UNIVERSITY OF SOUTHAMPTON

Application of Intelligent Signal Processing to Dynamic Measurement Systems

by

Seyed Mohammad Taghi Alhoseyni Almodarresi Yasin

A thesis submitted for the degree of

Doctor of Philosophy

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

March 2001

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE

DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

APPLICATION OF INTELLIGENT SIGNAL PROCESSING TO DYNAMIC MEASUREMENT SYSTEMS

by Seyed Mohammad Taghi Alhoseyni Almodarresi Yasin

A new method for dynamic measurement is presented. A feature extractor and two-layer artificial neural network is used to predict the final value of a sensor's response while it is still in oscillation. The method permits arbitrary inputs and initial conditions and does not make any assumptions about the model of the sensor. It also copes with non-linearity defects in primary sensors. Introducing a pre-processor as a feature extraction block before the neural network decreases the effect of noise and dramatically reduces the required number of neurones. This, in turn, reduces the complexity of computation and speeds up the real-time measurement. One important advantage of the proposed method is that it can be used in situations where the input function is an impulse, i.e. the transducer senses the measurand for only a very short time interval. This method also allows the possibility of using some features of the sensor signal, such as frequency, that are rarely used in other methods, despite them having a unique relation with the steady state value of the signal. Amplitude noise also has less effect on these characteristics. In addition dynamic neural networks are used in a novel way to cancel the interference signals. The proposed methods are established by theoretical analysis and justified by means of both simulation and measurements on real data.

Acknowledgments

I would like to express my thanks to Dr N M White for his guidance and supervision during the course of this research. I am also grateful for useful discussions and friendly atmosphere provided by the members of the research group.

Furthermore, I would like to thank the Ministry of Science, Research and Technology of Islamic Republic of Iran for financial support.

I would also like to thank my parents for providing me with inspiration, education and opportunities I needed to pursue an academic carrier.

The special thanks must go to my wife Mahboubeh, son Seyed Ali and daughter Maryam Alsadat whom although suffered seriously from homesickness over the last three years but patiently let me finish this project.

In the name of ALLAH

The beneficent, The merciful.

Table of Contents

List of Figures

List of Symbols and Terms

Symbol	Unit	Explanation
t	$[s]$	Time
$F(t)$	[-]	Stimulation function
$u(t)$	[-]	Unit step function
m	[kg]	Mass of a load
m_0	[kg]	Mass of a sensor
g	$[m/s^2]$	Gravity constant
f	[Hz]	Frequency
c	[-]	Damping factor
k	[-]	Spring constant
α	[-]	Damping ratio
ω_n	[rad/s]	Natural frequency
ω_0	[rad/s]	Resonance frequency
ω_d	[rad/s]	Damping frequency
V	[V]	Voltage
ANN	[-]	Artificial Neural Network
NN	[-]	Neural Network
MLP	[-]	Multi Layer Perceptron
RBF	[-]	Radial Basis Function
LMS	[-]	Least Mean Square

Symbol	Unit	Explanation
ISP	[-]	Intelligent Signal Processing
IIR	[-]	Infinite Impulse Response
FIR	[-]	Finite Impulse Response
PID	[-]	Proportional, Integral, Differential
DC	[-]	Direct Current

Chapter 1

Introduction

Dynamic measurement refers to the ascertainment of the final value of a sensor signal while its output is still in oscillation. It is used to speed up the process of measurement. Dynamic measurement methods are mainly signal processing procedures that are used to tackle the defects of elementary sensors.

Elementary sensors are one of the most important parts of any measurement system. Their main function is the conversion of a physical or chemical parameter to a measurable signal, usually in the electrical domain. Generally, the input domain of sensors are considered to be the following types: radiant, mechanical, thermal, magnetic and chemical. Five major defects [1] that influence dramatically the performance of raw sensors are:

1. *Cross sensitivity*

   Ideally, it is expected that a sensor will respond to just one type of physical parameter but this is not the case in many practical situations. For example, even if the pressure is constant, the output of a pressure sensor may change if the temperature fluctuates.

2. *Parameter drift*

   The nature of material in a primary sensor is never wholly time-invariant. It can slowly change because of different chemical process. Oxidation is a case in point. This in turn can cause the variation of offset and sensitivity of a primary sensor. Parameter drift points to this problem.

3. *Noise*

   Every unwanted signal that contaminates the desired signal is called noise. The source of some types of noise are external to the sensor. These types of noise corrupt the desired signal by conduction, capacitive links, mutual inductance or by radiation. On the other hand, some types of noise are produced by the internal material and elements that make the sensor. Examples of this type include thermal noise, shot noise and $\frac{1}{f}$ (low frequency) noise.

4. *Non-linearity*

   In many cases primary sensors are non-linear i.e. if the physical parameter x_1 produces y_1, then a scaled version of the input kx_1 (k is a constant) will not produce a similarly scaled output ky_1. Graphically, the diagram of the relation between input and output of the sensor is not a straight line that passes through the origin.

5. *Memory*

   The output of a sensor at any time, may not only depend on the value of physical parameter at that instant, but also on the past history of the input. This memory effect arises primarily because of stored energy in the sensor that can not be altered or dissipated instantaneously. Energy-storage elements are in the lump of material of every real sensor. They are analogues of capacitance and inductance in an electrical system. Having memory also means that a sensor cannot immediately follow a change in the physical input parameter. This defect is also called the time or frequency defect.

The effects of these defects can be rejected or dramatically reduced using digital signal-processing techniques. With the invention of microprocessors the housing of a sensor is able to contain processors running sophisticated software [2, 3, 4, 5]. The capability of the processor not only copes with the primary sensor defects, but is also used to perform many other difficult tasks such as: processing raw data and communicating them in a responsive manner to the peripheral environment, self testing and auto-calibration. A sensor that has a microprocessor in its housing is called an "intelligent sensor" or "smart sensor" [6, 7, 8].

1.1 Thesis Overview

Classical signal processing techniques have been used extensively in dynamic measurement [9, 10, 11, 12]. They are largely based on linear, local and stationary mathematical models. These methods are reviewed in chapter 2.

Real-world sensors are often non-linear and their structures vary with time. The new field of Intelligent Signal Processing (ISP) [13] does not impose a simple mathematical model on the sensor. Rather it extracts the structure of the sensor using smart learning techniques and signal data. There is no need to guess equations to model a complex transfer function of a sensor. Instead the intelligent or 'model-free' techniques learn the behaviour of the sensor [14, 15, 16, 17]. Artificial neural networks(ANNs) are the most important black-box tools in ISP [18, 19, 20]. In chapter 3 the basic concepts of artificial neural networks are described.

Chapter 4 presents a new method for dynamic measurement using neural networks. To investigate this technique in practice, a special sensor called tri-beam load cell is used as an example. The details about this sensor are also given in this chapter.

The simulation and experimental results of using the neural network method when a sensor is stimulated by an impulse function are presented in chapter 5.

A new adaptive method for low frequency noise cancellation is described in chapter 6. If the frequency spectrum of the noise is within the bandwidth of the desired signal then it cannot removed by standard filtering methods, and adaptive filtering method should be used. The latter methods, however, are applicable when the noise and signal are uncorrelated. The method proposed in chapter 6 overcomes to this limitation.

Chapter 7 is dedicated to summarizing the results and conclusion. It also contains the suggestions for further work using Intelligent Signal Processing in dynamic measurement.

Chapter 2

Classical Methods in Dynamic Measurement

2.1 Introduction

Dynamic measurement is an important requirement in many situations. The application of the measurand to the raw sensor results in a transient output waveform that can sometimes take a considerable time to settle sufficiently before a stable measurement is achievable [21]. Dynamic measurement requires the system to determine the final value of the measurand before the transient effects have decayed. Several methods for dynamic measurement have been proposed. Most of them use system identification or inverse system identification techniques, and can be classified in two major categories which will be described in the following sections.

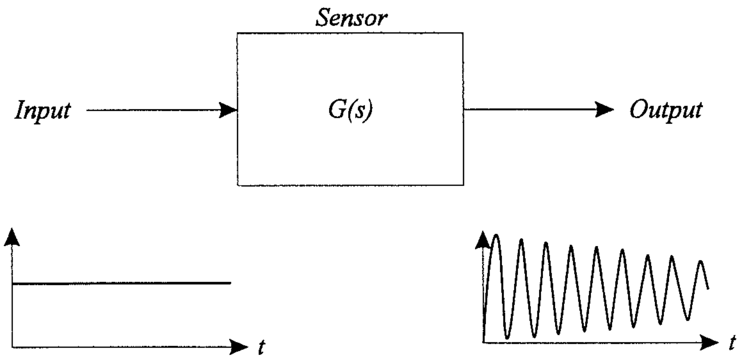


Figure 2.1: A typical step response of a primary sensor.

2.2 Adaptive Digital Filtering

Adaptive digital methods use the basic systems theory. The primary sensor is considered as a system with transfer function $G(s)$. A typical response of a second order sensor when its input is a step function is illustrated in figure 2.1.

The general principle for eliminating the transient time is shown in figure 2.2. A filter having the reciprocal characteristic of the sensor is cascaded with it. Therefore, the transfer function of the whole system is *"one"* which means that any changes in the input transfer to the output without any distortion. This is also referred to as pole-zero cancellation.

The transfer function of a sensor can change for different measurands. For example, the characteristic of any load sensor changes when a load is applied to the transducer because the mass of the load contributes to the inertial parameters of the system. Therefore the transfer function of the digital filter should change accordingly. In practice, an adaptive digital filter is used. Adaptive digital filters are linear systems [22, 23] requiring the assumption

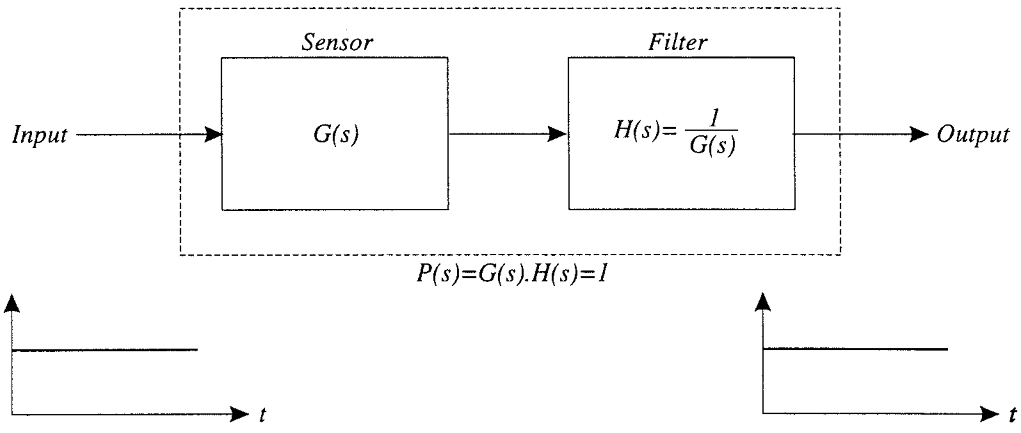


Figure 2.2: A sensor is cascaded with its inverse system. Inputs appear in the output without any distortion.

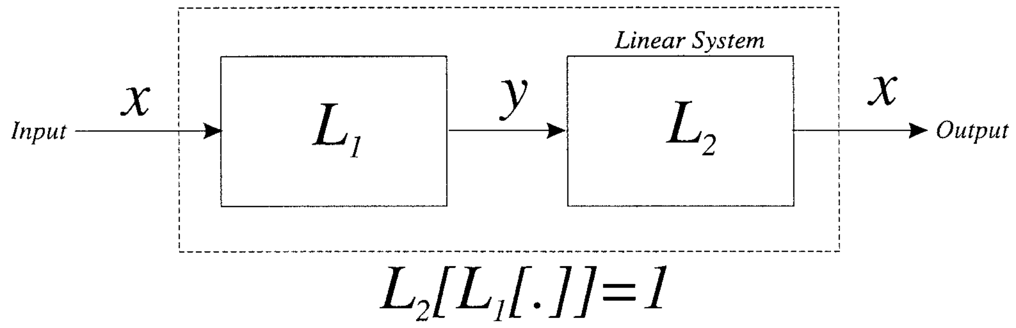


Figure 2.3: If the inverse of a system is a linear system, then the system is a linear system too.

that the sensor behaves as a linear system. As a result the adaptive digital filtering method cannot be used for non-linear primary sensors. To prove the necessity of the above condition; consider the systems shown in figure 2.3. It is known that L_2 is a Linear system and it is also the inverse of L_1. Suppose that x_1 and x_2 are two inputs. y_1 and y_2 are defined as:

$$y_1 = L_1[x_1] \tag{2.2.1}$$

$$y_2 = L_2[x_2] \tag{2.2.2}$$

The characteristic function of the whole system is one; since:

$$L_2[y_1] = x_1 \tag{2.2.3}$$

$$L_1[y_2] = x_2 \tag{2.2.4}$$

and also if a and b are two arbitrary constants the following equation can be applied:

$$L_2[L_1[ax_1 + bx_2]] = ax_1 + bx_2 \tag{2.2.5}$$

The right hand side of the above equation can be rewritten using equations (2.2.3), (2.2.4) and linearity properties of L_2:

$$\begin{aligned} L_2[L_1[ax_1 + bx_2]] &= ax_1 + bx_2 \\ &= aL_2[y_1] + bL_2[y_2] \\ &= L_2[ay_1] + L_2[by_2] \\ &= L_2[ay_1 + by_2] \\ &= L_2[ay_1 + by_2] \end{aligned} \tag{2.2.6}$$

The above equation results to:

$$L_1[ax_1 + bx_2] = ay_1 + by_2 \tag{2.2.7}$$

Using equations (2.2.1) and (2.2.2) to rewrite the right hand side of the above equation:

$$\begin{aligned} L_1[ax_1 + bx_2] &= ay_1 + by_2 \\ &= aL_1[x_1] + bL_2[x_2] \end{aligned} \tag{2.2.8}$$

The above equation shows that L_1 is a linear system.

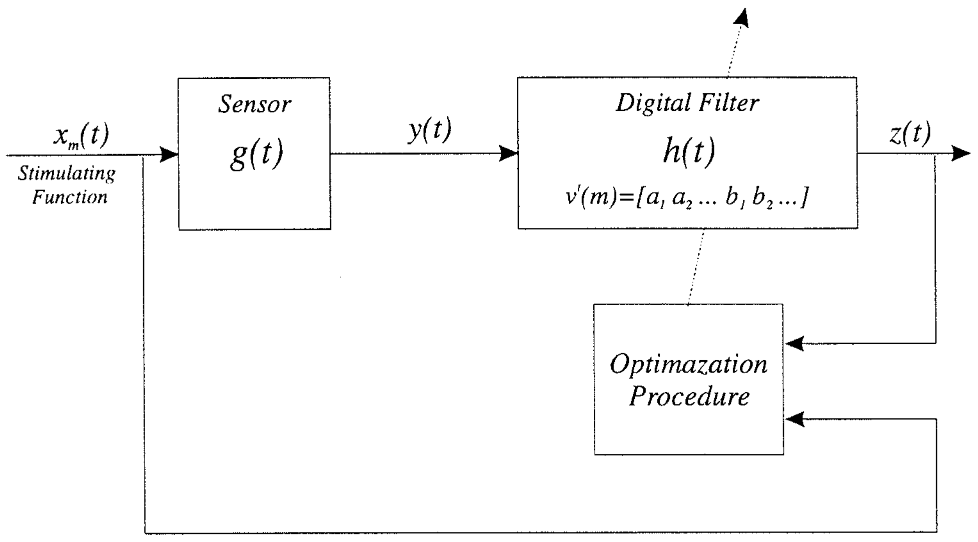


Figure 2.4: Inverse system identification technique is used in off-line mode for determination of the digital filter parameters which depend on the measurand.

Inverse system identification techniques are used to determine the transfer function of the digital filter that is mathematically defined as:

$$H(s) = A\prod_{i=0}^{N}\frac{(s-a_i)}{(s-b_i)}$$

Where A is a constant, N is the order of the system and a_i and b_i show the zeros and poles of the transfer function respectively.

It is assumed that this transfer function is the reciprocal of the sensor's transfer function i.e.:

$$H(s) = \frac{1}{G(s)}$$

Figure 2.4 shows the block diagram of the method that is used in off-line mode to determine the parameters of $H(s)$. The parameters depend on the measurand so the procedure should be repeated for different measurand. The sensor is stimulated by signal $x_m(t)$ which excites the sensor into its dynamic transient response. Impulse function, Step function and Ramp function are examples of signals that can be used as stimulating signal. Theoretically, these function do the same job. Practical limitations, however, determine the type of stimulating signal that can be used. Some of the points that should be considered are: First, producing ideal impulse and step function is impractical because in real world the level of a signal cannot change in zero second; therefore there will always be an error due to using these type of inputs. The amount of error depends on the sensor specifications. In the case of using step function if the transient time constant of the sensor is one order greater than the rise time of the step function then the error will be negligible. Second, certain type of sensors impair if they are stimulated by an impulse function.

The dynamic transient response of sensor, $y(t)$, feeds the adaptive digital filter. If the adaptive digital filter is a perfect inverse of the sensor then its output, $z(t)$, is identical to the stimulating function, $x_m(t)$. Optimization procedure block compares $x_m(t)$ and $z(t)$ and if they are not the same then it alters the parameters of the adaptive digital filter accordingly. Two factors dramatically infiuence the efficiency of optimization procedure. First, the rule of comparing two signals $x_m(t)$ and $z(t)$. Second, initial parameter values.

To compare two signals, a cost function is defined. For illustration purpose, suppose that the sensor behaves like a second order system and the stimulating function is a unit step function. Figures 2.5, 2.6, 2.7 and 2.8 show some examples of cost functions. The first cost function can be defined mathematically as:

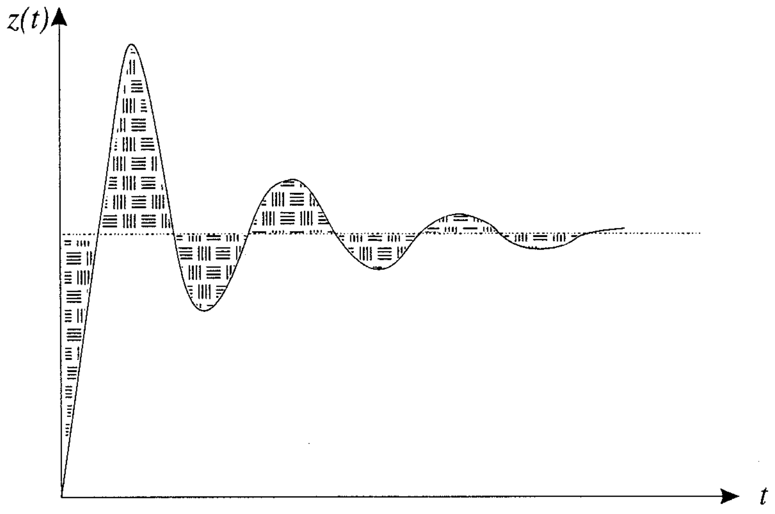


Figure 2.5: Cost function based on the area between sensor's response and step function (c_1).

$$c_1 = \sum_{n=1}^{N} |z(n) - 1|$$

Where $z(n)$ shows the samples of $Z(t)$. This cost function calculates the shaded area in figure 2.5 which is the area between the step function and the sensor response. The second cost function evaluates the area between the first peak of the response and the corner of step function. Mathematically it can be written as:

$$c_2 = |z_x - 1| \times t_x$$

The cost function shown in figure 2.7 calculate the distance between the first peak and trough i.e.:

$$c_3 = z_x - z_n$$

If the times that of the first peak and trough happen is added to the previous cost function the forth cost function results:

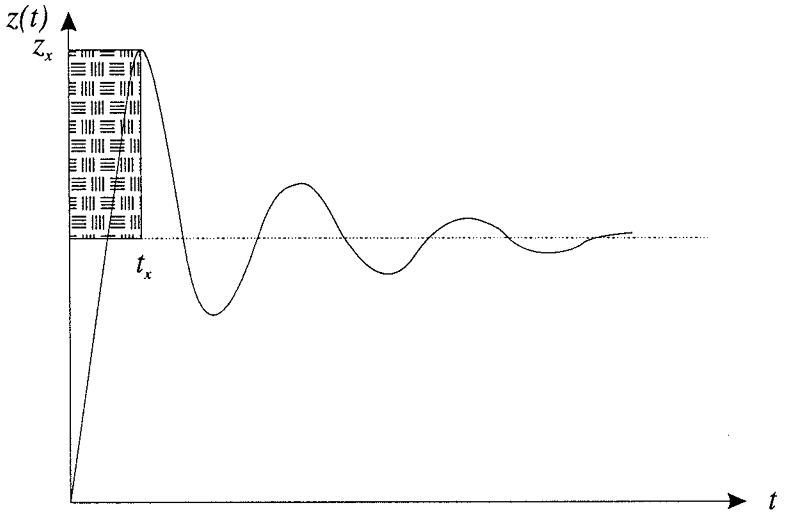


Figure 2.6: Cost function based on the area between the first peak's response and step's corner (c_2).

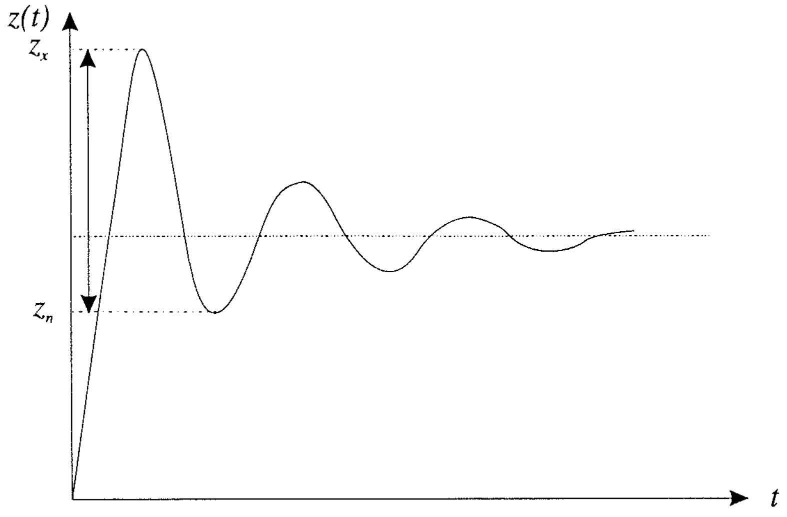


Figure 2.7: Cost function based on the distance of the first peak and trough (c_3).

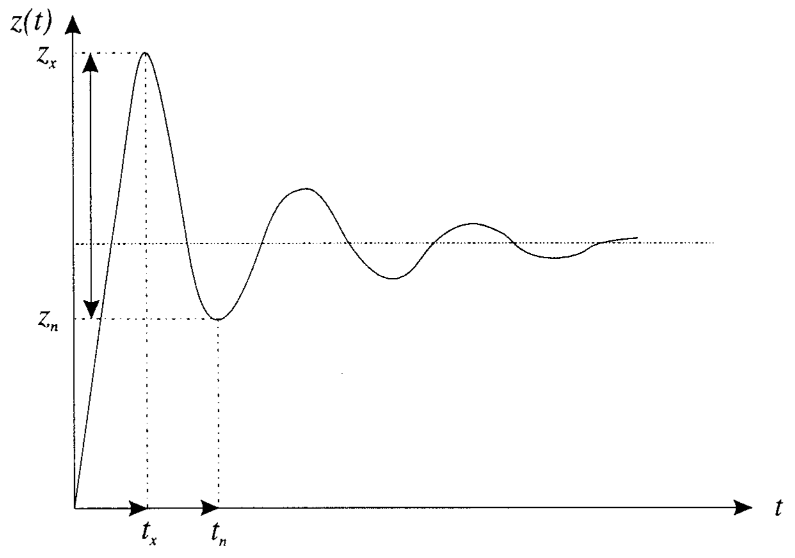


Figure 2.8: Cost function based on the distance of the first peak and trough plus the their occurrence times (c_4).

$$c_3 = (z_x - z_n) + (t_x + t_n)$$

Optimization techniques such as the Steepest Descent method [24, 25] and the Simplex method [26, 27] are used to minimize the cost function. The selection of cost function is a non-trivial problem. First, the amount of computation will be reduced if the cost function is well behaved, i.e., it descends smoothly to the optimal point. Secondly, it will have to be evaluated many times over during the search sequence, so even the smallest saving of mathematical complexity can largely decrease the overall computation time. Despite the importance of the cost function selection, there is no straightforward way to show which cost function should be chosen. Simulation has shown that for a second order system, the cost function that is based on the error area between the step function and the output of the system (c_1) is successful, although it is fairly costly in processing time [26].

Beside the cost function, the initial parameters values impact dramatically the performance of the optimization procedure. If they are chosen appropriately then the optimization procedure finds the minimum in a smaller number of iterations. Furthermore, most of the performance surfaces of cost functions have local minima. Some initial values of parameters cause the optimization procedure to become trapped in a local minimum instead of finding the global minimum. To avoid local minima, the procedure of finding the minimum should be repeated a few times with different initial parameters values. The procedure, however, can be speeded up using extra information. For example, from the output waveform of the sensor the order of the model is guessed. If the sensor is a second order system then the zeroes of the adaptive digital filter can be approximated by examination of the sensor. Suppose $a \pm jb$ and $c \pm jd$ show the zeroes and poles of the sensor (where a,b, c and d are real constants.) then the transfer function of the sensor can be written as:

$$G(s) = \frac{[s-(a+jb)][s-(a-jb)]}{[s-(c+jd)][s-(c-jd)]}$$

The response of the sensor to a unit step response is:

$$y(t) = \frac{a^2+b^2}{c^2+d^2} + k\frac{a^2+b^2}{c^2+d^2}\, e^{ct}\cos(d\,t+\theta)$$

where

$$k\cos(\theta) = \frac{a^2+b^2}{c^2+d^2} - 1$$

The constant parameters of poles, c and d can be calculated as:

$$c = \frac{\ln\frac{y_1-y_s}{y_2-y_s}}{(t_1-t_2)/f}$$

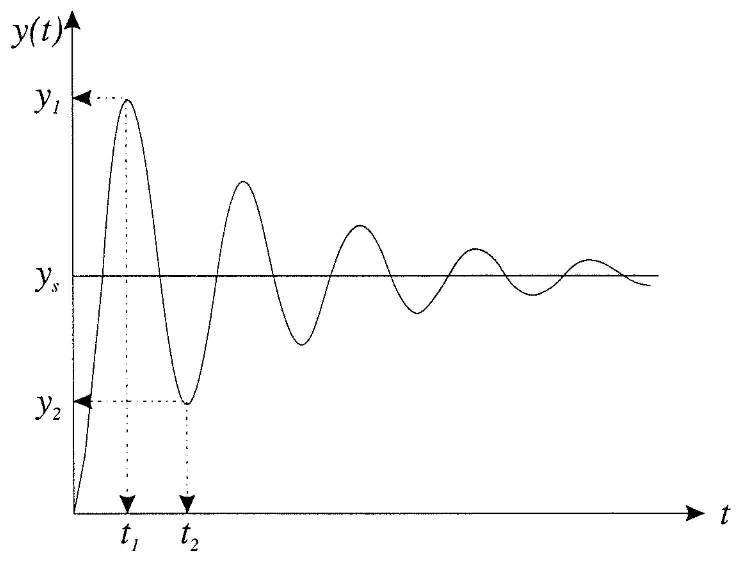


Figure 2.9: The zeroes of a second order system can be determined from the time response.

$$d = \frac{\pi}{t_2 - t_1} f$$

where f is the sampling frequency of the output and y_1, y_2, y_s, t_1 and t_2 are constants shown in figure 2.9. Thus the poles of sensor which are the zeroes of adaptive digital filter can be calculated. Due to noise they are not accurate but they are near the ideal. Other parameters of the adaptive filter are chosen arbitrarily. Then the optimum values are found by optimization procedure.

Assume $\boldsymbol{v}$ is defined as a vector that contains all of the parameters of adaptive filter i.e.:

$$\boldsymbol{v} \triangleq \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ b_1 \\ b_2 \\ \vdots \end{bmatrix}$$

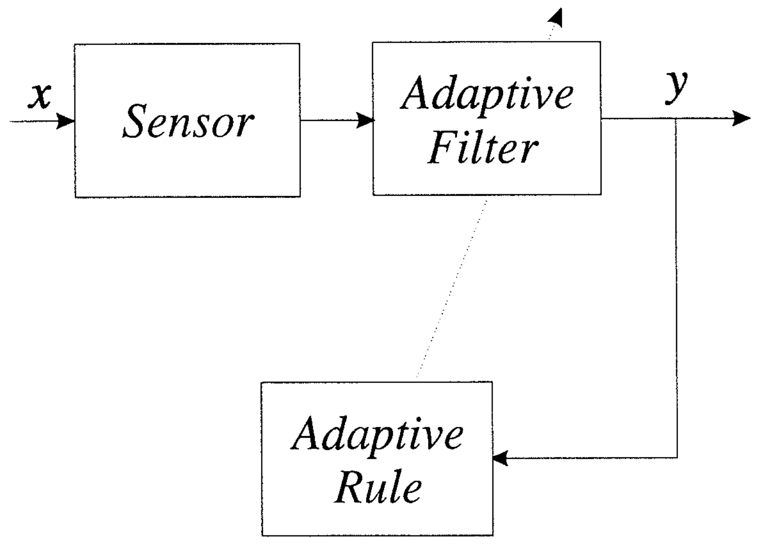


Figure 2.10: An adaptive rule is needed for on-line adjustment of the filter parameters.

The elements of $\boldsymbol{v}$ can be calculated for different measurands, m, using the above procedure in off line mode. To emphasize that $\boldsymbol{v}$ depends on m, it can be written as $\boldsymbol{v}(m)$. Knowing $\boldsymbol{v}(m)$ does not help to speed up the real time measurement because m is unknown in the first instance when a new measurement begins, so the parameters of the adaptive filter can not be set to appropriate amounts in order that the filter behaves as an inverse system and the transient time is canceled. Therefore, as figure 2.10 shows, an adaptive rule is required to modify the parameters of the adaptive filter according to the measurand. This rule is a crucial element but there is not a straightforward solution for it. As an example, for a load cell, it is found by simulations [26] that the suited filter has got a pair of conjugate zeros, $z_{1,2} = a \pm jb$, and the

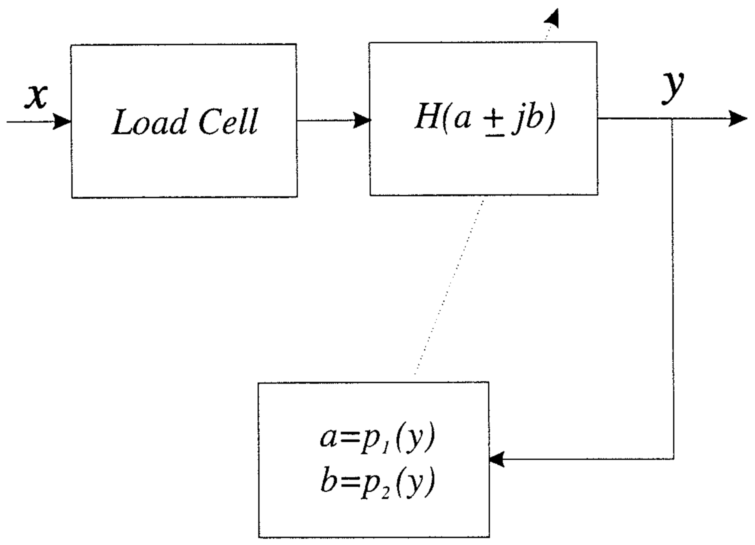


Figure 2.11: Block diagram of real time measurement for load cell.

relationship between them and load can be modelled as two polynomials:

$$a = p_1(m) = \frac{k_1}{m + m_0} \tag{2.2.9}$$

$$b = p_2(m) = \sqrt{\frac{k_2}{m + m_0} + \frac{k_3}{(m + m_0)^2}} \tag{2.2.10}$$

Where m_0 and m are the masses of the sensor and load respectively; and k_1, k_2 and k_3 are three unknown constants. If the values of a and b are known for two different m then k_1, k_2, k_3 and m_0 can be calculated from the equations 2.2.9 and 2.2.10. The inverse system identification procedure that was described already is used for this purpose. The sensor is stimulated with two different known loads and the zeroes of filter are obtained by optimization technique. This procedure should be repeated for each sensor in the calibration phase and the results were saved in memories attached to the sensor. The real time measurement operation is shown in block diagram of figure 2.11. In this block diagram m has been substituted with y since the output of the whole system

y is proportional to m. First the zeroes of the filter are set to arbitrary values. Then the the output y is calculated. This new value of y is used to calculate the zeroes of the filter once again. Repeating these steps results in a rapid approach to the steady state value of y.

