THE SIMULATION AND OPTIMIZATION OF STEADY STATE PROCESS CIRCUITS BY MEANS OF ARTIFICIAL NEURAL NETWORKS

by

CHRISTIAAN ALDRICH

dissertation submitted in
accordance with the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ENGINEERING

at the

University of Stellenbosch

Promoter: Prof J S J van Deventer

- November 1993 -

DECLARATION

I hereby certify that this dissertation is my own original work, except where specifically acknowledged in the text. Neither the present dissertation, nor any part thereof, has previously been submitted at any other university.

16 November 1993

Aan Annemarie

haar ondersteuning was opsigself genoeg rede om hierdie werk te voltooi

SYNOPSIS

Since the advent of modern process industries engineers engaged in the modelling and simulation of chemical and metallurgical processes have had to contend with two important dilemmas. The first concerns the ill-defined nature of the processes they have to describe, while the second relates to the limitations of prevailing computational resources.

Current process simulation procedures are based on explicit process models in one form or another. Many chemical and metallurgical processes are not amenable to this kind of modelling however, and can not be incorporated effectively into current commercial process simulators. As a result many process operations do not benefit from the use of predictive models and simulation routines and plants are often poorly designed and run, ultimately leading to considerable losses in revenue.

In addition to this dilemma, process simulation is in a very real way constrained by available computing resources. The construction of adequate process models is essentially meaningless if these models can not be solved efficiently - a situation occurring all too often.

In the light of these problems, it is thus not surprizing that connectionist systems or neural network methods are singularly attractive to process engineers, since they provide a powerful means of addressing both these dilemmas. These nets can form implicit process models through learning by example, and also serve as a vehicle for parallel supercomputing devices. In this dissertation the use of artificial neural networks for the steady state modelling and optimization of chemical and metallurgical process circuits is consequently investigated.

The first chapter is devoted to a brief overview of the simulation of chemical and metallurgical plants by conventional methods, as well as the evolution and impact of computer technology and artificial intelligence on the process industries.

Knowledge of the variance-covariance matrices of process data is of paramount importance to data reconciliation and gross error detection problems, and although various methods can be employed to estimate these often unknown

variances, it is shown in the second chapter that the use of feedforward neural nets can be more efficient than conventional strategies.

In the following chapter the important problem of gross error detection in process data is addressed. Existing procedures are statistical and work well for systems subject to linear constraints. Non-linear constraints are not handled well by these methods and it is shown that back propagation neural nets can be trained to detect errors in process systems, regardless of the nature of the constraints.

In the fourth chapter the exploitation of the massively parallel information processing structures of feedback neural nets in the optimization of process data reconciliation problems is investigated. Although effective and sophisticated algorithms are available for these procedures, there is an ever present demand for computational devices or routines that can accommodate progressively larger or more complex problems. Simulations indicate that neural nets can be efficient instruments for the implementation of parallel strategies for the optimization of such problems.

In the penultimate chapter a gold reduction plant and a leach plant are modelled with neural nets and the models shown to be considerably better than the linear regression models used in practice. The same technique is also demonstrated with the modelling of an apatite flotation plant. Neural nets can also be used in conjunction with other methods and in the same chapter the steady state simulation and optimization of a gravity separation circuit with the use of two linear programming models and a neural net are described.

OORSIG

Sedert die ontstaan van prosesingenieurswese, het ingenieurs gemoeid met die modellering en simulasie van chemiese en metallurgiese prosesse met twee belangrike dilemmas te kampe gehad. Die eerste het te make met die swak-gedefinieerde aard van chemiese prosesse, wat die beskrywing en dus ook die beheer daarvan kompliseer, terwyl die tweede verband hou met die beperkinge van huidige berekeningsmiddele.

Die prosesse wat tans gebruik word om chemiese prosesse te simuleer is gebaseer op eksplisiete prosesmodelle van een of ander aard. Baie chemiese en metallurgiese prosesse kan egter nie op 'n eksplisiete wyse gemodelleer word nie, en kan gevolglik ook nie doeltreffendheid deur kommersiële prosessimulators beskryf word nie. Die bedryf van baie prosesse vind derhalwe nie baat by die gebruik van voorspellende modelle en simulasie-algoritmes nie en aanlegte word dikwels suboptimaal ontwerp en bedryf, wat uiteindelik tot aansienlike geldelike verliese kan lei.

Prosessimulasie word op die koop toe ook beperk deur die beskikbaarheid van berekeningsfasiliteite. Die konstruksie van geskikte prosesmodelle hou geen voordeel in as hierdie modelle nie doeltreffendheid opgelos kan word nie.

Teen die agtergrond van hierdie probleme is dit nie verrassend dat neurale netwerke 'n besondere bekoring vir prosesingenieurs inhou nie, aangesien hulle beide hierdie dilemmas aanspreek. Hierdie nette kan implisiete prosesmodelle konstrueer deur te leer van voorbeelde en dien ook as 'n raamwerk vir parallelle superrekenaars. In hierdie proefskrif word die gebruik van kunsmatige neurale netwerke vir gestadigde toestandsmodellering en optimering van chemiese en metallurgiese prosesse gevolglik ondersoek.

Die eerste hoofstuk word gewy aan 'n kort oorsig oor die simulasie van chemiese en metallurgiese aanlegte met konvensionele tegnieke, asook die ontwikkeling en impak van rekenaartegnologie en skynintelligensie in die prosesnywerhede.

Kennis van die variansie-kovariansie-matrikse van prosesdata is van kardinale belang vir datarekonsiliasie en die identifikasie en eliminasie van sistematiese foute en alhoewel verskeie metodes aangewend kan word om hierdie

onbekende variansies te beraam, word daar in die tweede hoofstuk getoon dat die gebruik van neurale netwerke meer doeltreffend is as konvensionele strategieë.

In die volgende hoofstuk word die belangrike probleem van sistematiese fout-opsporing in prosesdata ondersoek. Bestaande prosedures is statisties van aard en werk goed vir stelsels onderworpe aan lineêre beperkinge. Nie-lineêre beperkinge kan nie doeltreffend deur hierdie prosedures hanteer word nie en daar word gewys dat terugwaarts-propagerende nette geleer kan word om sulke foute in prosessisteme op te spoor, ongeag die aard van die beperkinge.

In die vierde hoofstuk word die rekonsiliasie van prosesdata met behulp van massiewe parallelle dataverwerkingstrukture soos verteenwoordig deur terugvoerende neurale nette, ondersoek. Alhoewel doeltreffende en gesofistikeerde algoritmes beskikbaar is vir die optimering van die tipe probleme, is daar 'n onversadigbare aanvraag na rekenaars wat groter en meer komplekse stelsels kan akkommodeer. Simulasie dui aan dat neurale nette effektief aangewend kan word vir die implementering van parallelle strategieë vir dié tipe optimeringsprobleme.

