Neural Network Based Decision Support:

Modelling and Simulation of Water Distribution Networks

A thesis submitted in partial fulfilment of the requirements of The Nottingham Trent University for the degree of Doctor of Philosophy.

Bogdan Gabrys

October 1997

Abstract

This work concerns the applicability of neural networks for the implementation of decision support (DS) systems in operational control of industrial processes. Decision support has two distinct but closely interrelated aspects: mathematical modelling of physical plants and processes, and the modelling of a decision making process. The first one forms the basis for detailed optimization of operations and the second one attempts to mimic an abstract mental reasoning about plant's operation by human operators. This research attempts to integrate both aspects of decision support within a single computational framework of neural networks. The prototype DS system is validated using case-studies taken from the water industry. The optimal control of water systems is a challenging problem because the models are non-linear and large-scale and measurements are noisy and frequently incomplete. The results of this research are general and are directly applicable to other systems, for example, gas and power utilities or road traffic systems.

In the first part of the project the neural network approach to the state estimation problem and confidence limit analysis for water systems is proposed. Since state estimation process is a computationally demanding task new approaches to solving it are constantly being looked for. The neural networks are one of the possible options. The resulting algorithms are the mixture of the well known and tested ways of solving systems of nonlinear equations (the Newton-Raphson method), the optimization criterions (the LS, LAV and their variations) and a relatively new artificial neural network (ANN) technique of finding the solution to the overdetermined systems of linear equations. The problems of bad data rejection, ill-conditioning, arriving at the solution within a predefined period of time are addressed and suitable ANN techniques are proposed and evaluated.

No state estimator can give accurate results from inaccurate data. A way of utilising the neural networks, that have been used to produce the state estimates, for quantifying the measurement uncertainty impact on the state estimates is shown. Two methods of obtaining the confidence limits in form of upper and lower bounds for each state estimate are investigated. The first method presents the usage of neural networks to find the sensitivity matrix which enables calculation of these bounds. In broader terms the way of finding inverse and pseudoinverse matrices, using ANN, is shown. The second method utilizes the superposition principle where each disturbance is analysed separately and the partial results are gradually combined to produce overall confidence limits.

Finally an integrated neural based system for state estimation and confidence limit analysis has been developed and tested for realistic water distribution network.

The second part of this project concerns the development of flexible fuzzy neural recognition system and its application to water systems' state interpretation task.

First a new general fuzzy neural network for clustering and classification is proposed. It can process both deterministic and fuzzy input patterns, combines the supervised and unsupervised learning techniques within a single training algorithm, grows to meet the demands of the problem and learns on-line.

The problems of fault diagnosis and water state interpretation are then addressed. A completely new approach to bad data detection and identification in water systems based on pattern examinations using neural recognition system is demonstrated. The use of state estimates and residuals with corresponding confidence limits is examined. The extensive performance studies for 24 hour of water network operations with particular emphasis on detection and correct location of leakages are carried out.

The work described in this Thesis is the Author's own, unless otherwise stated, and it is, as far as he is aware, original.

Acknowledgements

The research work that is presented in this thesis has been carried out under the supervision of Professor Andrzej Bargiela. I am most grateful for his continual encouragement and many fruitful discussions.

I would also like to thank my friends and colleagues in the Department of Computing.

Finally, but not least, I would like to express a special thanks to my wife Iwona for her understanding and moral support throughout the course of this work.

I dedicate this work to my parents whose continuous support and encouragement over the years have been invaluable.

Table of contents

List of Figures

List of Tables

Glossary of Terms

Activation function

A function that maps the neuron's infinite domain to a prespecified range.

Adaptive Fuzzy Leader Clustering (AFLC)

A hybrid neural-fuzzy system which can be used to learn cluster structure in a self-organizing, stable manner. The algorithm utilizes a single point to represent the center of each cluster and uses Euclidean distance from a cluster prototype as a measure of similarity.

Adaptive Resonance Theory (ART)

A class of neural network architectures that carry out stable self-organization of recognition categories in response to arbitrary sequences of input patterns. The concept of adaptive resonance was introduced by Grossberg (Grossberg, 1976b) and was first cast into a neural network formalism by Carpenter and Grossberg (Carpenter & Grossberg, 1987).

Artificial Neural Network (ANN)

Mathematical models of information processing that have certain performance characteristics in common with biological neural networks.

Competitive learning

Unsupervised learning used in a class of neural networks where a group of neurons compete for the right to be active. In the most extreme example the activation of the neuron with the largest output is set to 1 and the activations of all other nodes are set to 0. This mechanism is often called "winner-takes-all".

Confidence Limit Analysis (CLA)

The process of quantification of the inaccuracy of state estimates caused by the input data uncertainty.

Fuzzy Min-Max Neural Networks

Clustering and classification neural networks that are built using hyperbox fuzzy sets. A hyperbox defining a region of the n-dimensional pattern space is used as a cluster representation. A fuzzy Hamming distance from the edges of an n-dimensional hyperbox is used to calculate the degree to which an input pattern belongs to this hyperbox.

Hazen-Williams Formulae

The most commonly used empirical formulae describing the frictional head loss in a pipe.

Hyperbox

A representation of a cluster in the min-max clustering and classification neural networks (Simpson, 1992; Simpson, 1993). A hyperbox is completely defined by its min point and its max point. The combination of the min-max points and the hyperbox membership function defines a fuzzy set (cluster).

Hyperbox membership function

A function describing the degree to which an input pattern fits within the hyperbox.

LAV criterion

An optimization criterion where the sum of absolute values of errors is minimised.

Learning (training) algorithms

Procedures for setting, modifying the values of the neural network weights.

LS criterion

An optimization criterion where the sum of the squared values of errors is minimised.

Minimax (Chebyshev) criterion

An optimization criterion where the largest error is minimised.

Monte Carlo simulation

A method used to analyse the influence of parameters variability on the solution of systems. Very often used for sensitivity analysis of the nonlinear systems where the analytical relationship between the variability of parameters and system's solution cannot be found and repeated simulation for a large number of parameters with random variations is used instead.

Neural Network Architecture

Arrangement of artificial neurons organized into layers and linked with weighted interconnections.

Outlier

A measurement containing large error.

Overdetermined set of equations

The set of equations where a number of equations is greater than a number of unknowns.

Pseudomeasurements

Values which represent consumption predictions or information about water distribution network topology.

Reynolds number (*Re*)

The dimensionless parameter indicating the type of flow (i.e. laminar or turbulent flow).

Sensitivity matrix

A matrix in which the (i,j)-th element relates the sensitivity of the i-th estimated value to the variations in the j-th element of the measurement vector.

State estimation

A procedure that provides a means of reconciling the discrepancies between the mathematical model of the system and the input data.

State vector

The minimum set of variables that uniquely describe a system and from which other variables may be calculated.

Supervised learning

A learning algorithm in which each input pattern is presented with an associated target output and weights are gradually updated so that the error between the desired and the network's output is reduced.

Unsupervised learning

A means of modifying the weights of a neural network without specifying the desired output for any input patterns.

Chapter 1

Introduction

1.1. Project description

The overall aim of this research project is to investigate the applicability of neural networks for the implementation of decision support (DS) systems in operational control of industrial processes.

Decision support has two distinct but closely interrelated aspects: mathematical modelling of physical plants and processes, and the modelling of a decision making process. The first one forms the basis for detailed optimization of operations and the second one attempts to mimic an abstract mental reasoning about plant's operation by human operators. This research attempts to integrate both aspects of decision support within a single computational framework of neural networks. The research will build on our recent research results concerning analog neural networks, and extends other researchers' work which confirmed the feasibility of mapping both numerical and fuzzy optimization tasks onto appropriate neural networks.

The novelty of the project is in defining more efficient neural network models for optimization, as well as enhancing existing models, and then combining them into a coherent environment for robust optimization of industrial processes. Neural network technology has matured enough to indicate its potential use in real-time industrial situations.

The prototype DS system will be validated using case-studies taken from the water industry. The optimal control of water systems is a challenging problem because the models are non-linear and large-scale and measurements are noisy and frequently incomplete. The results of this research are general and are directly applicable to other systems, for example, gas, electric power or road traffic systems.

All parts of the project and their interconnections can be presented in a form of the block diagram presented at Figure 1-1.

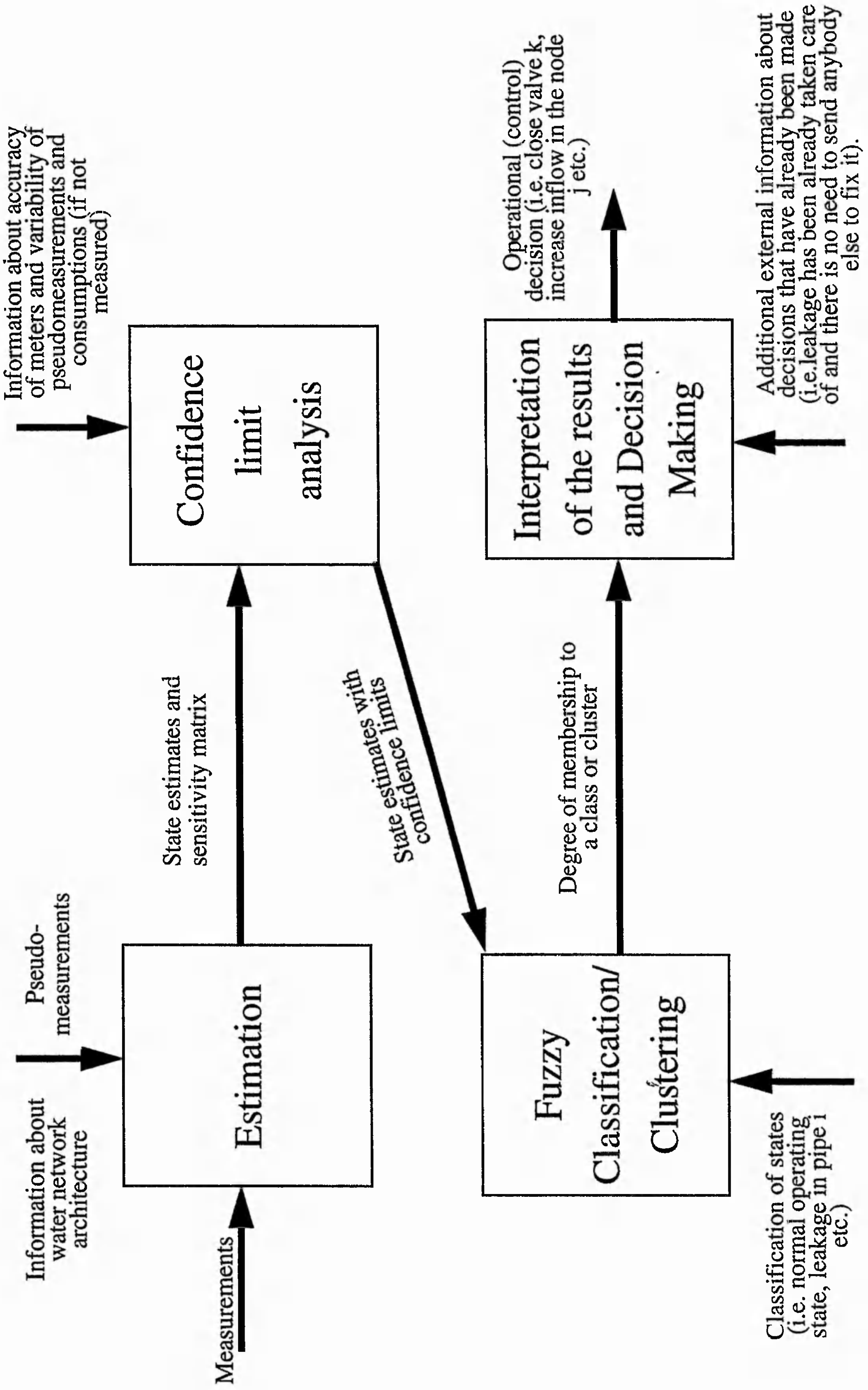

Information about water network architecture
Pseudo-measurements
Measurements
Estimation
State estimates and sensitivity matrix
Information about accuracy of meters and variability of pseudomeasurements and consumptions (if not measured)
Confidence limit analysis
State estimates with confidence limits
Fuzzy Classification/ Clustering
Classification of states (i.e. normal operating state, leakage in pipe i etc.)
Degree of membership to a class or cluster
Interpretation of the results and Decision Making
Additional external information about decisions that have already been made (i.e.leakage has been already taken care of and there is no need to send anybody else to fix it).
Operational (control) decision (i.e. close valve k, increase inflow in the node j etc.)

Figure 1-1: Block diagram illustrating different phases of the project and their interconnections.

1.2. Water distribution systems

Modern water distribution systems, as the other two major utility systems - electricity and gas, are characterised by their complexity and large scale.

Additionally, it is quite difficult to identify a typical water distribution system. Each one has some unique characteristics due to the water source, service area topography, history of the system, etc. In general, all that can be said is that there are water sources and water users and that they are connected by pipes. The pipes can be made of many different materials (cast iron, steel, concrete etc.) and may be connected in many configurations.

There may be a single source such as a central pump station, or water may be supplied by a large number of wells. While pumps are a common component of many systems, there are networks which do not have any pumping and the water is supplied from some sources at high elevation.

Most systems contain some storage capacity in the form of tanks which are connected directly to the system, from which water must be pumped or which hold water under pressure.

Valves are required to shut off lines, suppress surges, release air, drain pipes, or control pressure.

Booster pumping may be required to provide adequate pressure in certain portions of a system when there is significant variation in elevation or use rate. On the other hand, pressure reducing valves serving just the opposite purpose may be needed.

Thus, water distribution systems consisting of large number of pipes, pumps, valves etc. are indeed complex hydraulic systems.

As indicated by the block diagram of the project (Figure 1-1) this project is focused on operational control of water distribution systems rather than design or management of such systems.

Efficient control of a complex water distribution system requires accurate information about its current operating state. At present in the water industry, modern telemetry hardware systems are being installed to meet these needs. Unfortunately, due to financial constraints, it is not practical to measure all variables of interest. By variables of interest we mean here heads at all network nodes and inflows at fixed -head nodes which are the components of the state vector of the system since given this information and the static parameters of the network all other variables, such as pipe flows or consumptions, may be calculated immediately.

A practical way of finding the system state is to solve a set of mass balance equations which combine the network topology data, the measured or estimated consumptions and the inflows into the system. This method, known as load-flow method, is often used in water network simulation studies. However, from on-line control point of view the load-flow method has two major drawbacks. Firstly, if one measurement is incorrect or lost, the load-flow approach gives incorrect results or no results at all. And secondly, the method uses only the system inflows and consumptions which, as in the case of predicted values, may carry considerable errors, while other more accurate and readily available measurements are not used.

A method that overcomes these drawbacks is known as a state estimation procedure and over the last two decades has been the key point for the implementation of monitoring and control of large scale public utility systems. Its strength lays in processing all available measurements and formulating the problem in terms of redundant equations. This redundancy is essential for the successful performance of state estimation procedure since it enables the erroneous information to be filtered out. In water systems the degree of redundancy is achieved by combining the measurement information with the pseudomeasurements[1]. Thus, by increasing the number of measurements it is possible to improve both the reliability and accuracy of state estimation.

The simulation of any complex engineering system will always include a degree of uncertainty. No meters can be fully accurate, no mathematical model can fully reflect the intricacies of a real system's behaviour and no engineer's knowledge is complete. Water distribution systems are no exception to this rule.

This measurement uncertainty has clearly an impact on the accuracy to which state estimates can be calculated. It is, therefore, very important that the level of uncertainty present in state estimates can be quantified in some way if these estimates are to be used as the basis for making control decisions.

1. Pseudomeasurements represent consumption predictions or information about network topology.

The quantification of the inaccuracy of calculated state estimates caused by the input data uncertainty is known in water distribution systems as confidence limit analysis (CLA). In the effect of applying this procedure the lower and upper limits for each state estimate value are produced and the state vector is rather presented with corresponding confidence limits than in deterministic form.

The material presented so far, concerning the state estimation and CLA, is well researched and documented. But although the knowledge of the current operating state, and how accurate the estimates are, is absolutely essential it is only the first step on the way from measurement readings to operational decision making. The second step, that seems to be much more difficult, is the task of interpreting or classifying the current state of the network (i.e. normal operating state, leakage in area *i*, etc.) and subsequently, on the basis of this classification, making an operational decision (i.e. close valve *k*, do nothing, etc.).

There have been attempts to develop operational decision making algorithms based on the probabilistic mathematical models[1]. However, there has also been emphasized the need for human operator to close the control loop, especially in "non-standard", difficult to formalise situations where the operator's ability to make a choice is required.

Therefore, the state interpretation task, as well as making operational decisions, is often carried out by experienced, human operator. Experience, in this case, is the key word because the decision making process, although rational, is largely not formalised.

This points to the need for years of learning from other operators/engineers, observing the behaviour of the system and recollection of similar events from the past.

1.3. Why neural networks?

As modern computers become ever more powerful, scientists continue to be challenged to use machines effectively for tasks that are relatively simple for humans. Based on examples, together with some feedback from a "teacher", we learn easily to recognize the letter A or distinguish a cat from a bird. More experience allows us to refine our responses and improve our performance. Although eventually, we may be able to describe rules by

1. Probabilistic mathematical model - a model that takes into consideration and quantifies the uncertainty existing in the system. Such a model is seen as reflecting more naturally the operator's reasoning about the system.

which we can make such decisions, these do not necessarily reflect the actual process we use. Even without a teacher, we can group similar patterns together. Yet another common human activity is trying to achieve a goal that involves maximizing a resource while satisfying certain constraints. Each of these types of problems illustrates tasks for which computer solutions may be sought.

Traditional, sequential, logic-based digital computing excels in many areas, but has been less successful for other types of problems. The development of artificial neural networks began approximately 50 years ago, motivated by a desire to try both to understand the brain and emulate some of its strengths. Early successes were overshadowed by rapid progress in digital computing. Also, claims made for capabilities of early models of neural networks proved to be exaggerated, casting doubts on the entire field.

Recent renewed interest in neural networks can be attributed to several factors. Training techniques have been developed for the more sophisticated network architectures that are able to overcome the shortcomings of the early, simple neural nets. High-speed digital computers make the simulation of neural processes more feasible. Technology is now available to produce specialised hardware for neural networks. However, at the same time that progress in traditional computing has made the study of neural networks easier, limitations encountered in the inherently sequential nature of traditional computing have motivated some new directions for neural network research. Fresh approaches to parallel computing may benefit from the study of biological neural systems, which are highly parallel. The level of success achieved by traditional computing approaches to many types of problems leaves room for a consideration of alternatives.

Neural nets are of interest for researchers in many areas for different reasons. Electrical engineers find numerous applications in signal processing and control theory. Computer scientists find that neural nets show promise for difficult problem areas such as an artificial intelligence and pattern recognition. For applied mathematicians, neural nets are a powerful tool for modelling problems for which the explicit form of the relationships among certain variables is not known. For us they are especially interesting because of two potential application areas: optimization problems and pattern classification.

1.4. Organisation of the thesis

Chapter 2 is wholly devoted to mathematical modelling in water distribution systems. First the basic laws governing the hydraulics of water systems and mathematical models of simple elements that are parts of these systems are described. Next the methods of combining these laws and models of elements to produce concise model of the water network are presented. Since the uncertainty is an inherent part of water systems the ways of introducing it into the network model are discussed. Having developed the system of equations describing the pipe network some numerical technique is required to arrive at a solution. So in the next section the existing solution techniques are reviewed. Finally, the problem of large scale systems is tackled.

Chapter 3 introduces artificial neural networks. What are they? How do they work? Where can they be applied? How are they implemented? A background information about the neural networks organised in four sections attempting to find the answers to these questions is provided in this chapter.

The material presented in the next four chapters can be regarded as the major contribution of this thesis. It can be divided into two main streams: the usage of neural networks for the optimization problems treated in Chapter 4 and Chapter 5 and the application of the classification/clustering neural networks to fault diagnosis and water state interpretation tasks covered in Chapter 6 and Chapter 7.

Chapter 4 describes the application of analog neural networks for water network state estimation. Several neural networks for solving overdetermined systems of linear equations according to least squares (LS) criterion, least absolute value (LAV) criterion and the combination of these two criterias are presented. The problems of bad data rejection, ill-conditioning and arriving at the solution within a predefined period of time are addressed. Computational results for a realistic 34-node water network are also given.

In Chapter 5 a neural network approach to confidence limit analysis is discussed. An integrated neural system, based on NNs from Chapter 4, for state estimation and CLA is described in detail. Its performance and results produced in a form of state vector with corresponding confidence limits for the same 34-node water network are reported.

Chapter 6 documents the development of a new general fuzzy classification and clustering neural network that is able to process input patterns in both deterministic and fuzzy forms, combines the supervised and unsupervised learning algorithms, can produce fuzzy or crisp classification decisions and can grow to meet the demands of the problem.

A completely new approach to fault detection and identification in water distribution systems based on neural pattern recognition system is proposed and presented in Chapter 7. The emphasis is put on diagnosis of topology errors (e.g. leakages, wrong statuses of valves). Strengths and limitations of the neural recognition systems trained for state estimates and residuals including confidence limits are investigated. The computational results are given for the large training and data sets covering 24 hour of water distribution network operations.

Finally, Chapter 8 presents the main conclusions of the project and some suggestions for further research.

Chapter 2

Mathematical modelling of water distribution networks (WDN)

2.1. Basic laws in WDN

Solving many water distribution design and operation problems requires an understanding of the equations of closed conduit hydraulics. Usually, as it will be shown later in this chapter, the solution process involves simultaneous consideration of the energy and continuity equations and some independent relationship describing head loss.

The most important equations are the continuity, momentum, and energy equations. Since the problem of steady-state estimation is treated in this thesis the integral forms (taking into consideration average velocity and pressure) of continuity and energy equations are presented. The momentum equation, that may be used directly to evaluate the force causing a change of momentum in a fluid, have been omitted. The reason for this omission is the fact that the applications where the momentum equation is used include:

- determining forces on pipe bends and junctions, nozzles and hydraulic machines - useful for designing the water network; or
- solving problems when the flow is unsteady.

The project and problems described in this thesis are not concerned with either of these applications. For the similar reason the differential forms of these equations, used when information on such things as velocity distribution within the pipe is required, are also neglected. More information on dynamic behaviour of the fluids can be found in (Casey, 1992; Chadwick & Morfett, 1986; Merritt, 1967; Walski, 1984).

2.1.1. Continuity equation

During any time interval δt, the principle of conservation of mass implies that for any control volume the mass flow entering minus the mass flow leaving equals the change of mass within the control volume. It can be written as follows

$$m_{in} - m_{out} = \Delta m_{store} \qquad \text{(EQ 2-1)}$$

where: m_{in} - mass flow entering

m_{out} - mass flow leaving

Δm_{store} - change of mass stored

If the flow is steady, then the mass must be entering (or leaving) the volume at a constant rate. If we further restrict our attention to incompressible flow, then the mass of fluid within the control volume must remain fixed. In other words, the change of mass within the control volume is zero.

Taking these assumptions into consideration and knowing that:

$$q = \frac{m}{\rho t} \qquad \text{(EQ 2-2)}$$

where: ρ - density of fluid (water)

t - unit of time

m - mass flow

q - volumetric flow

we can rewrite the equation (EQ 2-1):

$$q_{in} - q_{out} = 0 \qquad \text{(EQ 2-3)}$$

This form of continuity equation will be used for water distribution problems discussed later in this work.

2.1.2. Energy equation

In general terms for the fluid (water) flowing in a pipe the energy equation states that, given the energy at the entry to the pipe, the energy at the exit from the pipe equals the energy at the entry, plus the net work done on the fluid (work done on water minus work done by water), minus any energy losses due to friction.

In a form of mathematical equation this can be written as follows:

$$E_{ex} = E_{en} + W - H \qquad \text{(EQ 2-4)}$$

where: E_{ex} - energy at the exit point

E_{en} - energy at the entry point

W - net work done on the fluid

H - friction energy loss

This energy equation in simplified form for a pipe loop will be presented and used with appropriate head-loss-flow formula when the models of the network will be discussed.

2.1.3. Types of flow and head loss formulas

Before the end of 19th century classical hydrodynamicists had long been puzzled by certain aspects of flow which did not conform to the known mathematical formulations. Towards the end of the 19th century, Reynolds designed an experiment in which a filament of dye was injected into a flow of water. The discharge was carefully controlled, and passed through a glass tube so that observations could be made. Reynolds discovered that the dye filament would flow smoothly along the tube as long as the velocities remained very low. If the discharge rate was increased gradually, a point was reached at which the filament became wavy. A small further increase in discharge was then sufficient to trigger a vigorous eddying motion, and the dye mixed completely with water.

Thus this experiment demonstrated that there were two kinds of flow - laminar and turbulent. Reynolds found that transition from laminar to turbulent flow occurred at a critical velocity for a given pipe and fluid. He expressed his results in terms of the dimensionless parameter, *Re*, called Reynolds number

$$Re = \frac{VD}{\nu} \qquad \text{(EQ 2-5)}$$

where D is the pipe diameter, V is the average velocity of flow and ν is the kinematic viscosity.

He found that for *Re* less than about 2000 the flow was always laminar, and that for *Re* greater than about 4000 the flow was always turbulent. For *Re* between 2000 and 4000, he found that the flow could be either laminar or turbulent, and termed this the transition region.

In a further set of experiments, he found that for laminar flow the frictional head loss in a pipe was proportional to the velocity, and that for turbulent flow the head loss was proportional to the square of the velocity.

These two results had been previously determined by Hagen and Poiseille (h~V) and Darcy and Weisbach (h~V^2), but it was Reynolds who put these equations in the context of laminar and turbulent flow.

Since most of the flows which are encountered in water distribution systems are turbulent flows we restrict our considerations to turbulent flows.

The Darcy-Weisbach head loss equation for turbulent flows can be written as follows:

$$h = \lambda\frac{LV^2}{2gD} \qquad \text{(EQ 2-6)}$$

where λ is the pipe friction factor, L is the pipe length and g is the acceleration due to the gravity.

The original investigators presumed that the friction factor was constant. Nikuradse, however, found that the turbulent flow could be divided into three regions and that the value of friction factor depends on relative roughness (*k/D*) of the pipe and *Re*. These three kinds of turbulent flow can be described as follow:

- *Smooth turbulence* - the limiting line of turbulent flow that is approached by all values of relative roughness (*k/D*) as *Re* decreases.
- *Transitional turbulence* - the region in which λ varies with both *Re* and *k/D*. In practice, most of pipe flow lies within this region.
- *Rough turbulence* - the region in which λ remains constant for a given *k/D*, and is independent of *Re*.

The following equation:

$$\frac{1}{\sqrt{\lambda}} = -2\log\left(\frac{k}{3.7D} + \frac{2.51}{Re\sqrt{\lambda}}\right) \qquad \text{(EQ 2-7)}$$

that relates the friction factor to *k/D* and *Re* is known as the Colebrook-White transition formula. It is applicable to the whole of the turbulent region for commercial pipes using an effective roughness value determined experimentally for each type of pipe.

Although the Darcy-Weisbach equation using Colebrook-White formula is the most accurate for the head loss assessment it had not been easy to use for engineers' hand calculations. There was a need for simpler empirical formulae. For water distribution system analysis the most commonly used of empirical formulas is the Hazen-Williams equation.

$$q_{ij} = 0.27746 C_{ij} D_{ij}^{2.63} \left(\frac{|H_j - H_i|}{L_{ij}} \right)^{0.54} \qquad \text{(EQ 2-8)}$$

where: q_{ij} - flow from node j to node i

C_{ij} - Hazen-Williams coefficient for pipe

D_{ij} - diameter of pipe

L_{ij} - length of pipe

H_j - head at node j

H_i - head at node i

$H_j > H_i$;

or for computer program implementation

$$q_{ij} = R_{ij}^{-0.54} (H_j - H_i) |H_j - H_i|^{-0.46} \qquad \text{(EQ 2-9)}$$

where R_{ij} is the resistance between nodes i and j given by

$$R_{ij} = 10.742 C_{ij}^{-1.85} L_{ij} D_{ij}^{-4.87} \qquad \text{(EQ 2-10)}$$

Most water system engineers have a very good feel for the meaning of the Hazen-Williams *C* factor, while pipe roughness remains a mystery to many practising engineers. It has to be stressed, however, that Hazen-Williams formula is valid in the transition turbulent flow region, and using it for flows outside this region may produce errors (Casey, 1992; Walski, 1984; Powell, 1992), so the caution must be taken in applying it.

Nevertheless for the purposes of this project the head loss has been calculated using the Hazen-Williams equation.

2.1.4. Head/flow models for other elements

Pumps

Pumps are usually modelled by a parabolic equation:

$$H_j - H_i = a_{ij}q_{ij}^2 + b_{ij}q_{ij} + c_{ij} \tag{EQ 2-11}$$

where: a_{ij}, b_{ij}, c_{ij} are empirically determined constants.

$$q_{ij} = -b_{ij} \pm \sqrt{\frac{b_{ij}^2 - 4a_{ij}(c_{ij} - |H_j - H_i|)}{2a_{ij}}} \tag{EQ 2-12}$$

Since the root $q_{ij} = -b_{ij} - \sqrt{\ldots}$ causes unstable operation the root $q_{ij} = -b_{ij} + \sqrt{\ldots}$ is chosen from (EQ 2-12) to calculate flow for a pump.

Valves

For the purpose of the modelling of control valves the same formula (EQ 2-8) can be used as for pipes. This formula has to be only slightly modified to take into account the controlling action of the valve.

The models presented below are written with the assumption of the ideal, steady state conditions and the dynamics of control valves depending on the physical construction of the valve are neglected.

- pressure reducing valve - reducing a pressure immediately downstream of its position in the pipeline to a pre-determined value H_{PRV}

$$q_{ij} = \begin{cases} R_{ij}^{-0.54}|H_j - H_i|^{0.54} & \text{if } H_{PRV} \ge H_j > H_i \\ R_{ij}^{-0.54}|H_{PRV} - H_i|^{0.54} & \text{if } H_i < H_{PRV} < H_j \\ 0 & \text{if } H_{PRV} \le H_i \end{cases} \tag{EQ 2-13}$$

- pressure sustaining valve - sustaining a pre-determined pressure H_{PSV} immediately upstream of its position in the pipeline

$$q_{ij} = \begin{cases} R_{ij}^{-0.54}|H_j - H_i|^{0.54} & \text{if } H_{PSV} \le H_i < H_j \\ R_{ij}^{-0.54}|H_{PSV} - H_i|^{0.54} & \text{if } H_i < H_{PSV} < H_j \\ 0 & \text{if } H_{PSV} \ge H_j \end{cases} \tag{EQ 2-14}$$

- non-return valve - also called check valve, retaining valve, reflux valve or foot valve is the valve through which flow can proceed in one direction only. A reversal of flow causes the valve to close and remain closed until flow is re-established in the unique direction.

$$q_{ij} = \begin{cases} R_{ij}^{-0.54}\left|H_j - H_i\right|^{0.54} & \text{if } H_j > H_i \\ 0 & \text{if } H_j \le H_i \end{cases} \qquad \text{(EQ 2-15)}$$

- control valve - actuating the flow in the pipe

$$q_{ij} = \varphi R_{ij}^{-0.54}\left|H_j - H_i\right|^{0.54} \qquad \text{(EQ 2-16)}$$

where: $0 \le \varphi \le 1$

2.2. Network modelling

Water *network* modelling is essentially combining all the mathematical models of elements and laws presented in previous section into a consistent set of equations describing as accurately as possible the behaviour of the system at hand.

A typical water distribution system may serve many thousands of consumers and may consist of a large number of interacting elements. Even with the computing power that is available today, modelling on this scale is impractical for most applications. Therefore, modelling on more manageable scale is required.

There are methods of combining system elements to form an "equivalent" single element (Hamberg & Shamir, 1988; Powell, 1992; Walski, 1984). For instance, pipes connected in series or parallel can be combined to form a single pipe element in the model, a small loop of pipes can be combined to form a simpler pipe junction in the model, or pumps working in parallel can be combined to form a single "equivalent" pump model. In similar manner groups of consumers can be lumped together and represented in the model as a single node with consumption being a sum of individual consumptions. It has to be stressed, however, that extreme care must be taken when reducing the network in this way because of the danger of reducing the potential model accuracy (Eggener & Polkowski, 1976).

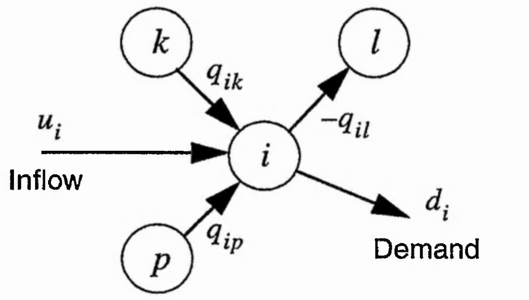
2.2.1. Network models

In previous section the continuity (EQ 2-3) and energy (EQ 2-4) equations were presented in a context of a single node and pipe. Here they are rewritten in a more detailed form for a water network consisting of p pipes and n nodes.

Applying the continuity equation (EQ 2-3) to n nodes we obtain a set of so called nodal equations:

$$\sum_{j \in \Omega_i} q_{ij} + u_i = d_i \qquad i = 1 \ldots n \qquad \text{(EQ 2-17)}$$

where n is the number of nodes in the network, Ω_i is a set of nodes connected to node i, u_i is the inflow to node i, and d_i is a demand (also called consumption or load) in node i.



$$q_{ik} - q_{il} + q_{ip} + u_i = d_i$$

Figure 2-1: Continuity law for *i*-th node.

Applying the energy equation (EQ 2-4) written for a single pipe to a pipe loop, since the beginning and ending points are the same ($E_{en} = E_{ex}$), the head loss around the loop is equal to zero:

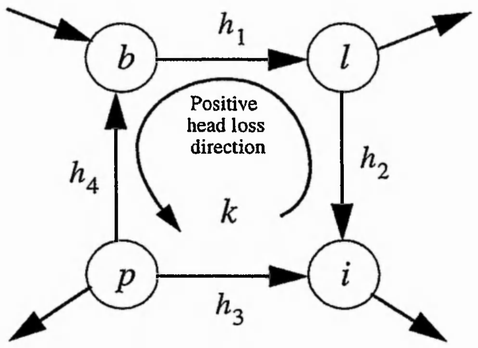
$$\sum_{i \in \Omega_k} h_i = 0 \qquad k = 1 \ldots m \qquad \text{(EQ 2-18)}$$

where h_i is the head loss in i-th pipe, Ω_k is a set of pipes which comprise the k-th loop, k is the number of loop.

The number of independent loops l for a network with n nodes and p pipes can be determined from the rule:

$$l=p-n+1 \qquad \text{(EQ 2-19)}$$

An independent loop is a loop for which the energy equation cannot be derived from the energy equations written for the other loops.



$$h_1 + h_2 - h_3 + h_4 = 0$$

Figure 2-2: Energy law for *k*-th loop.

The continuity law (EQ 2-17) and energy law (EQ 2-18) can be now coupled with a suitable head-flow formula (e.g. Hazen-Williams or Darcy-Weisbach/Colebrook-White) to construct a set of network equations.

These network equations relate either the network's nodal pressures or the network's flows to measurement or pseudomeasurement values and can be expressed by the following equation:

$$g(\mathbf{x})=\mathbf{z} \qquad \text{(EQ 2-20)}$$

where **x** is an *n*-dimensional *state vector* which can consist of nodal pressures and/or flows, **z** is the *measurement vector* that consists of real measurements values and pseudomeasurements, and g() is a nonlinear *network function.*

The state vector is made up of *n* independent state variables. It may include nodal pressure variables and/or flow variables, but must be sufficient to completely specify the operating state of the system. When this is the case, any other system variable can be calculated directly from **x**.

There are three main ways of constructing the network equation.

In the first of these methods the network equation is set up by using the flow rates as unknowns (state variables) and writing one energy equation for each independent loop and one continuity equation for each node. It results in deriving *p* equations (where *p* is the number of pipes) called the flow equations since the flows are the unknowns.

Second approach to setting up the network equation is to write energy equations in such a way that, for an initial solution, the continuity equation is satisfied. Then it becomes a matter of correcting the flows in each loop in such a way that the continuity equations are not violated. This can be done by adding a correction to the flow to every pipe in the loop. These corrections are the unknowns in a set of *l* equations - one for each loop.

In the third method the network equation is derived by combining the continuity equation for each node with head loss equations. The state vector in this method consists of nodal pressures and inflows into fixed head nodes if any of such nodes are in the system.

It has been found (Osiadacz, 1988) that the last method, based on node equations, is much better for solving large networks than the other two. Therefore, this method is used for setting up network equation throughout this work.

2.2.2. Numerical solution techniques

Since the network equation is nonlinear and direct solution of the systems of nonlinear simultaneous equations is not feasible, it is necessary to use iterative solution methods. In general, these methods start with an estimated solution which is iteratively refined by repeated corrections until the deviation from the true solution is reduced to an acceptable tolerance value. One of the earliest methods is the one of Hardy-Cross based on loop by loop iterative computations. This method was very useful for hand calculations because only one loop flow correction is made at a time. This, however, is also the reason for very slow convergence.

Definitely more powerful and faster method is the Newton-Raphson method. It obtains the solution to a system of nonlinear equations by iteratively solving system of linear equations. This method has been adopted in this work for solving the network equation. Hence, more detailed explanation of the Newton-Raphson method in a context of the network equation is given below.

Expanding g(**x**) by an initial guess of the state vector $\boldsymbol{x}^{(0)}$, using a first-order Taylor series and defining $z^{(0)} = \boldsymbol{g}(\boldsymbol{x}^{(0)})$, we obtain

$$z = z^{(0)} + \Delta z \qquad \text{(EQ 2-21)}$$

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}^{(0)}) + \boldsymbol{J}^{(0)}\Delta \boldsymbol{x} \qquad \text{(EQ 2-22)}$$

Using the linearised models (EQ 2-21), (EQ 2-22) and the network equation (EQ 2-20) we obtain the following set of linear equations:

$$J^{(k)} \Delta x^{(k)} = z - g(x^{(k)}) \qquad \text{(EQ 2-23)}$$

where: $J^{(k)} = \left.\frac{\partial g}{\partial x}\right|_{x = x^{(k)}}$ - Jacobian matrix evaluated at $x^{(k)}$

$\Delta x^{(k)}$ - the correction vector

$x^{(k)}$ - the current estimate of the state vector

z - the measurement vector

$g(x^{(k)})$ - the network function evaluated at $x^{(k)}$

k=0,1,... - step of the iterative solution finding process

Since the network equation (EQ 2-20) is nonlinear, the solution finding is an iterative process with the consecutive state estimates calculated by under-relaxation of the linear solution

$$x^{(k+1)} = x^{(k)} + \gamma \Delta x^{(k)}, \text{k=0,1,...} \qquad \text{(EQ 2-24)}$$

where γ is the coefficient taking values between 0.6 and 0.8.

If all elements of $\Delta x^{(k)}$ in *k*-th iteration are lower or equal to a predefined convergence accuracy, the iteration procedure stops. Otherwise, a new correction vector is calculated using equation (EQ 2-23) with $x^{(k+1)}$ instead of $x^{(k)}$.

2.2.3. Uncertainty of the network model and solution

A critical part of analysing a water distribution system is to combine the numerical techniques embodied in a computer model with a description of the physical system to arrive at a model of the system that can be used with confidence. In reality a model of a specific system consists not only of a computer program but also of data describing the system. Usually, these data are the weakest link in the modelling process. Computer programs can produce results that are accurate to several decimal places, but when the data used to produce the model are inaccurate one cannot expect to obtain reliable, accurate results.

As it has been mentioned before, the simulation of any complex system involves a certain degree of uncertainty that cannot be avoided. It is, however, essential to be aware of the potential sources of inaccuracies in the modelled system and their possible impact on the network solution.

There are two main sources of uncertainty and errors in water distribution system simulations. First is associated with the modelling of the physical elements and represents static (or slowly changing) inaccuracy of network model. The second source of uncertainty has dynamic nature and is associated with inaccurate predictions of consumptions and inaccuracies of measured values. While, for instance, the pipe's C factor or friction factor usually varies gradually over years or decades, consumptions and flows in the network change from minute to minute and are of unpredictable nature (How to predict that someone will turn on water somewhere at some particular moment?).

Model inaccuracy

There have been a lot of research work done in order to develop methods for improving the network model accuracy. One of the sources of inaccuracies of network model is a simplified representation of a physical system.

The effect of skeletonization on the results produced by simplified models was studied in (Eggener & Polkowski, 1976) and it was found that the resulting errors are small if the skeletonization is done properly. In the same work *C*-values were identified as the weakest factor in water network models, however, it was also suggested that with enough effort put into input data development, any practical degree of accuracy can be attained.

In many publications (Cesario & Davis, 1984; Coulbeck, 1984; Lansey, 1988; Ormsbee & Chase, 1988; Ormsbee & Wood, 1986; Shamir & Howard, 1977; Walski, 1984) the topic of network calibration was explored. The calibration is known as a process of network parameters adjusting (especially *C*-values and friction factors) so that the model predictions agree with observed pressures and flows.

In all the publications dealing with methods of assessing and improving the accuracy of network models it has been emphasized that models must be calibrated and recalibrated regularly. On the basis of the above briefly discussed research work it can be said that there will be only a small amount of residual inaccuracy if the network model is constructed and calibrated properly.

Consumption predictions and measurement uncertainty

The problem of measurement and pseudomeasurement uncertainty and their influence on accuracy of the network equation solution has received much less attention in the literature than the investigations of model accuracy and model calibration.

Constructing the network equation (EQ 2-20) it was assumed that the minimum data required to find the solution, namely at least one reference pressure measurement, the inflows into the network, and consumptions (or their predictions) at nodes, were known. In practise these values are measured with finite accuracy specified by the type of meter used or predicted in case of some nodal consumptions, making this type of data very unreliable.

One way of reducing the effect of variations of predictions is to consider a large number of consumers when individual use is evened out and the predictions can be carried out quite accurately (Fallside & Perry, 1975c; Sterling & Bargiela, 1985; Canu et al., 1990; Cubero, 1991). However, when the scale of prediction is reduced to a nodal level, with smaller number of consumers considered, accuracy suffers. Short-time water demand prediction methods based on time series analysis can be found in (Chen, 1988; Quevedo et al., 1988). There have been many suggestions for how nodal consumptions should be modelled and predicted (Suter & Newsome, 1988; Wright & Cleverly, 1988). These involved modelling different types of consumptions, e.g. domestic use, industrial use etc., separately and combining them to represent overall nodal consumption. However, the fact that predictions carry a substantial inaccuracy remains unchanged and some way of quantifying their impact on model performance is required.

Although accuracy of meters used in water distribution systems allows, in most of the cases, to obtain more reliable information than in case of predictions of nodal consumptions and other pseudomeasurements, the inaccuracies introduced in such a way into the system model cannot be overlooked. In (Bargiela & Hainsworth, 1988) techniques were shown relating the position and accuracy of the meter, to the corresponding benefit that can be derived from these measurements, in the process of mathematical modelling.

Another way of diminishing the effect of inaccuracies in measurements and pseudomeasurements on the solution is to redefine the method of solving the network equation. Since the number of unknowns is equal to the number of equations in the linearised model (EQ 2-23) of the network equation (EQ 2-20) used in the Newton-Raphson method, each inaccurate data has a huge influence on the solution. It could even lead to the case when there would be no solution to the set of equations (EQ 2-20) at all. The more robust method, known as the state estimation procedure, utilizing all available measurements and pseudomeasurements is, therefore, used. Using all available information results in constructing overdetermined set of equation (number of equations is greater than the number of unknowns) and the solution can no longer be found by simply solving a square set of equations. The solution finding problem has to be defined as the optimization of a suitably chosen cost function. On the other hand, these additional measurements, also known as redundant measurements since they are not absolutely necessary to arrive at some solution, allow the corrupted data to be rejected and to obtain a more reliable solution. The estimation procedures are discussed in more detail in Chapter 4.

The methodology and algorithms for quantifying the impact of measurement and pseudomeasurement inaccuracies on the state estimate vector in water distribution systems were first introduced in (Bargiela & Hainsworth, 1989) under the name of confidence limit analysis. This problem is also considered later in Chapter 5.

2.3. Large scale systems

On-line monitoring and control of a water distribution system requires an efficient and reliable state estimator.

The difficulties to satisfy the conflicting requirements on speed, accuracy, low memory occupation and capability of detecting and identifying anomalous data increase with the system's size. For some large-scale systems, because of their shear size, the estimation task is computationally so demanding that a conventional solution, using "integrated" methods, is impossible on the available computer or the solution computing time is so great that it is no longer suitable for on-line applications.

The above mentioned difficulties, together with the natural evolution towards hierarchical control strategies of the utility systems have influenced the conceptual foundations leading to multi-level state estimation approaches. Another factor that has also influenced and guided this evolution is the progress in developing the general hierarchical theory of large-scale systems.

The problems encountered when estimating the state of large scale systems and the possible solutions using hierarchical, decomposed techniques have been discussed in many publications (Bargiela, 1992; Brdys et al., 1990a; Brdys et al., 1990b; Brdys & Ulanicki, 1994; Coulbeck et al., 1988a; Coulbeck et al., 1988b; El-Keib et al., 1992; Fallside & Perry, 1975a; Fallside & Perry, 1975b; Hartley, 1996; Hartley & Bargiela, 1995; Hosseinzaman & Bargiela, 1992; Lin et al., 1989; Osiadacz & Salimi, 1988a; Osiadacz & Salimi, 1988b; Sundareshan & Elbanna, 1990a; Sundareshan & Elbanna, 1990b; Ulanicki, 1991; Ulanicki & Orr, 1991; Van Cutsem & Ribbens-Pavella, 1983).

The basic idea is to reduce the complexity associated with the "integrated" high order problem by decomposing it into lower order subproblems. In the case of the distributed systems, the physical structure of the problem can be utilised when decomposing it into a number of smaller subproblems. Each of these subproblems will be solved by a conventional centralised method. These are solved individually and the solutions are recombined in some way to achieve the solution of overall problem. To take into account the interactions between the subproblems, solution usually proceeds iteratively with information exchange between subproblems and a coordinating master problem. The manner in which coordination is achieved characterises the different decomposition methods.

In practise, the hierarchical (also called multi-level, decomposed, decentralised) algorithm may be implemented within the two different structures:

i) *aggregation of subsystems* - where the system comprises a certain number of subsystems, each controlled by a local control centre and coordinated by an upper supervisory dispatching centre. For implementation of this method, a set of spatially distributed computers can be used, each of them in charge of a subsystem.

ii) *decomposition into subsystems* - where the system is decomposed into subsystems but is controlled as a whole in a single dispatching centre. In this case, two computer structures may be envisaged for implementing this decomposed algorithm:

- a multiprocessor system, composed for example of a set of microprocessors operated as peripherals of a main computer. Each microprocessor is in charge of one subsystem whereas the main computer deals with synchronization, data sharing and coordination tasks. In fact, this structure combines sequential and parallel operations. It is well suited to estimation algorithms having considerable information exchanges, owing to the high-speed links connecting main computer with microprocessors. Examples of this type of implementation for water networks can be found in (Bargiela, 1992; Hartley, 1996; Hartley & Bargiela, 1995; Hosseinzaman & Bargiela, 1992).

- a single computer. In this case, the various procedures of all levels are carried out simultaneously. Of course, the data are all centralized in this computer. Although practically all decentralised algorithms have the potential to be implemented on parallel architectures a lot of reported results of hierarchical method implementations have been obtained via simulation on a single computer (El-Keib et al., 1992; Osiadacz & Salimi, 1988a; Osiadacz & Salimi, 1988b).

The advantages of using the hierarchical algorithms, like the increased speed of solution finding or dimension reduction of problems to be solved, come at a price. The decentralised state estimation algorithms are more complicated than the conventional ones. This is due to the need for an additional coordination level where, among others, the problem of unequal estimates in common links or nodes of the individual subnetworks, has to be resolved. Theoretical developments of estimation theory show that, given a set of all measurements in the network, the most accurate estimate is the "integrated" one, i.e. that obtained by minimizing a suitable global criterion. In this sense some of the hierarchical algorithms are not optimal.

As it has been put in (Van Cutsem & Ribbens-Pavella, 1983) the integrated state estimator is "*robust by experience and optimal by definition*" (provided, of course, that the estimation algorithm is properly chosen), but it requires heavy information flows for modelling and comparatively large computing times both for the estimation procedure itself and for bad data analysis.

The aim of neural network approach to the state estimation problem and confidence limit analysis, presented in Chapter 4 and Chapter 5 respectively, is to utilise the naturally parallel structure of neural networks promising high computational efficiency while optimizing the global criteria ensuring the optimal solution. This would combine the efficiency of hierarchical algorithms implemented on multiprocessor systems with known robustness and optimality of "integrated" estimators.

Chapter 3

Artificial neural networks - an overview

3.1. What is a neural network?

There are various points of view as to the nature of an artificial neural net. For example, is it a specialized piece of hardware or a computer program? We shall take the view that neural nets are basically mathematical models of information processing. They provide a method of representing relationships that is quite different from Turing machines or computers with stored programs. As with other numerical methods, the availability of computer resources, either software or hardware, greatly enhances the usefulness of the approach, especially for large problems.

3.1.1. Biological neural systems

The human information processing system consists of the biological brain. The basic building block of the nervous system is the neuron, the cell that communicates information to and from the various parts of the body. Figure 3-1 shows a simplified representation of a biological neuron. The neuron consists of a cell body called *soma*, several spine-like extensions of the cell body called *dendrites*, and a single nerve fibre called the *axon* that branches out from the soma and connects to many other neurons.

The many dendrites receive signals from other neurons. The connections between neurons occur either on the cell body or on the dendrites at junctions called synapses. The signals are electric impulses that are transmitted across synaptic gap by means of chemical process. A helpful analogy is to view the axons and dendrites as insulated conductors of various impedance that transmit electrical signals to the neuron (Churchland, 1986; Kandel & Schwartz, 1985). The nervous system is constructed of billions of neurons with the axon from one neuron branching out and connecting to as

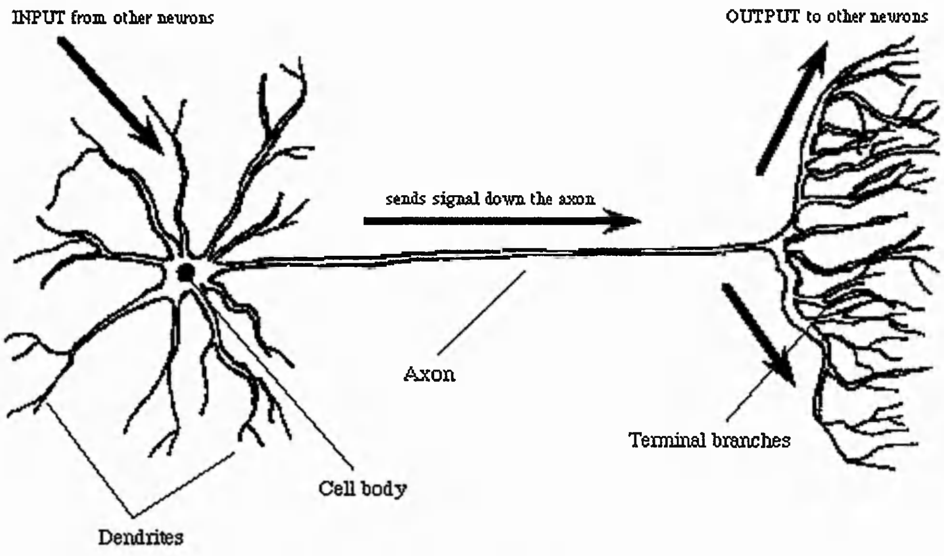


Figure 3-1: Biological neuron

many as 10,000 other neurons. All the neurons - interconnected by axons and dendrites that carry signals regulated by synapses - create a neural network.

3.1.2. Artificial neural networks

An artificial neural network is an information-processing system that has certain performance characteristics in common with biological neural networks. The extent to which a neural network models a particular biological neural system varies. For some researchers, this is a primary concern, for others, the ability of the net to perform useful tasks (such as approximation of a function) is more important than the biological plausibility of the net. Although our interest lies almost exclusively in the computational capabilities of neural networks, we shall briefly present some features of biological neurons that may help to clarify the most important characteristics of artificial neural networks.

An artificial neural network is characterized by:

a) its topology of interconnected neurons with their non-linear activation functions (called its *architecture*),

b) its method of encoding information (called its *training* or *learning* algorithm).

Artificial neural networks are made up of large number of individual models of the biological neurons (artificial neurons). Each neuron is connected to other neurons by means of directional communication links, each with an associated weight. The neuron models that are used are typically much simplified versions of the actions of a real neuron. The weights represent the information used by the net in solving a particular problem.

Several key features of the processing elements of artificial neural networks are suggested by the properties of biological neurons:

- The processing element receives many signals.
- Signals may be modified by weight at the receiving synapse.
- The processing element sums the weighted inputs.
- Under appropriate circumstances (sufficient input), the neuron transmits a single output.
- The output from a particular neuron may go to many other neurons.
- Information processing is local.
- Memory is distributed: a) long memory resides in the neurons' synapses or weights, b) short-memory corresponds to the signals sent by the neurons.
- A synapse's strength may be modified by experience.
- Neurotransmitters for synapses may be exitatory or inhibitory.