In the on-line measurement, one of the most time consuming procedures is the calculation of the filter parameters for the new y. It has been suggested that a look-up table can be used for all possible rounded static output values. This, however, requires a tremendous amount of memory to save the parameter values [28].

The characteristic function of some sensors are like a low pass filter. Adaptive digital filters as an inverse system of the sensor behave like a high pass filter. Hence this amplifies the high frequency noise. In order to reduce the effect of noise without a negative impact on the speed of adaption, a digital low pass filter is cascaded to the adaptive filter. The bandwidth of the digital filter is chosen to be very wide at the beginning of the adaption process. Therefore it will not delay the output reaching the static value. As the output of the adaptive filter is close to the static value, however, the band width decreases to cancel the effect of noise. The rule for changing the bandwidth of the digital filter is obtained by simulation. As an example consider the following first order low pass filter:

$$H_n(z) = \frac{1 - e^{-\frac{T}{\tau}}}{1 - z^{-1}e^{-\frac{T}{\tau}}}$$

where T is the sampling period and τ is the filter time constant. The time constant τ bounds the bandwidth of the filter. The lower values of τ result in a wider bandwidth and vice versa. The adaptive rule for the noise filter can

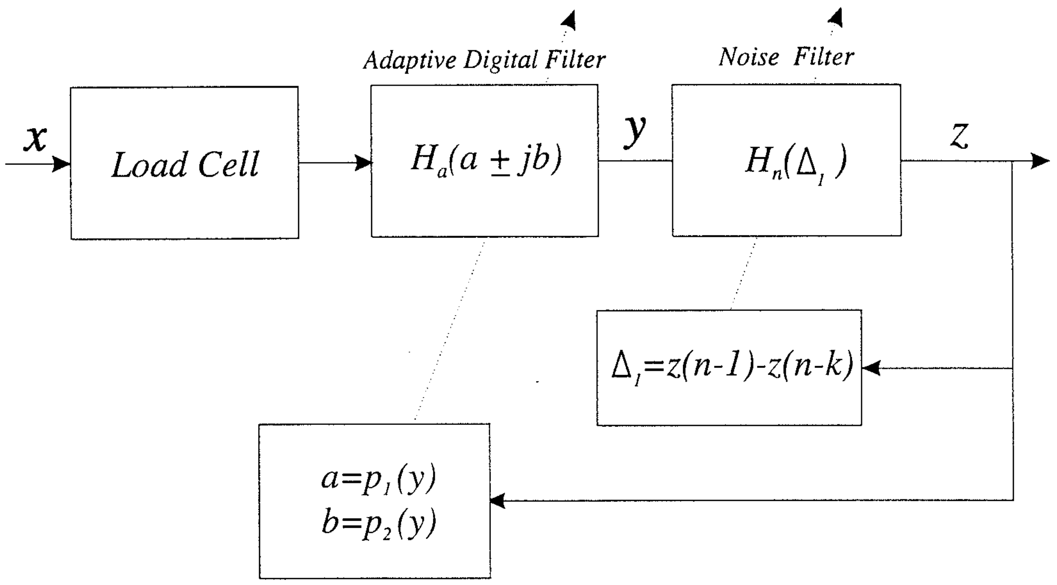


Figure 2.12: Adaptive digital method block diagram using Δ_1.

be defined as:

$$\tau = \frac{1}{\alpha + \beta\Delta}$$

where the constants α and β depend on the level of noise, and are chosen by trial and error. Δ is a variable that is used to change the value of τ and consequently the bandwidth of the filter. Two options for Δ are shown in figure 2.12 and 2.13. First definition defines Δ as:

$$\Delta_1 \triangleq z(n-1) - z(n-k) \tag{2.2.11}$$

where $z(i)$ denotes the i^{th} sample of the system's output.

In the second option, the difference between the output of the adaptive digital filter and the noise filter has been used for definition of Δ:

$$\Delta_2 \triangleq z(n-1) - y(n-1) \tag{2.2.12}$$

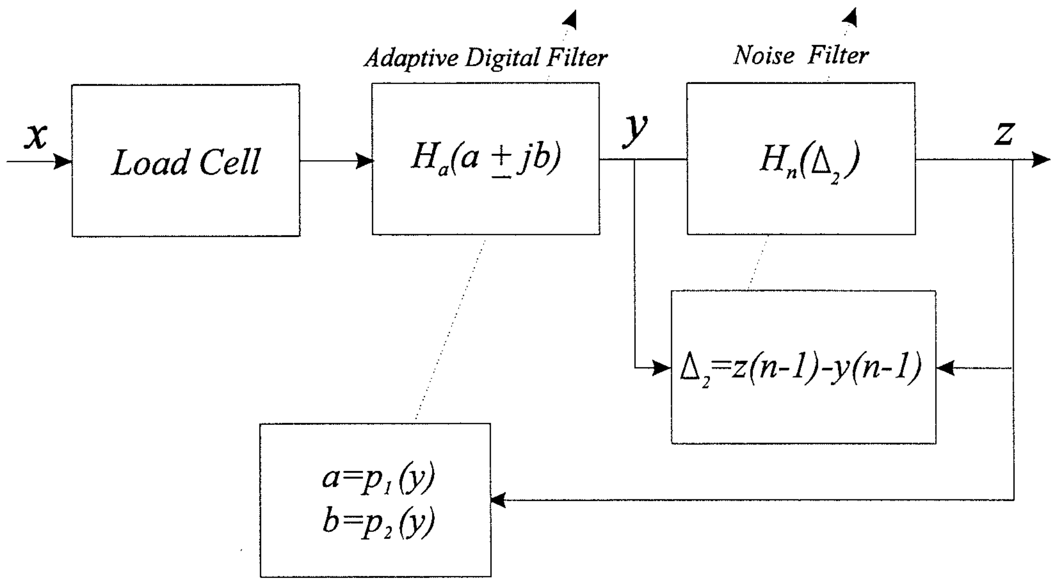


Figure 2.13: Adaptive digital method block diagram using Δ_2.

where $y(n-1)$ and $z(n-1)$ show the last samples of the noise filter and adaptive digital filter respectively.

Regardless of defining Δ as Δ_1 or Δ_2, it decreases in the steady state condition and hence the time constant of the noise filter, τ increases. This turns out to be a narrowband noise filter that rejects the noise effectively, and it is desirable for steady state condition. In the non steady state condition Δ is large, so the time constant of the noise filter, τ is small. This means the output of the adaptive filter comes out quickly from the output of the noise filter. Therefore the adaptive digital rule can rapidly adjust the parameters of the adaptive digital filter.

2.3 Choice of Low Pass Filter

This approach focuses on the fact that the process of filtering essentially involves averaging the signal in order to cancel noise and transient variations. Systems with different dynamics or noise components require distinct low pass filters for optimum processing of their output signals. When the measurand or other environmental parameters of the sensor change, the dynamics of the sensor or the noise vary. As a result, the optimum filter should be chosen according to the circumstance. Finding the optimum filter in each case has been done by Tariq using simulation [29]. The differences between optimum filters are in the type, order and cutoff frequency. Finite impulse response (FIR) filters suppress noise better than infinite impulse response (IIR) filters. The required order of the FIR filters, however, is considerably higher. It is normally between 50 to 400. In addition, it takes a long time to find optimum filters. The result on a weighing system shows that if just 81 different cases are considered and a successive approximation technique is used for searching, then it takes about 2.5 minutes of simulation time using a PC system. Above all, there is no suggestion on how optimum filters should be selected in practice. The difficulty is due of the fact that during the process of measurements the measurand is unknown, on the other hand for choosing the optimum filter the measurand should be known.

2.4 Kalman Filtering

Sometimes the sensor signal is contaminated with a very low frequency noise that is in the main bandwidth of the desired signal. The process of low pass filtering removes the transient variation and high frequency noise but does not eliminate this noise. The very low frequency noise causes the DC level in the output to fluctuate very slowly which decreases the accuracy [30]. A Kalman filter is applied to estimate the DC level. The Kalman filtering method requires the input function to be precisely described mathematically. In addition, it needs a mathematical model that expresses the dynamic behaviour by state equations, such as that illustrated below [31, 32]:

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}(t)\ \boldsymbol{x}_t + \boldsymbol{b}\ u_t$$

$$y_t = \boldsymbol{c}^T(t)\ \boldsymbol{x}_t + n_t$$

Where $\boldsymbol{x}$ is a vector whose elements record the state of the sensor, and u, y and n are the input, output and noise respectively. The subscript $_t$ denotes to the value of functions at a point in time. $\boldsymbol{A}, \boldsymbol{b}$ and $\boldsymbol{c}$ are identified by the specifications of the sensor. $\boldsymbol{A}$ and $\boldsymbol{b}$ depend on the measurand and are therefore time dependent. The aim of Kalman filtering is to estimate the state vector with each new measurement. In each time period, a new measurement is provided by the y_t and the estimated state vector $\widehat{\boldsymbol{x}}_t$ is calculated via the following recursive process:

$$\widehat{\boldsymbol{x}}_t = \widehat{\boldsymbol{x}}_{t-1} + \boldsymbol{K}_t(y_t - \boldsymbol{c}^T\widehat{\boldsymbol{x}}_{t-1})$$

$\boldsymbol{K}_t$, which is commonly described as the Kalman gain, is related to statistical specifications of the noise and state variables and also the dynamic behaviour of the sensor [33].

2.5 Model Parameter Estimation

Another classical method for dynamic measurement is based on model parameter estimation. This approach is based on the idea that a parametric model can be developed for the sensor from experimental data or by analytical methods. It is also assumed that the value of the measurand appears directly or it can be calculated from the parameters of the model. In each measurement, a short duration of the response is used to extract the parameters. The methods that use the model parameter approach are different depending on the type of model that they choose for the sensor and the procedure that is used to find parameters of the model.

2.5.1 Discrete-Time Model with Recursive Least Squares Procedure

With this method, it is assumed that the sensor can be modeled as a linear system. The z-domain model of the sensor with unknown parameters is derived. The parameters are estimated by fitting the model to a short duration of the measured sensor signal. The recursive least squares (RLS) procedure is used for the estimation of parameters. As an example consider a sensor that behaves as a second order system. When its input is a scaled step function, $Mu(t)$, the output of the sensor in z-domain, Y, can be written as:

$$Y(z) = \frac{k + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} U(z) \tag{2.5.1}$$

Where k, b_1, b_2, a_1 and a_2 are parameters of the system and $U(z)$ is the z-transform of the step function.

The measurand M is proportional to the steady state value of y. This can be shown mathematically as:

$$\lim_{n \to \infty} y(nT) = \alpha M$$

Where T is the sampling period and α is a constant. α can be obtained experimentally by a single static calibration.

On the other hand the end-value theorem of the z-transform implies:

$$\lim_{n \to \infty} y(nT) = \lim_{z \to 1} Y(z)$$

$$= \frac{k + b_1 + b_2}{1 + a_1 + a_2}$$

Therefore:

$$M = \frac{k + b_1 + b_2}{\alpha(1 + a_1 + a_2)}$$

or

$$M = \frac{b}{\alpha(1 + a_1 + a_2)} \qquad (2.5.2)$$

where $b = k + b_1 + b_2$.

Consequently the problem of finding the steady state value of the sensor is converted to a problem of identifying b, a_1 and a_2, according to the measurements: $y(n)$. The time domain equation for $y(n)$ can be extracted from 2.5.1 as:

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) + b \qquad , \qquad n \geqslant 2 \qquad (2.5.3)$$

This equation and 2.5.2 suggest that, in theory, five sample points from the beginning of the sensor output is enough to determine the measurand. The existence of the noise, however, requires more samples and using estimation procedures for parameter identification. Experimental results show if a third order low pass filter is cascaded with the sensor to eliminate the noise effect then a recursive least squares procedure needs 200 samples to estimate the measurand with an accuracy of $\pm 1\%$ [11].

2.5.2 Continuous-Time Model with Non-Linear Regression Procedure

This method is based on the non-linear regression fitting of a time-domain model to the output waveform of a sensor. The details of this method can be clarified by considering a sensor that behaves as an under damped second order system. A parametric model for the output of the sensor is as follows:

$$y(t) = \theta_0 + e^{-\theta_1 t}\, \theta_2\, \sin(\theta_3 t + \theta_4)$$

Where $\theta_0, \theta_1, \theta_2, \theta_3$ and θ_4 are the parameters of the model that are related to dynamic of the sensor and also to the measurand, M. When θ_0 is determined, the steady state value of the sensor can be calculated from the following formula:

$$M = k_1\theta - k_2$$

Where k_1 and k_2 are two constant that can be found by off-line calibration. To estimate the parameters of the model the vector $\boldsymbol{\theta}$ is defined as:

$$\boldsymbol{\theta} \triangleq \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$$

Then the output of the sensor can be written as:

$$y(t) = g(\theta, t) + \varepsilon(t)$$

Now, the problem of steady state prediction can be expressed as a non-liner regression: Find a set of parameters, $\boldsymbol{\theta}$, that minimize the modeling error, $\varepsilon(t)$, in the least-squares sense. The commonly employed Gauss-Newton iterative method can be used for this purpose. The computational complexity of this scheme, however, increases as the third power of the number of unknown model parameters. The number of parameters can be reduced to one by, firstly, determination of two parameters from off-line measurement. Secondly, two other parameters can be estimated by polynomial curve fitting, which is, a relatively low complexity operation. These latter parameters relate to $y(0)$ and $y'(0)$. If a polynomial of degree N fits $y(t)$ i.e.:

$$y(t) = \sum_{n=0}^{N} a_i t^i$$

then a_0 and a_1 are the estimations for $y(0)$ and $y'(0)$ respectively.

Simulation results show that for one parameter the non-linear regression procedure needs at least 80 samples to converge to the exact value [10].

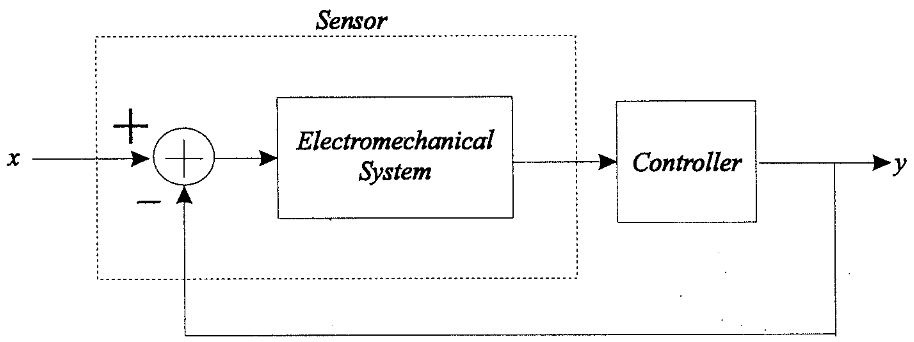


Figure 2.14: Block diagram of a controllable sensor.

2.6 Methods Applicable For Controllable Sensors

Sensors usually have one input and one output. The input is the physical or chemical phenomena that is the subject of the measurement. The sensor produces a measurable signal, usually an electrical waveform, which is the output. Certain types of sensor, however, have two inputs. An electromagnetic weighcell is an example. One of its inputs is the weight of the load. This input moves the position of the beam balance and this movement is sensed by a position sensor. The position sensor produces an electrical signal that is proportional to the displacement of the beam balance. This signal feeds a controller that generates an electrical current. The output of the controller is the second input to the electromagnetic weighcell. It creates a compensation electromagnetic force in order to return the beam balance to the initial position. The output of the controller, as depicted in figure 2.14, is used as the output of the electromagnetic weighcell because it is proportional to the input load.

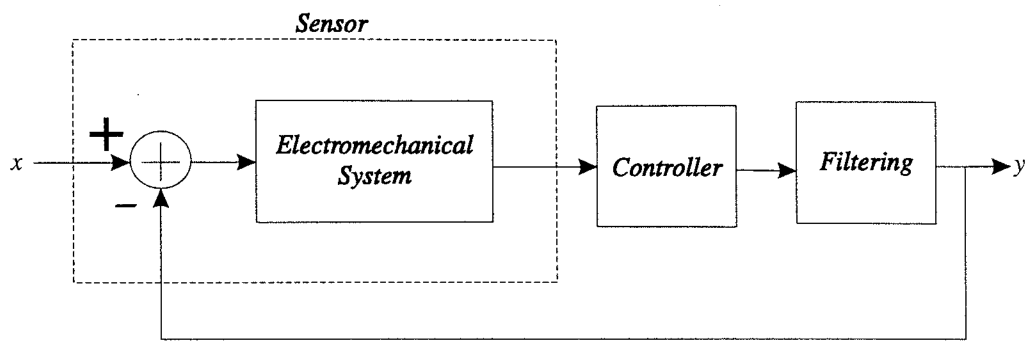


Figure 2.15: Block diagram of the integrated control and filtering method.

2.6.1 Integrated Control and Filtering

Although the methods mentioned in the sections 2.2 and 2.3 are usable for the controllable sensors, there is some room for further improvements based on the idea of the integrated control and filtering method. The system shown in figure 2.14 is clearly a position control loop. Control theory indicates that there is a contradiction between steady state error and transient time. In other words if the transient time reduces in order to speed up the measurement process, the accuracy decreases. Integrating control and filtering, as illustrated in figure 2.15, and using Linear Quadratic Gaussian methods to design the optimal control and filtering can increase both speed and accuracy, subject to the mechanical limitation of the dynamics of the sensor system [34].

2.6.2 Fuzzy Control

Traditionally the controller shown in figure 2.14 is a PID (proportional, Integral, Differential) controller. Its parameters are determined based on a simplified linear model for the sensor. With the fuzzy control method, a fuzzy logic technique is used to design the controller. The fuzzy rules are designed according to the desired response for the unit step input to the sensor. This method does not use any filter to reduce the effect of noise. The feeding mechanism of the sensor, however, is deliberately designed to prevent the production of external noise. It particularly reduces the generation of noise having low frequency components [35].

Chapter 3

Neural Networks

3.1 Introduction

The speed and computing ability of digital computers have progressed dramatically during recent years. Nowadays it is trivial to work with personal computers that perform a variety of well-defined tasks with a rapidity and reliability unrivaled by humans. For example, no human can match the speed of an ordinary personal computer for inverting a matrix. Nonetheless, the keyboard is still the main means that humans communicate with computers because speech and handwriting recognition have not yet been satisfactory solved. These tasks are effortlessly carried out by human adults. The brain uses billions of neurones that work in parallel to solve complicated problems. Two characteristics are common in the problems that are more effectively solved by the brain than serial computers. They are generally ill-defined and need huge amount of processing. The speed of logic gates in serial computers is about 10^8 times of the speed of a neurone in the brain [36] but this tremendous amount of processing can be used efficiently if they coordinate to solve a problem. This issue, however, is not a simple task. Thus computers are needed that have a natural parallel architecture and work on similar principles to those used in the brain. In fact, artificial neural networks were invented to mimic the nervous systems of biological creatures. They are not yet an exact copy of a nervous system. The main similarity between a nervous system and an artificial neural network is that both use a large number of simple elements that can learn and work together to solve complicated problems. The simplest element of a neural network is called a neurone. Neurones have different structures. In addition, they are connected to each other in many different manners to realize a wide range of artificial neural network classes. Each class has its own characteristics and suitable for special applications [37, 38]. In the following sections two categories of neural networks that are used for function approximations will be discussed and then some learning algorithms that are used frequently to train networks will be reviewed.

3.2 Multi Layer Perceptron (MLP)

3.2.1 Neurone Model

The basic neurone used in MLP networks is shown in figure 3.1. The vector:

$$\boldsymbol{x} \triangleq \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

is the input to the neurone. The components of the input vector are weighted by coefficient weights and then the sum is computed. This is called the inter product of the weight row vector,

$$\boldsymbol{w} \triangleq \begin{bmatrix} w_1 & w_2 & \cdots & w_N \end{bmatrix}$$

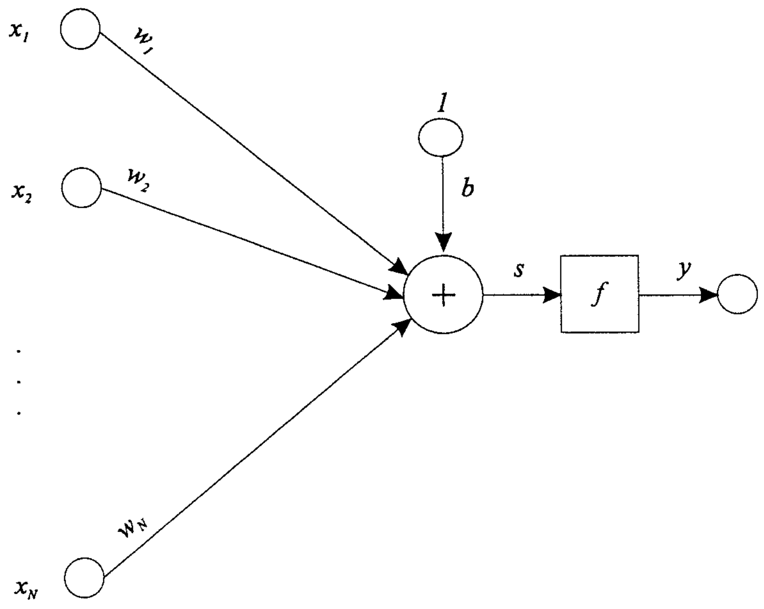


Figure 3.1: The basic neurone of MLP networks.

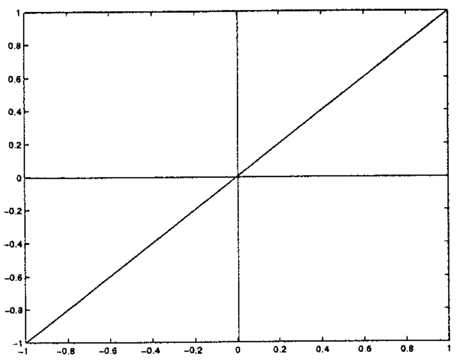
and input vector, $\boldsymbol{wx}$. The neurone has a bias b, which is summed with the weighted inputs to form:

$$s = \boldsymbol{wx} + b$$

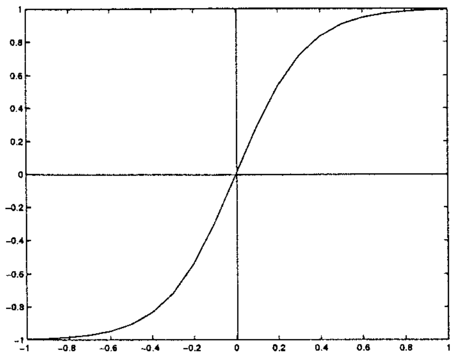
This sum, s, is the argument of the transfer function f. Here, f is typically either a sigmoid or a linear function, that takes the argument s and produces the output y:

$$\begin{aligned} y &= f(s) \\ &= f(\boldsymbol{wx} + b) \end{aligned}$$

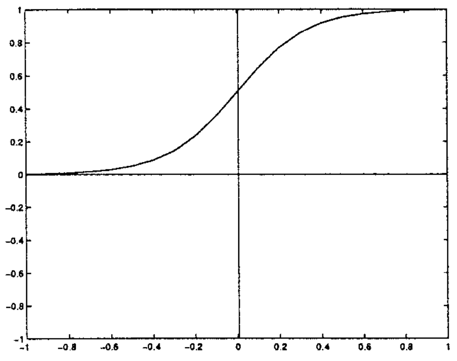
Sigmoid functions are monotonically increasing s-shaped functions, such as the hyperbolic tangent. Some examples of transfer functions are shown in figure 3.2.



(a) Linear function:
$y(x) = x$



(b) Sigmoid function:
$y(x) = \tanh(3x) = \frac{1-e^{-6x}}{1+e^{-6x}}$



(c) Sigmoid function:
$y(x) = \frac{1}{1+e^{-6x}}$

Figure 3.2: Examples of linear and sigmoid transfer functions. Sigmoid functions are monotonically s-shaped functions.

3.2.2 Architecture

Figure 3.3 shows a two layer MLP network. The first layer consists of M^1 neurones in which the inputs are all N elements of the input vector, $\boldsymbol{x}$. Typically, the number of neurones in the first layer, M^1, is not equal to the number of elements of input vector, N. The output of the first layer can be calculated as:

$$\boldsymbol{y}^1 = \boldsymbol{f}^1(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1) \tag{3.2.1}$$

Where the following definitions apply:

The superscript denotes the layer's number;

The vector

$$\boldsymbol{y}^1 \triangleq \begin{bmatrix} y_1^1 \\ y_2^1 \\ \vdots \\ y_{M^1}^1 \end{bmatrix}$$

is the outputs of neurones in the first layer;

The input vector has been shown by:

$$\boldsymbol{x} \triangleq \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N^1 \end{bmatrix}$$

The matrix

$$\boldsymbol{W}^1 \triangleq \begin{bmatrix} w_{11}^1 & w_{12}^1 & \cdots & w_{1N}^1 \\ w_{21}^1 & w_{22}^1 & \cdots & w_{2N}^1 \\ \vdots & \cdots & \ddots & \cdots \\ w_{M^11}^1 & w_{M^12}^1 & \cdots & w_{M^1N}^1 \end{bmatrix}$$

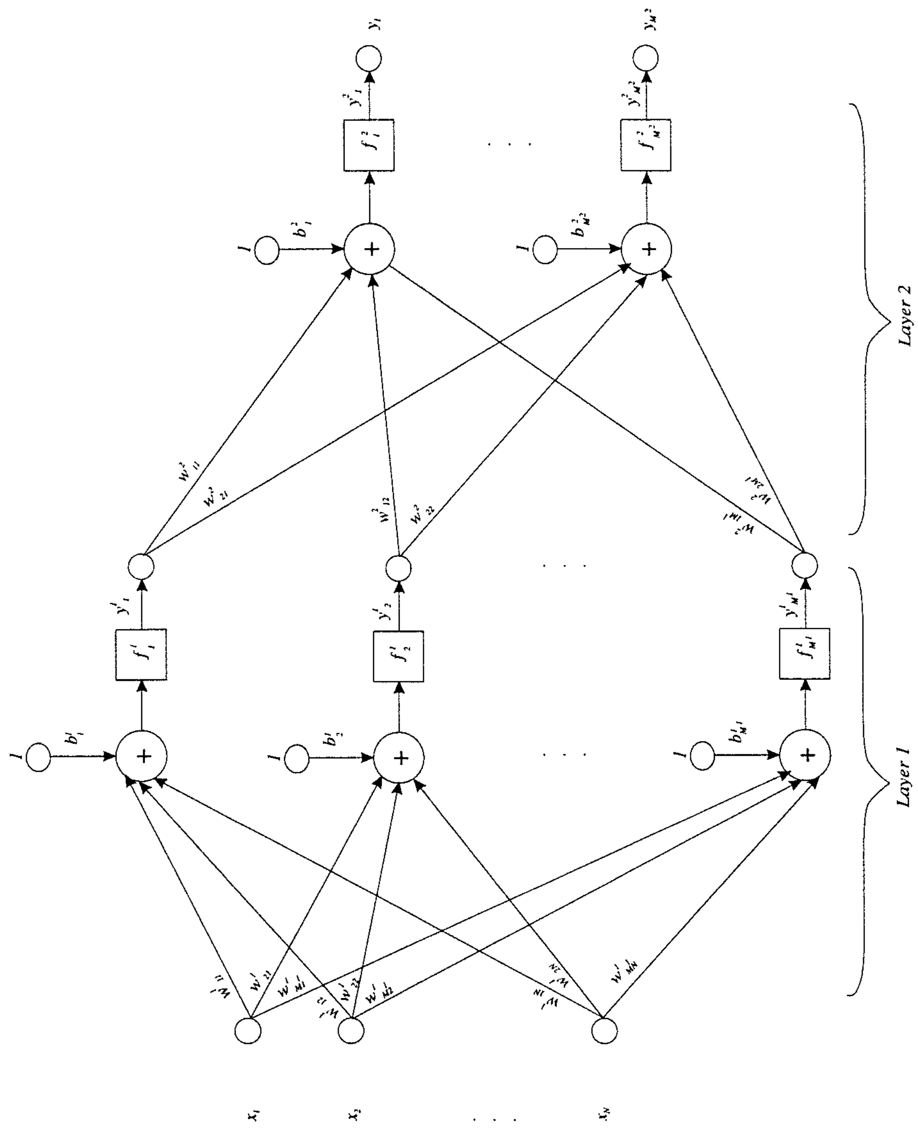


Figure 3.3: A Multi Layer Perceptron (MLP) network with two layers.

contains the coefficient weights in the first layer. The element w_{ij}^1 is the weight coefficients from the j^{th} element of the input vector to the the i^{th} neurones; The biases of the neurones in the first layer is displayed by:

$$\boldsymbol{b}^1 \triangleq \begin{bmatrix} b_1^1 \\ b_2^1 \\ \vdots \\ b_{M^1}^1 \end{bmatrix}$$

The vector $\boldsymbol{f}^1$ consist of the transfer functions of the neurones in the first layer:

$$\boldsymbol{f}^1 \triangleq \begin{bmatrix} f_1^1 \\ f_2^1 \\ \vdots \\ f_{M^1}^1 \end{bmatrix}$$

The second layer has the same structure as the first layer unless its inputs are the outputs of the first layer. Therefore in the equation 3.2.1 if

- All of the superscripts are changed to 2
- $\boldsymbol{x}$ is substituted for $\boldsymbol{y}^1$

then the output of second layer is obtained as:

$$\boldsymbol{y}^2 = \boldsymbol{f}^2(\boldsymbol{W}^2\boldsymbol{y}^1 + \boldsymbol{b}^2) \tag{3.2.2}$$

In the figure 3.3, the output of the second layer is the output of the neural network; So by replacement of $\boldsymbol{y}^2$ with $\boldsymbol{y}$ and combination of equations 3.2.1 and 3.2.2, the output of the net can be explicitly expressed in terms of the input vector as follows:

$$\boldsymbol{y} = \boldsymbol{f}^2(\boldsymbol{W}^2(\boldsymbol{f}^1(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1) + \boldsymbol{b}^2))$$

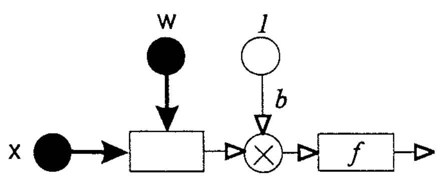


Figure 3.4: The basic neurone in the first layer of RBF networks.

3.3 Radial Basis Function (RBF)

3.3.1 Neurone Model

Figure 3.4 shows the basic neurone which is used in the first layer of Radial Basis Function networks. The neurone has two inputs:

The input vector:

$$\boldsymbol{x} \triangleq \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

and the weight row vector:

$$\boldsymbol{w} \triangleq \begin{bmatrix} w_1 & w_2 & \cdots & w_N \end{bmatrix}$$

Firstly, the Euclidean distance between the two above vectors are calculated:

$$\delta \triangleq \| \boldsymbol{w}^T - \boldsymbol{x} \| \triangleq \sqrt{\sum_i (w_i - x_i)^2}$$

Secondly this distance is multiplied by the neurone's bias, b, and then the result is fed into the neurone's transfer function. So the output of neurone is

$$\begin{aligned} y &= f(b\,\delta) \\ &= f(b\,\|\boldsymbol{w}^T - \boldsymbol{x}\|) \end{aligned}$$

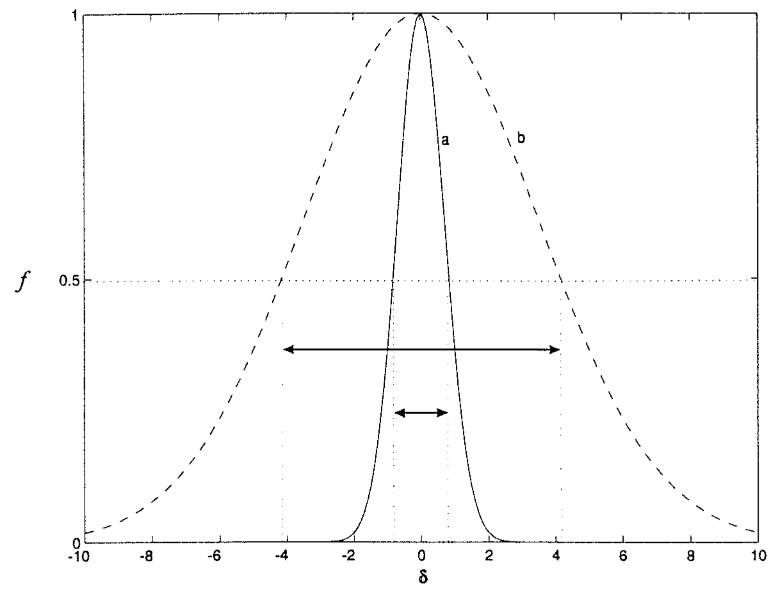


Figure 3.5: Radial basis transfer function, $f(b\,\delta) = e^{-(b\,\delta)^2}$, when (a) $b = 1$, (b) $b = 0.1$.