In die voorlaaste hoofstuk word die konneksionistiese modellering van 'n goudreduksie- en 'n logingsaanleg beskryf en daar word aangetoon dat die neurale netwerk-modelle aansienlik beter resultate lewer as die lineêre regressie-modelle wat in die praktyk gebruik word. Dieselfde tegnieke vir die modellering van 'n flottasie-aanleg vir apatiet word ook bespreek. Neural nette kan ook saam met ander metodes aangewend word en in dieselfde hoofstuk word die gebruik van twee lineêre programmeringsmodelle en 'n neural net om 'n gravitasieskeidingsbaan onder gestadigde toestande te simuleer en te optimeer, beskryf.

ACKNOWLEDGEMENTS

A thesis like this is never written in isolation and although the author's own original work, it is with deep gratitude that the contributions of the following persons are acknowledged:

- Prof Jannie van Deventer for his incredible patience, inspiration and *sine qua non* support;
- My wife Annemarie for her understanding of the inherent incompatibility of family life and projects such as these;
- My family and friends for their encouragement, often in subtle ways they are not even aware of, and
- Last but not least, to Him who is the first principle of all knowledge.

CONTENTS

CHAPTER 1

Introduction

Summary

In this introductory chapter an overview of the ill-defined nature and dimensional character of modelling and simulation problems in the chemical and metallurgical processing industries is presented. A cursory introduction to the field of artificial intelligence and its application in the process industries is given. Special emphasis is placed on the use of artificial neural networks as far as the description of large or complex processes is concerned, and the motivation and specific objectives of this study are subsequently highlighted.

1.1 OBJECTIVES OF CHAPTER 1

This chapter outlines the nature and background of process modelling and simulation in the chemical and metallurgical industries, with special reference to

- the nature of problems related to process modelling and simulation in the chemical and metallurgical engineering industry;
- the field of artificial intelligence with special emphasis on connectionist systems or neural networks;
- current research and applications of neural networks in the chemical and metallurgical process industries;
- the motivation and specific objectives of this study

1.2 THE NATURE OF PROCESS MODELLING AND SIMULATION PROBLEMS

Numerous demands have to be met by modern chemical and metallurgical process technologies. Equipment should for example be selected to yield the maximum return on investment, raw materials and energy

resources should be utilized in the most efficient way, and the plant has to be reliable and flexible, while also complying with growing world-wide demands that operations should have as little impact on the environment as possible (Kraslawski et al., 1992).

All these requirements provide a continuous driving force to find new or improved methods of designing and optimizing process plants (see appendix B for a brief review of process simulation methodology), in spite of the extraordinary complexity and scale of the problem. Considering that design activities consume approximately 10-15% of the funds required to move from an initial concept to the manufacturing of a product, and that the design step fixes approximately 80% of the cost involved in production (Westerberg, 1991), the ceaseless demand for improvement in simulation and design is not surprizing. A plant designed for the chemical conversion and processing of raw materials consists of a great number of different subsystems, fittings and process units. All these systems can be intricately connected, each affecting the other in different ways. Process conditions can furthermore cover ranges of many orders of magnitude and chemicals can interact with one another in unpredictable and often destructive ways.

It is therefore crucial that the approach to the design and investigation of such systems is focused on as few features of the system as possible, without seriously compromising the character of the system. It is only by separating the more important elements of the system from the less important ones, that the otherwise impenetrable confusion can be arranged in an orderly coordinated hierarchical structure amenable to investigation and understanding. In their quest for this elusive goal, engineers are faced with two basic dilemmas. The first concerns the nature of the processes they have to describe, while the second is related to the processing of information regarding the describable processes.

Despite extensive world-wide fundamental research and development, the majority of chemical and metallurgical processes are ill-defined to such an extent that they simply can not be modelled adequately from first principles alone. The main reason for this is the immense chasm in the body of knowledge concerning the behaviour of physico-chemical

resources should be utilized in the most efficient way, and the plant has to be reliable and flexible, while also complying with growing world-wide demands that operations should have as little impact on the environment as possible (Kraslawski et al., 1992).

All these requirements provide a continuous driving force to find new or improved methods of designing and optimizing process plants (see appendix B for a brief review of process simulation methodology), in spite of the extraordinary complexity and scale of the problem. Considering that design activities consume approximately 10-15% of the funds required to move from an initial concept to the manufacturing of a product, and that the design step fixes approximately 80% of the cost involved in production (Westerberg, 1991), the ceaseless demand for improvement in simulation and design is not surprizing. A plant designed for the chemical conversion and processing of raw materials consists of a great number of different subsystems, fittings and process units. All these systems can be intricately connected, each affecting the other in different ways. Process conditions can furthermore cover ranges of many orders of magnitude and chemicals can interact with one another in unpredictable and often destructive ways.

It is therefore crucial that the approach to the design and investigation of such systems is focused on as few features of the system as possible, without seriously compromising the character of the system. It is only by separating the more important elements of the system from the less important ones, that the otherwise impenetrable confusion can be arranged in an orderly coordinated hierarchical structure amenable to investigation and understanding. In their quest for this elusive goal, engineers are faced with two basic dilemmas. The first concerns the nature of the processes they have to describe, while the second is related to the processing of information regarding the describable processes.

Despite extensive world-wide fundamental research and development, the majority of chemical and metallurgical processes are ill-defined to such an extent that they simply can not be modelled adequately from first principles alone. The main reason for this is the immense chasm in the body of knowledge concerning the behaviour of physico-chemical

processes that instead of narrowing, appears to expand as the frontiers of science and technology are pushed ever further. This ill-defined nature of the processes that engineers have to harness and control in order to meet the growing demands of consumer societies requires the use of alternative modelling methodologies, which are not based on the use of knowledge in an explicit analytical form.

The second problem that the process engineer has to come to grips with is the huge information processing burdens posed by complex engineering problems. In the early 1960s for example, the workhorse of the day (an IBM mainframe) could integrate a simple differential equation over a weekend, while complex equations could take many weeks to solve. Meteoric progress has been made since, and much more powerful machines are available today, but these are still pitifully inadequate when viewed against the background of modern chemical engineering problems, which could involve complex relationships between many thousands of variables with domains spanning several orders of magnitude.

With these problems in mind it is not difficult to understand the extraordinary interest that artificial neural networks (the fundamentals of which are briefly reviewed in appendix A) have sparked in the process engineering community since the late 1980s. Not only do they serve as a vehicle for the construction of implicit models of ill-defined processes, they are also one of the pillars of a major new computational paradigm that promises to increase the power of available computing platforms by several orders of magnitude over the next few years. Despite the avalanche of research funds that has flowed into the research and development of neuralware and related techniques over the last several years, the technology is still young and much needs to be done to move it from research laboratories into the commercial sector, especially as far as the chemical and metallurgical industries are concerned.