Yet another important characteristic that artificial neural networks share with biological neural systems is *fault tolerance*. Biological neural systems are fault tolerant in two respects. First, they are able to recognize many input signals that are similar but not identical to any input that was seen before. Second, damage to individual neurons can occur in the brain without a severe degradation in its overall performance (Hopfield, 1982; Hopfield et al., 1983; Hopfield, 1984). If a portion of a brain is removed, the knowledge of the concept or idea is still retained through the redundant, distributed encoding of information. In a similar manner, artificial neural networks can be designed to be insensitive to small damage to the network, and the network can be retrained in cases of significant damage.

In the final attempt to answer the question: What is a neural network? let us quote the definition taken from (Hecht-Nielsen, 1988):

> A neural network is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and carry out localized information processing operations)

interconnected together with unidirectional signal channels called connections. Each processing element has a single output connection which branches ("fans out") into as many collateral connections as desired (each carrying the same signal - the processing element output signal). The processing element output signal can be of any mathematical type desired. All of the processing that goes on within each processing element must be completely local: i.e., it must depend only upon the current values of the input signal arriving at the processing element via impinging connections and upon values stored in the processing element's local memory.

3.2. Fundamental features of ANNs

After an attempt to explain, in general terms, what an artificial neural network is and where the inspiration for neural computing came from, the subsequent sections will present typical neural network architectures and training algorithms. For the reason that this thesis is not dedicated to neural networks themselves but rather their applications to water systems' specific problems, the remaining sections of this chapter will have general and grossly introductory character. The more detailed description will be given: for analog neural networks used in state estimation process in Chapter 4 and for fuzzy classification/clustering neural algorithms in Chapter 6.

3.2.1. Artificial neurons and activation functions

Artificial neurons

Artificial neurons, also referred to as nodes or processing elements, are the ANN components where most, if not all, of the computing is done. The most commonly used neuron model is depicted in Figure 3-2 and is based on the model proposed by McCulloch and Pitts in 1943 (McCulloch & Pitts, 1943). Each neuron input, x_1 - x_n, is weighed by the adjustable values w_1 - w_n. A bias, or offset, in the node is characterized by an additional constant input of 1 weighted by the value of w_0. The output, y, is obtained by

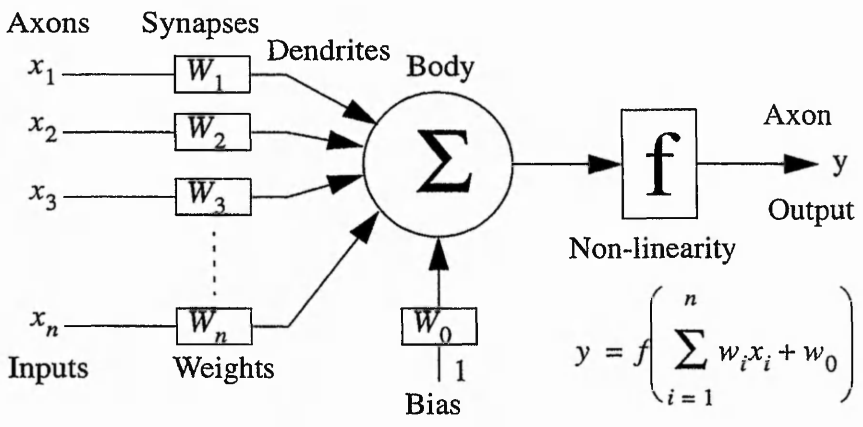


Figure 3-2: McCulloch-Pitts model of neuron.

summing the weighted inputs to the neuron and passing the result through a non-linear activation function, f(). Mathematically this operation is defined as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + w_0\right) \quad \text{(EQ 3-1)}$$

Various types of non-linearity are possible and some of these are shown below.

Activation functions

Activation functions, also called threshold functions or squashing functions, map the neuron's infinite domain (the input) to a prespecified range (the output). Four common activation functions are the linear, ramp, step, and sigmoid functions. Table 3-1 shows the mathematical equations describing these functions and their typical shapes.

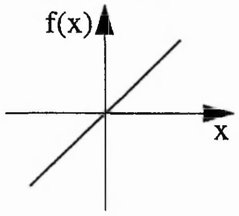
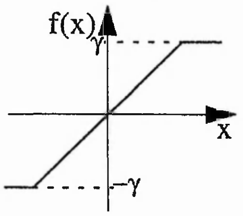
Name and mathematical description	Shape	Remarks
Linear function $f(x) = \alpha x$		α is a real-valued constant that regulates the magnification of the neuron activity x.
Ramp function $f(x) = \begin{cases} \gamma & \text{if } x \geq \gamma \\ x & \text{if } \lvert x \rvert < \gamma \\ -\gamma & \text{if } x \leq -\gamma \end{cases}$		The output is bounded to the range $[-\gamma,+\gamma]$. Values γ and $-\gamma$ are commonly referred to as the saturation levels.

Table 3-1: Four common activation functions.

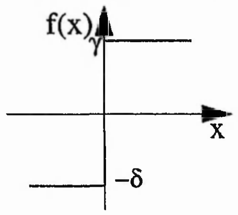
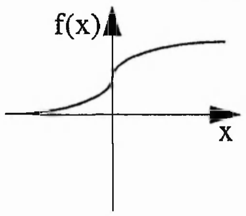
Name and mathematical description	Shape	Remarks
Step function $f(x) = \begin{cases} \gamma & \text{if } x>0 \\ -\delta & \text{otherwise} \end{cases}$		Step function respond only to the sign of the input, emitting +γ if the input sum is positive and -δ if it is not. γ and δ are positive scalars. Often step function is binary in nature emitting a 1 if x>0 and 0 otherwise.
Sigmoid function $f(x) = (1+e^{-x})^{-1}$		Sigmoid function is bounded, monotonic, non-decreasing function that provides a graded, nonlinear response. The saturation levels are 0 and 1. In the ANNs described in Chapter 4 another type of sigmoid function is used, the hyperbolic tangent f(x)=βtanh(x) which has saturation levels at -β and +β

Table 3-1: Four common activation functions.

3.2.2. Typical architectures

ANN architectures, or topologies, are formed by organizing neurons into layers (also called fields or slabs) and linking them with weighted interconnections.

There are three primary neuron interconnection schemes: lateral connections, inter-layer connections, and recurrent connections. *Lateral connections* are connections between neurons in the same layer of neurons. *Inter-layer connections* are connections between neurons in different layers. And finally, *recurrent connections* are connections that loop and connect back to the same neuron.

Interlayer connection signals propagate in one of two ways, either forward or back. *Feedforward* signals only allow information to flow amongst neurons in one direction. *Feedback* signals allow information to flow amongst neurons in either direction and/or recursively. On the basis of these two types of signal propagation the difference between two methods of information recall in ANNs can be defined as follow. During feedforward recall, the input cue is passed through the memory, represented by the weights W, and produces an output response in one pass. During feedback recall, the input cue is passed through the memory and produces an output response that is, in turn, fed back into the memory until the cue and response cease to change. The neural networks in Chapter 4 are the examples of feedback recall.

Layer configurations combine layers of neurons, information flow and connection schemes into a coherent architecture. Layer configurations include lateral feedback, layer feedforward, and layer feedback. A layer that receives input signals from the environment is called an input layer and a layer that emits signals to the environment is called an output layer. Any layers that lie between input and output layers are called hidden layers and have no direct contact with the environment. Figure 3-3 illustrates four common ANN topologies.

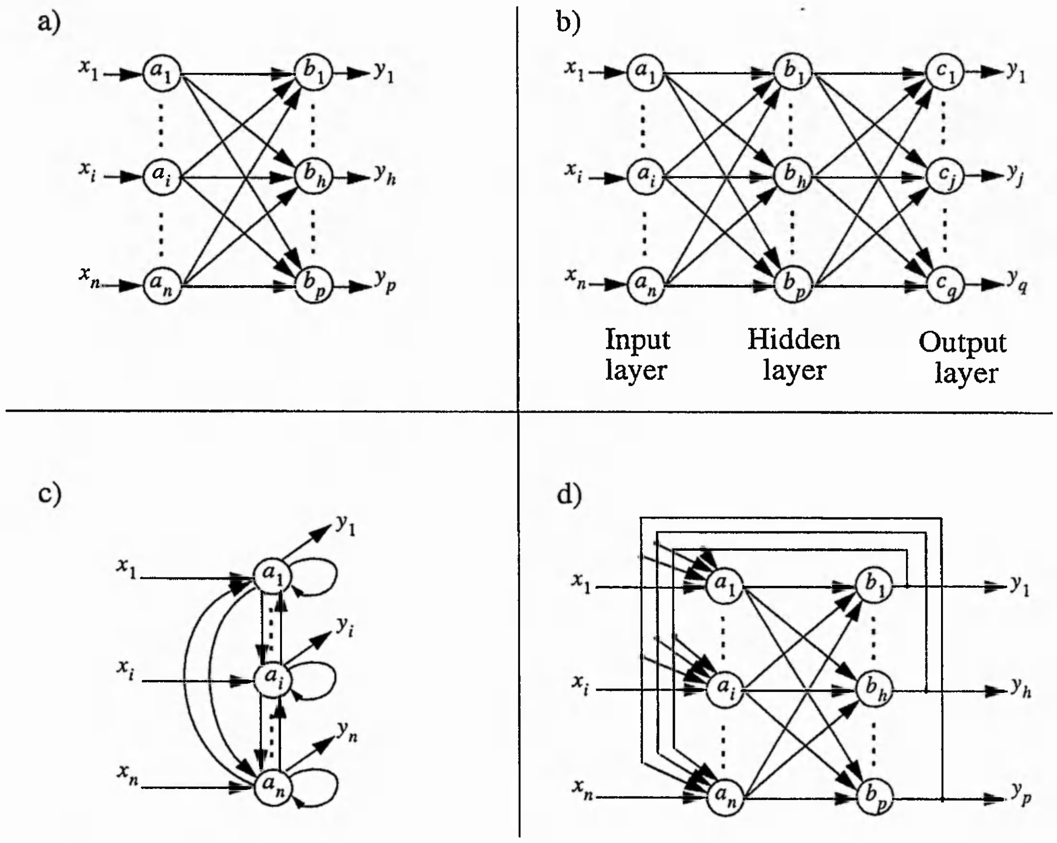


Figure 3-3: Four common ANN architectures: a) two-layer feedforward ANN; b) three-layer feedforward ANN; c) one-layer lateral feedback ANN; d) two-layer feedback ANN.

3.2.3. Training/Learning algorithms

In addition to the architectures, the method of setting the values of the weights (learning or training) is an important distinguishing factor of different neural nets. As it has been pointed out in (Hassoun, 1995), in the context of artificial neural networks, the process of learning is best viewed as an optimization process. More precisely, the learning process can be viewed as "search" in a multidimensional (weight) space for a solution, which gradually optimizes a prespecified objective (criterion) function. This view allowed Hassoun to unify a wide range of existing learning rules which otherwise could have looked more like a diverse variety of learning procedures.

All learning methods can be classified into two categories, supervised learning and unsupervised learning, although aspects of each may co-exist in a given architecture. In addition, there are nets whose weights are fixed without an iterative training process.

In *supervised learning* (also called *learning with a teacher*) each input vector, pattern or signal is presented with an associated target output vector. Usually the weights are gradually updated with each step of the learning process so that the error between the desired (given) target and the network's output is reduced.

On the other hand, *unsupervised learning*, also referred to as self-organization, is a process that incorporates no external teacher. Unsupervised learning involves the clustering or detection of similarities among unlabelled patterns of a given data set. Here, the weights and the outputs of the network are usually expected to converge to representations of the input data.

There is some ambiguity in the labelling of training methods as supervised or unsupervised and some authors find a third category, reinforcement learning or self-supervised learning, useful. *Reinforcement learning* involves updating the network's weights in response to an "evaluative" teacher signal; this differs from supervised learning, where the teacher signal is the "correct answer".

In general, however, there is a useful correspondence between the type of training that is appropriate and the type of problem we wish to solve. Some examples of ANN applications are given in the next section.

3.3. ANN applications

The purpose of this section is to give a sample of various areas of ANN applications and to illustrate a strong preference for using certain types of neural nets to solve certain types of problems.

Multilayer, feedforward, supervised ANN applications

Among the supervised learning methods for multilayer neural nets the backpropagation algorithm is by far the most popular. Backpropagation and its variations have been applied to a wide variety of problems, including pattern recognition, signal processing, image compression, speech recognition, medical diagnosis, prediction, nonlinear system modelling, and control.

One of the earliest applications of backpropagation was the system known as NETtalk that converts English text into speech (Sejnowski & Rosenberg, 1987). Another example of a multilayer feedforward ANN application is a neural based adaptive interface system, known as Glove-Talk, that maps hand gestures to speech (Fels & Hinton, 1993).

The recognition of handwritten digits is a classic problem in pattern recognition. Specifically, the Postal Service is interested in the recognition of handwritten ZIP codes on pieces of mail. A backpropagation network has been designed to recognize segmented numerals digitized from handwritten ZIP codes that appeared on U.S. mail (Le Cun et al., 1989).

ALVINN (autonomous land vehicle in a neural network) - a backpropagation-trained feedforward network designed to drive a modified Chevy van (Pomerleau, 1991) - is an example of a successful application using sensor data in real time to perform a real-world perception-control task.

Clinical diagnosis is often fraught with great difficulty because multiple, often unrelated disease states can surface with very similar historical, symptomalogic, and clinical data. As a result, physicians' accuracy in diagnosing such diseases is often poor. Feedforward multilayer neural networks trained with backpropagation have been reported to exhibit improved clinical diagnosis over physicians and traditional expert-system approaches (Bounds et al., 1988; Yoon et al., 1989; Baxt, 1990).

One of the major objectives for the management of a water supply and distribution system is the forecasting of the daily demand. The multilayer feedforward ANNs, reported in (Canu et al., 1990; Cubero, 1991), have been used to accomplish this task.

Feedforward, unsupervised ANN applications

The best known ANN in this group is the self-organizing map, developed by Kohonen, which has the special property of effectively creating spatially organized "internal representations" of various features of input signals and their abstractions. The self-organizing map has been particularly successful in various pattern recognition tasks involving very noisy signals.

One of the applications demonstrating the power of the map method when dealing with difficult stochastic signals is the area of speaker-independent recognition of speech. The example of the self-organizing map application to speech recognition is the "phonetic typewriter" net (Kohonen, 1988).

Other areas where self-organizing maps have been successfully used include control of robot arm (Graf & LaLonde, 1988; Veelenturf, 1995), EEG signal analysis (Veelenturf, 1995), control of industrial processes, especially diffusion processes in the production of semiconductor substrates (Marks & Goser, 1988).

Feedback, unsupervised ANN applications

Dynamic associative memories (DAMs), the most representative in this group, are a class of recurrent ANNs that utilize a learning/recording algorithm to store vector patterns as stable memory states. A part of the DAMs are Hopfield networks that have been successfully applied to many combinatorial optimization problems- situations that require the minimization of multiple-constraint cost function to determine the set of optimal system parameters.

An example of the optimization problem, that was addressed in (Hopfield & Tank, 1985) using recurrent neural network, is the classical travelling salesperson problem. A salesperson wants to visit n cities, once each, along a path that ends at the initial city. The problem is to perform this loop in such a way as to minimize the total mileage. An interesting feature of the solution proposed by Hopfield and Tank is the fact that weights are defined by the problem (they are the distances between the cities) and not set using some learning method.

A similar ANN has been used for solving the water network state estimation presented in Chapter 4.

Other than combinatorial optimization applications, the Hopfield ANN's ability to reconstruct entire patterns from partial cues stands out as one of its primary application strengths. In addition, the Hopfield ANN's nearest-neighbour response and fault tolerance qualities are also appealing. Because of these qualities, pattern classification and noise removal from patterns are key Hopfield network's applications.

One more type of ANN and its applications is worth mentioning here: ART (Adaptive Resonance Theory) clustering neural network. This unsupervised ANN has been particularly of interest for us due to the fact that the second part of this project involves developing clustering/classification neural net for water network state classification.

The reported applications of the ART network include the clustering of motor unit potentials in the electromyogram (EMG) signal (Wang, 1991) or radar signal classification (Mertz et al., 1992).

3.4. ANN implementations

A wide variety of ANN implementations have been developed that attempt to streamline the computation and take advantage of the inherent parallelism. In (Simpson, 1990) we find that the broad range of existing ANN implementations can be placed into three categories:

1) *computer implementations* - defined as any software implementation that is created on a machine that was not made explicitly for ANN processing
2) *electronic implementations* - defined as any electronic implementation that is made with the sole purpose of performing ANN processing
3) *optical/electro-optical implementations* - defined as any ANN implementation that involves the use of optical components

3.4.1. Computer implementations

Computer implementations of ANNs can be further divided into implementations on supercomputers, massively parallel computers and conventional computers. Although supercomputers (e.g. Cray from Cray Research, Nec's SX-2, etc.) and massively parallel computers (e.g. Inmos Transputers, Texas TMS320, CAPP of University of Massachusetts, etc.) were not specially designed for neural implementations, in many cases very high performance rates have been obtained. However, very often these powerful computers are limited by their inability to provide efficient processing with extensive human interaction, the expensive purchase/operation costs or very little portability due to the fact that implementations are programmed in languages that are often unique to each machine.

On the other hand conventional computers using any programming language can implement virtually any ANN; it just takes longer to process. As a matter of fact majority of ANN implementations are software simulations and there are several versatile ANN simulation packages available today for common computers (neural-net-faq.html). The conventional computer is the first step to any engineering project concerning ANNs and, depending on the size of application, it could be sufficient for the entire project.

3.4.2. Electronic implementations

- ***PC accelerator cards and neurocomputers***

Very large networks may only be practical with specialized neural network hardware. While large general purpose parallel machines can certainly provide sufficient performance, cheaper alternatives are available with co-processor, or accelerator, cards for PC. There are also more elaborate neurocomputers with multiple boards and extensive software environments. Such neurocomputers may be expensive but are still much cheaper than the big parallel mainframes.

Several of the now available accelerator cards simply use fast RISC processors (e.g. NeurodynamX XR50 - Intel i860, Vision Harvest NeuroSim - Intel i860) or DSPs (e.g. BrainMaker Accel. - TI TMS320C25 DSP, Neural Tech NT6000 - TI TMS320C20 DSP) as coprocessors to speed up the network processing. A disadvantage with many such co-processors cards is that they have to use the slow PC bus.

Examples of neurocomputers include: the Adaptive Solutions CNAPS that uses the Inova N64000 chip on VME boards in a custom cabinet run from a UNIX host; the HNC SNAP Neurocomputer with 2 VME boards, each with four NAP 100 chips; the Siemens SYNAPSE-1 using a systolic array of 8 MA-16 chips in a custom cabinet with a Unix host.

- ***Integrated circuits***

The implementation of neural networks in the form of integrated circuits has been emerging from a research and development phase and moving into a phase in which neural chips are available in the electronic marketplace. Perhaps it is most accurate to say that two activities are now overlapped: the exploratory research continues while, at the same time, early generations of working chips have been produced.

Implementing a neural network architecture in silicon is like trying to map a three-dimensional space onto a plane. The multiple interconnectivity of neurons in a network demands a three-dimensional geometry to avoid an impossible multiplicity of crisscrossing synapses, rapidly rising with the increasing number of neurons.

Another geometric constrain involves the number of input/output lines that are needed. The numbers of I/O nodes that a single chip can contain is limited by the pinout counts that can be provided for integrated circuits - and the pinout count is limited, in turn, by the sizes of the bonding pads and of the leads from pads to the macroscopic outer world of switches and power supplies.

Therefore, it was necessary to develop physical arrangements that could preserve the three-dimensional requirements of the network, as well as the needed multiplicity of connections to the outer world. Various ways of tackling this problem have been generated.

We can divide the VLSI implementations into three broad categories: digital, analog, and hybrids.

On the basis of available chips the digital VLSI neural networks can be further divided into few categories including:

- *slice architectures* - based on the bit slice concept of conventional digital processors, the neural network slice chips provide building blocks to construct networks of arbitrary size and precision. Examples of the NN slice chips include: the Hitachi WSI, the NeuraLogix NLX-420 Neural Processor Slice, or the Philips Lneuro 1.0.
- *multi-processor chips* - based on the idea of putting many small processors on a chip. Two architectures dominating this design can be distinguished: single instruction with multiple data (SIMD) and systolic arrays. Examples of SIMD chips include Inova N64000 and HNC 100 NAP, and a systolic array system can be built with Siemens MA-16.
- *radial basis functions (RBF)* - as the name points out these chips aim at the simulation of RBFs. Two commercial products now available are: Intel's Ni1000 and IBM's ZISC036.
- *other digital design* - examples of these include Micro Circuit Engineering MT19003 Neural Instruction Set Processor and Hitachi Wafer Scale Integration chips. First is simple RISC processor optimized for implementation of multi-layer nets. In turn

Hitachi produced wafers for implementing Hopfield networks and multi-layer networks using backpropagation learning algorithm.

Among analog hardware neural nets we can find Intel's 80170NW ETANN and Synaptics' Silicon Retina chips, and although analog technology can exploit some physical properties to obtain high speed and densities the analog design can be very difficult to implement and use due to the variations in manufacturing, temperature, etc.

The third group of integrated circuits for NN implementations includes hybrid designs attempting to combine the best of analog and digital techniques. Typically, the external signals are digital to facilitate integration into digital systems, while internally some or all of the processing is analog. Examples of this group include AT&T's ANNA, Bellcore's or Ricoh's RN-200.

The material on neural hardware presented in this section has been compiled from the overview papers (Heemskerk, 1995; Ienne & Kuhn, 1995; Lindsay and Lindblad, 1994) where more detailed information concerning architectures and other technical parameters can be found.

3.4.3. Optical implementations

There are a few good reasons for using optics - either instead of, or along with electronics - to implement neural nets. Foremost among these is the fact that light rays do not suffer, as electrical wires do, from cross-talk. No matter how closely you pack the optical paths in and out of a processing element, or even if they cross, they do not perturb one another.

Thus, optics allows very large fan-ins and fanouts among processors. This is an ideal attribute for neural processing elements in a multiply connected network.

In addition to the lack of cross-talk, optical paths are physically easier to pack together than electronic paths. The electronic paths leading in and out of a processor are wire leads, which need to be bonded to bonding pads in the package that contains a processing chip, and need to come out of the package in the form of pins that can be connected to sockets or traces on a pc board. The constraints on implementing these I/O connections are rapidly becoming the governing factor in how small you can make an electronic chip.

Optically, the I/O leads can be tightly packed bundle of hair-thin light-conducting fibres - or, better yet, mere paths through the air between optical processors. The light must be brought to a photosensitive area, but it need not and cannot be bonded to it. Making and breaking connections, difficult in electronics, is trivial in optics.

Low power consumption is another benefit of optics. The ratio of energy dissipation for an electronic system versus that for an equivalent optical system is very large.

In summary, because neural networks are characterized by large number of processing elements, with a large multiplicity of interconnections, it is a good idea to implement them in a technology that not only allows large fan-ins and fanouts but also keeps power dissipation low, even when all this is happening in a small volume. Optics does it all.

At the same time, electronics is much further along in terms of having been practically implemented and perfected over decades of computer technology. For the time being, then, it remains the principal way in which ANNs are built, despite the theoretical advantages of optics.

Chapter 4

Neural state estimation

4.1. Introduction

Efficient control of a complex water distribution system requires accurate information about its current operating state. Over the last two decades the water industry has been making a significant investment in modern telemetry hardware systems to meet these needs. Unfortunately, due to financial constraints, it is not practical to measure all variables of interest. Therefore, the information supplied by the telemetry system must be supplemented by the less accurate predictions of consumptions at the nodes in the network. These predictions are frequently referred to as pseudomeasurements. Measurements and pseudomeasurements are used to calculate flows and pressures in the distribution network through the use of state estimators which provide a means of reconciling the discrepancies between the mathematical model of the system and the input data.

In general, the state estimation problem can be viewed as the process of constructing and optimization of a suitably chosen cost (also called energy or Lyapunov) function. The choice of the optimization criterion (the type of cost function) characterises different state estimators. According to the criterion used the state estimation procedures can be divided into the following three major groups:

- *least squares (LS) criterion* - where the sum of the squared differences between the measured and estimated values is minimised;
- *least absolute value (LAV) criterion* - where the sum of the absolute differences between the measured and estimated values is minimised; and
- *minimax (also called Chebyshev) criterion* - where the maximum difference between the measured and estimated values is minimised.

The proper choice of the criterion depends on the specific applications and greatly on the type of errors that are likely to occur in the system. Due to this fact, that will be discussed in more detail in the following sections of this chapter, only the first two criterions (LS and LAV) and their variations have been practically used in water systems' state estimation problem.

The state estimators gradually became the key utility for the implementation of monitoring and control of large scale public utility systems such as water, gas or electric power distribution systems.

However, with the increasing complexity of modern water distribution systems there is a need for more efficient state estimators which will form a basis for the implementation of real time control of these systems. Among the potential algorithms and techniques for state estimation neural network based estimators are of great interest because of their potential computational efficiency due to massively parallel nature of ANN. The recent wave of interest in artificial neural network models has led to new theoretical results and advances in VLSI technology (see Chapter 3, Section 3.4) that make it possible to fabricate microelectronic network of high complexity. Much of this interest began when ANNs were devised to solve some optimization problems (Hopfield & Tank, 1985; Tank & Hopfield, 1986). While the full potential of neural networks for mathematical optimization can only be realised with appropriate computing hardware, their performance, in this work, has been assessed through the simulation studies.

4.2. State estimation in water distribution networks

4.2.1. Review of state estimation methods

Before we begin the proper review of state estimation methods for water systems let us take a closer look at the optimization criterions mentioned above and find out why the LS and LAV criterions have been so popular amongst water systems researchers.

From robust statistics (Hampel et al., 1987; Huber, 1981) it is known that the LS, LAV and minimax criterions are optimal for certain error distributions. The standard LS criterion, that has been the most popular one and in use for a long time, is optimal for the Gaussian (normal) distribution only. However, in many applications the assumption that the distribution of measurement errors is Gaussian is unrealistic. For a non-Gaussian error distribution a standard LS estimation may be very poor, especially where measurements contain large errors called "measurement outliers". In order to reduce the influence of the outliers the more robust iteratively re-weighed LS or LAV estimator can be used. The LAV criterion may be also preferable when very little is known about the distribution of errors. It was also shown that the use of LAV criterion produce optimal results for an error distribution having long tails, i.e. the Cauchy distribution. If the error distribution has sharply defined transitions, such as the uniform distribution, the Chebyshev criterion can be the most suitable choice. Because the maximum deviation is minimized in the minimax criterion it is an appropriate one to be used when the data are relatively free from outliers.

When we now look at the most likely errors to occur in water networks we find that there are two main types of errors: these associated with transducer noise, A/D conversions etc. that can be classified as having Gaussian distribution and these associated with topological anomalies, caused by the physical system deviating from the original system as modelled (e.g. due to a new pipe burst), and meter malfunctions, that can be classified as gross errors or outliers. In view of these facts the choice of LS and LAV estimators for water systems state estimation seems to be justified.

Although the choice of appropriate optimality criterion is absolutely crucial the algorithms used to solve these optimization problems are also very important. As a matter of fact the problems with using the LAV and minimax criterions, mainly due to the non-differentiability of the objective function which may and have caused some analytical and numerical problems, have been another reason why the LS criterion is so popular. In water systems different algorithms such as linear programming, non-linear programming, unconstrained optimization have been used to solve the state estimation problem. A review of the currently used techniques that fall under these headings is presented below.

The comparison of the weighted least squares (WLS) problem solved using the augmented matrix approach with the LAV problem solved using linear programming technique can be found in (Bargiela, 1984). Via simulation results Bargiela found that the LS estimator in its augmented matrix formulation is computationally efficient and exhibits very good numerical stability characteristics, especially in the case of structurally ill-conditioned systems. However, the LS approach was found to be intrinsically sensitive to measurement outliers thus requiring further bad data processing followed by re-estimation of the state variables. In contrast, the LAV produced unbiassed estimates automatically rejecting the bad data but the solution time of the linear programming technique dramatically increased with the size of the network preventing its on-line application to large-scale problems. To enhance the efficiency of implementations both methods utilized the sparsity of matrices involved in problem formulations.

The sensitivity of the LS estimators to the outliers has been recognised and reported in many other publications dealing with the utility systems state estimation problem (Dopazo et al., 1970; Falcao et al., 1981; Gabrys & Bargiela, 1995; Handschin et al., 1974; Hartley, 1996; Merill & Schweppe, 1971; Powell et al., 1988; Powell, 1992; Schweppe et al., 1970; Sterling & Bargiela, 1984). In order to diminish the influence of the bad data on the final solution several techniques used with the LS estimators have been developed.

In (Schweppe et al., 1970) two tests were used: observing the weighted sum of squared residuals for detection of bad data and using the list of largest normalised residuals as a guide for the identification of bad data points.

A similar method was proposed in (Dopazo et al., 1970). This technique, based on hypothesis testing theory, uses a Student's t-test instead of normalised residuals for identification of bad data points.

Another approach to improvement of the LS estimates in presence of the bad data is the use of methods penalizing the largest residuals so that the potential bad data have a reduced influence on the final estimates. In these methods of state estimation the non-quadratic cost functions, which approximate to a standard LS criteria when all the data are good, are often used (Falcao et al., 1981; Handschin et al., 1974; Merill & Schweppe, 1971). Another set of examples of iteratively reweighted LS estimators, based on detecting the largest residuals, can be found in (Hartley, 1996; Powell et al., 1988; Powell, 1992).

An alternative formulation to the LS criterion, that can be classified as another case of a non-quadratic criterion and is known as the weighted least absolute method (WLAV), has been proposed by a number of authors for utility systems applications (Bargiela, 1984; Falcao et al., 1981; Gabrys & Bargiela, 1995; Hartley, 1996; Kotiuga & Vidyasagar, 1982; Sterling & Bargiela, 1984). There are three principal numerical algorithms to solve the WLAV optimization problem found in the literature: linear programming (Bargiela, 1984; Barrodale & Zala, 1986; Bazaraa & Jarvis, 1977; Sterling & Bargiela, 1984), non-linear programming (Bazaraa & Shetty, 1979; Beck et al., 1983) and approximate methods (Bargiela, 1995; Christensen & Soliman, 1989; Christensen & Soliman, 1990; Cichocki & Bargiela, 1997; Cichocki & Unbehauen, 1992a; Cichocki & Unbehauen, 1992b; Gabrys & Bargiela, 1995; Hartley, 1996). The neural methods, using LAV criterion, presented in the following sections of this chapter fall into the last category.

4.2.2. Formulation of the state estimation problem

While presenting the network equation (EQ 2-20) and discussing the methods of setting up this set of equations it was assumed that all needed measurements represented by the vector **z** are accurate and known. However, in real situation (as it was explained in Chapter 2, Section 2.2.3) the measurements are not fully accurate due to the finite meter accuracy, noise, gross errors or the necessity of using the inaccurate predictions of consumptions. Therefore, the network equation (EQ 2-20) should be written in the following form:

$$\mathbf{z}=g(\mathbf{x})+\omega \qquad \text{(EQ 4-1)}$$

where ω is the unknown vector, that accounts for measurement noise, model errors and disturbances; **z** is the vector of m measurements contaminated by errors and noise; g() is the nonlinear function (also called network function) describing the system; **x** is the state vector consisting of n-f nodal pressures and f fixed-head node inflows; and $m \geq n$.

Because the measurements, to be used in the calculation of the system state, are contaminated there is a need to use all available information (all available measurements) in the estimation process. These measurements fall into four categories:

- nodal pressure or head measurements;
- fixed-head node inflows;
- pipe flows; and
- consumer demands.

Each of these measurements (pseudomeasurements) is associated with one type of equation constituting to the set of equations (EQ 4-1) and therefore we can distinguish the following four types of equations:

1) For nodal pressure (head) measurements if the *i*-th measurement, z_i, corresponds to the *j*-th pressure (head) variable, x_j, the network function $g_i(\boldsymbol{x})$ is given by

$$x_j = z_i \tag{EQ 4-2}$$

2) For fixed-head node inflows if the *i*-th measurement, z_i, corresponds to the *j*-th fixed-head node inflow variable, x_j, the network function $g_i(\boldsymbol{x})$ is given by

$$x_j = z_i \tag{EQ 4-3}$$

3) For the pipe flows if the *i*-th measurement, z_i, is a flow measurement, the network function $g_i(\boldsymbol{x})$ is given by

$$q_{jk}(x_j, x_k) = z_i \tag{EQ 4-4}$$

where q_{jk} is the hydraulic pressure-flow relationship for the element (e.g. pipe, valve, pump as given in Chapter 2, Section 2.1.3 and Section 2.1.4) placed between node *j* and node *k*.

4) For consumer demands if the *i*-th element in **z**, z_i, is a consumer demand prediction or measurement, the equations represent the mass-balances at nodes as given by (EQ 2-17) and can be written as

$$\sum_{j \in \Omega_k} q_{kj} + x_l = z_i \tag{EQ 4-5}$$

where *k* is the load node, Ω_k is a set of nodes connected to node *k*, x_l is the inflow variable for node *k*.

Having defined the overdetermined $(m \geq n)$ set of simultaneous equations (EQ 4-1) taking into account the unknown vector of measurements errors, **r**, we can now formulate the estimation problem associated with (EQ 4-1).

The state estimation can be expressed as a problem of minimization of discrepancies between the actual measurements and the values calculated from the mathematical model subject to a suitable optimality criterion.

Using the least squares criterion the state estimation problem can be expressed as:

$$\min_{\hat{x}} E_2(\hat{x}) = \frac{1}{2}(z - g(\hat{x}))^T W (z - g(\hat{x})) \qquad \text{(EQ 4-6)}$$

Similarly, using the least absolute values criterion the state estimation is expressed as:

$$\min_{\hat{x}} E_1(\hat{x}) = w^T |z - g(\hat{x})| \qquad \text{(EQ 4-7)}$$

where:

$\hat{x} \in R^n$ - an estimate of the state vector **x**

$w \in R^m$ - measurement weight vector

$W = diag[w_1, w_2, \ldots, w_m]$ - measurement weight matrix

The proposed solution to the state estimation problem (EQ 4-6) or (EQ 4-7) is based on the Newton-Raphson method described in Chapter 2, Section 2.2.2.

After the linearisation of (EQ 4-1)we obtain the following set of equations:

$$J(\hat{x}^{(k)})\Delta x = z - g(\hat{x}^{(k)}) + r \qquad \text{(EQ 4-8)}$$

where:

$J(\hat{x}^{(k)}) \in R^{mxn}$ - Jacobian matrix evaluated at $\hat{x}^{(k)}$

r - vector of residuals (an estimate of ω)

k=0,1,... - step of the estimation process

Equations (EQ 4-6) and (EQ 4-7) can be therefore expressed as

$$\min_{\Delta x} E_2(\Delta x) = \frac{1}{2}(\Delta z - J(\hat{x}^{(k)})\Delta x)^T W (\Delta z - J(\hat{x}^{(k)})\Delta x) \qquad \text{(EQ 4-9)}$$

and

$$\min_{\Delta x} E_1(\Delta x) = w^T |\Delta z - J(\hat{x}^{(k)})\Delta x| \qquad \text{(EQ 4-10)}$$

As it was explained in Chapter 2, because the measurement equations (EQ 4-1) are nonlinear, the solution to (EQ 4-6) or (EQ 4-7) is an iterative process with the consecutive state estimates calculated by under-relaxation of the linear solution

$$\hat{x}^{(k+1)} = \hat{x}^{(k)} + \gamma \Delta x^{(k)}, \text{ k=0,1,...} \qquad \text{(EQ 4-11)}$$

If all elements of Δx in k-th iteration are lower or equal to a predefined convergence accuracy, the iteration procedure stops. Otherwise, a new correction vector is calculated using equation (EQ 4-8) with $\hat{x}^{(k+1)}$ instead of $\hat{x}^{(k)}$ and minimizing a criterion function.

As we can see from the above formulation the problem of state estimation in water distribution systems has been defined as the process of iterative solving of the overdetermined system of linear equations (EQ 4-8) with respect to some optimality criteria. In this work this overdetermined system of linear equations is solved via the feedback neural networks that are discussed in the following sections.

4.3. ANNs for solving systems of linear equations

4.3.1. General method description

For the sake of readability and clarity of further derivations let us rewrite the linearised network equation (EQ 4-8) in the following form

$$\boldsymbol{A}\Delta \boldsymbol{x} = \boldsymbol{b} + \boldsymbol{r} \qquad \text{(EQ 4-12)}$$

where $\boldsymbol{A} = \boldsymbol{J}(\hat{\boldsymbol{x}}^{(k)})$ and $\boldsymbol{b} = \boldsymbol{z} - \boldsymbol{g}(\hat{\boldsymbol{x}}^{(k)})$.

Cichocki and Unbehauen in (Cichocki & Unbehauen, 1992a) proposed a three step scheme of obtaining a neural network for solving systems of linear equations. The key step is to construct an appropriate computational energy function $\boldsymbol{E}$ so that the lowest energy state will correspond to the desired optimal solution. Next by employing a general gradient approach for minimization of a function, the minimization problem is transformed into a set of ordinary differential or difference equations. And finally, on the basis of these differential or difference equations ANN architectures, with appropriate synaptic weights and nonlinear activation functions, are designed. This scheme will be used for constructing practically all neural networks presented in the following sections.

The minimization problems described by (EQ 4-9) and (EQ 4-10) can be generalised as follows:

$$\min_{\Delta x} E(\Delta x) = \sum_{i=1}^{m} \sigma_i [r_i(\Delta x)] \qquad \text{(EQ 4-13)}$$

where:

$\boldsymbol{E}$ is a general cost (energy) function

$r_i(\Delta\boldsymbol{x}) = \boldsymbol{a}_i^T \Delta x - b_i$ is the i-th element of the vector of residuals

$\sigma_i[r_i]$ represents a suitably chosen convex functions.

In a special case when $\sigma_i(r_i) = w_i r_i^2/2$ we obtain the standard weighted least-squares criterion (EQ 4-9) and for $\sigma_i(r_i) = w_i|r_i|$ we have the weighted least absolute values criterion (EQ 4-10).

The minimization of the energy function described by (EQ 4-13) by standard gradient descent method leads to the following system of nonlinear differential equations written in the matrix form

$$\frac{d\Delta\boldsymbol{x}}{dt} = -\mu(t)\nabla E(\Delta\boldsymbol{x}) = -\mu(t)\boldsymbol{A}^T f(\boldsymbol{r}(\Delta\boldsymbol{x})) \qquad \text{(EQ 4-14)}$$

where $\mu(t) = [\mu_{ij}(t)]$ is nxn positive-definite matrix that is often diagonal $\mu(t) = diag(\mu_1, \mu_2, \ldots, \mu_m)$ and $\nabla E(\Delta\boldsymbol{x})$ is a gradient of the energy function $E(\Delta\boldsymbol{x})$. In general the entries of the matrix $\mu(t)$ depend on the time and the vector $\Delta\boldsymbol{x}$.

The system of equations (EQ 4-14) can be written in the following scalar form

$$\frac{d\Delta x_j}{dt} = -\sum_{p=1}^{n}\mu_{jp}\left(\sum_{i=1}^{m}a_{ij}\left(f_i\left(\sum_{k=1}^{n}a_{ik}\Delta x_k - b_i\right)\right)\right) \qquad \text{(EQ 4-15)}$$

where

$$f_i(r_i(\Delta\boldsymbol{x})) = \frac{\partial\sigma_i(r_i)}{\partial r_i} \qquad \text{(EQ 4-16)}$$

is an activation function dependant on the type of $\sigma_i(r_i)$ and for instance:

- for $\sigma_i(r_i) = w_i r_i^2/2$ we have $f_i(r_i(\Delta\boldsymbol{x})) = w_i r_i(\Delta\boldsymbol{x})$ - linear activation function
- for $\sigma_i(r_i) = w_i|r_i|$ we have $f_i(r_i(\Delta\boldsymbol{x})) = sign[w_i r_i(\Delta\boldsymbol{x})]$ - signum activation function.

The system of differential equations (EQ 4-15) can be implemented directly by an artificial neural network depicted at Figure 4-1.

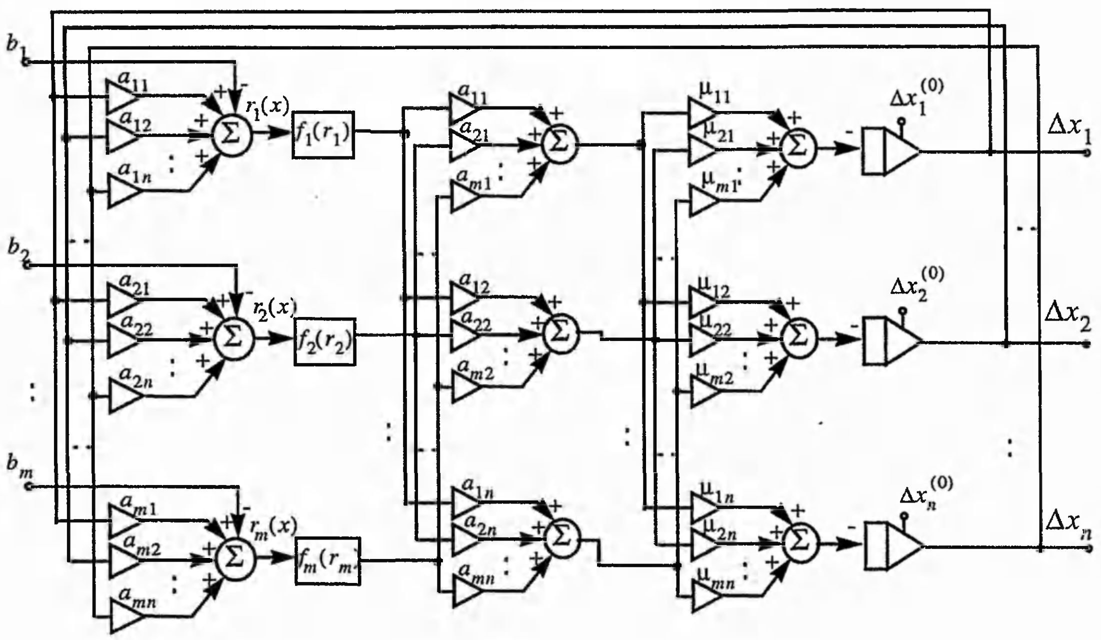


Figure 4-1: Neural network for solving systems of linear equations.

The circuit consists of three layers of artificial neurons and is an example of feedback neural network with connection weights set to represent the system of overdetermined linear equations and appropriate energy function to be minimized rather than adjusted according to some learning algorithm via the process of presenting a sequence of training vectors. These connection weights denoted by a_{ij} and μ_{jp} represent the coefficients of the matrices **A** and μ. The weights can be fixed or time-variable depending on the entries of the matrices **A** and μ. Note that matrix **A** representing the Jacobian matrix and vector $\boldsymbol{b} = \boldsymbol{z} - \boldsymbol{g}(\hat{\boldsymbol{x}}^{(k)})$ change in every step of Newton-Raphson method and therefore appropriate connection weights in ANN have to be set accordingly. Although in most of the networks presented below the matrix μ is diagonal with constant coefficients set at the beginning of simulation it will be shown in Section 4.3.4 that adaptive selection of $\mu_{jp}(t)$ can greatly increase the convergence rate.

However, the specific choice of the coefficients $\mu_{jp}(t)$ must ensure the stability of the differential equations and an appropriate convergence speed to the stationary solution state. It is easy to prove that the system of differential equations (EQ 4-15) is stable (i.e. it has always a stable asymptotic solution) since

$$\frac{dE}{dt} = \sum_{j=1}^{n} \frac{\partial E}{\partial \Delta x_j} \frac{d\Delta x_j}{dt} = (\nabla E(\Delta \boldsymbol{x}))^T \left(\frac{d\Delta \boldsymbol{x}}{dt}\right) \qquad \text{(EQ 4-17)}$$

$$= -(\nabla E(\Delta \boldsymbol{x}))^T \mu(t) \nabla E(\Delta \boldsymbol{x}) \leq 0$$

when the matrix $\mu(t)$ is positive-definite for all values of $\Delta \mathbf{x}$ and t.

The neural network from Figure 4-1 has been implemented in MATLAB and SIMULINK in more compact form using vector and matrix notations and is depicted in Figure 4-2.

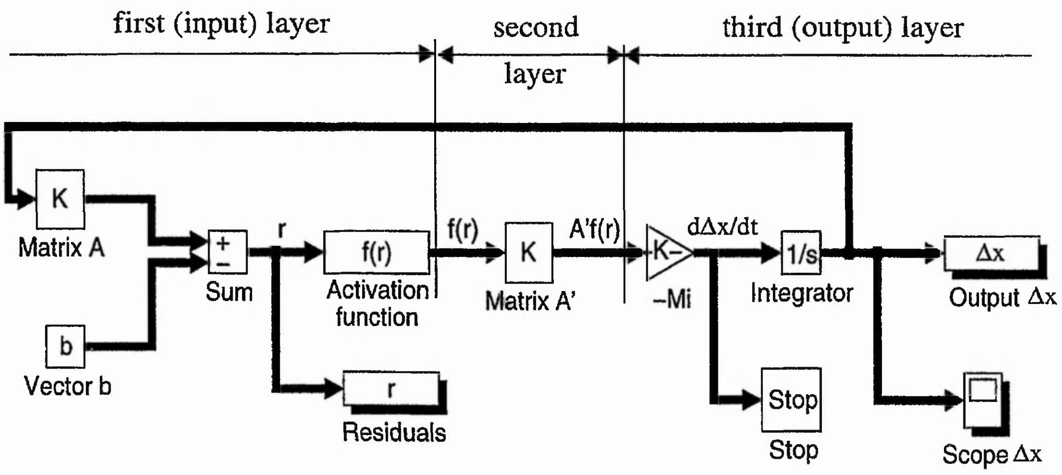


Figure 4-2: ANN for solving a system of linear equations (EQ 4-12) based on the system of differential equations (EQ 4-14) with optional activation functions (implementation in MATLAB and SIMULINK)

On the basis of the above implementation let us explain in a few words the functions of each of three layers.

The first layer senses actual variables $\Delta x_j(t)$ and computes the actual errors $e_i(\Delta \boldsymbol{x})$

$$e(\Delta \boldsymbol{x}) = f\left(\sum_{k=1}^{n} a_{ik}\Delta x_k - b_i\right), \quad i = 1, 2, \ldots, m \qquad \text{(EQ 4-18)}$$

The second layer estimate the gradient components $\dfrac{\partial \varepsilon(\Delta \boldsymbol{x})}{\partial \Delta x_p}$

$$\frac{\partial \varepsilon(\Delta \boldsymbol{x})}{\partial \Delta x_p} = \sum_{i=1}^{m} a_{ip} e_i(\Delta \boldsymbol{x}), \quad p = 1, 2, \ldots, n \qquad \text{(EQ 4-19)}$$

The third layer comprises "response elements" and constitutes the proper learning system (when μ_j changes during the simulation).

$$\frac{d\Delta x_j}{dt} = -\mu_j\left(\frac{\partial\varepsilon(\Delta x)}{\partial\Delta x_p}\right) \quad , \quad \Delta x_j(0) = \Delta x_j^{(0)} \quad , \quad j = 1, 2, \ldots, n \qquad \text{(EQ 4-20)}$$

So far it has been implicitly assumed (via the fact that the differential equations have been used) that the implementation of the ANN is continuous-time but it can be shown that by applying the standard first-order discretization technique (Euler integration rule) to the system of differential equations (EQ 4-15) (with matrix μ diagonal), this system can be converted to the difference equations

$$\Delta x_j^{(k+1)} = \Delta x_j^{(k)} - \mu_j^{(k)}\Delta t\sum_{i=1}^{m} a_{ij}\left(f_i\left(\sum_{p=1}^{n} a_{ip}\Delta x_p^{(k)} - b_i\right)\right) \qquad \text{(EQ 4-21)}$$

with $\Delta x_j(0) = \Delta x_j^{(0)}$, $j = 1, 2, \ldots, n$.

It should be noted that for discrete-time algorithms the controlling parameter $\mu_j^{(k)}\Delta t$ must be bounded in a small range $(0 < \mu_j^{(k)}\Delta t \le \mu_{max}\Delta t)$ to assure stability of the algorithms. However, a small $\mu_j^{(k)}\Delta t$ means that the convergence to a solution is slow, while a large $\mu_j^{(k)}\Delta t$ means that oscillations may occur and stability may be lost. Such an effect occurred for the ANN's C implementations using the systems of difference equations. On the other hand, for continuous-time algorithms (simulated using more sophisticated integration methods i.e. Gear method provided with MATLAB software package) the parameters $\mu_j > 0$ can be set to a theoretically arbitrarily large value without affecting the stability of the system.

4.3.2. Iteratively reweighted least squares criterion

In the presence of outliers or wild noise in the observation vector **b** (and effectively in vector of measurements **z**), the standard least squares criterion can provide very poor, biased estimates. In order to reduce the influence of the outliers we will employ the non-quadratic criterions, which approximate to a standard LS criteria when all the data are good. This approach is also known as the iteratively reweighted least squares criterion

(Bargiela, 1995; Cichocki & Bargiela, 1997; Cichocki & Unbehauen, 1992a; Gabrys & Bargiela, 1995;Hampel et al., 1987;Huber, 1981; Merill & Schweppe, 1971).

Three types of non-quadratic functions, to be put in place of $\sigma_i[r_i]$ in (EQ 4-13),have been used: logistic function, Huber's function and Talvar's function. They were chosen because of their robust behaviour in the presence of outliers. Their mathematical forms are as follows:

a) Logistic function

$$\sigma_i(r_i) = \sigma_L(r_i) = \frac{\beta}{\alpha} ln(\cosh(\alpha r_i)) \qquad \text{(EQ 4-22)}$$

b) Talvar's function

$$\sigma_i(r_i) = \sigma_T(r_i) = \begin{cases} r_i^2/2 & \text{for } |r_i| \le \beta \\ \beta^2/2 & \text{for } |r_i| > \beta \end{cases} \qquad \text{(EQ 4-23)}$$

c) Huber's function

$$\sigma_i(r_i) = \sigma_H(r_i) = \begin{cases} r_i^2/2 & \text{for } |r_i| \le \beta \\ \beta|r_i| - (\beta^2/2) & \text{for } |r_i| > \beta \end{cases} \qquad \text{(EQ 4-24)}$$

Now using equation (EQ 4-16) we can calculate appropriate activation functions to be used in the first layer of the ANN. These activation functions are represented by the following mathematical formulas:

a) Sigmoid activation function

$$f_i(r_i) = \frac{\partial\sigma_L(r_i)}{\partial r_i} = \frac{\partial\left(\frac{\beta}{\alpha} ln(\cosh(\alpha r_i))\right)}{\partial r_i} = \beta\tanh(\alpha r_i) \qquad \text{(EQ 4-25)}$$

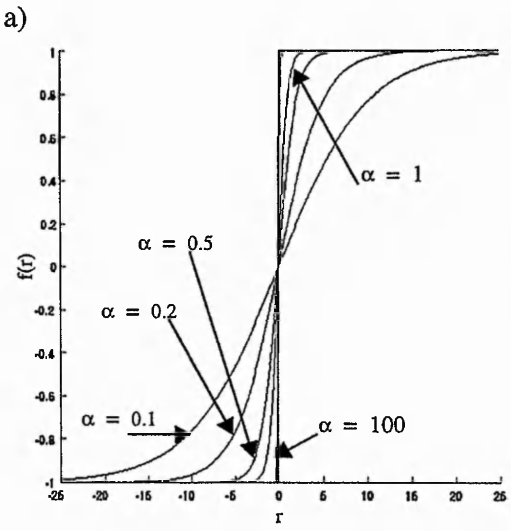
b) Talvar's activation function

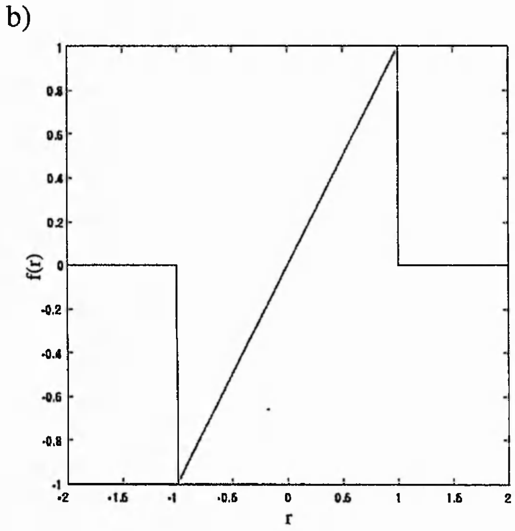
$$f_i(r_i) = \frac{\partial\sigma_T(r_i)}{\partial r_i} = \begin{cases} r_i(x) & \text{for } |r_i| \le \beta \\ 0 & \text{otherwise} \end{cases} \qquad \text{(EQ 4-26)}$$

c) Huber's activation function

$$f_i(r_i) = \frac{\partial\sigma_H(r_i)}{\partial r_i} = \begin{cases} -\beta & \text{for } r_i < -\beta \\ r_i(x) & \text{for } |r_i| \le \beta \\ \beta & \text{for } r_i > \beta \end{cases} \qquad \text{(EQ 4-27)}$$

The graphical representations of these functions are shown in Figure 4-3.