Here f is defined as:

$$f(b\,\delta) \triangleq e^{-(b\,\delta)^2}$$

Radial transfer functions are shown for two different b values in Figure 3.5. The output of a RBF neurones indicates the similarity between the input vector and the weight vector. It is one, the maximum, when the two vectors are exactly same and it is zero when two vectors are quite different. The neurones' bias controls the sensitivity of neurone. When the input to the neurone is a vector similar to the weight vector, the Euclidean distance between them, which is the input for the RBF transfer function, is zero. When the input of RBF transfer function is zero, its output is the maximum i.e. one. As the similarity between the input and the weight vector decreases, the Euclidean distance between the two vectors increases. As a result a greater value is fed to the

RBF transfer function. The RBF Transfer function, however, produces a lower value. In an extreme case, when the input vector is different from the weight vector and consequently the Euclidean distance tends to infinity, the output of the neurone is zero. The parameter b, the bias of the neurone, determines the sensitivity or the response width area of the neurone. In figure 3.5 the area which the output of the transfer function is greater than 0.5 is shown. For smaller b, this area is wider. This means as b decreases the sensitivity of neurone to dissimilarity of vectors decreases and the neurone responds to a wider range in the input space.

3.3.2 Architecture

Figure 3.6 shows a two layer RBF network. The matrix $\boldsymbol{X}$ has been constructed from the input vector:

$$\boldsymbol{x} \triangleq \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

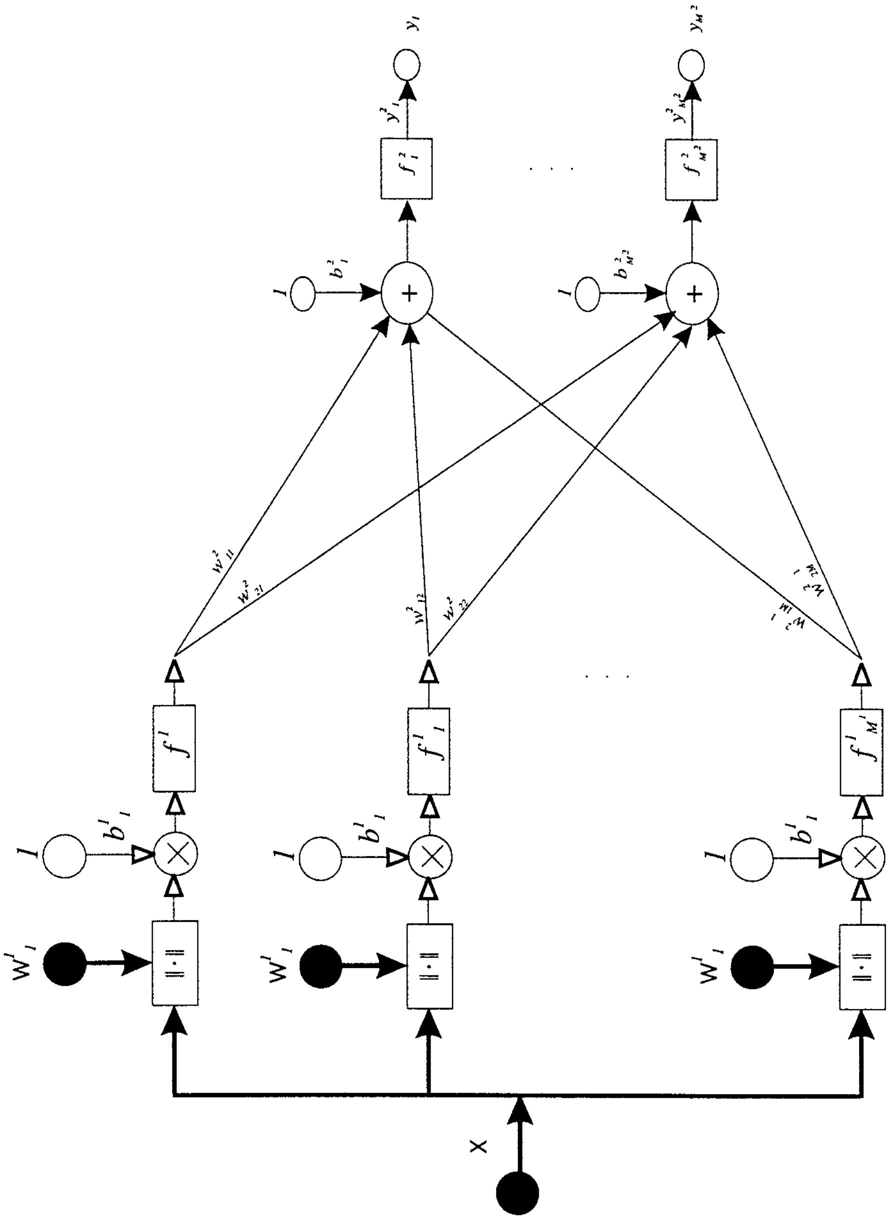


Figure 3.6: A Radial Base Function (RBF) network with two layers.

as follows:

$$\boldsymbol{X} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{(M^1 \times 1)} * \boldsymbol{x}^T$$

$$= \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ x_1 & x_2 & \cdots & x_N \\ \vdots & \cdots & \ddots & \cdots \\ x_1 & x_2 & \cdots & x_N \end{bmatrix}_{(M^1 \times N)}$$

The matrix

$$\boldsymbol{W}^1 \triangleq \begin{bmatrix} w_{11}^1 & w_{12}^1 & \cdots & w_{1N}^1 \\ w_{21}^1 & w_{22}^1 & \cdots & w_{2N}^1 \\ \vdots & \cdots & \ddots & \cdots \\ w_{M^1 1}^1 & w_{M^1 2}^1 & \cdots & w_{M^1 N}^1 \end{bmatrix}$$

contains the weight vectors in the first layer. The i^{th} row is the weight vector of the i^{th} neuron in the first layer.

The biases of the neurones in the first layer is displayed by:

$$\boldsymbol{b}^1 \triangleq \begin{bmatrix} b_1^1 \\ b_2^1 \\ \vdots \\ b_{M^1}^1 \end{bmatrix}$$

The vector $\boldsymbol{f}^1$ consist of the transfer functions of the neurones in the first layer:

$$\boldsymbol{f}^1 \triangleq \begin{bmatrix} f_1^1 \\ f_2^1 \\ \vdots \\ f_{M^1}^1 \end{bmatrix}$$

The neurones in the second layer have the same structure as neurones in a MLP network. Moreover, their transfer functions are linear. Therefore in the figure 3.6, the output of the net can be expressed in terms of the input as follows:

$$y = f^2(W^2(f^1(b^1||W^1 - X||)) + b^2)$$

3.4 Learning Algorithms

One important feature of neural networks is their ability to learn from experimental data. This feature makes neural networks very appealing for applications in which experimental data are readily available, but obtaining a precise model is not trivial. The neural network learns the behaviour of the system from the experimental data. During the training process, the input patterns and corresponding desired responses are presented to the network. An adoption algorithm automatically adjusts the weights so that the output responses to the input patterns will be as close as possible to their respective desired responses. There are different algorithms for training the networks that can be chosen irrespective of the neural network category.

3.4.1 Least Mean Square (LMS)

One of the most popular methods for adapting the weight is the LMS (Least Mean Square) algorithm. This algorithm minimizes the sum of squares of the error signals over the training set. The error signal is defined as the difference between the desired response and the output [36]. When the neurone is embedded in a multi-element neural network, however, an error signal is not directly available and more complicated procedures such as the back-propagation procedure must be used for adapting the weight vectors.

3.5 Generalisation

After adjusting the weight so that the network responds correctly to the trained samples if the network responds correctly to the unseen patterns, it is said that generalisation has taken place. Learning and generalisation are among the most useful attributes of a neural network. A network can have several layers. The two-layer network is surprisingly powerful. With a sufficient number of hidden elements, a sigmoid network with two layers can implement any continuous input-output mapping to an arbitrary accuracy [39].

Chapter 4

The Neural Network Method

4.1 Introduction

In this chapter first the theoretical basis of using neural network in dynamic measurement is established. Then the details of the experimental apparatus that is used to justify the theoretical and simulation results are described.

4.2 Theoretical Analysis

In the neural network approach, the sensor is considered as a non-linear mapping box. Figure 4.1 shows the functional block diagram of the sensing process. Variable x refers to the physical property that must be determined by the sensor. Ideally, the effect of the physical phenomenon i.e. mass (for a load cell) on the sensor must be constant from the start of the measurement process to the end of it. For example, the input to the sensor could be a step function, although this is difficult to achieve in practice. For a load cell the input function not only depends on the shape and type of the load but also on the way that the load is put onto the load-weighing platform [41]. The block "feeding

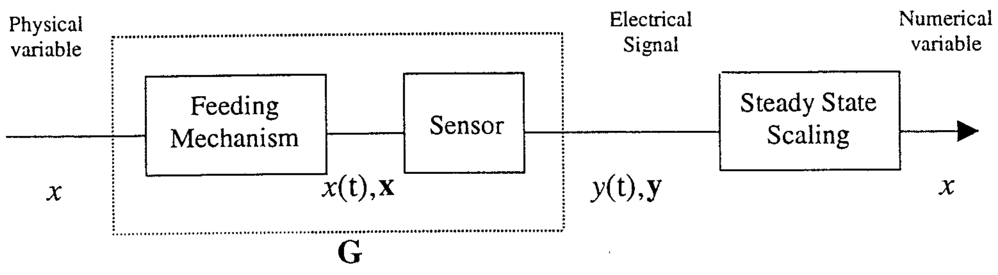


Figure 4.1: Block diagram of sensing process.

mechanism" represents how the constant physical variable x is applied to the sensor. Typically this will be a time dependent function $x(t)$. The sensor converts the waveform $x(t)$ to the output waveform $y(t)$. In digital systems, the waveforms are shown by their samples and these can be considered as vectors. Therefore the two blocks (feeding mechanism and the sensor) map the value x to vector $\boldsymbol{y}$. Mathematically this can be written as $\boldsymbol{y} = G(x)$, where G shows the mapping rule. The mapping rule G is generally a non-linear transform and if the inverse of G is found then the value of x can be determined from $\boldsymbol{y}$ as shown in figure 4.2. In many cases a small fraction of $\boldsymbol{y}$ has sufficient information to determine x. Therefore it is not necessary to wait for all of the samples of $y(t)$ and x can be predicted by observing the first few elements of $\boldsymbol{y}$.

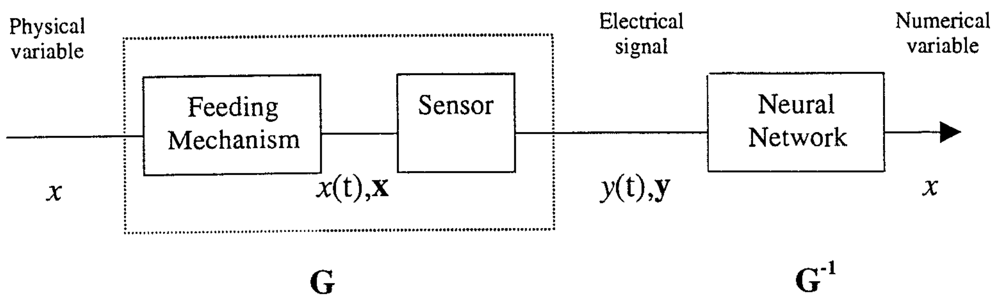


Figure 4.2: Neural network as an inverse system.

4.3 Experimental Apparatus

4.3.1 The Tri-beam Load Cell

A tri-beam load cell is used as a typical sensor [42]. Figure 4.3 shows the diagram of this load sensor. It is based on thick-film technology, therefore offering all its advantages of cheapness and robustness. The structure is planar and produced by inexpensive process of stamping out a steel shape and then printing it with gauges and conductors. The cell consists of three beams joined and supported at the centre with three pins near their extremities supporting the weigh pan. Thick-film strain gauges are screen printed onto the structure on both sides of beams. The twelve piezoresistors are connected in series and the interconnection is achieved through conducting film. A connection is made from one side of the beam to the other by a wire link. The strain gauges form a four arm active bridge arrangement. It is shown theoretically that this transducer is independent of eccentric loading if the centre of load is anywhere inside the circle passing through the three support pins [42]. It is interesting

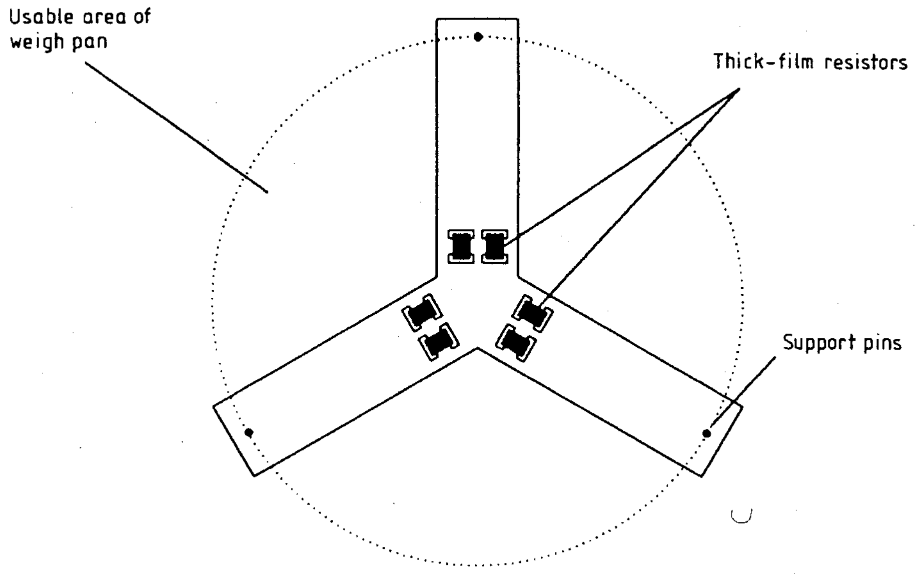


Figure 4.3: Diagram of the tri-beam load cell. The planar cell is supported at the centre and pins on the arms support the weigh pan[42].

to note that this sensor has a highly resonant structure and it becomes viable only with the introduction of intelligent sensor techniques. A photograph of tri-load beam cell is shown in figure 4.4.

Model of The Tri-Beam Load Cell

The tri-beam load cell's model is used for simulation. It has been shown [9] that the general equation for the dynamic response of tri-beam load cell including the applied mass m is given by:

$$\begin{cases} (m+m_0)\frac{d^2}{dt^2}y(t) + c\frac{d}{dt}y(t) + ky(t) = F(t) \\ y(0^-) = 0 \\ \frac{d}{dt}y(0^-) = 0 \end{cases} \tag{4.3.1}$$

Where m is the mass being weighed, m_0 is the effective mass of the sensor, c is the damping factor, k is the spring constant, $F(t)$ is the force function and $y(0^-)$ and $\frac{d}{dt}y(0^-)$ are two constants that show the initial position and the initial velocity of the platform, respectively. If the load is applied to the load cell without any bouncing then:

$$F(t) = m \cdot g \cdot u(t)$$

Where g is the gravitational constant and $u(t)$ the unit step function. Equation 4.3.1 can be written as follows for $t > 0$:

$$\begin{cases} \frac{d^2}{dt^2}y(t) + 2\alpha\frac{d}{dt}y(t) + \omega_n^2 y(t) = \frac{g}{m_0+m}m \\ y(0^-) = y_0 \\ \frac{d}{dt}y(0^-) = y_1 \end{cases} \tag{4.3.2}$$

Where $\alpha = \frac{c}{m_0+m}$ is the damping ratio and $\omega_n^2 = \frac{k}{m_0+m}$ is the natural frequency.

Equations 4.3.2 has the following solution:

$$y(t) = 2|k_1|e^{-\alpha t}cos(\omega_d t + \angle k_1) + y_p \tag{4.3.3}$$

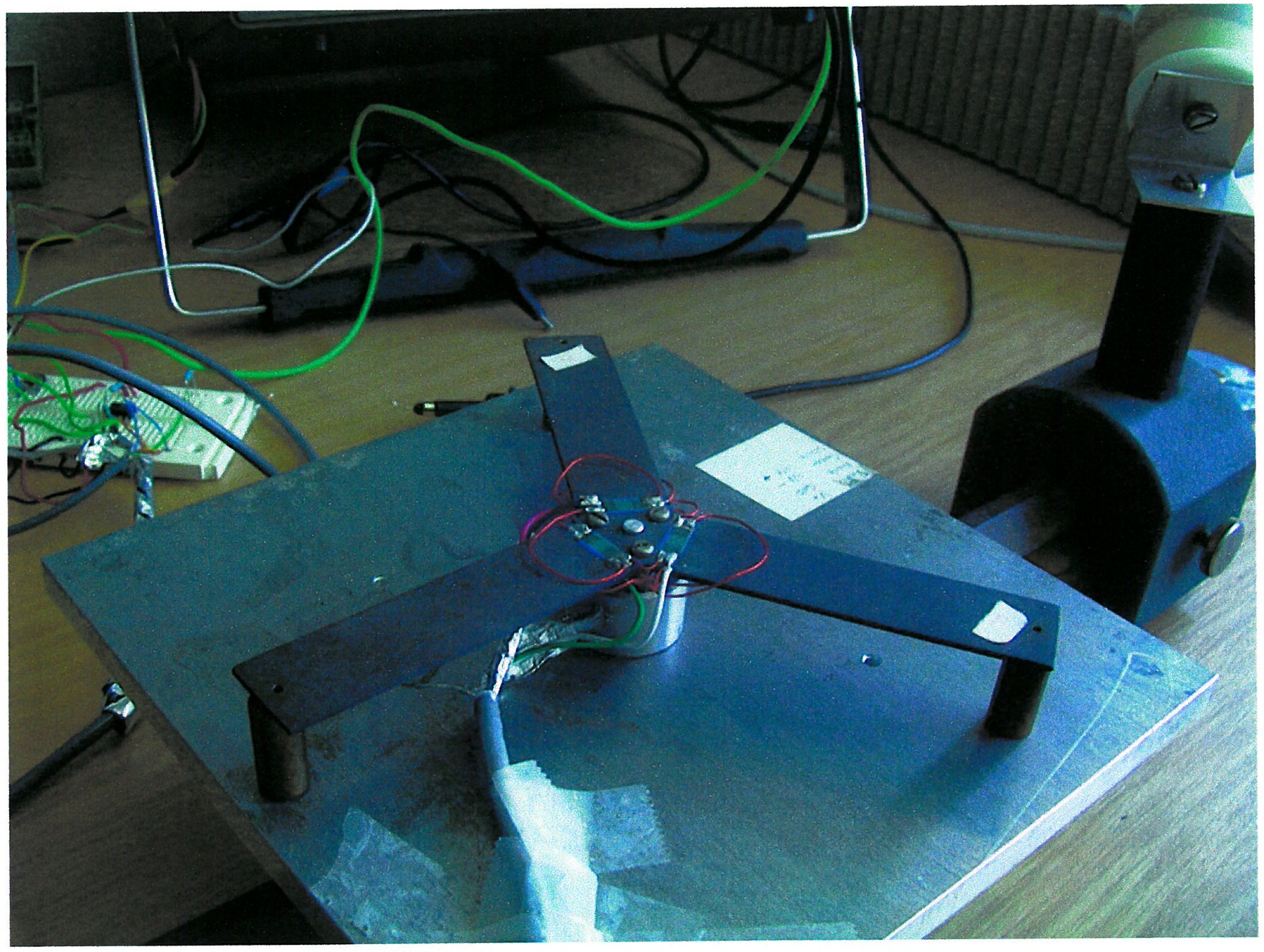


Figure 4.4: A photograph of the tri-beam load cell.

Where $k_1 = \frac{y_1-(y_0-y_p)(-\alpha-j\omega_d)}{2j\omega_d}$, $\omega_d = \sqrt{\omega_n^2 - \alpha^2}$ and $y_p = \frac{g}{k}m$.

ω_d is the damped frequency.

From the experimental data and using equation 4.3.3, the constants in equation set 4.3.1 were found to be $c = 3.5$ and $k = 27000$ [43].

4.3.2 Instrumentation Amplifier

The analogue output of the sensor is converted to a digital signal for further processing. The output signal is of the order of few milli volts; hence it cannot be used directly to feed the data acquisition card. An amplifier circuit was built to boost the signal [44, 45, 46, 47]. The circuit is based on the instrumentation amplifier AD525 [48]. This type of amplifier is used to suppress the effect of noise, especially the common mode noise. The gain of amplifier is set to 100 by setting appropriate link on the amplifier chip.

4.3.3 Data Acquisition Hardware and software

Data acquisition was achieved using LabVIEW version 4.1 [49, 50, 51, 52] and a data acquisition card, `AT_MIO_16E_10`[53, 54], manufactured by National Instruments. This card is capable of sampling up to 100k samples per second with 12 bit resolution.

4.4 Neural Network Software

The Neural Network Tool Box of MATLAB is used to implement the neural networks on a serial computer. This toolbox adds powerful commands to

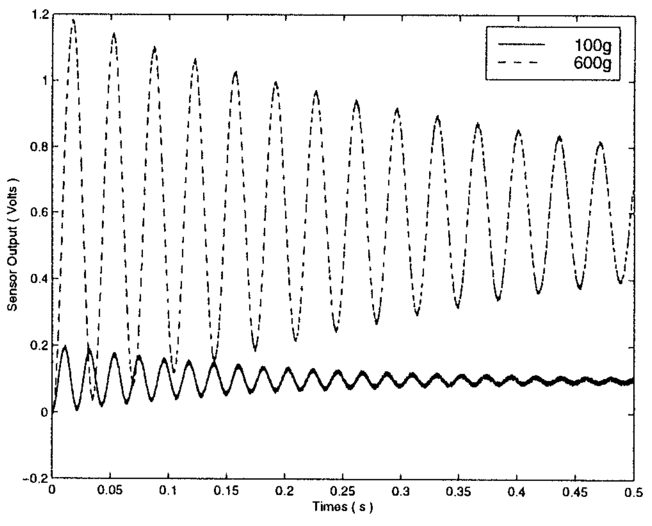


Figure 4.5: Simulated responses for 100 g and 600 g loads when the initial conditions(position and velocity of the platform) are zero.

MATLAB for implementation, training and simulation of neural networks. Different types of neural network and training algorithms can be chosen. In addition, it provides the number of training epochs, the performance of the network after each epoch and the number of operations that are needed for the output calculation. The latter can be used as an index for the time that is required in real time processing [55, 56, 57, 58].

4.5 Simulation Results

Figure 4.5 shows the simulated responses for applied masses of 100 and 600 g when the initial conditions are zero. It is important to note, however, that the initial conditions influence the shape of transient part of the response. Figure 4.6 shows the response of the sensor for two different initial conditions

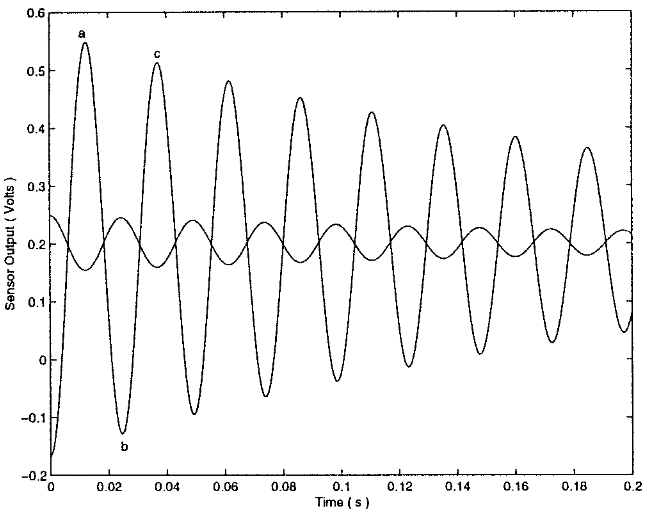


Figure 4.6: Simulated response for two different initial conditions(position and velocity of the platform) when the load is 200 g.

when m equals 200 g. Figures 4.5 and 4.6 show that the final values can be anticipated from the envelope of the waveforms. In fact, if h_1, h_2 and h_3 are three successive extreme points (such as a, b and c in figure 4.6) then it can be shown analytically that:

$$y_p = \frac{h_1 h_3 - h_2^2}{h_1 - 2h_2 + h_3} \qquad (4.5.1)$$

The equation 4.5.1 shows that each three successive extreme points have sufficient information to determine the final value. Therefore, these points were chosen as the features of the output signal. In addition, the time intervals between two successive intersect points of the output signal and the line $y = y_p$ were added to the features. This extra information increases the immunity of the method against noise and also its fault tolerance as will be discussed on

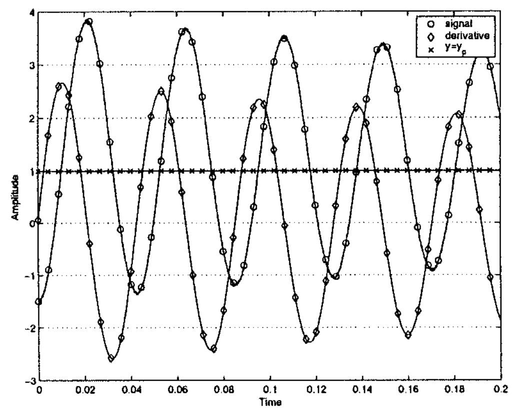


Figure 4.7: The peaks and troughs of the first derivative of the signal correspond to the intersect points of the output signal and the line $y = y_p$.

page 59 and illustrated in figures 4.13 and 4.14. At the beginning of the measurement y_p is unknown but as figure 4.7 illustrates, the peaks and troughs of the first derivative of the signal correspond to the desired points. In practice, a derivative filter cascaded with a low pass filter was used for obtaining the first derivative of the signal. The low pass filter was needed because the derivative filter naturally amplified the high frequencies of the input signal more than the lower frequencies. Therefore, if the signal was connected to the derivative filter directly, the amplitude of the noise was boosted and many fake peaks and troughs would appeared. The interval times are equal to half of the damping period, $2\pi/\omega_d$, and equations 4.3.3 and 4.5.1 show that there is a unique relation between them and the final value. Using these features instead of the raw samples of output signal, as shown in figure 4.8, causes the number of neurones in the neural network to reduce dramatically and hence reduces the computation time in the training and operating phases. The feature extractor finds the first three extreme points and two intervals time, and then passes

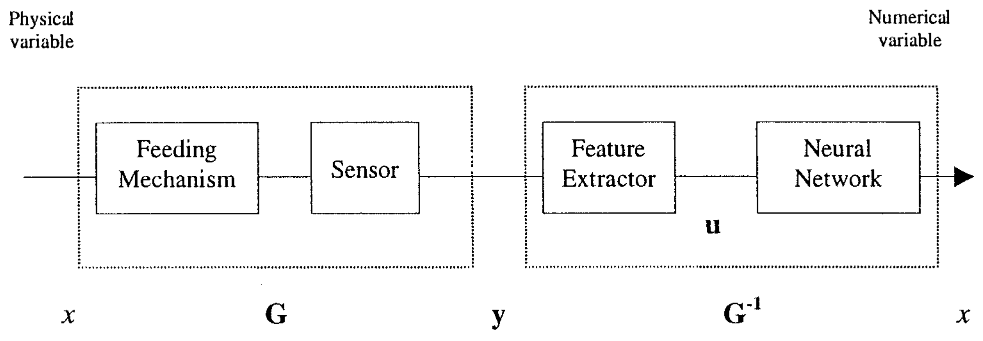


Figure 4.8: A feature extractor reduces the number of neurones in the neural network.

them to the neural network.

A multi-layer perceptron neural network architecture was used for simulation purpose. The network has five input neurones, eleven hidden neurones, and one output neurone. The transfer function for the first layer is sigmoid but for the second layer, a linear function is chosen because the output can exceed beyond [-1,1]. A set of 100 patterns is used for training the neural network. These patterns were generated by choosing masses that were uniformly distributed over the range of 100 g to 1000 g. i.e. $100, 200, 300 \cdots 1000$ g. The output for each load was generated for 10 different initial conditions that were chosen randomly. For testing purposes, the patterns were generated by choosing masses that were different from that used for training i.e. $150, 350, 550, 750$ and 950 g. In addition, the experimental data shows that the amplitude of the electronic noise is about 20 mV peak to peak. Therefore, a random signal with uniform distribution over -10 mV to 10 mV is added to the signal. The result

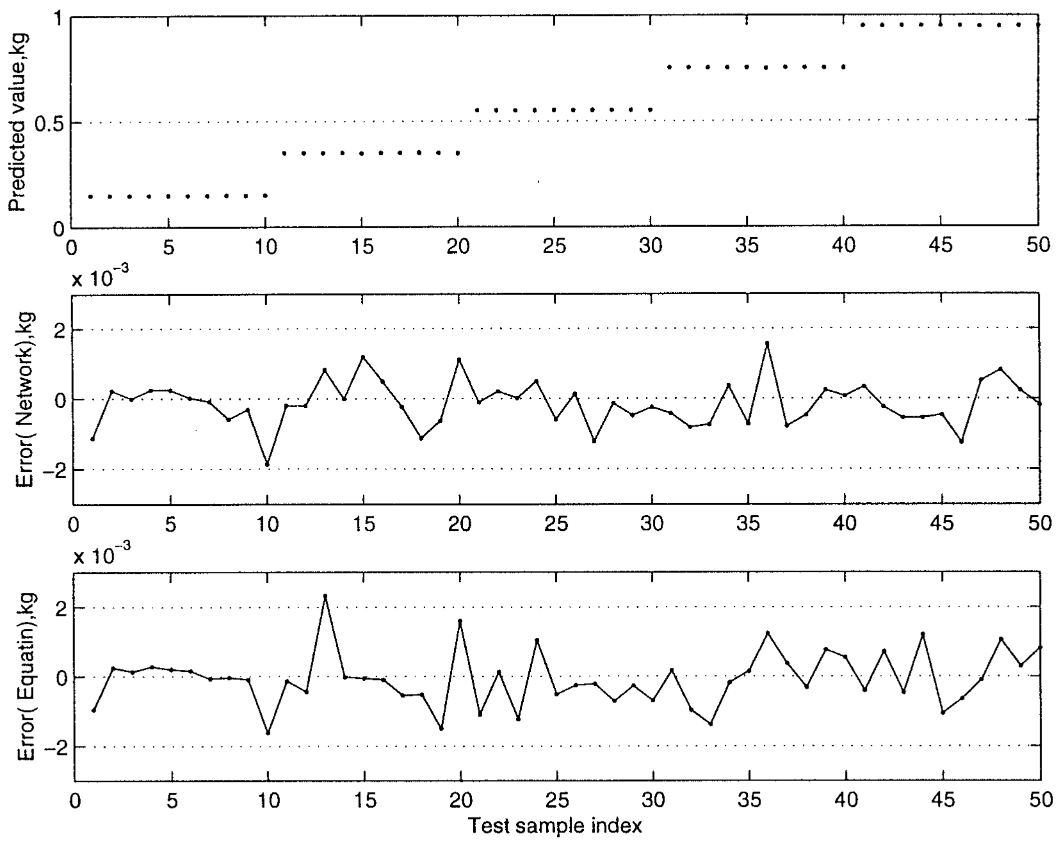


Figure 4.9: The output of the neural network, the network error and the equation error.

of the test for load 150, 350, 550, 750 and 950 g are shown in figure 4.9. Again, 10 patterns for each load were generated by choosing different initial conditions randomly. In this figure, the upper graph shows the values predicted by neural network and the middle one shows the absolute error(the actual value minus the value predicted by network) for each point. The lower graph shows the error if equation 4.5.1 used. It can be seen that the neural network predicts the final values to an accuracy of 0.2% of full scale. In addition, the error is a random signal whose mean is near zero and its magnitude range is consistent with the error range of the mathematical model. Figures 4.10, 4.11

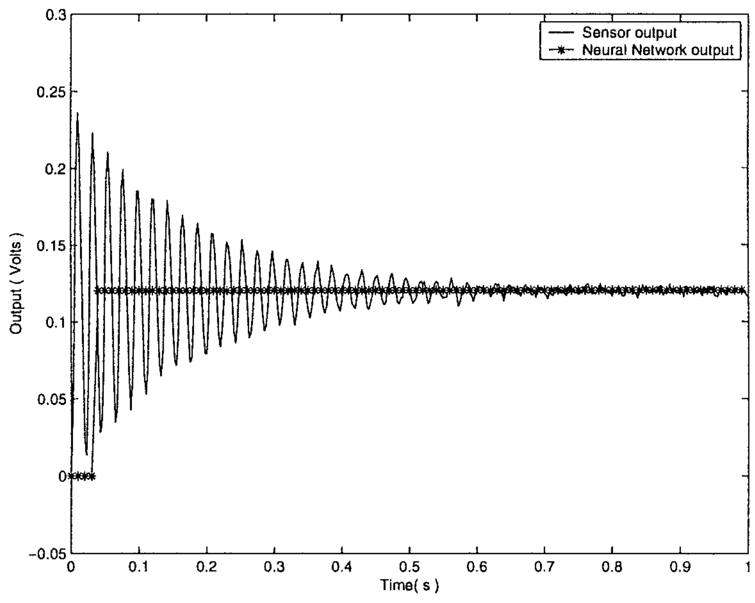


Figure 4.10: The output of the sensor and neural network when m=120 g.

and 4.12 show the time diagram of the sensor and the neural network output for loads 120, 570 and 930 g.