This dissertation is consequently an investigation into the use of artificial neural networks in the modelling and optimization of steady-state process circuits. Special emphasis is placed on ill-defined processes, that is processes not readily represented by analytical or fundamental models. Since the modelling of plants of this nature is essentially data driven, a

large part of the investigation revolves around the processing of plant data. These techniques include some aspects of the estimation of statistical parameters, the screening of the data for various types of errors, as well as the reconciliation of the data prior to use in the actual modelling of plants or process units. In the final chapter the modelling of industrial plants, as well as the optimization of a gravity separation circuit with a hybrid linear programming neural net model is discussed.

The rest of this chapter is devoted to a brief introduction to the field of artificial intelligence, with special reference to artificial neural network technology.

1.3 PROGRESS IN ARTIFICIAL INTELLIGENCE

In this section the application of neural nets in process engineering is discussed after a brief look at the use of artificial intelligence techniques to solve process engineering problems. These techniques are concerned with alternative methods for the use of knowledge representing ill-defined processes and have increasingly been used in the quest for solutions to modelling problems or the enhancement of existing solution strategies in the process engineering industry.

1.3.1 Artificial intelligence

By the end of World War II several groups of scientists of the United States and England were working on what is now known as a computer. Although Alan Turing, the principal British scientist at the time, suggested the use of logical operators (such as OR, AND, NOT, etc.) as a basis for fundamental instructions to these machines, the majority of investigators favoured the use of numeric operators (+,-,<,etc.). It was only with the shifting emphasis on methods to allow computers to behave more like humans that the approach advocated by Turing had begun to attract new interest. This entire research effort and its commercial repercussions are known as *artificial intelligence* (AI), and comprize many aspirations, ranging from the design of machines to do various things considered to be intelligent, to machines which could provide insight into the mental faculties of man. Although different workers in the field have different goals, all seek to design machines that

can solve problems. In order to achieve this goal, two basic strategies can be pursued (Minsky, 1993).

The first strategy or top-down approach has been developed productively for several decades and entails the reduction of large complex systems to small manipulable units. These techniques encompass heuristic programming, goal-based reasoning, parsing and causal analysis and are efficient systematic search procedures, capable of the manipulation and rearrangement of elements of complex systems or the supervision or management of the interaction between subsystems interacting in intricate ways. The disadvantages of symbolic logic systems such as these are their inflexibility and restricted operation which limits them to very narrow domains of knowledge.

Bottom-up strategies (i.e. connectionist procedures) endeavour to build systems with as little architecture as possible. These systems start off with simple elements (such as simplified models, small computer programs, elementary principles, etc.) and move towards more complex systems by connecting these units to produce large-scale phenomena. As a consequence, these systems are very versatile and capable of the representation of uncertain approximate relations between elements or the solution of problems involving large numbers of weak interactions (such as found in pattern recognition and knowledge retrieval problems). Connectionist systems can on the other hand not reason well and are not capable of symbolic manipulation and logic analyses.

The field of artificial intelligence is diversifying continuallyand has grown to comprize the major branches concerned with knowledge-based systems, neural nets, fuzzy logic techniques, as well as genetic algorithms.

1.3.2 Fuzzy logic

Fuzzy logic or fuzzy systems use if-then rules in a similar way as rule-based expert systems to define relationships. The rules usually define a particular set of input states, and provide descriptions of the consequences if those particular states prevail. Unlike expert systems, fuzzy logic systems use membership functions to attach numerical values to the antecedents of rules to denote the extent to which these premises

are valid (Berardinis, 1992). Since these systems provide for smooth continuous valued transitions between different sets of outcomes, they are particularly attractive to process engineers. These systems are used in main-frame data base applications (Klimasauskas, 1992), and have found major commercial application in the electronic control circuitry in automobiles, vacuum cleaners, air conditioners, washing machines, chlorine controllers for water purification plants, control systems for cement kilns, etc. since 1982 when only one patent was registered, compared to 1460 in 1992 (Dambrot, 1992; Kahaner, 1991; Rosenbaum, 1992).

1.3.3. Genetic algorithms

Genetic algorithms are constituted by mathematical techniques inspired by the biological process of evolution and are mainly used for direct search and optimization. In contrast to Monte-Carlo procedures, genetic algorithms have a strongly directed component which reduces search time considerably and improves convergence. The ability of genetic algorithms to find near-optimal solutions in huge search variable spaces, is especially useful when hill-climbing or gradient search techniques fail owing to noise in the data, entrapment in local minima, or discontinuities in the objective functions (Klimasauskas, 1992; Sikora, 1992).

1.3.4 Knowledge-based systems

Until recently knowledge-based or expert systems were undoubtedly the most important branch of artificial intelligence - to such an extent that they were often confused with the entire field of artificial intelligence itself.

An expert system essentially consists of methods of maintaining a data base or a knowledge base of facts and relationships, as well as structured routines for searching the data base as efficiently as possible. These systems provide the facility to trace a search process (i.e. explain the conclusions of a decision-making process) in order to help the user evaluate the decisions. Expert systems have made important contributions to the efficient development of complex systems, the development of more logically complex systems than previously thought possible, and

have furthermore reduced the technical level of expertise required to develop these systems (Klimasauskas, 1992).

Knowledge-based systems were confined to the research laboratories of a few universities till as recently as 1980, but had established a durable niche in the process engineering industry less than a decade later. Today the United States, the United Kingdom, the European Economic Community and Japan are all involved in major research programmes concerned with the development and implementation of expert systems. Large chemical engineering concerns such as Du Pont and Exxon Chemicals are organizing AI departments and venture capital flows into a multitude of expert system companies which have mushroomed all over the world. In the process engineering sector these systems are used globally, by such companies as British Petroleum Chemicals, to cope with plant dynamics, process scale-up and the allocation of utilities, Shell Oil Company for the interpretation and filtering of alarm signals, Blue Circle cement company, etc., saving many millions of dollars for these companies in the process (Barnwell & Ertl, 1987; Allott, 1991).

In retrospect the breakthrough for knowledge-based systems in the process engineering sector in the late 1980s-early 1990s was a direct result of the maturation of technology allowing the use of real-time on-line applications.

1.3.5 Artificial neural networks

Neural nets are powerful mathematical techniques inspired by the study of the human brain. Unlike traditional expert systems, where knowledge is stored explicitly in a data base or as a set of rules or heuristics, neural nets generate their own rules by learning from examples, as is explained in more detail in appendix A. Items of knowledge are distributed across the network and reasonable responses are obtained when the net is presented with incomplete, noisy or previously unseen inputs. From the perspective of cognitive modelling of process systems know-how, these pattern recognition and generalization capabilities of neural nets are much more attractive than the symbol manipulation methodology of expert systems, especially as far as complex, ill-defined systems are concerned.