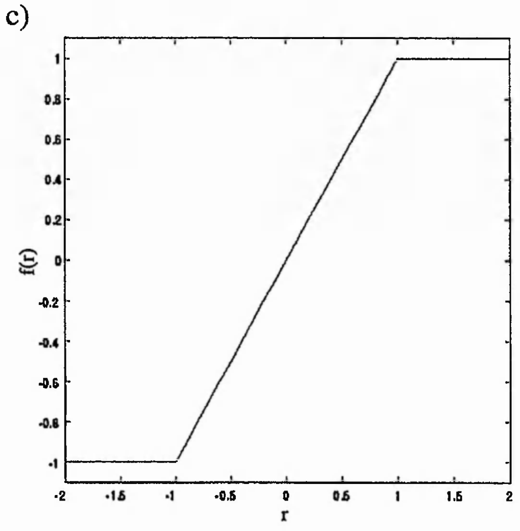


Figure 4-3: Activation functions with $\beta = 1$
a)Sigmoid function
b)Talvar's function
c)Huber's function

The use of nonlinear activation functions in the first layer of "neurons" is essential for overdetermined linear systems of equations since it enables us to obtain more robust solutions, which are less sensitive to outliers in comparison to the standard linear implementation. The sigmoid and Huber's activation functions prevent large absolute values of residuals from being greater than prescribed cut-off parameter $\beta > 0$. The Talvar's activation function provides that all equations (from the system described by (EQ 4-12)) with large absolute values of residuals above the cut-off parameter β are neglected and they have no influence on the final solution.

Another useful property of the sigmoid activation function is that if the cut-off parameter β is chosen large with $\alpha = 1/\beta$, then we will come close to the standard least squares criterion, but taking $\beta = 1$ and α large gives something similar to the LAV criterion, where the sum of the absolute values of the residuals is minimized since then the activation function given by (EQ 4-25) closely approximates the signum function.

4.3.3. Augmented lagrangian with regularization

The above proposed schemes and structures are suitable for well-conditioned problems. However, for ill-conditioned problems such schemes may be prohibitively slow and they may even find a solution with large error. For the purpose of improving the convergence properties and the accuracy of the desired networks we can use the following energy function (augmented Lagrangian function) for the problem (EQ 4-12):

$$E(\Delta x) = \frac{1}{2}r^T(\Delta x)Kr(\Delta x) + \lambda^T r(\Delta x) - \frac{\varphi}{2}\lambda^T\lambda \qquad \text{(EQ 4-28)}$$

$$E(\Delta x) = \frac{1}{2}\sum_{i=1}^{m} k_i r_i^2(\Delta x) + \sum_{i=1}^{m}\left(\lambda_i r_i(\Delta x) - \frac{\varphi}{2}\lambda_i^2\right) \qquad \text{(EQ 4-29)}$$

where:

r(Δx)=AΔx-b - the residuals

λ - the Lagrange multipliers

$K = diag(k_1, k_2, \ldots, k_m)$ - the weighting penalty coefficients $(0 \le k_i \le 1)$

$\varphi \ge 0$ - the regularization parameter

The augmented Lagrangian is obtained from the ordinary Lagrangian by adding penalty terms (Bertsekas, 1982). Since an augmented Lagrangian can be ill-conditioned a regularization term with coefficient φ is introduced to eliminate the instabilities associated with the penalty terms. The problem of the minimization of the above defined energy function can be expressed as:

a) a set of differential equations

$$\frac{d\Delta x_j}{dt} = -\mu_j \sum_{i=1}^{m} (k_i r_i(\Delta x) + \lambda_i)\, a_{ij} \qquad \text{(EQ 4-30)}$$

$$\frac{d\lambda_i}{dt} = \rho_i (r_i(\Delta x) - \varphi\lambda_i) \qquad \text{(EQ 4-31)}$$

with i=1,2,...,n; j=1,2,...,m; $\mu_j > 0$; $\rho_i \geq 0$

b) a set of difference equations

$$\Delta x_j^{(k+1)} = \Delta x_j^{(k)} - \mu_j \Delta t \sum_{i=1}^{m} (k_i^{(k)} r_i^{(k)} (\Delta \boldsymbol{x}) + \lambda_i^{(k)}) a_{ij} \qquad \text{(EQ 4-32)}$$

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + \rho_i \Delta t (r_i^{(k)} (\Delta \boldsymbol{x}) - \varphi \lambda_i^{(k)}) \qquad \text{(EQ 4-33)}$$

or presented in the compact form:

$$\frac{d\Delta \boldsymbol{x}}{dt} = -\mu A^T \boldsymbol{y} \qquad \text{(EQ 4-34)}$$

$$\frac{d\lambda}{dt} = \rho\, ((A\Delta \boldsymbol{x} - \boldsymbol{b}) - \varphi\lambda) \qquad \text{(EQ 4-35)}$$

where: $\boldsymbol{y} = [\lambda_1 + k_1 r_1, \lambda_2 + k_2 r_2, \ldots, \lambda_m + k_m r_m]^T$

$\Delta \boldsymbol{x} = [\Delta x_1, \Delta x_2, \ldots, \Delta x_n]^T$

$\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_m]^T$

$\mu = diag(\mu_1, \mu_2, \ldots, \mu_n)$

$\rho = diag(\rho_1, \rho_2, \ldots, \rho_m)$

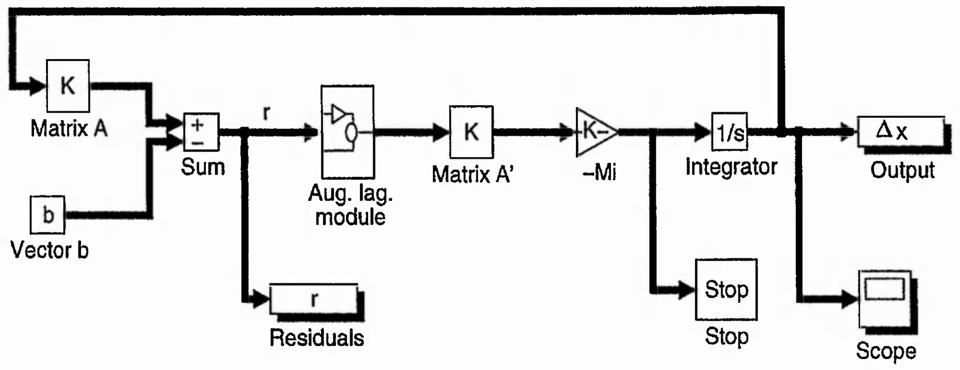
$\varphi \geq 0$

The sets of differential equations (EQ 4-34) and (EQ 4-35) has been implemented in the form of ANN depicted in Figure 4-4.

We can see that the structure of this network is very similar to the one shown in Figure 4-2. The only difference to be noticed is the presence of subnetwork, representing the Lagrangian elements from the energy function, in place of activation functions. It has to be said, however, that various activation functions can be used within this structure in the way they were used in the ANN from Figure 4-2.

4.3.4. Artificial neural network with time processing independent of the size of the problem

In some real time applications it is required to assure that the specified energy function E(Δx) reaches the minimum at a prescribed finite period of time, say t_r, or that E(Δx) become close to the minimum with a specified error $\delta > 0$ (where δ is an arbitrarily chosen positive very small number). Such a problem can be solved by making the a)



b)

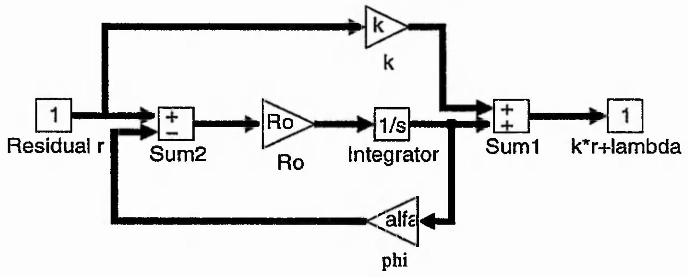


Figure 4-4: a) ANN architecture for solving the system of the linear equations (EQ 4-12) based on the system of the differential equations (EQ 4-34), (EQ 4-35) (implementation in MATLAB and SIMULINK); b) Aug. Lag. module-subsystem of a)

coefficients $\mu_j(t)$ adaptive during the minimization process, under the assumption that the initial value $E(\Delta x^{(0)})$ and the minimum (final) value of E(**Δx***) of the energy function E(**Δx**(t)) are known or can be estimated (Cichocki & Unbehauen, 1992a).

Let us consider the problem (EQ 4-12), which can be mapped to the system of differential equations

$$\frac{d\Delta x}{dt} = -\mu_0(t) A^T (A\Delta x - b) \qquad \text{(EQ 4-36)}$$

where the adaptive parameter $\mu_0(t)$ can be defined as

$$\mu_0(t) = \begin{cases} \mu_{0max} & \text{for } E(\Delta x(t)) = 0 \\ \dfrac{\mu}{r^T A A^T r} & \text{otherwise} \end{cases} \qquad \text{(EQ 4-37)}$$

Note that for this problem $E(\Delta x^{(0)}) = \frac{1}{2}\sum_{i=1}^{n} b_i^2$ and E(**Δx***)=0. The time derivative of the energy function associated with the system of differential equations (EQ 4-36) is:

$$\frac{dE}{dt} = \sum_{i=1}^{n} \frac{\partial E}{\partial \Delta x_j}\left(\frac{d\Delta x_j}{dt}\right) = [\nabla E(\Delta x))]^T\left(\frac{d\Delta \mathbf{x}}{dt}\right) \qquad \text{(EQ 4-38)}$$

$$= -\mu_0(t)\, r^T A A^T r = -\mu < 0 \qquad \text{for} \quad \frac{d\Delta \mathbf{x}}{dt} \neq 0$$

and $\frac{dE}{dt} = 0$ only for $\frac{d\Delta \mathbf{x}}{dt} = 0$.

It follows that the energy function decreases in time linearly during the minimization process as:

$$E(\Delta x) = E(\Delta x^{(0)}) - \mu t \qquad \text{(EQ 4-39)}$$

and reaches the value $\delta \approx 0$ after the time

$$t_r = \frac{E(\Delta x^{(0)}) - \delta}{\mu} = \frac{\frac{1}{2}\sum_{1=1}^{n} b_i^2 - \delta}{\mu} \qquad \text{(EQ 4-40)}$$

By choosing $\mu = E(\Delta x^{(0)})/t_{max}$ we find that the system of equations (EQ 4-36) reaches the stationary point in the prescribed time $t_r \equiv t_{max}$ independent of the size of the problem. The suitable network has been implemented and is depicted in Figure 4-5.

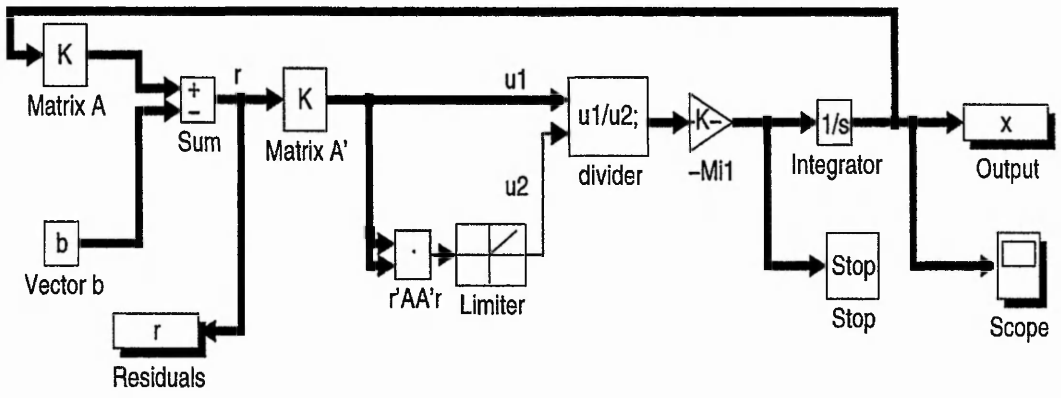


Figure 4-5: ANN providing a linearly decreasing in time energy function giving a predetermined speed of convergence (implementation in MATLAB and SIMULINK)

4.3.5. Neural network model by using inhibition principle

Inhibition plays an important self-regulatory control function in many artificial neural networks mainly for various decision making and selection tasks. In general, the function of the inhibition subnetwork is to suppress some signals (e.g. the strongest signals) while allowing the other signals to be transmitted for further processing.

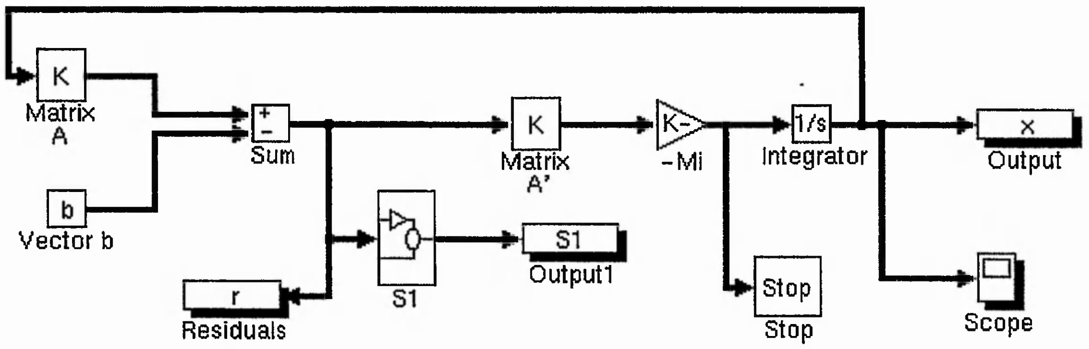
The inhibition principle has been also heavily exploited in the water systems state estimation procedures (Hartley, 1996; Powell et al., 1988). However, the algorithms used by Hartley and Powell are based on iteratively re-weighting the equations with large residuals. New weights are inversely proportional to the current values of residuals. This scheme (i.e. find LS solution, identify large residuals, re-weight accordingly to the residual values, find LS solution etc.) tends to be slowly convergent. By convergence in this case we understand the instance when all equations with large residuals have been sufficiently suppressed so that they do not influence the final solution.

In this work the application of this method has been expanded to neural state estimation (Gabrys & Bargiela, 1995) and carried one step further. The basic idea is that after finding the standard LS estimates $\hat{x}$, instead of using slow process of re-weighting we can compute all residuals $r_i(\hat{x})$ (i=1,2,...,m) and select from them the l largest residuals that correspond to the largest measurement errors in the observed vector **b** and suppress them completely the moment they have been identified. The number l depends on the rejecting criterion and for instance when we want to obtain the LAV estimates the m-n equations corresponding to the m-n largest residuals have to be suppressed (rejected). On the other hand one may use the rejecting criterion stating that all equations corresponding to the residuals $|r_i(\hat{x}) - \bar{r}(\hat{x})| > 3\sigma$ (where σ is a standard deviation, and $\bar{r}(\hat{x})$ is the average value of residuals calculate for $\hat{x}$) should be rejected.

Example of LAV estimation using an ANN incorporating the inhibition principle.

The computation occurs in two phases. Suitable ANNs for carrying the first and second phase of calculations are depicted in Figure 4-6 and Figure 4-7 respectively. In the first phase all values of $S_i = 1$ (i=1,2,...,m) (closed switches) and the network computes the least-squares solution $\hat{x}$ of the problem (EQ 4-12) and the associated residuals $r_i(\hat{x})$ (i=1,2,...,m). At the same time the suitable subnetwork (S1 in the Figure 4-6b) selects continuously in time the *m-n* largest (in absolute value) residuals $r_i(\hat{x})$ from the set of *m* residuals $r_i(\hat{x})$. In the second phase of computation the *m-n* (found in the first phase) largest residuals are inhibited by setting corresponding S_i to zero (opening corresponding switches), while allowing smallest residuals to be further processed in the network. Thus in the second phase of computation only n equations are selected for which the residuals are minimized to zero while the rest of the equations is discarded.

a)



b)

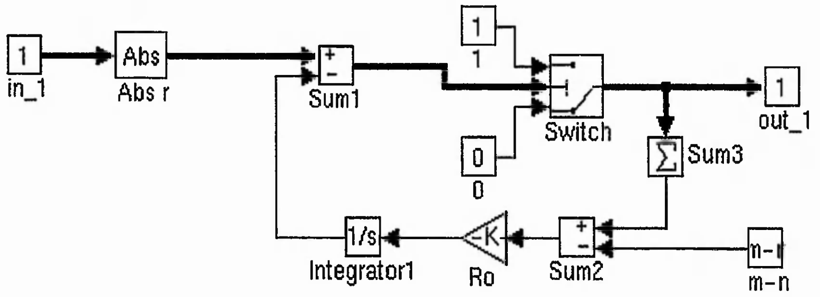


Figure 4-6: a) ANN to compute the first phase of solving least-absolute norm problem using the inhibit principle, b) S1 subnetwork selecting the (m-n) largest (in absolute value) residuals

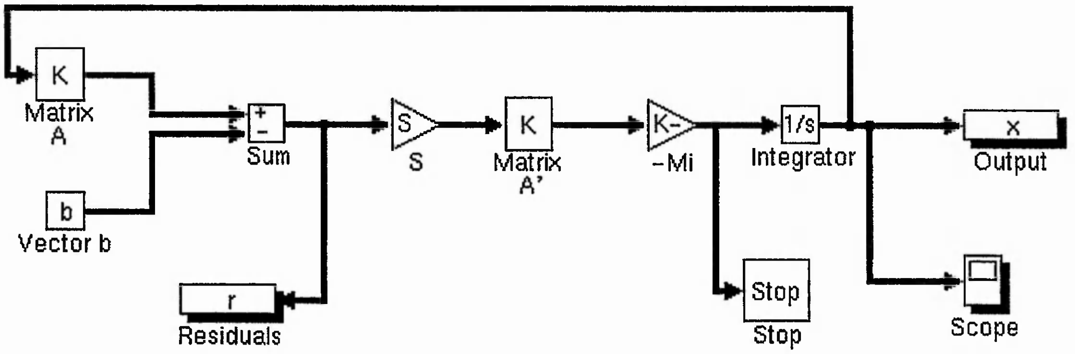


Figure 4-7: ANN to compute the second phase of solving least-absolute norm problem using the inhibit principle (where vector **S=S1** and **S1** found in the first phase)

4.4. Computational results

The neural networks presented in this chapter have been first tested for many simple examples. The performance statistics like times of simulation, estimated times of convergence assuming the hardware implementation, etc. and all values of parameters used while simulating the ANNs for several simple examples are included in Appendix A.

The performance of the proposed methods for water system state estimation was tested on the realistic 34-node network (42 state variables) depicted at the Figure 4-8 (Gabrys & Bargiela, 1995).

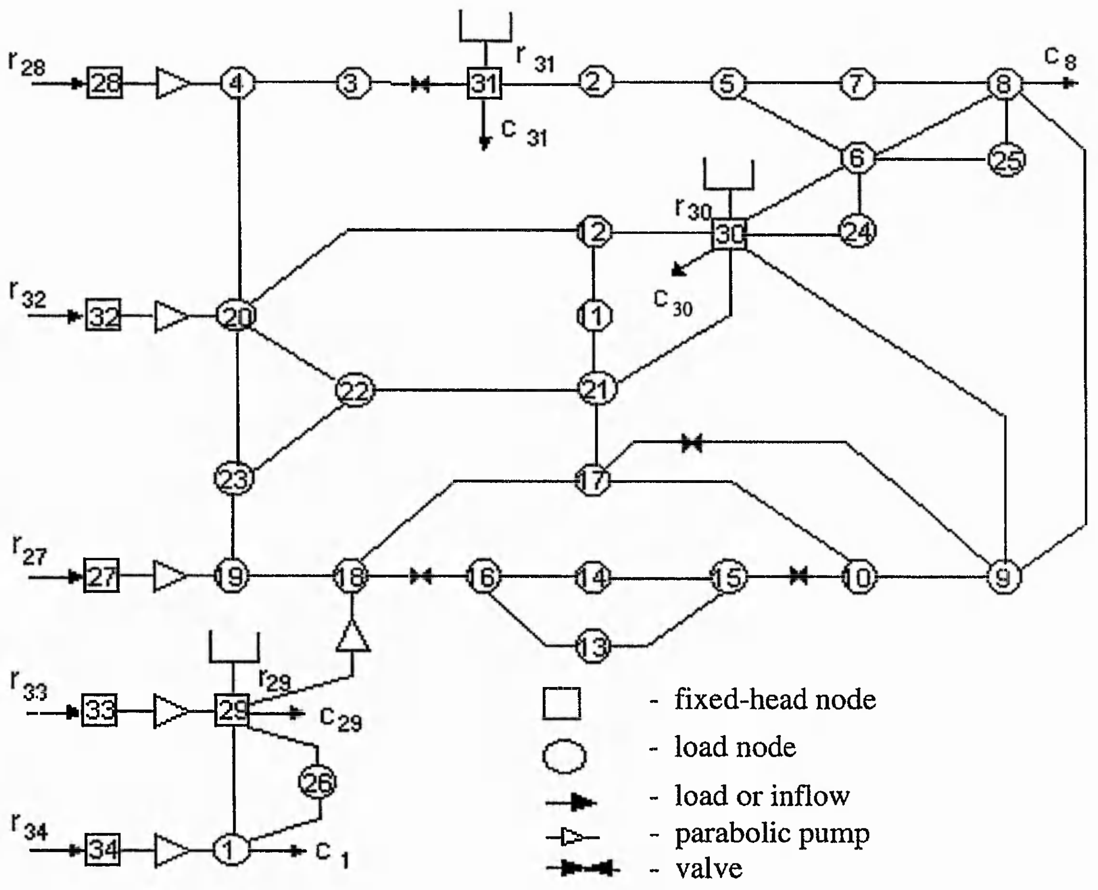


Figure 4-8: 34 - node water network

A complete definition of network parameters are contained in (Sterling & Bargiela, 1984). In order to achieve sufficient measurements redundancy (defined as a ratio of the number of measurements and pseudomeasurements to the number of state variables), the set of the mass balance equations was augmented by a number of several flow and pressure measurements.

Two sets of measurements were processed having redundancy ratios 1.74 and 1.4.

Since the unbiased estimates are the most important feature of state estimators the testing examples were devised to include some 'bad data'. The effect of 'bad data' measurements was simulated by introduction of systematic gross errors in head and flow measurements.

The specification of these errors are given in the following examples.

Example 1:

Introduced gross errors:

head in node 22 = 42.59 [m Aq] (exact value = 46.59 [m Aq])

load in node 8 = -0.025 $[m^3/s]$ (exact value = -0.075 $[m^3/s]$)

Example 2:

Introduced gross errors:

head in node 22 = 42.59 [m Aq] (exact value = 46.59 [m Aq])

head in node 29 = 35.70 [m Aq] (exact value = 31.70 [m Aq])

head in node 30 = 48.58 [m Aq] (exact value = 43.58 [m Aq])

load in node 8 = -0.025 $[m^3/s]$ (exact value = -0.075 $[m^3/s]$)

Table 4-1 and Table 4-2 show the state estimates calculated for redundancy ratios 1.74 and 1.4 respectively. The corresponding state estimation errors are shown in Table 4-3 and Table 4-2. The ANN used to obtain these results is the one from Figure 4-2 with sigmoid activation function. The LS and LAV estimates were obtained by changing the α and β parameters of sigmoid activation function. The state vector shown in column 2 of Table 4-1 and Table 4-2 is the state vector obtained by exact network simulation and is referred to as a vector of reference values. State estimates of LS and LAV methods calculated for data not including gross errors are presented in column 3 and 4 of Table 4-1 and Table 4-2 respectively. State estimates calculated for data including gross errors are presented in columns 5, 6, 7 and 8.

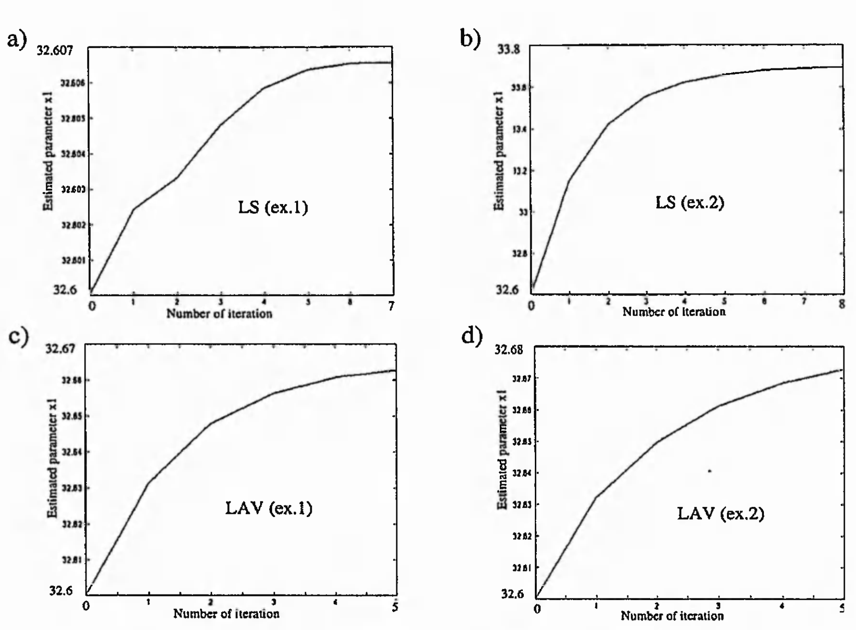


Figure 4-9: Estimates of the head in the node 1 (**x1**) using: a) LS estimator for example 1, b) LS estimator for example 2, c) LAV estimator for example 1, d) LAV estimator for example 2

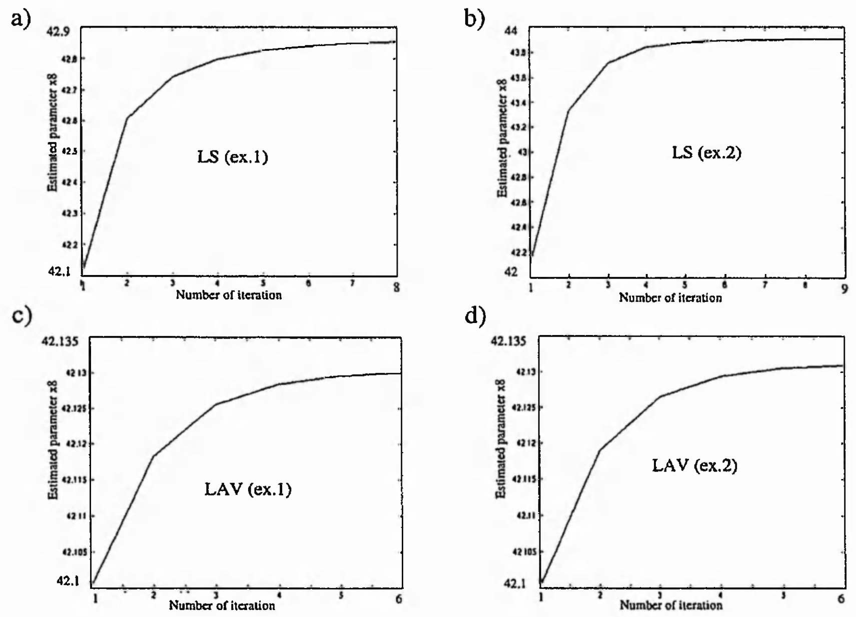


Figure 4-10: Estimates of the head in the node 8 (**x8**) using: a) LS estimator for example 1, b) LS estimator for example 2, c) LAV estimator for example 1, d) LAV estimator for example 2

LAV (Least Absolute Values) method.

Table 4-1 (columns 6 and 8) shows the results of examples in which higher measurement redundancy has been used. Table 4-2 (columns 6 and 8) shows the corresponding results for lower measurement redundancy. Comparison of these results indicates that a smaller number of equations (measurements) was sufficient for accurate estimation with a specific pattern of measurements considered. However, an increased number of measurements contributes mainly to an improved reliability of the estimation and ensures the rejection of a larger spectrum of errors. In conclusion, the LAV problem solution is median solution and passes through at least n (n - number of state variables) of the m data points (measurements). The feature of producing interpolatory fits that closely approximate most of the data while neglecting gross errors is an extremely useful property of the LAV criterion. Provided sufficient basic measurements are available, the LAV estimator can then act as filter for incoming data.

LS (Least Squares) method.

The ordinary LS problem solution is the mean solution since it tries to satisfy all the equations in the set, but usually this solution will not solve exactly any of these equations. The results shown in columns 5 and 7 of Table 4-1 and Table 4-2 are a very good example of the influence of gross errors on a standard LS state estimation. A measurement containing gross error has the biggest effect on estimation of the state variables in the node where the error occurred and nodes of the closest vicinity. An increased number of measurements, in this case, helps to reduce an influence of gross errors (averaging process) but the main cause of using the state estimation methods is insufficient number of measurements.

Figure 4-9 and Figure 4-10 illustrate the convergence of the estimation process (variables x1 and x8 respectively) for the LS and LAV estimators.

All simulations have been carried out on Sun Workstation using SIMULINK (Dynamic System Simulation Software) and MATLAB (High-Performance Numeric Computation and Visualization Software) programs. The corresponding lapsed times for these simulations were of order of hundreds of seconds. Various integration algorithms have been used for many different values of the parameters γ, α, β and μ. The results presented in Table 4-1 and Table 4-2 have been obtained for γ=0.6, $\mu = 1e^{6}$, Gear integration algorithm. Parameters α and β have been set as follows: α=0.1, β=10 for LS and α=500, β=1 for LAV estimators.

State variable	Exact value	LS	LAV	LS (Ex.1)	LAV (Ex.1)	LS (Ex.2)	LAV (Ex.2)
1	32.638	32.658	32.661	32.662	32.661	33.630	32.666
2	43.749	43.756	43.759	43.460	43.757	43.507	43.757
3	46.041	46.058	46.042	45.547	46.042	45.700	46.042
4	46.618	46.645	46.622	46.133	46.622	46.342	46.622
5	43.265	43.267	43.271	43.160	43.269	43.369	43.270
6	43.024	42.980	42.971	43.129	42.988	43.654	42.990
7	42.402	42.384	42.371	42.860	42.400	43.260	42.401
8	42.130	42.108	42.097	42.871	42.140	43.361	42.142
9	43.798	43.789	43.783	43.604	43.780	44.157	43.781
10	47.950	47.943	47.948	47.879	47.947	47.931	47.948
11	44.664	44.677	44.670	44.329	44.669	44.745	44.670
12	44.004	44.020	44.014	43.762	44.001	44.288	44.003
13	49.274	49.298	49.300	49.206	49.300	49.484	49.301
14	49.099	49.095	49.098	49.086	49.098	49.119	49.098
15	49.057	49.052	49.054	49.047	49.054	49.066	49.054
16	49.298	49.319	49.321	49.212	49.321	49.545	49.322
17	47.970	47.965	47.968	47.878	47.968	48.010	47.969
18	49.338	49.341	49.342	49.218	49.342	49.599	49.344
19	49.029	49.040	49.038	48.818	49.037	49.107	49.038
20	46.618	46.645	46.622	46.135	46.622	46.352	46.622
21	45.623	45.645	45.631	45.249	45.629	45.580	45.630
22	46.588	46.614	46.592	46.131	46.589	46.355	46.589
23	48.379	48.386	48.380	48.087	48.378	48.348	48.379
24	43.249	43.231	43.224	43.186	43.233	43.760	43.236
25	42.532	42.499	42.488	42.964	42.519	43.472	42.522
26	32.086	32.098	32.101	32.091	32.101	33.149	32.107
27	-15.233	-15.196	-15.196	-15.173	-15.196	-15.201	-15.196
28	-33.521	-33.499	-33.500	-33.487	-33.500	-33.482	-33.500
29	31.692	31.699	31.702	31.670	31.702	32.680	31.707
30	43.582	43.607	43.601	43.436	43.600	43.980	43.603
31	44.188	44.199	44.198	43.753	44.197	43.783	44.197
32	-45.710	-45.750	-45.721	-45.877	-45.721	-45.912	-45.721
33	-36.572	-36.582	-36.581	-36.566	-36.581	-36.299	-36.580
34	-12.184	-12.197	-12.197	-12.162	-12.197	-11.665	-12.192
35	0.0723	0.0722	0.0723	0.0722	0.0723	0.0714	0.0723
36	0.0927	0.0925	0.0926	0.0897	0.0926	0.0886	0.0926
37	-0.0229	-0.0229	-0.0229	-0.0225	-0.0229	-0.0168	-0.0229
38	-0.0519	-0.0523	-0.0522	-0.0554	-0.0523	-0.0514	-0.0522
39	-0.0391	-0.0392	-0.0390	-0.0393	-0.0390	-0.0411	-0.0390
40	0.0254	0.0261	0.0254	0.0252	0.0254	0.0243	0.0254
41	0.0614	0.0614	0.0614	0.0616	0.0614	0.0635	0.0614
42	0.1061	0.1063	0.1063	0.1067	0.1063	0.1039	0.1063

Table 4-1: 34-node-system state estimates (73 equations; redundancy ratio=1.74)

1-34: nodal heads (m Aq) at nodes 1-34;

35-42: fixed-head nodes in/out flows (m^3/s) at nodes 27-34

LS - least squares method; LAV - least absolute values method

State Variable	Exact value	LS	LAV	LS (Ex.1)	LAV (Ex.1)	LS (Ex.2)	LAV (Ex.2)
1	32.638	32.662	32.661	32.607	32.663	33.697	32.673
2	43.749	43.750	43.756	43.326	43.756	43.823	43.758
3	46.041	46.069	46.052	45.273	46.052	45.811	46.073
4	46.618	46.656	46.637	45.859	46.635	46.468	46.662
5	43.265	43.250	43.263	43.096	43.263	43.820	43.269
6	43.024	42.971	42.976	43.068	42.978	44.150	43.000
7	42.402	42.363	42.380	42.853	42.391	43.829	42.393
8	42.130	42.094	42.106	42.854	42.130	43.907	42.131
9	43.798	43.789	43.786	43.470	43.780	44.638	43.815
10	47.950	47.943	47.944	47.725	47.946	48.048	47.947
11	44.664	44.681	44.672	44.165	44.670	45.010	44.671
12	44.004	44.022	44.016	43.619	44.007	44.658	44.039
13	49.274	49.306	49.303	49.016	49.306	49.703	49.315
14	49.099	49.098	49.095	48.932	49.096	49.271	49.097
15	49.057	49.055	49.052	48.897	49.050	49.214	49.050
16	49.298	49.327	49.324	49.022	49.327	49.734	49.337
17	47.970	47.967	47.967	47.719	47.969	48.148	47.970
18	49.338	49.349	49.346	49.029	49.348	49.767	49.358
19	49.029	49.049	49.044	48.591	49.046	49.248	49.058
20	46.618	46.656	46.637	45.863	46.635	46.484	46.661
21	45.623	45.652	45.640	45.028	45.637	45.804	45.663
22	46.588	46.624	46.607	45.863	46.606	46.499	46.632
23	48.379	48.396	48.388	47.843	48.389	48.488	48.405
24	43.249	43.225	43.227	43.115	43.223	44.245	43.253
25	42.532	42.487	42.497	42.933	42.509	44.005	42.520
26	32.086	32.103	32.101	32.016	32.105	33.230	32.115
27	-15.233	-15.197	-15.197	-15.145	-15.197	-15.231	-15.198
28	-33.521	-33.500	-33.499	-33.479	-33.500	-33.474	-33.500
29	31.692	31.704	31.702	31.570	31.706	32.778	31.717
30	43.582	43.606	43.603	43.315	43.590	44.392	43.629
31	44.188	44.206	44.198	43.527	44.201	43.910	44.201
32	-45.710	-45.747	-45.725	-45.929	-45.721	-45.951	-45.721
33	-36.572	-36.582	-36.581	-36.560	-36.581	-36.298	-36.580
34	-12.184	-12.196	-12.193	-12.145	-12.198	-11.659	-12.191
35	0.0723	0.0722	0.0723	0.0719	0.0723	0.0716	0.0723
36	0.0927	0.0925	0.0925	0.0886	0.0926	0.0874	0.0927
37	-0.0229	-0.0229	-0.0229	-0.0223	-0.0229	-0.0170	-0.0229
38	-0.0519	-0.0523	-0.0523	-0.0554	-0.0524	-0.0510	-0.0522
39	-0.0391	-0.0392	-0.0391	-0.0394	-0.0391	-0.0427	-0.0393
40	0.0254	0.0261	0.0257	0.0249	0.0254	0.0235	0.0254
41	0.0614	0.0614	0.0614	0.0619	0.0614	0.0633	0.0614
42	0.1061	0.1063	0.1063	0.1073	0.1063	0.1033	0.1062

Table 4-2: 34-node-system state estimates (59 equations; redundancy ratio 1.4)

1-34: nodal heads (m Aq) at nodes 1-34;

35-42: fixed-head nodes in/out flows (m^3/s) at nodes 27-34

LS - least squares method; LAV - least absolute values method

State variable	Exact val-ue	State estimation errors					
		LS	LAV	LS (Ex.1)	LAV (Ex.1)	LS (Ex.2)	LAV (Ex.2)
1	32.638	0.020	0.023	0.024	0.023	0.992	0.028
2	43.749	0.007	0.010	-0.289	0.008	-0.242	0.008
3	46.041	0.017	0.001	-0.494	0.001	-0.341	0.001
4	46.618	0.027	0.004	-0.485	0.004	-0.276	0.004
5	43.265	0.002	0.006	-0.105	0.004	0.104	0.005
6	43.024	-0.044	-0.053	0.105	-0.036	0.630	-0.034
7	42.402	-0.017	-0.031	0.458	-0.002	0.858	-0.001
8	42.130	-0.021	-0.033	0.741	0.010	1.231	0.012
9	43.798	-0.009	-0.015	-0.194	-0.017	0.359	-0.017
10	47.950	-0.007	-0.002	-0.071	-0.003	-0.019	-0.002
11	44.664	0.013	0.006	-0.335	0.005	0.081	0.006
12	44.004	0.016	0.010	-0.241	-0.003	0.284	-0.001
13	49.274	0.024	0.026	-0.068	0.026	0.210	0.027
14	49.099	-0.004	-0.001	-0.013	-0.001	0.020	-0.001
15	49.057	-0.005	-0.003	-0.010	-0.003	0.009	-0.003
16	49.298	0.021	0.023	-0.086	0.023	0.247	0.024
17	47.970	-0.005	-0.002	-0.092	-0.002	0.040	-0.001
18	49.338	0.003	0.004	-0.120	0.004	0.261	0.006
19	49.029	0.011	0.009	-0.211	0.008	0.078	0.009
20	46.618	0.027	0.004	-0.483	0.004	-0.266	0.004
21	45.623	0.022	0.008	-0.373	0.006	-0.043	0.007
22	46.588	0.026	0.004	-0.457	0.001	-0.232	0.001
23	48.379	0.007	0.001	-0.292	-0.001	-0.031	0.000
24	43.249	-0.018	-0.025	-0.063	-0.016	0.511	-0.013
25	42.532	-0.033	-0.043	0.432	-0.013	0.940	-0.010
26	32.086	0.012	0.015	0.005	0.015	1.063	0.021
27	-15.233	0.037	0.037	0.060	0.037	0.032	0.037
28	-33.521	0.022	0.021	0.034	0.021	0.039	0.021
29	31.692	0.007	0.010	-0.022	0.010	0.988	0.015
30	43.582	0.025	0.019	-0.146	0.018	0.398	0.021
31	44.188	0.011	0.010	-0.434	0.009	-0.405	0.009
32	-45.710	-0.040	-0.011	-0.167	-0.011	-0.202	-0.011
33	-36.572	-0.010	-0.009	0.006	-0.009	0.273	-0.008
34	-12.184	-0.013	-0.013	0.022	-0.013	0.519	-0.008
35	0.0723	-0.0001	0	-0.0001	0	-0.0009	0
36	0.0927	-0.0002	-0.0001	-0.0030	-0.0001	-0.0041	-0.0001
37	-0.0229	0	0	0.0004	0	0.0061	0
38	-0.0519	-0.0004	-0.0003	-0.0035	-0.0004	0.0005	-0.0003
39	-0.0391	-0.0001	0.0001	-0.0002	0.0001	-0.0020	0.0001
40	0.0254	0.0007	0.0000	-0.0002	0.0000	-0.0011	0.0000
41	0.0614	0	0	0.0002	0	0.0021	0
42	0.1061	0.0002	0.0002	0.0006	0.0002	-0.0022	0.0002

Table 4-3: 34-node-system state estimation errors for the state estimates shown in Table 4-1(73 equations; redundancy ratio=1.74)

1-34: nodal heads (m Aq) at nodes 1-34; 35-42: fixed-head nodes in/out flows (m^3/s) at nodes 27-34; LS - least squares method; LAV - least absolute values method

State Variable	Exact value	State estimation errors					
		LS	LAV	LS (Ex.1)	LAV (Ex.1)	LS (Ex.2)	LAV (Ex.2)
1	32.638	0.024	0.023	-0.031	0.025	1.059	0.035
2	43.749	0.001	0.007	-0.423	0.007	0.074	0.009
3	46.041	0.028	0.011	-0.768	0.011	-0.230	0.032
4	46.618	0.038	0.019	-0.759	0.017	-0.150	0.044
5	43.265	-0.015	-0.002	-0.169	-0.002	0.555	0.004
6	43.024	-0.053	-0.048	0.044	-0.046	1.126	-0.024
7	42.402	-0.039	-0.021	0.451	-0.011	1.427	-0.009
8	42.130	-0.035	-0.024	0.724	0.000	1.777	0.001
9	43.798	-0.009	-0.012	-0.328	-0.018	0.840	0.017
10	47.950	-0.007	-0.006	-0.225	-0.003	0.098	-0.003
11	44.664	0.017	0.008	-0.499	0.006	0.346	0.007
12	44.004	0.018	0.012	-0.385	0.003	0.654	0.035
13	49.274	0.032	0.029	-0.258	0.032	0.429	0.041
14	49.099	-0.001	-0.004	-0.166	-0.003	0.172	-0.001
15	49.057	-0.002	-0.005	-0.160	-0.006	0.157	-0.006
16	49.298	0.029	0.026	-0.275	0.029	0.436	0.039
17	47.970	-0.002	-0.003	-0.251	-0.001	0.178	0.000
18	49.338	0.011	0.008	-0.309	0.010	0.429	0.020
19	49.029	0.020	0.015	-0.437	0.017	0.219	0.029
20	46.618	0.038	0.019	-0.755	0.017	-0.134	0.043
21	45.623	0.029	0.017	-0.595	0.014	0.181	0.040
22	46.588	0.036	0.019	-0.725	0.018	-0.088	0.044
23	48.379	0.017	0.009	-0.536	0.010	0.109	0.026
24	43.249	-0.024	-0.022	-0.134	-0.026	0.996	0.004
25	42.532	-0.045	-0.035	0.401	-0.022	1.473	-0.012
26	32.086	0.017	0.015	-0.070	0.019	1.144	0.029
27	-15.233	0.036	0.036	0.088	0.036	0.002	0.035
28	-33.521	0.021	0.022	0.042	0.021	0.047	0.021
29	31.692	0.012	0.010	-0.122	0.014	1.086	0.025
30	43.582	0.024	0.021	-0.267	0.008	0.810	0.047
31	44.188	0.018	0.010	-0.661	0.013	-0.278	0.013
32	-45.710	-0.037	-0.015	-0.219	-0.011	-0.241	-0.011
33	-36.572	-0.010	-0.009	0.012	-0.009	0.274	-0.008
34	-12.184	-0.012	-0.009	0.039	-0.014	0.525	-0.007
35	0.0723	-0.0001	0	-0.0004	0	-0.0007	0
36	0.0927	-0.0002	-0.0002	-0.0041	-0.0001	-0.0053	0
37	-0.0229	0	0	0.0006	0	0.0059	0
38	-0.0519	-0.0004	-0.0004	-0.0035	-0.0005	0.0009	-0.0003
39	-0.0391	-0.0001	0	-0.0003	0	-0.0036	-0.0002
40	0.0254	0.0007	0.0003	-0.0005	0	-0.0019	0
41	0.0614	0	0	0.0005	0	0.0019	0
42	0.1061	0.0002	0.0002	0.0012	0.0002	-0.0028	0.0001

Table 4-4: 34-node-system state estimation errors for the state estimates shown in Table 4-2 (59 equations; redundancy ratio 1.4)

1-34: nodal heads (m Aq) at nodes 1-34; 35-42: fixed-head nodes in/out flows (m^3/s) at nodes 27-34; LS - least squares method; LAV - least absolute values method

4.5. Concluding remarks

To the author's knowledge the neural network approach to the state estimation problem in water systems has not been reported in the literature before and in this sense it is an original contribution of this work. The resulting algorithms are the mixture of the well known and tested ways of solving systems of nonlinear equations (the Newton-Raphson method), the optimization criterions (the LS, LAV and their variations) and a relatively new ANN technique of finding the solution to the overdetermined systems of linear equations.

This chapter presented several neural network's formulations of the problems that traditionally have been solved using techniques like the linear programming or Gauss elimination. It has been found, through the simulation study, that neural network based state estimators provide an efficient means of water system state estimation. It is interesting to notice that all the neural networks discussed in this chapter are very similar and can be implemented by one "general purpose" artificial neural network with a suitable control subsystem which depending on the requirements will be able to find a solution of the system of linear equations according to the desired criterion (LS or LAV criterion).

It has been confirmed that while the LS estimates have shown to be strongly affected by any change in the measurement vector, the LAV estimates proved to be resistant to large changes in the data. This is a very useful property when the known data in the measurement vector are contaminated with occasional gross errors.

The main restriction of a VLSI implementation of neural networks is the number of connections between the processing units on a chip. It is envisaged that, with the current rate of development in microelectronic and optical technology, it will be possible to implement an arbitrarily large ANN in the near future. Consequently the state estimation process, as discussed in this chapter, will be accomplished in a time of order of hundred microseconds.

Chapter 5

Confidence limit analysis - a neural network approach

5.1. Introduction

In the previous chapter the state estimation algorithms that produce optimal state estimates for certain distributions of errors were described. It is to say that for a given set of measurements the obtained state estimates are close to the true operating state. Although for a given set of measurements and estimation criterion there is only one optimal solution, due to the inaccuracies of measurements there are many possible, different combinations of measurement values. This implies that there are many feasible, different state estimate vectors.

Since the uncertainty is inevitable part of water distribution systems it is very important, from the safety of the system operational control point of view, to know how the inaccuracies can affect the estimated solution or in other words how reliable the state estimates are. In the scientific literature the process of assessing the influence of perturbations or inaccuracies of mathematical models or measurements is known under the names of sensitivity analysis, error propagation, perturbation theory or calculating of confidence intervals. In water systems the quantification of the inaccuracy of state estimates caused by the input data uncertainty is called confidence limit analysis (Bargiela & Hainsworth, 1988).

Some applications, semi-automated or on-line decision support for instance, need a confidence limit analysis procedure that can produce uncertainty bounds in real time. This is, however, the task requiring much of computational effort. ANNs, considering their known properties like massively parallel structure, fault tolerance etc., are seen as a means of overcoming the computational complexity.

In this chapter, after presenting the review of previous research, the ways of utilizing the neural networks, discussed in Chapter 4, for the purpose of finding confidence limits are given. Then, an integrated neural based system for state estimation and confidence limit analysis in water network is presented. This system is tested for a realistic water network.

5.2. Review of previous research

Several works have been published which deal with the problems of state estimation under measurement uncertainty. Many of these are concerned with the problem of uncertainty in utility systems (Bargiela & Hainsworth, 1988; Borkowska, 1974; Dopazo et al., 1975; Hainsworth, 1988; Sobierajski, 1979; Stuart & Herget, 1973).

In the previous chapter the deterministic state estimation algorithms were presented which find the state that best fits the measurement data. As it was shown this state estimation approach relies on statistical or probabilistic assumptions about behaviour of the measurement errors. Similar assumptions about the behaviour of measurement errors can be used to provide an indication of the reliability or accuracy of these deterministic state estimates.

In (Dopazo et al., 1975; Stuart & Herget, 1973) it was shown that assuming the statistical properties of measurement errors are known in advance the state covariance matrix $(\boldsymbol{J}^T\boldsymbol{R}^{-1}\boldsymbol{J})^{-1}$ (where $\boldsymbol{R}$ is the measurement covariance matrix and $\boldsymbol{J}$ is the Jacobian matrix calculated for the optimal state estimates) can be found. The elements on the leading diagonal of this state covariance matrix correspond to the variance of each state variable and thus they have been suggested as a measure of the accuracy of state variables. It can be carried even further since from statistics it is known that, with 99% probability, the true value of the variable is contained within the interval enclosed by its estimated value ± 3 times its standard deviation. This, however, implies the normal distribution of errors that, as it was discussed in previous chapter, is very often unrealistic assumption.

Other statistical methods, using probability density functions defined for each state variable in order to calculate the probability that a given variable is above or below certain limit, have also been used (Borkowska, 1974; Sobierajski, 1979). This approach requires even more accurate information about the type of distribution of measurement errors which, again, is not available in enough detail for water distribution systems.

The most widely used method of assessing the sensitivity of the linear (or linearised) system solution is based on the system matrix condition number (Golub & Van Loan, 1986). The condition number for LS problem is defined as the ratio of the largest and smallest singular values of the system matrix. In the case of water networks presented in this work this system matrix is Jacobian calculated for the optimal state estimates. If the columns of Jacobian matrix are nearly linearly dependent then condition number is large and we say that the problem is ill-conditioned. And the ill-conditioned LS problems have sensitive solutions. It means that small perturbations in the vector of measurements can cause large changes in the solution. In this sense, the condition number gives a measure for the quality of the solution for a particular measurement set. However, since only one value is provided by this method it cannot be used to produce absolute, tight bounds for each individual state variable as is required by confidence limit analysis. In other words while the bound obtained using condition matrix method can be justified for some perturbations in the measurement vector it will be a gross overestimate for others.

The extensive work on quantification of the influence of measurement and pseudomeasurement uncertainties in water distribution systems, called confidence limit analysis, has been carried out by Bargiela and Hainsworth (Bargiela & Hainsworth, 1989; Hainsworth, 1988). In this work we adopted the following model with unknown-but-bounded errors used by Bargiela and Hainsworth:

$$z = g(\boldsymbol{x}) + \omega \ , \ |\omega_i| \le |e_i| \ , \ \text{i=1,...,m} \qquad \text{(EQ 5-1)}$$

where $\boldsymbol{e}$ is the vector representing the maximum expected measurement errors.

Please notice that the assumption of the unknown-but-bounded errors means that the knowledge of statistical properties of errors is not required and the only restriction imposed is the one of errors falling within a range bounded by $\boldsymbol{e}$. However, it also means that the data has to be free of gross errors, or in other words the gross errors have to be filtered out before using the above model. Since the gross errors are of random and unbounded nature the reasonable boundary in $\boldsymbol{e}$ could not be set.

Let us now review several confidence limit analysis algorithms.

In general, there is no direct, theoretical way of finding the sensitivity of the solution to the changes in model parameters in nonlinear systems as the one described by (EQ 5-1). The uncertainty in the measurement data means that the measurement vector, instead of being single valued, can take any one value of a whole range of feasible states. The basic idea is to use the deterministic state estimator repeatedly for a large number of measurement vectors chosen from within this range. This approach to the sensitivity analysis of nonlinear systems is known as the Monte Carlo method. Each calculated state estimates are checked against the maximum and minimum values obtained from previous simulations and new maxima are set where appropriate. In this way the error bounds for the state variables can be gradually increased until, after many trials, their limits are asymptotically reached.

Although the Monte Carlo method is effective in providing realistic state error vectors, its computational complexity tends to be a major drawback. Even for a medium-sized system the number of feasible measurement vectors is enormous, thus rendering this approach impractical for on-line control applications. In view of these limitations, the methods based on the linearised model of the system have been used.

In contrast to the nonlinear systems the perturbation theory or sensitivity analysis for linear systems has been well documented and researched (Cope & Rust, 1979; Golub & Van Loan, 1986). It is due to the fact that linear systems are simpler and there are a lot of existing, formal methods used to analyse such systems. Bargiela and Hainsworth proposed two alternative methods, to the Monte Carlo method, based on an accurate linearisation of the system model.

First of these methods uses the linearised network equations as constraints in a mathematical optimization problem and confidence limits are found solving the standard linear programming problem. This method produced the acceptable results (the linearisation did not significantly affect the values of calculated error bounds) much faster than the Monte Carlo method. However it was reported to be not efficient enough for on-line decision support.

In the second method the linearisation of the system model was used to construct the sensitivity matrix. The sensitivity matrix, as given in (Bargiela & Hainsworth, 1989), is the inverse of the Jacobian matrix calculated for the state estimates obtained using the deterministic state estimator. So in this method first a state estimate is produced on the assumption that the measurement vector is correct. Then the possible error of the measurement set is considered and it is used, together with the sensitivity matrix, to predict the resulting error in the state vector. Due to significantly fewer mathematical operations required, this approach proved to be more efficient than previously discussed two methods while producing almost identical results as the linear programming method.