In the above examples, a neural network with 78 parameters used 5 features of the signal to calculate the steady state value of the signal. In fact, the final value was calculated from the following equation:

$$y = \boldsymbol{W}^2_{1\times 11}(\boldsymbol{f}^1_{11\times 1}(\boldsymbol{W}^1_{11\times 5}\boldsymbol{x}_{5\times 1} + \boldsymbol{b}^1_{11\times 1}) + b^2) \tag{4.5.2}$$

where $\boldsymbol{W}^2_{1\times 11}$ and $\boldsymbol{W}^1_{11\times 5}$ are two matrices with 11 and 55 parameters respectively, $\boldsymbol{b}^1_{11\times 1}$ is a vector with 11 parameters and b^2 is a scalar. All of these parameters are determined in the neural network training phase. $\boldsymbol{f}^1_{11\times 1}$ is vector consist of 11 sigmoid functions. The 5 input features is shown by vector $\boldsymbol{x}_{5\times 1}$.

On the other hand a rather simpler equation 4.5.1 used just 3 features to calculate the final value. It seems that using neural network not only does not

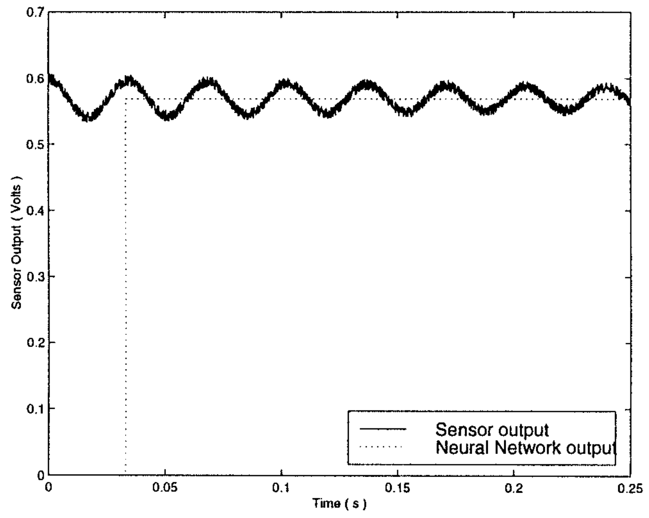


Figure 4.11: The output of the sensor and neural network when m=570 g.

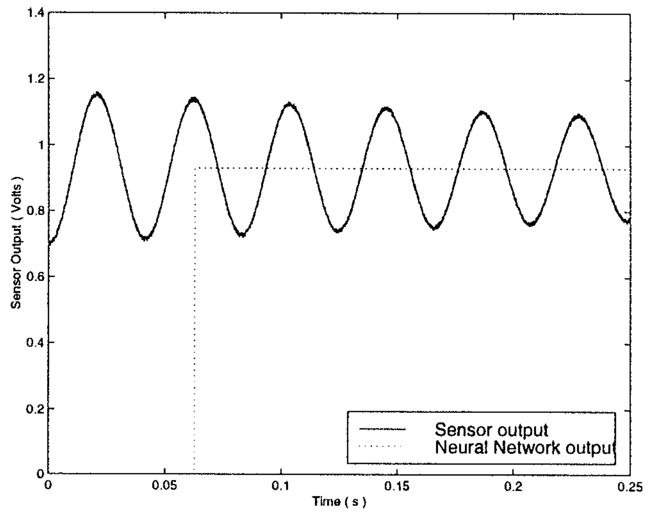


Figure 4.12: The output of the sensor and neural network when m=930 g.

have any advantages but also it has drawbacks. The following points, however, will show that this is not the case.

- Equation 4.5.1 is applicable for second order linear time invariant systems. It may be possible to derive a model for higher order systems but this model was chosen because:

  a) This is a simple model whose behaviour has been fully studied so it is a good reference to evaluate the new method.

  b) All of the previous dynamic measurements methods [26, 9, 11, 10, 29, 34], reviewed in chapter 2, have been established using a second order system. To make comparison between these techniques and the neural network solution possible a second order system was used.

  c) The Tri-beam load cell [42] was utilized as a typical sensor for obtaining the experimental data. It is a highly resonant system, so it presents a very demanding application for the dynamic measurement technique. Shi has modeled it as a second order system [26].

  Sensors are typically modelled as second order systems, but actually some of them have higher order characteristics. The difficulty of determining a more complicated model, and its parameters, has led to the acceptance of a second order model. In addition, the computing complexity of classical dynamic measurement methods increases dramatically when the number of parameters of the model rises [10] so that a simple model is to be preferred. The tri-beam load cell is a case in point; its output signal is the summation of the three distinct beams. In theory, all of the beams are identical, but in practice there are inevitable differences. Suppose each lever behaves as a second order system, and $G_i(s)$ shows its transfer function ($i = 1, 2, 3$). $G_i(s)$ can be written as:

$$G_1(s) = \frac{(s - a_1)}{(s - b_1)(s - c_1)} \tag{4.5.3}$$

$$G_2(s) = \frac{(s - a_2)}{(s - b_2)(s - c_2)} \tag{4.5.4}$$

$$G_3(s) = \frac{(s - a_3)}{(s - b_3)(s - c_3)} \tag{4.5.5}$$

where a_i, b_i and $c_i (i = 1, 2, 3)$ are constants that show the zeros and poles of transfer functions. The transfer function of the whole system, $G(s)$, obtains as follows:

$$\begin{aligned} G(s) &= G_1(s) + G_2(s) + G_3(s) \\ &= \frac{(s - a_1)}{(s - b_1)(s - c_1)} + \frac{(s - a_2)}{(s - b_2)(s - c_2)} + \frac{(s - a_3)}{(s - b_3)(s - c_3)} \end{aligned} \tag{4.5.6}$$

The above equation reveals that the tri-beam load cell is at least a sixth order system.

Neural network methods remove the problem of the system identification, parameter estimation and adaption of them according to environment variations. It also paves the way to use more precise models for sensors.

- Suppose that the feature extractor fails to present the correct value for one of the features. To simulate the new situation, the third feature was substituted with a random number taken from noise sequence. Figure 4.13 shows the effect of this fault on the network and the results from the equation 4.5.1. While the latter is like a random signal, the

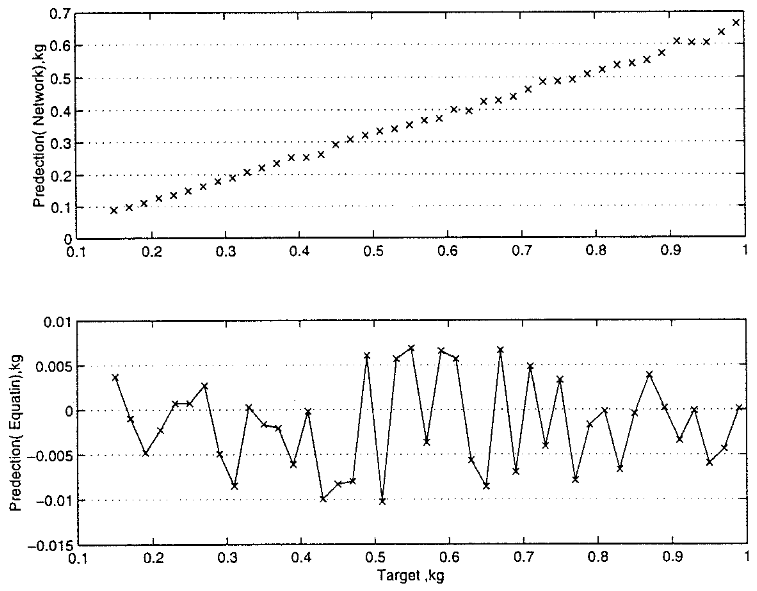


Figure 4.13: The effect of losing one feature.

neural network followed the inputs, although error is considerable. Figure 4.14 illustrates the results after the neural network was retrained for the new situation. The neural network shows a performance similar to the situation that it was fed with 5 features. This example demonstrates the advantage of neural network method in its ability to adapt new conditions and fault tolerance.

The last example revealed that neural network with 4 inputs could have the same performance as with 5 inputs. So it is reasonable to ask what is the optimum number of inputs. From equation 4.5.2 it can be concluded that as the number of input features and the number of neurones in the hidden layer reduce, the size of matrices and vectors in the equation decrease which in turn the lower number of arithmetic operations needed, and therefore the online operation is faster. It is not, however, very important if a parallel structure or

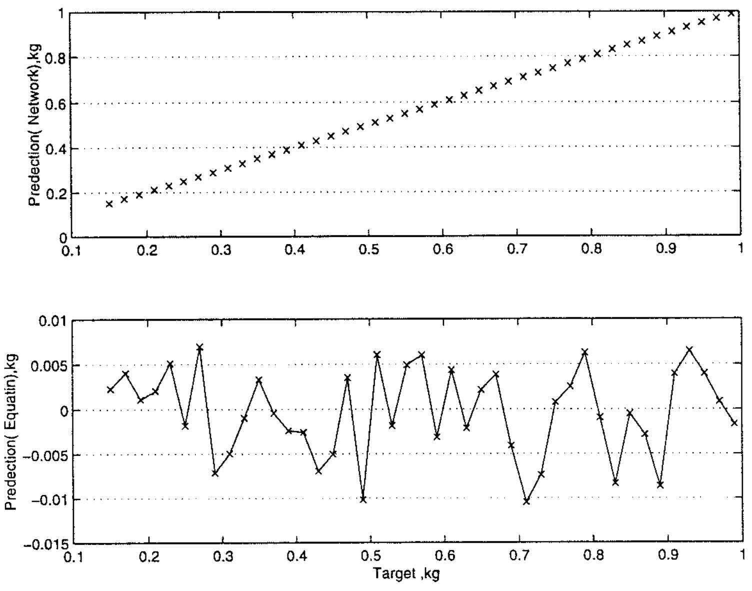


Figure 4.14: The effect of losing one feature after a training phase.

a fast processor is used for the network implementation. The important factor is the number of free parameters that are equal to the number of elements of weight matrices and bias vector. This number should be chosen according to the complexity of the problem in hand. If a network does not have enough free parameters, it cannot learn the behaviour of the system. On the other hand, a higher number of free parameters together with the low number of training samples could lead to over fitting i.e. the network memorizes the training samples and produces a precise output for them but fails to correctly answer to the unseen inputs. Therefore it is desirable to minimize the number of input features and the number of neurones in hidden layer.

To minimize the number of input features those features should be selected that effectively represent data and retain most of the intrinsic information content of the data. The classical procedures are principal factor analysis and components analysis. Both of these methods reduce dimensionality by forming a linear combination of the features. The latter reduce the number of features by discarding those linear combinations that have small variance and retain those terms that have large variances. The object of factor analysis is to find a lower dimensional representation that accounts for the correlation among the features. Usually, it is more profitable to exploit knowledge of the problem domain to obtain more informative features as it was done for the tri-beam load cell. The important point is that while including independent features helps the accuracy and reliability of the method, the irrelevant(or features with no new information) should be discarded.

To find the optimum number of neurones in the hidden layer, networks with 1 to 17 neurones in hidden layer are trained and tested. The procedure was repeated 50 times. For each case the maximum error is recorded. Figure 4.15 shows the results. Choosing 11 neurones increased the probability that the trained network produces less error. It should be pointed out that the pattern of figure 4.15 is not unique because of the ill-posedness of any finite set of data representing a target function. If the training and testing samples change, different patterns result; and the optimum number of neurones varies from 6 to 12. For a network to be able to generalize, the number of parameters should be less than the number of samples in the training set. As a rule of thumb, the number of samples should be 10 times that of the parameters [37]. If the samples are limited, an alternative solution is to stop the training before the network overfits. In the above example 78 parameters and 100 training samples were used but the training phase stopped after 15 epochs. Using 11

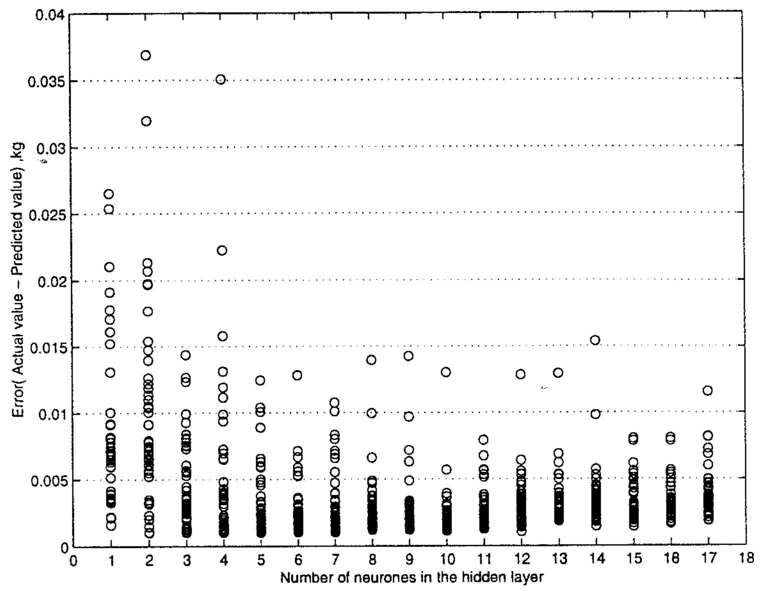


Figure 4.15: The effect of number of neurones in hidden layer.

neurones speeds up the training phase and this is of particular importance in application when the environment changes, therefore requiring the sensor to be retrained. The trained networks was tested by unseen samples to make sure over fitting has not happened. Figure 4.16 shows a typical mean square error of the network - the average squared error between the network outputs and the targets - for 100 epochs. It should be added that to make the training more efficient, the input and targets were scaled so that they fell in the range [-1,1]. Also, the default training algorithm in MATLAB, *'trainlm'*, was used. This algorithm appears to be the fastest method for training moderate-sized feedforward networks. It requires, however, the storage of some matrices which can be quite large for certain problems and therefore more memory is needed [56].

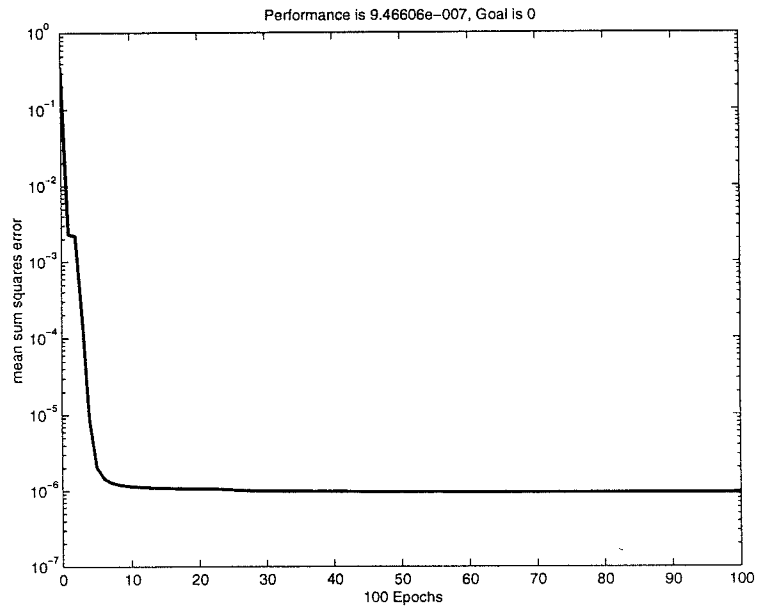


Figure 4.16: Performance curve.

4.6 Experimental Results

The final prediction method proposed in the previous sections was verified by means of practical experiments. The tri-beam load cell was used as the weighing sensor. Two sets of loads, one for producing the training samples and the other for testing samples, were chosen. The training load set consisted of 0, 94.01, 193.10, 291.61, 389.2, 487.3, 584.5, 681.3, 780.5, 878.3 and 975.7 g and the testing load set consisted of 148.29, 542.3 and 832.2 g. These loads were obtained by a combination of 50 and 100 g masses. The actual values of the 50 g and 100 g masses differ somewhat from the nominal values. The actual value of loads was measured using an Oertling weighing balance. This device measures masses below 300 g with an accuracy [59, 60, 61] of 0.01 g and masses between 300 to 3000 g with the accuracy of 0.1 g. The masses in training set were chosen to cover the range of zero to one kg uniformly. Three masses, near the lower, middle and upper range of training loads respectively, were chosen for the testing set. The loads in the testing set are different from the loads in training set, therefore the generalization property of the trained network can be asserted. For each load, 50 samples were captured. A program was developed in LabWindow/CVI environment (Ver 3.1) [62, 63, 64, 65] to acquire 1000 samples of the output of the sensor immediately after stimulating the sensor and for future processing write them to a binary file. These were subsequently read in to the MATLAB environment for the next step of the process. The sampling frequency was 10000 samples per second. In order to avoid problems with bounce while putting the load on the sensor, the method described in [9] was used. This was performed by firstly attaching the desired load, m, on the

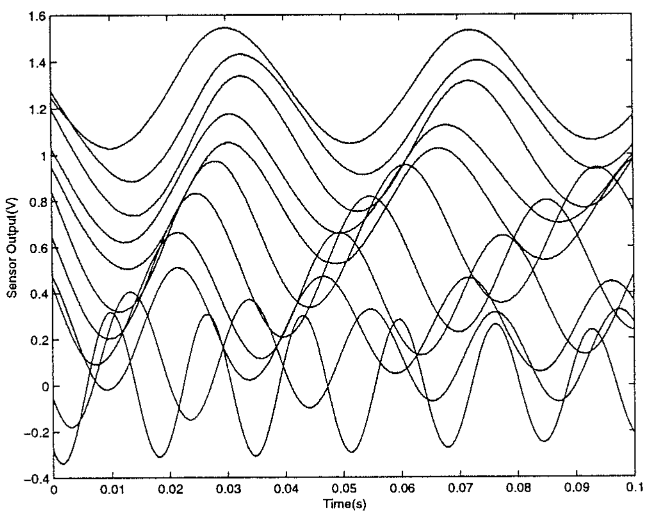


Figure 4.17: Typical waveforms used for training the neural network.

sensor and then putting another mass, m_s, on the top of the load m. The mass m_s was lifted off instantaneously. In this way, the inverse step response corresponding to the load m was obtained. Figures 4.17 and 4.18 show some typical waveforms of the training and test samples respectively. 30 samples of different masses were used with a backpropagation algorithm to train a neural network with eleven hiden neurones and one output neurone. The transfer functions of the first layer neurones were sigmoid whereas the output layer had a linear transfer function. The training of the neural network was achieved in 15 epochs. Figure 4.19 shows the histogram of the percentage error when 150 unseen samples were used to test the trained network. It is clear from the histogram that in this method the maximum of error is below $\pm 1.5\%$. The training procedure was repeated many times with different conditions and each time the maximum error was calculated. The results showed that if the number of training samples for each class were greater than 30 then the

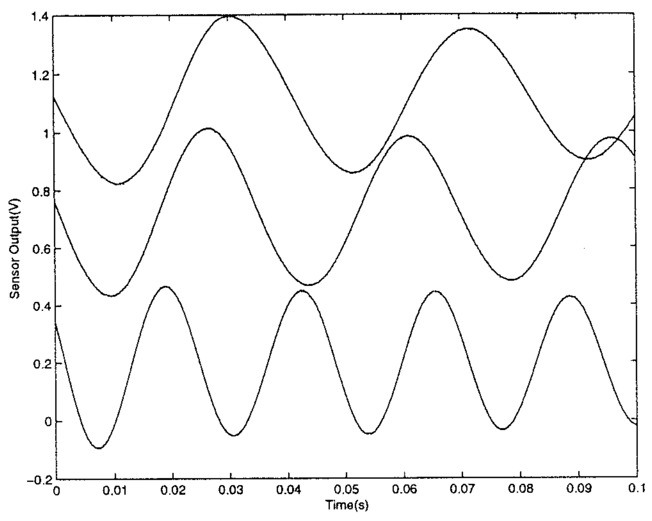


Figure 4.18: Typical waveforms used for testing the neural network.

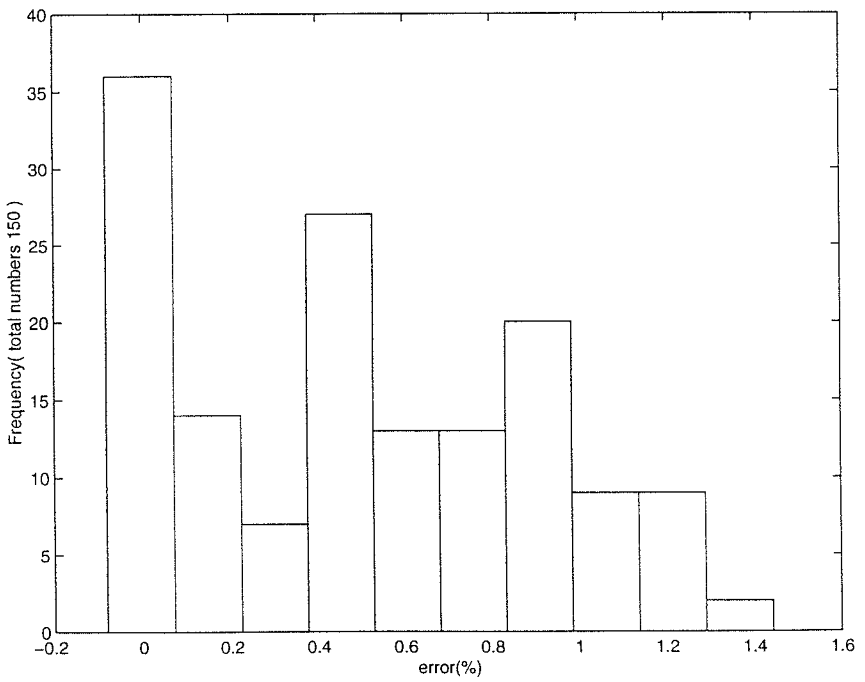


Figure 4.19: Histogram of percentage error.

maximum error is less than $\pm 1.5\%$. In addition, several different structures for the neural network were considered by adding a layer and changing the number of neurones in each layer. No considerable improvement in respect of reducing the maximum error was observed.

Chapter 5

Impulse Response

5.1 Introduction

In this chapter a novel method for dynamic measurement is described. This method deals with circumstances where the transducer senses the measurand for a very short period and the output of the sensor is a highly oscillatory signal and its steady state value is zero [66]. Limitation on the sensing time stems from the nature of some measurement tasks. Measuring the specifications of the coins in a vending machine is a case in point. When a coin is inserted into a vending machine it passes through a special route to reach to the decision point. There are several sensors throughout the way that measure different properties of the coin such as thickness, diameter and electrical conductivity. Based on these measurements, the coin will be accepted or rejected. It takes about 75 ms for a coin to move from the entering point to the decision point. Therefore the sensing duration for each transducer is very short and the final value of the sensors should be established in less than 75 ms.

Limitation on the sensing time can also arise from the demand for increasing the speed of the process of measurement. For example, consider weighing postal parcels when the load sensor is set beneath of a part of the carrier belt. The weight of each parcel is determined as it passes from that part of the strip. As the speed of the belt increases the available sensing time decreases. Similarly, the whole time for determining the final value of the sensor reduces. This is the case for any production line which requires measurement.

In the above applications, the sensor steady state response is zero, irrespective of the amplitude of its input. Therefore those classical methods which are based on the steady state response, such as adaptive digital filtering, can not be employed. This chapter shows that neural networks can successfully be applied to solve this problem. Firstly, simulation techniques are applied to establish the method and to obtain the optimum solution. Then experimental data are used to verify the simulation results.

5.2 System Analysis

System simulation was carried out on a tri-beam load cell. The sensor was described in section 4.3.1. The general equation for the dynamic response of this sensor is given by the equation 4.3.1. If a mass drops onto the load cell from a fixed height, h, and bounces clear, so it is in contact with the load for a very shot period, then the input function can be written as:

$$F(t) = M\delta(t)$$

Where M is a constant that depends on m and $\delta(t)$ is the unit impulse function. Equation 4.3.1 can be written as follows for $t > 0$ because the impulse function is zero for $t > 0$ and its effect only changes the initial conditions[67].

$$\begin{cases} m_0 \frac{d^2}{dt^2} y(t) + c\frac{d}{dt} y(t) + ky(t) = 0 \\ y(0^+) = 0 \\ \frac{d}{dt} y(0^+) = v_{2a} \end{cases} \tag{5.2.1}$$

where m_0 is the effective mass of the sensor, c is the damping factor, k is the spring constant, $y(t)$ is the position of the platform, $y(0^-)$ and $\frac{d}{dt}y(0^-)$ show the initial position and the initial velocity of the platform respectively and v_{2a} is a constant. To calculate the latter constant suppose:

v_1 shows the speed of load just before collision,

v_{1a} shows the speed of load just after collision,

v_2 shows the speed of the sensor plate just before collision,

v_{2a} shows the speed of the sensor plate just after collision,

If the load falls from the height of h with zero initial velocity then

$$v_1 = \sqrt{2gh}$$

where g is the gravitational constant. The sensor plate is at rest initially hence $v_2 = 0$.

Newton's Second law implies:

$$mv_1 + m_0 v_2 = mv_{1a} + m_0 v_{2a}$$

where m shows the mass of the load and m_0 is the effective mass of the sensor.

By definition [68] the restitution coefficient , e, is

$$e = \frac{v_{2a} - v_{1a}}{v_1 - v_2}$$

thus $e = 1$ refers to an elastic impact (impact with no energy loss) and $e = 0$ indicates inelastic or plastic impact.

Using this information v_{2a} can be calculated as:

$$v_{2a} = \frac{(1+e)m}{m+m_0}\sqrt{2gh}$$

Knowing v_{2a}, the solution for equation (5.2.1) can be written as

$$y(t) = \frac{(1+e)m}{m+m_0}\sqrt{2gh}\,\frac{1}{\omega_d}\,e^{-\alpha t}\,\sin(\omega_d t) \tag{5.2.2}$$

where $\alpha = c/m_0, \omega_n^2 = k/m_0$ and $\omega_d = \sqrt{\omega_n^2 - \alpha^2}$. The variables α, ω_d and ω_n show the damping ratio, the damped frequency and the natural frequency respectively.

Equation 5.2.2 shows that the steady state of the output is zero for different values of m. Therefore it is impossible to determine the input value from the steady state response. The information in the transient time response, however, can be exploited for measurement because the amplitudes of peaks and troughs only depend on the magnitude of m, if the loads fall from a fixed height. The transient time response can be envisaged as a unique pattern that is produced by the sensor for each measurand. Thus the measurement process can be modeled as a nonlinear mapping or regression problem [69] which can be solved effectively by neural networks. Radial Basis Function (RBF) and Multi Layer Perceptron (MLP) networks are two appropriate configurations for this type of application [37].

In the following sections the above proposed method will be investigated by means of simulation and experimental data. Firstly the parameters of the RBF networks are specified when the number of features of the signal which is used to feed the neural networks is fixed but the level of noise is altered. The change in the number of the neurones in the networks and the percentage of error are considered. The number of neurones in a network is an indication of the processing power that is required for implementation of the network. The percentage of the error determines the capability of the method coping with noise and also if it is suitable for a specific application. Secondly, the level of noise is fixed but the number of features of the signal which is used to feed the neural networks is changed and its effect on the percentage of the error and the number of neurones in the network is observed. These help to find the optimum preprocessing method. All the above steps are repeated for the MLP configuration. Then the RBF and MLP optimum solutions are compared to find out the optimum solution. Finally the results of simulation are utilized to design a network using experimental data.

5.3 Simulation Results Using RBF Configuration

To train a Radial Basis Function (RBF) neural network a total of 360 patterns are used. These patterns were generated by choosing 12 masses that were uniformly distributed over the range of 1 to 12 g. For each mass, 30 signals were simulated using equation 5.2.2. In reality the signal is corrupted by noise, therefore a random sequence is added to the signal. Then the first peak and trough of the signals were extracted as two key features. The features of each signal made a vector which was the pattern of that signal.

For testing purposes patterns were generated in the same way as described above with the exception of choosing 7 masses between 1 and 12 g. Most of these were different from those used for training. Therefore 210 patterns were used to test the trained network.

5.3.1 Noise and Number of Features

In the real world, sensor signals are always contaminated with noise and the amplitude of noise varies over time. It is important to investigate the capability of the measurement method to eliminate the effect of noise. For this purpose a random sequence was added to the signal. The random entries of the sequence was chosen from a uniform distribution in the interval $(0.0, b)$ where b determines the level of noise. It was changed from 0.02 to 2 percent of the maximum of the sensor's output of the relevant mass. For each b a RBF network was trained and tested. Figure 5.1 shows a typical error surface for a trained network. The number of neurones in the trained network and the maximum error that produced by testing pattern were recorded. This procedure was repeated thirty times. Each time a number was taken from a random variable generator that produced numbers uniformly distributed between 0.02 and 2. This number was scaled according to the maximum of the sensor signal to obtain the corresponding value for b. For the purpose of comparison the results were sorted ascendingly based on the level of the noise. They are

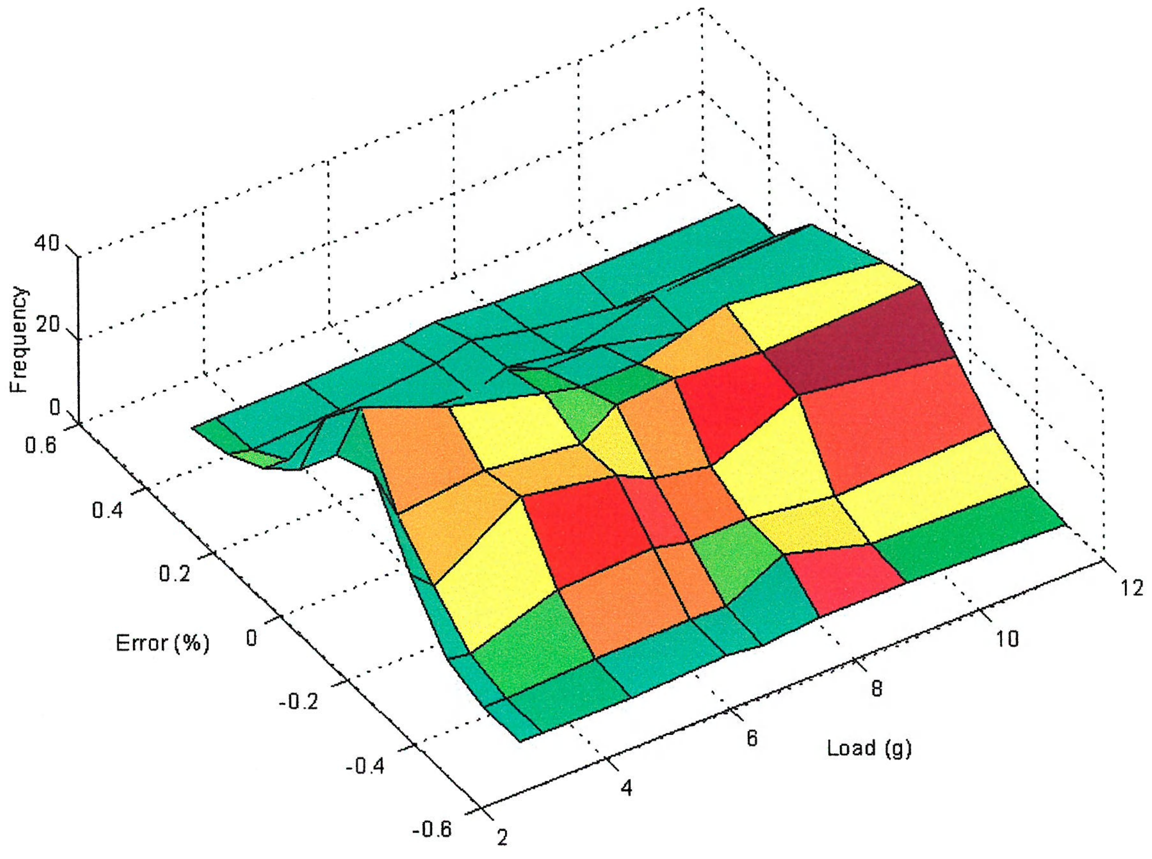


Figure 5.1: The distribution of the maximum error of a trained network is nearly a normal distribution with expectation equals zero.