Many parallels can be drawn between the development of knowledge-based systems and that of neural nets. Both had suffered from an overzealous approach in the early stages of their development. In the mid-1980s for example, a common perception had temporarily made its way into the process engineering community that knowledge-based systems had failed to live up to expectations (Allott, 1991). Like their rule-based counterparts, neural nets are also sometimes seen as 'solutions looking for problems'. Although the application of neural nets in the process engineering industry has not matured yet, there is every reason to believe that like other computational methods it will also find a solid niche in this field. A closer look at the historic development of neural nets will underpin the analogous paths of these two branches of artificial intelligence.

The field of neural networks had its inception in the 1940s when the paper of McCulloch and Pitts on the modelling of neurons, and Hebb's book *The Organization of Behaviour* first appeared in the 1940s. The interest sparked by these publications was further buoyed when Rosenblatt presented his Mark I Perceptron in 1958 and Widrow the ADALINE in 1960, but came to a dramatic end in 1969 when Minsky and Papert showed that the capabilities of the linear nets studied at the time were severely limited (Eberhart & Dobbins, 1990). These revelations caused a virtually total cessation in the availability of research funding and many talented researchers left the field permanently. The initial interest in neural nets was only revived again some 14 years later in the early 1980s, and since then the field of neural networks has seen phenomenal growth, passing from a research curiosity to commercial fruition. This growth has in part been fomented by improvements in very large scale integration (VLSI) technology (Goser et al., 1989), as well as the efforts of a small number of investigators who had continued to work during the 1970s, despite a lack of funds and public interest. As had happened to expert systems several years ago, neural network business has soared; from an approximately $7 million industry in 1987, to an estimated $120 million industry in 1990 (Gardner, 1990).

1.4 NEUROCOMPUTERS

Ever since computers became generally available to the process engineer in the 1960s, computing power has grown explosively and this trend is expected to continue in the foreseeable future, with an order of magnitude increase in capability approximately every five years (Stadtherr & Vegeais, 1985). From the humble devices in the 1960s, with not much more computing power than some of today's pocket calculators, new ground was broken with the introduction of Control Data Corporation's CDC 3600 and CDC 6600 machines in the early 1970s. These computers enabled engineers to simulate more complex problems such as radiation damage in metals or the ground state of a hydrogen molecule from first principles. A further order of magnitude increase in computational power was gained when the Cray-1 became available in 1976. The Cray-1 is estimated to be within an order of magnitude from the maximum capability one can expect from a single processor machine, and made the use of multiple processor machines imperative in the quest for more powerful devices.

The concept of parallel computing is not new; the early ENIAC machine (1946) and ILLIAC (mid-1970s) were based on the principle of parallelism. As a result, subsequent generations of Cray machines were all multi-processor machines. The Cray XMP-2x series could calculate two processes simultaneously, the XMP-4x series four processes, the XMP-8x eight processes, etc. Today 64-processors machines are being built along these lines.

Several approaches to parallel computer design can be taken, each appropriate to different methods of solution of chemical engineering problems (Best, 1990). These strategies are mainly concerned with the optimal combination of the number and the power of the process units implemented in the parallel structure and range from carrying out large modules of calculation on relatively few processors (coarse-grained parallelism), to carrying out very small units of calculation on a very large number of primitive processors (fine-grained parallelism). The former strategy is embodied in the design of vector parallel processing machines for example, in which a relatively small number of high performance process units are integrated. These machines are especially well-suited to

applications involving repetitive computations of the same type and are actively researched in Japan[1] and the United States.

In contrast, neurocomputers are based on fine-grained parallelism. These computers consist of arrays of primitive interconnected processors with a small amount of memory that operate concurrently in either a digital or an analog design and are essential for the development of practical applications of neural network technology (Roth, 1990). First generation systems were characterized by pipelined implementations of digital VLSI technology with some low-level parallelism (Hecht-Nielsen, 1988). In reality these systems consisted of neural network simulators and did not exploit the supercomputing potential afforded by the parallel structures or device physics of today's neural networks. Current analog systems can attain high packing densities and are attractive for high speed applications such as discussed in chapter 4. Roth (1990) mentions for example an analog VLSI implementation of a Hopfield neural net with 256 process elements and 130 000 fixed resistive weights that can converge in less than 1.4 μs.

Although considerable progress has recently been made with VLSI technology, the performance of digital neurocomputers is limited by placement and routing problems on the silicon wafers from which they are mostly constructed (Treleaven et al., 1989).

The most powerful neural net chips at present can be compared with the biological intelligence of a fly. The universally reconstructable artificial neural net (URANN) designed by Korea Telcom Research Centre (KTRC), has a size of 13 x 13 mm, 135 424 synapses and an operating speed of approximately 1 GHz.

Siemens' Synapse I is a neurocomputer consisting of eight special chips or neuroprocessors, each of which is capable of 800 000 000 synaptic weightings per second. The chips are based on CMOS technology, and the neurocomputer based on these chips is capable of approximately 5 x 10^9 connections per second.

---

[1]Fujitsu's VPP500/222 with 222 process units for example has a theoretical rating of up to 355 gigaflops.

Analog and digital neurocomputers can both be designed for general or special purpose applications.

1.5.1 General purpose neurocomputers

General purpose neurocomputers are programmable and can support a wide variety of neural network models, analogous to the frameworks provided by traditional computers. A distinction is made between parallel processor arrays and commercial coprocessor boards. Parallel processor arrays have cellular structures composed of large numbers of processing units connected in regular topologies and attain high performance and parallelism through increased numbers of these units. Commercial coprocessors usually consist of signal processing or floating point accelerator boards which can plug into the back of a personal computer or workstation. These accelerators attain high performance by augmenting the processing and storage capabilities of the host computer or workstation (PC, Apollo, Sun, VAX, etc.) by several orders of magnitude.

1.5.2 Special purpose neurocomputers

Special purpose neurocomputers are often very high performance systems based on the direct implementation of neural network models in electronic hardware. The electronic structures of these computers typically resemble the structures of simplified models of biological neurons and are currently enjoying the attention of research groups all over the world (AT&T Bell Laboratories, California Institute of Technology and Bellcore Laboratories in the USA, NEC and Fujitsu in Japan, and Siemens and Texas Instruments in Europe).

1.6 CURRENT APPLICATIONS OF NEURAL NETS IN PROCESS ENGINEERING

1.6.1 Research developments

Research and development have seen rapid growth in recent years and are mainly directed at process fault detection and diagnosis, process control and process modelling and classification. Process fault detection and diagnosis is currently a very important problem in process automation (Sorsa et al., 1991) and the use of neural network techniques

have been studied intensively. Venkatasubramanian et al. (1990) for example, proposed the use of multilayer feedforward neural networks with sigmoidal transfer functions for the fault detection and diagnosis of chemical processes and concluded that these types of networks yield promising results, even when trained with sparse data. The use of feedforward nets to diagnose faults in a heptane-to-toluene process in steady state was similarly investigated by Fan et al. (1993), who recommended the addition of functional links to the input layer of a feedforward neural net, while Hoskins et al. (1991) used a chemical plant simulator (Syschem plant) to generate data for fault diagnosis with a neural net. Despite the complexity of the plant, the system with two hidden layers performed well, and was recommended for use in actual plants. Sorsa et al. (1991) recommended the use of a multilayer perceptron with hyperbolic tangent nodes to detect faults in a heat exchanger-stirred tank experiment, while Naidu et al. (1990) discussed the use of back propagation neural nets to detect sensor failures in non-linear time-invariant plants. Kramer and Leonard (1990) highlighted some of the drawbacks of neural nets used for fault detection, such as poor robustness and difficulty to generalize with sparse data. They recommended inter alia the use of distance-based classifiers and the development of different training algorithms to improve performance.