In the following sections we will show how to use neural networks presented in the previous chapter to obtain confidence limits. All algorithms presented below are based on the linearised model of the system. There will also be shown the implications of using different estimation criterions while calculating confidence limits and dealing with overdetermined systems of equations.

5.3. Neural linearised confidence limit algorithms

The linearised model of the system, as presented in Chapter 2, was written in the following mathematical form

$$J^{(k)}\Delta x^{(k)} = z - g(\hat{x}^{(k)}) \qquad \text{(EQ 5-2)}$$

Let $\hat{b}$ represent the difference $z - g(\hat{x})$ and $\hat{J} = \left.\frac{\partial g}{\partial x}\right|_{x = \hat{x}}$. If $\hat{x}^{(k)}$ in equation (EQ 5-2) is replaced by the best estimate of the true state vector, $\hat{x}$, for the system, $\hat{b}$ will then represent the difference between the measured vector and the values of the measured variables calculated for $\hat{x}$.

With the symbols introduced above the linear equation (EQ 5-2) takes form:

$$\hat{J}\Delta x = \hat{b} \qquad \text{(EQ 5-3)}$$

It was shown in (Hainsworth, 1988) that for the water systems the linearised model (EQ 5-3) can be used while analysing the influence of measurement uncertainties in the non-linear model (EQ 5-1) and the confidence limits produced using this linearised model compare very well with the results obtained from the Monte Carlo method.

So, now it is of interest to see how the solution of the equation (EQ 5-3) is affected by perturbation in vector of measurements $\hat{b}$ (effectively in vector **z** since $g(\hat{x})$ is constant for a fixed state vector). While Hainsworth carried out the confidence limit analysis for square systems of equations where as long as the Jacobian matrix is non-singular there is only one solution the CLA for overdetermined systems of equations involves the optimization criterias. For a square system of linear equations every change in the system's parameters results in changes in the solution. On the other hand, a solution to an overdetermined system need not to be affected by changes to some system parameters at all or the influence of disturbances is averaged in the process of optimization. However, the need to use an optimization criterion makes the CLA more difficult. It will be shown now that the sensitivity matrix method resulted from the sensitivity consideration of the overdetermined system of linear equations solution according to *the ordinary least squares criterion* and it should not be used directly when the state estimate vector is found using LAV or other weighted criteria.

5.3.1. Sensitivity matrix method

If we introduce the vector of perturbations δb into (EQ 5-3) we have

$$\hat{J}(\Delta x + \Delta x_p) = \hat{b} + \delta b \qquad \text{(EQ 5-4)}$$

where δb is a vector of perturbations and Δx_p is a vector of changes in state estimates caused by perturbations δb.

If $\hat{J}$ has full column rank then there is a unique LS solution Δx to the equation (EQ 5-3) and it solves the symmetric positive definite system

$$\hat{J}^T \hat{J} \Delta x = \hat{J}^T \hat{b} \qquad \text{(EQ 5-5)}$$

$$\Delta x = (\hat{J}^T \hat{J})^{-1} \hat{J}^T \hat{b} \qquad \text{(EQ 5-6)}$$

Now when we do the same for the set of equations (EQ 5-4) we have:

$$\hat{J}^T \hat{J}(\Delta x + \Delta x_p) = \hat{J}^T(\hat{b} + \delta b) \qquad \text{(EQ 5-7)}$$

$$\Delta x + \Delta x_p = (\hat{J}^T \hat{J})^{-1} \hat{J}^T(\hat{b} + \delta b) \qquad \text{(EQ 5-8)}$$

Combining (EQ 5-6) and (EQ 5-8) we obtain the following relationship between the changes in the vector of measurements and the solution of (EQ 5-3) according to LS criterion

$$\Delta x_p = (\hat{J}^T \hat{J})^{-1} \hat{J}^T \delta b \qquad \text{(EQ 5-9)}$$

Matrix $S = (\hat{J}^T\hat{J})^{-1}\hat{J}^T \in R^{nxm}$ is the pseudoinverse of the Jacobian matrix $\hat{J}$. It is also called *the sensitivity matrix* because the (*i,j*)-th element s_{ij} of this matrix relates the sensitivity of the *i*-th element, x_i, of the vector $\hat{x}$ to the *j*-th element, z_j, of the measurement vector **z**.

Vector $\delta\boldsymbol{b}$ in (EQ 5-9) consists of values that are within the range: $\pm$variability of consumption or $\pm$accuracy of meter in case of a measurement. The underlying principle of the CLA is the consideration of the worst possible case. It means that the maximum variabilities of consumptions and inaccuracies of meters are assumed during the calculations. Now, for one state variable x_i, calculating its error bound is just a matter of maximizing $s_i\delta\boldsymbol{b}$, where s_i is the *i*-th row of the sensitivity matrix $\boldsymbol{S}$.

Since the equation (EQ 5-9) is linear the calculated bounds are symmetrical meaning that upper bound is equal to the absolute value of the lower bound for each state variable.

The main problem in the sensitivity matrix method is finding the sensitivity matrix itself, but before the way of utilizing the neural networks presented in Chapter 4 for the purpose of calculating the inverse and pseudoinverse matrices will be shown, let us clarify that the sensitivity matrix method should not be directly used for calculating confidence limits when criterions other than LS have been used to obtain the state estimates.

The above presented sensitivity analysis of the LS solution to (EQ 5-3) resulted in the direct relationship between hypothetical disturbances (uncertainties) $\delta\boldsymbol{b}$ and the vector of changes in the LS solution $\Delta\boldsymbol{x}_p$, caused by $\delta\boldsymbol{b}$. Unfortunately for other optimization criteria (e.g. LAV or weighted LS where weights depend on current values of the residuals and effectively on a particular measurement vector) it is not possible to derive the direct solution of the type represented by (EQ 5-6). The nonlinearities involved in these optimization criteria make also the derivation of the sensitivity-perturbation type of formula represented by (EQ 5-9) impossible. In such cases the sensitivity analysis can be carried out using the Monte Carlo method. This method, however, is not suitable for on-line confidence limit analysis. It is therefore suggested that the criteria resistant to the gross errors in data should be used as preliminary filters. Once the data is gross error free, which is implied by the model (EQ 5-1) anyway, the LS criterion and more efficient CLA (e.g. sensitivity matrix method) can be used.

5.3.2. Pseudo-inverse matrix - a neural approach

In (Golub & Van Loan, 1986) it can be found that the matrix $A^{\dagger} \in R^{nxm}$, referred to as the pseudo-inverse of $\boldsymbol{A}$, is the unique minimal 2-norm solution to the problem

$$\min_{Y \in R^{nxm}} \|\boldsymbol{A}\boldsymbol{Y} - \boldsymbol{I}\|_2 \qquad \text{(EQ 5-10)}$$

and satisfies the four Moore-Penrose conditions:

(i) $\boldsymbol{AYA} = \boldsymbol{A}$; (ii) $\boldsymbol{YAY} = \boldsymbol{Y}$; (iii) $(\boldsymbol{AY})^T = \boldsymbol{AY}$; and (iv) $(\boldsymbol{YA})^T = \boldsymbol{YA}$

In order to find the pseudo-inverse matrix $\boldsymbol{Y}$ the problem (EQ 5-10) can be mapped to the task of finding the solutions to the m problems

$$\min_{Y_i \in R^n} \|\boldsymbol{A}\boldsymbol{Y}_i - \boldsymbol{I}_i\|_2 \qquad \text{i=1,...,m} \qquad \text{(EQ 5-11)}$$

where $\boldsymbol{Y}_i$ is the i-th column of the matrix Y; $\boldsymbol{I}_i$ is the i-th column of the identity matrix $\boldsymbol{I} \in R^{mxm}$.

(EQ 5-11) represents the LS problems of finding the solution to the overdetermined sets of linear equations as discussed in previous chapter. Therefore in order to find the pseudo-inverse matrix $\boldsymbol{Y}$ we can use the neural network shown in Figure 4-2 with linear activation function. Repeating computation m times we find m columns of the pseudo-inverse matrix Y. Please notice that the columns of $\boldsymbol{Y}$ can be found independently thus significantly reducing the computation time.

Having found the sensitivity matrix, the inverse or pseudoinverse of the Jacobian matrix $\hat{J}$, we can now carry out the maximization process in order to obtain the confidence limits for state variables as given in (Bargiela & Hainsworth, 1989).

For the i-th state variable, calculating its error bound was done by maximizing the product of the i-th row of the sensitivity matrix $\boldsymbol{S}$ and the vector $\delta\boldsymbol{b}$. This maximization was performed for each row of the sensitivity matrix separately. However, the same results can be easily obtained by making all values of the sensitivity matrix positive and assuming positive values for the elements of the vector $\delta\boldsymbol{b}$. In this way the error bounds are obtained by straightforward matrix-vector multiplication without the need for the maximization process repeated for each state variable separately.

5.3.3. Superposition method

The sensitivity matrix presented in the previous section contains a lot of information about possible behaviour of the system in the near future. The value of the s_{ij} element of this matrix says how sensitive the *i*-th state variable is to the changes in the *j*-th measurement at this particular operating state while the sign of the s_{ij} element says if the value of the *i*-th state variable will increase or decrease when the *j*-th measurement will increase or decrease. For example if our *j*-th measurement is the consumption at some node, looking at the *j*-th column of the sensitivity matrix will tell us which pressures in the network can be affected the most by this change and whether they will increase or decrease. Such information might have a paramount consequence for safety of the system operation.

However, when the major values of interest are the confidence limits for the state variables the procedure of obtaining them can be made quicker than the one finding the sensitivity matrix first and then computing the confidence limits as described above.

Again, utilizing the fact that the equation (EQ 5-9) is linear and each measurement has potentially the same degree of influence on the final solution each perturbation can be analysed separately and their impacts can be summed up. This technique is known under the name of the superposition and has been widely used for analysis of linear systems (e.g. in linear circuit analysis). The proposed method of finding the confidence limits utilizes the neural network presented in Figure 4-2 with the linear activation function.

Making the vector $\Delta \boldsymbol{b}$ of the equation $\boldsymbol{J}\Delta \boldsymbol{x}_{cl}^{(i)} = \Delta \boldsymbol{b}^{(i)}$ successively $\begin{bmatrix}\Delta b_1 & 0 & \dots & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & \Delta b_2 & \dots & 0\end{bmatrix}^T$,..., $\begin{bmatrix}0 & 0 & \dots & \Delta b_m\end{bmatrix}^T$ the confidence vector $\boldsymbol{x}_{cl}$ is found by summing up the absolute values of $\Delta \boldsymbol{x}_{cl}^{(i)}$ for each source of inaccuracy Δb_i, i=1,...,m.

$$\boldsymbol{x}_{cl} = \sum_{i=1}^{m} \left|\Delta \boldsymbol{x}_{cl}^{(i)}\right| \qquad \text{(EQ 5-12)}$$

Notice that the vector x_{cl} representing error bounds for state vector $\hat{x}$ is sequentially increased for each non-zero source of inaccuracy (each non-zero element of Δb). Since in every water distribution network model there is a number of mass balance equations written for nodes where there is no inflows or consumptions (i.e. it is represented by $\Delta b_i = 0$) the effective number of possible sources of inaccuracy is diminished by the number of such equations. Therefore, in comparison to sensitivity matrix method the smaller number of iterations is required to obtain the confidence limits.

In the following sections of this chapter it is shown that using relatively simple neural structure described in previous chapter and incorporating some control elements it is possible to construct a system finding optimal state estimates and subsequently assessing the reliability of this solution via confidence limit analysis.

5.4. An integrated neural system for state estimation and CLA

The system depicted in Figure 5-1 is an implementation of a Newton-Raphson method for solving systems of nonlinear equations. The description of this method for water distribution network was given in Chapter 2.

Apart from the state estimation achieved by means of implementing N-R method the system also incorporates the necessary logic and other elements (integrators, absolute value block etc.) to obtain confidence limits for the state estimates. The design and the performance of this system has been reported in (Gabrys & Bargiela, 1996).

The functionality of this system can be summarised as follows:

1) For the initial guess $\hat{x}^{(0)}$ calculate the Jacobian $J^{(0)}$ and the right hand side of the linearised system of equations $z - g(\hat{x}^{(0)})$.

2) Using the analog neural network for solving systems of linear equations ("Neural Estimator") solve $J^{(0)} \Delta x^{(0)} = z - g(\hat{x}^{(0)})$.

3) If all elements of $\Delta x^{(k)}$ (k=0,1,2,... - number of iteration) are lower or equal to predefined accuracy, go to point 7, otherwise go to point 4.

4) Adjust the current state estimate values according to formula: $\hat{x}^{(k+1)} = \hat{x}^{(k)} + \gamma \Delta x^{(k)}$.

5) For $\hat{x}^{(k+1)}$ calculate the Jacobian $J^{(k+1)}$ and the right hand side of the linearised system of equations $z - g(\hat{x}^{(k+1)})$.

6) Using neural based state estimator find the solution to the system of linear equations $J^{(k+1)} \Delta x^{(k+1)} = z - g(\hat{x}^{(k+1)})$ and go to point 3.

7) For the Jacobian calculated for the optimal state estimate vector $\hat{x}$, calculate the confidence limits making the right hand side vector Δb of equation $\hat{J} \Delta x_{cl}^{(i)} = \Delta b^{(i)}$ successively $\begin{bmatrix}\Delta b_1 & 0 & \dots & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & \Delta b_2 & \dots & 0\end{bmatrix}^T$,...,$\begin{bmatrix}0 & 0 & \dots & \Delta b_m\end{bmatrix}^T$ and summing $x_{cl} = \sum_{i=1}^{m} \left|\Delta x_{cl}^{(i)}\right|$, where x_{cl} is the vector of confidence limits of $\hat{x}$ calculated for the vector of disturbances Δb and the state vector $\hat{x}$.

5.4.1. Detailed description of the system and subsystems

There are two output signals from the "Neural Estimator" block. First - LEC (Linear Estimation Control) is the control signal normally set to 0 (zero). It changes from 0 (zero) to 1 (one) (an impulse is generated) every time when the convergence criterion (accuracy condition) of the neural estimator is met. Only then the second signal x_l, which is a vector of current estimates of the neural estimator, is allowed to be processed.

Following the signals LEC and x_l we now go to the block "NR Control". The function of this subsystem is to produce control signals enabling us to distinguish between the state estimation and the confidence limit analysis stages of computation.

Four control signals C1, C2, C3 and C4 of the "NR Control" block decide if the signal y=LEC*x_l is directed to: the "Newton-Raphson method integrator" block - the state estimation stage or the "Confidence Limits Integrator" block - the confidence limits finding stage.

Referring to Figure 5-3 we can see that C1 is equal to LEC when C2 is 1 (one) and 0 (zero) otherwise. C2 is 1 (one) as long as the convergence criteria of the Newton-Raphson method is NOT satisfied, namely when there is even one value of the vector y (at the state estimation stage) greater than predefined accuracy. C2 changes to zero when the state estimation process is completed and we go to the confidence limits analysis stage.

C3 is a logical negation of C2. When C3=1 (equivalent of C2=0) the C4 =LEC. At this stage signal y is directed to the "Confidence Limits Integrator" block.

STAGE 1

At the stage 1 (C2=1) after every adjusting of the state vector **x**, the change of the Jacobian and the right hand side of the linearised system (RHSLS) of equations, synchronised by C1, is carried out. These newly calculated values are than set in the "Neural Estimator" block (Jacobian - Matrix A, Jacobian transposed - Matrix AT, RHSLS - Vector b) and the new adjustments of the state vector are computed. It has to be stressed that there must be enough time available between two subsequent impulses of LEC for calculating Jacobian, RHSLS and setting these parameters in the "Neural Estimator" block. If the time of carrying out those operations is not known or can not be determined

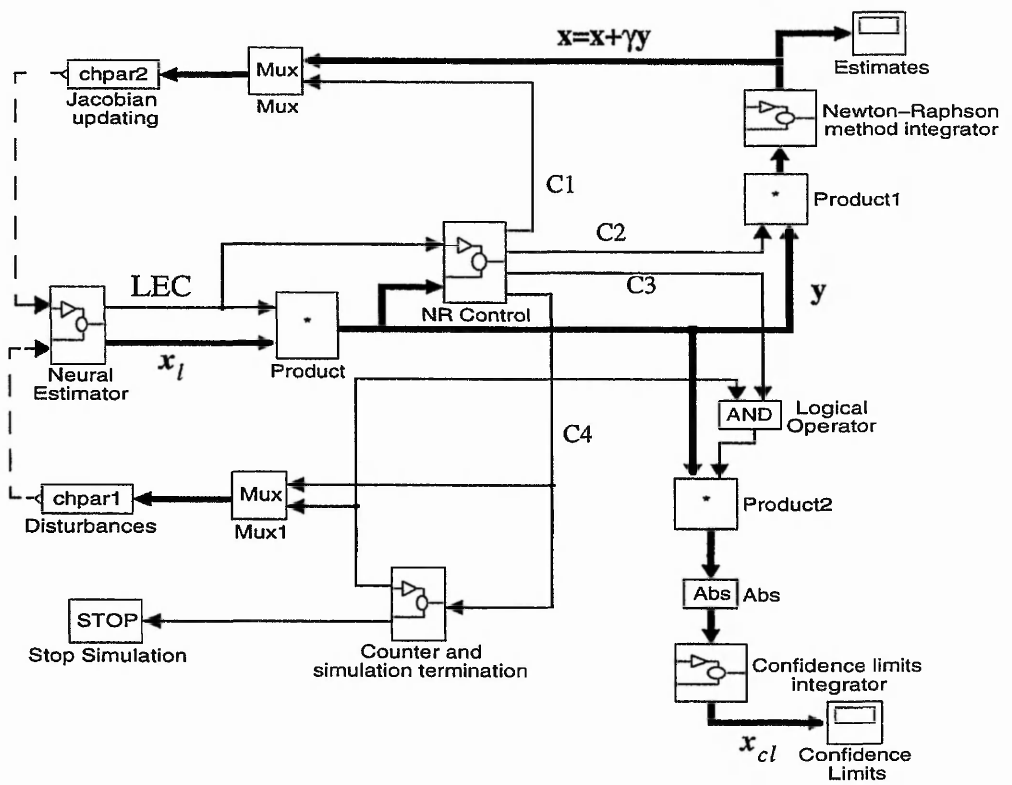


Figure 5-1: The system for estimation (based on Newton-Raphson method) and confidence limit analysis.

the easy solution could be an introduction of a new control signal. The function of this signal would be to make sure that after generating the first LEC impulse there would not be generated the next LEC impulse before the newly calculated Jacobian and RHSLS were not set up in the "Neural Estimator" subsystem.

STAGE 2

Since the adapted method of the confidence limits finding is based on sequential solving of a system of linear equations, we can use the "Neural Estimator" to achieve it. As it has been described previously the matrix $A = \hat{J}$ of the system of equations **Ax=b** (at this stage) does not change. It is the Jacobian calculated for the optimal state estimate vector $\hat{\mathbf{x}}$. Making vector **b** successively $\left[\Delta b_1\ 0\ \ldots\ 0\right]^T$, $\left[0\ \Delta b_2\ \ldots\ 0\right]^T$,..., $\left[0\ 0\ \ldots\ \Delta b_m\right]^T$ (synchronised by the signal C4) and summing the influences of each disturbance on state vector ("Confidence limit integrator" block), we finally get the confidence limits for given vector of disturbances **Δb** and calculated vector of state estimates $\hat{\mathbf{x}}$.

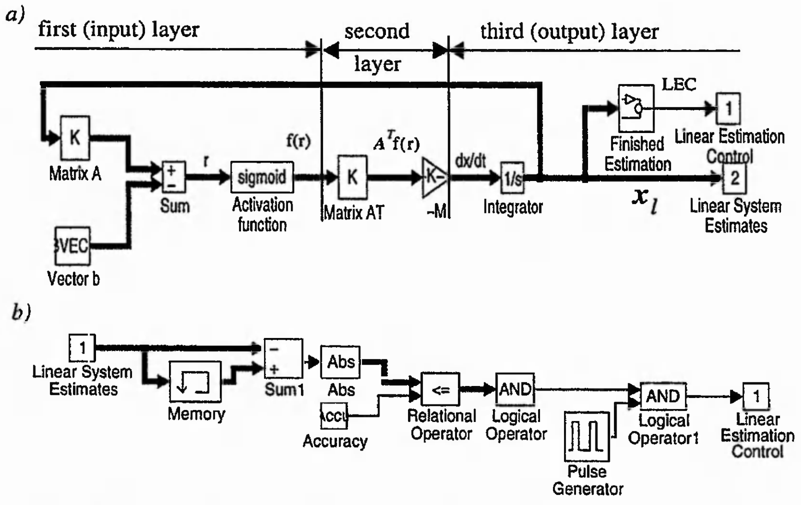


Figure 5-2: a) Neural network for solving systems of linear equations - Subsystem of the system from Figure 5-1, b) The "Finished Estimation" subsystem of a) - generating impulse when the optimal state estimates are found.

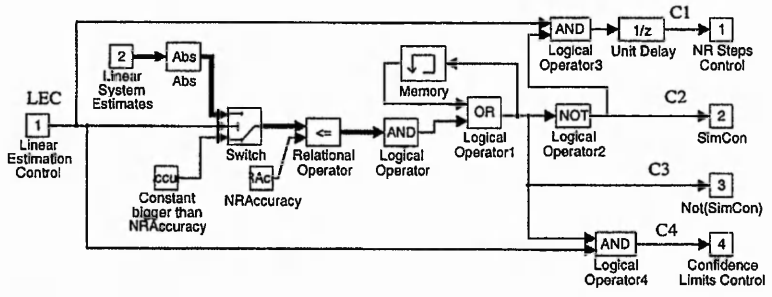


Figure 5-3: The "NR Control" subsystem of the system from Figure 5-1

The simulation terminates when the influence of the last disturbance on the state vector has been adjusted to the vector of confidence limits.

5.5. Simulation results

The performance of the proposed methods for water-system state estimation and CLA was tested on the realistic 34-node network (42 state variables) presented at Figure 4-8. The results are presented in two forms:

a) table containing the state estimates and corresponding confidence limits for three different cases

b) time diagrams - in order to show the relations between control signals and iteratively calculated values of state estimates and confidence limits.

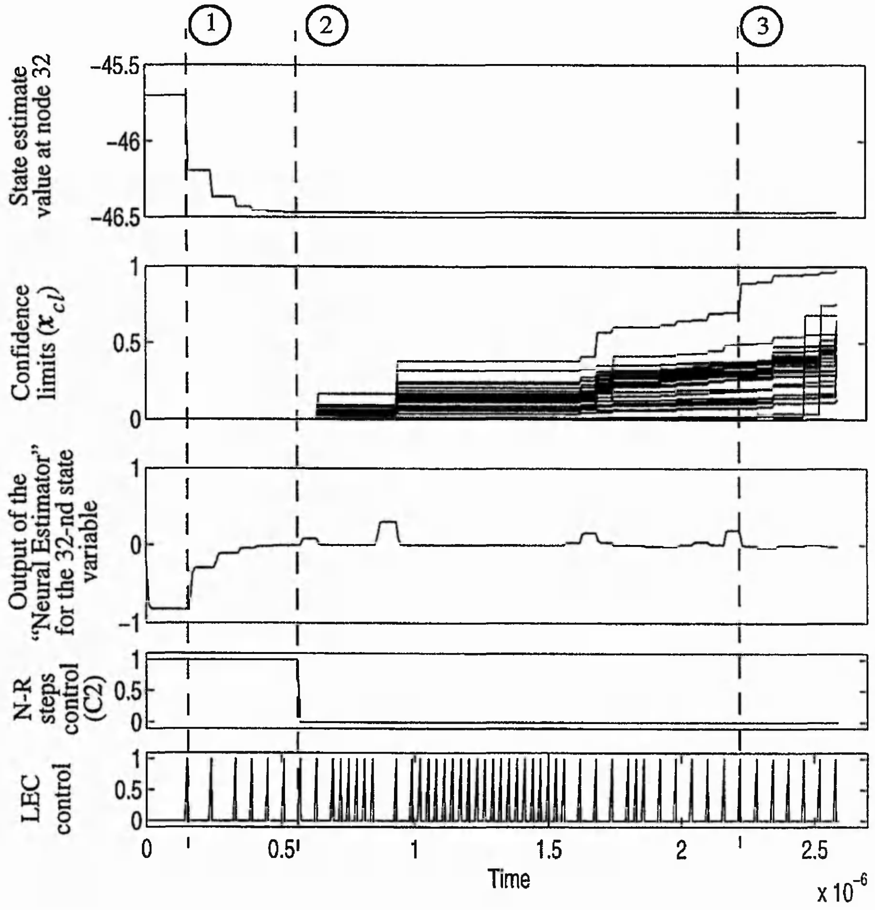


Figure 5-4: The simulation time diagram for 34-node water network - case 3. Dashed lines mark three phases of simulation: 1) First stage - state estimates adjusting; 2) State estimation finished, beginning of the second stage; 3) Second stage - confidence limits finding

Three different cases are considered below.

First we consider the case of minimal set of measurements when we have available only one reference head measurement at node 30. The state estimates and corresponding confidence limits for this case are shown in columns 3 and 4 of Table 5-1 respectively. The purpose of calculating the confidence limits is to obtain an information about how far from the real state the estimated values could be in the worst case. The requirement to have the state estimates as close as possible to the real state is equivalent to the requirement of having the confidence limits to be as tight as possible. The means of achieving that is the introduction of additional accurate measurements into the system.

In the highlighted row (columns 3 to 6) of Table 5-1 we can see the worst (the biggest) confidence limit for the estimated state vector. In order to improve it, in the second case apart from the reference head (node 30) we have also measured the head at node 28. With the metering accuracy of 2%, the measured value can vary within the range of ± 0.7024.

The state estimates and corresponding confidence limits for the second case are shown in columns 5 and 6 of Table 5-1 respectively. Comparing the confidence limits for the 28-th state variable it can be seen that the considerable improvement has been achieved.

Repeating the procedure of finding the biggest confidence limits for the second case two new measurements have been added to the system in the third case. These were the heads in nodes 33 and 34. The state estimates and corresponding confidence limits for the third case are shown in columns 7 and 8 of Table 5-1 respectively. The two highlighted rows (columns 5 to 8) show the variables of interest. In this case apart from the improvement regarding state variables for nodes 33 and 34 we can observe the big improvement in nodes in their direct vicinity (nodes 1, 26, 29) as well as in nodes laying a bit further (nodes 13, 14,15,16,17,18, and 19). The effect wears off with the increasing distance from the meter.

The simulation time diagram for this case is shown in Figure 5-4.

In all cases the simulated time of the calculations (the time that would be required by the actual neural network) was in order of microseconds. It needs to be pointed out, however, that the simulation of the neural network was performed on a serial computer (SparcStation IPC) and the corresponding lapsed time for the simulation was of order of hundreds of seconds.

It is commonly known that with the bigger number of measurements the reliability of estimation increases. It is due to averaging property of systems with high redundancy ratio. Redundancy ratio is defined as the ratio of the number of equations to the number of unknowns. In other words the influence of single measurement or pseudomeasurement or strictly speaking its inaccuracy is smaller (averaged) for the system with high redundancy ratio.

On the other hand introducing a new measurement we introduce a new source of inconsistency which is the finite accuracy of a meter. The conclusion that can be drawn from these considerations and results is as follows. Addition of the new measurement for *i*-th state variable can have the tightening effect on the confidence limit of this variable only if the error resulted from the inaccuracy of the meter is smaller than confidence limit calculated for existing set of meters.

5.6. Concluding remarks

In this chapter, the problem of reliability of the deterministic state estimation solution of water distribution system, given uncertain measurements, has been examined.

The uncertainty resulting from inaccurate data has been first introduced to the mathematical model of the water distribution system using the concept of unknown-but-bounded errors. The reason behind including this uncertainty into the model is the fact that no deterministic state estimator can produce accurate results from inaccurate data. Therefore, in order to ensure the safe and reliable operation of the water distribution system the state estimates have to be qualified with the degree of confidence that can be put in them.

In (Bargiela & Hainsworth, 1989) it was shown that the results produced by the Monte Carlo method are the most mathematically reliable but the computational burden involved makes this method an unrealistic proposition for real-time applications. On the other hand, the sensitivity matrix method, based on an accurate linearisation of the system model, was said to be computationally efficient while producing comparable results with those of the Monte Carlo method.

State variable	Exact state	State estimates (ex.1)	Confidence limits (ex.1)	State estimates (ex.2)	Confidence limits (ex.2)	State estimates (ex.3)	Confidence limits (ex.3)
1	32.6566	33.2501	1.9336	33.2500	1.9251	32.6906	0.6503
2	43.7173	43.7581	0.3647	43.7843	0.3728	43.7254	0.4234
3	46.0554	46.1320	0.4666	46.1814	0.4754	46.0530	0.4621
4	46.6476	46.7040	0.4468	46.7568	0.4535	46.6187	0.4199
5	43.2138	43.1581	0.2449	43.1734	0.2560	43.1439	0.3001
6	42.9352	42.8410	0.1701	42.8553	0.1768	42.8168	0.2339
7	42.3122	42.1258	0.3563	42.1339	0.3574	42.1239	0.3619
8	42.0405	41.8311	0.3776	41.8418	0.3804	41.8194	0.4069
9	43.7703	43.7536	0.0415	43.7686	0.0469	43.7054	0.1279
10	47.9759	48.0797	0.8288	48.0949	0.8263	47.8126	0.5228
11	44.6783	44.6814	0.1452	44.7043	0.1574	44.6290	0.1930
12	44.0043	44.0056	0.0565	44.0271	0.0686	43.9596	0.1301
13	49.3371	49.5421	1.0835	49.5603	1.0814	49.2088	0.5546
14	49.1419	49.3029	1.0651	49.3169	1.0616	48.9583	0.6168
15	49.0994	49.2563	1.0576	49.2698	1.0541	48.9131	0.6175
16	49.3563	49.5702	1.0824	49.5900	1.0805	49.2422	0.5369
17	47.9938	48.1156	0.8156	48.1294	0.8141	47.8566	0.4793
18	49.3736	49.5980	1.0817	49.6149	1.0804	49.2728	0.5213
19	49.0700	49.2514	0.8981	49.2762	0.9069	48.9854	0.5082
20	46.6476	46.7040	0.4468	46.7566	0.4530	46.6181	0.4182
21	45.6441	45.6825	0.2950	45.7180	0.3069	45.5983	0.2866
22	46.6183	46.6792	0.4385	46.7255	0.4480	46.5794	0.3936
23	48.4099	48.5566	0.7486	48.5869	0.7649	48.3352	0.4772
24	43.1961	43.1332	0.1117	43.1468	0.1178	43.1066	0.1753
25	42.4415	42.2816	0.2864	42.2931	0.2902	42.2661	0.3257
26	32.1117	32.6021	1.7594	32.6035	1.7515	32.0839	0.6300
27	-15.1991	-14.9994	0.9501	-14.9739	0.9568	-15.2686	0.5679
28	-33.4978	-35.1219	3.0669	-33.4718	0.6839	-33.4716	0.6840
29	31.7221	32.1481	1.6298	32.1521	1.6229	31.6796	0.6221
30	43.5819	43.5820	0.0000	43.6007	0.0229	43.5372	0.1659
31	44.1703	44.3208	0.5354	44.3604	0.5494	44.2578	0.5855
32	-46.3812	-46.4228	0.9293	-46.3792	0.9382	-46.4679	0.9750
33	-36.5478	-36.1842	2.2204	-36.1941	2.2091	-36.6956	0.7564
34	-12.1963	-11.6190	2.1379	-11.6198	2.1279	-12.2956	0.3880
35	0.0723	0.0729	0.0022	0.0729	0.0022	0.0728	0.0023
36	0.0927	0.0908	0.0025	0.0926	0.0015	0.0928	0.0015
37	-0.0229	-0.0236	0.0012	-0.0236	0.0012	-0.0240	0.0014
38	-0.0518	-0.0516	0.0018	-0.0517	0.0018	-0.0513	0.0021
39	-0.0391	-0.0396	0.0015	-0.0396	0.0015	-0.0394	0.0016
40	0.0254	0.0251	0.0013	0.0251	0.0013	0.0253	0.0013
41	0.0614	0.0612	0.0020	0.0612	0.0020	0.0611	0.0022
42	0.1063	0.1061	0.0028	0.1061	0.0028	0.1048	0.0038

Table 5-1: 34-node water network state estimates and confidence limits

1-34: nodal heads (m Aq) at nodes 1-34;
35-42: fixed-head nodes in/out flows (m^3/s) at nodes 27-34
All the results have been obtained for the following parameters: variability of consumptions - ±10 %, accuracy of head measurements - ±2 %, coefficient γ of Newton-Raphson method - 0.6, integration time constant - 10e-8 [s]

Bearing in mind that the confidence limit analysis for water distribution systems can be carried out for the linearised model without significant loss in accuracy of obtained results the ANN for solving overdetermined systems of linear equations has been used for this purpose. First, however, it has been shown that the sensitivity matrix method resulted from the sensitivity analysis of the ordinary LS solution to the overdetermined system of linear equations and it should not be used when the estimates were obtained using other optimization criteria. It is, therefore, advised that first the more robust criteria should be used to identify and reject gross errors and only then the full estimation and confidence limit analysis using sensitivity matrix method could be carried out for the remaining gross errors free data.

Assuming now that the data is free from gross errors two ANN based methods for CLA have been proposed. The first is a neural implementation of the sensitivity matrix method and the second one utilizes the linearity of the system and is based on the superposition principle.

Since ultimately the neural network structures presented in this work should be implemented in hardware (e.g. using VLSI or electro-optical technology) a suitable integrated, dynamical, neural based system for state estimation and CLA has been developed and simulated. A major conclusion that can be drawn from these simulations is that using relatively simple neural network solving systems of linear equations it is possible to significantly reduce the time of obtaining results of state estimation and CLA in such complex and nonlinear systems as water distribution networks. It must be emphasized once again that providing hardware implementation of this neural structure the computation could be accomplished in order of microseconds, regardless of system size, compared to seconds or minutes required by standard procedures executed on serial computers.

Chapter 6
Neural networks for classification and clustering

6.1. Introduction

In previous chapters the ANN based state estimation and CLA algorithms for water distribution networks were presented. These two algorithms are the first two steps on the way from measurement readings to the operational control decisions. Before any control decision can be made the state of the network has to be interpreted - classified. This interpretation task is usually carried out by an experienced, human operator. However, the growing size and complexity of modern water distribution systems makes this task more and more difficult. The need for the "diagnosis" of a water network state (i.e. normal operating state, leakage between node *i* and node *j* etc.) has prompted our investigation into classification and clustering neural networks.

The main thesis of the second part of this research is that the fuzzy state clustering and classification performed by neural networks mimics to a large extent the high level information processing by human operators.

6.2. Pattern recognition - general remarks

A lot of scientific effort has been dedicated to pattern recognition problems. The reason for this is the fact that pattern recognition is a key element to many engineering solutions. Different medical, image processing, control, and diagnostic applications, to mention only a few, all require the ability to accurately classify a situation.

A field of pattern recognition strongly follows recent achievements of different areas of science and engineering incorporating them for formation of more efficient and reliable classification algorithms. On the other hand, a rapid development of technology enabled building of more efficient recognition systems, capable of processing huge volumes of data such as, for example, in satellite image processing or vision channels for intelligent robots.

Despite tremendous achievements obtained, pattern recognition is confronted with continuous challenge coming from human beings. Humans seem to be more efficient in solving many complex classification tasks which still cannot be handled easily by a computer.

It should be mentioned that human reasoning is somewhat fuzzy in nature. Fuzzy sets can be seen as a suitable tool able to cope with the uncertain or ambiguous data so often encountered in real life and usually used by man. Hence, to enable the system to deal with the ambiguous data in an effective manner, one may incorporate the concept of fuzzy sets into the neural network.

6.2.1. A general model for pattern recognition problem

All pattern recognition problems can be partially or completely described in terms of the goal of classifying some set of measurements into categories. In a general case the pattern recognition process can be abstracted as a sequence of three steps: data acquisition, feature extraction and classification procedure.

At the first step data to be classified is collected via a set of sensors. The other two steps ultimately have the same goal, namely to transform the input into a form that is trivially convertible into a class decision. Traditionally the feature extraction is used to convert the raw input into a form that is easily classified. This is a common place to incorporate domain-specific knowledge that will greatly enhance performance over the blind use of automatic techniques. For example, for water distribution systems, in the context of pattern recognition problem, the state estimation procedure can be viewed as a robust method used to transform raw measurements into a vector of state estimates (feature vector) uniquely describing the system. Finally at the third stage of the scheme, the classifier is constructed. Trained classifiers are developed using a set of feature vectors. Typically the training develops a set of discriminant functions, one for each class, which are optimized in an attempt to produce the largest value for the function corresponding to a correct class. In other words a transformation between classes and feature vectors is established. This transformation could be, for instance, a Bayesian rule of computation an *a posteriori* class probability, nearest-neighbour rule, nearest prototype classifier, etc. One of more recent methods of implementing the discriminant functions are the systems combining neural networks with fuzzy sets. These methods will be described in more detail in later sections of this chapter.

6.2.2. Desirable properties of pattern classifier

The ultimate product of this part of research is a classification system. Before embarking on description of existing methods of constructing pattern classifiers, let us quote a list of properties (Simpson, 1992) that a good pattern classifier should possess. These are:

- **On-Line Adaptation**

A pattern classifier should be able to learn new classes and refine existing classes quickly and without destroying old class information.This property is sometimes referred to as on-line adaptation or on-line learning.

- **Nonlinear Separability**

A pattern classifier should be able to build decision regions that separate classes of any shape and size.

- **Overlapping Classes**

In addition to pattern classes being nonlinearly separable, they also tend to overlap. A pattern classifier should have the ability to form a decision boundary that minimizes the amount of misclassification for all of the overlapping classes. The most popular method of minimizing misclassification is the construction of a Bayesian classifier. Unfortunately, to build a Bayesian classifier requires knowledge of the underlying probability density function for each class. This information is quite often unavailable.

- **Training Time**

A very desirable property of a pattern classification approach, able to learn nonlinear decision boundaries, is a short training time.

- **Soft and Hard Decisions**

A pattern classifier should be able to provide both soft and hard classification decisions. A hard, or crisp, decision 0 or 1. A pattern is either in a class or it is not.A soft decision provides a value that describes the degree to which a pattern fits within a class.

- **Verification and Validation**

It is important that a classifier, neural or traditional, have a mechanism for verifying and validating its performance in some way.

- **Tuning Parameters**

A classifier should have as few parameters to tune in the system as possible. Ideally, a classifier system will have no parameters that need to be tuned during training. If there are parameters, the effect these parameters have on the system should be well understood.

- **Nonparametric Classification**

Parametric classifiers assume a priori knowledge about the underlying probability density functions of each class. If this information is available, it is possible to construct very reliable pattern classifiers, but often this information is not available. If the classifier is nonparametric, it should be able to describe the underlying distribution of the data in a way that provides reliable class boundaries.

6.2.3. Clustering and classification - standard and neural algorithms

There are two possible training strategies while constructing pattern classification procedures: supervised and unsupervised learning.

In supervised learning (also called supervised classification or pattern classification), class labels are provided with pattern exemplars and the decision boundary between classes that minimizes misclassification is sought.

In unsupervised learning (also called unsupervised classification, cluster analysis, or pattern clustering), the training pattern data is unlabelled and one has to deal with the task of splitting a set of patterns into a number of more or less homogenous clusters with respect to a suitable similarity measure in such a sense that the patterns which are similar are allocated to the same cluster, while the patterns which differ significantly are put in different clusters. Apart from the clustering method applied the final result is always a partition of patterns in disconnected or overlapped clusters.

There are many different techniques that have been offered for solving classification and clustering problems. In the following two sections a brief review of some traditional, fuzzy and neural network classification and clustering techniques respectively is given.

6.2.3.1. Classification techniques

1) ***Statistical Classification - Bayesian classifier:*** Statistical approach to pattern classification is based on the assumption that patterns are drawn from a mixed population of *c* statistical distributions, that have known prior probabilities and class-conditional probability density functions (PDFs). Given prior probabilities and PDFs Bayes rule allows us to compute an *a posteriori* probability *p(i/x)* which is essentially the probability of pattern *x* belonging to class *i*. The statistical methods of pattern classifications are well covered in the literature and good treatments of the subject can be found in (Duda & Hart, 1973; Devijver & Kittler, 1982). It was demonstrated on many occasions that the existence of situations where complete knowledge of the statistical model describing the pattern-generating mechanism enables the designer to determine both the optimal structure and the performance of the pattern classifier. However, in practical applications, such a perfect knowledge of the underlying distributions is practically never available. The derivation of the optimal solution to a classification problem will, as a rule, be beyond the reach of the practitioners. They will usually have either to satisfy themselves with an approximation to the optimal solution, or to resort to suboptimal classification methods such as those briefly discussed below.

2) ***k - Nearest Neighbour Classifiers:*** By contrast to the statistical classification the k-nn rule exchanges the need to know the underlying distributions for that of knowing a large number of correctly labelled sample patterns from each class. In principle, this shift in viewpoint brings us a lot closer to the reality of practical problems. The mode of operation can be described as follows: Given an unlabelled input pattern *x*, and a large set of correctly labelled patterns *S*, find the k nearest neighbours to *x* from *S* and assign *x* to the class which is most heavily represented in the k-nn neighbourhood. Further discussion of the k-nn rule and its derivatives can be found in (Dasarathy, 1990; Devijver & Kittler, 1982; Duda & Hart, 1973).

3) ***Discriminant Functions:*** While the previous two approaches did not presuppose any particular "shape" for the decision boundary in case of discriminant functions the functional form of the decision boundary is selected a priori. In other words, the analytical expression of the decision boundary is given except for the values of a set of parameters. This set of parameters is estimated on the basis of training set to produce a problem specific classifier. For numeric-valued patterns, the discriminants are based on

one of two concepts. In one case, the discriminant consists of a measure of distance, and patterns are classified in accordance with the class membership of nearest neighbours, or with the nearest prototype or cluster center. In the other case, the discriminants are hypersurfaces, and patterns are classified in accordance whether they are on one side or another of a hypersurfaces or a set of hyperplanes.

4) ***Neural Network Classifiers:*** When neural network is operating as a classifier net, it can be regarded simply as a box, or algorithmic representation of a classifier function. A number of different learning rules, network architectures and class representations have been used in order to produce effective mapping from feature space to class space. Some neural networks, such as probabilistic neural network (Specht, 1990), learning vector quantization (Kohonen, 1984), or radial basis functions (Hassoun, 1995), make use of the statistical mean and variance of data clusters to define center and size of pattern classes. Some neural networks, such as perceptron (Pao, 1989), create linear discriminant functions that partition the pattern space. Some neural networks, such as back-propagation, the Boltzman machine, optimize a cost function that gives an input-output mapping that in turn produces nonlinear decision boundaries. And finally, some neural networks, such as the related coulomb energy neural network, and the min-max classification neural network (Simpson, 1992), build decision boundaries by creating subsets of the pattern space.

5) ***Fuzzy Classification:*** Inability of the traditional (statistical) approaches for pattern classification to deal with problems where a process of "hard" labelling is difficult to perform or completely artificial, called for more flexible way of labelling and accounting for certain types of uncertainty. The theory of fuzzy sets which allows for fuzzy membership functions was suggested as a way to remedy this difficulty. The earliest reference to the use of fuzzy sets in pattern recognition, with a general description of the classification process indicating a role of fuzzy sets, was (Bellman et al., 1966). Since then the combination of fuzzy sets and pattern classification has been studied by many people. The fuzzy k-nn classifier was developed (Bezdek et al., 1986) and shown to be a generalised version of the standard k-nn classifiers. Another fuzzy generalization of the standard classification technique is fuzzy c-means algorithm (Bezdek, 1981). Keller and Hunt first attempted to incorporate the idea of membership functions into the classical perceptron algorithm (Keller & Hunt, 1985). Other works on fuzzy classification include: (Pedrycz, 1990) discussing a state-of-the-art (as of then) methodology and algorithms of

fuzzy sets in the field of pattern recognition, (Bezdek, 1992) is an excellent discussion on interaction between statistical, fuzzy, and neural-like models for pattern recognition system design, in (Bezdek, 1981) Bezdek studied the application of fuzzy objective functions for pattern recognition. The aforementioned flexibility of fuzzy sets and the computational efficiency of neural networks with their proved record in pattern recognition problems has caused a great amount of interest in the combination of the two (Bezdek, 1992; Mitra & Pal, 1994; Pedrycz, 1992; Simpson, 1992; Yager & Zadeh, 1994). Many of the efforts have focused on methods of implementing fuzzy rules in a neural network framework and techniques of parallelizing successful fuzzy system applications.

6.2.3.2. Clustering techniques

1) ***Traditional Clustering:*** There are many clustering algorithms that have been developed to date, including ISODATA, FORGY, WISH, and CLUSTER (Dubes & Jain, 1976), many of which are commercially sold. Jain (Jain, 1986) has reduced these clustering techniques to two popular methods:

- *Hierarchical Clustering:* A hierarchical clustering technique imposes a hierarchical structure on the data which consists of a sequence of clusters.
- *Partitional Clustering:* A partitional clustering technique organizes patterns into a small number of clusters by labelling each pattern in some way. Unlike hierarchical clustering, which offers several partitions of the data, partitional clustering finds a single cluster partition.

In addition to the two techniques cited above, there are also combinations of the two clustering approaches that are employed. There are many books that describe classical approaches to pattern clustering, including (Anderberg, 1973; Everitt, 1974; Hartigan, 1975; and Duda & Hart, 1973)

2) ***Fuzzy Clustering:*** Fuzzy sets bring a new dimension to traditional clustering systems by allowing a pattern to belong to multiple clusters to different degrees. Bezdek has organized fuzzy clustering algorithms into five categories:

- *Relation Criterion Functions:* Clustering driven by optimization of criterion function which assesses partitions according to some global property of the grouped data. Ruspini (Ruspini, 1969) was the first to utilize this technique in the fuzzy community and he and Bezdek (Bezdek, 1981) have since considerably extended this pioneering work.

•*Object Criterion Functions:* Clustering directly on the data set A in the n-dimensional feature space according to some objective function is the most popular form of the fuzzy pattern clustering. The fuzzy c-means and fuzzy ISODATA algorithms introduced by Dunn (Dunn, 1974) and generalised by Bezdek (Bezdek, 1981), are the most popular techniques for this class of fuzzy clustering algorithms.

•*Convex Decomposition:* The decomposition of a fuzzy partition (a set of fuzzy clusters) into a combination of convex sets. The use of the convex decompositions may provide added insight into data structure that otherwise might be lost. Bezdek & Harris (Bezdek & Harris, 1979) describe three algorithms that can perform this decomposition.

•*Numerical Transitive Closures:* The extraction of crisp equivalence relations from fuzzy transitive similarity relations. This technique is closely related to hierarchical methods based on graph-theoretic models.

• *Generalised Nearest Neighbour Rules:* Although the nearest neighbour algorithm is used mostly for classification, there is a clustering version as well. This technique is primarily used once the data set has already been partitioned using another clustering algorithm such as fuzzy c-means.

3) ***Neural Network Clustering:*** Neural network clustering offers the ability to determine the size, shape, number, and placement of pattern clusters adaptively while intrinsically operating in parallel. In addition, the use of clustering to form sensory maps has a strong biological support. Although there is a large number of neural networks available today there are only two primary neural clustering techniques currently in widespread use:

•*Competitive Learning:* Similar to the c-means clustering algorithm, competitive learning finds the centroids of decisions regions in the n-dimensional pattern space. Although this form of neural network learning seems to have been introduced by Grossberg (Grossberg, 1972, Grossberg, 1976a) and von der Malsburg (von der Malsburg, 1973), it has been most successfully championed by Kohonen (Kohonen, 1984), who has extended the neural dynamics to include topographic constraints.

•*Adaptive Resonance Theory:* Similar to the leader cluster algorithm, adaptive resonance theory nondestructively creates pattern “codes” (clusters). The concept of adaptive resonance was introduced by Grossberg (Grossberg, 1976b) and was first cast into a neural network formalism by Carpenter and Grossberg (Carpenter & Grossberg, 1987). There have been numerous extensions and refinements since (Carpenter & Grossberg, 1987). The most recent results of ART evolution are the algorithms combining ideas of ART and fuzzy logic.

6.2.3.3. Concluding remarks

From the above brief descriptions of the existing classification and clustering techniques it is not difficult to notice that the focus of our investigations will be on fuzzy neural methods. The large scale of the problem, the fuzzy nature of the input patterns (state estimates with confidence limits), the need for the accommodation of labelled and unlabelled patterns called for a system that could grow to accommodate the needs of the problem, has potential to combine supervised and unsupervised learning strategies within a consistent, single frame and provides flexibility of modelling and processing uncertainty associated with fuzzy systems. Fuzzy neural networks for classification and clustering developed from the concept of adaptive resonance theory seem to have a great potential of fulfilling all of those requirements.

Evolution of ART NNs has led to developing various fuzzy neural networks by different, independent researchers. The first two examples of such networks are Fuzzy ART (Carpenter & Grossberg, 1994) and Fuzzy ARTMAP (Asfour et al., 1993; Carpenter et al., 1992; Carpenter & Grossberg, 1994) developed by Carpenter and Grossberg, the original inventors of the ART neural networks. Fuzzy ART is a generalisation of ART1 (Carpenter & Grossberg, 1987) that incorporates operations from fuzzy logic. It is unsupervised network that can learn to classify both analog and binary input patterns. The categories formed by Fuzzy ART are hyperboxes. Fuzzy ARTMAP is the supervised neural network built from two Fuzzy ART modules (ARTa and ARTb). During supervised learning ARTa receives a stream of input patterns and ARTb receives a stream of the correct predictions. These modules (ARTa and ARTb) are linked by an associative learning network and an internal controller that ensures autonomous system operation. Although, a number of successful applications have been reported, Fuzzy ARTMAP has unnecessarily complicated control structure resulting from combination of two clustering networks. Moreover, it allows hyperbox clusters to overlap, which in turn results in pattern cluster ambiguity - a pattern can have full membership in more than one cluster.

Another very interesting development influenced by the original ART neural network is adaptive fuzzy leader clustering (AFLC) neural network (Kim & Mitra, 1993; Newton & Mitra, 1992; Newton et al., 1992; Pemmaraju & Mitra, 1993). AFLC is a hybrid neural-fuzzy system which can be used to learn cluster structure in a self-organizing, stable manner. The architecture consists of a learning rule in the form of the fuzzy K-means clustering algorithm embedded in a control structure similar to that found in ART-1. AFLC is primarily used as a classifier of feature vectors employing an on-line learning scheme. It utilizes a fuzzy membership function to describe the degree to which the input pattern belongs to each cluster. A single point - centroid of the cluster - is used to represent each cluster. The membership function is effectively an Euclidean distance from a cluster prototype (centroid). Adapting the clusters is based on the attempt to optimally position a cluster prototype. Although the AFLC NN has some interesting properties there are two main disadvantages: a) the improper cluster representation (centroid) for the type of data we need to classify and b) each adjusting of centroid requires all the data belonging to the cluster.

The third and the most compelling alternative are the min-max clustering and classification neural networks developed by Simpson (Simpson, 1992; Simpson, 1993). These neural networks seem to be especially interesting because of their representation of classes (clusters) which is a hyperbox in n-dimensional pattern space. A hyperbox is completely defined by pairs of min-max points. By analogy the state of the water network after the confidence limit analysis can be viewed as a hyperbox in n-dimensional space defined by upper and lower bounds for each state variable. Some other properties like on-line learning, the number of clusters (classes) that grows to meet the demands of the problem, the potential to combine the clustering and classification within one network are equally appealing and worth further investigation.

6.3. The Fuzzy Min-Max Clustering and Classification Neural Networks

The fuzzy min-max clustering and classification neural networks are built using hyperbox fuzzy sets. A hyperbox defines a region of the n-dimensional pattern space, and all patterns contained within the hyperbox have full cluster/class membership. A hyperbox is completely defined by its min point and its max point. The combination of the min-max points and the hyperbox membership function defines a fuzzy set (cluster). In the case of classification hyperbox fuzzy sets are aggregated to form a single fuzzy set class.

Learning in the fuzzy min-max clustering and classification neural networks consists of creating and adjusting hyperboxes in pattern space as they are received. It is an expansion/contraction process.The learning process begins by selecting an input pattern and finding the closest hyperbox to that pattern that can expand (if necessary) to include the pattern. If a hyperbox cannot be found that meets the expansion criteria, a new hyperbox is formed and added to the system. This growth process allows existing clusters/classes to be refined over time, and it allows new clusters/classes to be added without retraining. One of the residuals of hyperbox expansion is overlapping hyperboxes. Hyperbox overlap causes ambiguity.It is reasonable to assume that a pattern can have the same partial membership in more than one cluster/class. It is not reasonable to assume that a pattern can completely belong to more than one cluster/class. In the case of classifying NN the overlap is eliminated for hyperboxes that represent different classes. A contraction process is utilized to eliminate any undesired hyperbox overlaps.

In summary, the fuzzy min-max clustering and classification learning algorithm is a four-step process:

1) *Initialization:* Initialize all the min-max points prior to any learning.

2) *Expansion:* Identify the hyperbox closest to the input pattern that can be expanded and expand it. If an expandible hyperbox cannot be found, add a new hyperbox.

3) *Overlap Test:* Determine whether the recent expansion caused any undesired overlap between hyperboxes.