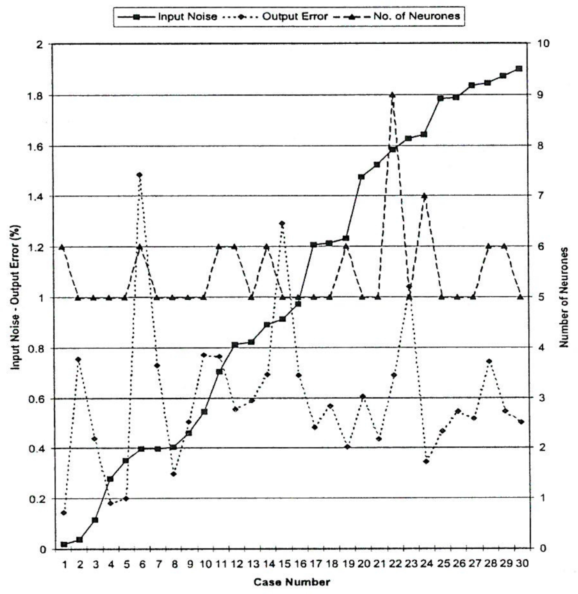


Figure 5.2: The effect of input noise on the RBF network architecture and the output error when *two* features of the signal are used.

shown in figures 5.2 to 5.10. These figures show if the level of the input noise is less than 2 percent, then only nine neurones in the hidden layer are needed to keep the maximum output error less than 1.5 percent. They also show that increasing the number of features from 2 to 10 slightly improves the level of output error. The results in figure 5.9 is different from others in respect that the number of neurones in hidden layer remains fixed at 7, and also the output error for all networks is consistently low. For further investigation, first the training and testing set was changed, then the case number was increased to 50 but in both instances the same behaviour was observed. Then the number of training samples was decreased, the output noise increased but still it was nearly the same for all the cases, and also the number of neurones in the hidden layer was 7. This unusual behavior merits further investigation.

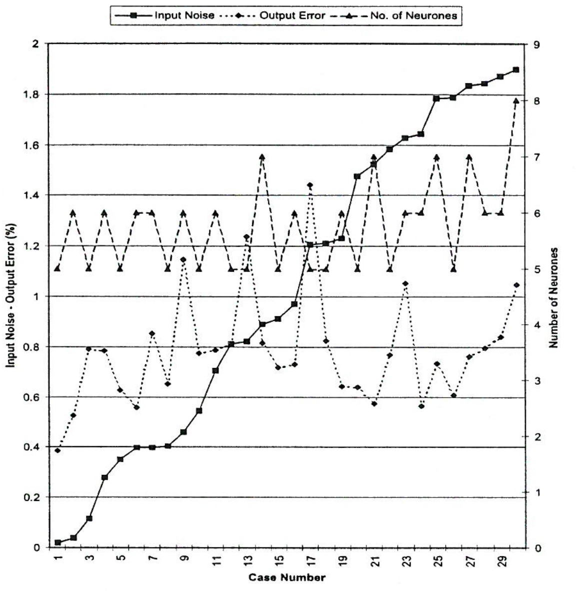


Figure 5.3: The effect of input noise on the RBF network architecture and the output error when *three* features of the signal are used.

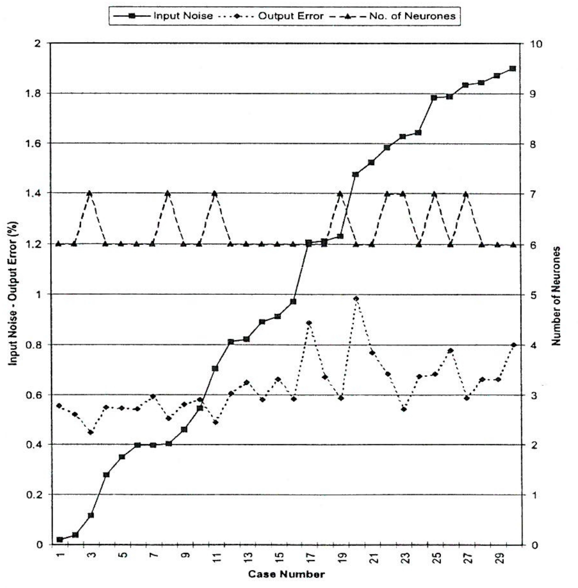


Figure 5.4: The effect of input noise on the RBF network architecture and the output error when *four* features of the signal are used.

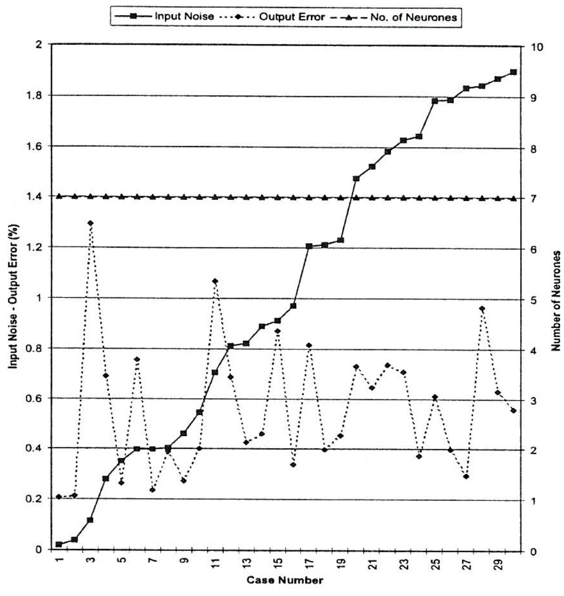


Figure 5.5: The effect of input noise on the RBF network architecture and the output error when *five* features of the signal are used.

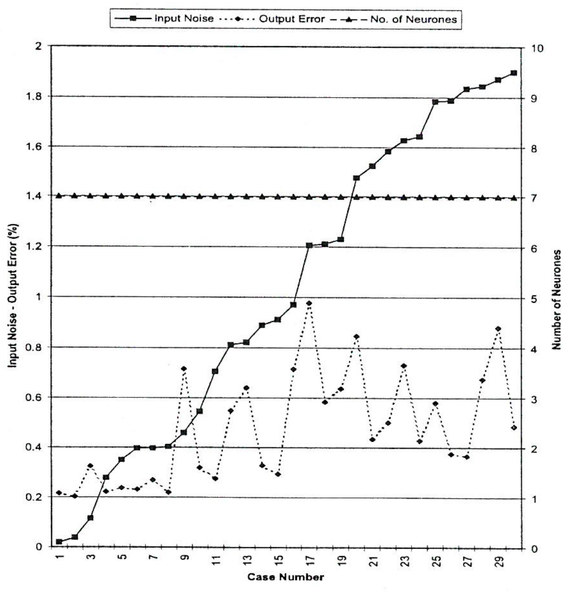


Figure 5.6: The effect of input noise on the RBF network architecture and the output error when *six* features of the signal are used.

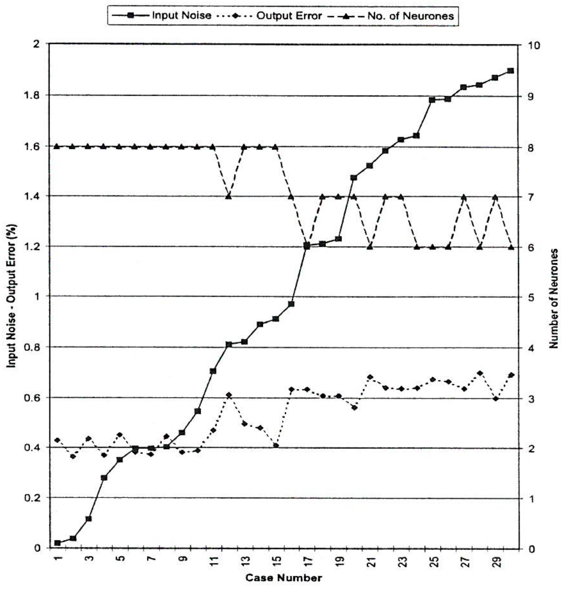


Figure 5.7: The effect of input noise on the RBF network architecture and the output error when *seven* features of the signal are used.

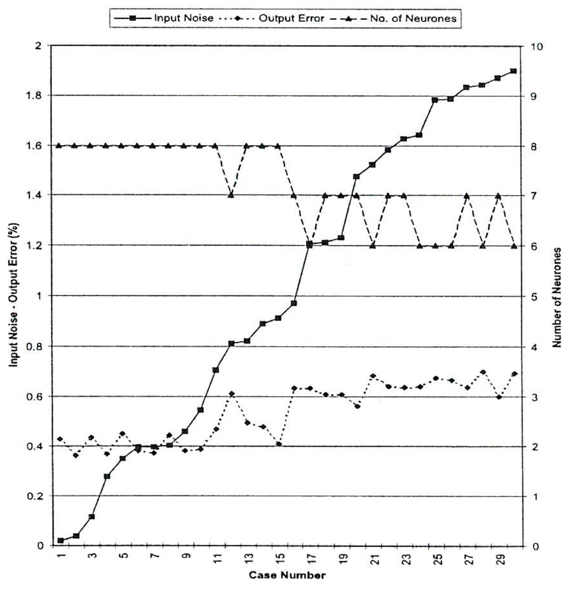


Figure 5.8: The effect of input noise on the RBF network architecture and the output error when *eight* features of the signal are used.

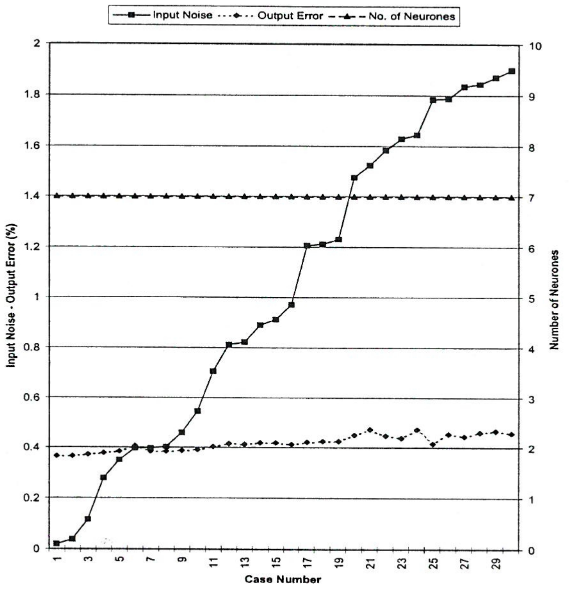


Figure 5.9: The effect of input noise on the RBF network architecture and the output error when *nine* features of the signal are used.

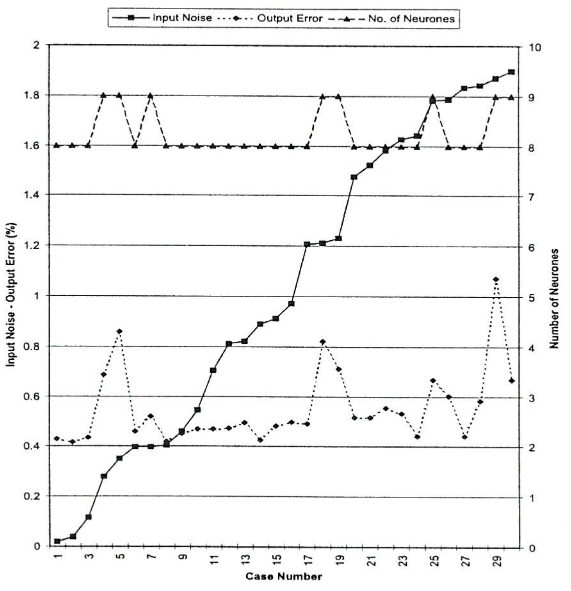


Figure 5.10: The effect of input noise on the RBF network architecture and the output error when *ten* features of the signal are used.

5.4 Simulation results Using MLP Configuration

The same numbers and procedure that were described in the section 5.3 for producing training and testing pattern were used to train and test Multi Layer Perceptron networks. For the RBF networks the number of neurone in the hidden layer is determined by the training procedure but for the MLP networks this number should be set by the user at the beginning of the training phase. Figures 5.7 shows the number of neurones in RBF networks with maximum of error less than 0.8 percent is seven. Therefore for the purpose of comparing the performance of two configuration the number of neurones in hidden layer for MLP configuration was chosen to be seven. In addition to investigate the effect of the number of features on the performance of the MLP networks the process is repeated when the number of neurones in the hidden layer is set to two.

5.4.1 Noise and Number of Features

To investigate the effect of noise random sequences were added to the signals. The distribution and the range of the numbers of the random sequence was same as described in section 5.3.1. Figure 5.11 and 5.12 present the results when respectively seven and two features of the signal were used. They show if the level of the input noise is less than two percent then MLP networks are able to find the impulse response with an error less than 0.8 percent.

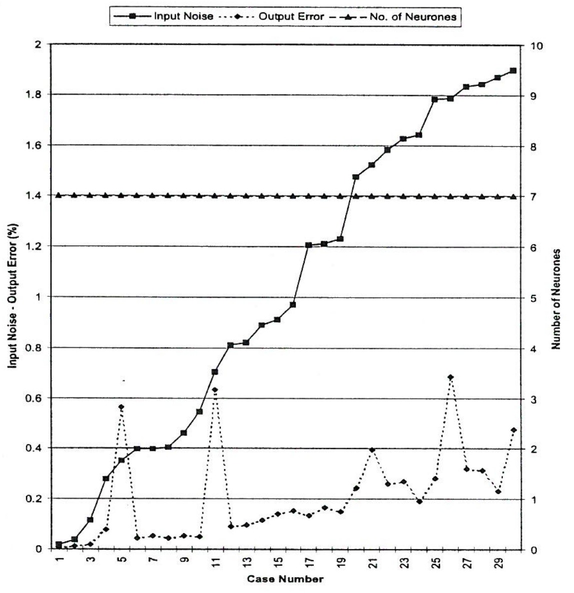


Figure 5.11: The effect of input noise on the MLP network architecture and the output error when *seven* features of the signal are used.

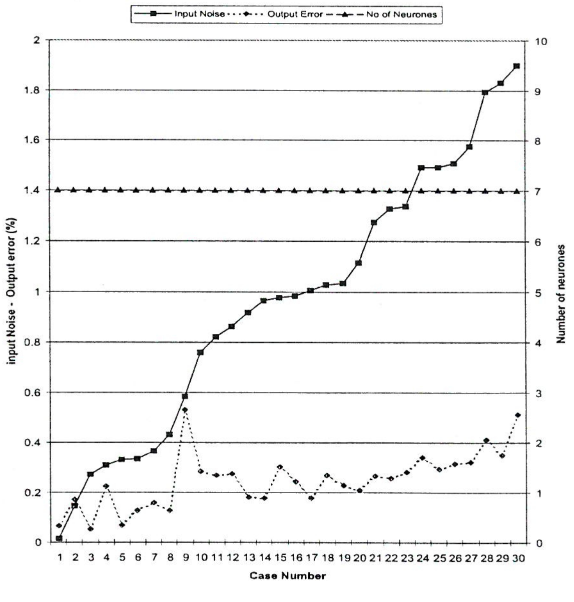
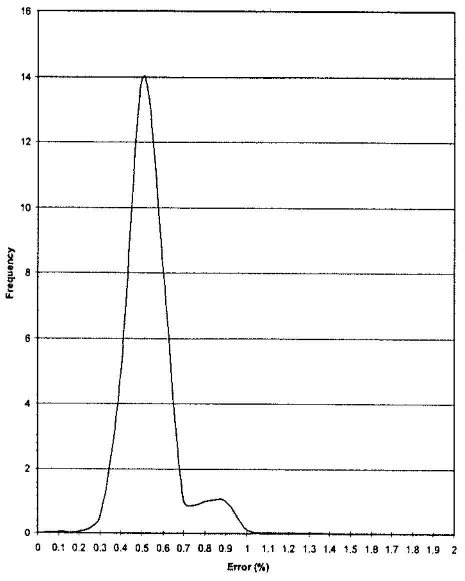


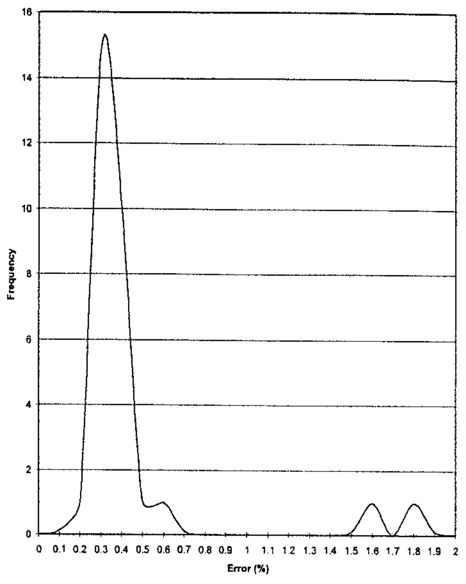
Figure 5.12: The effect of input noise on the RBF network architecture and the output error when *two* features of the signal are used.

5.5 Comparison Between RBF and MLP Configurations

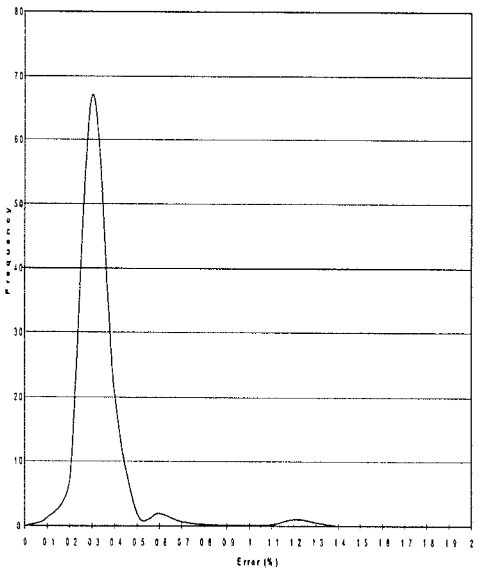
Figures 5.7 and 5.11 show the result of RBF and MLP networks respectively when they were trained and tested under the same conditions. The networks from both configurations have the same number of neurones in the hidden layer. The other important factor is the output error. For RBF networks the maximum magnitude of error for all thirty cases is less than 0.6 percent but for MLP networks in two cases the maximum exceeds from 0.6 percent. So it appears that the RBF configuration has a better performance. Nevertheless the maximum error for 60 percent of the MLP networks is less than one percent while none of the RBF networks show such performance. This means that if the MLP configuration was chosen, it is quite likely to yield a network which produces a very low output error although there is a low risk that the training algorithm creates a network which could produce an output error more than the RBF networks could produce. To investigate this issue, one hundred RBF networks and one hundred MLP networks were trained. The level of noise was set to two percent and seven features of the input signals were used. Each network was tested with thirty signals. The maximum errors produced by the networks were recorded. Figure 5.13(a) shows the smooth line histogram of the maximum error produced by the RBF networks. The results for MLP networks is represented in figure 5.13(b). The main lobe of both histograms is like a normal distribution. The mean and variance of the main lobe for MLP networks is less than the one for RBF networks. This means that choosing the



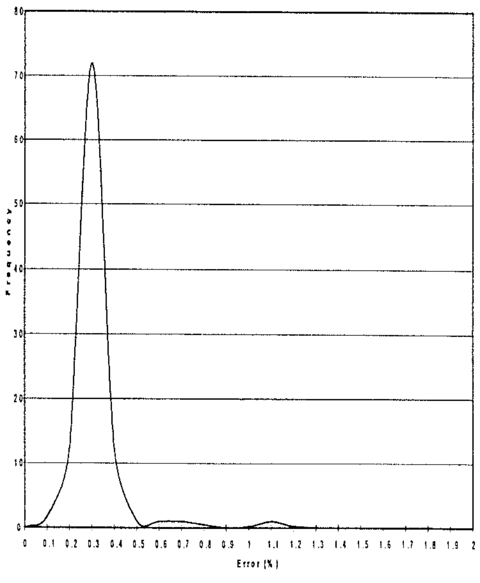
(a) RBF networks, 7 features.



(b) MLP networks, 7 features.



(c) MLP networks, 9 features.



(d) MLP networks, 11 features.

Figure 5.13: The smooth line histogram of maximum error when the level of noise was 2%.

MLP configuration network increases the probability of obtaining a network with high accuracy. While the probability that the maximum error for a RBF network is less than or equal to 0.3 percent is zero. A considerable number of MLP networks satisfy this condition. The maximum error of 94% of the MLP networks is less than 0.5 percent compared to half of the RBF networks. On the other hand, the worst RBF network generates less than one percent error, which is nearly half of the amount of that produced by the worst MLP network. Nevertheless the probability of getting a MLP network to produce a high error is low. This probability can be reduced further if more features are used. Figure 5.13(c) shows the results when nine features are used, but other conditions do not change. The side lobes dramatically attenuate and also shift to a lower error. There is a limit on improvement of the histogram using more features. This can be concluded by comparing figure 5.13(c) with figure 5.13(d) which shows the results of using nine and eleven features respectively.

5.6 Choice of Training Pattern

The previous networks are trained by a total of 360 signals. These signals were generated by choosing 12 points uniformly distributed over the input range and producing thirty signals for each point. If the number of the points reduces then the process of gathering experimental data, and also the training process, speeds up. To investigate this issue the number of points is reduced to six that were uniformly distributed over the range of input. The level of noise was set to 2 percent and seven features of the input signal were used. Thirty signals were produced for each point. Figures 5.14 and 5.15 show the results for RBF and MLP networks respectively. They are disappointing because the maximum error for all of the RBF networks and most of the MLP networks is more than two percent. If these two latter figures are compared with figures 5.13(a) and 5.13(b) respectively, it shows how reducing the number of training points can result in poor networks. In certain circumstances, increasing the number of input features can improve the performance of networks. Hence, under the same conditions the number of features were increased to eleven and the process of testing and testing networks were repeated. The results are illustrated in Figure 5.16, revealing that the increase of input features can not compensate the decrease of the learning points.

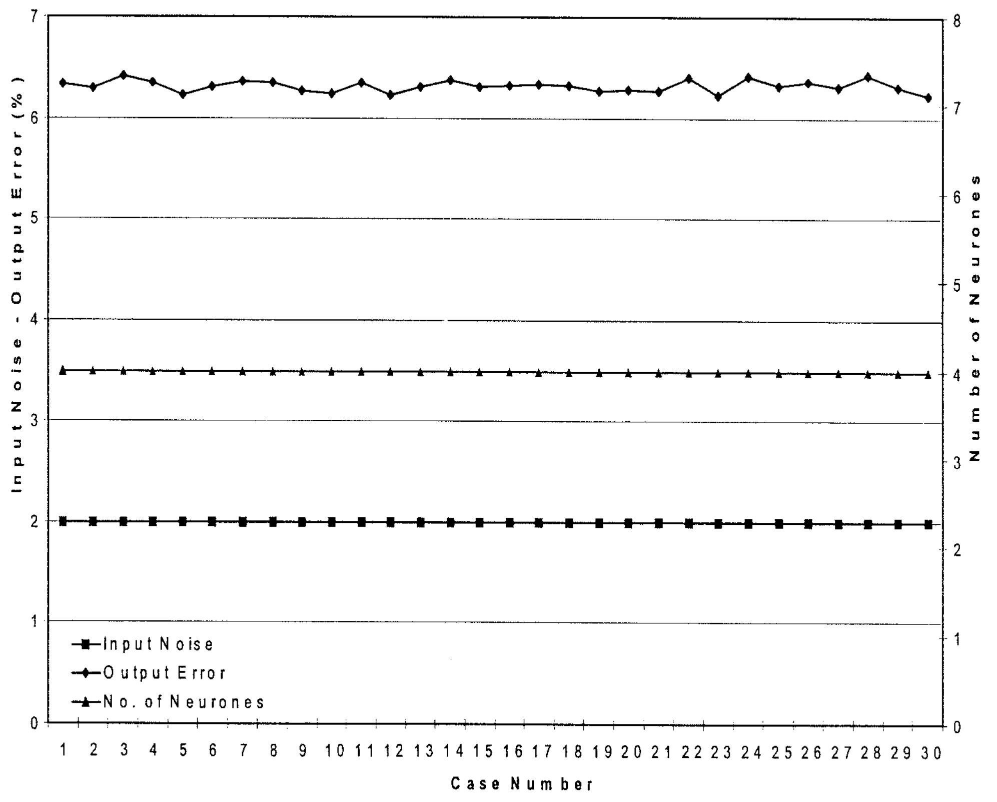


Figure 5.14: Effect of decreasing the number of training points on RBF networks.

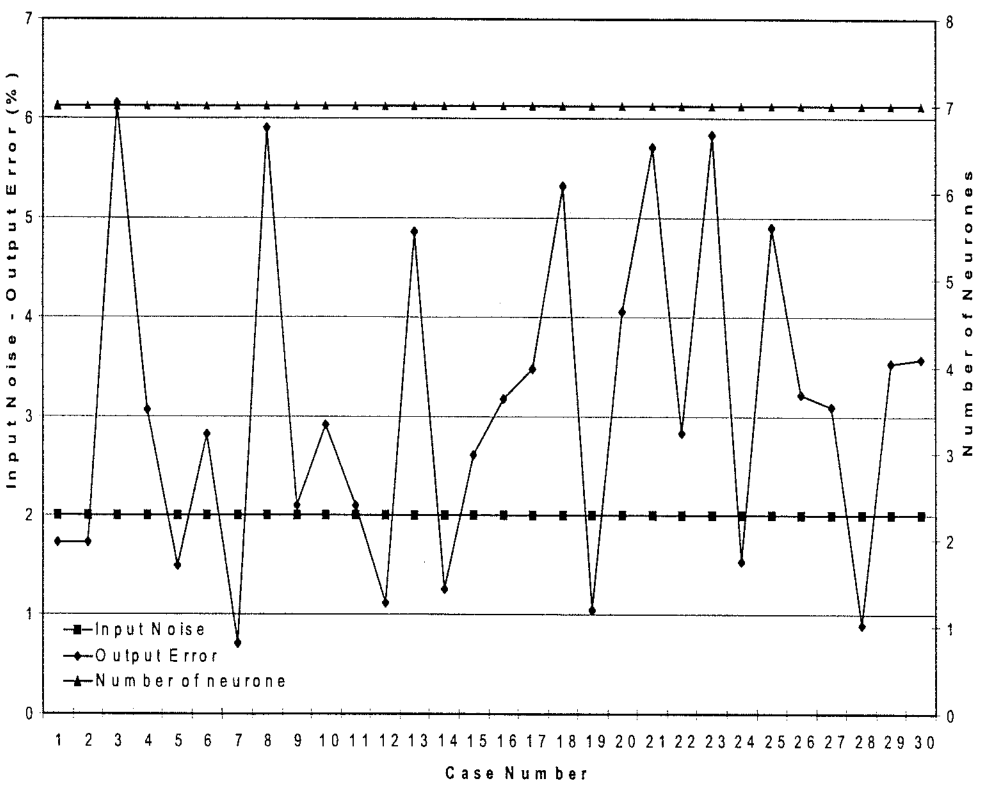


Figure 5.15: Effect of decreasing the number of training points on MLP networks.

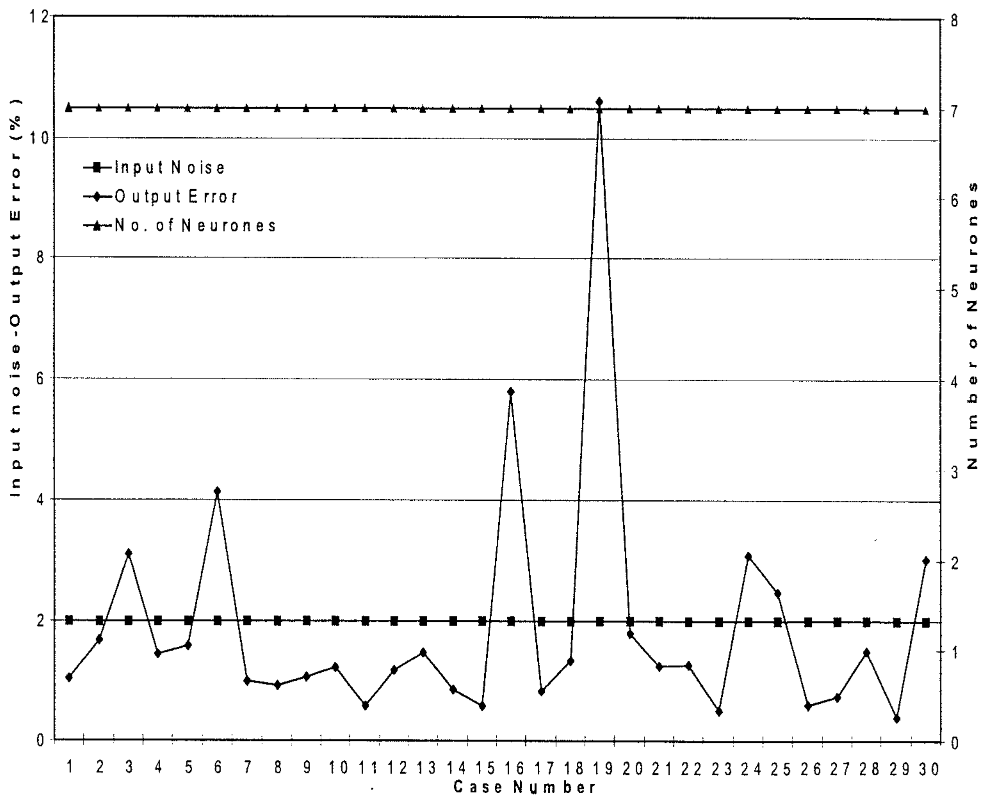


Figure 5.16: Increasing the number of features can not compensate the decrease of training points.

5.7 Output Deviation

Each measurement system is designed to work under specific conditions such as input range, ambient temperature and so on. If it is used beyond the designated ranges then the output is not reliable and could be far from the real value. It is expected, however, that if an input, in the allowed range, is disturbed by the noise or other unwanted factors then the output consistently deviates from the actual value. In other words, if the input signal's discrepancy is minor then the output deviation is small and vice versa. To investigate the capability of the neural network method in this respect, a network for the range of 1 to 12 g was designed to function when the level of noise was two percent or less. Then one hundred signals for a 6 g load were produced. These signals were contaminated with noise. The amplitude of noise changed from two to fifteen percent of the maximum of the sensor's output for the 6 g mass. The signals were applied to the network. Figure 5.17 shows the level of input noise and the output error. It reveals that the output of network is highly erratic and is not related to the input when the input noise exceeds the allowed range. This cannot arise from the neural network alone. Its response to signals with less than two percent noise confirm that it correctly learnt the behaviour of the mapping function, and if the load changes around 6 g the response changes accordingly. Therefore the effect of high level noise is not just changing the level of input to the network. Figure 5.18 shows a noisy signal. The preprocessing module detects the amplitudes of peaks and troughs of the sensor's signal as the features of the signal that passed them to the neural network for processing. The small squares in figure 5.18 show the nine first features of the high

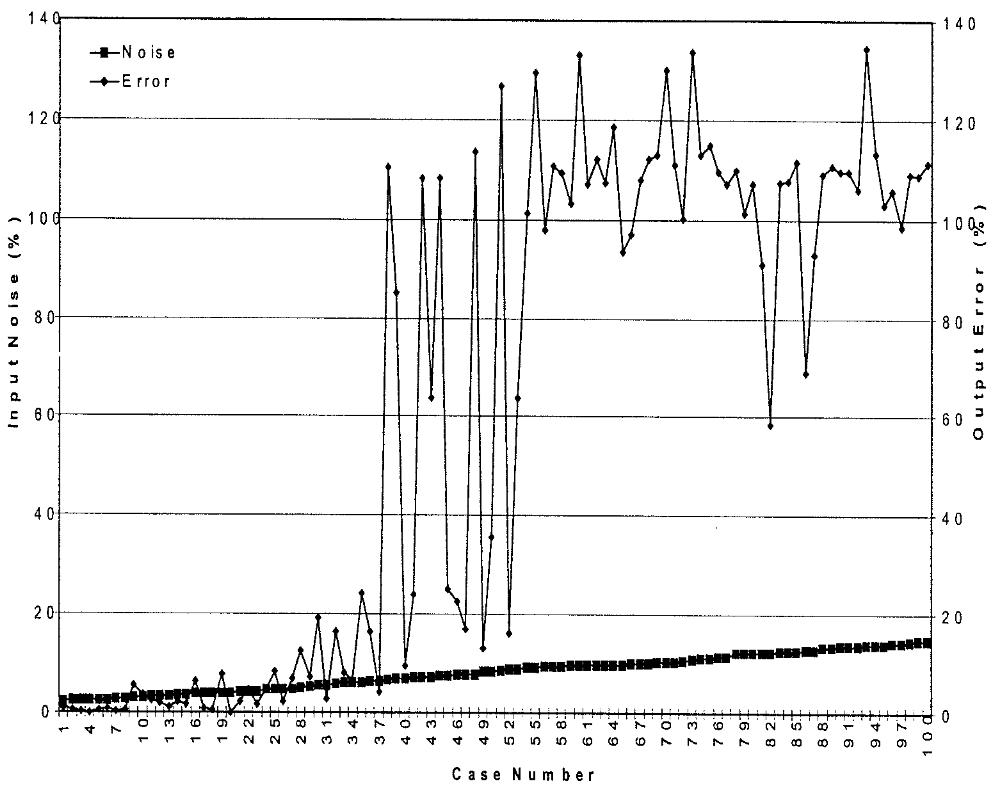


Figure 5.17: Output error when the input noise exceeds from the allowed level and the signal is not filtered.

