of load	1.5	kg
mt	total mass of trolley	3.5	kg
dl	damping cable velocity	0.1	m/s
dt	damping velocity trolley	0.1	m/s
dθ	damping angle velocity	0.01	rad/s
Lmax	maximum cable length	2	m
g	constant of gravity	9.8	m/s^2
x0	initial position trolley	0	m
xdot0	initial velocity trolley	0	m/s
L0	initial cable length (minus 1)	0	m
Ldot0	initial velocity cable	0	m/s
th0	initial angle	0	rad
thdot0	initial angle velocity	0	rad/s
k	2x6 matrix which defines the parameters of the LQG-controller	see chapter 6	

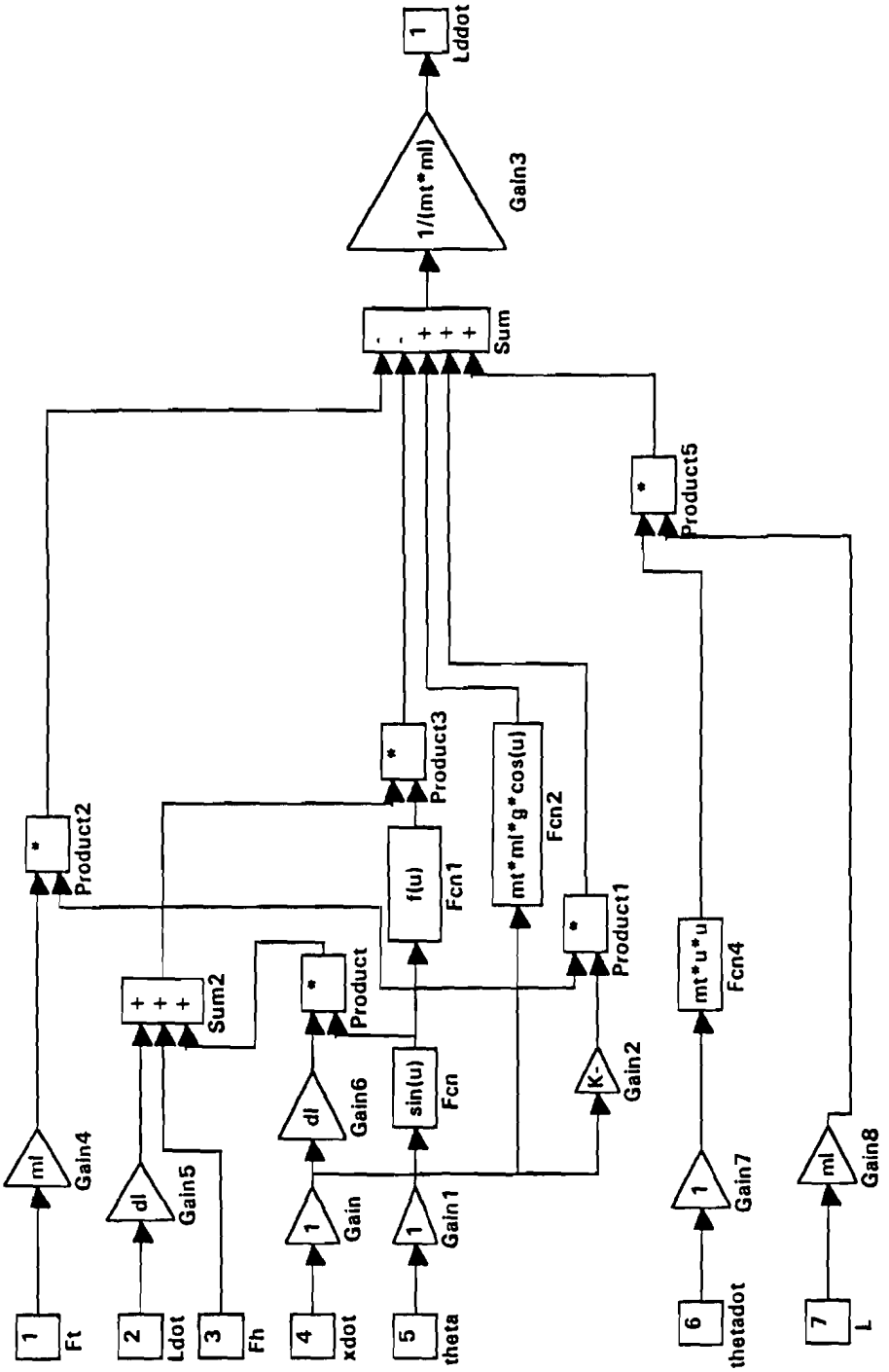
The MATLAB window is named "workspace". The input variables and constants must have a value in the workspace before starting the simulation. The output variables are in the workspace after the simulation. The k-matrix is arranged as follows:

$$\begin{array}{c} \\ F_t \\ F_h \end{array} \begin{array}{c} \begin{array}{cccccc} x & xdot & L & Ldot & \theta & thetadot \end{array} \\ \begin{bmatrix} k(1,1) & k(1,2) & 0 & 0 & k(1,5) & k(1,6) \\ 0 & 0 & k(2,3) & k(2,4) & 0 & 0 \end{bmatrix} \end{array}$$

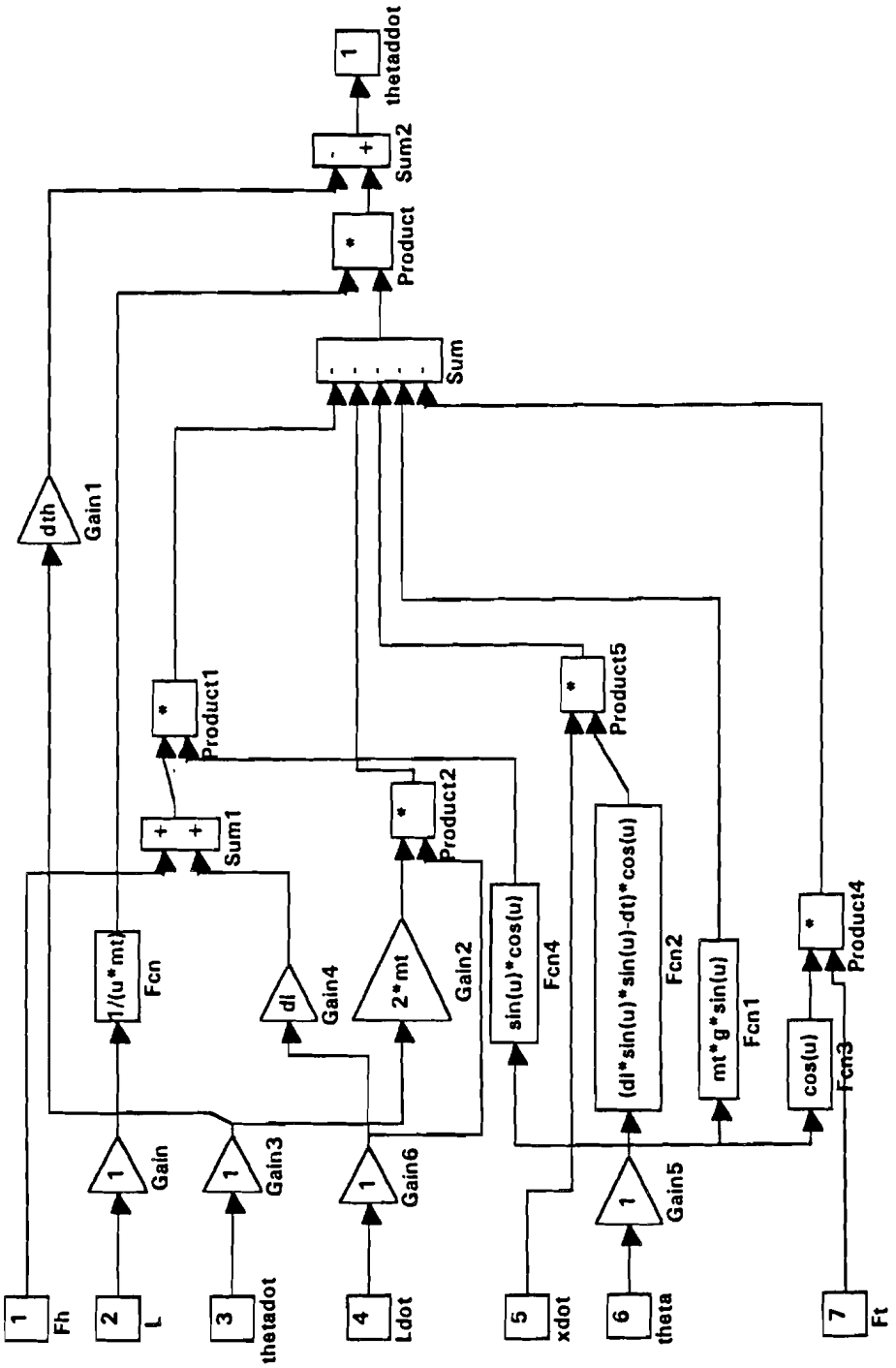
The three equations (3.8) for xddot, (3.9) for thetaddot and (3.10) for Lddot are realized by the subblocks SUB3, SUB2 and SUB1 respectively. In the realizations on the following pages, the input variables are numbered input blocks. In subblock SUB1 for example is theta the fifth input.