Investigations related to the use of neural nets in the field of process control are especially numerous. By using a neural net with a single hidden layer to control the pH of a stirred tank system to neutralize wastewater from a plant, Hunt and Sbarbaro (1991) obtained an absolute model mismatch of approximately 5%, which constituted a marked improvement over a traditional controller. In another investigation Hunt et al. (1992) investigated a large variety of different neural net architectures and proposed a structure consisting of a learning vector quantization and a back propagation neural net connected in series in the feedback loop for the optimal control of non-linear process systems. Psichogios and Ungar (1991) similarly investigated direct and indirect model-based control of a non-linear exothermal continuous stirred tank reactor and found the performance of neural networks markedly better than that of a controller based on a linear autoregressive moving average with exogenous input (ARMAX) model. Ydstie (1990) studied direct and indirect adaptive control with a neural net with one hidden layer, and

demonstrated the viability of neural nets for the control of discrete event dynamic systems and processes with non-linear dynamics. Bhat and McAvoy (1990) led an investigation into the control of the pH of a continuous stirred tank and showed neural nets to provide a more generalized methodology than that based on the traditional autoregressive moving average (ARMA) and convolution models used for control.

In some of the work concerning the construction of connectionist models of ill-defined processes, Kito et al. (1992) used a back propagation neural net to estimate the strength of acid sites generated synergistically in binary mixed oxides. The acid strength was modelled in terms of various physico-chemical properties and the results were found to be in good agreement with experimental data. Karim and Rivera (1992) demonstrated feedforward and recurrent neural net methodologies to estimate the state of fermentation processes. They came to the conclusion that the conjugate gradient method of training these nets to minimize the error of predictions was more efficient than methods based on steepest descent. Both types of net yielded comparable results, but the more complex recurrent net took a longer time to train.

The use of neural nets to predict the long term behaviour of chemical processes has also come under close scrutiny recently. As an alternative to many-steps-ahead prediction with multilayer back propagation neural nets, Su and McAvoy (1992) proposed the use of a parallel system identification method. They derived a training algorithm for an external recurrent neural net and used the net to identify the dynamic behaviour of a biological wastewater plant and a catalytic reformer in a petroleum refinery. In another investigation Rico-Martínez et al. (1992) studied the capability of neural nets to predict the long term behaviour and observed bifurcations in the electrodissolution of copper in phosphoric acid. They used two networks in series, the first as a non-linear principal component extractor and the second to process the data, and reported close agreement between the results of the model and experimental data.

Reuter et al. (1993) and Reuter and Van Deventer (1990) proposed a simple generalized approach for the simulation and identification of batch and mixed flow mineral processing and metallurgical reactors with neural

nets. The method is based on the use a trained neural net to relate the parameters of the kinetic rate equations to the process conditions and is demonstrated with examples including among other a Tennessee copper rougher circuit, a Nchanga sulphide rougher circuit, zinc-ferrite leaching and the precipitation of jarosite. Reuter et al. (1992) demonstrated the use of sigmoidal back propagation neural nets for the modelling of complex metal-slag equilibrium processes. By making use of published data they showed that a small network with a single hidden layer could be trained successfully with relatively sparse process data to model among other the activities in binary metal solutions, the distribution of manganese and sulphur between pig iron and slag, the distribution of copper between metal and slag, the distribution of tin oxides and iron oxides between metal and slag, as well as the viscosity of lead smelting slags. Neural nets were shown to be similarly successful in the modelling of the kinematic viscosities of crude oil fractions (Van der Walt et al., 1993a).

In order to overcome some of the shortcomings of back propagation neural networks used for the modelling of multivariable processes of high dimension, Van Der Walt and Van Deventer (1992, 1993) proposed the use of a hybrid subspace neural net model which could make better use of sparse data to extract the characteristic features of a process. As a first step the global variable space is characterized by a trained neural net, which is used to estimate the first order partial derivatives of the system. This is followed by a perturbation analysis which is used to subdivide the global variable space into various subspaces, each of which incorporates only those independent variables which influence the dependent variables significantly in the particular domain of the subspace. These subspaces can subsequently be modelled more accurately by making use of (neural net) models to relate the reduced number of independent variables to the dependent variables. Their method has been demonstrated with a carbon-in-leach (CIL) reactor. These authors suggested the use of higher order neural networks and proposed a regression network (Van der Walt et al., 1993, 1993e) consisting of arrangements of nodes with various types of activation functions and additive and multiplicative summation rules, which allow the combination of parametric, as well as non-parametric relationships in the neural net model. The problem posed by sparse process data can

thus be alleviated to some extent in that some of the relationships between variables are specified beforehand. The method was demonstrated through the modelling of a hydrocyclone classifier.

Not much attention has been paid to the use of neural net expert systems in the processing industry. In one of the few studies reported Smets and Bogaert (1992) used two neural networks to predict the stress corrosion of austenitic stainless steels in chloride-bearing water. The nets were trained by means of case histories of failures of these metals under similar circumstances and managed to extract the principal features of these processes remarkably well.

These examples of research and development efforts are by no means exhaustive, but are merely intended to serve as an indication of the vast scope and potential of neural net techniques in the chemical and metallurgical industries.

1.6.2 Commercial applications

Despite the promise artificial neural networks appear to hold for the chemical and metallurgical processing industries, the first commercial applications for neural nets only saw the light approximately a year (1992) ago, with the implementation of a hybrid neural net/fuzzy control system from Pavilion Technologies in Eastman Kodak's refinery in Texas. Other commercial applications include hybrid control systems sold by Neural Applications Corporation, consisting of neural nets as well as expert systems, used in arc furnaces. These systems are used to optimize the positions of the electrodes of the arc furnaces used for the smelting of scrap metal in steel plants, and are estimated to save approximately $US 2 000 000 annually on the operating costs of each furnace.

The most recently reported commercial application of a neural net in the process industry concerns the control of a nuclear fusion reactor at AEA Technology's Culham Laboratory in Oxfordshire (Geake, 1993). The optimal conditions for fusion in the Compass tokamak reactor occur where the turbulence in the plasma is minimal, and cannot be calculated fast enough by conventional computers, which can take hours or even days to compute the setup of the magnetic fields needed to produce

suitable plasma shapes in the reaction chamber. The problem is solved by making use of a neural net that can do the necessary calculations in approximately ten microseconds (significantly faster than the fluctuations in the plasma that typically last for a few hundred milliseconds). The Compass net obtains data from 16 magnetic field sensors inside the chamber and has four output nodes linked to the magnetic controls of the system. An added advantage is the flexibility of the network, which can be retrained (by sets of approximately 2000 exemplars at a time) when the implementation of different control strategies are warranted. Conventional controllers in contrast, can only cope with narrow ranges of process conditions.