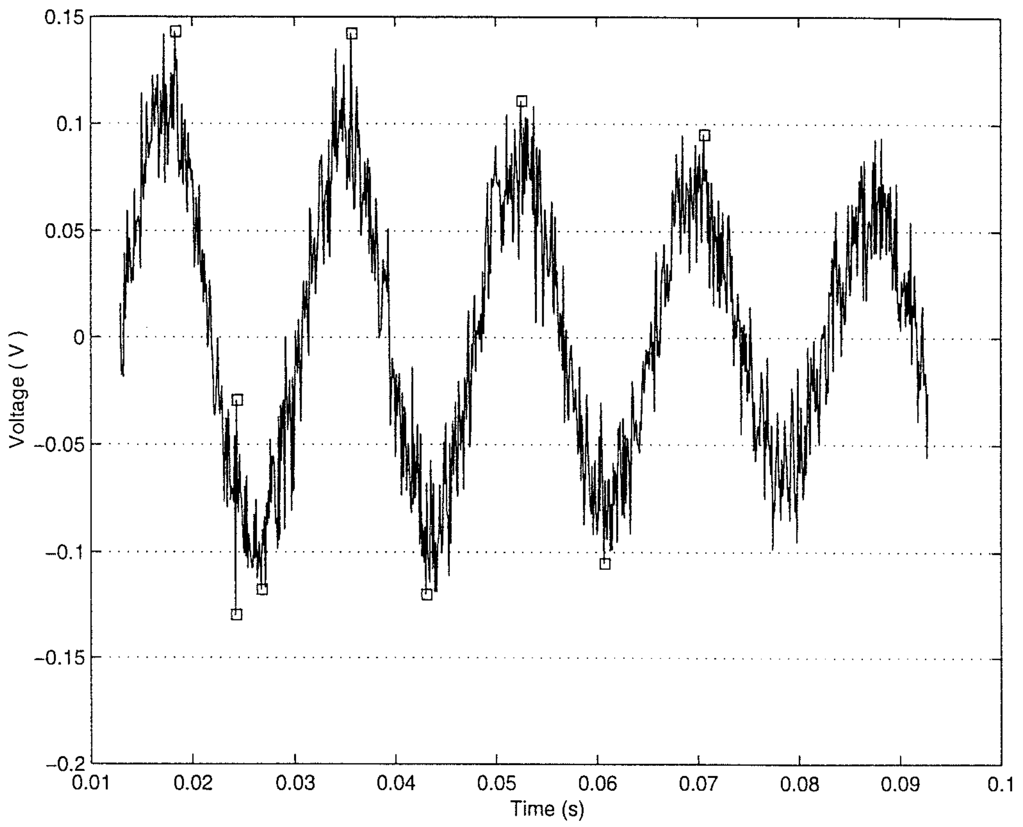


Figure 5.18: Noisy signal.

noisy signal detected by preprocessing module. The high level of noise makes such fake peaks and troughs that leads the preprocessing unit to confuse in detecting the right peaks and troughs. To prevent this problem, the algorithm for finding peaks and troughs was changed by defining the time windows that each of them contained one period of the sensor signal. Then the maximum or minimum points in each window were found. Figure 5.19 illustrates the result of applying one hundred test patterns. Comparing this figure with the previous figure shows that a considerable improvement has been accomplished. In the next step before feature extraction, a Finite Impulse Response (FIR) filter of order 50 is added to reduce the power of noise. The frequency response of the filter is illustrated in figure 5.20. Figure 5.21 shows the high noisy signal after low pass filtering and the peaks and troughs which are founded by feature extraction procedures. The result of repeating the previous experiment illustrated in figure 5.22 show that the erratic behavior of the output disappeared completely.

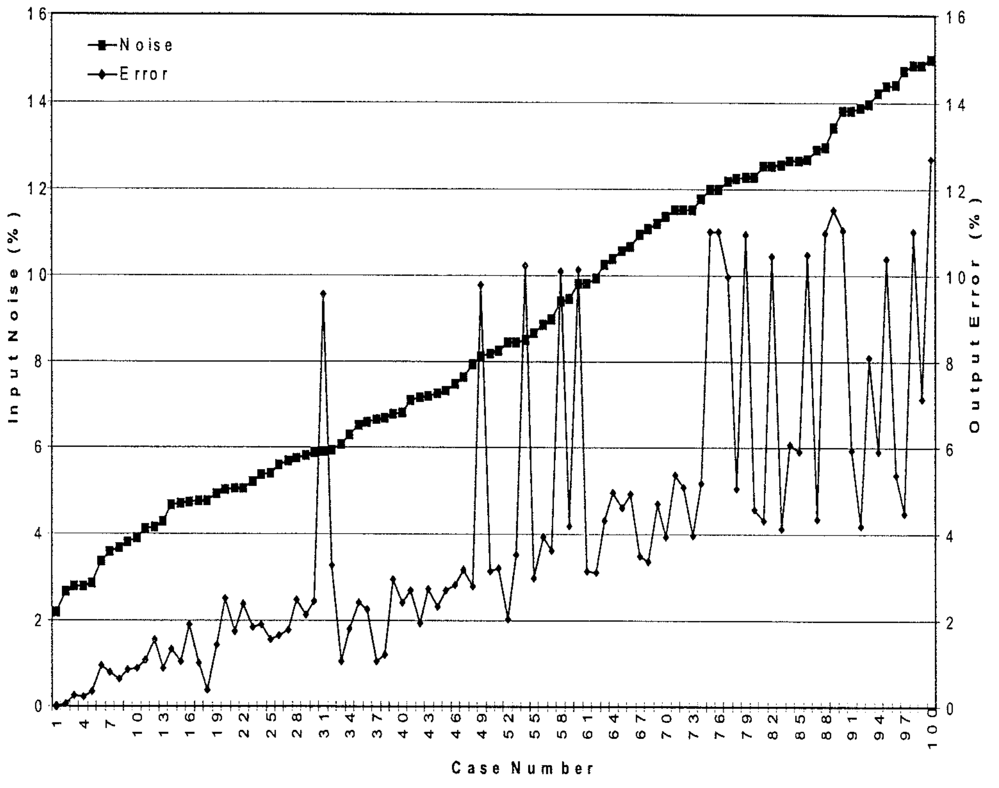


Figure 5.19: Output error when the input noise exceeds from the allowed level and the windowing procedure were used for features extraction.

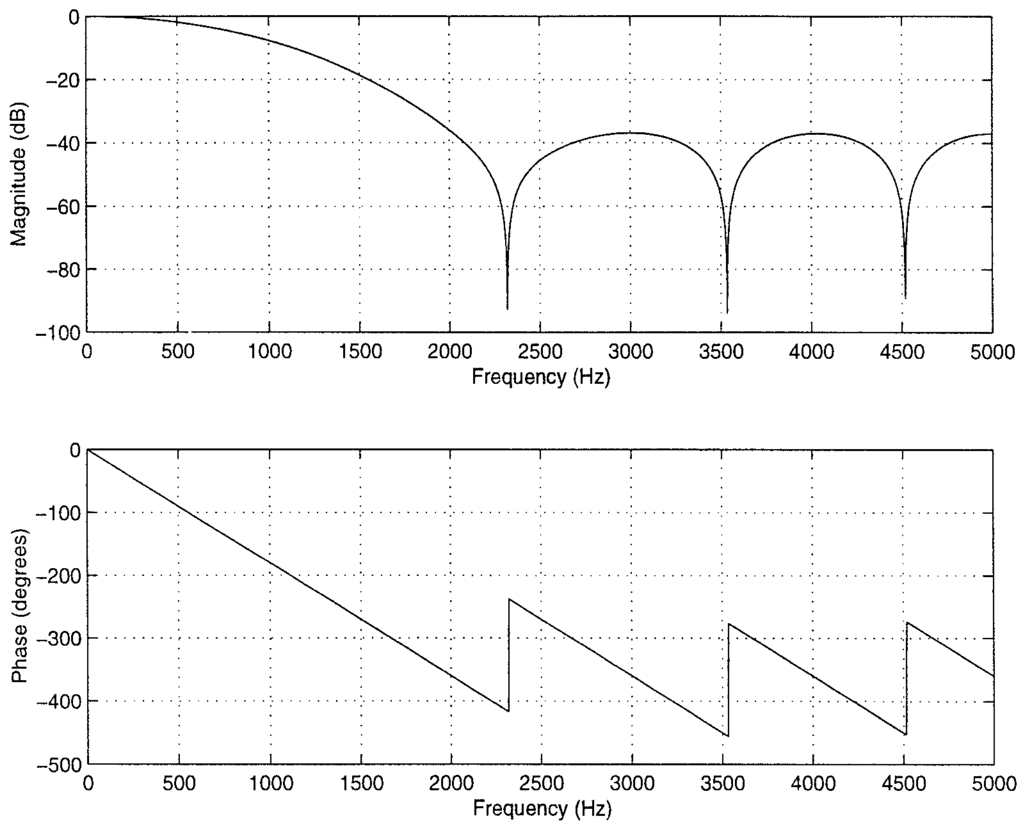


Figure 5.20: Frequency response of the low pass filter.

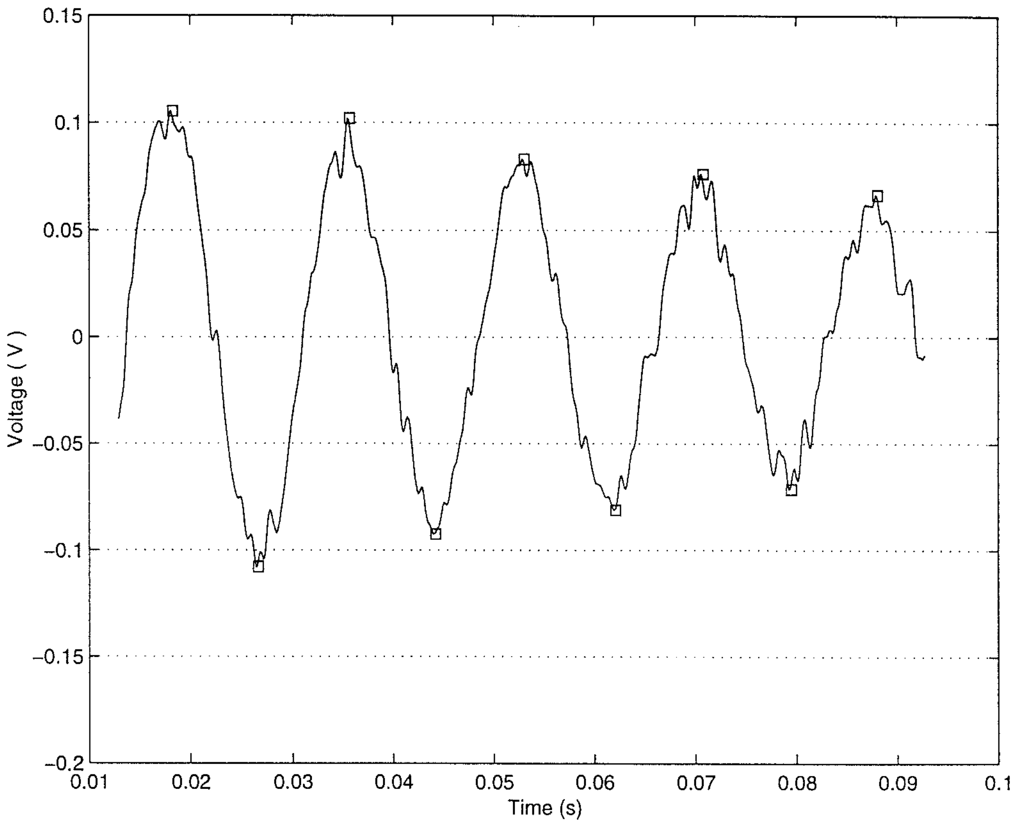


Figure 5.21: A noisy signal after filtering. The squares shows the features extracted by preprocessing unit. The windowing technique was used.

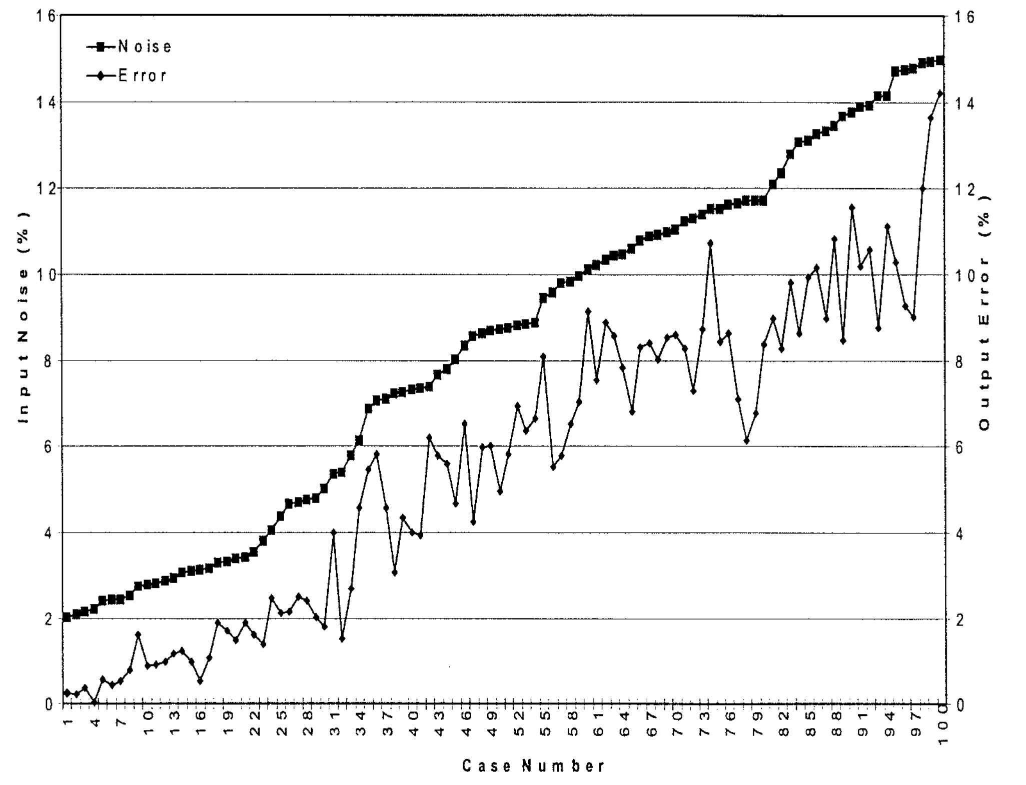


Figure 5.22: Output error when the input noise exceeds from the allowed level. Filtering and windowing technique maintain the consistency of the output.

5.8 Experimental Setup

Figure 5.23 shows the system that was set up to obtain the impulse response of the tri-beam load cell. To stimulate the sensor by an impulse input, the load should contact the transducer for a very short period. Therefore a pillar was added to the weigh pan. The pillar is made from polyvinyl chloride (PVC) and its diameter is 31 mm . It is fixed to the centre of the weigh pan. The top of the pillar is diagonal and its slope from the horizontal is ten degrees. Hence when a load falls on the top surface of the pillar it is immediately thrown away and a short impact is generated. A ramp was set up to carry the loads from a fixed height to the top of the the pillar. From the end of reel, loads freely drop on to the pillar surface. The loads are disks of metal with different diameters, and are made from aluminum and bronze. The response of the sensor to the loads is a very short period signal which is produced after collision. To capture this signal, a data acquisition card `AT_MIO_16E_10` manufactured by National Instruments was used. The necessary software is written in LabWindws/CVI (version 5.5) environment. Data acquisition needs to be initiated immediately after the load hits the sensor. An infra red transmitter and receiver [70, 71] was situated on the route of loads in the ramp and produced a triggering signal for the start of data acquisition. The data acquisition card does not have hardware triggering facility so the software monitors the triggering signal and starts data capturing at appropriate time.

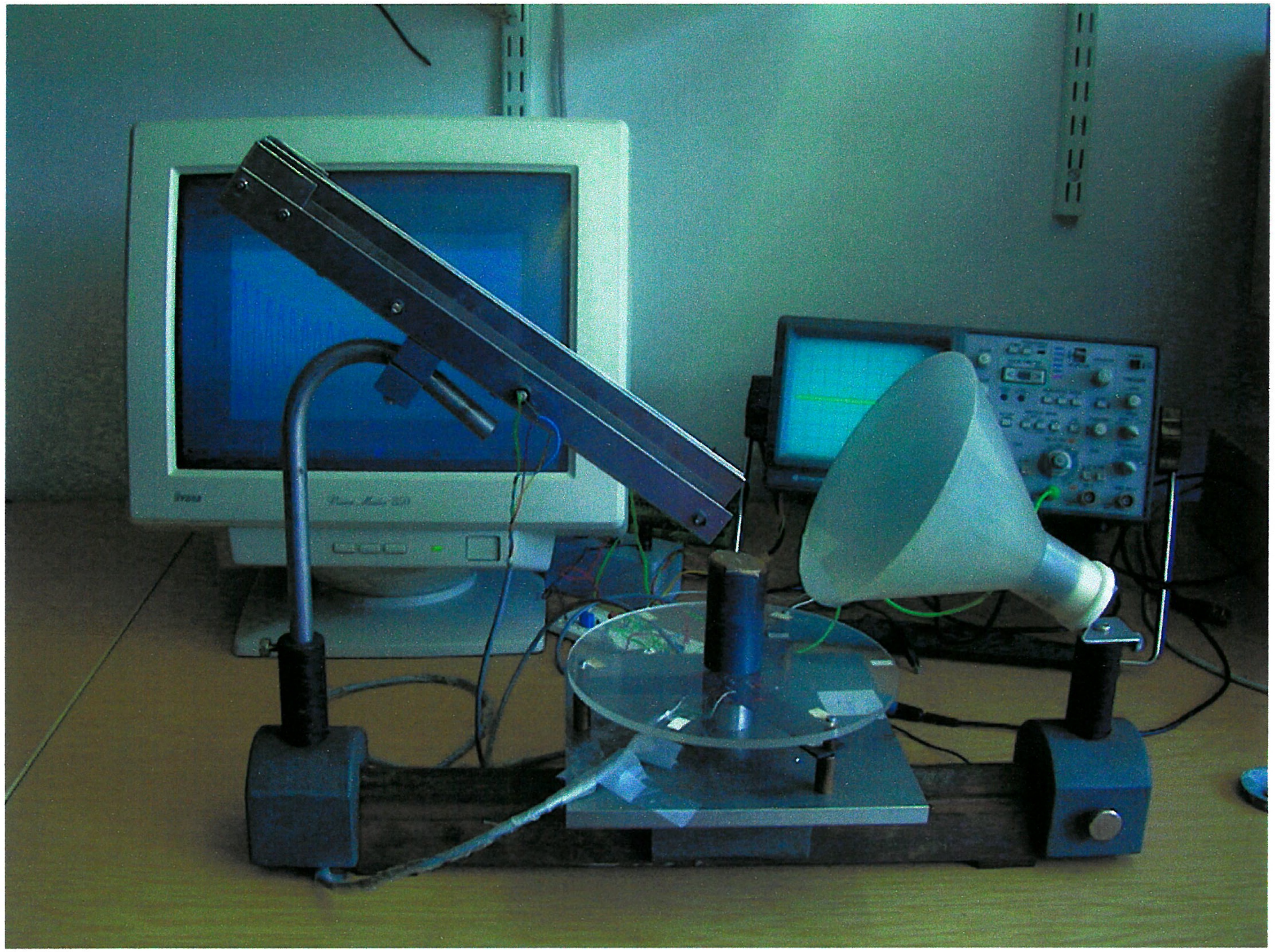


Figure 5.23: The experimental set up.

5.9 Experimental Consideration

The following list gives several experimental considerations that need to be adhered in order to achieve accurate and repeatable signals:

- The height and slope of the ramp, and also its position against the sensor should remain constant during the experiments.
- This is necessary to minimize the friction between the reel and the loads. During the experiments it was revealed that covering the surface of the route with a solid plastic material improves the repeatability of signals.
- It is essential that the load starts moving toward the sensor from a fixed point.
- The width of the route in the ramp should match the width of disks, otherwise the disk bounces between the wall of the route, which affects the repeatability of the signal.
- The impact of the load on the sensor causes all of the loose parts around the experimental set up to vibrate. This produces unwanted signals that contaminated the desired sensor signal. Therefore the weigh pan and the pillar should be firmly fixed to the rest of the sensing structure. Also the whole set of the system should be put on a mounting which does not shake easily and can damp the oscillation of the apparatus.

5.10 Experimental Results

Figure 5.24 shows a typical impulse response of the sensor. The details of the

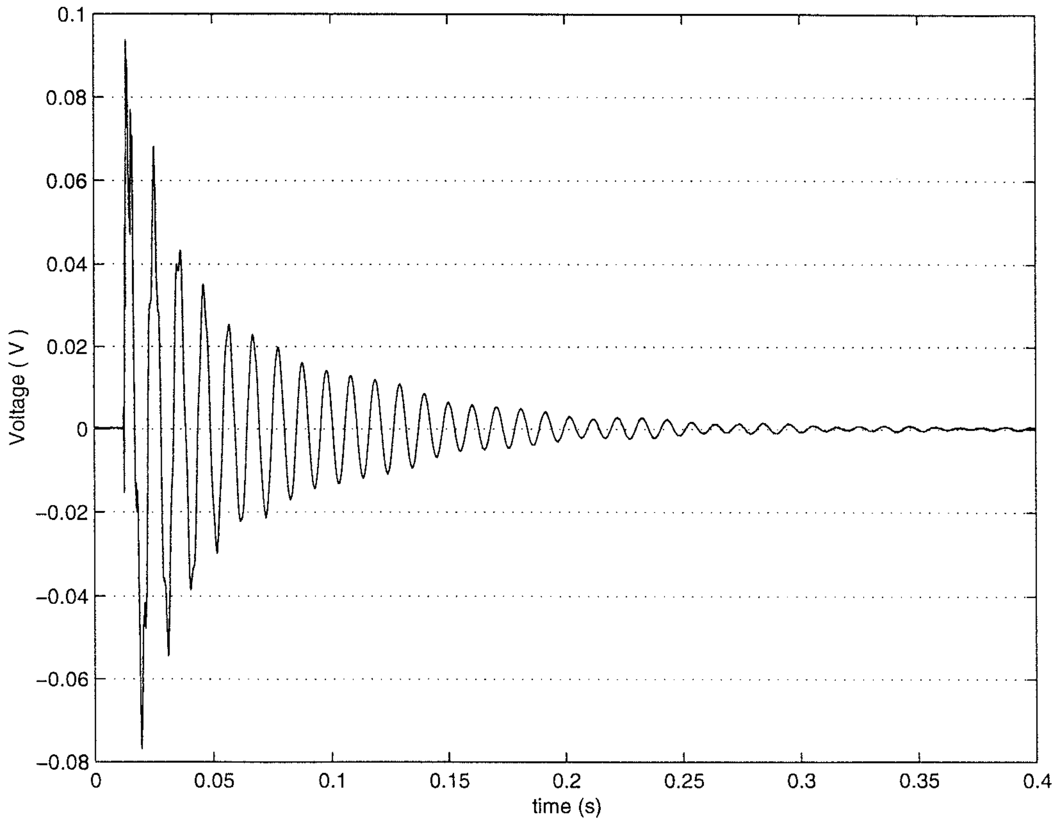


Figure 5.24: A typical experimental impulse response.

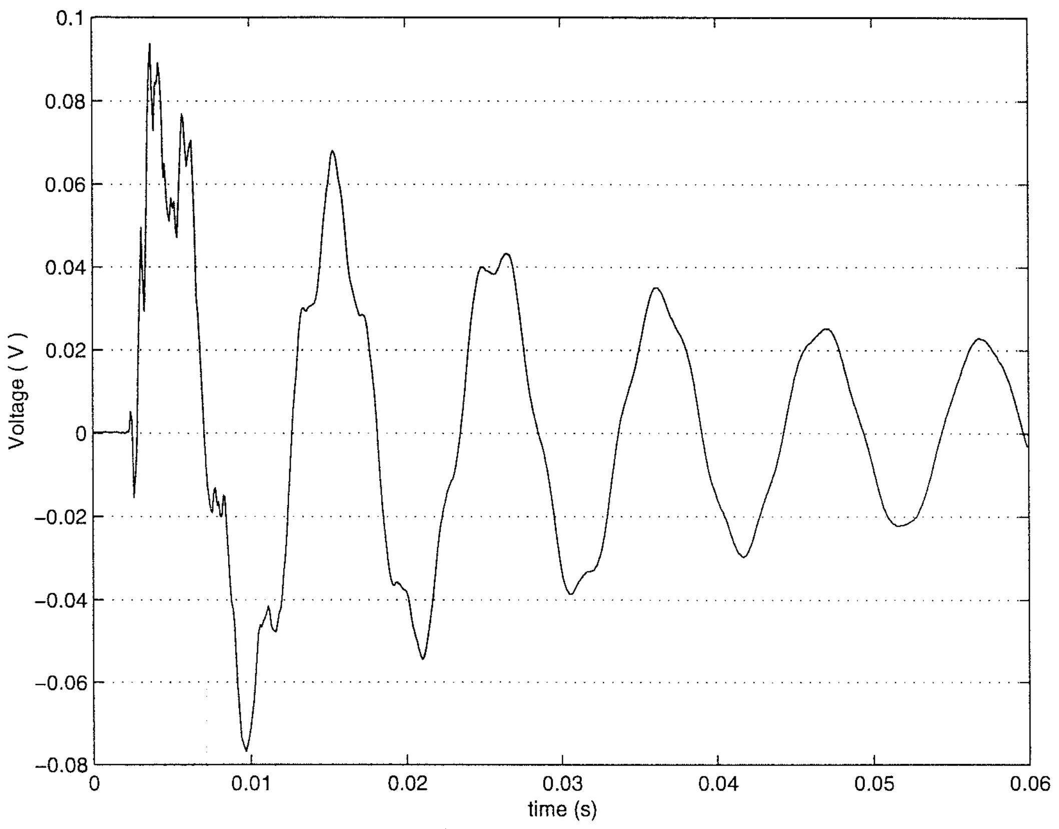


Figure 5.25: The details of first peak and trough of the impulse response which is illustrated in figure 5.24.

first peak and trough can be seen in figure 5.25 which shows just a small part of the previous figure. The signal is noisy and before feature extraction it is filtered by the low pass FIR filter whose frequency response has been shown in figure 5.20.

Six loads were used for obtaining experimental data. They were disks with the same width but different diameter and material. Their masses were 4.26, 5.26, 6.26, 7.26, 8.26, 9.26 g respectively. These were measured by an Oertling weighing balance with an accuracy 0.01 g. For each load, fifty signals were captured. Thirty of them were used for training purpose and twenty of them for testing the trained network. Based on the results of simulations, a MLP network with seven neurones in the hidden layer was chosen for training. Also the first peak and trough of the signal were extracted as two features to feed the neural network. The result is illustrated in figure 5.26. The mean of the error signal is not zero. This is due to the amplifier drift and the fact that the samples for training and testing were captured in different sessions. Nevertheless, the maximum error is less than 0.8 percent. The experimental results therefore support the simulation results and confirm that the neural networks can be successfully employed in impulse response measurement.

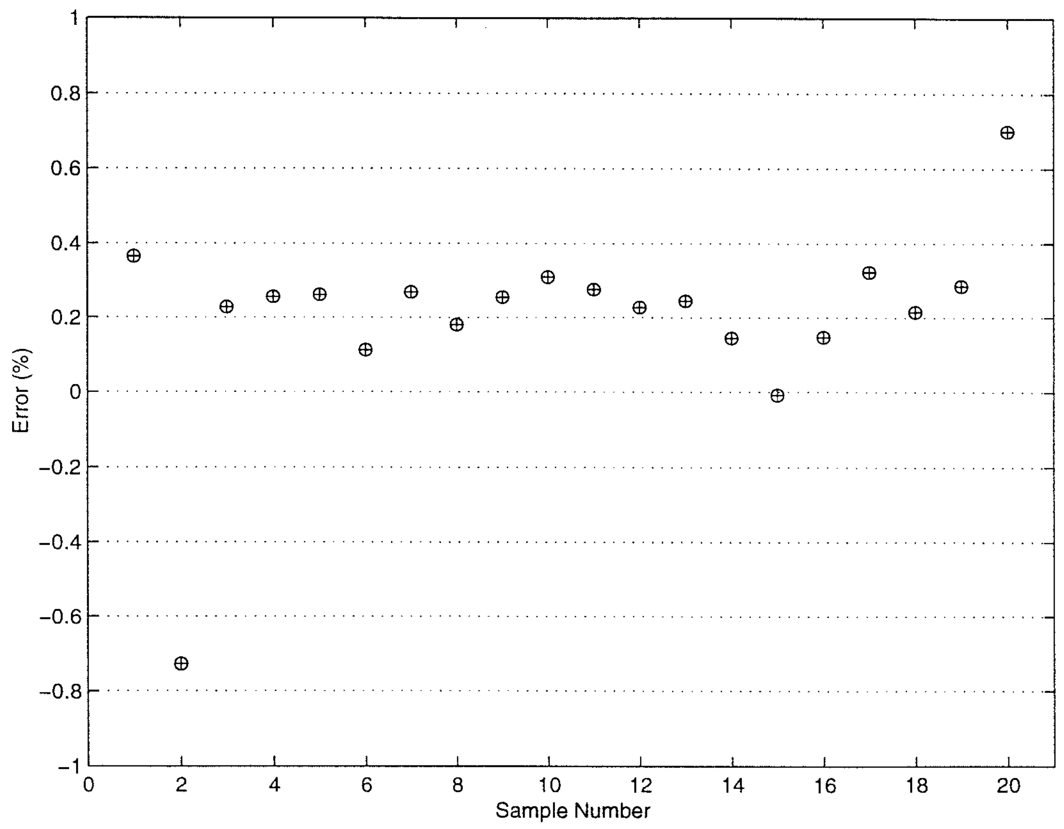


Figure 5.26: The result from experimental data.

Chapter 6

Interference Cancellation

6.1 Introduction

Unwanted signals that contaminate the desired sensor signal can be classified in two categories: The first category is that of high frequency noise, which is mainly produced by the internal process of the sensor and the electronics used for signal conditioning. The frequency band of this type of noise is generally out of the frequency band of the main signal, hence it can be removed by an appropriate low pass filter as discussed in the previous chapter. The other category consists of interference signals that are produced by the devices and sources around the sensor. The effect of the 50 Hz mains 'hum' is a an example of such noise. Another example is the effect of shaking in a weighing system. The electro-motor and other equipment that are needed to rotate the belt cause the load cell sensor to oscillate and produce an unwanted signal.

An interference signal can be modeled as the effects of a disturbance source that produces another input to the sensor as shown in figure 6.1. The frequency band of the interference signals are in the range of the desired signal and

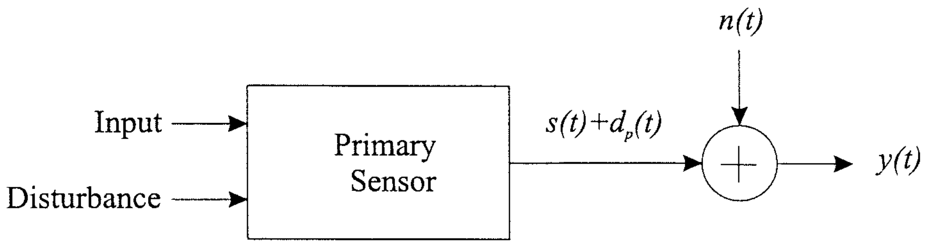


Figure 6.1: Block diagram of sensor system with an interference signal.

usually vary as a function of time. Consequently standard filtering method can not be used and adaptive methods are required. Adaptive methods often achieve a degree of noise rejection that would be difficult or impossible to achieve by direct filtering [36].