Remark: the limits of the limited integrators are set very high, because we are not interested in the physical limitations of the system, but rather in the dynamic behaviour.

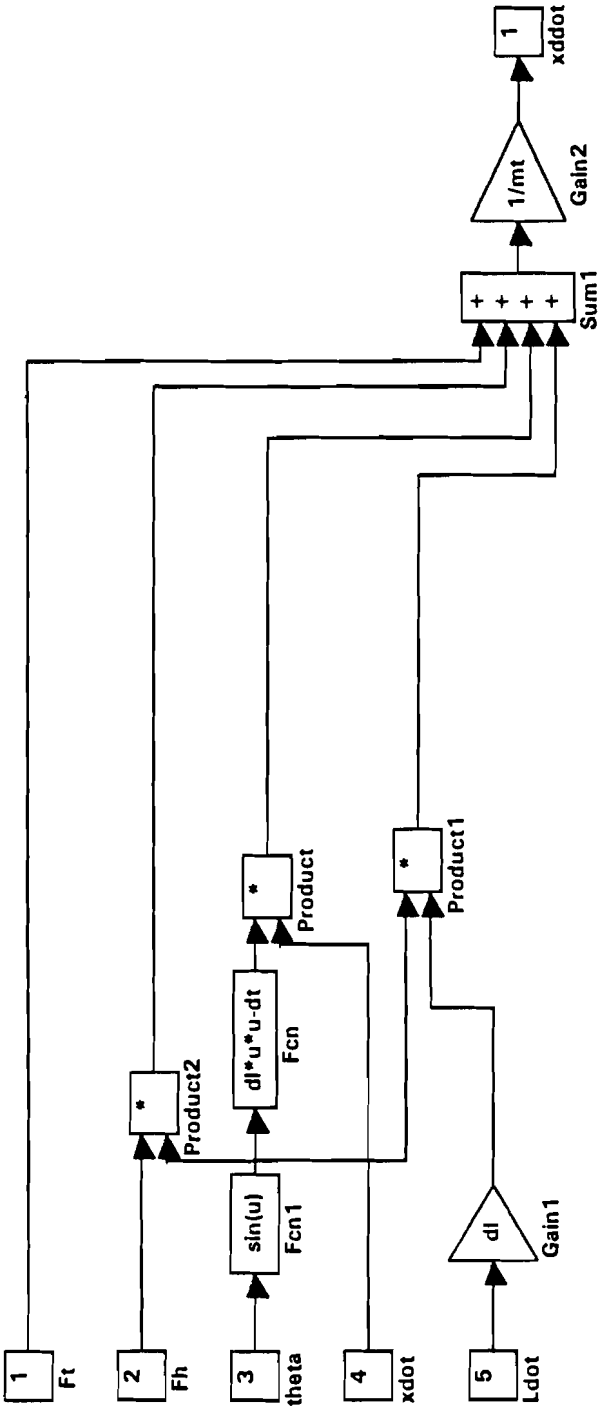
C1. Realization subblock SUB1



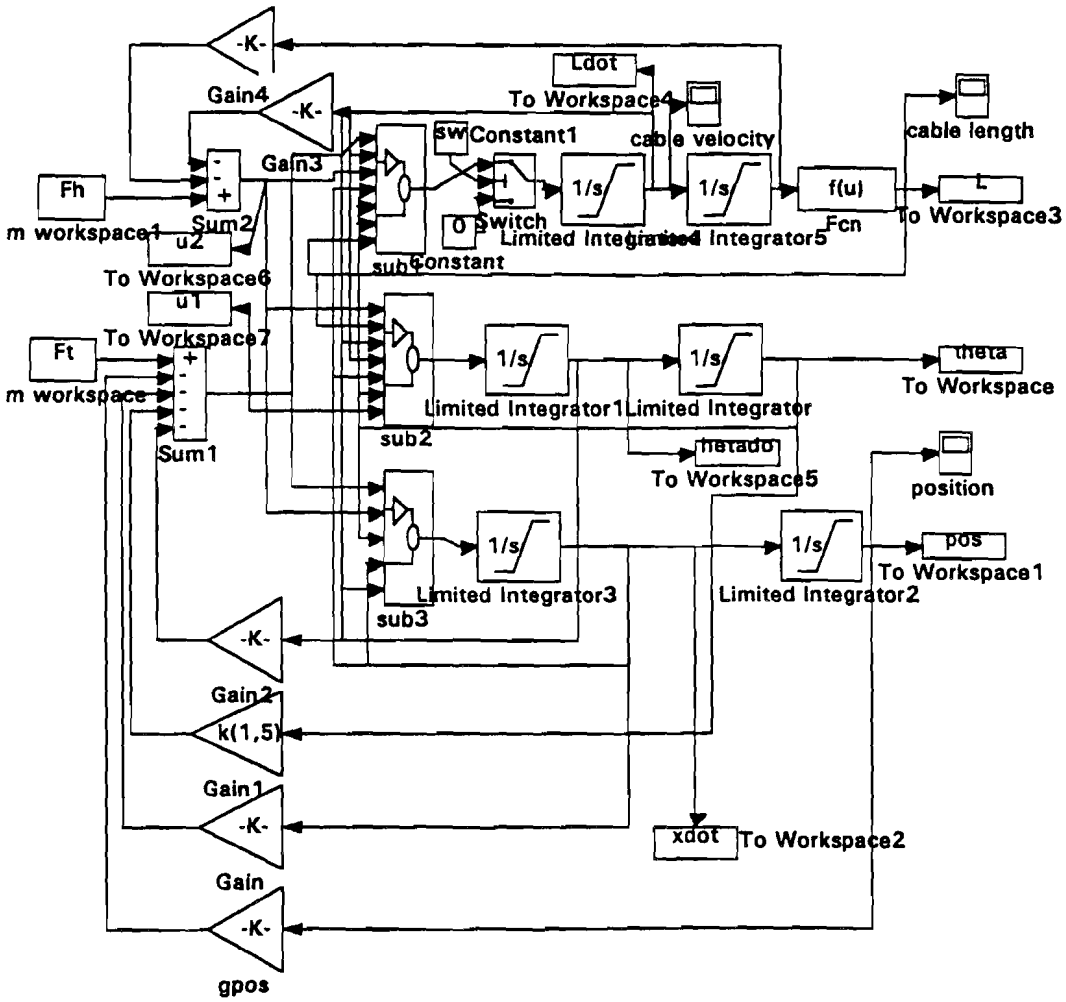
C2. Realization subblock SUB2



C3. Realization subblock SUB3



C4. Overall block scheme of implementation



APPENDIX D. Software description (MIMO)

This appendix holds the following parts:

D1. Compilation of C-routines
D2. Description of the MIMO program to train the neural network
D3. Description of the MIMO program to simulate with the neural network

D1. Compilation of C-routines

Because the C-routines used here are linked with MATLAB, the compilation differs from the normal compilation procedure. To compilate the C-routines on the VAX/VMS machine the following steps must be executed successively:

1. Log in into MATLAB and exit matlab, otherwise the MATLAB-related files are not known:

```
$ matlab
>> exit
```

2. Compile the C-routine and link to MATLAB:

```
$ cc <filename>
$ cmexd <filename>
$ clinkd <filename>
```

The following files are made:

```
<filename>.lis
<filename>.mexd
<filename>.obj
<filename>.opt
```

If the VAX-debugger is used, the compilation procedure is different from the above described procedure. First a special code must be used in the C-routine to invoke the debugger. This code must be placed where the program has to enter the debugger:

```
# include libdef
# define SS$_debug 1132
  lib$signal (SS$_debug);
```

The file "CLINKD.COM" must exist in the current directory. Compile with the following commands:

```
$ cc/debug/noopt <filename>
$ @clinkd <filename>
```

D2. Description of the MIMO program to train the neural network

The calling syntax of the MIMO program "MOD5MIMO.C" is:

```
[A_new, B_new, W1_new, W2_new, J] =
= MOD5MIMO(x,k,d_max,connect, A_old,B_old,W1_old,W2_old,i1,i2,L1,L2,
                alpha_ARMA,alpha_N,beta,batch,flag,pf,fixed);
```

This program trains the neural network following the backpropagation algorithm. The structure is lent from the so called "model5", described in [1]. In this model, a linear ARMA-model and two neural networks can estimate the input-output relation parallel to each other.