1.6.3 Future neurocomputer technologies

As far as the construction of computer circuitry goes, silicon is doubtlessly the medium of choice, not least due to its abundant availability and cheapness. Despite the use of multilayers and epitaxy techniques, the material as used in the industry today poses many significant disadvantages with regard to the design of neuralware, where packing density and compactness are of the utmost importance.

As a result impressive progress has been made with alternative computer technologies. Of these, optical or electro-optical computing is probably the closest to maturity (Louri, 1991; Lupo, 1989). Light beams have a far greater capacity to carry information than electronic circuits (Canham, 1993,) and the optical implementation of neural nets has the potential for attaining very high connectivity, because beams of light can pass through one another without undue interference (Abu-Mostafa & Psaltis, 1987; Roth, 1990; Williams, 1987). Until recently the expanding field of optoelectronics has been hampered by the use of complex and expensive materials such as gallium arsenide, but other materials such as porous silicon has recently been identified as a promising new alternative for the construction of computer devices (Canham, 1993).

Other technologies being investigated include molecular computing which could in principle be based on the formation of supramolecular switching devices that can self-assemble and then switch between different configurations, but are not expected to reach maturity in the foreseeable future (Bradley, 1993).

1.7 SPECIFIC OBJECTIVES OF THIS STUDY

It is the overall objective of this study to develop new strategies for the modelling and simulation of processing systems based on the use of connectionist methods. Since these techniques are essentially data driven, a large part of the investigation focuses on the use of neural networks for the processing of raw plant data in order to improve the quality of models based on these data. In the final part of this dissertation the construction of connectionist plant models (which could be based on plant data processed by the techniques described in the previous chapters) is investigated. Both these aspects related to ill-definedness, as well as the computationally intensive nature of process modelling are addressed. Chapters 2, 3 and 5 are devoted to the use of feedforward neural nets and chapter 4 explores the parallelist implementation of traditional algorithms in feedback connectionist structures to enhance computing power. The specific objectives of this dissertation are addressed in sequence in the following chapters and include the:

- estimation of variation in process data, especially if these data reflect quasi-steady state behaviour;
- detection and location of gross errors in plant data, especially based on plant models subject to non-linear constraints;
- more efficient reconciliation of noisy process data, and
- modelling and optimization of metallurgical plants and process circuits with neural nets and comparison with traditional models.

CHAPTER 2

Estimation of measurement error variances

Summary

Measurements of flow variables generally violate material balance constraints owing to the presence of random and possibly systematic errors in the data. The variance-covariance matrix of the measurement errors which is required to solve the data reconciliation problem, is unfortunately not always available and has to be estimated. These estimates are based on historic data or redundant observations of the state variables over a period of time. By making use of conventional techniques the variances and covariances of the measurement errors can be estimated from the corresponding variances and covariances of the constraint residuals, in conjunction with additional heuristic information, provided the constraints are linear. In chapter 2 it is shown that neural nets can similarly be used to estimate these variances and covariances in process systems, regardless of the nature of the constraints imposed on the measurement data.

2.1 OBJECTIVES OF THIS CHAPTER

In chapter 2 the use of neural nets to estimate stochastic parameters in quasi-steady state process systems is explored. In particular

- Appropriate connectionist structures and neurodynamics are investigated as a basis for the estimation of variance-covariance matrices of chemical process systems;
- The use of these neural nets to estimate the variances of systems subject to linear and non-linear constraints is investigated, and
- The accommodation of heuristic information necessary to estimate the variance-covariance matrix of the system is investigated.

2.2 BACKGROUND THEORY

The monitoring of plant performance and the verification of system models are crucially dependent on reliable sets of steady state component and total flow rate data. In general the measured data violate the process constraints of the system, owing to random fluctuations in the observed values, or even systematic errors in these values due to

erroneous measurements or large discrepancies between the actual behaviour of the system and the behaviour predicted by the system model. The observations or measurements consequently have to be reconciled with the process constraints, usually through the minimization of a quadratic criterion. This criterion is typically a function of the differences between the measured and the adjusted values, weighted by the inverse of the measurement error covariance matrix (Hodouin & Everell, 1980). General reconciliation methods (discussed in more detail in chapter 4) are usually based on the assumption that measurement errors are randomly Gaussian with a known covariance matrix, and distributed around a zero mean. In many practical situations, this matrix is unknown and has to be estimated (Holly, et al., 1989; Narasimhan et al., 1986). The most obvious method of doing so involves the analysis of N redundant observations $x_{j,1}$, $x_{j,2}$, .. $x_{j,N}$ of the particular state variable X_j over a period of time. That is

$$\text{var}(X_j) = [\, N - 1 \,]^{-1} \sum_{i=1}^{N} (x_{j,i} - x_{j,AVG})^2 \qquad (2.1)$$

$$\text{cov}(X_j, X_k) = [\, N - 1 \,]^{-1} \sum_{i=1}^{N} (x_{j,i} - x_{j,AVG})(x_{k,i} - x_{k,AVG}) \qquad (2.2)$$

This method of estimating variances or covariances is reliable only if the system is truly in a steady state, which is not generally the case (Almasy and Mah, 1984). In most process systems steady state is a relative concept, defined by the time frame over which the system is considered. In process systems the expected values of the state variables of the system are generally subject to change, which could be manifested by a series of small fluctuations around a fixed point in the variable space, or a gradual drift from such a point, and when these deviations from steady state behaviour are significant, the system is considered to be in a quasi-steady state. The estimation of the stochastic parameters prevailing in systems such as these at a particular instant is consequently impeded by the change in the system's behaviour during the succee[illegible]rval over which sampling takes place. Under these circums[illegible]use of an indirect method of estimating the variances of the o[illegible]riables has been recommended (Almasy & Mah, 1984).

In the proposed method the relationship between the covariances of the residuals of the quasi-steady state process system constraints and the covariances of the measurement residuals is used as a basis to derive a more accurate estimate of the covariances of the state variables than is possible with equations 2.1 & 2.2. (In this dissertation *variance* is meant to indicate an element on the diagonal of a covariance matrix, while *covariance* is meant to indicate an off-diagonal element of such a covariance matrix). Unlike the measurement errors, the constraint residuals can be computed directly from measurements $x_{i,k}$, without requiring an estimate of the true value of the state variable X_i. It is consequently possible to compute the sample covariance matrix of the residuals of a set of system variables and use these values in conjunction with other heuristic information to estimate the covariance matrix of the measurements of these variables. The additional information concerns the relationships between the elements of the covariance matrix of the measurements, for example the assumption that the off-diagonal elements of the matrix are zero or very small owing to the use of independent instruments in the measurement of the state variables, or that the expected values of the different state variables all have the same tendency to drift in a particular direction in the variable space. Unfortunately the relationship between the covariance matrix of the measurement errors and the constraint residuals is well-defined for linear systems only, which poses a severe restriction on the procedure as regards its use in the chemical engineering industry.