6.2 Simulation of the Interference Signal

The interference sources which have a repetitive pattern are considered in this section. Their effect on the output of the sensor is essentially a periodic signal. According to the Fourier theorem periodic signals can be made by combination of a series of harmonics of sine and cosine waves. Four different interference signals were simulated. Figure 6.2 shows the samples. The first one is 70 Hz sine wave. The others are combinations of the first signal and its first three harmonics. They are produced by choosing different amplitudes for the components.

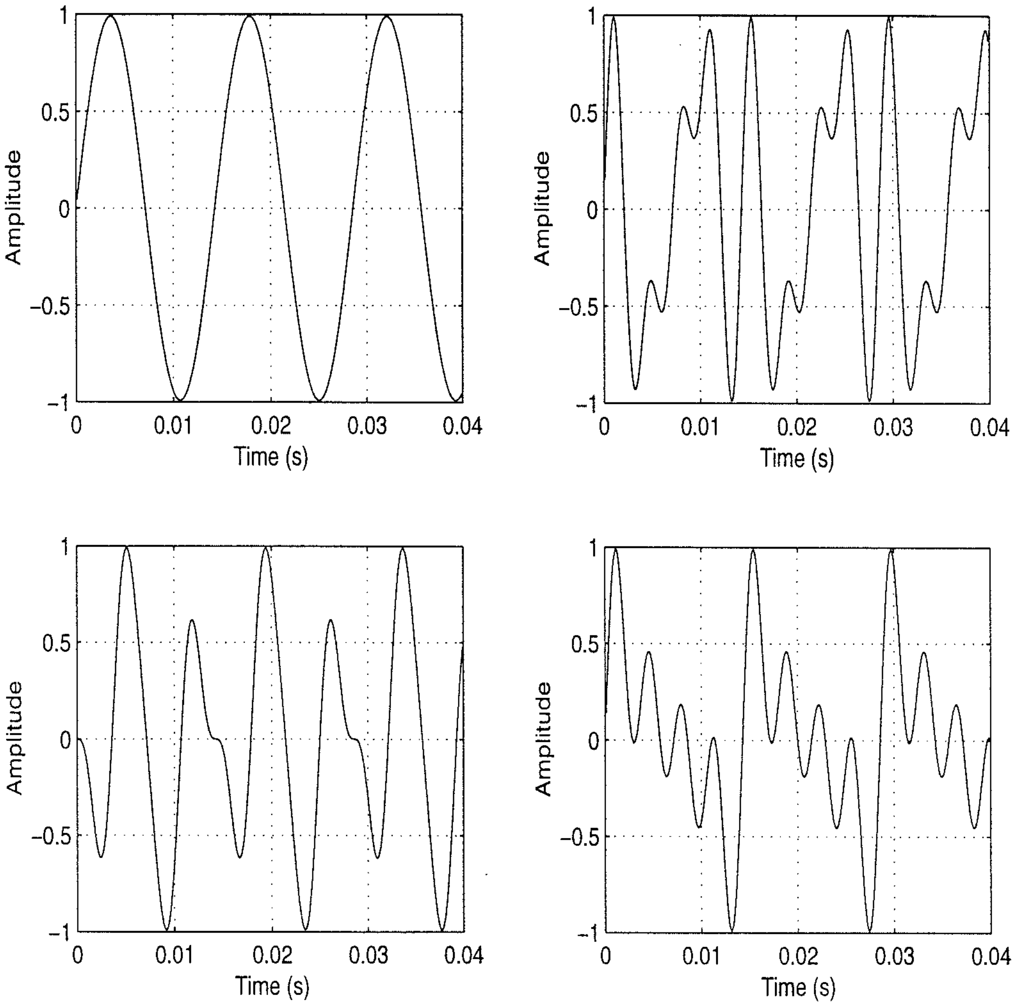


Figure 6.2: Interference signals.

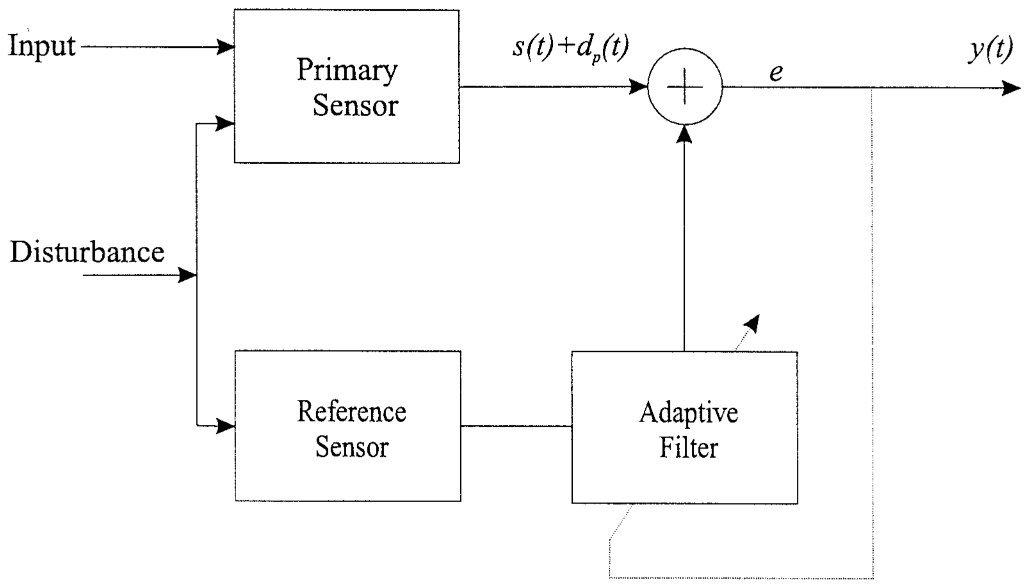


Figure 6.3: Adaptive method for the cancellation of interference signal.

6.3 Adaptive Filter Method

Adaptive filtering is a variation of optimal filtering that has been successfully used in many applications. This method is based on using an extra or reference sensor located at a point that just picks up the interference signal. The output of the reference sensor processed and subtracted from the output of the primary sensor containing both signal and interference. As a result, the the interference signal is attenuated or eliminated by cancellation. Figure 6.3 shows a block diagram of the method. The high frequency noise does not show in this block diagram for the purpose of simplicity and because the previous chapter describes how its effect can be eliminated by a low pass filter. The signal $d_p(t)$ is the effect of the disturbance source in the output of the primary sensor. The adaptive filter adjusts its elements to minimize the error signal e. Since its target signal is $d_p(t) + s(t)$, it tries to produce this signal, but it only

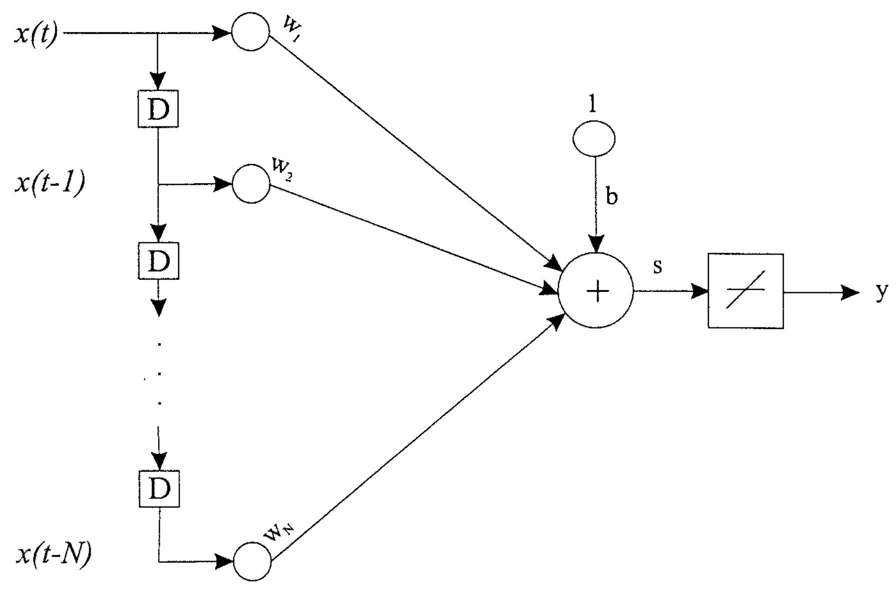


Figure 6.4: Combination of a MLP neural network and a tapped delay line as an adaptive linear filter.

knows about the interference signal and can reproduce $d_p(t)$ that is linearly correlated with the interference signal. The signal e equals the target signal minus the output of the network. In this way the signal e will be closest to the signal $s(t)$ which is the output of the primary sensor due to the real input.

A MLP neural network, which has a linear transfer function, can be combined with a tapped delay line to implement an adaptive filter as shown in figure 6.4. The input signal feeds the tapped delay line which provides an input vector for the neural network. The elements of the vectors are the input signal at the current time and at delays varying from 1 to $N-1$ time steps.

Figure 6.5 shows a signal contaminated with interference and the output of the system using an adaptive filter to cancel the interference. This figure shows that the adaptive filter method has not been able to cancel the interference signal. In this case the correlation coefficient between the sensor signal and

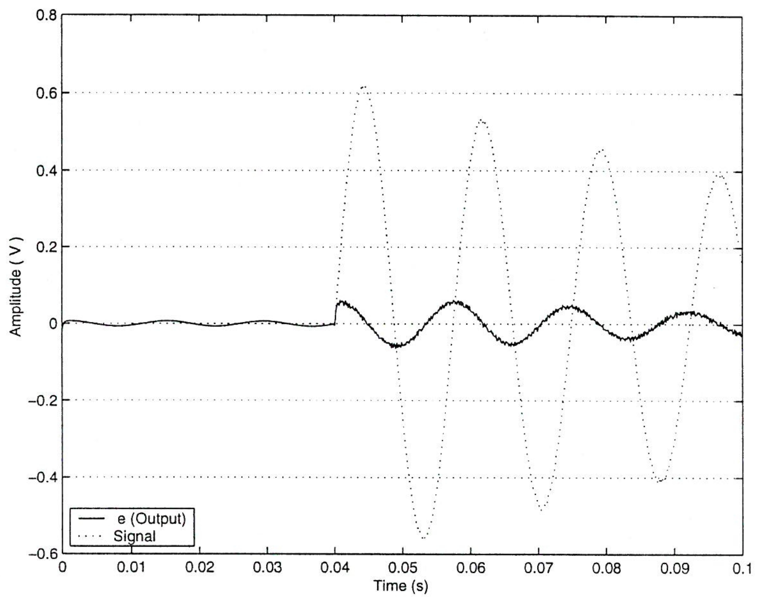


Figure 6.5: The error signal is not like the main signal because the main and interference signals are correlated.

interference signal is 0.19. The high degree of correlation between the sensor signal and interference causes the adaptive filter to fail to reproduce just the interference signal, so e is not close to the $s(t)$. In fact, the filter produces the signal plus interference as is revealed from the figure 6.6 which shows the output of the filter.

In the next section, a neural network is used in a novel way in order to cancel the interference signal irrespective of its correlation with the main signal.

6.4 Adaptive System Identification Method

Figure 6.7 shows the block diagram of the new method. It consists of two phases. Phase 1 which begins a short time before the main signal stimulates

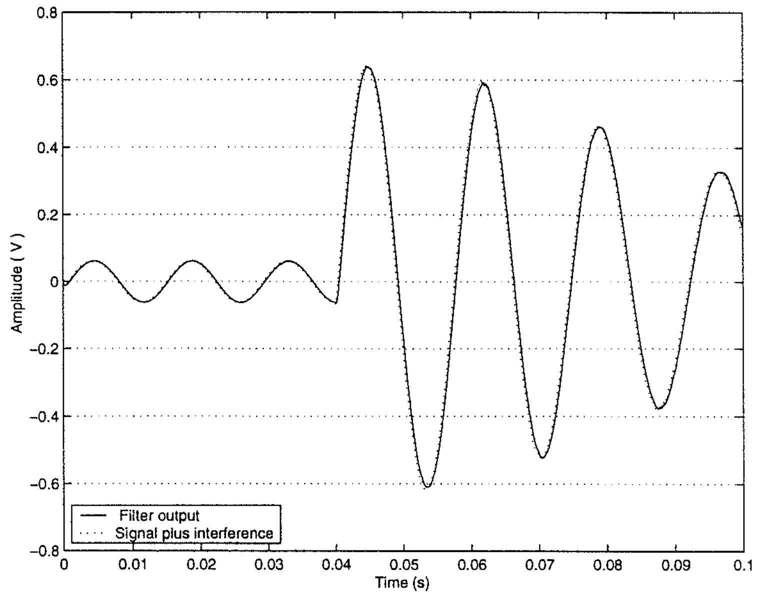
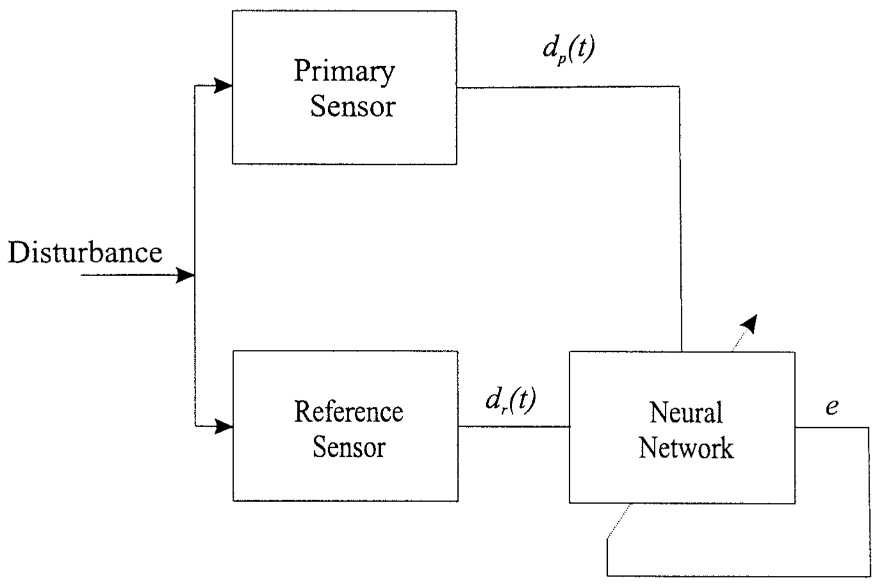


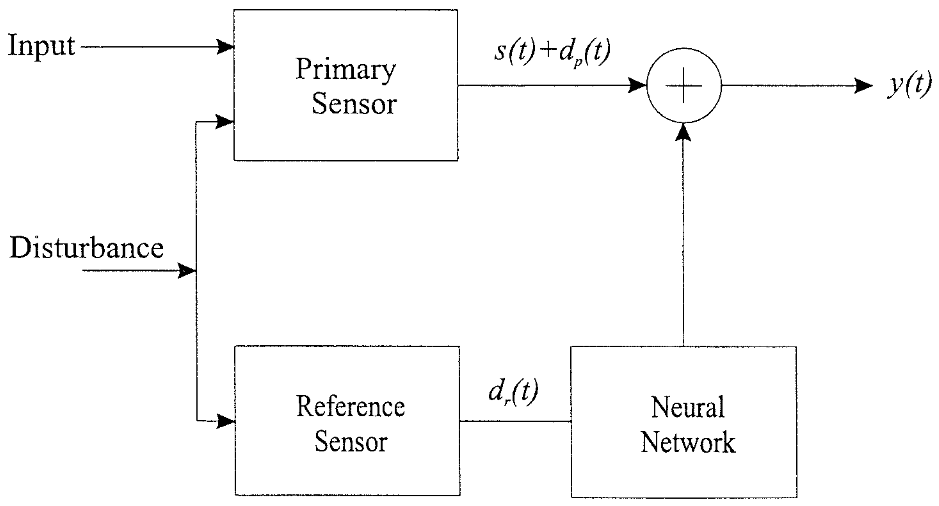
Figure 6.6: When the main and interference signals are correlated the adaptive filter produces signal plus interference.

the primary sensor, is the system identification phase. In this phase a MLP network has inputs connected to the output of the reference sensor and its target is the output of the primary sensor. This is trained to mimic the behaviour of the primary sensor stimulated by the interference signal. At the point that the main signal excites the primary sensor, phase 1 finishes and the phase 2 starts. During phase 2 the output of the MLP network is equal $d_p(t)$ i.e. that part of the primary sensor output which is due to the interference signal. The desired signal is obtained by subtracting the outputs of the primary sensor and the neural network.

Figure 6.8 shows the block diagram of the neural network that is used for the system identification. The number of neurones in the hidden layers of the neural network and the number of inputs or the number of delay units are



(a) Training phase



(b) Operating phase

Figure 6.7: Adaptive system identification method.

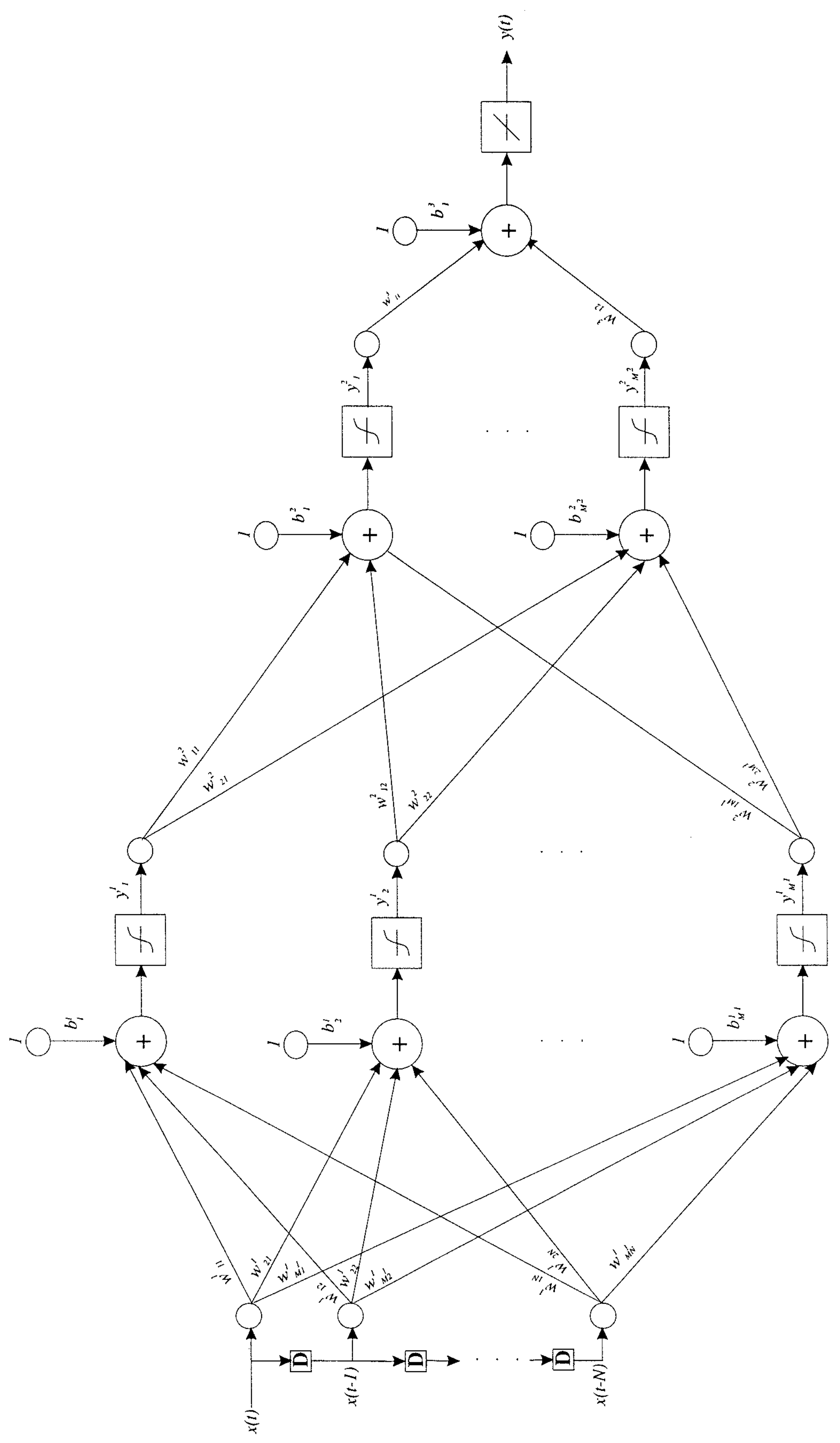


Figure 6.8: The dynamic neural network is used for interference cancellation.

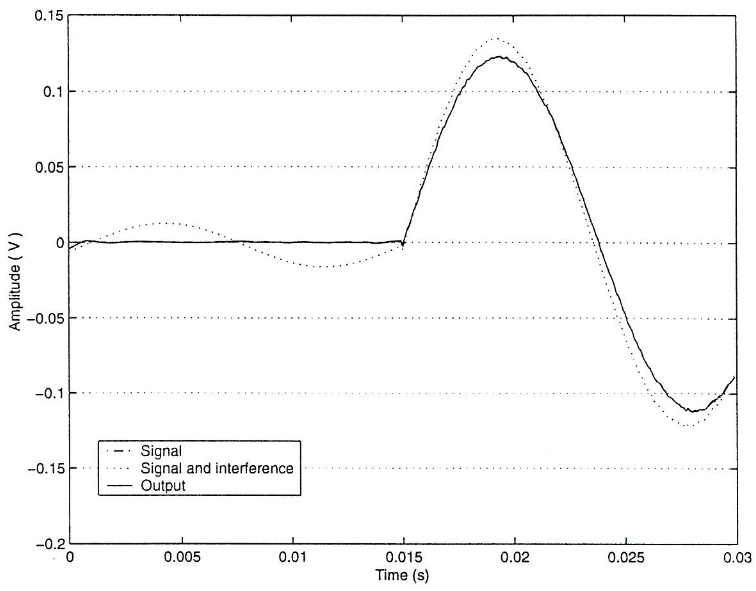


Figure 6.9: The result of using adaptive system identification method for cancelling the first interference signal shown in figure 6.2.

determined by trial and error. Figures 6.9 to 6.12 show the results of using this method for the interference signals shown in figure 6.2. The interference signals have been successfully removed. While the interference signals made 46% percent error on the features of the signal, after noise cancelling the error is less than 0.3%.

6.5 Drift

Drift in the amplification circuit can change the DC level of the sensor output which in turn causes error. The method described in previous section removes the effect of both interference signal and drift. This is illustrated in figure 6.13 that shows the output of the system when the interference signal is similar to one that is used in figure 6.12 except a DC level is added to it.

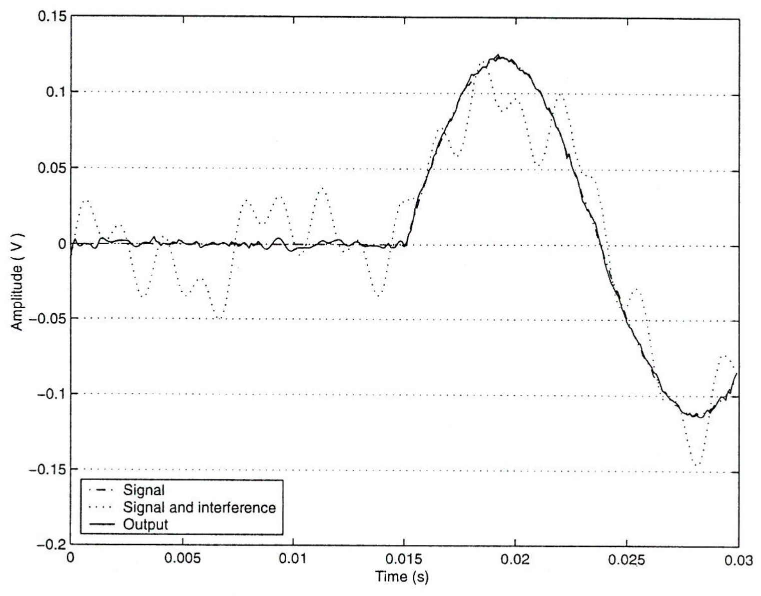


Figure 6.10: The result of using adaptive system identification method for cancelling the second interference signal shown in figure 6.2.

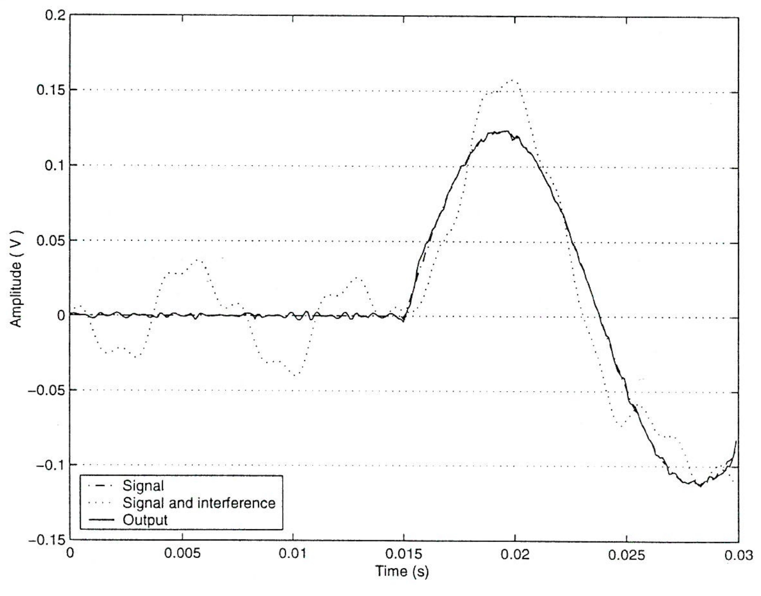


Figure 6.11: The result of using adaptive system identification method for cancelling the third interference signal shown in figure 6.2.

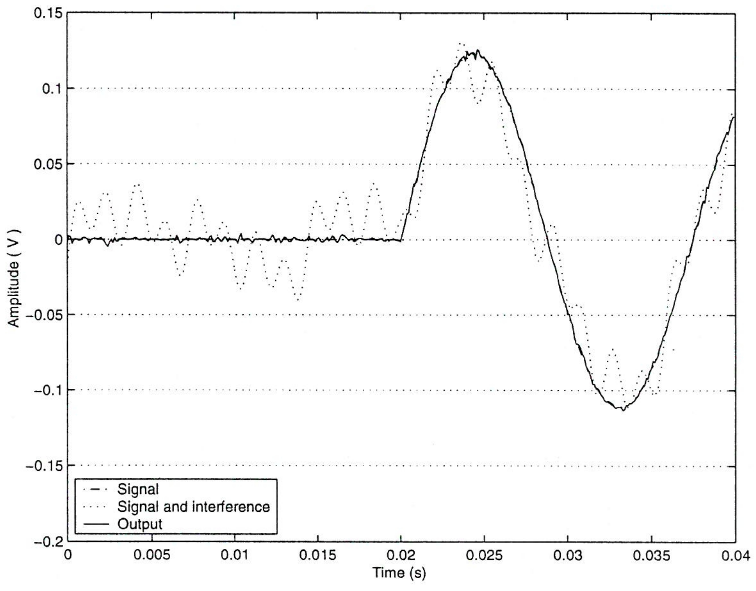


Figure 6.12: The result of using adaptive system identification method for cancelling the fourth interference signal shown in figure 6.2.

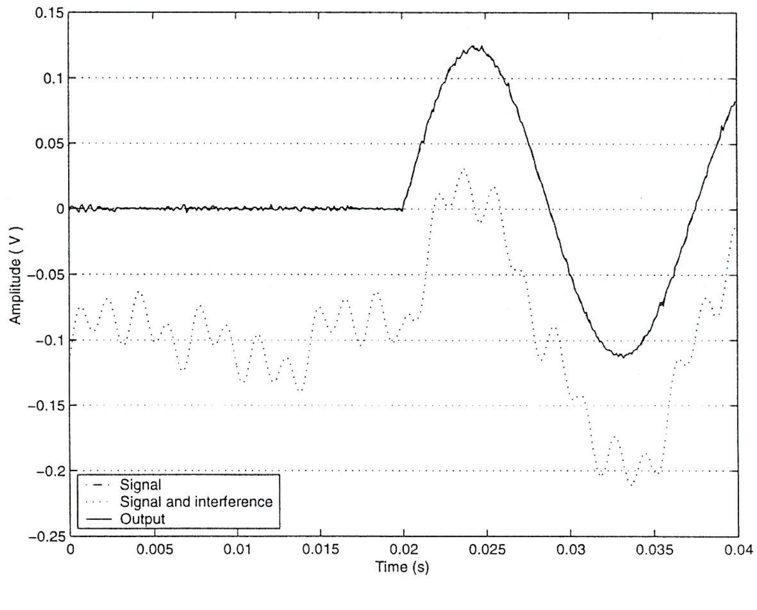


Figure 6.13: The adaptive system identification cancel the effect of drift as well as the interference signal.

It should be pointed out that, in the case of an impulse response, the DC level can be easily removed by a coupling capacitor but this is not the case for a step response. Thus the adaptive identification method can provide a good solution.

6.6 Training Time

The neural network training time affects the speed of the measurement. The actual time needed for training depends on the hardware and software that are used for implementation of the neural network and the training algorithm. In this work, the programmes are written in the MATLAB environment and run on a PC, hence the number of epochs is used as a measure of time. When a neural network is trained for the first time it takes about 70 epochs to meet the condition of mean square error less than or equal to 0.001. The number of epochs in the subsequence measurements is zero. If the amplitude or frequency of interference signal or the level of drift changes by 10% then the neural network is trained in less than 5 epochs.

6.7 Interference Frequency

While the parameters of the system are fixed, the frequency of the interference signal was changed over a wide range around the damping frequencies of the main signal. The output of the system for two frequency are shown in figures 6.14 and 6.15. The system copes with interference signals when their frequencies are greater or near the frequency of the main signal but it

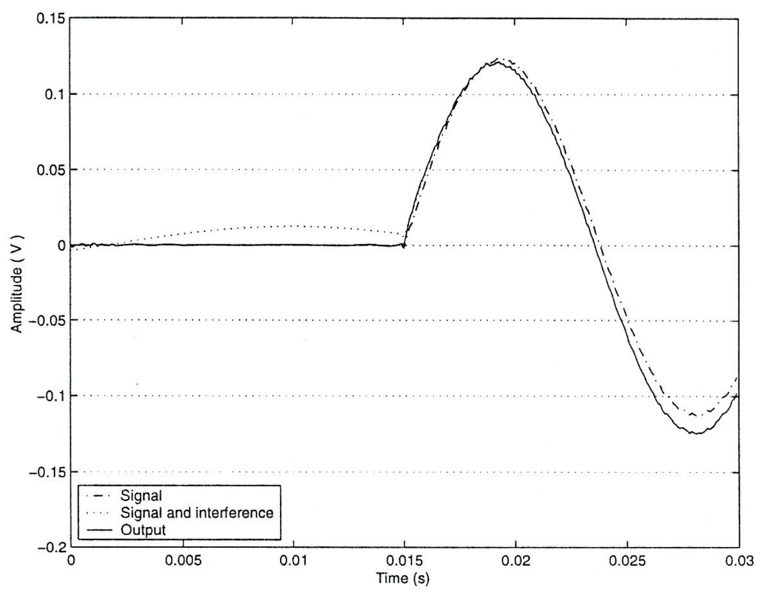


Figure 6.14: The frequency of interference signal is less than the frequency of the main signal.

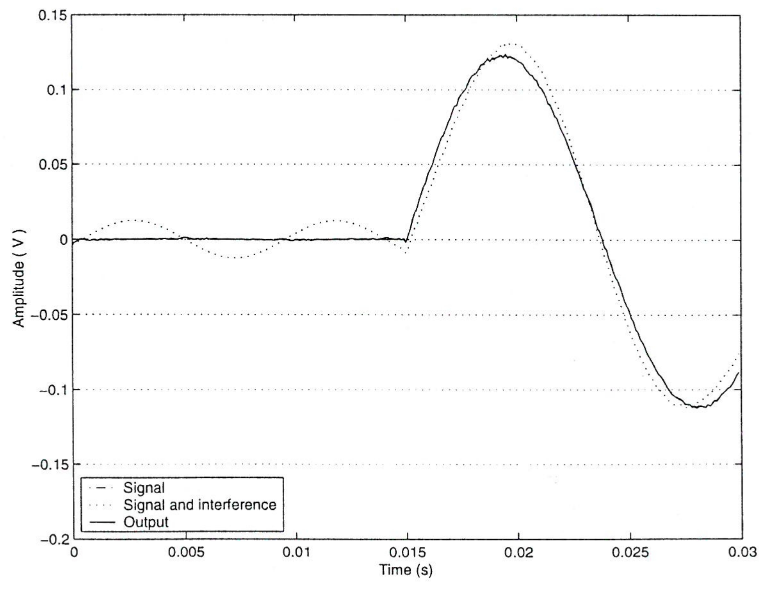


Figure 6.15: The frequency of interference signal is more than the frequency of the main signal.