The maximum values, defined as constants in the program are:

Constant	**Description**	**Actual value**
AMax	maximum number of delays of output plus 1	3
BMax	maximum number of delays of input plus 1	AMax
MaxOut	maximum number of outputs of the MIMO system	5
MaxIn	maximum number of inputs of the MIMO system	MaxOut
NMax	the maximum of the maximum number of processing elements in one of the layers or of the neural network inputs	20
LMax	maximum number of hidden layers	3
bias	constant into the processing elements	1

Input parameters:

Parameter	rows	columns	Description
x	MaxIN+ MaxOut	k	Rows 1..MaxIn contain the input learn patterns $u_1..u_{MaxIn}$. Rows MaxIn+1..MaxOut contain the output learn patterns $y_1..y_{MaxOut}$. If the input is not used, the row must contain zeros.
k	1	1	number of samples
d_max	1	1	model order
connect	3	(MaxIn + MaxOut)* (d_max+ 1)	matrix which defines the model structure. See example below

Example connect matrix:

The connect matrix indicates whether an input or output is connected or not. The matrix consists of ones (connected) and zeros (not connected). The matrix has three rows:

First row: Defines inputs and outputs of ARMA model
Second row: Defines inputs and outputs of Neural Network 1
Third row: Defines inputs and outputs of Neural Network 2

Suppose:

MaxOut=MaxIn=3
d_max=2
The MIMO system has 2 inputs and 2 outputs and we use only
$u_1(k)$, $u_2(k)$, $u_1(k-1)$, $y_1(k-1)$, $y_2(k-1)$, $y_1(k-2)$.

The connect matrix will be:

	u_1	u_2	u_3	y_1	y_2	y_3												
ARMA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NN1	1	1	0	0	0	0	1	0	0	1	1	0	0	0	0	1	0	0
NN2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				delay 0						delay 1						delay 2		

In this example the input $u_3(k)$ and the output $y_3(k)$ don't exist and therefore is the corresponding column filled with zeros. Also the ARMA model and the Neural Network NN2 is not used.

Continuation of input parameters:

Parameter	Rows	Columns	Description
A_old	MaxOut	AMax*MaxOut	contains the AR-parameters of the ARMA-model (MIMO). See example below for representation
B_old	MaxIn	BMax*MaxOut	contains the MA-parameters of the ARMA-model (MIMO). See example below for representation
W1_old	NMax+1	LMax*(NMax+1)	contains the weighting factors of the neural network NN1. See example below
W2_old	NMax+1	LMax*(NMax+1)	contains the weighting factors of the neural network NN2. Same representation as W1_old

Example representation ARMA-parameters:
Consider a MIMO system with two inputs and two outputs.
MaxIn=MaxOut=2
d_max=2
AMax=3
BMax=3

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & x & x & x & x \\ 0 & 0 & x & x & x & x \end{bmatrix}}_{A_{old}} \begin{bmatrix} y_1(k) \\ y_2(k) \\ y_1(k-1) \\ y_2(k-1) \\ y_1(k-2) \\ y_2(k-2) \end{bmatrix} + \underbrace{\begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \end{bmatrix}}_{B_{old}} \begin{bmatrix} u_1(k) \\ u_2(k) \\ u_1(k-1) \\ u_2(k-1) \\ u_1(k-2) \\ u_2(k-2) \end{bmatrix}$$

where: x = value of parameter
0 = zero

Example representation weighting factors:
Consider a neural network with two hidden layers of five nodes each, with just one input and one output. The structure can be denoted by $N_{1\,5\,5\,1}$.
NMax=5
LMax=3

$$Wl\_old = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \vdots & 0 & 0 & 0 & 0 & 0 & 0 & \vdots & 0 & 0 & 0 & 0 & 0 & 0 \\ a & c & 0 & 0 & 0 & 0 & \vdots & d & e & * & * & * & * & \vdots & g & h & * & * & * & * \\ b & * & 0 & 0 & 0 & 0 & \vdots & * & * & * & * & * & * & \vdots & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & 0 & 0 & 0 & 0 & \vdots & * & * & * & * & * & * & \vdots & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & 0 & 0 & 0 & 0 & \vdots & * & * & * & * & * & * & \vdots & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & 0 & 0 & 0 & 0 & \vdots & * & * & * & * & * & * & \vdots & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where: 0 = zero
* = weighting factors
a = from bias 1 to node 1 of layer 1
b = from bias 1 to node 2 of layer 1
c = from input 1 to node 1 of layer 1
d = from bias 2 to node 1 of layer 2
e = from node 1 of layer 1 to node 1 of layer 2
f = from node 2 of layer 1 to node 3 of layer 2
g = from bias 3 to node 1 of layer 3 (=output)
h = from node 1 of layer 2 to node 1 of layer 3 (=output)