4) *Contraction:* If the overlap test identified overlapping hyperboxes, contract the hyperboxes to eliminate overlap.

Having implemented and tested these NNs a new fuzzy neural algorithm based on the concept of the Fuzzy Min-Max Classification and Clustering NNs has been developed (Gabrys & Bargiela, 1997a).

This new NN combines the functionality of both the Fuzzy Min-Max Classification and Clustering NNs and at the same time a few major changes have been made to accommodate the input in a form of the state vector with confidence limits and improve the effectiveness of the algorithm. We will refer to this new algorithm as a Generalised Fuzzy Min-Max (GFMM).

The remainder of this chapter is organised as follows. In Table 6-1 the detailed description and comparison of the original fuzzy min-max and GFMM algorithms is given. The discussion and reasoning behind changes in GFMM in comparison to the original Fuzzy Min-Max NNs is given in Section 6.4. The neural network implementation of the GFMM algorithm is given in Section 6.5. In Section 6.6 the examples are presented. And finally the last Section is assigned for discussion and conclusions.

Fuzzy Min-Max Clustering	Fuzzy Min-Max Classification	GFMM clustering-classifying algorithm for fuzzy inputs
1) Initialization: $V_j = \mathbf{1}$ - min points $W_j = \mathbf{0}$ - max points	**1) Initialization:** $V_j = \mathbf{1}$ - min points $W_j = \mathbf{0}$ - max points	**1) Initialization:** $V_j = \mathbf{1}$ - min points $W_j = \mathbf{0}$ - max points
2) Input: $A_h = [a_{h1}, a_{h2}, \ldots, a_{hn}] \in I^n$ - n-dimensional input vector (pattern)	**2) Input:** $\{A_h, d_h\}$ - ordered pair A_h - n-dimensional input vector $d_h \in \{1, 2, \ldots, m\}$ - the index of one of the m classes	**2) Input:** $\{X_h, d_h\}$ - ordered pair $X_h = \begin{bmatrix} X_h^l & X_h^u \end{bmatrix}$ - the h-th input pattern in a form of lower and upper limits vectors $d_h \in \{0, 1, 2, \ldots, m\}$ - the index of one of the $m+1$ classes, where $d_h = 0$ means that the input vector is unlabelled.
3) Fuzzy hyperbox Membership Function $b_j(A_h) = \frac{1}{n}\sum_{i=1}^{n}[1-f(a_{hi}-w_{ji}, \gamma)-f(v_{ji}-a_{hi}, \gamma)]$ where: $f(x,\gamma) = \begin{cases} 1 & if \quad x\gamma > 1 \\ x\gamma & if \quad 0 \le x\gamma \le 1 \\ 0 & if \quad x\gamma < 0 \end{cases}$ - two parameter ramp threshold function V_j - min point of the j -th hyperbox W_j - max point of the j - th hyperbox γ - sensitivity parameter	**3) Fuzzy hyperbox Membership Function** $b_j(A_h) = \frac{1}{2n}\sum_{i=1}^{n}[max(0, 1-max(0, \gamma min(1, a_{hi}-w_{ji})))$ $+max(0, 1-max(0, \gamma min(1, v_{ji}-a_{hi})))]$ V_j - min point of the j -th hyperbox W_j - max point of the j - th hyperbox γ - sensitivity parameter	**3) Fuzzy hyperbox Membership Function** $b_j(X_h) = \min_{i=1..n}(min([1-f(x_{hi}^u-w_{ji}, \gamma_i)], [1-f(v_{ji}-x_{hi}^l, \gamma_i)]))$ where: $f(x,\gamma) = \begin{cases} 1 & if \quad x\gamma > 1 \\ x\gamma & if \quad 0 \le x\gamma \le 1 \\ 0 & if \quad x\gamma < 0 \end{cases}$ - two parameter ramp threshold function V_j - min point of the j -th hyperbox W_j - max point of the j - th hyperbox $\gamma = [\gamma_1, \gamma_2, \ldots, \gamma_n]$ - sensitivity parameters

Table 6-1: Comparison of fuzzy min-max and GFMM algorithms

Fuzzy Min-Max Clustering	Fuzzy Min-Max Classification	GFMM clustering-classifying algorithm for fuzzy inputs
4) Hyperbox Expansion Find the hyperbox $B_j \in C$ that provides the highest degree of membership and allows expansion (if needed). $\sum_{i=1}^{n} (max(w_{ji}, a_{hi}) - min(v_{ji}, a_{hi})) \leq n\Theta$ If this constraint is satisfied **adjust** min and max points using the equations: $v_{ji}^{new} = min(v_{ji}^{old}, a_{hi})$ for each i=1,..,n $w_{ji}^{new} = min(w_{ji}^{old}, a_{hi})$ for each i=1,..,n If all committed hyperboxes are exhausted, then select uncommitted one and apply adjusting.	**4) Hyperbox Expansion** Find the hyperbox $B_j \in C$ that provides the highest degree of membership and allows expansion (if needed). $\sum_{i=1}^{n} (max(w_{ji}, a_{hi}) - min(v_{ji}, a_{hi})) \leq n\Theta$ and class(B_j)=d_h If this constraint is satisfied **adjust** min and max points using the equations: $v_{ji}^{new} = min(v_{ji}^{old}, a_{hi})$ for each i=1,..,n $w_{ji}^{new} = min(w_{ji}^{old}, a_{hi})$ for each i=1,..,n If all committed hyperboxes are exhausted, then select uncommitted one, apply adjusting and set $class(B_j)=d_h$.	**4) Hyperbox Expansion** Find the hyperbox $B_j \in C$ that provides the highest degree of membership and allows expansion (if needed). $\forall_{i=1...n} (max(w_{ji}, x_{hi}^{u}) - min(v_{ji}, x_{hi}^{l})) \leq \Theta$ and $if\ d_h = 0\ then\ adjust\ B_j$ $else\ if\ class(B_j) = \begin{cases} 0 \Rightarrow & adjust\ B_j \\ & class(B_j) = d_h \\ d_h \Rightarrow & adjust\ B_j \\ else \Rightarrow & take\ another\ B_j \end{cases}$ Adjust B_j operation: $v_{ji}^{new} = min(v_{ji}^{old}, x_{hi}^{l})$ for each i=1,..,n $w_{ji}^{new} = min(w_{ji}^{old}, x_{hi}^{u})$ for each i=1,..,n If all hyperboxes from committed set $B_j \in C$ are exhausted without possible expansion, then select uncommitted one $B_k \in U$, adjust and set $class(B_k) = d_h$.

Table 6-1: Comparison of fuzzy min-max and GFMM algorithms

Fuzzy Min-Max Clustering	Fuzzy Min-Max Classification	GFMM clustering-classifying algorithm for fuzzy inputs
5) Hyperbox Overlap Test The expansion creates an overlap between B_j and B_k if one of the four cases is satisfied for each of the n dimensions: Case 1: $v_{ji} < v_{ki} < w_{ji} < w_{ki}$ Case 2: $v_{ki} < v_{ji} < w_{ki} < w_{ji}$ Case 3: $v_{ji} < v_{ki} \le w_{ki} < w_{ji}$ Case 4: $v_{ki} < v_{ji} \le w_{ji} < w_{ki}$	**5) Hyperbox Overlap Test** Assuming that hyperbox B_j was expanded in previous step test for overlapping with B_k if $class(B_j) \neq class(B_k)$. While testing for the overlap the smallest overlap along any dimension and the index of the dimension is saved for use during contraction portion of the algorithm. Initially $\delta^{old} = 1$. Case 1: $v_{ji} < v_{ki} < w_{ji} < w_{ki}$ $\delta^{new} = min(w_{ji} - v_{ki}, \delta^{old})$ Case 2: $v_{ki} < v_{ji} < w_{ki} < w_{ji}$ $\delta^{new} = min(w_{ki} - v_{ji}, \delta^{old})$ Case 3: $v_{ji} < v_{ki} \le w_{ki} < w_{ji}$ $\delta^{new} = min(min(w_{ki} - v_{ji}, w_{ji} - v_{ki}), \delta^{old})$ Case 4: $v_{ki} < v_{ji} \le w_{ji} < w_{ki}$ $\delta^{new} = min(min(w_{ki} - v_{ji}, w_{ji} - v_{ki}), \delta^{old})$ If $\delta^{old} - \delta^{new} > 0$ then $\Delta = i$ and $\delta^{old} = \delta^{new}$, signifying that there was overlap for the $\Delta - th$ dimension and overlap testing will proceed with the next dimension. If not, the testing stops and the minimum overlap index variable is set to indicate that the next contraction step is not necessary i.e. $\Delta = -1$.	**5) Hyperbox Overlap Test** Assuming that hyperbox B_j was expanded in previous step test for overlapping with B_k if $class(B_j) = \begin{cases} 0 \Rightarrow & \text{test for overlapping with all the other hyperboxes} \\ else \Rightarrow & \text{test for overlaping only if } class(B_j) \neq class(B_k) \end{cases}$ The smallest overlap along any dimension, the index of the dimension and index of the case are saved. The four cases as in the Min-Max classification algorithm are considered. If $\delta^{old} - \delta^{new} > 0$ then $\Delta = i$, $\delta^{old} = \delta^{new}$ and $case = l$ ($l = \{1,2,3,4\}$ - the case for which the smallest overlap was found). If not, $\Delta = -1$ - next contraction step is not necessary.

Table 6-1: Comparison of fuzzy min-max and GFMM algorithms

Fuzzy Min-Max Clustering	Fuzzy Min-Max Classification	GFMM clustering-classifying algorithm for fuzzy inputs
6) Hyperbox Contraction If the hyperboxes B_j and B_k are overlapping the overlap is eliminated on a dimension-by-dimension basis: Case 1: $v_{ji} < v_{ki} < w_{ji} < w_{ki}$ $v_{ki}^{new} = w_{ji}^{new} = \frac{v_{ki}^{old} + w_{ji}^{old}}{2}$ Case 2: $v_{ki} < v_{ji} < w_{ki} < w_{ji}$ $v_{ji}^{new} = w_{ki}^{new} = \frac{v_{ji}^{old} + w_{ki}^{old}}{2}$ Case 3: $v_{ji} < v_{ki} \le w_{ki} < w_{ji}$ if $w_{ki} - v_{ji} < w_{ji} - v_{ki}$ then $v_{ji}^{new} = w_{ki}^{old}$ otherwise $w_{ji}^{new} = v_{ki}^{old}$ Case 4: $v_{ki} < v_{ji} \le w_{ji} < w_{ki}$ if $w_{ki} - v_{ji} < w_{ji} - v_{ki}$ then $w_{ki}^{new} = v_{ji}^{old}$ otherwise $v_{ki}^{new} = w_{ji}^{old}$	**6) Hyperbox Contraction** If $\Delta > 0$ then only the $\Delta - th$ dimensions of the two hyperboxes are adjusted. To determine the proper adjustment to make the same four cases are examined: Case 1: $v_{j\Delta} < v_{k\Delta} < w_{j\Delta} < w_{k\Delta}$ $v_{k\Delta}^{new} = w_{j\Delta}^{new} = \frac{v_{k\Delta}^{old} + w_{j\Delta}^{old}}{2}$ Case 2: $v_{k\Delta} < v_{j\Delta} < w_{k\Delta} < w_{j\Delta}$ $v_{j\Delta}^{new} = w_{k\Delta}^{new} = \frac{v_{j\Delta}^{old} + w_{k\Delta}^{old}}{2}$ Case 3: $v_{j\Delta} < v_{k\Delta} \le w_{k\Delta} < w_{j\Delta}$ if $w_{k\Delta} - v_{j\Delta} < w_{j\Delta} - v_{k\Delta}$ then $v_{j\Delta}^{new} = w_{k\Delta}^{old}$ otherwise $w_{j\Delta}^{new} = v_{k\Delta}^{old}$ Case 4: $v_{k\Delta} < v_{j\Delta} \le w_{j\Delta} < w_{k\Delta}$ if $w_{k\Delta} - v_{j\Delta} < w_{j\Delta} - v_{k\Delta}$ then $w_{k\Delta}^{new} = v_{j\Delta}^{old}$ otherwise $v_{k\Delta}^{new} = w_{j\Delta}^{old}$	**6) Hyperbox Contraction** If $\Delta > 0$ then only the $\Delta - th$ dimensions of the two hyperboxes are adjusted. Having saved the index of overlapping case in the previous step we can go to adjusting without examining the cases again. The adjusting process is the same as in Min-Max Classification algorithm. Case 1: $v_{j\Delta} < v_{k\Delta} < w_{j\Delta} < w_{k\Delta}$ $v_{k\Delta}^{new} = w_{j\Delta}^{new} = \frac{v_{k\Delta}^{old} + w_{j\Delta}^{old}}{2}$ Case 2: $v_{k\Delta} < v_{j\Delta} < w_{k\Delta} < w_{j\Delta}$ $v_{j\Delta}^{new} = w_{k\Delta}^{new} = \frac{v_{j\Delta}^{old} + w_{k\Delta}^{old}}{2}$ Case 3: $v_{j\Delta} < v_{k\Delta} \le w_{k\Delta} < w_{j\Delta}$ if $w_{k\Delta} - v_{j\Delta} < w_{j\Delta} - v_{k\Delta}$ then $v_{j\Delta}^{new} = w_{k\Delta}^{old}$ otherwise $w_{j\Delta}^{new} = v_{k\Delta}^{old}$ Case 4: $v_{k\Delta} < v_{j\Delta} \le w_{j\Delta} < w_{k\Delta}$ if $w_{k\Delta} - v_{j\Delta} < w_{j\Delta} - v_{k\Delta}$ then $w_{k\Delta}^{new} = v_{j\Delta}^{old}$ otherwise $v_{k\Delta}^{new} = w_{j\Delta}^{old}$

Table 6-1: Comparison of fuzzy min-max and GFMM algorithms

6.4. The discussion and reasoning behind changes in the GFMM classification/clustering algorithm.

All the differences between the original Fuzzy Min-Max NNs and our algorithm can be seen in Table 6-1 presented in Section 6.3. In the following subsections the origins of and reasons behind changes are explained in detail.

6.4.1. Input

As mentioned earlier, the developed algorithm is to be used for the classification of the states of the water distribution network. More specifically the classification procedure is to process the confidence limits for each state variable. This is achieved by specifying the input to classification/clustering algorithm as a pair of two vectors: $X_h = [X_h^l \; X_h^u]$ - the lower and upper limits for the state vector. In other words instead of a point in n-dimensional space that has to be classified we have a hyperbox with the min point determined by the vector X_h^l and the max point determined by the vector X_h^u. It can be observed, however, that when the min and max points are equal our hyperbox shrinks to the point. It is to say that our algorithm is capable of classification/clustering inputs in a form of the n-dimensional vector without any changes to the algorithm because a point in n-dimensional space is simply the special case of a hyperbox with the min and max points equal.

Because of the size of the modern water distribution networks it is impossible to predict and cover all possible combinations of consumption-inflow patterns and anomalies that can occur in the network during day-to-day operations. It is not difficult to imagine that some of the network states can be labelled (i.e. normal operating state etc.) and others cannot.

In order to allow labelled and unlabelled inputs to be processed an additional index, $d_h = 0$ meaning that the input pattern is not labelled, has been introduced. Using neural network phraseology we attempt to define hybrid, supervised (labelled inputs - classification) and unsupervised (unlabelled inputs - clustering), NN. This fusion of supervised and unsupervised NNs resulted in several changes in our algorithm in comparison to the original Fuzzy Min-Max NNs. These alterations are discussed below.

6.4.2. The differences between membership functions

The fuzzy hyperbox membership function plays the crucial role in the Fuzzy Min-Max Classification and Clustering algorithms.The decisions whether the presented input pattern belongs to the particular class or cluster, whether the particular hyperbox is to be expanded, depend mainly on the membership value describing the degree to which an input pattern fits within the hyperbox. Following (Simpson, 1992) let the j-th hyperbox fuzzy set, B_j, be defined by the ordered set

$$B_j = \{X_h, V_j, W_j, b_j(X_h, V_j, W_j)\} \qquad \text{(EQ 6-1)}$$

for all h=1,2,...,m, where $X_h = [X_h^l \; X_h^u]$ is the h-th input pattern, $V_j = (v_{j1}, v_{j2}, \ldots, v_{jn})$ is the min point for the j-th hyperbox, $W_j = (w_{j1}, w_{j2}, \ldots, w_{jn})$ is the max point for the j-th hyperbox, and the membership function for the i-th hyperbox is $0 \le b_j(X_h, V_j, W_j) \le 1$.

It is a natural assumption that the degree of membership of X_h for the hyperbox B_j is one if X_h is contained within the hyperbox B_j, and the degree of membership decreases as X_h moves away from the hyperbox B_j.

Unfortunately neither the membership function presented in (Simpson, 1992) (Figure 6-2) nor the membership function presented in (Simpson, 1993) (Figure 6-3) satisfy this assumption. The two dimensional example shows that even for patterns that are far from the hyperbox the membership values are large. It can also be observed that the membership values do not decrease steadily with increasing distance from the hyperbox.

To meet the required criteria a new membership function has been defined and is presented in Figure 6-4. The short interpretation of this function could be put in words as the minimum value of maximum min-max hyperbox points violations for all dimensions. The one dimensional membership function is shown in Figure 6-1.

It can be noticed that in the membership function from point c the sensitivity parameter $\gamma = [\gamma_1, \gamma_2, \ldots, \gamma_n]$ that regulates how fast the membership values decrease, is specified for each dimension. The reason why we introduced the sensitivity parameter for each dimension (in comparison to one γ for all dimensions in original algorithm) is the fact that our input patterns can consist of different physical quantities (i.e. node pressures, inflows) that may need to be tuned using different γ_i.

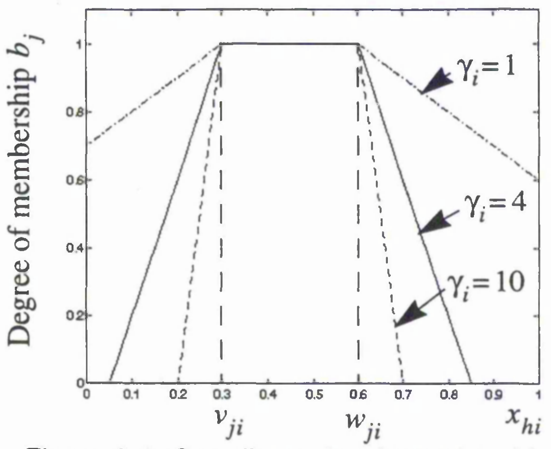


Figure 6-1: One dimensional membership function for the hyperbox B_j and i-th dimension. Examples for different γ.

a)

$$b_j(A_h) = \frac{1}{2n}\sum_{i=1}^{n} [max(0, 1 - max(0, \gamma min(1, a_{hi} - w_{ji}))) + +max(0, 1 - max(0, \gamma min(1, v_{ji} - a_{hi})))]$$

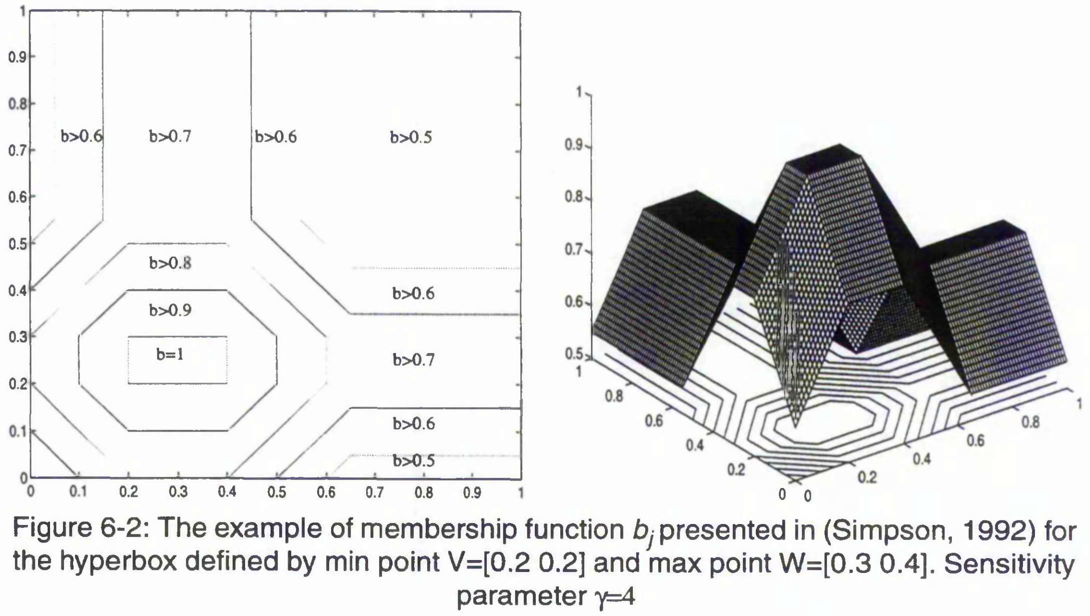


Figure 6-2: The example of membership function b_j presented in (Simpson, 1992) for the hyperbox defined by min point V=[0.2 0.2] and max point W=[0.3 0.4]. Sensitivity parameter γ=4

b) $$b_j(A_h) = \frac{1}{n}\sum_{i=1}^{n}[1 - f(a_{hi} - w_{ji}, \gamma) - f(v_{ji} - a_{hi}, \gamma)]$$

$$\text{where } f(x,\gamma) = \begin{cases} 1 & if \quad x\gamma > 1 \\ x\gamma & if \quad 0 \le x\gamma \le 1 \\ 0 & if \quad x\gamma < 0 \end{cases}$$

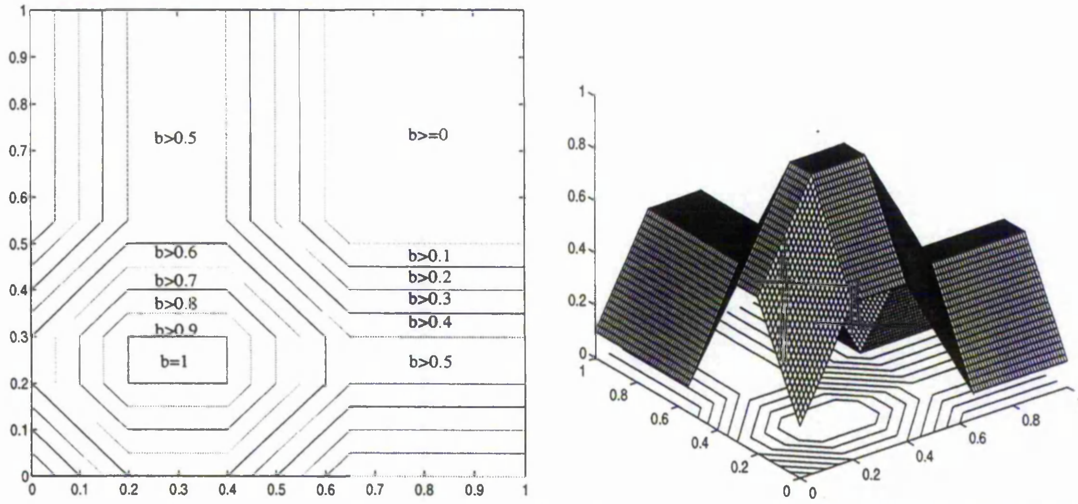


Figure 6-3: The example of membership function b_j presented in (Simpson, 1993) for the hyperbox defined by min point V=[0.2 0.2] and max point W=[0.3 0.4]. Sensitivity parameter γ=4

c) $$b_j(X_h) = \min_{i=1..N}(\min([1 - f(x^u_{hi} - w_{ji}, \gamma_i)], [1 - f(v_{ji} - x^l_{hi}, \gamma_i)]))$$

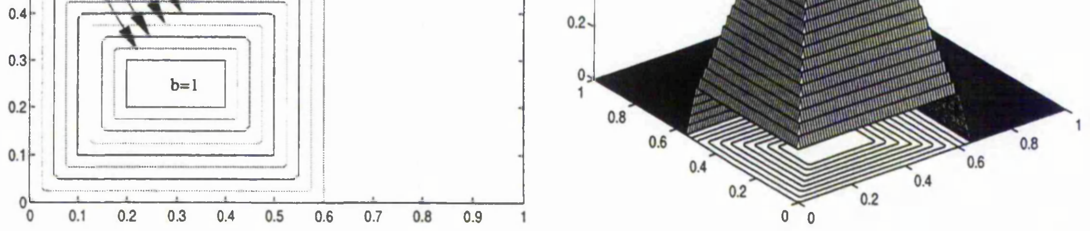


Figure 6-4: The two dimensional example of membership function b_j used in the GFMM classification/clustering algorithm. The hyperbox is defined by min point V=[0.2 0.2] and max point W=[0.3 0.4]. Sensitivity parameter γ=4.

6.4.3. Hyperbox expansion

The constraint regulating the maximum size of the hyperbox is slightly different in our algorithm than in original versions.

Not using the sum in the constraint allows us to control the size of the hyperbox for each dimension. We know that the difference between max and min value for each dimension will not be greater than the user specified value Θ, which cannot be said in case of using the original constraint.

Other differences in the expansion constraint result from admitting both labelled and unlabelled input patterns.

Assuming that the part of the hyperbox expansion constraint represented by the formula $\underset{i=1\ldots n}{\forall} (max(w_{ji}, x_{hi}^{u}) - min(v_{ji}, x_{hi}^{l})) \leq \Theta$ has been met we have to consider the following possibilities:

1) If the input pattern X_h is not labelled ($d_h = 0$) then the hyperbox B_j can be adjusted to include this pattern.

2) If the input pattern is labelled ($d_h \neq 0$) - belongs to the particular class specified by d_h - the three additional cases have to be considered:

a) If the hyperbox B_j is not a part of any of the existing classes ($class(B_j) = 0$) then adjust the hyperboxB_j to include the input pattern X_h and since this input is labelled as belonging to the class specified by d_h set $class(B_j) = d_h$.

b) If the hyperbox B_j is a part of the class specified by index d_h of the current input pattern X_h ($class(B_j) = d_h$) then adjust the hyperbox B_j.

c) If neither a nor b then take another hyperbox and test for possible expansion.

6.4.4. Hyperbox overlap test

Similarly to the hyperbox expansion part of the algorithm the difference between our algorithm and the Fuzzy Min-Max Classification NN arises from the admittance of labelled and unlabelled input patterns.

As a consequence we have to tackle the problem of overlapping hyperboxes not only being part of different classes (like in the Fuzzy Min-Max Classification NN) but also all these hyperboxes that are not labelled.

The resulting scheme could be described as follows.

1) If the hyperbox B_j, expanded in the last expansion step, is not labelled then test for overlapping with all the other hyperboxes. This ensures that all unlabelled hyperboxes do not overlap with any of the other existing ones.

2) If the hyperbox B_j, expanded in the last expansion step, belongs to one of the existing classes then test for the overlap only with the hyperboxes not being part of the same class as B_j. Notice that this allows the hyperboxes belonging to the same class to overlap.

6.4.5. Hyperbox contraction

There have been no changes in this part of algorithm in comparison to the Fuzzy Min-Max Classification NN, but few problems have been noticed and partially solved by introduction of an adaptive max size of the hyperbox. In other words the user defined parameter Θ is changed during the learning process. The problems that have just been mentioned will be discussed later during presentation of examples.

6.4.6. An adaptive maximum size of the hyperbox

In the original Fuzzy Min-Max NNs the user defined parameter Θ controlling the maximum size of created hyperboxes is set up at the beginning of learning process and stays the same all the time. To find the best value of this parameter the network has to be trained for several different Θs and verified by checking the number of misclassifications.

After testing the algorithm for different types of data we can say that setting the parameter Θ at the beginning and not changing it during the training of the network can have undesired effects on performance or the number of created hyperboxes. A large value of Θ can cause too many misclassifications, especially when there are complex, overlapping classes. On the other hand when Θ is small many unnecessary hyperboxes may be created, especially for concentrated, stand-alone groups of data which normally would form one class. But of course small Θ helps to resolve overlapping classes.

These problems have been addressed by introducing an adaptive maximum size of the hyperbox.

The idea is to start training the network with large Θ and decrease it (if necessary) in subsequent presentations of the data. In original versions of the algorithm the training stops after presenting the data once. Using the adaptive maximum size of the hyperbox requires defining the stopping condition - in other words when the training should be assumed to be completed.

Let us first consider the simplified case where input patterns are points in n-dimensional space. Assuming that there are not two identical points in the data labelled as belonging to two different classes we can say that the training is completed when after presentation of all input patterns there have been no misclassifications for the training data.

In the case of input patterns being represented by lower and upper bound values for each dimension it is a reasonable assumption that two patterns labelled as belonging to two different classes can have overlapping regions. Consequently the stopping condition has to be augmented in order to ensure the finite training time. This has been achieved by specifying the minimum value that the parameter Θ can take.

After this augmentation the stopping condition is defined as follows.

The training is completed when:

a) after presentation of all training patterns there have been no misclassifications for the training data

or

b) the minimum, user specified value of the parameter Θ has been reached

The equation defining the decreasing process of Θ after each presentation of the training data is

$$\Theta^{new} = \varphi\Theta^{old} \qquad \text{(EQ 6-2)}$$

where: φ – the coefficient responsible for the speed of decrease of Θ ($0<\varphi<1$)

6.5. Implementation of the neural network

The neural network that implements the GFMM clustering-classification algorithm is shown in Figure 6-5. The network grows adaptively to meet the demands of the problem. The input layer has 2*n processing elements, two for each of the n dimensions of the input pattern $X_h = [X_h^l \; X_h^u]$. Each second layer node of this three-layer neural network represents a hyperbox fuzzy set where the connections of first and second layer are the min-max points and the transfer function is the hyperbox membership function. The min

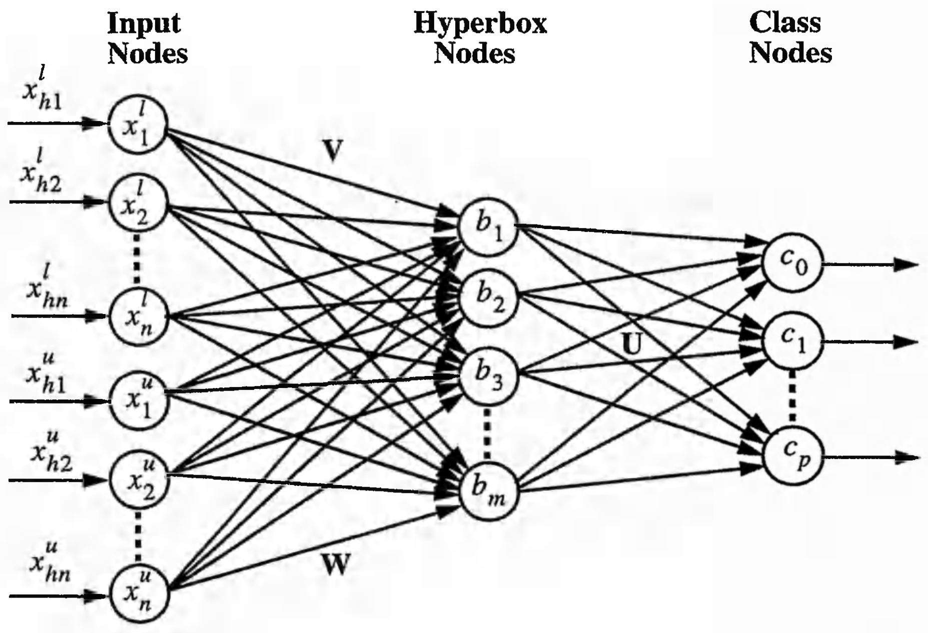


Figure 6-5: The three layer neural network that implements the GFMM clustering-classification algorithm.

points are stored in the matrix **V** and the max points are stored in the matrix **W**. The connections are adjusted using the algorithm described in section 6.3. The min points matrix **V** is applied to the first n input nodes representing the vector of lower boundaries X^l_h of the input pattern and the max points matrix **W** is applied to the other n input nodes representing the vector of upper boundaries X^u_h of the input pattern. A detailed view of the *j*-th second layer node is shown in Figure 6-6. The connections between the second and third layer nodes are binary values. They are stored in the matrix **U**. The equation for assigning the values of **U** is

$$u_{jk} = \begin{cases} 1 & \text{if } b_j \text{ is a hyperbox for class } c_k \\ 0 & \text{otherwise} \end{cases} \qquad \text{(EQ 6-3)}$$

where b_j is the *j*-th second layer node and c_k is the *k*-th third layer node. Each third layer node represents a class. The output of the third layer node represents the degree to which the input pattern X_h fits within the class *k*. The transfer function for each of the third layer nodes is defined as

$$c_k = \max_{j=1}^{m} b_j u_{jk} \qquad \text{(EQ 6-4)}$$

for each of the $p+1$ third layer nodes. Node c_0 represents all unlabelled hyperboxes from the second layer. There are two main ways that the outputs of the class (third) layer nodes can be utilized. If a soft decision is required, the outputs are utilized directly. If a hard decision is required, the class layer node with the highest value is located and its node value is set to 1 to indicate that it is the closest pattern class, and the remaining third layer node values are set to 0. This last operation is commonly referred to as a winner-takes-all response.

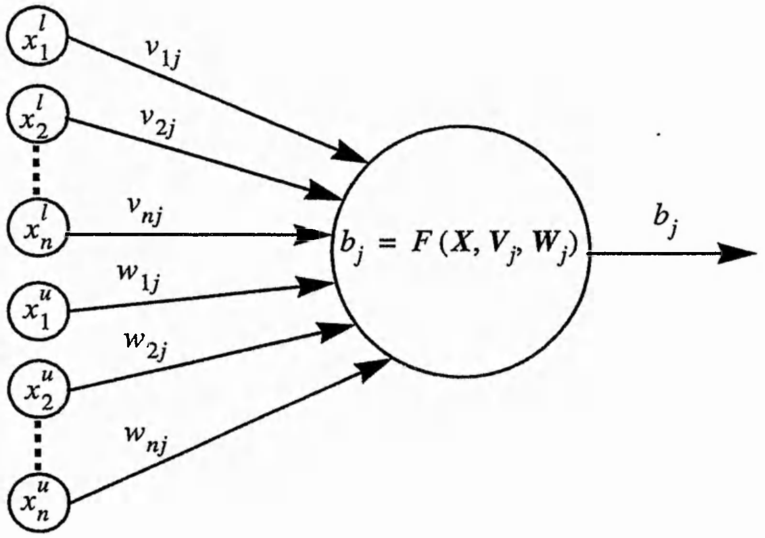


Figure 6-6: A detailed view of the j-th second layer node. The node with its associated membership function and connections in form of vectors V_j and W_j represents a hyperbox fuzzy set.

6.6. Examples

The following examples illustrate different aspects of the classification and clustering algorithms presented in previous sections.

6.6.1. Example 1 - Two dimensional clustering

A two dimensional data set consisting of 26 data points was constructed to show how

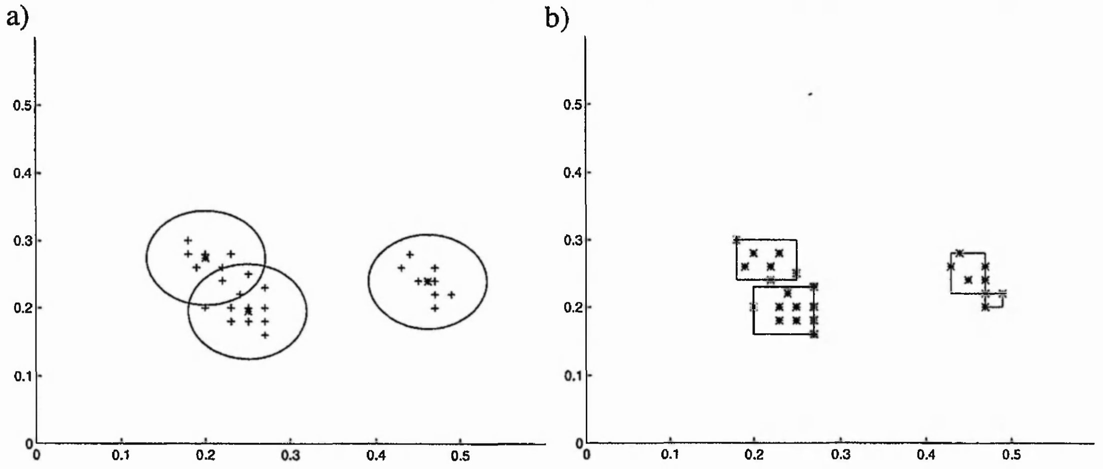


Figure 6-7: FLC and Min-Max differences between representation and created clusters

the Adaptive Fuzzy Leader Clustering NN and GFMM NN used as a clustering NN perform. In Figure 6-7 we can observe the difference between representation of clusters in these two algorithms:

a) centroids (represented by '*') and Euclidean distance (circles representing maximum range for each cluster)

b) hyperboxes (for a two dimensional case - rectangles) and Hamming distance (distance measured from the edges of an n-dimensional hyperbox).

However designing the pure clustering NN is not our prime objective. Testing of these two algorithms for various threshold parameters τ in AFLC and various maximum hyperbox sizes Θ in GFMM algorithm showed that:

a) it is very important to have a feel for the appropriate values of these parameters for specific data sets and

b) one has to be prepared for possibility of getting different sets of clusters for the same set of training data since both algorithms are sensitive to the order of pattern presentation.

6.6.2. Example 2 - Classification of patterns in form of lower and upper boundaries vectors

The data set used in this example was constructed in order to show the performance of GFMM algorithm on fuzzy input patterns and at the same time to present the potential advantages of the adaptive maximum size of a hyperbox scheme. This data set consists of 42 input patterns representing three classes. The first and second class have been constructed in such a way that finding the boundaries between them is non-trivial while the third class is a set of patterns standing alone and not overlapping with the other two. Two slightly different contraction procedures have been used. The difference is reported in the table below and regards only Cases 1 and 2 of the contraction part of the algorithm:

Procedure 1 (as presented in Table 6-1)	Procedure 2
Case 1: $v_{j\Delta} < v_{k\Delta} < w_{j\Delta} < w_{k\Delta}$ $v_{k\Delta}^{new} = w_{j\Delta}^{new} = \frac{v_{k\Delta}^{old} + w_{j\Delta}^{old}}{2}$ Case 2: $v_{k\Delta} < v_{j\Delta} < w_{k\Delta} < w_{j\Delta}$ $v_{j\Delta}^{new} = w_{k\Delta}^{new} = \frac{v_{j\Delta}^{old} + w_{k\Delta}^{old}}{2}$	Case 1: $v_{j\Delta} < v_{k\Delta} < w_{j\Delta} < w_{k\Delta}$ $v_{k\Delta}^{new} = w_{j\Delta}^{old}$ Case 2: $v_{k\Delta} < v_{j\Delta} < w_{k\Delta} < w_{j\Delta}$ $w_{k\Delta}^{new} = v_{j\Delta}^{old}$

The training in both cases was performed in 11 passes through the data set and testing produced perfect recall. The starting growth parameter was Θ=0.1 and the coefficient responsible for the decreasing speed of Θ was φ=0.9. The training resulted in the formation of 9 hyperboxes for procedure 1 and 8 hyperboxes for procedure 2. The created hyperboxes are shown in Figure 6-8. To show the differences between contraction procedure 1 and 2 the detailed picture of created hyperboxes for class 1 and 2 are presented in Figure 6-9 and Figure 6-10 respectively. The actual difference between these two procedures is that (for the cases 1 and 2) in the first (original) one both the overlapping hyperboxes B_j and B_k are contracted while in the second one the currently expanded hyperbox B_j is allowed to fully include the input pattern and the contraction is applied only to the hyperbox B_k. In this example using the second contraction procedure resulted in all training patterns having full memberships in appropriate hyperboxes, classes.

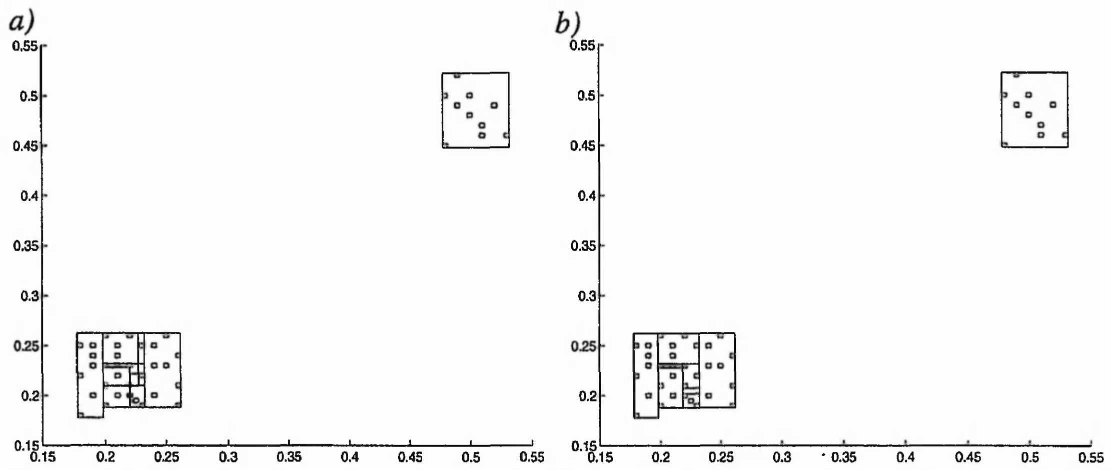


Figure 6-8: The result of neural network training for two different contraction procedures: a) procedure 1; b) procedure 2

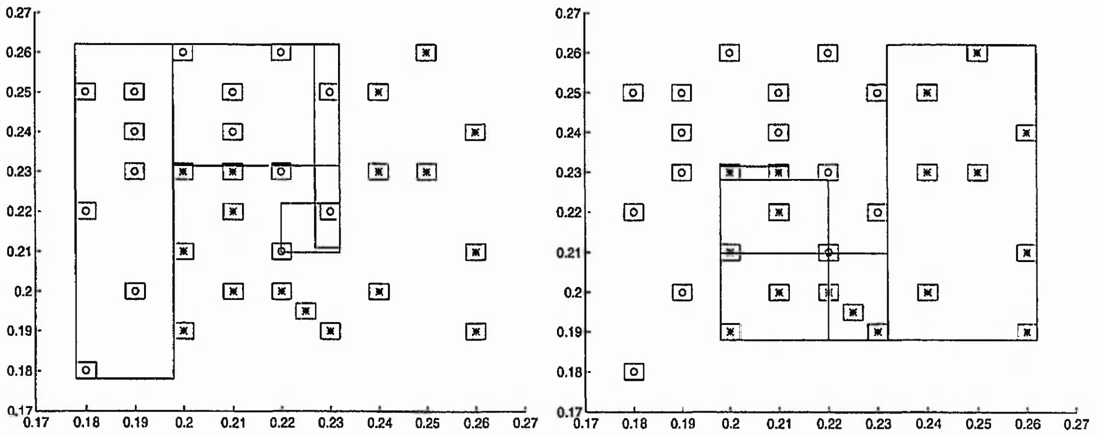


Figure 6-9: Class 1 and class 2 of the example for the first contraction procedure.The small rectangles with ‘o’ inside - input patterns that belong to the class 1, the rectangles with “*’ inside - input patterns that belong to the class 2

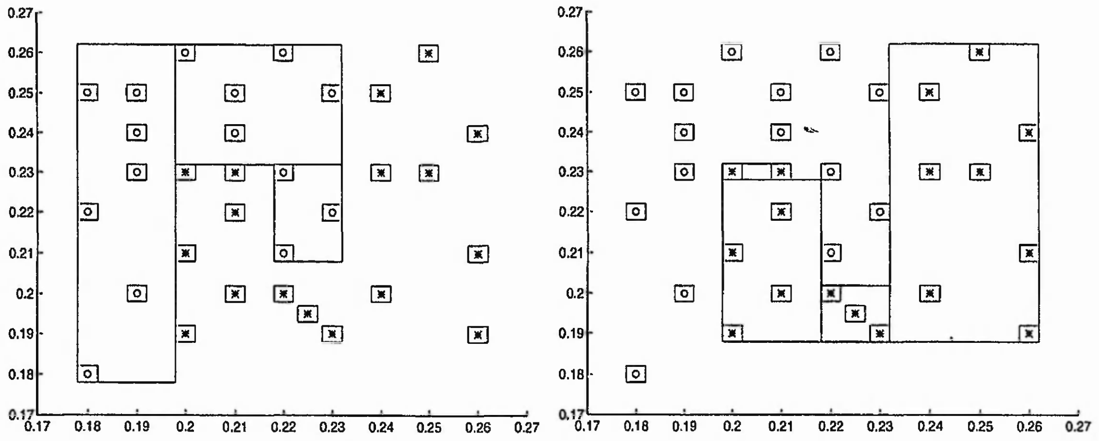


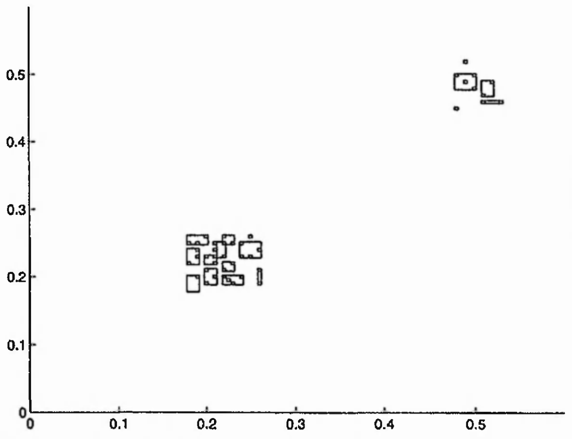
Figure 6-10: Class 1 and 2 of the example for the second contraction procedure.The small rectangles with ‘o’ inside - input patterns that belong to the class 1, the rectangles with “*’ inside - input patterns that belong to the class 2

To illustrate the superior performance of the algorithm with the adaptive maximum size of a hyperbox over the algorithms with parameter Θ preset and kept constant during the training, the training of the network for a few different constant Θs have been carried out. The results are shown in Figure 6-11.

In the table the short statistic is shown in a form of a number of hyperboxes created and number of misclassifications for values of Θ ranging between 0.08 and 0.033.

To obtain the perfect recall (zero misclassifications) for the training data the growth parameter had to be set to Θ=0.033 which resulted in the formation of 17 hyperboxes in comparison to 9 and 8 formed while using the adaptive maximum size of a hyperbox scheme.

Notice that 5 hyperboxes (Figure 6-11) have been formed for class 3 while 1 would be sufficient.



Q	Number of hyperboxes	Misclassifi-cations
0.08	4	10
0.07	6	6
0.06	7	3
0.05	9	6
0.04	12	2
[illegible]	[illegible]	[illegible]

Figure 6-11: The result of NN training for the 42 input pattern data set (3 classes). Left: the hyperboxes created for Θ=0.033 - the biggest Θ for which there have been no misclassifications for the training data. Right: the table showing the number of created hyperboxes and number of misclassifications for various Θ (Θ was constant during training).

6.6.3. Example 3 - The example of clustering/classification of labelled and unlabelled fuzzy input patterns.

This example was constructed to show the ability of the algorithm to process fuzzy labelled and unlabelled input patterns. The data set consists of 26 patterns from which 15 are labelled as belonging to one of 4 classes and the remaining 11 are unlabelled. The starting growth parameter Θ=0.1 and φ=0.9. The training has been completed in 3 passes through the data set and 4 hyperboxes have been formed. The algorithm performed as expected and dealt successfully with both labelled and unlabelled patterns. It allowed unlabelled patterns to be included into the labelled hyperboxes while resolved all overlappings between hyperboxes from different classes.

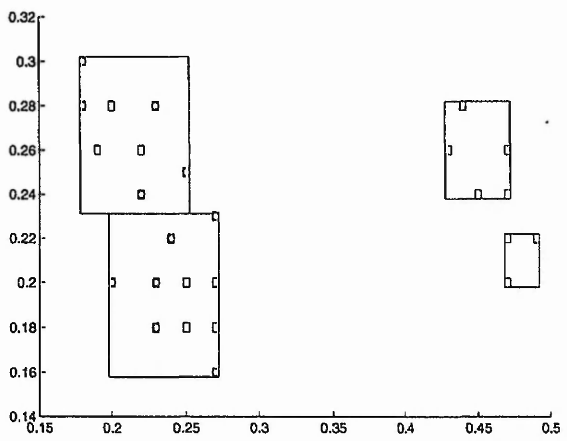


Figure 6-12: The example of clustering-classification of the fuzzy labelled and unlabelled patterns. The unlabelled patterns are the rectangles with 'o' mark inside.

6.6.4. Example 4 - The Fisher iris data

The Fisher iris data set was selected because of the huge number of results available from a wide range of classification techniques that can provide a measure of relative performance. The iris data consists of 150 four-dimensional feature vectors (patterns) in three separate classes, 50 for each class. In a way this example is very similar to example 2. In example 2 we considered the case of three classes where two of them were overlapping and the third easily distinguishable from the others. In the case of iris data we have two species of flowers that can be confused (similar features - class 2 and 3) and the third one with characteristic features allowing to distinguish it from the other two (class 1). Several test data sets have been used to determine the performance of our algorithm. The results will be presented for the following test data sets:

1) 25 randomly selected patterns from each class have been used for training and the remaining 75 for testing

2) all available data patterns have been used for training and testing

Since our algorithm evolved from the concept of the fuzzy min-max classification and clustering NNs it would be interesting first to compare the performance of these two algorithms.

For the test data set as in the point 1 the results presented in (Simpson, 1992) are as follows. The growth parameter was Θ=0.0175 and the number of hyperboxes built was 48. Training was performed in a single pass through the data set. The number of misclassifications was 2.

In comparison our algorithm produced 5 hyperboxes for starting parameter Θ=0.3 and φ=0.9. Training was completed in 3 passes through the data set. The number of misclassifications was 1.

The algorithm has been tested for different starting parameters Θ ranging from 0.7 to 0.03, two different contraction procedures (presented in example 2), and two test data sets (as shown above). Some of the results are shown in Table 6-2.

Test data set	Starting Θ	Contraction procedure 1				Contraction procedure 2			
		No of hyperboxes	Passes through data	Final Θ	Misclassifications	No of hyperboxes	Passes through data	Final Θ	Misclassifications
1	0.3	6(1/2/3)	7	0.1594	1	5(1/2/2)	3	0.2430	1
1	0.06	29(7/10/12)	1	0.06	2	29(7/10/12)	1	0.06	2
1	0.03	49(15/17/17)	1	0.03	0	49(15/17/17)	1	0.03	0
2	0.3	10(1/5/4)	16	0.0618	0	7(1/3/3)	7	0.1594	0
2	0.06	43(10/14/19)	1	0.06	0	43(10/14/19)	1	0.06	0

Table 6-2: The results of classification of the Fisher iris data by the proposed general fuzzy classification-clustering neural network

The numbers in brackets in "No of hyperboxes" column represent the number of hyperboxes formed for each of the three classes (class1/class2/class3)

As we can see from the results in Table 6-2 the proposed method produced, in general, considerably fewer hyperboxes then the fuzzy min-max classification NN with fewer misclassifications.

The comparison of the performance of the proposed algorithm with several other neural, fuzzy, and traditional classifiers on the same data set is presented in Table 6-3.

This comparative performance test demonstrates that the proposed algorithm performed better then other listed classifiers.

Technique	Misclassifications
Bayes classifier [1]	2
k-nearest neighbour [1]	4
Fuzzy *k*-NN [2]	4
Fisher ratios [1]	3
Ho-Kashyap [1]	2
Perceptron [3]	3
Fuzzy perceptron [3]	2
Fuzzy min-max NN [1]	2
GFMM algorithm [1]	1/0
GFMM algorithm [3]	0

Table 6-3: A comparison of the classification performance of various traditional, fuzzy, and neural classifiers

[1] Training set comprised 75 points (25 from each class) and test set comprised the remaining data points

[2] Training data comprised 36 data points (12 from each class) and test set comprised another 36 points. The results were then scaled up to 150 points for comparison.

[3] Training and testing data were the same.

The results in this table have been taken from (Simpson, 1992).

6.7. Discussion and conclusions

The need for the interpretation of the results of state estimation and confidence limit analysis of water distribution network directed our investigation into clustering and classification neural networks. This effort resulted in developing a new, robust fuzzy neural algorithm. Similar to the fuzzy min-max NNs this method utilizes min-max hyperboxes as fuzzy sets. The advantages of our generalised fuzzy clustering-classification neural system over the fuzzy min-max neural networks discussed in (Simpson, 1992) and (Simpson, 1993) can be summarised as follows:

1) Input patterns can be fuzzy (hyperboxes in pattern space) or deterministic (points in pattern space).

2) The fusion of clustering (unlabelled input patterns) and classification (labelled input patterns) resulted in more robust algorithm that can be used as pure clustering, pure classification or hybrid clustering-classification. This hybrid system exhibits an interesting property of finding decision boundaries between classes while clustering the data patterns that cannot be said to belong to any of existing classes. This property can be

particularly useful for multivariable system applications where it is practically not feasible to cover all possible states of the system at the training stage. Therefore, the mechanism to deal with new input patterns, that have never been presented to the neural network before, has to be provided.

3) Because of the introduction of the adaptive size of a hyperbox our method tends to define larger hyperboxes without sacrificing the recognition rate and as it has been shown in case of the Fisher iris data, it produced considerably less hyperboxes (5 or 6 in comparison to 48 produced by the fuzzy min-max NN) with less misclassifications.