In this dissertation a new method to estimate the covariance matrix of observed variables, similarly based on the relationship between the covariances of the constraint residuals and the covariances of the measurement residuals described in the literature (Almasy & Mah, 1984; Keller et al., 1992), is proposed. In contrast to the methods described, this technique, which involves the use of an artificial neural net to represent the relationship between the covariances of the constraint residuals and that of the measurement errors, does not require the system to be linear, or the relationship between the constraint and the measurement residuals to be well-defined. (The fundamentals of the neural nets on which the method is based is described in more detail in appendix A.) Complementary heuristic information required in the

estimation of the covariance matrix of the measurement errors can furthermore be incorporated in the structure of the net itself.

2.3 PROBLEM STATEMENT

Consider a process described by the following system of linear equations

$$\mathbf{C}.\mathbf{x} = \mathbf{0} \tag{2.3}$$

where $\mathbf{C}$ is a (n x p) coefficient matrix with $n < p$, representing the system constraints, and $\mathbf{x}$ is the true value of the state vector. The observed or measured value $\mathbf{x}'$, of this vector

$$\mathbf{x}' = \mathbf{x} + \mathbf{e} \tag{2.4}$$

is typically subject to an error $\mathbf{e}$, so that the constraint residuals $\mathbf{r}$ are related to the measurement vector $\mathbf{x}'$ as follows

$$\mathbf{r} = \mathbf{C}.\mathbf{x}' = \mathbf{C}.(\mathbf{x} + \mathbf{e}), \text{ where} \tag{2.5}$$

$$\mathbf{C}.(\mathbf{x} + \mathbf{e}) = \mathbf{C}.\mathbf{x} + \mathbf{C}.\mathbf{e} = \mathbf{C}.\mathbf{e} \tag{2.6}$$

If $\mathbf{e}$ is considered to be a Gaussian variable with a zero mean and a covariance matrix $\mathbf{V}_{|e}$, then

Measured variables

$$E(\mathbf{e}) = \mathbf{0} \tag{2.7}$$

$$\text{var}(\mathbf{e}) = E(\mathbf{e}\mathbf{e}^T) = \mathbf{V}_{|e} \tag{2.8}$$

Constraint residuals

$$E(\mathbf{r}) = E(\mathbf{C}.\mathbf{e}) \tag{2.9}$$

$$\text{var}(\mathbf{r}) = E[(\mathbf{C}.\mathbf{e})(\mathbf{C}.\mathbf{e})^T] = \mathbf{V}_{|r} \tag{2.10}$$

Since the system is linear

$$E(\mathbf{r}) = \mathbf{C}.E(\mathbf{e}) \tag{2.11}$$

$$\text{var}(\mathbf{r}) = \mathbf{C}.E(\mathbf{e}\mathbf{e}^T).\mathbf{C}^T = \mathbf{C}\mathbf{V}_{|e}\mathbf{C}^T = \mathbf{V}_{|r} \tag{2.12}$$

If the system is non-linear, such as would arise from conservation equations involving the products of stream flows (**F**) and component fractions (**f**), as well as other non-linear relationships which may ensue from different process circuit configurations, the relationship expressed by equation 2.3 is no longer valid and can not be used in traditional analytical procedures to obtain estimates of the statistical properties of the measured variables, i.e.

$$(\mathbf{F})^T.\mathbf{C}.\mathbf{f} = \mathbf{0} \qquad (2.13)$$

where

$\mathbf{F}' = \mathbf{F} + \mathbf{e}_F$, and $\mathbf{f}' = \mathbf{f} + \mathbf{e}_f$, so that

$$\mathbf{r} = \mathbf{F}^T.\mathbf{C}.\mathbf{f} = (\mathbf{F} + \mathbf{e}_F)^T.\mathbf{C}.(\mathbf{f} + \mathbf{e}_f) \qquad (2.14)$$

$$\mathbf{r} = (\mathbf{F})^T.\mathbf{C}.\mathbf{e}_f + (\mathbf{e}_F)^T.\mathbf{C}.\mathbf{f} + (\mathbf{e}_F)^T.\mathbf{C}.\mathbf{e}_f \qquad (2.15)$$

The constraint residuals are thus related to the measurement residuals $\mathbf{e}_F$ and $\mathbf{e}_f$ not only in terms of the constant coefficient matrix **C**, but also in terms of the state variables, **F** and **f**, the true values of which are not known exactly. This ill-defined relationship between the constraint and measurement residuals restricts the use of analytical methods for estimating the covariances of state variables (based on the covariances of the constraint residuals) to linear systems. This complication can be avoided by making use of an artificial neural net to represent the relationship between the properties of the constraint residuals and those of the measurement residuals. This relationship is not unique and needs to be further defined by additional heuristic knowledge.

2.4 ESTIMATION OF COVARIANCES BY MEANS OF NEURAL NETS

By presenting an artificial neural net with exemplars of the relationship between the covariances of the constraint residuals and that of the measurement errors, the net forms an internal representation of this relationship which can be used to provide an estimate of the covariance matrix of the measurement errors.

The general structure of the net is determined by the number of process variables and constraints of the system. If the system is generally

described by N system variables and M process constraints, the corresponding net has an input layer consisting of M^2 process nodes, and an output layer consisting of N^2 process nodes. This general structure of the neural net can be modified to account for the incorporation of additional information in the system. If the covariances of the state variables are deemed to be negligible for example, the circuit structure can be reduced to accommodate the diagonal of the covariance matrix of the measurement errors only. Besides alteration of the structure of the net to reflect additional knowledge of the process, such knowledge can also be accommodated in the set of exemplars used to train the net. The use of neural nets such as these is demonstrated in the following examples.

2.4.1 Example 2.1 (General process circuit subject to linear process constraints)

Consider the following generalized process circuit consisting of 4 nodes and 9 flow streams, F_1, F_2, .. F_9, as shown in figure 2.1. A material balance around each node yields the following system of linear equations

Node A: $F_1 - F_2 + F_8 = 0$ (2.16)

Node B: $F_2 - F_3 + F_5 - F_7 = 0$ (2.17)

Node C: $F_3 - F_4 - F_6 = 0$ (2.18)

Node D: $F_6 + F_7 - F_8 + F_9 = 0$ (2.19)

To illustrate the use of both a direct and an indirect (neural net) method to estimate variances, and in order to better evaluate the performance of different neural net structures, the system is considered to consist of two subsets of variables. The first subset (subset 1) comprized of variables F_1, F_2, F_6, F_7, F_8 and F_9 displays steady state benaviour and the expected values of all the variables in the subset are consequently independent of time. The second subset (subset 2) is comprized of variables F_3, F_4 and F_5 and displays quasi-steady state behaviour, in that the expected values of the variables of the subset are dependent on time. The time dependence of the variables in both subsets is indicated in table 2.1.