The matrix is divided in three parts by the vertical dots. The first part agree with the weighting factors of the input layer to the first hidden layer. The second part agree with the weighting factors from the first hidden layer to the second hidden layer. The third part agree with the weighting factors from the second hidden layer to the output layer. The parts are squared matrices of (NMax+1) by (NMax+1). Consider one part of the matrix. The columns in this part agree with the source nodes and the rows agree with the target nodes. The source nodes are the nodes in the previous layer. First comes the bias node, than the inputs 1 to NMax. Some examples are given in the formula above. The first row of the matrix is zero, because the bias constants have no inputs. In this example we considered only one input and one output. That's why the first parts has only two columns (from bias 1 and input 1) and the output layer has only one row (to output 1).

Continuation of input parameters:

Parameter	Size	Description
i1	column vector	defines the number of processing elements in each layer of the neural network NN1. **Example:** If the neural network has 10 inputs, 2 hidden layers with 5 nodes each, and 2 outputs, $i1=[10\ 5\ 5\ 2]^T$
i2	column vector	idem for neural network NN2
L1	scalar	number of layers in neural network NN1. In the example above, with 2 hidden layers, L1=3.
L2	scalar	idem for neural network NN2
alpha_ARMA	scalar	step size in steepest descent adjustment of the ARMA parameters
alpha_N	scalar	step size in steepest descent adjustment of the weighting factors of the neural network
beta	scalar	factor related to the momentum term [1]
batch	scalar	number of samples over which the gradient is calculated
flag	scalar	defines whether the Equation Error method (flag=0) or the Output Error method (flag=1) is used
pf	scalar	defines the kind of processing function: pf = 1 $f[x] = \frac{1}{1+e^{-x}}$ pf = 2 $f[x] = \tanh(x)$ pf = 3 $f[x] = \frac{1-e^{-x}}{1+e^{-x}}$
fixed	scalar	defines whether the ARMA parameters have to be adjusted simultanously (fixed=0) or not (fixed=1)

Output parameters:

Parameter	Rows	Columns	Description
A_new	MaxOut	AMax* MaxOut	contains the new AR-parameters of the ARMA-model. Same format as described under A_old
B_new	MaxIn	BMax*MaxIn	contains the new MA-parameters of the ARMA-model. Same format as described under B_old
W1_new	NMax+1	LMax*(NMax+1)	contains the estimated weighting factors of NN1
W2_new	NMax+1	LMax*(NMax+1)	idem for NN2
J	1	MaxOut	contains the Equation Error performance over the entire training phase for each output. Calculated as: $\frac{\sum_{k=1}^{N_s}(y(k)-\hat{y}_p(k))^2}{\sum_{k=1}^{N_s}(y(k))^2}$ where: N_s = number of samples; $\hat{y}_p(k)$ = predicted output; $y(k)$ = real output

D3. Description MIMO program to simulate with the neural network

The calling syntax of "OUT_MOD5MIMO" is:

[y_model] = OUT_MOD5MIMO(u,k,d_max,connect,A,B,W1,W2,i1,i2,L1,L2,pf);

This program simulates the outputs of the MIMO system for given inputs u. The estimated outputs are fed back as inputs to the neural network, as described in chapter 2. The initial values are set to zero, which may be a bad starting position for the simulation.

Constants:
The maximum values AMax, BMax, LMax, MaxOut, MaxIn, NMax and the constant bias factor must be the same as in "MOD5MIMO.C". One extra constant:

KMax: maximum number of input samples of the validation set.

Input parameters:
The input parameters k, d_max, connect, i1, i2, L1, L2 and pf must be the same as in "MOD5MIMO.C". For the parameters A, B, W1, W2 counts that this are the new estimated ARMA-parameters and weighting factors of the neural networks respectively. In fact, these matrices are the output parameters of "MOD5MIMO.C". One new parameter:

Parameter	Rows	Columns	Description
u	MaxIn	number of samples k	contains the input signals $u_1..u_{MaxIn}$ of the validation set

Output parameter:

Parameter	Rows	Columns	Description
y_model	MaxOut	KMax	each row contains an simulated output of the neural network with as many columns as samples in the input data set. The rest is filled with zeros.