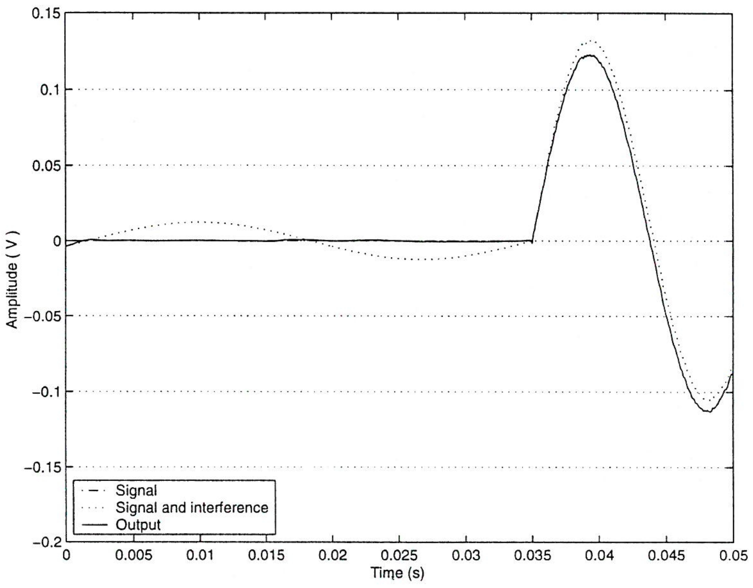


Figure 6.16: Neural network needs enough portion of interference signal for successful training.

deteriorats as the frequency of interference signal falls far from the main signal frequency. It is possible to change the parameters such as the number of delays in input, the number of layers of the network, the number of neurones in each layer and the number of samples before the start of the main signal, to find a solution which make the system able to remove the interference signal. Figure 6.16 shows interference signal in figure 6.14 has been removed by changing the parameters. As the parameters mentioned above have lower values, less time is needed for processing. Nevertheless, there are trade-offs between most of them and the processing speed. The number of samples before the start of main signal, however, can not be compensated by other factors. On the other hand, the simulation results show that at least the samples of one period of the interference signal are needed for system identification phase. Therefore a very low frequency interference signal can limit the frequency of the measurement

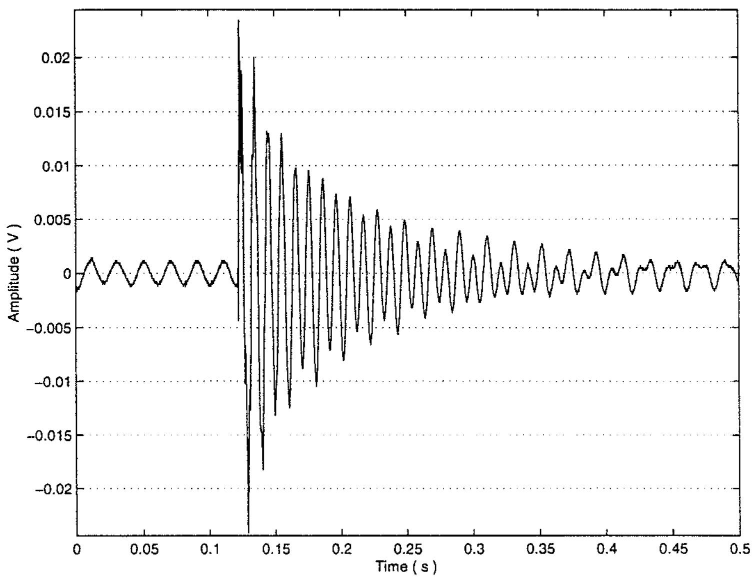


Figure 6.17: Output of the primary sensor.

although the result of the measurement is ready shortly after the stimulation of the sensor. This limitation, however can be overcome using a standard low pass filter because in this case the frequency band of the interference is far from the signal's spectrum.

6.8 Experimental Data

The experimental set described in chapter 5 was used for getting experimental data. The power line can produce a considerable interference signal in the output of the sensor as shown in figure 6.17. A second sensor was not available, therefore 50 sequence of interference signal were captured from the primary sensor. These sequences were added to 50 signals of a 4.26 g load that had been acquired in the absence of interference signal. Thus the primary signals in present of interference were obtained. The adaptive system identification

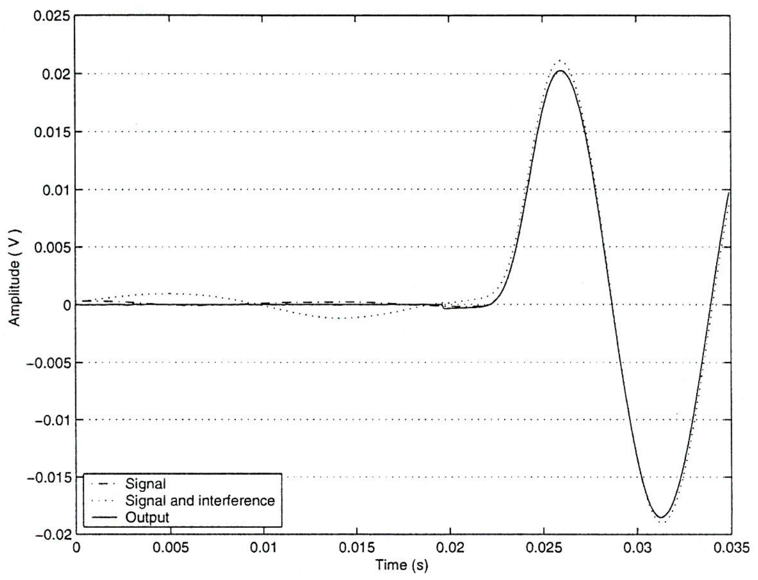


Figure 6.18: Cancellation of interference signal using adaptive system identification.

method was tested by these data and the results show that it properly removes the interference signal. Figure 6.18 shows the result for one signal. For the 50 signals, the histogram of error for the first and second features before and after using this method are shown in figures 6.19 and 6.20.

It shows that the percentage of error has been reduced by an order of 10 and for all of the signals is less than $\pm 1\%$. Therefore the adaptive system identification can be successfully employed in practice.

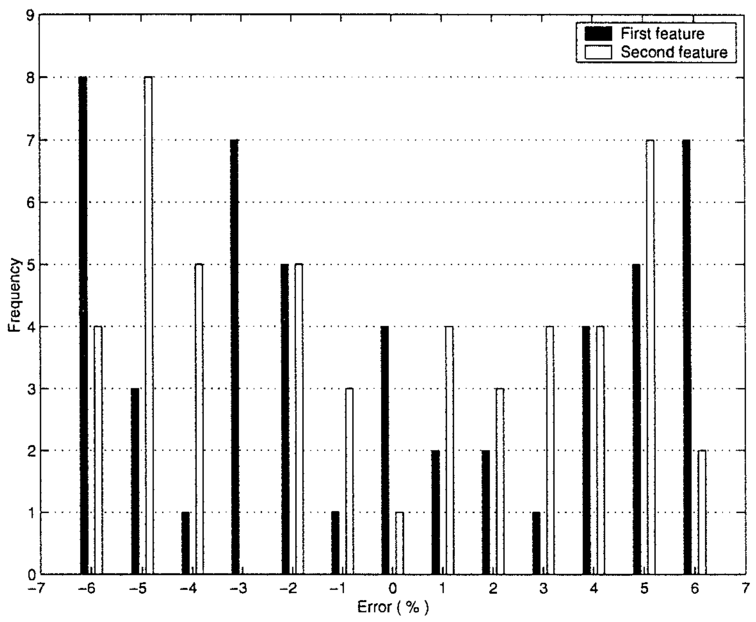


Figure 6.19: Histogram of the percentage of error for the first and second features before using adaptive system identification.

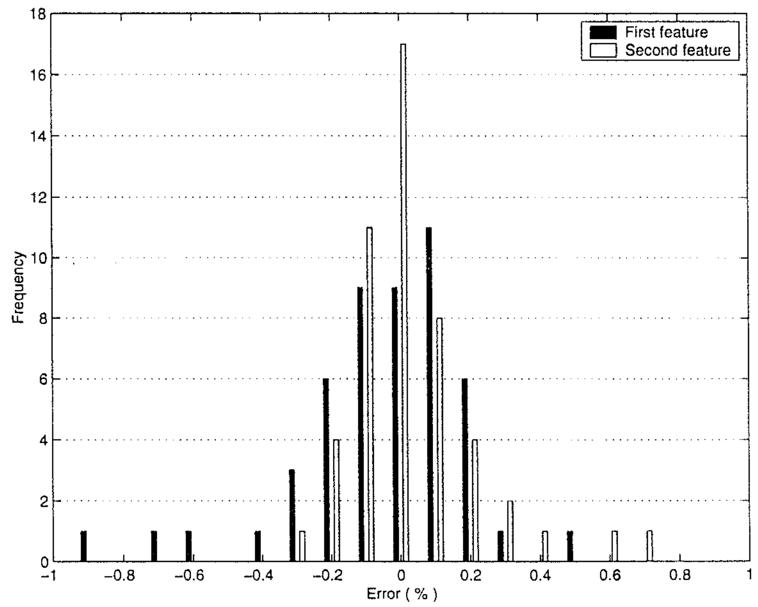


Figure 6.20: Histogram of the percentage of error for the first and second features after using adaptive system identification.

Chapter 7

Conclusions

7.1 Concluding Discussion

Dynamic measurement is an important need in the modern world. Its main aim is to predict the final value of the sensor output without waiting until the transient part of the response has decayed. The two significant advantages of dynamic measurement are:

- First, it speeds up the process of measurement. While this increases the speed of the whole process in general, it is vital for systems whose measurement time is limited.

- Second, dynamic measurement allows the use of inexpensive sensors having a highly oscillatory response. Low cost, accurate and reliable sensors have a fundamental significance for many of today's industrial and consumer products [72].

There are many different methods used for dynamic measurement. They can be classified in two categories: Classical methods and the new proposed method which is based on a neural network. Classical methods include integrated control and filtering, fuzzy control, adaptive digital filtering, choice of low pass filter, Kalman filtering and model parameter estimation. The use of a method depends on two factors: the type of sensor and the type of measurement process.

There are generally two types of sensor: controllable and non-controllable. Non-controllable sensors have one input and one output, whereas controllable sensors have two inputs and one output. The measurand is one input of a controllable sensor and the other input is a feedback signal from the output. Integrated control and filtering methods as well as fuzzy control are exclusively used for controllable sensors.

In general, there are two main types of measurement process: continuous and discrete. Discrete measurement deals with a set of measurands that have a fixed value and the sensor indicates these. For example, in some instances there is a finite set of loads that may be applied to the sensor. In this case, the value of the measurand does not change during the measurement process. With continuous measurement, however, the value of the measurand can vary with time. Measurement of the pressure of a water tank is a case in point. For continuous measurement, an adaptive digital filtering method, which is mainly a frequency compensation method, is necessary. With this technique, the frequency response of the sensor and the compensation block would be ideally equal to one, or more precisely they perform a delay unit, and the input signal appears at the output without any distortion. The Model parameter estimation methods and the neural network method are more suitable for discrete measurement because they predict the final value from a small fraction of the beginning of the signal. In fact for a non-noisy signal, the number of samples that is required is equal to the number of unknown parameters of the model. With the Model parameter estimation methods, when the number of parameters in the model increases, the computational complexity of the parameter procedure estimation increases dramatically. For example, with the non-linear regression method, the complexity increases as the third power of the number of unknown model parameters [10]. The neural network method does not have this problem. In addition, it eliminates the need for system identification, sensor modeling and look up tables for non-linearity correction. The neural network learns the behaviour of the sensor from the set of training patterns. A system that is designed based on neural network method has the ability to learn through contact with the environments and adjusts its own parameters automatically to adapt itself with the variation in the noise and sensor specifications. Moreover its design requires little or no prior knowledge of signal or noise characteristic so it can be used for a wide range of applications. In contrast, a system based on classical methods is limited to specific applications. Figure 7.1 shows the result of using neural network method on a highly oscillatory sensor. It predicted the final value of the sensor signal, quite a long time before the oscillation was damped.

The results of this study show that using a feature extractor can improve the efficiency of the neural network method. Firstly, the period of oscillation, which is rarely used in other methods, was adopted as a feature. This is

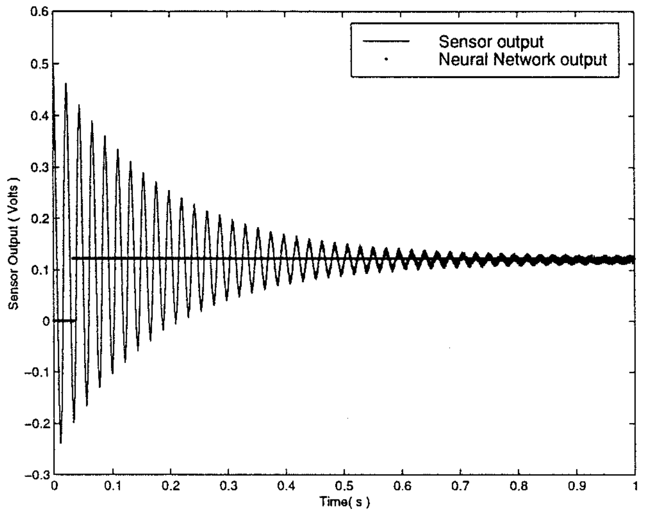


Figure 7.1: The output of the tri-beam load cell and neural network when the load is equal to 120 g.

important because it has a unique relation with the final value, and also the amplitude noise has less effect on it. Secondly, the number of neurones in the neural network reduces dramatically and hence both the training and weighing time decrease effectively. Thirdly, by using successive extreme points of the output signal, the initial conditions are not important. It was also shown that the preprocessing procedure has a great influence on the success or failure of the method. If the preprocessing unit could not cope with the noise then the response of the neural network will be unpredictable.

Neural network methods are especially useful in the situation where the sensor is stimulated for a short time period i.e. when the input is an impulse function. Classical methods such as adaptive digital filtering, however, are not applicable because they are based on averaging, and the average of the output is zero for each measurand. Although the final value is zero for different

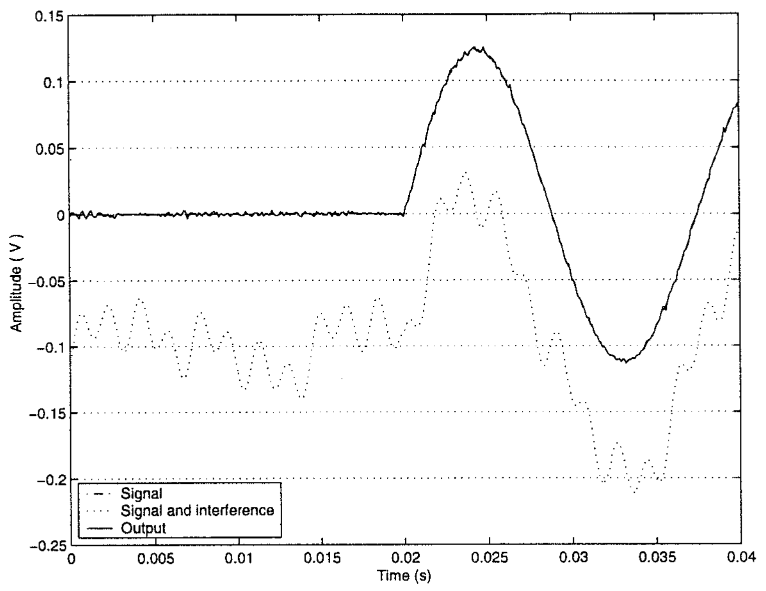


Figure 7.2: The adaptive system identification cancel the effect of drift as well as the interference signal.

measurands, initial section is different. Therefore it can be said that, in this situation, the neural network acts as a pattern recognizer that maps each different pattern to the right class.

The neural networks were also used in a novel way to cancel interference signals and drift effect. The conventional methods are only useful if the interference and sensor signals are uncorrelated. The proposed method in chapter 6, however, cancels the effect of interference even when it is correlated to the desired signal as shown in figure 7.2.

7.2 Further Work

In this study new methods were presented, based on neural networks, for the processing of a raw sensor signal and interference cancellation. The versatility and adaption properties of these methods fulfill the whole expectations for an intelligent sensor, if they are implemented in the housing of an elementary sensor. The following list summarizes the main area which are worthy for further investigation:

- Other input functions
- Continuous measurement
- Optimum training algorithm
- Number of neurones in hidden layer
- Hardware implementation
- Using evolutionary artificial neural network

7.2.1 Other Input Functions

In this thesis the step and impulse functions responses were fully investigated. In many applications, however, the input function may not necessarily be that of an ideal step or impulse function. Theoretically, the neural network method can cope with any type of input as long as it is repeatable. To find the optimum solution for other inputs, further simulations and experiments are needed. Experimental data should be acquired using automatic equipment for stimulating the sensor in order to satisfy the condition of the repeatability of the input. For example, the masses should be put on the tri-beam load cell by a mechanical mechanism.

7.2.2 Continuous Measurement

The method proposed in this study is suitable for discrete measurement. To use it in continuous measurement, it is suggested that the following modification be further investigated.

The Neural Network as an Inverse System

If the measurand does not contribute to the inertial parameters of the sensor i.e. the characteristic function of the sensor is independent from the measurand, then a dynamic neural network can be cascaded to the sensor. The neural network should learn to act as an inverse system of the sensor.

If it is possible to establish a relation between the parameters of the network and the output of the sensor then this method can be improved to use in the situations where the measurand changes the characteristic function of the sensor. In fact this method would be an analogue of the adaptive digital filtering technique with the exception that there is no need for sensor modeling and inverse system identification.

Two Subsystems

Figure 7.3 show the block diagram of another system that can be used for continuous measurement. The selector block chooses one of the two outputs of the subsystems as the system output. In the absence of any quick changes in the input of the sensor, the low pass filter output is the best indication of the sensor signal because the filter removes the effect of the high frequency noise from the steady state response of the sensor signal. When the input of

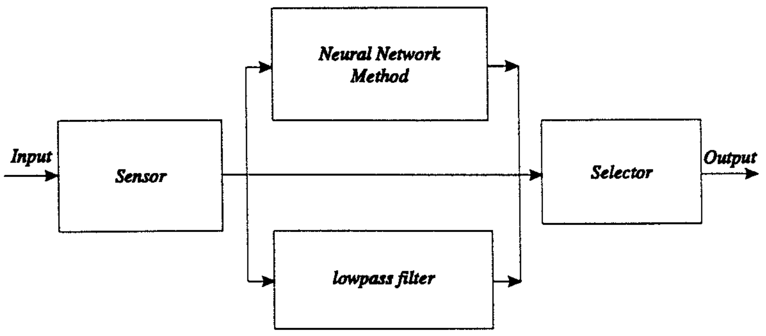


Figure 7.3: A system for continuous measurement.

the sensor changes rapidly, the neural network method block determines the steady state of the signal before the transient part decays. So its output should be switched to the output of the system. The selector block has an important role and should be designed elegantly.

7.2.3 Optimum Training Algorithm

There are a number of algorithms for neural network training. They are different in respect of the amount of memory they need and the speed of convergence. If the neural network is implemented by a microprocessor in the housing of the sensor, certain limitations regarding to the mentioned factors should be met. Hence a study of advantages and disadvantages of each training algorithm when they are used in the processing of the raw sensors signal are required.

7.2.4 Number of Neurones in Hidden Layer

In the new method described in this thesis, the neural networks were mainly used for approximation of a non-linear function. Theoretically any degree of accuracy is achievable if the number of neurones in the hidden layer is sufficient. There is no straightforward way, however, to determine the 'sufficient number' in each case. The number is obtained by trial and error. Choosing an incorrect number of neurones could result in an overestimate or underestimate of networks. Further work should be aimed at developing a technique which is capable of finding the optimum number of neurones without external help.

7.2.5 Hardware Implementation

The neural networks have an inherently parallel nature. A lot of work has been done for implementing neural networks using VLSI technology [73, 74, 75, 76]. Some commercial ICs are also available [77].. In this study the neural networks are implemented on a PC based on sequential processor. The full advantages of a neural network regarding the speed will be obtained if they are implemented using suitable hardware with parallel structure. This structure and the sensor can be combined later to implement in one chip.

7.2.6 Using Evolutionary Artificial Neural Network (EANNs)

Evolutionary artificial neural networks (EANNs) are a special class of neural networks that can adapt to an environment as well as changes in environment [78]. In addition to learning, evolution is another fundamental form of adaption in EANN. One distinct feature of EANNs is their adaptability to a dynamic environment. Evolutionary algorithms are used to change the weights and architecture of the networks, learning rule and input features. The two forms of adaption, i.e. evolution and learning in EANNs makes them suitable architectures to use with intelligent sensors. In a broader sense, an EANN can be regarded as a general processing unit that can be put in the housing of any raw sensor, and it select input features, changes the architecture of the networks and learning rule appropriately without human intervention.

Appendix A

List Of Publications

S.M.T Alhoseyni Almodarresi Yasin,

"Application of artificial neural network in smart sensors",

Proceedings of the Electrical and Electronics Engineering seminar of Iranian students in Europe, London, UK, July 1998.

S.M.T Alhoseyni Almodarresi Yasin and N.M. White,

"The Application of artificial neural network to intelligent weighing systems",

IEE proceedings- Science, Measurement and Technology, Vol. 146, pp. 265-269, 1999.

S.M.T Alhoseyni Almodarresi Yasin,

"Using artificial neural network to determine the impulse response of sensors",

Proceedings of the Electrical and Electronics Engineering seminar of Iranian students in Europe, Manchester, UK, May 1999.

S.M.T Alhoseyni Almodarresi Yasin and N.M. White,
"Using artificial neural network to determine the impulse response of sensors",
Proceedings of the tenth conference on sensors and their applications, pp. 97-101, Cardiff, Wales, 5-8 September 1999.

S.M.T Alhoseyni Almodarresi Yasin and N.M. White,
"Dynamic measurement using artificial neural networks",
Proceedings of the international conference on sensors and transducers, pp. 1-8, Birmingham, UK, 16-17 February 2000.

Presentation

S.M.T Alhoseyni Almodarresi Yasin,
"Using neural network in smart sensors",
The ECS learned society , Southampton 25 November 1998.

Bibliography

[1] J E Brignell and N M White. *Intelligent Sensor Systems.* Institiute of Physics Publishing Ltd, 1994.

[2] J E Brignell. The future of intelligent sensors: a problem of technology or ethics? *Sensors and Actuators A*, A 56:11–15, 1996.

[3] J E Brignell. Software techniques for sensor compensation. *Sensors and Actuators A*, 25-27:29–35, 1991.

[4] A H Taner and N M White. Virtual instrumentation: a solution to the problem of design complexity in intelligent instruments. *Measurement + Control*, 29:165–171, 1996.

[5] A H Taner. *Self-test and auto-calibration in intelligent sensor design aids for re-configurable ASICs.* PhD thesis, Department of Electronics and Computer Science,University of Southampton, 1996.

[6] E T Powner and F Yalcinkaya. From basic sensors to intelligent sensors: definitions and examples. *Sensor review*, 15:19–22, 1995.

[7] E T Powner and F Yalcinkaya. Intelligent sensors: structure and system. *Sensor review*, 15:31–35, 1995.

[8] N White. Intelligent sensors. *Sensor review*, 17:97–98, 1997.

[9] W J Shi N M White and J E Brignell. Adaptive filters in load cell response correction. *Sensors and Actuators A*, A 37-38:280–285, 1993.

[10] M Danaci and D H Horrocks. A non-linear regression technique for improved dynamic weighing. *Proceeding of the 12th European conference on circuit theory and design*, 1:507–510, 1995.

[11] W Q Shu. Dynamic weighing under nonzero initial conditions. *IEEE transactions on instrumentation and measurement*, 42:806–811, 1993.

[12] H B Bahar and D H Horrocks. Dynamic weight estimation using an artificial neural network. *Artificial Intelligence in Engineering 12*, 12:135–139, 1997.

[13] S Haykin and B Kosoko. Scanning the issue: Special issue on intelligent signal processing. *Proceeding of the IEEE*, 78(9):1415–1442, 1990.

[14] L T Fine. *Feedforward neural network methodology.* Springer, 1999.

[15] C J Harris. *Advances in intelligent control.* Taylor and Francis Ltd., 1994.

[16] M Brown and C J Harris. *Neorofuzzy addaptive modelling and control.* Prentice Hall International (UK) Limited, 1994.

[17] C G Moore C J Harris and M Brown. *Intelligent control: aspects of fuzzy logic and neural nets.* World Scientific, 1993.

[18] A Cichocki and R Unbehauen. *Neural networks for Optimization and signal processing.* John Wiley, 1993.

[19] F L Luo and R Unbehauen. *Applied neural networks for signal processing.* Cambridge university press, 1998.

[20] W T Miller R S Sutton and P J Werbos. *Neural network modeling and connectionism.* The MIT Press, 1990.

[21] P Huggins. A high speed loadcell. *Measurement and control*, 15:211–213, 1982.

[22] A V Oppenheim A S Willsky and I T Young. *Signals and systems.* Prentice-Hall, 1993.

[23] A V Oppenheim and R W Schafer. *Digital Signal Processing.* Prentice-Hall, 1975.

[24] D M Burley. *Studies in optimisation.* International textbook company Ltd., 1974.

[25] R O Duda and P E Hart. *Pattern Cassification and Scene Analysis.* John Wiley and Sons, Inc., 1973.

[26] W J Shi. *Dynamic frequency compensation for transducers.* PhD thesis, Department of Electronics and Computer Science,University of Southampton, May 1992.

[27] W Spendley G R Hext and F R Himsworth. Sequential application of simplex designs in optimaisation and evolutionary operation. *Technometrics*, pages 441–461, 1962.

[28] W J Shi and J E Brignell. On-line optimization in sensor frequency response compensation. *Sensors and Actuators A*, A 25-27:37–41, 1991.

[29] M Tariq W Balachandran M Halimic F Cecelja and Y M Enab. The simulation and application of adaptive filtering on an industrial checkweigher. *The 5th conference on signal processing application and technology*, pages 1066–1071, Octobber 1994.

[30] M Halimic and W Balachandran. Kalman filter for dynamic weighing system. *IEEE International symposium on industrial electronics*, pages 787–791, July 1995.

[31] L L Scharf. *Statistical signal processing.* Addison-Wesley publishing company,Inc., 1991.

[32] P P Kanjilal. *Adaptive prediction and predictive control.* Peter Peregrinus Ltd., on behalf of the IEE, 1995.

[33] D S G Pollock. *A handbook of time-series analysis, signal processing and dynamics.* Academic press, 1999.

[34] W Balachandran M Halimic M Hodzic M Tariq Y M Enab and F. Cecelja. Optimal digital control for dynamic weighing system. *Instrument/Measurement technology conference*, pages 293–298, April 1995.

[35] W Balachandran Y M Enab M Halimic and M Tariq. Intelligent robot dynamic weighing system. *Procs. SPIE*, 2353:398–409, 1994.

[36] B Widrow and S D Stearns. *Adaptive signal processing.* Prentice-Hall, Inc., 1985.

[37] S Haykin. *Neural Network.* Macmillan College Publishing Company, Inc, 1994.

[38] M H Hassoun. *Fundamentals of artificial neural networks.* Cambridge,Mass. M.I.T. Press, 1995.

[39] B Widrow. 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *proceeding of the IEEE*, 78(9):1415–1442, 1990.

[40] D K Frederick and A B Carlson. *Linear systems in communication and control.* John wiley and sons, Inc., 1971.

[41] O Masubuchi. Considration of response characteristics of automatic check-weigher. *Anritsu technical Bulletin*, 41:49–53, 1981.

[42] N M White. *A study of the piezoresistive effect in thick-film resistors and its application to load trasduction.* PhD thesis, Department of Electronics and Computer Science,University of Southampton, August 1988.

[43] S M T Alhoseyni Almodarresi Yasin and N M White. The application of artificial neural network to intelligent weighing systems. *IEE proceedings-Science, Measurement and Technology*, 146:265–269, 1999.

[44] G B Clayton. *Operational amplifiers.* Butterworth, 1986.

[45] P Brokaw. *An IC amplifier user's guide to decoupling, grounding, and making things go right for a change.* Analog Devices, Inc. AN-202.

[46] E Nash. *A practical review of common mode and instrumentation amplifiers.* Analog Devices, Inc.

[47] E Nash. *Errors and error budget analysis in instrumentation amplifier applications.* Analog Devices, Inc. AN-539.

[48] *AD524 Precision instrumentation amplifier datasheet.* Analog Devices, Inc., 1997.

[49] *LabVIEW user manual.* National Instruments Corporation, 1996.

[50] *LabVIEW data acquisiton basics manual.* National Instruments Corporation, 1997.

[51] *LabVIEW function and VI reference manual.* National Instruments Corporation, 1997.

[52] *G programming reference manual.* National Instruments Corporation, 1998.

[53] *NI-DAQ user manual for PC compatible.* National Instruments Corporation, 1995.

[54] *AT-MIO E series user manual.* National Instruments Corporation, 1995.

[55] *Using MATLAB (version 5).* The Math works, Inc., 1998.

[56] H Demuth and M Beale. *Neural Network Toolbox User's Guide.* The Math works, Inc., 1998.

[57] *Using MATLAB Graphics(version 5).* The Math works, Inc., 1998.

[58] *MATLAB apllication program interface guide (version 5).* The Math works, Inc., 1998.

[59] A S Morris. *Princioles of measurement and instrumentation.* Prentice Hall, 1988.

[60] W Bolton. *Engineering instrumentation and control.* Butterworths, 1980.

[61] K C Crandall R W Seabloom. *Engineering fundamentals in measurement, probability, statistics and dimentions.* McGrraw-Hill, Inc., 1970.

[62] *LabWindows/CVI user manual.* National Instruments Corporation, 1994.

[63] *LabWindows/CVI standard library reference manual.* National Instruments Corporation, 1994.

[64] *LabWindows/CVI advanced analysis library reference manual.* National Instruments Corporation, 1994.

[65] S P Harbison and G L Steele Jr. *C, a reference manual.* Prentice Hall, 1991.

[66] S.M.T Alhoseyni Almodarresi Yasin and N.M. White. Using artificial neural network to determine the impulse response of sensors. *Proceedings of the tenth conference on sensors and their applications*, pages 97–101, 1999.

[67] C A Desoer and E S Kuh. *Basic circuit theory.* MacGraw-Hill, 1969.

[68] J L Meriam and L G Kraige. *Engineering Mechanics.* John Wiley and Sons, Inc, 1998.

[69] S.M.T Alhoseyni Almodarresi Yasin. *Design and implementation of the Ion Mobilty Spectrum (IMS) cell and classification of its signals.* MSc Thesis, Isfahan University of Technology, Iran, 1991.

[70] *OP999 Photodiode datasheet.* Optek Technology, Inc., 1993.

[71] *OPE5594S Ifrared emmiting diode datasheet.* Optek Technology, Inc., 1993.

[72] J W Gardner. *Research and development of sensors and intelligent instrumentation (electronic noses) for complex odour and gas mixture analysis.* Draft submission for the degree of Doctor of Science, University of Warwick, December 1995.

[73] T Shima T Kimura Y Kamatani T Itakura Y Fujita and T Iida. Neuro chips with on-chip back propagation and/or hebbian learning. *IEEE Journal of solid-state circuits*, 27:1868–1876, 1992.

[74] A J Montalvo R S Gyurcsik and J J Paulos. An analog VLSI neural network with on-chip perturbation learning. *IEEE Journal of solid-state circuits*, 32:535–543, 1997.

[75] I Bayraktaroglu A S Ogrenci G Dundar S Balkir and E Alpaydin. Annsys: an analog neural network synthesis system. *Neural Network*, 12:325–338, 1999.

[76] P Treleaven M Pacheo M Vellasco. VLSI architectures for neural networks. *IEEE Micro*, pages 8–27, 1989.

[77] *ZISC036 Neurones data book.* IBM France, 1998.

[78] X Yao. Evolving artificial neural networks. *Proceeding of the IEEE*, 87(9):1423–1447, 1999.

S. M. T. Almodarresi Yasin and N. M. White, "Application of artificial neural networks to intelligent weighing systems," in *IEE Proceedings - Science, Measurement and Technology*, vol. 146, no. 6, pp. 265-269, Nov 1999.
doi: http://dx.doi.org/10.1049/ip-smt:19990679