It is also worth mentioning that training of this neural network is very fast and as long as there are no identical data belonging to two different classes, a recognition rate for training data is 100%. Since all the operations in the algorithm are relegated to simple compare, add, and subtract operations, the resulting algorithm is extremely efficient.

Because this method works by covering the pattern space with hyperboxes and in this way forming the decision boundaries its performance will deteriorate when the characteristics of the training and test data will be very different. Therefore it is important to provide as representative training data for the problem as possible.

Having developed the GFMM algorithm the next chapter presents the results of applying it to water distribution network.

Chapter 7

Fault detection and identification - neural classification in water systems

7.1. Introduction

Two broad categories of faults occurring in water distribution systems are considered in this work. The faults due to malfunctioning of transducers and telecommunication equipment are referred to as the measurement errors. And the faults due to leakages and wrong status of valves, invalidating the system model used in the estimation, are referred to as the topological errors.

The crucial difference between these two types of errors is the fact that although both are responsible for poor state estimates, the meter malfunctions do not have any bearings on the actual state of the system while the leakages or the valve status errors directly affect the physical system and can result in service disruptions. It is also the case that the measurement errors are uncorrelated and if there is a high enough local measurement redundancy it is often possible to reject erroneous data by using a suitable estimation procedure as described in Chapter 4. On the other hand, model based errors give rise to correlated changes in groups of incoming signals. In such a case the state estimation procedure trying to compensate for invalid network model results in a set of errors scattered across the network. In general, the rejection scheme used for uncorrelated measurement errors does not work very well in this case and some further analysis is required in order to diagnose the cause of error.

So, it is evident that the topological errors not only pose a much greater danger to the safety of water network operation but also are more difficult to locate and eradicate.

However, depending on the topology of the distribution network and the state estimator used, the aforementioned scatterings of errors form characteristic patterns that can be utilised to classify the state of the network.

This chapter presents the application of the GFMM neural network to this classification of the states of a water distribution system (Gabrys & Bargiela, 1997b).

The organisation of this chapter is as follows. The review of the previous work on the subject of bad data detection and identification is presented in Section 7.2. This is followed by Section 7.3 concerning the neural network based fault diagnosis system where the aspects of training and testing of the neural network using state estimates and residuals with confidence limits are discussed. And finally, the closing section of this chapter presents the discussion and conclusions.

7.2. Review of the previous work

Very often the algorithms found in the literature and referred to as bad data analysis are concerned with the identification and rejection of erroneous measurements and do not attempt to identify the underlying cause of the bad data. A number of methods dealing with the bad data detection and identification have been discussed in Section 4.2.1 of Chapter 4 while presenting different estimation methods.

This literature review will concentrate on methods of post-estimation stage of processing and interpreting the results of state estimation for fault detection purposes. Rather than only asking the questions: Are the state estimates accurate? How to construct the state estimation procedure in order to reject anomalous data?; one would like to know the answers to the questions: What do those state estimates mean? Is the current state a normal operating state of the network? Is there a leakage present that requires a remedial action? etc.

Somewhat surprisingly, while there is a large body of work concerning the robust estimation procedures the fault diagnosis algorithms in water distribution systems are not that well represented.

Bargiela introduced the idea of bad data analysis in water distribution systems state estimation (Bargiela, 1984). In order to distinguish between the measurement and topological errors his method checks the magnitude and sign of the weighted measurement residuals at each end of a pipe. It was shown that the presence of either a leakage or incorrect status of control valves is equivalent to neglecting a part of the actual network structure thus producing an imbalance at the network nodes adjacent to the pipe in question. The idea therefore was that the topological error can be thought of as a pair of erroneous load measurements for which the error terms (residuals representing the mass balances at those nodes) are carrying information about a type of topology error. Figure 7-1 gives a graphical representation of Bargiela's method.

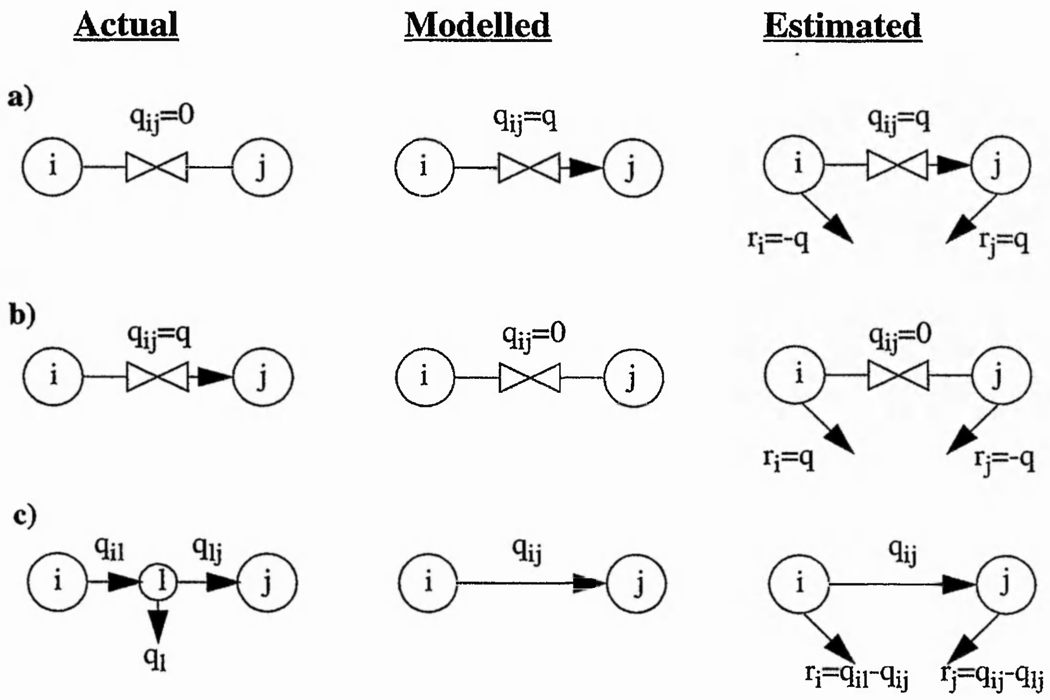


Figure 7-1: Identification of topological errors as presented by Bargiela: a) closed valve monitored as open; b) opened valve monitored as closed; c) leakage.

Although these ideas are very useful they rely on the high local measurement redundancy ratio so that the erroneous data can be rejected. It is not always the case and the effect of topological error occurrence cannot be restricted to the end nodes of affected pipe but is spread in the larger area around the leaking pipe.

This fact was recognised by Powell (Powell, 1992) whose method is based on finding paths linking groups of high measurement residuals. Once the connecting paths between high residuals have been identified, heuristics are applied to determine the location and cause of errors. In these heuristics the residuals are sorted by type, direction, magnitude and location. For instance if the leakage is present in the network the pressures in an area near to the leakage will decrease. On the other hand, if there is a blocked pipe in the network the pressure upstream of this pipe will be high and downstream it will be low. Since the changes in pressure are characteristic for different faults one should be able to observe those changes in the residuals representing the pressure measurements in mathematical model of the network.

Other publications on the subject for water distribution systems concern mainly leakage detection studies.

Pudar and Liggett (Pudar & Liggett, 1992) attempted the leak detection task by solving an inverse problem. This inverse problem is essentially the state estimation procedure with additional state variables being the unknown leaks. The method assumes that the leaks occur in the nodes and do not change the topology of the network. Furthermore, the locations of suspected leakages are assumed to be known. Unfortunately, both assumptions are a gross oversimplification.

Carpentier and Cohen in the paper (Carpentier & Cohen, 1993) tell the story about 10 years of involvement of their research group in the application of mathematical techniques for the management of complex water supply networks. One of the two main topics discussed is state estimation and leak detection. The leakage detection in this work is based on a comparison of the consumption values estimated on-line, using current, real flow measurements, with the pseudomeasurements of the same consumptions considered as standard values in the normal situation (without any leakages present). These pseudomeasurements are obtained from 24-hour mathematical model of the normal network operations. Throughout this work a heavy emphasis is put on the necessity of having the well calibrated model of the network. The performance of the method was tested on a real subnetwork of the water network of the city of Paris. The leakages were introduced to the network by opening fire-plugs in some places. The fact that experiments were carried out on the real network give additional weight to the results. The weights in the weighted least squares criterion are chosen in such a way that the errors occur in nodal mass balance equations and represent increase in nodal consumptions. Occurrence of a set of significant errors in some area of the network is treated as a sign of leakage presence in this area. No attempts were made to further process these errors in order to find a reduced number of the most likely pipe(s).

7.3. Neural network based fault diagnosis

7.3.1. Two interpretations of confidence limits

In any pattern classifier design problem it is necessary to have a representative set of accurate training examples. Since we would like to utilise the information about confidence limits in the process of constructing our classification system it is absolutely necessary to understand what is the meaning of those confidence limits when calculated for different values of state estimates and if and when they can be used in the training stage without compromising the performance of the pattern classifier.

The two interpretations of confidence limits referred to in the title of this section can be explained by imagining two different experiments.

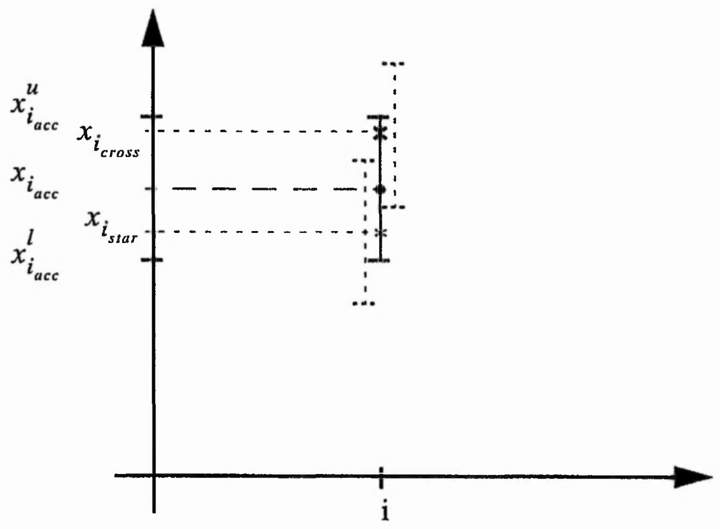


Figure 7-2: Graphical representation of state estimates and confidence limits for accurate and inaccurate measurements. $x_{i_{acc}}$ - estimate for the i-th state variable calculated for accurate measurements; $x^l_{i_{acc}}$, $x^u_{i_{acc}}$ - the lower and upper bound for $x_{i_{acc}}$; $x_{i_{star}}$ and $x_{i_{cross}}$ are the two examples of the i-th state variable estimate calculated for inaccurate measurements.

In the first experiment the estimates are calculated for accurate measurements. If one also assumes that the mathematical model of the process used in the estimation procedure accurately represents the behaviour of the physical system, a true state estimate of the system can be obtained. This is denoted by $x_{i_{acc}}$ in Figure 7-2 for the *i-th* state variable.

However, since the measurements have a finite accuracy it is interesting to know how sensitive is the state estimate to the measurement inaccuracies. In this case, the confidence limits calculated for the true state represent the boundaries within which all estimates of this true state will fall as long as the measurements used are at least as accurate as the ones taken to compute the confidence limits themselves. In Figure 7-2 the lower and upper bound for the *i-th* state variable are denoted $x^{l}_{i_{acc}}$ and $x^{u}_{i_{acc}}$ respectively. In other words, if for the purpose of pattern classification the true state was labelled as "the normal operating state" all the estimates falling within its confidence limits could be classified as "the normal operating state".

In the second experiment the estimates are calculated for a set of measurements that are measured with some finite error. This is to say that the true state is unknown and for a given set of inaccurate measurements one can only compute the best estimates of this true state. Two examples of the instantaneous estimates of the true state value $x_{i_{acc}}$ are denoted by $x_{i_{star}}$ and $x_{i_{cross}}$ in Figure 7-2. Unlike the confidence limits computed for the true state, the confidence limits found for any of the instantaneous estimates only indicate that the true state value is contained within their range. The confidence limits for $x_{i_{star}}$ and $x_{i_{cross}}$ are depicted in form of dashed vertical lines in Figure 7-2. When in the extreme case the estimated values were equal to $x^{l}_{i_{acc}}$ or $x^{u}_{i_{acc}}$ using the confidence limits for such estimates during the training of the classification network would mean the introduction of additional 50% error to the cumulative error resulting from inaccurate measurements.

It follows from the above that when the true state (or a very good estimate of it) can be computed, than the confidence limits found for such an estimate directly give a hyperbox (cluster) without the need to use a large number of instantaneous estimates during the training (which would arrive at the same cluster).

However, when the sufficiently accurate estimate of the true state cannot be found one has to resort to a large number of correctly labelled instantaneous estimates.

7.3.2. Generating the training data

While for the well maintained water distribution systems the normal operating state data can be found in abundance the instances of abnormal events are not that readily available. In order to observe the effects of abnormal events in the physical system one sometimes is forced to resort to deliberate closing of valves or opening of hydrants (to

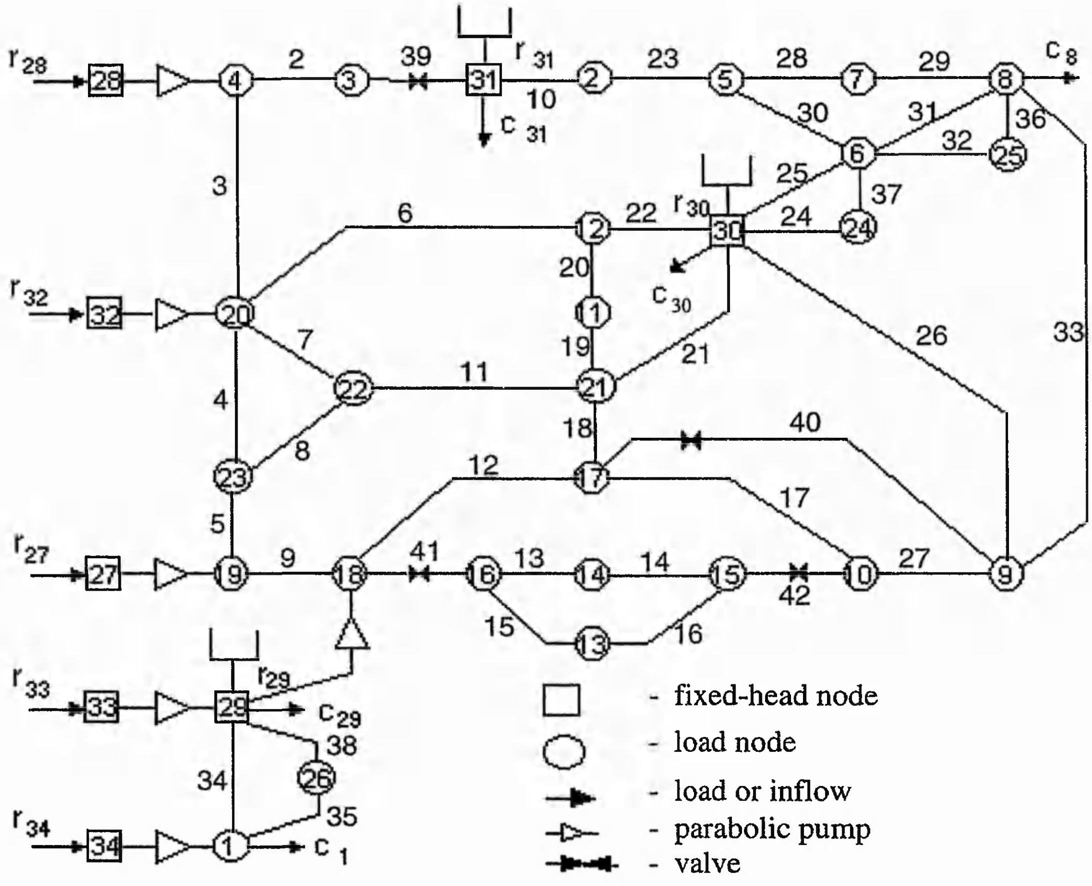


Figure 7-3: 34 - node water network with numbered pipes. The pipe numbers are at the same time the indexes of classes representing leakages at these pipes.

simulate leakages) (Carpentier & Cohen, 1993). Although such experiments can be very useful to confirm the agreement between the behaviour of the physical system and the mathematical model, it is not feasible to carry out such experiments for all pipes and valves in the system during the whole day or days as might be required in order to obtain the representative set of labelled data.

It is an accepted practice that, for processes where the physical interference is not recommended or even dangerous, mathematical models and computer simulations are used to predict the consequences of some emergencies so that one might be prepared for quick response. In our case the computer simulations were used to generate data covering 24 hour period for the water distribution network depicted at Figure 7-3.

The 24 hour profiles of consumptions and inflows that characterise the normal operating states throughout the day are shown at Figure 7-4, Figure 7-5 and Figure 7-6.

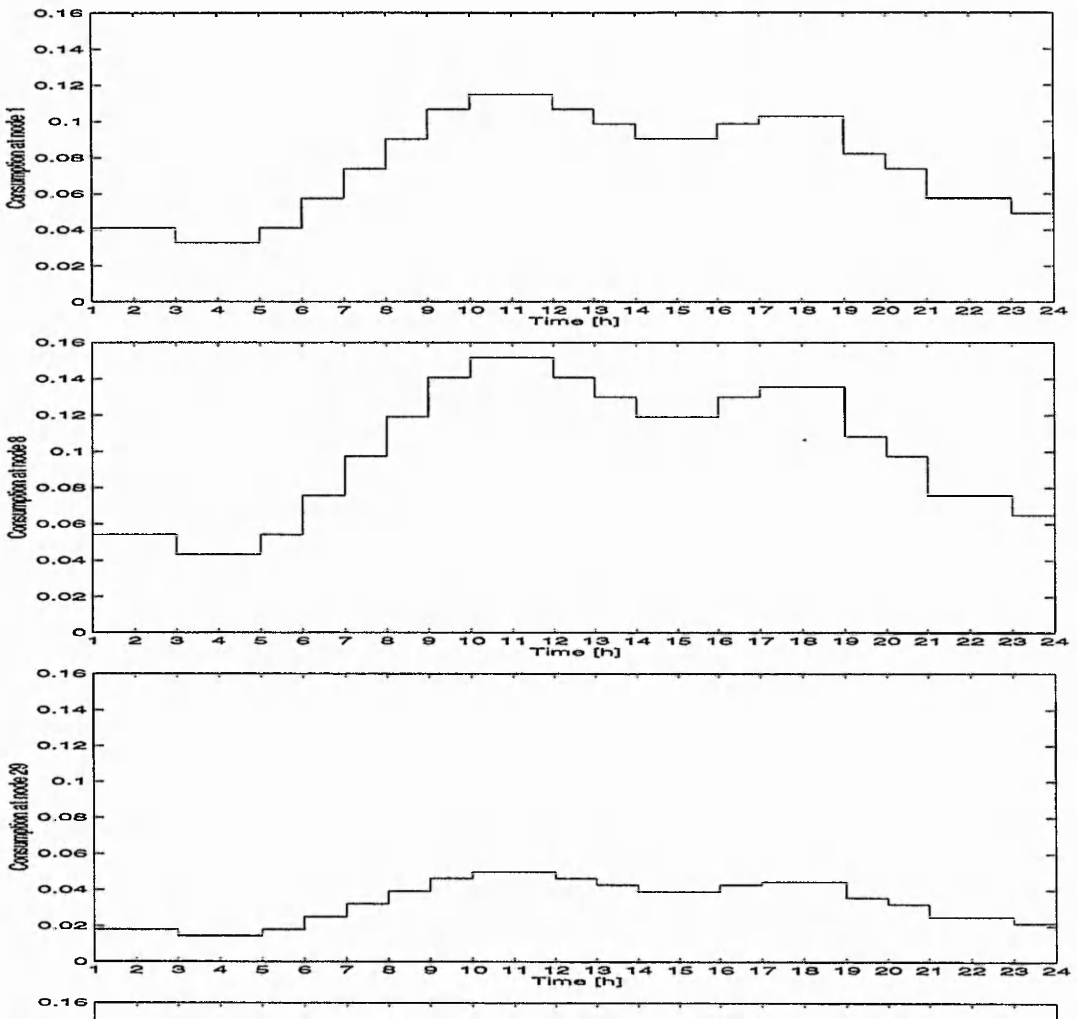


Figure 7-4: 24 hour profiles of consumptions at nodes 1, 8, 29, 30 and 31.

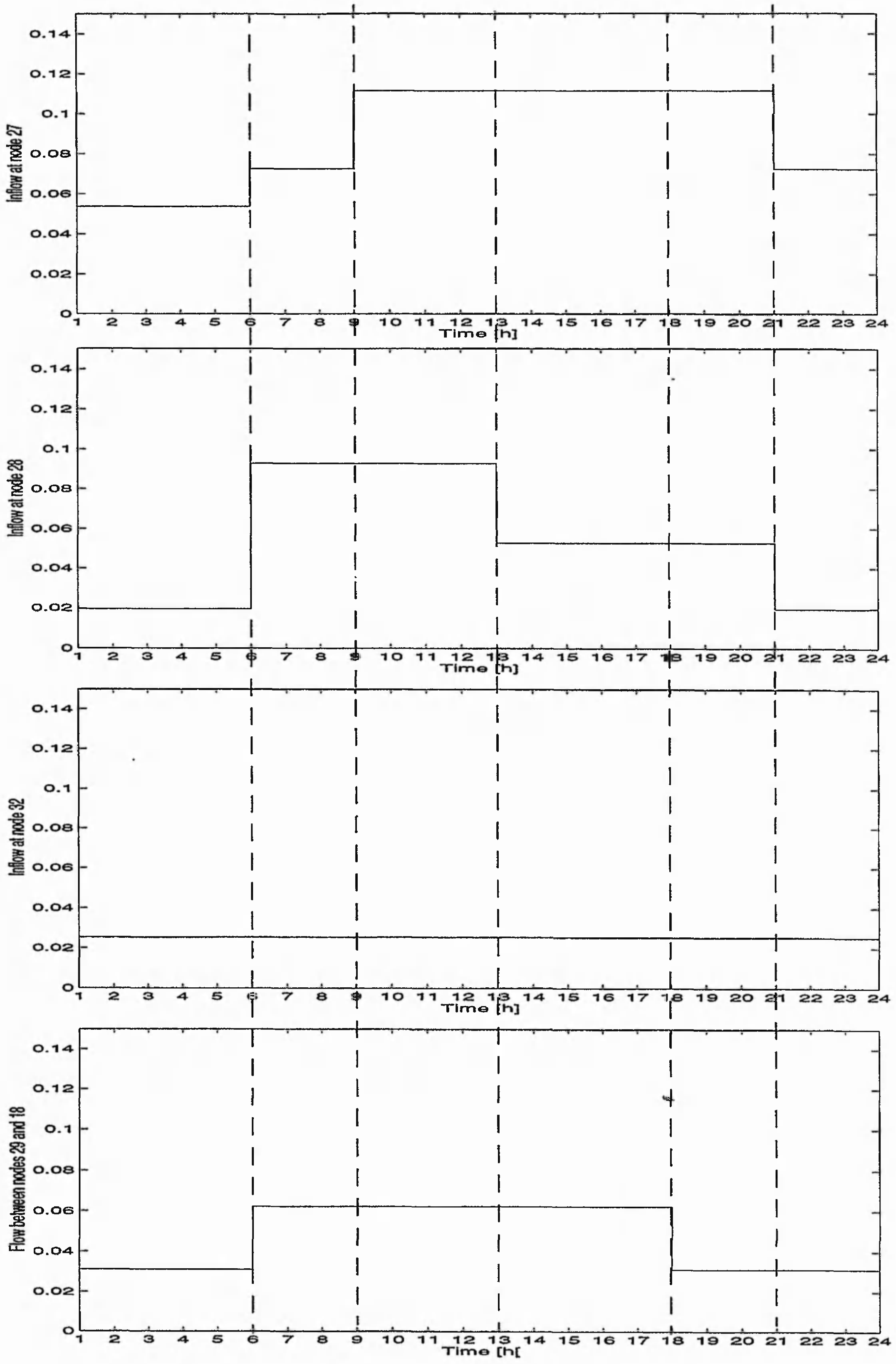


Figure 7-5: 24 hour profiles of inflows at fixed head nodes 27, 28, 32 and booster pump between nodes 29 and 18

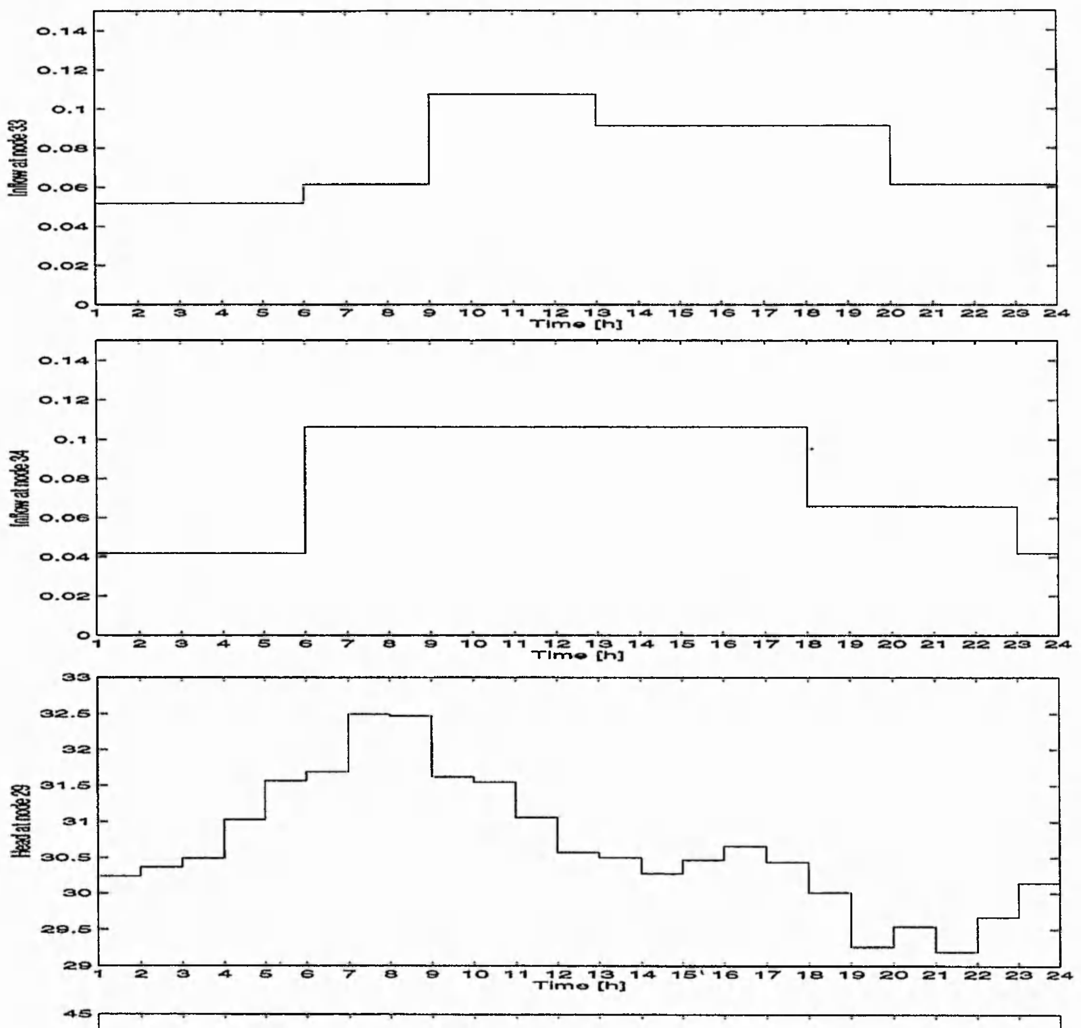


Figure 7-6: 24 hour profiles of inflows at fixed head nodes 33, 34 and heads at reservoir nodes 29, 30, 31.

From the classification system point of view the most interesting values are the state estimates, residuals and their confidence limits. Traditionally the confidence limit analysis was applied to the state estimates for the reasons explained in Chapter 5, while the bad data analysis was conducted using only residuals.

In this work we would like to explore the potential of using both state estimates with their confidence limits and residuals with their confidence limits for the fault diagnosis purposes.

It was shown in Chapter 5 how confidence limits can be found for state estimates but in order to carry out the confidence limit analysis for residuals the augmented matrix formulation of the LMS problem can be used.

In this formulation the solution to the LMS problem given by the equation:

$$\boldsymbol{A}^T(\boldsymbol{A}\Delta\boldsymbol{x} - \boldsymbol{b}) = \boldsymbol{0} \qquad \text{(EQ 7-1)}$$

can be rewritten in the following augmented form where the residuals, $\boldsymbol{r} = \boldsymbol{A}\Delta\boldsymbol{x} - \boldsymbol{b}$, are included directly into the vector of unknowns

$$\begin{bmatrix} -\boldsymbol{I} & A \\ A^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{r} \\ \Delta x \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix} \qquad \text{(EQ 7-2)}$$

Once the vector of unknowns has been augmented to include residuals the confidence limits can be now computed both for state estimates and residuals as described in Chapter 5.

The process of generating the training data is shown in the form of block diagram at Figure 7-7. It consists of three major blocks.

The system simulation is a substitute for the physical water distribution network. It is this module where the leakages are simulated by updating the topology information rather than opening hydrants. In the second module the estimation process is carried out for accurate measurements taken from the simulation module but without a knowledge of any anomalous event that might have happened, as would be the case in the real distribution network. In the third module the confidence limits are found for state estimates and residuals calculated at the estimation stage. Additionally to the state estimates with their confidence limits and residuals with their confidence limits the system's status or label of the current pattern is stored.

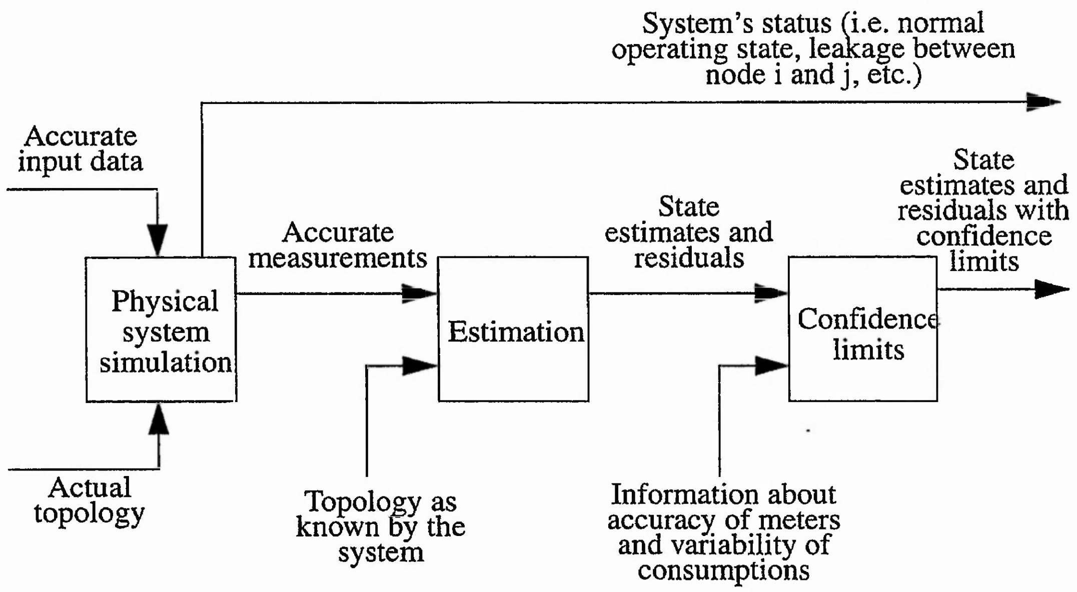

System's status (i.e. normal operating state, leakage between node i and j, etc.)
Accurate input data
Physical system simulation
Actual topology
Accurate measurements
Estimation
Topology as known by the system
State estimates and residuals
Confidence limits
Information about accuracy of meters and variability of consumptions
State estimates and residuals with confidence limits

Figure 7-7: Graphical representation of the training patterns generation scheme.

Simulation of leakages

In the physical system simulator, the leak is modelled as an additional demand lying midway between the two end nodes of the pipe. This additional demand is not modelled as a pressure dependent variable and thus can be set to any desired value. The reservoirs inflows and other network consumptions are adjusted to cover the additional demand resulting from leak. The pumping stations are assumed to produce a constant inflow and are not affected by leakages. By systematically working through the network, ten levels of leaks were introduced, one at a time, in every single pipe for every hour of the 24 hour period. This resulted in generating 9840 labelled input patterns for leakages ranging from 0.002 to 0.029 [m^3/s].

Simulation of wrong status of valves

In a similar fashion to the introduction of leakages valves in the network that are normally open were simulated as closed and labelled patterns in form of state estimates and residuals with confidence limits were stored. Additional 96 patterns have been generated in this way.

The whole set of parameters used during the generation of the training set are shown at Table 7-1.

Head measurements	1, 2, 4, 8, 11, 15, 17, 19, 22, 29, 30, 31
Fixed head inflow measurements	27, 28, 29, 30, 31, 32, 33, 34
Consumptions and pseudomeasurements	All nodes
Leak levels	0.002, 0.005, 0.008, 0.011, 0.014, 0.017, 0.020, 0.023, 0.026, 0.029 [m^3/s]
Parameters used in confidence limit analysis	
Accuracy of head measurements at load nodes	+- 0.1[m]
Accuracy of inflow measurements	+- 1%
Variability of consumptions	+- 10%

Table 7-1: Parameters used during generation of the training data set.

7.3.3. State estimates and classification system design

In the first attempt of constructing the recognition system based on state estimates the set of 9864 training patterns representing 42 categories were used. The training data spanned across 24 hour period of water network operation. The 42 categories stood for normal operating state and leakages in 41 pipes of the network depicted at Figure 7-3. The indexes d_h (see GFMM algorithm description Chapter 6, Table 6-1) of classes were chosen as follows: d_h=1 - normal operating state; d_h=2 - leakage in pipe between nodes 3 and 4; d_h=3 - leakage in pipe between nodes 4 and 20; and the rest of indexes as shown at Figure 7-3.

The training data had to be first scaled in order to be contained in the range (0,1) as required by the GFMM algorithm. The effective range of values for the nodes' head state variables was chosen to be between 20 and 60 [mH2O], and for inflows between -0.2 and 0.2 [m^3/s]. There were 6 state variables (heads in fixed-head nodes 27, 28, 32, 33, 34 and inflow at node 32) that did not change during the 24 hour simulation period and since they did not introduce any additional information that could be used to distinguish between patterns from different classes they were excluded from the training set. Ultimately the training set contained of 9864 examples of 36 dimensional input vectors.

The initial max size of hyperboxes was set to a fairly large value Θ=0.1 and the coefficient φ responsible for the speed of decreasing of Θ was set to φ=0.9. The training was completed after two runs through the training data and the final value of Θ was 0.09.

Although technically there were no misclassifications for the training set after examining the membership values it was noticed that a large number of input patterns have been classified as fully belonging to more than one class (in GFMM algorithm an input pattern can belong fully to two or more different classes only if it lays on the edge of two or more hyperboxes representing different classes and being adjacent to each other at the same time). Since the max size of hyperboxes was rather large it might have been the reason for the behaviour of the recognition system. In an attempt to check what role did the size of parameter Θ play in the above experiment another training for the same data was carried out but this time parameter Θ was set to the value of 0.04. This value was arrived at by analysing the ranges of variability of heads and inflows for minimum and maximum simulated leaks for each pipe at each hour. It guaranteed that a range of leaks in any particular pipe at any particular hour could be represented by a single cluster.

The training was completed after a single run through the data and as one could expect there were no misclassifications for the training set. Although the number of incidents where an input pattern have been classified as fully belonging to more than one class has been reduced, the problem of not being able to resolve important overlappings remained.

Further analysis of clusters formed during the training and membership values for the training set has revealed few important characteristics of the training data. Firstly, there are variables in the input patterns that vary over the 24 hour period but are assumed to be constant and are not affected by leakages for any particular hour. These variables are the pumping stations' inflows and heads in fixed-head nodes. One can view them as part of the state vector that characterises the pattern of network operations in different parts of the day. The other part of the state vector comprises of heads in the load nodes and reservoirs' inflows that do change when a leakage occurs. Consequently, the variables not affected by leakages and remaining constant prevented the training algorithm from resolving overlappings in the part of the input patterns that were affected by leakages. The reason for that was that according to the rules used to decide whether two hyperboxes overlap, no overlapping could be detected. A solution to this problem has been proposed by splitting the state estimates into two parts. These that are not affected by leakages but do carry important information about current state of operation of the network and cannot be simply discarded; and these that can be used to detect fault occurrence by the virtue of their "susceptibility" to any changes in the network. The way of combining the two parts of the input patterns will be described later in this section.

Apart from the problem described above, the analysis showed that the booster pump between nodes 29 and 18 acted as a separator of the upper and lower part of the network from Figure 7-3 and since changes in one part of the network had no bearings on the other part, only the upper part will be used in further investigations.

Before progressing to describing further experiments let us take a closer look at some statistics obtained from analysis of membership values for last trial.

The values in Table 7-2 are the average numbers of classes with significant membership values (in this case larger than 0.75) for leakages of magnitude varying from 0.002 [m^3/s] to 0.029 [m^3/s] occurring in different locations. The first thing to be noticed in this table are the results for classes with indexes 34, 35 and 38 (leakages in three pipes connecting nodes 1, 26 and 29). On one hand, the leakages occurring in any of these three pipes result in a similar pattern and even for large leakages it is difficult to pinpoint a single pipe where the leakage occurred. On the other hand, because nodes 1, 26, 29, 33 and 34 are separated by the booster pump between nodes 29 and 18 from the rest of the network, it is possible, even for very small leakages, to restrict the possible leakage area to those three pipes.

As for the rest of the network, one could try to identify the areas of the network in which leakages are more easily detected, compared to other areas, and to find the physical reason for such results. For instance, good results for pipes 10, 39, 23 can be explained by the fact that there are three accurate measurements in the vicinity (head measurements at nodes 2 and 31 and inflow measurement at node 31). Another positive example are pipes 14, 16 and 42 with head measurement at node 15. On the other hand, one could expect difficulties in detecting and locating a leakage in the area between nodes 8 and 30.

Index of class	Magnitude of leakage									
	2 [l/s]	5 [l/s]	8 [l/s]	11 [l/s]	14 [l/s]	17 [l/s]	20 [l/s]	23 [l/s]	26 [l/s]	29 [l/s]
2	29	20	16	15	13	11	9	6	3	2
3	34	22	17	16	14	13	11	9	6	3
4	37	22	18	16	15	14	13	10	7	4
5	36	22	18	16	14	11	10	7	6	5
6	33	31	24	20	18	17	13	7	6	5
7	36	23	18	17	15	14	13	10	8	5
8	37	23	19	16	16	14	12	10	8	6
9	35	21	18	15	13	11	9	6	5	4
10	15	13	9	6	5	3	2	1	1	1
11	35	27	20	18	17	15	14	12	9	4
12	32	21	17	14	12	9	7	5	4	2
13	30	18	13	10	7	5	4	3	3	3
14	27	16	11	8	4	3	3	3	1	1
15	32	20	17	13	10	8	6	4	3	3
16	29	17	12	10	6	4	4	3	3	2
17	31	21	16	14	11	9	6	5	3	2
18	35	25	19	18	15	14	11	9	5	4
19	33	33	26	20	17	16	11	6	5	4
20	32	30	28	24	21	17	10	7	5	3
21	33	31	29	25	21	17	11	8	6	5
22	31	27	26	24	21	16	10	9	7	4
23	27	14	10	8	6	5	4	3	3	2
24	30	27	25	23	19	15	11	10	7	5
25	30	27	25	22	19	16	11	10	7	5
26	30	27	25	22	18	14	10	8	7	4
27	35	27	24	20	17	13	7	4	3	2
28	32	24	17	15	12	10	8	6	5	4
29	32	27	24	20	17	14	11	8	7	5
30	32	30	22	15	11	9	7	5	4	3
31	31	27	23	19	17	16	11	8	7	5
32	31	28	23	20	17	15	11	9	7	5
33	31	28	24	21	18	16	12	9	8	6
34	3	3	3	3	3	3	3	3	3	3
35	3	3	3	3	3	3	3	3	3	3
36	31	26	22	18	16	13	9	7	6	4
37	31	28	24	22	19	16	12	10	8	5
38	3	3	3	3	3	3	3	3	3	2
39	18	16	13	11	9	5	2	1	1	1
40	34	27	24	20	16	13	6	4	3	3
41	33	21	18	15	12	9	8	6	5	3
42	29	19	14	11	9	6	4	3	3	1

Table 7-2: Average number of classes with membership values larger than 0.75.

Two level recognition system

The existence of multiple classes with full membership for a large number of testing patterns, in the examples described above, prompted an investigation into an alternative formulation of the recognition system. A two level recognition system, shown in Figure 7-8, has been proposed as a means of solving the problem. It has been established that it is beneficial to split the input vectors into two parts: $\mathbf{X}_{\mathbf{I}}$ representing the pump stations' inflows and heads in the fixed head nodes; and $\mathbf{X}_{\mathbf{II}}$ representing load nodes' heads and reservoirs' inflows. In the first level of the system inputs $\mathbf{X}_{\mathbf{I}}$ are processed and one of the n second level "experts" is selected for further processing of the second part of the input pattern $\mathbf{X}_{\mathbf{II}}$ to produce the water system state classification denoted by $\mathbf{C}$.

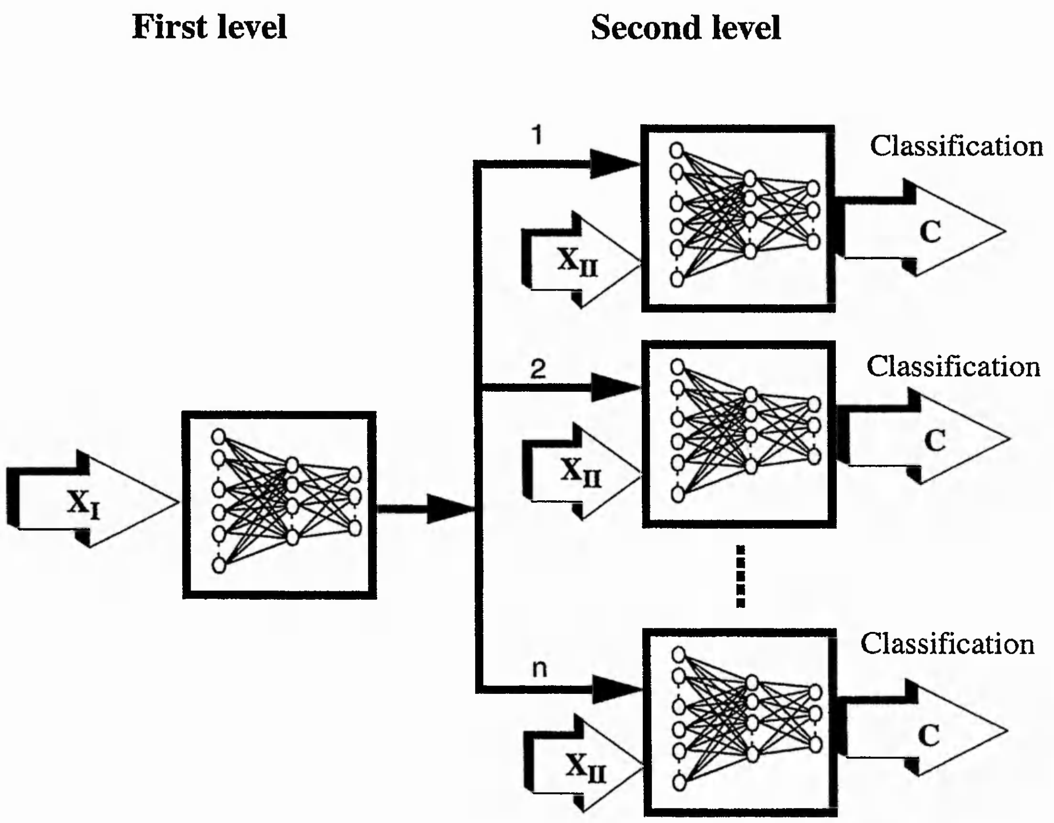


Figure 7-8: Two level recognition system. First level consists of one neural network of the type shown at Figure 6-5 and its purpose is to select one of the n second level "experts". Input to the first level NN, $\mathbf{X}_{\mathbf{I}}$, comprises all the variables not affected by occurrence of anomaly. Second level consists of n NNs. They are called "experts" since each of them is trained using only a part of training set and covers a distinctive part of 24 hour operational period. Input to the second level NNs, $\mathbf{X}_{\mathbf{II}}$, comprises all the variables sensitive to occurrence of anomaly. The output of the second level NNs is the classification of the water network state.

In terms of water distribution systems the purpose of the first level of this recognition system is to distinguish different typical behaviour of the water system (i.e. night load, peak load etc.) while the second level components are responsible for detection of anomalies for some characteristic load patterns. The second level can be, therefore, viewed as "experts". From this definition it is easy to see that if there are distinctive variations in typical network behaviour for different days of the week or seasons of the year they can be accommodated without the need to retrain the existing networks; a new expert network is added to the second level and the size of the first level network is increased accordingly. It can be now appreciated why such a great emphasis has been put on the property of GFMM neural network to be able to grow to meet the demands of the problem.

Although the initial reason behind splitting the input vectors into two parts was the need to eliminate full membership of input patterns in multiple classes, there are additional benefits of this operation. Firstly, the dimensions of the input patterns processed by neural networks in the first and second level are reduced in comparison to the full input pattern **X** since $\mathbf{X}^{kx1}$, $\mathbf{X}_{\mathbf{I}}^{px1}$, $\mathbf{X}_{\mathbf{II}}^{lx1}$ and k=p+l. Secondly, the fact that only one of the "experts" is selected for further processing also means that the other n-1 "experts" are not active. In this way another dimensionality reduction is achieved since each of the second level networks covers only small part of the day rather than 24 hour period.

The performance of the above two level recognition system has been tested for the upper part of the water distribution system shown at Figure 7-3. Input to the first level network consisted of inflows to nodes 27, 28, 32, flow between nodes 29 and 18 and heads at the reservoir nodes 30 and 31. Six characteristic inflow patterns could be found for six periods during 24 hour water network operation and they are marked by dashed vertical lines in Figure 7-5. The training set has been split into six parts and each of the six second level "expert" neural networks has been trained separately. First, a number of training sessions has been conducted for different values of parameter Θ and for training sets constituting the state estimates with and without confidence limits. Before progressing to a more detailed analysis of the recognition system performance it has to be highlighted that a major goal of avoiding the full membership of testing patterns in multiple classes has been achieved. The comparative results of the testing carried out for the training sets are presented in Table 7-3.

The first interesting result is the comparison of the performance of the recognition system trained for input patterns with and without confidence limits. It can be seen that the inclusion of the confidence limits in the training set has resulted in significantly less misclassifications. It is believed that the information carried by the confidence limits allowed the training algorithm to resolve overlappings in a more robust way.

The second important result of this testing is concerned with the significance of the parameter Θ. Referring to Table 7-3 one can notice a huge difference in the number of misclassifications when Θ is too large for the data used. It has been found that since the potential variability of different state variables, reflected by the confidence limits, can vary substantially from one state variable to another and from one operational period of the day to another, it is beneficial to set parameter Θ separately for each dimension of the input patterns and for each of the second level "experts". The small number of misclassifications, as reported in the row "variable" Θ of Table 7-3, illustrates the point.

Training set	Parameter Θ	Misclassification rates			
		Highest membership	Top 2 alternatives	Top 3 alternatives	Top5 alternatives
State estimates computed for accurate measurements without confidence limits	0.02	26.97%	15.59%	10.90%	6.93%
	0.01	8.17%	4.54%	3.38%	2.14%
State estimates computed for accurate measurements including confidence limits	0.02	18.67%	11.57%	8.49%	5.34%
	0.01	1.03%	0.36%	0.23%	0.11%
	Variable*	0.03%	0.01%	0%	0%

Table 7-3: Misclassification rates for a test set consisting of 9144 examples of state estimates computed for accurate measurements.

* Parameter Θ was determined separately for each dimension of each of the six subsets of the training set and was set to the value of the largest input hyperbox for each of these six subsets.

While the above results are very encouraging one needs to remember that testing has been carried out for state estimates computed for accurate measurements. In order to truly test the performance of the recognition system an independent large testing set has been generated.

7.3.4. Generating the testing data set

In a similar fashion to the training data generation the process of testing data generation can be pictured in a form of block diagram shown at Figure 7-9.

The accurate values obtained from water system simulator are fed into the Telemetry block where random errors, laying within ranges specified by assumed accuracy of meters and variabilities of consumptions, are added to accurate measurements in order to simulate the noisy environment of the real water distribution system.

These noisy measurements are sent to Estimation block where the best state estimates, according to LS criterion for a given set of measurements, are found. The corresponding residuals and system's status are also saved.

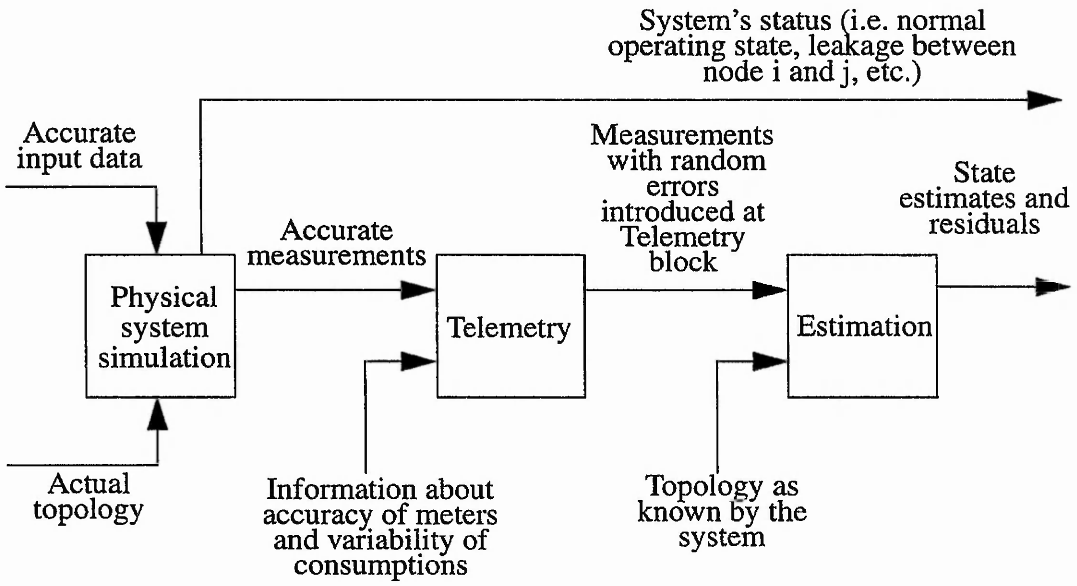

System's status (i.e. normal operating state, leakage between node i and j, etc.)
Accurate input data
Measurements with random errors introduced at Telemetry block
State estimates and residuals
Accurate measurements
Physical system simulation
Telemetry
Estimation
Actual topology
Information about accuracy of meters and variability of consumptions
Topology as known by the system

Figure 7-9: Graphical representation of the testing set generation scheme.

In this way by systematically working through the network, for each of the normal situations and for each of the ten levels of leaks introduced, one at a time, in every single pipe, for every hour of the 24 hour period, ten estimations have been performed for ten random sets of measurement errors. This resulted in generating 91440 test data patterns.

7.3.5. Results for the testing set

The two systems that performed best for the training data (see Table 7-3) have been put to the test using the above described testing set comprising of 91440 examples. The results of this testing are shown in Table 7-4 and Table 7-5. The percentage of misclassified input patterns for the class with the highest membership value, top 2, top3 and top 5 alternatives have been used as a means of assessing the ability to correctly detect and locate leakages. Additionally, the share of patterns representing different levels of leakages in the overall misclassification rate is presented.

Mem-bership values	Overall misclassi-fication	Split of misclassifications according to different levels of leaks									
		2 [l/s]	5 [l/s]	8 [l/s]	11 [l/s]	14 [l/s]	17 [l/s]	20 [l/s]	23 [l/s]	26 [l/s]	29 [l/s]
Highest	16.8274%	33.22%	18.33%	11.02%	8.18%	7.12%	6.01%	5.02%	4.41%	3.59%	3.12%
Top2	6.1144%	57.23%	21.41%	8.80%	4.69%	2.77%	2.06%	1.06%	0.88%	0.61%	0.50%
Top3	3.1868%	73.61%	17.64%	5.42%	1.99%	0.75%	0.24%	0.14%	0.10%	0.10%	0%
Top5	1.4129%	86.22%	11.07%	2.09%	0.62%	0%	0%	0%	0%	0%	0%

Table 7-4: Misclassification rates for testing set consisting of 91440 examples. Training carried out for accurate state estimates with confidence limits and variable Θ.

Mem-bership values	Overall misclassi-fication	Split of misclassifications according to different levels of leaks									
		2 [l/s]	5 [l/s]	8 [l/s]	11 [l/s]	14 [l/s]	17 [l/s]	20 [l/s]	23 [l/s]	26 [l/s]	29 [l/s]
Highest	17.6345%	31.94%	18.43%	11.88%	8.45%	7.31%	6.13%	5.01%	4.38%	3.51%	2.98%
Top2	6.5420%	54.70%	21.90%	10.10%	4.98%	3.04%	2.19%	1.07%	1.09%	0.45%	0.48%
Top3	3.4547%	70.53%	19.21%	6.46%	2.12%	0.76%	0.32%	0.32%	0.19%	0.09%	0%
Top5	1.5092%	85.65%	11.45%	2.32%	0.58%	0%	0%	0%	0%	0%	0%

Table 7-5: Misclassification rates for testing set consisting of 91440 examples. Training carried out for accurate state estimates with confidence limits and Θ=0.01.

The first row in Table 7-4 illustrates the overall rate of misclassified patterns for the class with the highest membership value. This is equivalent to the hard decision classifiers that are specifically designed to choose only one class which is closest to the input pattern. The rate of almost 17% of misclassified testing patterns leaves some room for improvement although over 62% of all those misclassifications were recorded for patterns representing leakages of magnitude less or equal to 8 [l/s]. It is interesting to note that as much as 56% of all 2 [l/s] leakages from the testing set were misclassified. Let us, however, remember that the variation of some consumptions can be as much as 14 [l/s] which can easily hide the 2 [l/s] leakage. Nevertheless, it is clear that the hard classifier is not the best option in this case. The subsequent rows of the Table 7-4 illustrate the flexibility of the recognition system developed in this research. In contrast to the hard decision classifiers a number of alternatives can be easily obtained and sorted with respect to the membership values. Utilising this property the tests for the top 2, 3 and 5 alternatives have been carried out and misclassification rates calculated. The overall misclassification rate has been dramatically improved by only looking at the top 2 alternatives and reduced to 6.11%. When the top 5 alternatives have been considered the overall misclassification fell to 1.51% and practically there were no misclassifications for leakages larger or equal to 11 [l/s].

While the recognition or misclassification rate is a universally accepted indicator of the pattern recognition system performance some other statistics can be very useful in an attempt to understand and explain these results. One such a statistic is presented in Table 7-6 where the average number of classes with membership values larger than 0.75 for different locations and magnitudes of leakages, represented by patterns from the testing set, is presented.

To understand the meaning of the numbers in Table 7-6 let us analyse the first row which says that if the pipe 2 has a leakage of 2 [l/s] then the processing of the corresponding input pattern by the recognition system will result, on average, in 22 classes with membership values larger than 0.75. In other words, the input pattern is not distinctive enough to be classified, with a reasonable level of confidence, as a leakage in pipe 2 since there are 21 other locations (pipes) that can be said to be viable alternatives.

As one would expect, the number of alternatives decreases with the increase of magnitude of leakage. However, the speed of this decrease is not uniform throughout the network and while for some locations the number of alternatives can be reduced to one or two for relatively small leakages (e.g. pipe 10), for others, even for large leakages, there are still four or five possibilities (e.g. pipe 24).

To illustrate further the difference in the ability to locate a leakage in different parts of the network and to give an example of the type of output from the recognition system the results of two extreme cases are shown in Table 7-7 and Table 7-8.

In the first case,Table 7-7, patterns representing leakages in pipe 24 were analysed and the membership values for all 39 classes and the whole range of leakage magnitudes were found. All the membership values larger than 0.75 are highlighted. It can be seen that the number of alternatives with high degree of membership for the smallest leakage is rather large and even for the largest leakage the number of possible leaking pipes is reduced only to four. On the other hand, the consistency with which the correct area of the network is picked as the possible region of anomaly occurrence is a very good result in itself.

Pipe no.	Magnitude of leakage									
	2 [l/s]	5 [l/s]	8 [l/s]	11 [l/s]	14 [l/s]	17 [l/s]	20 [l/s]	23 [l/s]	26 [l/s]	29 [l/s]
2	22	5	2	1	1	1	1	1	1	1
3	25	10	6	3	3	2	1	1	1	1
4	28	12	7	5	4	3	3	2	2	2
5	31	14	8	6	4	3	3	2	2	2
6	35	11	5	3	2	2	1	1	1	1
7	27	11	6	5	4	3	3	2	2	2
8	29	13	8	5	4	3	2	2	1	1
9	31	14	8	6	4	3	3	3	2	2
10	5	2	2	2	2	1	1	1	1	1
11	34	12	5	2	1	1	1	1	1	1
12	34	13	7	5	3	1	1	1	1	1
13	30	10	5	3	2	2	2	2	2	2
14	25	5	3	2	1	1	1	1	1	1
15	32	13	7	4	3	2	1	1	1	1
16	29	8	4	3	2	2	2	2	2	2
17	34	12	4	2	1	1	1	1	1	1
18	35	13	4	2	1	1	1	1	1	1
19	33	14	6	4	3	2	2	1	2	2
20	30	16	13	10	6	4	3	2	2	2
21	31	16	11	7	4	2	2	1	1	1
22	25	13	11	8	6	5	5	4	4	4
23	21	6	4	2	1	1	1	1	1	1
24	25	13	10	8	6	5	5	5	4	4
25	25	14	11	9	7	6	5	5	5	4
26	24	13	8	6	5	4	4	4	4	4
27	32	15	9	6	3	2	2	2	2	1
28	34	8	3	2	2	2	1	1	1	1
29	31	10	7	5	5	4	3	2	2	1
30	34	15	6	4	3	2	1	1	1	1
31	29	15	11	7	5	4	3	3	2	2
32	28	15	12	10	8	6	4	4	4	3
33	28	13	7	4	3	3	2	2	2	2
36	29	15	12	9	6	5	4	4	3	2
37	27	14	12	10	8	7	6	5	4	3
39	5	2	2	2	1	1	1	1	1	1
40	32	15	8	5	3	2	2	2	2	2
41	31	14	7	6	4	3	2	2	2	2
42	31	10	5	2	1	1	1	1	1	1

Table 7-6: Average number of classes with membership value larger than 0.75 obtained for testing set consisting of 91440 examples.

Class Index	Magnitude of leakage									
	2 [l/s]	5 [l/s]	8 [l/s]	11 [l/s]	14 [l/s]	17 [l/s]	20 [l/s]	23 [l/s]	26 [l/s]	29 [l/s]
1	0.3502	0	0	0	0	0	0	0	0	0
2	0.6749	0.1806	0	0	0	0	0	0	0	0
3	0.7276	0.3098	0	0	0	0	0	0	0	0
4	0.7492	0.3634	0	0	0	0	0	0	0	0
5	0.7666	0.3928	0.0340	0	0	0	0	0	0	0
6	0.8595	0.6499	0.4375	0.2566	0.0971	0	0	0	0	0
7	0.7430	0.3471	0	0	0	0	0	0	0	0
8	0.7579	0.3841	0	0	0	0	0	0	0	0
9	0.7716	0.2399	0.0304	0	0	0	0	0	0	0
10	0.4540	0	0	0	0	0	0	0	0	0
11	0.7988	0.4872	0.1646	0	0	0	0	0	0	0
12	0.7942	0.1626	0	0	0	0	0	0	0	0
13	0.5832	0	0	0	0	0	0	0	0	0
14	0.3736	0	0	0	0	0	0	0	0	0
15	0.7536	0.0132	0	0	0	0	0	0	0	0
16	0.5188	0	0	0	0	0	0	0	0	0
17	0.7362	0.0428	0	0	0	0	0	0	0	0
18	0.8292	0.5668	0.2964	0	0	0	0	0	0	0
19	0.8841	0.7129	0.5387	0.3959	0.2745	0.1222	0.0027	0	0	0
20	0.9491	0.8810	0.8100	0.7698	0.7510	0.7010	0.6839	0.6516	0.6336	0.6554
21	0.9288	0.8216	0.7059	0.6140	0.5370	0.4199	0.3267	0.2203	0.1807	0.0984
22	0.9918	0.9878	0.9757	0.9835	0.9396	0.9238	0.9117	0.9486	0.8725	0.8787
23	0.6340	0.1492	0	0	0	0	0	0	0	0
24	0.9941	0.9937	0.9823	0.9798	0.9417	0.9352	0.9353	0.9682	0.9312	0.9604
25	0.9851	0.9716	0.9483	0.9464	0.9579	0.9268	0.9181	0.8939	0.9344	0.9301
26	0.9939	0.9699	0.9455	0.9086	0.8522	0.8297	0.8152	0.8548	0.7853	0.8027
27	0.9153	0.7680	0.7612	0.3209	0.1726	0.0388	0	0	0	0
28	0.8022	0.5558	0.3314	0.2355	0.0768	0	0	0	0	0
29	0.8900	0.7342	0.6162	0.5006	0.4266	0	0.1864	0	0	0
30	0.8442	0.7422	0.5413	0.3001	0.0633	0	0	0	0	0
31	0.9463	0.8780	0.7827	0.7436	0.7017	0.3273	0.5372	0.3435	0.3405	0
32	0.9564	0.8983	0.8306	0.7845	0.7519	0.6340	0.6244	0.5567	0.5537	0.3976
33	0.9573	0.8921	0.7831	0.7262	0.7501	0.2882	0.6228	0.5346	0.5742	0.0903
36	0.9460	0.8716	0.7875	0.7250	0.6305	0.0194	0.4617	0.1261	0.1556	0
37	0.9713	0.9355	0.8890	0.8632	0.8499	0.7933	0.7581	0.7066	0.7188	0.6850
39	0.4931	0	0	0	0	0	0	0	0	0
40	0.9134	0.8100	0.7117	0.1844	0.0376	0	0	0	0	0
41	0.7744	0.1275	0	0	0	0	0	0	0	0
42	0.5602	0	0	0	0	0	0	0	0	0

Table 7-7: Examples of membership values for testing patterns representing leakages in pipe 24 ranging from 2 to 29 [l/s].

Class index	Magnitude of leakage									
	2 [l/s]	5 [l/s]	8 [l/s]	11 [l/s]	14 [l/s]	17 [l/s]	20 [l/s]	23 [l/s]	26 [l/s]	29 [l/s]
1	0.4426	0	0	0	0	0	0	0	0	0
2	0.7618	0.4671	0.1374	0	0	0	0	0	0	0
3	0.7092	0.3379	0	0	0	0	0	0	0	0
4	0.6876	0.2842	0	0	0	0	0	0	0	0
5	0.6702	0.2413	0	0	0	0	0	0	0	0
6	0.5774	0	0	0	0	0	0	0	0	0
7	0.6938	0.3005	0	0	0	0	0	0	0	0
8	0.6789	0.2635	0	0	0	0	0	0	0	0
9	0.6652	0.2290	0	0	0	0	0	0	0	0
10	0.9828	0.9853	0.9872	0.9918	0.9548	0.9735	0.9764	0.9467	0.9400	0.9853
11	0.6380	0.1604	0	0	0	0	0	0	0	0
12	0.6426	0	0	0	0	0	0	0	0	0
13	0.6587	0	0	0	0	0	0	0	0	0
14	0.5929	0	0	0	0	0	0	0	0	0
15	0.6618	0.0407	0	0	0	0	0	0	0	0
16	0.6577	0	0	0	0	0	0	0	0	0
17	0.6209	0	0	0	0	0	0	0	0	0
18	0.6076	0.0808	0	0	0	0	0	0	0	0
19	0.5527	0	0	0	0	0	0	0	0	0
20	0.4877	0	0	0	0	0	0	0	0	0
21	0.5080	0	0	0	0	0	0	0	0	0
22	0.4450	0	0	0	0	0	0	0	0	0
23	0.8028	0.4985	0	0	0	0	0	0	0	0
24	0.4427	0	0	0	0	0	0	0	0	0
25	0.4517	0	0	0	0	0	0	0	0	0
26	0.4316	0	0	0	0	0	0	0	0	0
27	0.5216	0	0	0	0	0	0	0	0	0
28	0.6346	0.0710	0	0	0	0	0	0	0	0
29	0.5385	0	0	0	0	0	0	0	0	0
30	0.5926	0	0	0	0	0	0	0	0	0
31	0.4905	0	0	0	0	0	0	0	0	0
32	0.4805	0	0	0	0	0	0	0	0	0
33	0.4795	0	0	0	0	0	0	0	0	0
36	0.4909	0	0	0	0	0	0	0	0	0
37	0.4655	0	0	0	0	0	0	0	0	0
39	0.9436	0.9247	0.8703	0.8858	0.9152	0.8940	0.1054	0	0	0
40	0.5234	0	0	0	0	0	0	0	0	0
41	0.6624	0.1551	0	0	0	0	0	0	0	0
42	0.6364	0	0	0	0	0	0	0	0	0

Table 7-8: Examples of membership values for testing patterns representing leakages in pipe 10 ranging from 2 to 29 [l/s].

In the second case, Table 7-8, patterns representing leakages in pipe 10 were considered. In contrast to the previous case, here there is no problem in deciding what is the location of the leakage since the number of alternatives is reduced to two even for small leakages with class 10 (leakage in pipe 10) dominating throughout the whole range of leakages.

Inclusion of patterns representing wrong statuses of valves

So far the recognition system has been trained and tested for one type of anomaly: leakages. The training data for wrong statuses of valves have been deliberately left aside in order to test the recognition system's ability to expand and include new data without the need for retraining the existing system from the beginning.

A set of 96 new input patterns representing wrong statuses of the four control valves (Figure 7-3) have been presented to the system consisting of six "experts" trained to recognise normal operating state and leakages in 38 pipes for 24 hour period of operations. As a result in each of the "expert" networks a number of hyperbox (second layer) nodes and four class (output layer) nodes have been added. The hyperbox nodes were used to store information about these new input patterns while each of the four new class nodes represented the wrong status of one of the four valves.

Not only the extension to the neural networks in order to include new classes was a straightforward task but testing has shown that the performance of the recognition system have not been compromised at all. A simple explanation to this is the fact that leakages and wrong status of a valve have different effect on the network pattern. While a leakage can result only in pressure drop, the effect of a closed valve or blocked pipe is the increase of pressure on one side of the valve and decrease of pressure on the other side.

7.3.6. Residuals and classification

Residuals have been traditionally preferred in bad data analysis. The reason for this is the fact that a residual is a mismatch between the actual measurements and the value of the measured quantity as computed by the estimation algorithm (see (EQ 4-1) in Chapter 4). This mismatch can be viewed as the amount that the measurement model cannot account for. If it is assumed that the model represents the system with the expected accuracy, then one can think of the residuals as estimates of the measurement errors. Since certain assumptions about the measurement errors are made when the model is established, it should be expected that, if no anomaly is present, the residuals will tend to behave in a manner that confirms these assumptions.

Theoretically, for an accurate model of the network and accurate measurements all the residuals should be zero irrespectively of the operating state (e.g. night load, peak load etc.). This property makes the use of residuals in bad data analysis very desirable since the presence of any anomalies could be detected by monitoring the deviation of the residuals from this zero reference point. In theory it should also be possible to locate faults by investigating individual residuals or patterns of residuals.

Similarly to the training set consisting of state estimates, the training set comprising residuals has been split into the six parts according to the six characteristic inflow patterns shown in Figure 7-5. The residuals representing mass balances in network nodes with corresponding confidence limits have been used. The training data have been scaled and mapped onto the [0,1] range. The training has been completed after one run through the training data.

Unfortunately, the testing showed a very poor recognition rate with a high number of input patterns representing large leakages being misclassified. We concluded that this poor performance was due to the inability of the training algorithm to resolve overlappings in a robust way. This, in turn, was caused by the fact that for typical variabilities of consumptions and inaccuracies of meters encountered in water distribution systems, the ratio of noise (quantified as confidence limits) to the useful signal (value of the residual that would result from the occurrence of an anomaly) was very high. In terms of the recognition system it means that there is a large number of input hyperboxes concentrated around the zero reference point with big overlapping regions. This renders an attempt to resolve overlappings impractical since too much information is lost in the process.

Examples of the behaviour of residuals for different levels of leakages between nodes 3 and 4 are shown in Figure 7-10. Figure 7-10a presents an example of residuals found in the course of state estimation carried out for accurate measurements. As one can see, there is a steady divergence from the zero reference point ("normal operating state") with the increase of the magnitude of leakage. An example of influence of the typical random measurement errors on these "accurate" residuals can be seen at Figure 7-10b. The monotonic trend caused by the leakage is severely distorted by the measurement noise. Finally, Figure 7-10c shows effective ranges within which one of the residuals can vary.

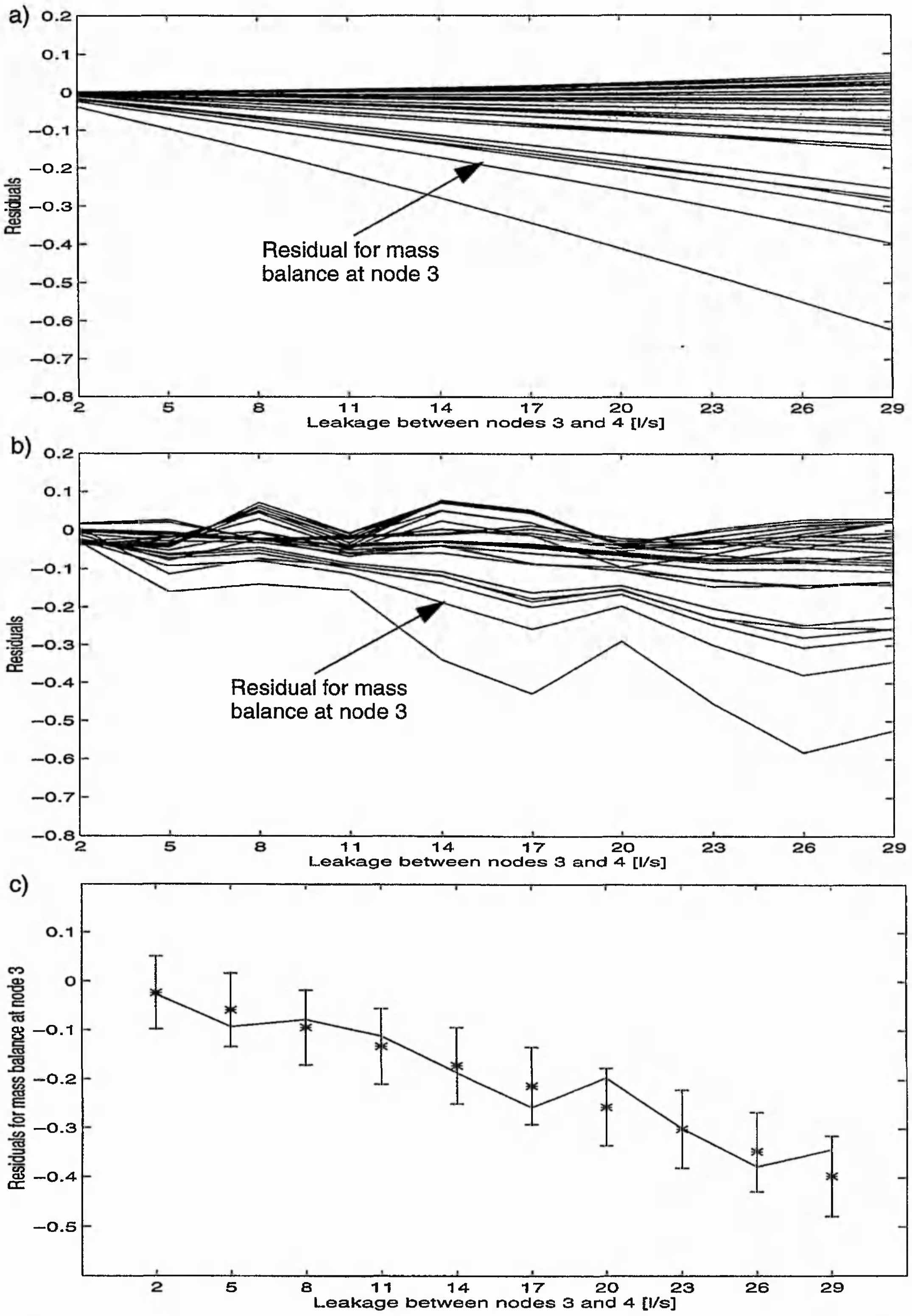


Figure 7-10: Examples of residuals for different levels of leakages between nodes 3 and 4; a) residuals found for accurate measurements; b) residuals affected by typical measurement inaccuracies; c) example of confidence limits for residual representing mass balance at node 3: accurate values marked with "*" and equivalent noisy ones represented by solid line.

Since the loss of information through the attempts to resolve overlappings for the training set was such that the recognition rate was unacceptable in the next experiment the attempts to resolve any overlappings have been abandoned.

The only controlling parameter restricting clusters from free growing was the maximum size of the hyperbox Θ. The multiple full memberships for the testing data were the expected consequence of this action. With the multiple full memberships the graded output of the recognition system allowing to produce a ranking of most likely classes (see Table 7-7 and Table 7-8) had to be to a large extent sacrificed. The fact that processing of an input pattern was likely to produce the highest membership value in a number of classes meant that all the classes had to be treated as equally feasible alternatives.

The average number of alternatives based on examining classes with full membership values for the testing set consisting of 91440 examples is shown in Table 7-9.

Although, in general, the results are worse than for the state estimates discussed in the previous sections, some interesting features can be noticed. Firstly, there is a strong correlation between the ability to restrict the suspected leakage area to a small number of pipes and the proximity to accurate meters and an anticorrelation with the proximity to the least accurate consumption nodes. So, the leakages in pipes 12, 13, 14, 15, 16, 17, 41 and 42 that are quite far from the consumption nodes (main source of uncertainty) and in the vicinity of few accurate meters, can be detected and located much more easily than the leakages in pipes 21, 22, 24, 25, 26, 28, 29, 30, 31, 32, 33, 36 and 37. Looking at the results for pipe 10 and 23 the importance of accurate measurements is very evident. Although these pipes lay in the region where it is the most difficult to detect and locate a leakage the benefit of three accurate measurements (heads at nodes 2 and 31 and inflow at node 31) is evident.

The average numbers of alternatives shown in Table 7-9 were calculated for 24 hour period and, as such, can be treated as an indicator of how well, on average, the recognition system performs. It has been an accepted engineering knowledge that there are periods during the day when it is easier to detect leakages than in other times. The fact that consumptions during the night are at the lowest level but the leakages stay virtually the same has been widely exploited when performing a "corrective maintenance" of a distribution system.

Pipe no.	Magnitude of leakage									
	2 [l/s]	5 [l/s]	8 [l/s]	11 [l/s]	14 [l/s]	17 [l/s]	20 [l/s]	23 [l/s]	26 [l/s]	29 [l/s]
2	36	34	29	21	15	12	9	7	6	4
3	36	35	30	22	17	13	10	7	7	5
4	37	32	25	17	12	9	8	6	5	4
5	35	24	13	9	7	5	4	3	3	2
6	36	37	35	34	29	25	20	15	12	9
7	36	35	31	24	17	13	10	8	7	6
8	37	31	22	15	11	8	6	5	4	3
9	34	19	12	8	5	4	3	3	3	2
10	34	27	16	7	3	2	2	1	1	1
11	36	36	33	27	22	17	13	9	7	6
12	34	15	6	4	3	2	1	1	1	1
13	28	8	4	3	3	3	3	2	3	2
14	21	4	3	3	3	2	2	1	1	1
15	33	14	8	5	4	2	2	1	1	1
16	26	6	3	3	3	3	3	3	3	2
17	31	10	5	3	2	2	1	1	1	1
18	36	23	12	7	6	5	4	3	2	2
19	36	36	29	18	11	7	5	4	3	2
20	35	35	29	20	14	10	8	6	4	2
21	35	36	36	34	30	26	21	17	12	9
22	35	34	34	30	26	22	16	13	10	8
23	35	27	18	11	4	3	2	2	1	1
24	35	35	33	30	27	23	19	16	13	10
25	35	35	33	31	27	25	20	17	14	12
26	35	35	33	29	27	22	17	14	10	8
27	37	32	25	22	12	7	5	3	2	2
28	35	35	33	30	26	21	18	13	11	8
29	35	35	33	29	26	22	18	14	10	8
30	35	35	34	31	25	19	16	10	7	4
31	35	35	33	30	28	23	20	17	14	11
32	35	35	34	32	28	27	24	19	18	16
33	35	35	34	31	29	25	22	19	17	14
36	34	34	33	29	26	22	19	16	12	10
37	35	35	34	32	28	27	22	19	16	14
39	35	30	24	18	13	8	6	4	2	2
40	37	28	19	13	7	4	3	3	2	2
41	33	19	10	7	5	4	3	3	2	1
42	28	9	4	2	2	2	2	2	2	2

Table 7-9: Average number of alternatives based on the number of classes with full membership value.

Table 7-10 presents detailed results for an 11 [l/s] leakage in pipe 10 using 9 data sets in which random measurement errors are superimposed on the readings for each hour of the 24 hour period. Essentially the table illustrates that the recognition system based on residuals is strongly susceptible to measurement errors. Rather than try to assess the overall performance over the 24 hour period, individual results and average values for the six characteristic periods of operation are examined.

Hour	Number of alternatives for different random sets of measurement errors									Average no. of alternatives for 6 characteristic periods	Overall average no. of alternatives
1	1	3	2	2	1	2	3	2	1	2	7
2	2	2	2	2	2	2	3	2	2		
3	2	3	3	2	3	2	3	2	3		
4	4	2	2	3	2	3	2	3	3		
5	2	2	2	3	3	2	2	2	1		
6	7	12	35	8	5	5	4	33	5	12	
7	3	5	7	36	6	16	6	37	18		
8	6	2	39	2	2	4	3	5	3		
9	4	39	31	5	39	39	5	34	38	14	
10	30	5	29	5	27	6	4	2	4		
11	5	2	5	1	5	1	39	29	5		
12	10	5	4	5	38	5	5	5	4		
13	3	9	4	5	29	5	2	5	4	8	
14	3	5	4	4	5	3	3	5	29		
15	3	9	4	32	3	4	5	27	5		
16	5	3	5	9	27	5	31	5	4		
17	6	3	3	3	4	5	4	5	1		
18	3	9	7	36	8	5	8	3	8	7	
19	6	5	7	5	8	8	11	8	5		
20	6	5	5	11	5	4	6	6	4		
21	2	5	3	2	5	7	4	2	4	4	
22	4	2	3	4	4	3	3	4	2		
23	3	4	3	4	3	4	2	4	3		
24	4	6	5	3	4	1	8	5	4		

Table 7-10: Examples of number of alternatives produced by the recognition system for the input patterns from testing set representing leakage of 11 [l/s] magnitude in pipe 10.

Firstly, looking at average number of alternatives for the six periods one can see the stark difference between the performance of the recognition system for the first period (hours 1 to 5 - "night load") where the variations of consumptions are smallest and the third period (hours 9 to 12 - "peak load") where the variations of consumptions are largest.

Secondly, looking at the individual results for the "peak load" period one can see a huge influence of the measurement errors. On the one hand, they can help to locate the leaking pipe when they push the residuals out from the zero reference point but, on the other hand, they can also make the detection of even a large leakage impossible when they reduce its distinctive "fingerprint" by pulling the residuals towards the zero reference point.

7.3.7. Detection of anomaly based on residuals

The neural network recognition systems discussed in the previous sections have been trained for data including examples of both normal and anomalous modes of operation. Consequently, the recognition systems combined the ability to detect and identify (locate) the anomaly at the same time.

In the literature, however, the problems of detection and identification are very often treated separately. One could argue that for some problems, especially those where faults occur rarely, it might be beneficial to be able to first detect faults and only if they are present to try to identify them.

As it was explained before, residuals analysis is a particularly attractive option in fault detection procedures since, irrespectively of the time of the day, if no anomaly is present the residuals should always be close to zero. The question: How close? depends on the accuracy of the measurements but this can be quantified by the confidence limit analysis.

A neural network fault detection system based on residuals is essentially a simpler version of the recognition system discussed in the previous section. Since one is only concerned with distinguishing between the presence and absence of faults, the detection system consists of just one class representing the normal operating state. Depending on the membership value calculated for this class the input pattern is classified as a normal operating state or a fault.

The training data set consisted of all the residuals vectors with their confidence limits representing normal operating state from the original training data set. The data have been scaled and mapped into the [0,1] range with zero points placed in the middle of this range.

The detection system consisted of six hyperboxes, one for each of the six characteristic operation periods during the day.

The normal operating states for "night load" period and the "peak load" period, pictured in form of vertical bars, and examples of the input patterns representing a leakage in pipe 2, marked by '*', are shown at Figure 7-11. Once again, the tighter confidence limits for the "night load" period are the reason for the better performance of the detection system during this particular period compared to the rest of the day.

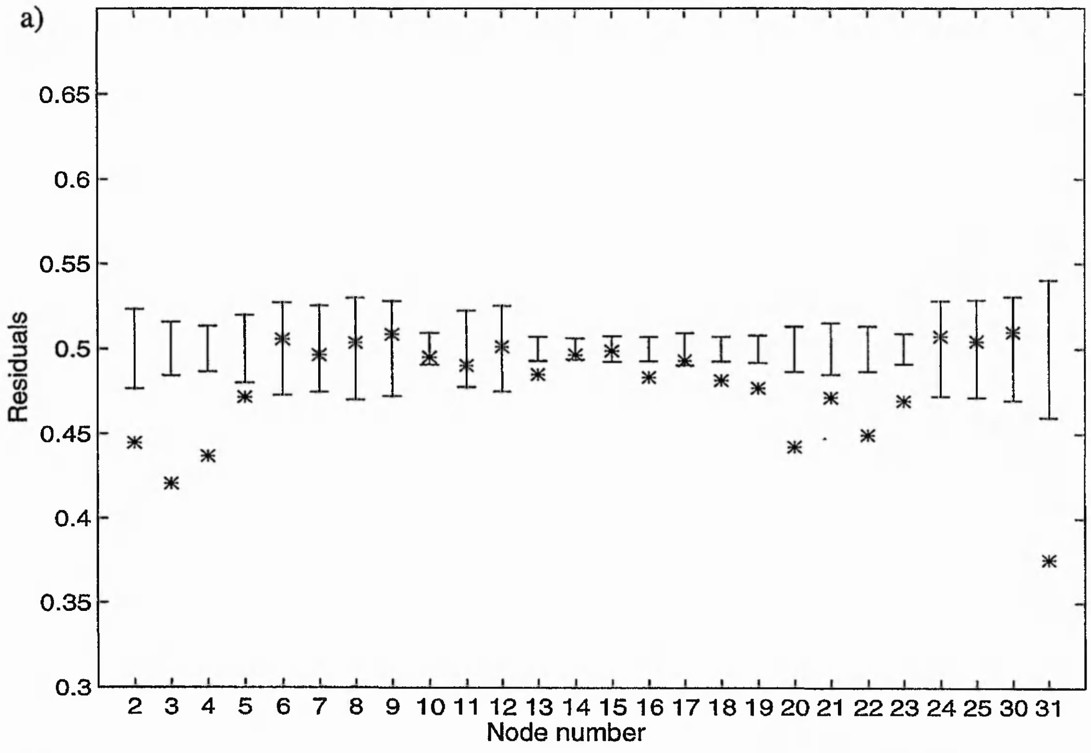
A very practical point is that the graphical representation of the hyperboxes and the residuals makes it quite easy to understand the reason for success or failure of fault detection. The retrieving of the lower and upper bounds for each residual is a straightforward procedure since they are stored in the neural network as the weights for minimum and maximum points defining the hyperbox.

The test has been performed for the full set of 91200 examples representing leakages taken from the test set described earlier in this chapter. The results for the "night load" period are shown in the form of percentages of detected faults for each pipe and 10 levels of leakages in Table 7-11.

These results are effectively the confirmation of what has been discovered in the previous experiments. For some leakage locations the 100% detection rate was achieved for leakages as small as 5 [l/s] while for other locations the 100% detection rate could not be accomplished even for leakages of 20 [l/s]. In a large number of locations the detection of leakages of magnitude of 5 [l/s] or smaller is practically impossible. For roughly half of the network pipes the threshold for which 50% or more of the leakages are detected is 8 [l/s]. The vast majority of 14 [l/s] leakages could be detected in all possible locations in the network.

As one can see not only the identification of faults can cause problems but even their detection is not a simple task when one has to deal with high levels of uncertainty so common in public utility systems.

a)



b)

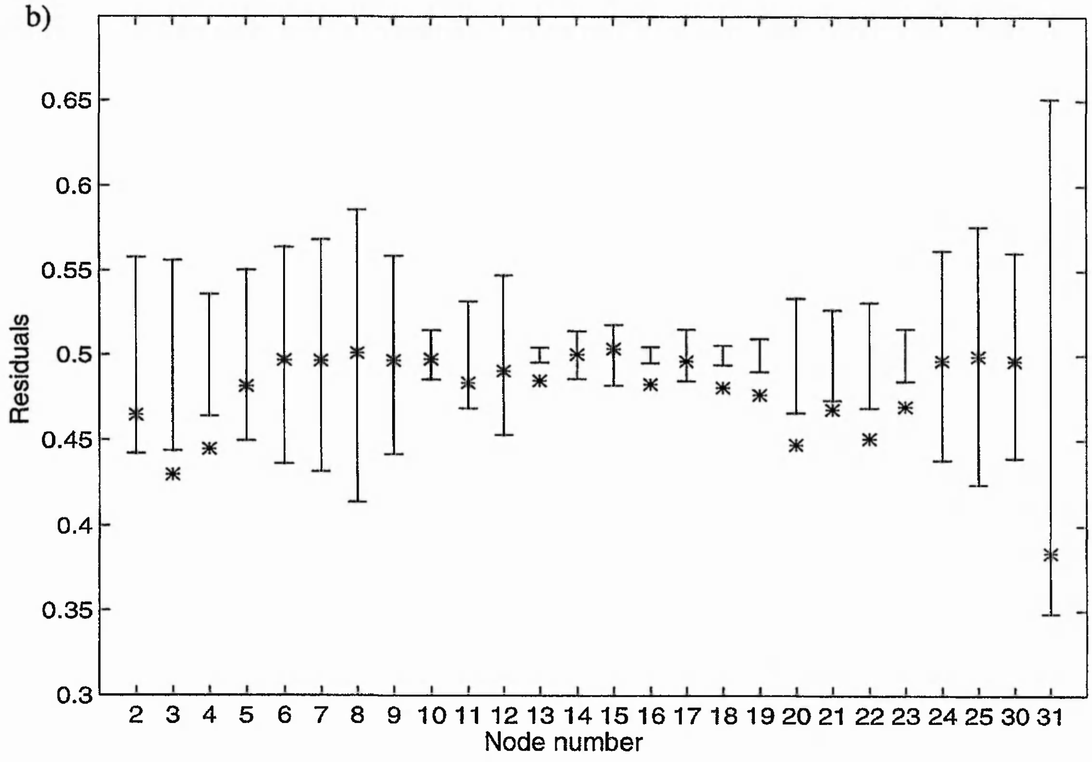


Figure 7-11: The visualisation of boundaries (vertical bars) for normal operating state for: a) night and b) peak load. Examples of input patterns representing leakage in pipe 2 (29 [l/s]) marked by * are used to illustrate the ability to more easily detect leakages at night due to tighter confidence limits resulting from lower level of consumption variations.

Pipe no.	Magnitude of leakage									
	2 [l/s]	5 [l/s]	8 [l/s]	11 [l/s]	14 [l/s]	17 [l/s]	20 [l/s]	23 [l/s]	26 [l/s]	29 [l/s]
2	0	22	76	100	100	100	100	100	100	100
3	0	2	47	98	100	100	100	100	100	100
4	0	24	60	98	100	100	100	100	100	100
5	20	60	98	100	100	100	100	100	100	100
6	0	2	13	31	64	80	98	100	100	100
7	0	2	56	96	100	100	100	100	100	100
8	7	24	82	98	100	100	100	100	100	100
9	24	87	100	100	100	100	100	100	100	100
10	4	69	100	100	100	100	100	100	100	100
11	0	7	38	71	93	100	100	100	100	100
12	24	93	100	100	100	100	100	100	100	100
13	49	100	100	100	100	100	100	100	100	100
14	71	100	100	100	100	100	100	100	100	100
15	33	100	100	100	100	100	100	100	100	100
16	58	100	100	100	100	100	100	100	100	100
17	40	98	100	100	100	100	100	100	100	100
18	11	51	98	100	100	100	100	100	100	100
19	0	4	47	76	98	100	100	100	100	100
20	0	7	49	64	91	100	100	100	100	100
21	0	0	11	33	62	82	98	100	100	100
22	0	2	24	56	82	98	100	100	100	100
23	2	47	100	100	100	100	100	100	100	100
24	0	0	29	64	84	96	100	100	100	100
25	0	2	36	42	71	91	100	100	100	100
26	0	4	31	67	73	98	100	100	100	100
27	16	29	60	87	100	100	100	100	100	100
28	0	0	18	40	71	91	100	100	100	100
29	0	2	22	42	76	100	100	100	100	100
30	0	2	13	36	62	98	100	100	100	100
31	0	4	24	44	69	96	100	100	100	100
32	0	0	18	40	62	78	100	100	100	100
33	0	4	24	40	60	87	100	100	100	100
36	0	7	40	60	62	96	100	100	100	100
37	0	2	27	44	73	84	98	100	100	100
39	2	73	100	100	100	100	100	100	100	100
40	7	47	64	96	100	100	100	100	100	100
41	27	84	100	100	100	100	100	100	100	100
42	64	100	100	100	100	100	100	100	100	100

Table 7-11: Percentage of the fault detection based on examining the "normal operating state" class membership values for the "night load" period (hours 1 to 5 am).

7.4. Discussion and conclusions

An important notion tested in this research was that artificial neural networks have a potential to mimic, to a large extent, the high level information processing performed by human operators. On the basis of the material included in the last two chapters concerning the development and application of a neural recognition system, we can examine how far the analogy between human operators and ANNs could be carried.

It can be safely said that any person employed as an operator of the water distribution system has to be first trained to be able to perform the expected tasks well. This usually involves a stage of getting familiar with the specific features of a particular distribution system i.e. the locations of reservoirs, types and characteristic operations of pumping stations, locations of control valves, locations of major consumption points etc. There is also likely to be a stage of learning what methods and equipment is used to monitor and control the network and this could include a familiarisation with the specific estimation method and/or other software used. Once the initial training is completed the learning continues through the day to day operations.

By analogy, there are also two major modes of operation of ANNs: the network training mode and the proper operation mode. The information about the specific distribution system (e.g. topology of the network) is included implicitly in the mathematical model. The data processing methods (e.g. type of estimation procedure) and the equipment characteristics (e.g. accuracies of meters) are reflected in the training data. While some ANNs are very inflexible when it comes to learning new patterns while in operation (due to a complicated training algorithm or a rigorous specification of the architecture) the neural systems developed during this research have the ability to learn on-line and expand to include new information and to reinforce the stored knowledge. This is analogous to the human operator gaining experience as a result of the long term exposure to everyday water system operations.

The analogies between the operation of the neural system and human operators could be shown for a number of smaller points but they all stem from the two important features of the ANN. Firstly, instead of a sequential, strictly algorithmic analysis the signals are processed in groups and looked at as patterns which is much closer to the way humans tend to handle large dimensional data. Secondly, the neural system has the ability to grow, include new information, gain experience. Whichever name we choose there simply is a capacity for building on the top of the existing abstraction of the system.

This research has demonstrated that both state estimates with their confidence limits and residuals with their confidence limits can be successfully used to train the neural recognition system, although each approach has its own advantages and disadvantages.

A great advantage of processing residuals is the fact that irrespectively of the operating state, when there is no anomaly present in the system, all the residuals are zero, or in practical terms, they are contained within the confidence limits. This represents a universal reference point (hyperbox) that can be labelled as normal operating state. All the other patterns representing leakages and other malfunctions could be mapped into the space around this zero reference point.

Unfortunately, it has also been found that residuals are very susceptible to typical levels of measurement noise present in water systems. A high ratio of noise to useful signal seriously hampered the ability of the recognition system based on residuals to detect and identify the cause of abnormal operation. In particular, it was only possible to reduce the suspected location of some leakages to a certain area of the network rather than one or two pipes.

Much better results have been obtained for the system based on state estimates. The overlapping regions of different classes could be resolved in much more robust way. This was due to the fact that the state estimates can differ significantly from one hour of operation to another, resulting in spatially separated input patterns. Also, the ratio of the measurement noise to the useful signal was much smaller for state estimates than for residuals. However, the very fact that state estimates are different for different operational periods, means that there is no universal reference point representing normal operating state. Instead there are normal operating states for different load patterns which are represented by different hyperboxes. This also means that while the performance of the recognition system is very good for a range of typical operating states for which it has been trained, the occurrence of a completely new pattern does not result in any meaningful classification, though the system can remember this new pattern for subsequent labelling. This is similar to a human response to a new object. Although one does not know how to classify the new object, one is able to remember its shape and other characteristic features for later referencing or subsequent association with other objects (naming).

As one can see each of the approaches has its strengths and limitations. In a way, however, they complement each other and the use of both of them at the same time should result in better overall performance.

One of the benefits of training and testing the recognition systems for the 24 hour period of operations was the opportunity to establish the role of accurate measurements in the process of detecting and identifying faults. The ability to detect and correctly locate smaller leakages in the direct neighbourhood of accurate meter(s) was the recurring theme for all the recognition systems.

Another important factor that can play a crucial role in the performance of the recognition system is the choice of the estimation procedure. In this work the LS estimator was used. As it was explained in Chapter 4 the LS estimator can be very much affected by the presence of large errors in the data, but at the same time, it is the most robust procedure in a sense of convergence. While it has to be said that the use of some other estimation procedure (e.g. LAV) could result in even better recognition rates, especially in the regions of the network where local measurement redundancy is high enough for erroneous data to be completely rejected, there is a danger that the estimation procedure will not converge at all when the local measurement redundancy is not available. However, the use of the estimator that would be able to reduce the influence of the largest measurement errors and still retain the robust behaviour of the LS procedure in terms of convergence would definitely improve the recognition rates.

One last feature of the neural recognition system, that has not yet been mentioned in this section, is the fuzzy character of its output. In our opinion the ability to produce a graded response is a very powerful tool. A lot of additional information can be extracted from the fuzzy classification output. For instance, although the information about the level of leakage has not been included in the training set the estimate of the size of the leakage could be inferred from the membership values. To illustrate the potential benefits of using fuzzy outputs one can imagine a simple two class recognition system with the output $\mathbf{C}=[C_1\ C_2]$. Let us now examine three examples of the output from the fuzzy, $\mathbf{C_F}$, and the hard decision, $\mathbf{C_H}$, classifiers:

a) $\mathbf{C_F}=[0.98\ 0.99]$ and $\mathbf{C_H}=[0\ 1]$;

b) $\mathbf{C_F}$=[0 0.01] and $\mathbf{C_H}$=[0 1]; and

c) $\mathbf{C_F}$=[0 0.9] and $\mathbf{C_H}$=[0 1].

In all three cases the output of the hard classifier based on the principle of choosing one class has been the same. On the basis of these results there is no doubt that the input pattern belongs to the second class. While in fact, in two out of the above three cases, the results from fuzzy classifier indicate that it is not at all such a straightforward answer. From the fuzzy classification results one can see that in the first case both classes are almost equally feasible alternatives while in the second case, all that can be said is that the input pattern is closer to the second class but it can hardly be classified as belonging to this class.

This type of analysis turned out to be very useful in the interpretation of the classification results for water distribution systems presented in this chapter. As a matter of fact many of the conclusions could only be drawn because the graded answers in a form of fuzzy membership values were available in the first place.

As a final statement it can be said that a completely new approach to detection and identification of faults in the water distribution systems based on the fuzzy classification and clustering neural network has been proposed. An assumption that fault diagnosis can be based on pattern analysis without a need to employ any heuristics or specialist knowledge has been confirmed by the computational results. Since the specific topology and measuring equipment information is included in the mathematical models used to generate the input patterns a number of useful studies, i.e. the influence of the levels of uncertainty on the ability to detect and identify leakages in different areas of the distribution network, meter placement or distribution of errors in the network for a given set of measurements, can be carried out using the fuzzy classification results.

Chapter 8

Conclusions and further research

8.1. Conclusions

The aim of this research project was to investigate the applicability of neural networks for the implementation of decision support systems in operational control of industrial processes. The nonlinear models and large scale of the water distribution systems made them both a very challenging problem to be tackled and a very good validation example for the prototype decision support system.

There have been two distinctive parts of the project. In the first part, the usage of neural networks for the optimization based on mathematical model of the water distribution system has been investigated. A particular emphasis has been placed on the application of simple feedback neural networks to the problems of state estimation and confidence limit analysis.

The main thesis of the second part of the project was that the high level of information processing by human operators could be mimicked, to a large extent, by a suitable neural based recognition system. A development of the flexible fuzzy neural network for pattern recognition and its application to the water system state interpretation/classification task have been covered.

Short summaries of the problems uncovered and solutions found in the course of investigations as well as the main conclusions of the project are presented below.

8.1.1. Neural networks and water network state estimation

Over the last two decades state estimators gradually became the key utility for the implementation of monitoring and control of large scale public systems such as water, gas or electric power distribution systems.

In this work it was shown that relatively simple neural networks for solving systems of linear equations can be utilized in the process of the water network state calculations. While simulating these neural networks some important issues involved in the state estimation process have been addressed.

The first of these problems is bad data rejection properties of different estimators. It was shown that by simply changing the parameters of a NN's sigmoid activation function we can switch from LS criterion, trying to satisfy all the equations in the set and therefore being strongly affected by gross errors, to LAV criterion having the ability to discard anomalous data completely. In the above solution the fact that the sigmoid activation function can approximate a linear function (LS criterion) or a signum function (LAV criterion) is utilised. In this way the more flexible neural structure was obtained while overcoming the problem of discontinuities associated with using signum function in the original formulation of the LAV criterion.

In the context of water system state estimation it has been observed that the LAV solution can converge prohibitively slowly or not converge at all under some conditions. This problem can be solved by using neural networks with inhibition principle. Retaining the bad data rejection quality of the LAV criterion the robustness of LS criterion is also preserved.

The ill-conditioned problems are the second major issue that has been addressed in water system state estimation. While all neural networks presented in this work avoid direct inversion of the matrices, which is a major source of estimation errors in a case of ill-conditioned problem, for the purpose of improving the convergence properties and the accuracy of the desired network the augmented Lagrangian with regularization has been used. The resulting neural network is a little more complicated than the previously discussed ones but it showed expected qualities in a case of ill-conditioned problems.

The last issue that may not be that crucial in the case of water systems but could be of paramount importance in some other real time applications is the ability of neural networks to arrive at the solution within predefined period of time. As it was shown here this can be accomplished by adaptively changing some weights during the minimization process.

While the full potential of the neural networks presented in this work can be realized when implemented in hardware, i.e. using VLSI or electro-optical technology, they have been simulated here as dynamical systems built from blocks, like adders, multipliers, integrators or nonlinear transfer functions, in order to facilitate the transformation from simulation models to the hardware implementation. The simulations have been carried out assuming realistic parameters for the equivalent physical elements (i.e. integration times etc.) and thus it has been found out that the time for the state estimate calculations of large water network could be in order of microseconds.

8.1.2. Neural networks and confidence limit analysis

No state estimator can give accurate results from inaccurate data.

This simple statement prompted an investigation (Bargiela & Hainsworth, 1989) into the precise nature and level of the measurement uncertainty impact on the accuracy to which state estimates can be calculated. It is believed that the safety of the operational control can be enhanced when operators are given not only the information about estimates of the current operating state but also an indication of how reliable these estimates are for a given set of measurements at a particular operating state. In confidence limit analysis this information is provided in the form of upper and lower bounds for each state estimate variable.

The best known and mathematically the most reliable method of quantifying the state uncertainty is the Monte Carlo method. The accuracy of this method derives from its full treatment of the network model non-linearity. However, its computational inefficiency renders it unsuitable for on-line control applications.

Using extensive simulations, it has been found out that algorithms based on the linearised network model produce results that compare well with the Monte Carlo results while the computation time is much shorter.

In this work it was shown how the neural networks that have been used to produce state estimates can be used to obtain confidence limits. The first method presents the usage of neural networks to find the sensitivity matrix being a vital part of the sensitivity matrix algorithm. In broader terms the way of finding inverse and pseudoinverse matrices has been demonstrated. The second method utilizes the superposition principle where each disturbance is analysed separately and the partial results are gradually combined to produce overall confidence limits.

Similarly to the state estimation process the confidence limits could be arrived at very quickly once the neural network is implemented in hardware. A dynamical, integrated neural based system for state estimation and confidence limit analysis, using simple elements, has been constructed in order to analyse dynamical properties of the proposed methods.

8.1.3. Neural networks and fault detection

The state estimation is an absolute must in the monitoring and control of modern water distribution (utility) systems if only for ensuring that pressures in the network are not too low for any service disruptions, but at the same time, not too high to cause an unnecessary risk of pipe bursts. While monitoring against the recommended minimum and maximum pressures is the most basic activity there is much more useful information contained in the state estimate vector. In order to be able to exercise a greater control over the distribution network the accurate classification of the situation is needed. Since the safety of operation is of paramount importance the quick detection and correct identification of abnormal events should be one of the main priorities in any complex system.

In this work the interpretation of the water network state has been attempted by employing a fuzzy neural recognition system. A completely new approach to detection and identification of faults based on neural pattern analysis has been presented.

First a new general fuzzy neural network for clustering and classification has been proposed. A number of requirements imposed by the problem (water network state interpretation) was used as a guidance in this neural network development process. Among all the features of this neural network there are five that can be regarded as the most important:

- it can process both deterministic (point in the n-dimensional pattern space) and fuzzy input patterns (described by the lower and upper limits on each input vector variable);
- it combines the supervised and unsupervised learning techniques within a single training algorithm which means that the nonlinear decision boundaries between classes can be formed at the same time as the clusters are formed for the unlabelled data;
- it can grow to meet the demands of the problem since there is no restriction on the number of nodes and new nodes are added to the network when they are needed;

- it can produce hard (where only one class is chosen) and fuzzy classification (where a number of possible alternatives is given); and
- it learns on-line.

This neural algorithm has been subsequently used as a main building block in development of water distribution network fault diagnosis system.

The emphasis has been put on the task of detection and accurate location of topological errors. It has been shown that both the state estimates with their confidence limits and residuals with their confidence limits can be successfully used to train the neural recognition system. However, it has also been found that, due to the high susceptibility of the residuals to the typical measurement noise, the ability to detect and identify faults by the recognition system based on residuals can be seriously reduced. On the other hand, the recognition system based on state estimates performed better for the data for which it has been trained, due to much lower ratio of the noise to the useful signal and larger spatial separation of patterns representing different classes.

The importance of the accurate measurements in the process of detecting and identifying anomalies has also been evident in the testing results. Smaller leakages could be detected and more accurate location of the malfunction could be found in the direct vicinity of accurate meters.

The performance of the recognition systems has been tested for a large number of topological errors (i.e. 10 different levels of leakages have been simulated for all pipes) over the 24 hour period of operations of a realistic water distribution network. This not only allowed to confirm the commonly known fact in the water industry that any faults can be more easily detected during the night period when the variations of consumptions are smallest, but also gave an opportunity to study what impact different operating conditions can have on the ability to diagnose faults.

8.2. Further research

As the simulation studies presented in this thesis have shown there is a huge potential in applying simple neural networks for solving systems of linear equations to complex nonlinear problems. However, since the full potential of these neural networks can only be realised when implemented in hardware, the research on suitable electronic or electro-optical circuits should be considered a priority.

While the GFMM classification and clustering neural network has performed very well for a number of different training and testing examples one drawback has been identified: the hyperboxes created in the training stage depend on the order of presentation of the training patterns. Consequently the final division of the pattern space can be slightly different each time when the order of presentation of the training patterns is changed. Further research should explore the optimisation of the GFMM algorithm, so that the final partitioning of the pattern space is independent of the order of presentation of the training patterns.

Since application of the fuzzy neural recognition systems to the water network state identification task has shown to be successful the following topics deserve further research effort:

- Study of the recognition system performance in association with different state estimation procedures;
- Adaptation of the neural pattern recognition system to the detection of multiple malfunctions;
- Combination of the fuzzy outputs of the recognition systems based on state estimates and different types of residuals (i.e. representing mass balances and measurements).

Since the pattern recognition is a key element in many engineering solutions, the extension of the fuzzy neural algorithm developed here for pattern classification and clustering to control and prediction problems is also suggested.

References

Anderberg, 1973

Anderberg, M., *Cluster Analysis for Applications*. New York: Academic Press, 1973

Anderson, 1995

Anderson, J.A., *An introduction to neural networks*, The MIT Press, 1995

Asfour et al., 1993

Asfour Y.R, G.A.Carpenter, S.Grossberg, G.W.Lesher, "Fusion ARTMAP: an adaptive fuzzy network for multi-channel classification", *Proceedings of the World Congress on Neural Networks (WCNN-93)*, pp.210-215, 1993

Bargiela, 1984

Bargiela, A., *On-line monitoring of water distribution networks*, Ph.D. Thesis, University of Durham, 1984

Bargiela, 1992

Bargiela, A., "Nonlinear network tearing algorithm for transputer system implementation", *Proc. TAPA'92*, pp.19-24, Nov. 1992

Bargiela, 1995

Bargiela, A., "High performance neural optimisation for real-time pressure control", *Electr. Proc. of High Performance Computing Conference, HPC Asia'95,* Taipei, Chap. AL34, pp.1-8, Sept. 1995

Bargiela & Hainsworth, 1988

Bargiela A. and G.D.Hainsworth, "Telemetry system design and on-line decision support with 'TCLAS' software", *Computer Applications in Water Supply*, vol.2, pp.43-58, 1988

Bargiela & Hainsworth, 1989

Bargiela A. and G.D.Hainsworth, "Pressure and Flow Uncertainty in Water Systems", *J. of Water Resources Plan and Management*, vol.115, no 2, pp.212-29, March 1989

Barrodale & Zala, 1986

Barrodale, I. and C.A.Zala, "L_1 and L_∞ curve fitting and linear programming algorithms and applications", in *Numerical algorithms*, Eds. J.L.Mohamed and J.E.Walsh, Oxford: Oxford University Press, pp.220-238, 1986

Baxt, 1990

Baxt, W.G., "Use of artificial neural network for data analysis in clinical decision-making: The diagnosis of acute coronary occlusion", *Neural Computation*, vol.2, pp.480-489, 1990

Bazaraa & Jarvis, 1977

Bazaraa, M.S. and J.J.Jarvis, *Linear programming and network flows*, John Wiley & Sons, 1977

Bazaraa & Shetty, 1979

Bazaraa, M.S. and C.M.Shetty, *Nonlinear programming: Theory and practice*, John Wiley & Sons, 1979

Beck et al., 1983

Beck, P., L.Lasdon, M.Engquist, “A reduced gradient algorithm for nonlinear network problems”, *ACM Trans. Mathematical Software*, vol.9, no.1, pp.55-70, March 1983

Bellman et al., 1966

Bellman, R.E., R.Kalaba, L.A.Zadeh, “Abstraction and pattern classification”, *J. Math. Anal. Appl.*, vol. 13, pp. 1-7, 1966

Bertsekas, 1982

Bertsekas, D.P., *Constrained Optimization and Lagrange Multiplier Methods*, New York: Academic, 1982

Bezdek, 1981

Bezdek, J.C., *Pattern Recognition with Fuzzy Objective Algorithms.* New York: Plenum Press, 1981

Bezdek, 1992

Bezdek, J.C., “Computing with uncertainty”, *IEEE Communications Magazine*, pp. 24-36, September 1992

Bezdek et al., 1986

Bezdek, J.C., S.Chuah, D.Leep, “Generalized k-nearest neighbor rules”, *Fuzzy Sets and Systems*, vol. 18, pp. 237-256, 1986

Bezdek & Harris, 1979

Bezdek, J.C. and J.Harris, “Convex decompositions of fuzzy partitions”, *J.Mathematical Analysis and Applications,* vol. 67, pp. 490-512, 1979

Borkowska, 1974

Borkowska, B., “Probabilistic load flow”, *IEEE Trans. PAS*, vol.PAS-93, no.3, May-June 1974

Bounds et al., 1988

Bounds, D.G., P.J.Lloyd, B.Mathew, and G.Wadell, “A multilayer perceptron network for the diagnosis of low back pain”, in *Proceedings of the IEEE conference on Neural Networks*, San Diego, vol.II, pp.481-489, 1988

Brdys et al., 1990a

Brdys, M.A., N.Abdullah, and P.D.Roberts, “Augmented model-based double iterative loop techniques for hierarchical control of complex industrial processes”, *Int. J. Control*, vol.52, no.3, pp.549-570, 1990

Brdys et al., 1990b

Brdys, M.A., N.Abdullah, and P.D.Roberts, “Hierarchical adaptive techniques for optimizing control of large-scale steady-state systems: optimality, iterative strategies, and their convergence", *IMA Journal of Math. Control & Info.*, vol.7, pp.199-233, 1990

Brdys & Ulanicki, 1994

Brdys, M.A. and B.Ulanicki, *Operational Control of Water Systems: Structures, Algorithms and Applications*, New York: Prentice Hall, 1994

Canu et al., 1990

Canu S., R.Sobral, R.Lengelle, "Formal Neural Networks as an Adaptive Model for Water Demand", *INNC 90 PARIS - INT Neural Network Conference*, vol. 1, pp.131-36, 1990

Carpenter & Grossberg, 1987

Carpenter, G.A. and S.Grossberg, "A Massively Parallel Architectures for a Self-Organizing Neural Pattern Recognition Machine", Computer Vision, Graphics, and Image Processing, vol. 37, pp.54-115, 1987

Carpenter & Grossberg, 1994

Carpenter, G.A. and S.Grossberg, "Fuzzy ARTMAP: A Synthesis of Neural Networks and Fuzzy Logic for Supervised Categorization and Nonstationary Prediction", in *Fuzzy Neural Networks, and Soft Computing,* Edited by R.R.Yager and L.A.Zadeh, pp. 126-166, 1994

Carpenter et al., 1992

Carpenter G.A., S.Grossberg,N.Markuzon,J.H.Reynolds, and D.B.Rosen, "Fuzzy ARTMAP: An Adaptive Resonance Architecture for Incremental Learning of Analog Maps", *IEEE Trans. on Neural Networks*, 1992

Carpentier & Cohen, 1993

Carpentier, P. and G.Cohen, "Applied mathematics in water supply network management", *Automatica*, vol.29, no.5, pp.1215-1250, 1993

Casey, 1992

Casey, T.J., *Water and wastewater engineering hydraulics*, New York: Oxford University Press, 1992

Cesario & Davis, 1984

Cesario, A.L. and J.O.Davis, "Calibrating water system models", *Journal of AWWA*, vol.76, no 2, July 1984

Chadwick & Morfett, 1986

Chadwick, A.J. and J.C.Morfett, *Hydraulics in civil engineering*, London: Allen & Unwin, 1986

Chen, 1988

Chen, Y.C., "Applications of time series analysis to water demand prediction", in *Computer Applications in Water Supply. Volume 1: Systems Analysis and Simulation*, Ed. B.Coulbeck and C.H.Orr, pp.289-296, Research Studies Press Ltd., 1988

Christensen & Soliman, 1989

Christensen, G.S. and S.A.Soliman, "A new technique for linear static state estimation based on weighted least absolute value approximations", *J. Optimization Theory and Apps.*, vol.61, no.1, pp.123-136, April 1989

Christensen & Soliman, 1990

Christensen, G.S. and S.A.Soliman, "Optimal filtering of linear discrete dynamic systems based on least absolute values approximations", *Automatica*, vol.26, no.2, pp.389-395, 1990

Churchland, 1986

Churchland, P., *Neurophilosophy: Toward a unified science of mind-brain*, The MIT Press, 1986

Cichocki & Bargiela, 1997

Cichocki, A. and A.Bargiela, "Neural networks for solving linear inequality systems" *Parallel Computing*, vol.22, no.11, pp.1455-1475, Jan. 1997

Cichocki & Unbehauen, 1992a

Cichocki A. and R.Unbehauen, "Neural networks for solving systems of linear equations and related problems", *IEEE Trans. Circuits Syst.,* vol.39, pp.124-138, Feb. 1992

Cichocki & Unbehauen, 1992b

Cichocki A. and R.Unbehauen, "Neural networks for solving systems of linear equations - Part II: Minimax and absolute value problem", *IEEE Trans. Circuits Syst.*, vol. 39, pp.619-633, September 1992

Cope & Rust, 1979

Cope, J.E. and B.W.Rust, "Bounds on solutions of linear systems with inaccurate data", *SIAM J. Numer. Anal*, vol.16, no.6, pp.950-963, December 1979

Coulbeck, 1984

Coulbeck, B., "A computer program for calibration of water distribution systems", *Journal IWES*, vol.38, no 1, 1984

Coulbeck et al., 1988a

Coulbeck, B., M.A.Brdys, C.H.Orr, J.P.Rance, "A hierarchical approach to optimized control of water distribution systems: Part I. Decomposition", *Optimal Control Applications & Methods*, vol.9, pp.51-61, 1988

Coulbeck et al., 1988b

Coulbeck, B., M.A.Brdys, C.H.Orr, J.P.Rance, "A hierarchical approach to optimized control of water distribution systems: Part II. Lower-level algorithm", *Optimal Control Applications & Methods*, vol.9, pp.109-126, 1988

Cubero, 1991

Cubero R.G., "Neural Networks for Water Demand Time Series Forecasting", *ANN INT WORKSHOP IWANN 91 PROCS*, pp.453-60, 1991

Dasarathy, 1990

Dasarathy, B.V., *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, 1990

Devijver & Kittler, 1982

Devijver, P. and J.Kittler, *Pattern Recognition: a Statistical Approach*, London, Prentice-Hall International, 1982

Dopazo et al., 1970

Dopazo, J.F., O.A.Klitin, G.W.Stagg, L.S.Van Slick, "State calculation of power systems from line-flow measurements", *IEEE Trans. PAS*, vol.89, no.7, pp.1698-1708, Sept-Oct 1970

Dopazo et al., 1975

Dopazo, J.F., O.A.Klitin, A.M.Sasson, "Stochastic load flows", *IEEE Trans. PAS*, vol.PAS-94, no.2, March-April 1975

Dubes & Jain, 1976

Dubes, R. and A.Jain, "Clustering techniques: The users dilemma", *Pattern Recognition,* vol.8, pp. 247-60, 1976

Duda & Hart, 1973

Duda, R.O. and P.E.Hart, *Pattern Classification and Scene Analysis.* New York: John Wiley, 1973

Dunn, 1974

Dunn, J., "A fuzzy relative of ISODATA process and its use in detecting compact well-separated clusters", *J.Cybern.,* vol. 3, pp. 32-57, 1974

Eggener & Polkowski, 1976

Eggener, C.L. and L.B.Polkowski, "Network models and the impact of modelling assumptions", *J. of AWWA*, vol.68, no 4, April 1976

El-Keib et al., 1992

El-Keib, A.A., J.Nieplocha, H.Singh, and D.J.Maratukulam, "A decomposed state estimation technique suitable for parallel processor implementation", *Transactions on Power Systems*, vol.7, no.3, pp.1088-97, August 1992

Everitt, 1974

Everitt, B., *Cluster Analysis.* New York: John Wiley, 1974

Falcao et al., 1981

Falcao, D.M., S.H.Karaki, A.Brameller, "Non-quadratic state estimation: a comparison of methods", *7th PSC Conf.*,Lausanne, pp.1002-1006, 1981

Fallside & Perry, 1975a

Fallside, F. and P.F.Perry, "Hierarchical optimisation of a water-supply network", *Proc. IEE*, vol.122, no.2, pp.202-208, Feb. 1975

Fallside & Perry, 1975b

Fallside, F. and P.F.Perry, "Hierarchical model for water-supply-system control", *Proc. IEE*, vol.122, no.4, pp.441-443, April 1975

Fallside & Perry, 1975c

Fallside, F. and P.F.Perry, "On-line prediction of consumption for water supply network control", *Proc. of 6th IFAC World Congress*, August 1975

Fausett, 1994

Fausett, L., *Fundamentals of neural networks: architectures, algorithms, and applications*, Prentice Hall International, 1994

Fels & Hinton, 1993

Fels, S.S. and G.E.Hinton, "Glove-Talk: A neural network interface between a data-glove and a speech synthesizer", *IEEE Transactions on Neural Networks*, vol.4, pp.2-8, 1993

Gabrys & Bargiela, 1995

Gabrys, B. and A.Bargiela, "Neural simulation of water systems for efficient state estimation", *Proceedings of Modelling and Simulation Conference ESM'95*, Prague, ISBN1-56555-080-3, pp.775-779, 1995

Gabrys & Bargiela, 1996

Gabrys, B. and A.Bargiela, "Integrated neural based system for state estimation and confidence limit analysis in water networks", *Proceedings of European Simulation Symposium*, Genoa, ISBN1-565555-099-4 (Vol.2), pp.398-402, October 1996

Gabrys & Bargiela, 1997a

Gabrys, B. and A.Bargiela, "General Fuzzy Min-Max Neural Network for Clustering and Classification", *submitted to IEEE Transactions on Neural Networks*, 1997

Gabrys & Bargiela, 1997b

Gabrys, B. and A.Bargiela, "Neural Net Aproach to Fault Diagnosis in Water Distribution Networks", *to be submitted to ASCE J. of Hydraulic Engineering*, 1997

Gill et al., 1981

Gill P.E., W.Murray and M.H.Wright, *Practical Optimization.* New York: Academic, 1981

Golub & Van Loan, 1986

Golub G.H. and C.F.Van Loan, *Matrix Computations*, Oxford: North Oxford Academic, 1986

Graf & LaLonde, 1988

Graf, D.H., and W.R.LaLonde, "A neural controller for collision-free movement of general robot manipulators", *Proc. IEEE Int. Conf. on Neural Networks, ICNN-88*, San Diego, pp.77-84, 1988

Grossberg, 1972

Grossberg, S., "Neural expectation: Cerebellar and retinal analogues of cells fired by unlearnable and learnable pattern classes", *Kybernetik,* vol. 10, pp. 49-57, 1972

Grossberg, 1976a

Grossberg, S., "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural detectors" *Biol. Cybern.*, vol. 23, pp. 121-34, 1976

Grossberg, 1976b

Grossberg, S., "Adaptive pattern classification and universal recoding: II. Feedback, oscillation, olfaction, and illusions." *Biol. Cybern.*, vol. 23, pp. 187-207, 1976

Hainsworth, 1988

Hainsworth G.D., *Measurement uncertainty in water distribution telemetry systems*, PhD Thesis, Trent Polytechnic, 1988

Hamberg & Shamir, 1988

Hamberg, D. and U.Shamir, "Schematic models for distribution systems design. I: Combination concept", *J. of Water Resources Planning and Management*, ASCE, vol.114, no 2, March 1988

Hampel et al., 1987

Hampel F.R., E.M.Roncheti, P.Roussew and W.A.Stahel, *Robust Statistics - The Approach Based on Influence Functions*, New York: Wiley, 1987

Handschin et al., 1974

Handschin, E., F.C.Schweppe, J.Kohloas, A.Fletcher, "Bad data analysis for power system state estimation", *IEEE, PES, Summer Meeting & Energy Resources Conf.*, Anaheim, Cal., July 14-19, 1974

Hartigan, 1975

Hartigan, J., *Clustering Algorithms.* New York: John Wiley, 1975

Hartley, 1996

Hartley, J.K., *Parallel algorithms for fuzzy data processing*, PhD Thesis, Nottingham Trent University, 1996

Hartley & Bargiela, 1995

Hartley, J.K. and A.Bargiela, "Parallel simulation of large-scale water distribution systems", *Proc. of ESM'95*, Prague, pp.723-727, June 1995

Hassoun, 1995

Hassoun, M.H., *Fundamentals of artificial neural networks*, The MIT Press, 1995

Hecht-Nielsen, 1988

Hecht-Nielsen, R., "Applications of counterpropagation networks", *Neural Networks*, 1, pp.131-140, 1988

Heemskerk, 1995

Heemskerk, J.N.H., "Overview of neural hardware", accessible via Internet at *ftp://ftp.mrc-apu.cam.ac.uk/pub/nn/murre/neurhard.ps*

Hopfield, 1982

Hopfield, J., "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences*, 79, pp.2554-2558, 1982

Hopfield, 1984

Hopfield, J., "Neurons with graded response have collective computational properties like those of two-state neurons", *Proceedings of the National Academy of Sciences*, 81, pp.3088-3092, 1984

Hopfield & Tank, 1985

Hopfield, J.J, and D.W.Tank, "Neural computation of decisions in optimization problems", *Biological Cybernetics*, vol.52, pp.141-152, 1985

Hopfield et al., 1983

Hopfield, J., D.Feinstein and R.Palmer, " "Unlearning" has a stabilizing effect in collective memories", *Nature*, 304, pp.158-159

Hosseinzaman & Bargiela, 1992

Hosseinzaman, A. and A.Bargiela, "Parallel simulation of nonlinear networks using diakoptics", *Proc. PACTA'92*, Sept. 1992

Huber, 1981

Huber, P.J., *Robust Statistic*, New York: Wiley, 1981

Ienne & Kuhn, 1995

Ienne, P. and G.Kuhn, "Digital systems for neural networks", in *Digital Signal Processing Technology* edited by P.Papamichalis and R.Kerwin, Critical Reviews Series CR57 Orlando, FL: SPIE Optical Engineering, pp.314-345, 1995

Jain, 1986

Jain, A. "Cluster analysis", in *Handbook of Pattern Recognition and Image Processing,* T.Young and K.S.Fu Eds., pp. 33-57, 1986

Kandel & Schwartz, 1985

Kandel, E. and J.Schwartz, *Principles of neuroscience*, Elsevier Publishing, 1985

Keller & Hunt, 1985

Keller, J.M. and D.J.Hunt, "Incorporating fuzzy membership functions into the perceptron algorithms", *IEEE Trans. PAMI,* vol. 1, pp. 693-699, 1985

Kim & Mitra, 1993

Kim Y.S. and S.Mitra, "Integrated Adaptive Fuzzy Clustering (IAFC) Algorithm", *IEEE/IEE Electronic Library (IEL),* 1993

Kohonen, 1984

Kohonen, T., *Self-Organization and Associative Memory.* Berlin: Springer-Verlag, 1984

Kohonen, 1988

Kohonen, T., "The 'neural' phonetic typewriter", *Computer*, vol.21, pp.11-22, 1988

Kotiuga & Vidyasagar, 1982

Kotiuga, W.W and M.Vidyasagar, "Bad data rejection properties of weighted least absolute value techniques applied to static state estimation", *IEEE Trans. PAS*, vol.PAS-101, no.4, pp.844-853, April 1982

Lansey, 1988

Lansey, K.E., "A procedure for water distribution network calibration considering multiple loading conditions", *Proc. Int. Symp. on Computer Modelling of Water Distribution Systems*, Kentucky, May 1988

Le Cun et al., 1989

Le Cun, Y., B.Boser, J.S.Denker, D.Henderson, R.E.Howard, and L.D.Jackel, "Backpropagation applied to handwritten ZIP code recognition", *Neural Computation*, vol.1, pp.541-551, 1989

Lin et al., 1989

Lin, J., L.Z.Li, B.W.Wan and P.D.Roberts, "Triple iterative loop technique for optimizing control of large-scale steady-state systems", *Int. J. Systems Sci.*, vol.20, no.11, pp.2107-2124, 1989

Lindsay and Lindblad, 1994

Lindsay, C.S. and T.Lindblad, "Review of hardware neural networks: a user's perspective", accessible via Internet at *http://msia02.msi.se/~lindsay/elba2html/ elba2html.html,* plenary talk given at the *Third Workshop on Neural Networks: From Biology to High Energy Physics*, Marciana Marina, Isola d'Elba, Italy, Sept. 26-30, 1994

von der Malsburg, 1973

von der Malsburg, C., "Self-organization of orientation sensitive cells of the striate cortex.", *Kybernetik*, vol. 14, pp. 85-100, 1973

Marks & Goser, 1988

Marks, K.M., and K.F.Goser, "Analysis of VLSI process data based on self-organizing feature maps", *Proc. Neuro Nimes'88*, Nimes, France, pp.337-347, 1988

McCulloch & Pitts, 1943

McCulloch, W.S. and W.H.Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, 5, pp.115-133, 1943

Merritt, 1967

Merritt, H.E., *Hydraulic control systems*, New York: John Wiley & Sons, 1967

Merill & Schweppe, 1971

Merill, H.M. and F.C.Schweppe, "Bad data suppression in power system static state estimation", *IEEE Trans. PAS*, vol.90, no.6, pp.2718-2725, 1971

Mertz et al., 1992

Mertz C.J., D.C.St.Clair, W.E.Bond, "SeMi-supervised Adaptive Resonance Theory (SMART2)", *IEEE*, 1992

Mitra & Pal, 1994

Mitra S. and K.Pal, "Self-Organizing Neural Network As a Fuzzy Classifier", *IEEE Trans. on Systems, Man and Cybernetics,* vol. 24, no. 3, March 1994

neural-net-faq.html

Neural Network FAQ (Frequently Asked Questions), maintained by Warren S. Sarle (saswss@unx.sas.com), URL: ftp://ftp.sas.com/pub/neural/FAQ.html

Newton & Mitra, 1992

Newton S.C. and S.Mitra, "An Adaptive Fuzzy System for Control and Clustering of Arbitrary Data Patterns", *IEEE/IEE Electronic Library (IEL),* 1992

Newton et al., 1992

Newton S.C., S.Pemmaraju, S.Mitra, "Adaptive Fuzzy Leader Clustering of Complex Data Sets in Pattern Recognition", *IEEE Trans. on Neural Networks,* vol.3, no.5, September 1992

Ormsbee & Chase, 1988

Ormsbee, L.E. and D.V.Chase, "Hydraulic network calibration using non-linear programming", *Proc. Int. Symp. on Computer Modelling of Water Distribution Systems*, Kentucky, May 1988

Ormsbee & Wood, 1986

Ormsbee, L.E. and D.J.Wood, "Explicit pipe network calibration", *Journal of Water Resources Planning and Management*, ASCE, vol.112, no 2, April 1986

Osiadacz, 1988

Osiadacz, A.J., "Comparison of numerical methods for steady-state simulation of gas networks", *Civ. Eng. Syst.*, vol.5, pp.25-30, March 1988

Osiadacz & Salimi, 1988a

Osiadacz, A.J. and M.A.Salimi, "Comparison between sequential and hierarchical simulation of gas networks. Part 2: Dynamic simulation of flow of gas in networks", *Info. and Decision Technologies*, vol.14, pp.99-123, 1988

Osiadacz & Salimi, 1988b

Osiadacz, A.J. and M.A.Salimi, "Hierarchical dynamical simulation of gas flow in networks", *Civ. Engng Syst.*, vol.5, pp.199-205, Dec. 1988

Pao, 1989

Pao, Yoh-Han, *Adaptive pattern recognition and neural networks*, Addison-Wesley, 1989

Pedrycz, 1990

Pedrycz, W., "Fuzzy sets in pattern recognition: methodology and methods", *Pattern Recognition,* vol. 23, no. 1/2, pp121-146, 1990

Pedrycz, 1992

Pedrycz W., "Fuzzy Neural Networks with Reference Neurons as Pattern Classifiers", *IEEE Trans. on Neural Networks,* vol.3, no.5, September 1992

Pemmaraju & Mitra, 1993

Pemmaraju S. and S.Mitra, "Identification of Noise in Clustering by Fuzzy Neural Networks", *IEEE/IEE Electronic Library (IEL)*, pp.1264-68, 1993

Pomerleau, 1991

Pomerleau, D.A., "Efficient training of artificial neural networks for autonomous navigation", *Neural Computation*, vol.3, pp.88-97, 1991

Powell et al., 1988

Powell, R.S., M.R.Irving, M.J.H.Sterling and A.Usman, "A comparison of three real-time state estimation methods for on-line monitoring of water distribution systems", in *Computer Applications in Water Supply. Volume 1: Systems Analysis and Simulation*, Ed. B.Coulbeck and C.H.Orr, pp.333-348, Research Studies Press Ltd., 1988

Powell, 1992

Powell, R.S., *On-line monitoring for operational control of water distribution networks*, PhD Thesis, University of Durham, May 1992

Pudar & Liggett, 1992

Pudar, R.S. and J.A.Liggett, "Leaks in pipe networks", *J. of Hydraulic Engineering*, vol.118, no.7, July, 1992

Quevedo et al., 1988

Quevedo, J., G.Cembrano, A.Valls, and J.Serra, "Time series modelling of water demand - a study on short-term and long-term predictions", in *Computer Applications in Water Supply. Volume 1: Systems Analysis and Simulation*, Ed. B.Coulbeck and C.H.Orr, pp.268-288, Research Studies Press Ltd., 1988

Ruspini, 1969

Ruspini, E., "A new approach to clustering", *Inform. Contr.*, vol. 15, pp. 22-32, 1969

Schweppe et al., 1970

Schweppe, F.C., J.Wildes, D.B.Rom, "Power system static state estimation. Parts I, II, III", *IEEE Trans. PAS*, vol.89, no.1, pp.120-135, Jan. 1970

Sejnowski & Rosenberg, 1987

Sejnowski, T.J., and C.R.Rosenberg, "Parallel networks that learn to pronounce English text", *Complex Systems*, vol.1, pp.145-168, 1987

Shamir & Howard, 1977

Shamir, U. and D.D.Howard, "Engineering analysis of water distribution systems", *Journal AWWA*, vol.69, no 9, Sept 1977

Simpson, 1990

Simpson, P.K., *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations*, Pergamon Press, 1990

Simpson, 1992

Simpson, P.K., "Fuzzy Min-Max Neural Networks - Part 1: Classification", *IEEE Trans on Neural Networks*, vol. 3, no. 5, pp.776-86, September 1992

Simpson, 1993

Simpson, P.K., "Fuzzy Min-Max Neural Networks - Part 2: Clustering", *IEEE Trans on Fuzzy Systems*, vol. 1, no. 1, pp.32-45, February 1993

Sobierajski, 1979

Sobierajski, M., "Optimal stochastic load flows", *Electrical Power Systems Research*, vol.2, 1979

Specht, 1990

Specht, D., "Probabilistic neural networks", *Neural Networks*, vol. 3, no.1, pp. 109-118, 1990

Sterling & Bargiela, 1984

Sterling M.J.H. and A.Bargiela, "Minimum Norm State Estimation for Computer Control of Water Distribution Systems', *IEE Proceedings,* vol.131, no.2, March 1984

Sterling & Bargiela, 1985

Sterling, M.J.H. and A.Bargiela, "Adaptive forecasting of daily water demand", in *Comparative models for electrical load forecasting*, Ed. D.W.Bunn and E.D.Farmer, John Wiley, 1985

Stuart & Herget, 1973

Stuart, T.A. and C.J.Herget, "A sensitivity analysis of weighted least squares state estimation for power systems", *IEEE PES Winter meeting*, New York, Jan-Feb 1973

Sundareshan & Elbanna, 1990a

Sundareshan, M.K. and R.M.Elbanna, "Design of decentralized observation schemes for large-scale interconnected systems: some new results", *Automatica*, vol.26, no.4, pp.789-796, 1990

Sundareshan & Elbanna, 1990b

Sundareshan, M.K. and R.M.Elbanna, "Large-scale systems with symmetrically interconnected subsystems: analysis and synthesis of decentralised controllers", *Proc. 29th CDC*, Dec. 1990

Suter & Newsome, 1988

Suter, J. and C.D.Newsome, "Network analysis: a user's viewpoint", in *Computer Applications in Water Supply. Volume 1: Systems Analysis and Simulation*, Ed. B.Coulbeck and C.Orr, pp.104-126, Research Studies Press Ltd., 1988

Tank & Hopfield, 1986

Tank D.W. and J.Hopfield, "Simple 'neural' optimization networks: an A/D converter, signal decision circuits, and a linear programming circuit", *IEEE Trans. Circuits Syst.,* vol. CAS-33, pp.533-541, May 1986

Ulanicki, 1991

Ulanicki, B., "Towards the implementation of a general purpose package for the optimization of water supply and distribution systems", *Civil Engineering Systems,* vol.8, no.4, pp.213-218, Dec. 1991

Ulanicki & Orr, 1991

Ulanicki, B. and C.H.Orr, "Unified Approach for the Optimization of Nonlinear Hydraulic Systems", *J. of Optimization Theory and Applications,* vol.68, no.1, pp.160-179, Jan. 1991

Van Cutsem & Ribbens-Pavella, 1983

Van Cutsem, Th. and M.Ribbens-Pavella, "Critical survey of hierarchical methods for state estimation of electric power systems", *IEEE Trans. on Power Apparatus and Systems*, vol.PAS-102, no.10, pp.3415-23, October 1983

Veelenturf, 1995

Veelenturf, L.P.J., *Analysis and applications of artificial neural networks*, Prentice Hall International, 1995

Walski, 1984

Walski, T.M., *Analysis of water distribution systems*, VanNostrand Reinhold Company, 1984

Wang, 1991

Wang, C., "A robust system for automated decomposition of the electromyogram utilizing a neural network architecture", Ph.D. dissertation, Department of Electrical and Computer Engineering, Wayne State University, Detroit, Mich.

Wright & Cleverly, 1988

Wright, W.G. and G.J.Cleverly, "Operational experience of GINAS and WATNET", in *Computer Applications in Water Supply. Volume 1: Systems Analysis and Simulation*, Ed. B.Coulbeck and C.Orr, pp.127-154, Research Studies Press Ltd., 1988

Yager & Zadeh, 1994

Yager, R.R. and L.A.Zadeh (editors), *Fuzzy sets, neural networks, and soft computing*, Van Nostrand Reinhold, 1994

Yoon et al., 1989

Yoon, Y.O., R.W.Brobst, P.R.Bergstresser, and L.L.Peterson, "A desktop neural network for dermatology diagnosis", *Journal of Neural Network Computing*, vol.6, pp.167-176, 1989

Zhuang & Balasubramanian, 1985

Zhuang, F. and R.Balasubramanian, "Bad data suppression in power state estimation with a variable quadratic-constant criterion", *IEEE Trans. PAS*, vol.104, no.4, pp.857-863, April 1985

Zhuang & Balasubramanian, 1987

Zhuang, F. and R.Balasubramanian, "Bad data processing in power system state estimation by direct data deletion and hypothesis tests", *IEEE Trans. on Power Systems*, vol.PWRS-2, no.2, pp.321-330, May 1987

Appendix A

Evaluation of neural architectures - simple examples

A.1. Introduction

Before the neural network algorithms from Chapter 4 were applied to the water system state estimation they were tested on a set of simple examples. Using simple examples allowed us to identify some problems and confirm some properties of the tested algorithms. From the simulation point of view it has been found out that integration methods play a very important role in the behaviour of the simulated neural circuits. The multi-step integration algorithms had to be used in order to ensure the stable operation of the NNs for wide range of parameters. Some of the examples and results for LS and LAV problems are presented below.

A.2. Least mean squares problem

Example 1: Let us consider the following simple, square set of equations

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \text{(EQ A-1)}$$

The theoretical solution (stationary point):

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \begin{bmatrix} 1.7777 & 1.5555 & -0.1111 \end{bmatrix}^T$$

On the basis of the systems of the differential and difference equations presented in Chapter 4 appropriate circuits have been designed and simulated on the computer.

All circuits (algorithms) converge to the correct solution independently of the initial starting point $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \begin{bmatrix} 1.7777 & 1.5555 & -0.1111 \end{bmatrix}^T$ with residuals $\begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$.

MATLAB and SIMULINK pseudo continuous time implementations.

The parameters of the simulation:

$\rho = \mu = 1e9$, integration time constant = 1e-8, k=0.9, $\beta = 5$, $\alpha = 0.2$

ANN architecture	On the basis of the system of equations	Activation function	Time of simulation [s]	Time of convergence [μ s]
Figure 4-2	(4-14)	Linear	1.0440	0.25
Figure 4-2	(4-14),(4-25)	Sigmoid	1.3991	0.25
Figure 4-2	(4-14),(4-26)	Talvar	1.6787	0.25
Figure 4-2	(4-14),(4-27)	Huber	1.8517	0.25
Figure 4-4	(4-34),(4-35)	-	0.7500	0.075
Figure 4-5	(4-36),(4-37)	-	0.4041	0.01

Table A-1: Time of the simulation and approximate time of convergence for different MATLAB applications

For $\rho = \mu = 1e10$ the time of simulation for all algorithms was under 0.5 [s].

C discrete implementations (parameter $\mu^{(k)} \Delta t = 2e-3$)

ANN architecture	On the basis of the system of equations	Activation function	Time of simulation [s] (user time)
Figure 4-2	(4-21)	Linear	0.0
Figure 4-2	(4-21),(4-25)	Sigmoid	0.0
Figure 4-4	(4-32),(4-33)	-	0.0

Table A-2: Time of the simulation for different C applications

From the results shown in the Table A-1 it can be seen that the neural network of Figure 4-4 provides better convergence properties in comparison to the structure of Figure 4-2 with any of the activation functions. It was also confirmed that for the neural structure of Figure 4-5 the solution can be found in the predetermined time when the parameter μ is adapted during the simulation in the way described in Chapter 4, Section 4.3.4. The stability problem has been observed in the discrete (C) implementations of the neural networks. If the integration time constant of the integrators is too large with large weights $\mu^{(k)}$ and/or $\rho^{(k)}$ the algorithm can be unstable. However, for continuous-time algorithms the weights can be arbitrary large without causing stability problems.

The examples of the convergence trajectories for the neural structures of Figure 4-4 and Figure 4-5 are shown in Figure A-1 and Figure A-2.

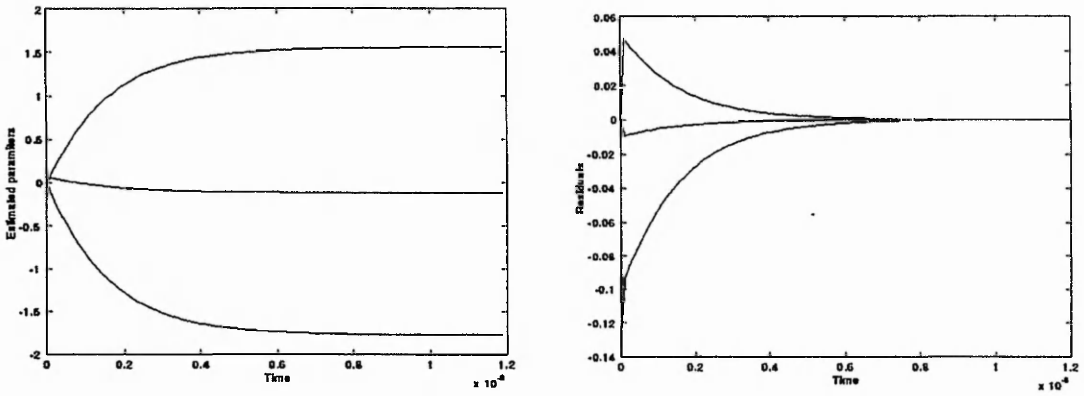


Figure A-1: Computer simulated trajectories of the estimated parameters and the residuals for Example 1 (for the circuit structure of Figure 4-4)

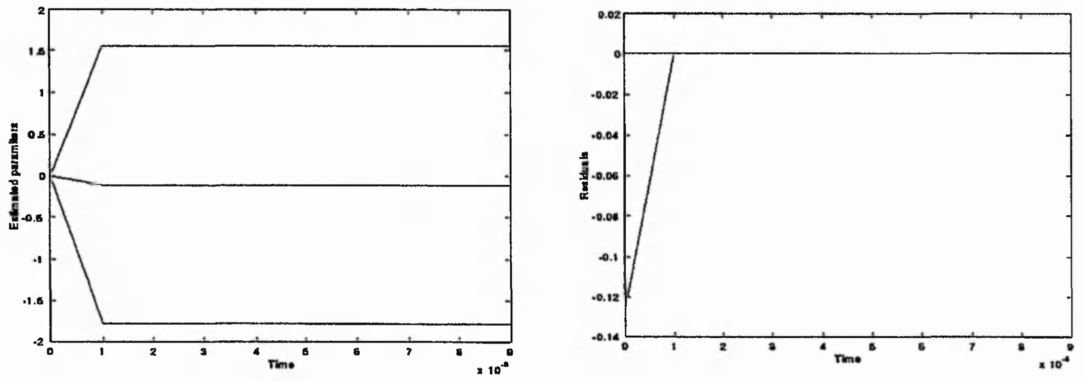


Figure A-2: Computer simulated trajectories of the estimated parameters and the residuals for Example 1 (for the circuit structure of Figure 4-5)

Example 2: Let us consider the following least squares value problem

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 3 \end{bmatrix} \qquad \text{(EQ A-2)}$$

The theoretical solution (stationary point) $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{bmatrix} 0.2 & 0.6 \end{bmatrix}^T$

with residuals $\begin{bmatrix} r_1 & r_2 & r_3 & r_4 & r_5 \end{bmatrix}^T = \begin{bmatrix} -0.2 & 0.4 & 0 & -0.4 & 0.2 \end{bmatrix}^T$

All circuits (algorithms) converge to the correct solution independently of the initial starting point.

MATLAB and SIMULINK pseudo continuous time implementations.

The parameters of the simulation:

$\rho = \mu = 1e9$, integration time constant = 1e-8, k=0.9, $\beta = 5$, $\alpha = 0.2$

ANN architecture	On the basis of the system of equations	Activation function	Time of simulation [s]	Time of convergence [μ s]
Figure 4-2	(4-14)	Linear	0.2753	~0.08
Figure 4-2	(4-14),(4-25)	Sigmoid	0.3509	~0.08
Figure 4-2	(4-14),(4-26)	Talvar	0.3771	~0.08
Figure 4-2	(4-14),(4-27)	Huber	0.4255	~0.08
Figure 4-4	(4-34),(4-35)	-	0.4915	~0.03
Figure 4-5 (for $\mu = 5e5$)	(4-36),(4-37)	-	0.4041	~0.95

Table A-3: Time of the simulation and approximate time of convergence for different MATLAB applications

C discrete implementations (parameter $\mu^{(k)}\Delta t = 2e-3$)

ANN architecture	On the basis of the system of equations	Activation function	Time of simulation [s] (user time)
Figure 4-2	(4-21)	Linear	0.0
Figure 4-2	(4-21),(4-25)	Sigmoid	0.0
Figure 4-4	(4-32),(4-33)	-	0.0

Table A-4: Time of the simulation for different C applications

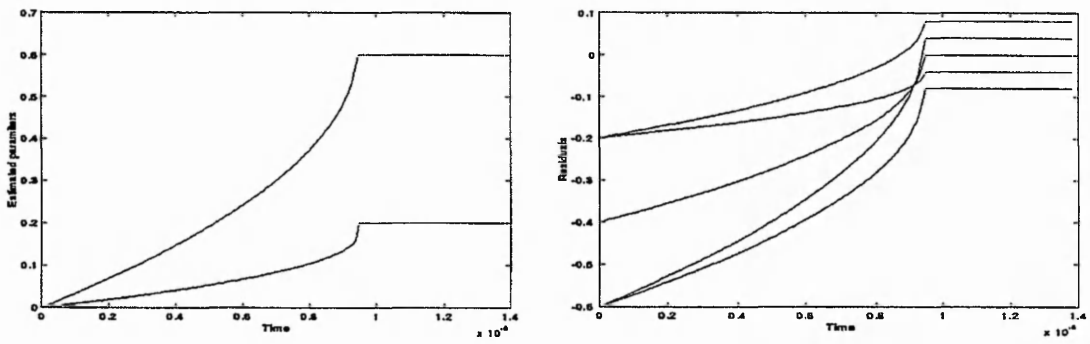


Figure A-3: Computer simulated trajectories of the estimated parameters and the residuals for Example 2 (for the circuit structure of Figure 4-5)

Special care has to be taken when using the neural structure with the Talvar activation function. Since at the first few iterations the residuals, depending on the initial starting point, can be artificially large it is possible that when using too small cut-off parameter β the correct solution will not be found. In case of the sigmoid activation function when the parameters α and β are given such values that the residuals are processed on the linear part of the sigmoid the solution is very close to the standard least squares estimation. The example when the sigmoid activation function is used to obtain the solution close to the least absolute value estimation is shown in the next section.

Example 3: Let us consider the following least squares value problem

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ -1 \\ -10 \end{bmatrix} \tag{EQ A-3}$$

The theoretical solution (stationary point): $x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \begin{bmatrix} 0.6 & 3.5 & -1.5 \end{bmatrix}^T$

All circuits (algorithms) converge to the correct solution independently of the initial starting point.

MATLAB and SIMULINK pseudo continuous time implementations.

The parameters of the simulation:

$\rho = \mu = 1e9$, integration time constant = 1e-8, k=0.9, $\beta = 5$, $\alpha = 0.2$

ANN architecture	On the basis of the system of equations	Activation function	Time of simulation [s]	Time of convergence [μ s]
Figure 4-2	(4-14)	Linear	1.4164	~3
Figure 4-2	(4-14),(4-25)	Sigmoid	1.8923	~3
Figure 4-2	(4-14),(4-26)	Talvar	1.7670	~3
Figure 4-2	(4-14),(4-27)	Huber	1.7346	~3
Figure 4-4	(4-34),(4-35)	-	0.9911	~1
Figure 4-5 (for $\mu = 1e5$)	(4-36),(4-37)	-	2.4789	~2

Table A-5: Time of the simulation and approximate time of convergence for different MATLAB applications

C discrete implementations (parameter $\mu^{(k)}\Delta t = 2e-3$)

ANN architecture	On the basis of the system of equations	Activation function	Time of simulation [s] (user time)
Figure 4-2	(4-21)	Linear	1.0
Figure 4-2	(4-21),(4-25)	Sigmoid	1.0
Figure 4-4	(4-32),(4-33)	-	0.0

Table A-6: Time of the simulation for different C applications

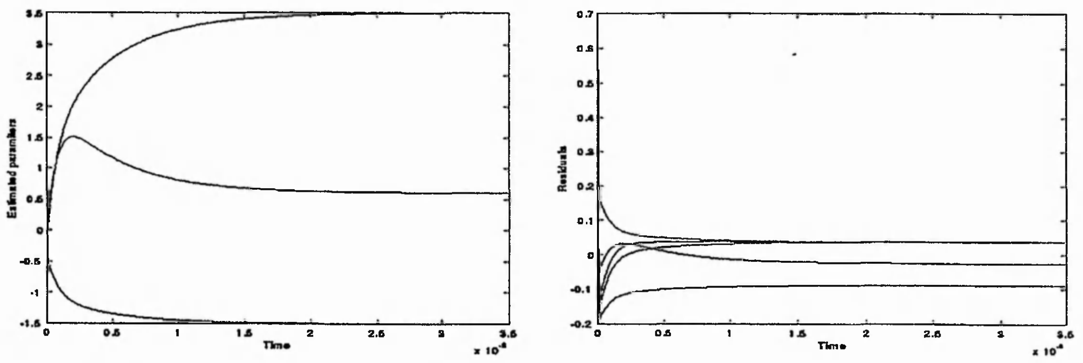


Figure A-4: Computer simulated trajectories of the estimated parameters and the residuals for Example 3 (for the circuit structure of Figure 4-2 with a linear activation function)

Example 4: Let us consider the ill-conditioned least squares value problem.

$$\begin{bmatrix} 1.1 & 2 & 3.1 \\ 4 & 5.1 & 6 \\ 7 & 8 & 8.9 \\ -3.1 & -4 & -5 \\ 5.5 & 6.9 & 8.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2 \\ 1.1 \\ 4.2 \\ -0.5 \\ 2.1 \end{bmatrix} \qquad \text{(EQ A-4)}$$

The condition number of the matrix A in this case is approximately equal to 200.

The theoretical solution (stationary point):

$$x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \begin{bmatrix} 1.6673 & 1.437 & -2.1198 \end{bmatrix}^T$$

All circuits (algorithms) converge to the correct solution independently of the initial starting point.

MATLAB and SIMULINK pseudo continuous time implementations.

The parameters of the simulation:

$\rho = \mu = 1e9$, integration time constant = 1e-8, k=0.9, $\beta = 5$, $\alpha = 0.2$

ANN architecture	On the basis of the system of equations	Activation function	Time of simulation [s]	Time of convergence [μ s]
Figure 4-2	(4-14)	Linear	0.5615	~0.7
Figure 4-2	(4-14),(4-25)	Sigmoid	0.6261	~0.7
Figure 4-2	(4-14),(4-26)	Talvar	0.6763	~0.7
Figure 4-2	(4-14),(4-27)	Huber	0.6020	~0.7
Figure 4-4	(4-34),(4-35)	-	0.6264	~0.3
Figure 4-5 (for $\mu = 1e7$)	(4-36),(4-37)	-	-	~1.3

Table A-7: Time of the simulation and approximate time of convergence for different MATLAB applications

C discrete implementations (parameter $\mu^{(k)}\Delta t = 2e-3$)

ANN architecture	On the basis of the system of equations	Activation function	Time of simulation [s] (user time)
Figure 4-2	(4-21)	Linear	24.0
Figure 4-2	(4-21),(4-25)	Sigmoid	57.0
Figure 4-4	(4-32),(4-33)	-	14.0

Table A-8: Time of the simulation for different C applications

Nothing new could be said on the basis of the results from *Example 3* that has not been said in the first two examples but it serves as a very good comparison to the ill-conditioned problem at hand. The convergence rate of the continuous-time implementations (Table A-7) in this example is longer but still comparable with the results for *Example 3* (Table A-5). However, it is evident, from the results presented in Table A-8 and Table A-6, that a special care has to be taken in a case of the neural network discrete implementations when dealing with ill-conditioned problems since the convergence time can be prohibitively slow.

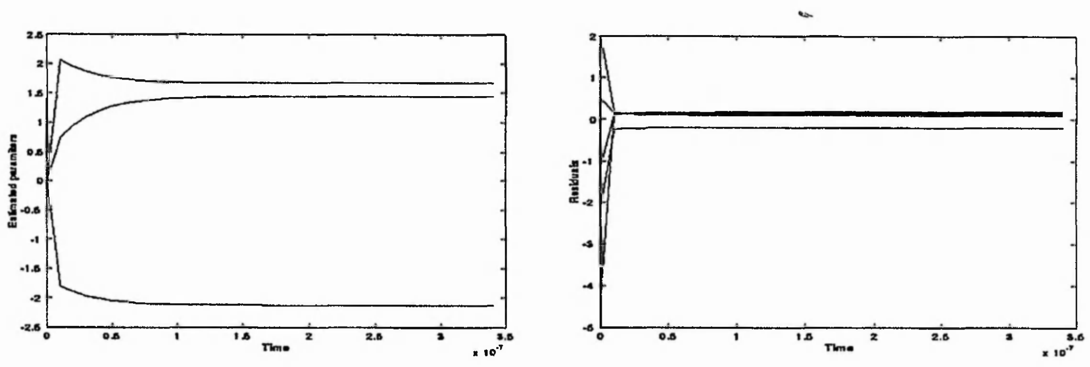


Figure A-5: Computer simulated trajectories of the estimated parameters and the residuals for Example 4 (for the circuit structure of Figure 4-4)

A.3. Least absolute value problem

Example 5: Let us consider the least absolute value problem described by the set of equations (EQ A-2).

The theoretical solution (stationary point for the least absolute value problem): $x=\begin{bmatrix}x_1 & x_2\end{bmatrix}^T = \begin{bmatrix}0.5 & 0.5\end{bmatrix}^T$

MATLAB and SIMULINK pseudo continuous time implementations.

The parameters of the simulation:

$\mu = 1e9$, integration time constant = 1e-8

ANN architecture	On the basis of the system of equations	Activation function	Time of simulation [s]	Time of convergence [μs]
Figure 4-2 ($\mu = 1e9$)	(4-14)	Signum	11.9652	~0.01
Figure 4-2 ($\beta = 1$, $\alpha = 100$)	(4-14),(4-25)	Sigmoid	0.2658	~0.07
Figure 4-2 ($\beta = 1$, $\alpha = 2000$)	(4-14),(4-25)	Sigmoid	0.3195	~0.01
Figure 4-6 and Figure 4-7 ($\mu = 1e9$, $\rho = 1e5$)	system using the inhibit principle	-	1.734	~0.11

Table A-9: Time of the simulation and approximate time of convergence for different MATLAB applications

ANN architecture	Found solution
Figure 4-2 ($\mu = 1e9$)	x=[0.4996 0.5003]
Figure 4-2 ($\beta = 1$, $\alpha = 100$)	x=[0.4917 0.5027]
Figure 4-2 ($\beta = 1$, $\alpha = 2000$)	x=[0.4979 0.5007]
Figure 4-6 and Figure 4-7 ($\mu = 1e9$, $\rho = 1e5$)	1 step: x=[0.2 0.6] 2 step: x=[0.5 0.5]

Table A-10: Solutions for different MATLAB applications

The presence of discontinuity in the neural network with signum activation function is responsible for the long simulation time. Time of convergence strongly depends on the integration time constant.

C discrete implementations (parameter $\mu^{(k)}\Delta t = 2e-3$

ANN architecture	On the basis of the system of equations	Time of simulation [s] (user time)	Found solution
Figure 4-6 and Figure 4-7 (the *m-n* largest residuals are found after completing the LS stage)	system using the inhibit principle	0.966	1 step: x=[0.2 0.6] 2 step: x=[0.5 0.5]

Table A-11: Time of the simulation for different C applications

ANN architecture	On the basis of the system of equations	Time of simulation [s] (user time)	Found solution
Figure 4-6 and Figure 4-7 (continuous tracking of the *m-n* largest residuals)	system using the inhibit principle	1.11	1 step: x=[0.2 0.6] 2 step: x=[0.5 0.5]
Figure 4-6 and Figure 4-7 (the *m-n* largest residuals are inhibited one at a time in the *m-n* steps of simulation)	system using the inhibit principle	1.22	in 4 steps x=[0.5 0.5]
Figure 4-2 ($\beta = 1$, $\alpha = 100$)	(4-21),(4-25)	0.332	x=[0.4917 0.5027]

Table A-11: Time of the simulation for different C applications

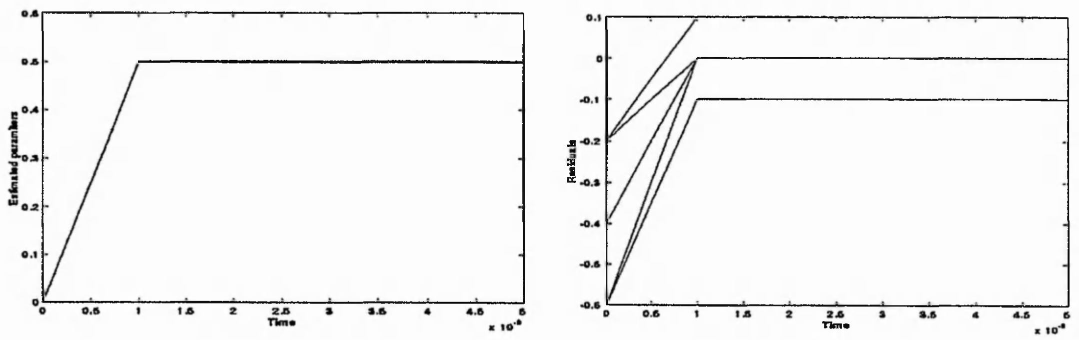


Figure A-6: Computer simulated trajectories of the estimated parameters and the residuals for Example 5 ($\beta = 1$, $\alpha = 2000$ for the circuit structure of Figure 4-2)

Example 6: Let us consider the least absolute value problem described by the set of equations (EQ A-3).

The theoretical solution (stationary point): $x = \left[x_1\ x_2\ x_3\right]^T = \left[1\ 2.75\ -1.375\right]^T$

The best results (in a sense of accuracy) have been obtained by using the slightly modified network from Figure 4-6 and Figure 4-7. In the first run the network was able to find the following standard least-squares solution $\boldsymbol{x1} = \left[0.6\ 3.5\ —1.5\right]^T$ with the residual vector $\boldsymbol{r}(\boldsymbol{x1}) = \left[-0.4\ 0.6\ 0.6\ -1.4\ 0.6\right]^T$.

In this case it is impossible to select two largest, in absolute value, residuals because $|r_1| = |r_3| = |r_5| = 0.6$. In the second run only S_4=0 (opened switch) and the network found then $\boldsymbol{x2} = \left[0.9182\ 2.6409\ -1.3409\right]^T$ with the residual vector $\boldsymbol{r}(\boldsymbol{x2}) = \left[-0.818\ 0.2182\ -0.1636\ -2.2273\ 0.0273\right]^T$.

On the basis of these results the inhibitive control network set S_2 to zero (opened switch), so in the third run only $S_1 = S_3 = S_5 = 1$ (closed switches), and the network found the stationary point $\boldsymbol{x} = \begin{bmatrix}1 & 2.75 & -1.375\end{bmatrix}^T$ with the residual vector $\boldsymbol{r}(\boldsymbol{x}) = \begin{bmatrix}0 & 0.375 & 0 & -2.125 & 0\end{bmatrix}^T$.

In the above example the inhibition control subnetwork is realized on the basis of the following principle. The circuit selects at first the largest signal (the largest residual) which is inhibited, i.e. the corresponding S_i is set to zero (opened switch). Next, the procedure is (*m*-*n*) times repeated for the rest of the signals.

By using the network with signum and sigmoid activation function we have obtained the less accurate results although very close to the correct solution.

Example *7*: Let us consider the least absolute value problem described by the set of equations (EQ A-4).

The theoretical solution (stationary point): $\boldsymbol{x} = \begin{bmatrix}x_1 & x_2 & x_3\end{bmatrix}^T = \begin{bmatrix}2 & 1 & -2\end{bmatrix}^T$

All circuits (algorithms) used for *Example 5* were able (in this case) to find solution very close to the correct one. The best results (in the sense of accuracy) have been obtained by using circuits employing the inhibit principle. These networks found the final solution $\boldsymbol{x} = \begin{bmatrix}x_1 & x_2 & x_3\end{bmatrix}^T = \begin{bmatrix}2 & 1 & -2\end{bmatrix}^T$.

Although it must be mentioned that the shortest time of simulation for C implementations, using simple Euler integration rule, was 24[s] in comparison to 0.7698 [s] in the case of the NN with sigmoid activation function approximating the signum function simulated using more sophisticated integration algorithms (MATLAB implementation).