For convenience the variables are assumed to be monitored independently, so that the off-diagonal elements of the covariance matrix of these variables are zero, making it necessary to estimate the diagonal elements of the covariance matrix only. In order to estimate these variances, the process variables F_1-F_9 were measured at time intervals t_1-t_{10}, as indicated in table 2.2. It is from these observations that an estimate of the variances of the process variables at time t_0 is required.

The artificial neural net used to estimate the variances of the process variables in the circuit, consisted of an input layer with 4 process nodes, corresponding to the variances of the 4 constraint residuals, and an output layer with 9 process nodes, corresponding to the variances of the 9 process variables, similar to the generalized system portrayed in figure 2.2. In order to train the net, a set of exemplars composed of the variances of the process constraint residuals and the corresponding variances of the process variables was generated (a variance of 0.0134 was used, based on a variable with a unitary expected value). As can be seen from figure 2.3, the net managed to form a generalized internal representation of the relationship between the variances of the process variables and those of the constraints. The weight matrix of the trained net is shown in table 2.3.

The variances of the constraint residuals calculated from the sample measurements at times t_0-t_{10} were subsequently presented to the net and the corresponding variances of the process variables were estimated. These estimates which are shown in table 2.4, are more accurate than the estimates calculated directly from the sample data (also shown in table 2.4).

2.4.2 Example 2.2 (Two-product separator subject to non-linear process constraints)

In this example a high tension roll separator is considered. The separator classifies a feed stream F_1, consisting of 2 components with mass fractions $f_{1,1}$ and $f_{1,2}$, into two product streams F_2 with component mass fractions $f_{2,1}$ and $f_{2,2}$, as well as F_3, with component mass fractions $f_{3,1}$ and $f_{3,2}$, as shown in figure 2.4. The flow streams and mass fractions are measured and typically violate the conservation equations of the system, viz.

$$F_1 - F_2 - F_3 = 0 \qquad (2.20)$$

$$F_1 f_{1,1} - F_2 f_{2,1} - F_3 f_{3,1} = 0 \qquad (2.21)$$

$$F_1 f_{1,2} - F_2 f_{2,2} - F_3 f_{3,2} = 0 \qquad (2.22)$$

Contrary to the previous system considered in example 2.1, all the variables in this system are in a quasi-steady state, so that a direct estimate of the covariance matrix of the state variables at time t_0 yields inaccurate results. The expected values of the system variables are shown in table 2.5. Owing to the non-linearity of the process constraints (equations 2.20-2.22), the relationship between the constraint residuals **r** and the measurement errors $\mathbf{e}_F$ and $\mathbf{e}_f$ is ill-defined (the true values of F_i and $f_{i,j}$ are unknown). The difficulties ensuing from the ill-definedness of this relationship make the use of a traditional numerical procedure inappropriate, whereas the use of an artificial neural net is not affected by the ill-definedness of the relationship between the variable and constraint residual variances and provides an effective means of estimating the covariances of the system variables.

To obtain an estimate of the covariance matrix $\mathbf{V}_{|F,f}$ of the process variables F_i and $f_{i,j}$ $(i,j = 1,2,3)$, an arbitrary set of values of the variables is corrupted by errors with known covariances (and zero means) and a neural net with a structure similar to that of the net shown in figure 2.2 is subsequently trained by means of these artificially generated exemplars to construct an internal representation of the relation between the covariances of the resultant constraint residuals and that of the variable residuals. The particular set of values used as a basis for the generation of synthetic training data is not critical, as long as it is large enough to ensure that the covariances of the constraint residuals (on which the estimation of the covariances of the variable measurements will be based) would be a subset of the training set. Failure to do so could result in grossly inaccurate estimates of the covariances of the variable measurements. Additional heuristic information on which the estimation of variances is based is either incorporated in the structure of the net itself, or in the exemplars used to train the net. If the interactions between variables are considered to be negligible for example, the exemplars used for training could only contain data relating variable and residual variances, or else the exemplars could contain data relating

variable and residual covariances in general and the neural net could then be structured so that the weights of the net relating variable and residual covariances are zero. In addition to the relationship between the statistical properties of variable measurements and those of the constraint residuals, all measurements are considered to be totally independent, so that all off-diagonal elements of the covariance matrix are deemed to be zero. This information is incorporated directly into the structure of the neural net, which could subsequently be trained to construct an internal representation of the relationship between the variances of the constraint and measurement residuals.

As before, the back propagation neural net trained to estimate the covariance matrix of the state variables consisted of an input and an output layer only. The input layer was composed of 4 computational elements, corresponding to the variances of each of the 4 constraint residuals, while the output layer consisted of 9 process nodes which corresponded to the variances of each of the 9 system variables. The net was trained by repeatedly presenting it with exemplars of the relation between the variances of the constraint and measurement residuals. Training of the sigmoidal output units was accomplished by the generalized delta rule (Rumelhart et al., 1986; Leonard & Kramer, 1990) also described in appendix A, through which the weights of the net could be modified until it was able to form an internal representation of the relationship between the covariance matrix of the flow variables $\mathbf{V}_{|F,f}$ and the covariance matrix of the constraint residuals $\mathbf{V}_{|r}$, as shown in figure 2.5. After convergence, the trained net was used to estimate the covariance matrix $\mathbf{V}_{|F,f}$ of the system variables, by presenting it with the computed sample variances of the measurement residuals. These variances were calculated directly from the measured data shown in table 2.6 by means of equation 2.1. The estimates obtained by the neural net are compared with the actual estimates in table 2.7. The weight matrix of the trained net is shown in table 2.8.

To further illustrate the way in which neural nets can be used in conjunction with heuristic data to estimate variances of sets of measurements, the assumption that the covariances of the variables are zero is modified by assuming a correlation between the measurements of the component mass fractions $f_{1,1}'$, $f_{1,2}'$, $f_{2,1}'$, $f_{2,2}'$, $f_{3,1}'$ and $f_{3,2}'$, i.e.

non-zero covariances $cov(f_{1,1}|f_{1,2})$, $cov(f_{2,1}|f_{2,2})$ and $cov(f_{3,1}|f_{3,2})$. Owing to the quasi-steady state of the system, direct computation of the sample covariances of the system from measurements such as those shown in table 2.9 is inaccurate once again and as a consequence the covariances have to be estimated by an indirect procedure. A neural net similar to the one used previously in conjunction with the assumption of zero covariances can again be used. In this case the net's structure would have to be modified to accommodate the three covariances of the mass fraction variables. The net is consequently composed of an input layer with 3 process nodes (one for each variance and covariance of the constraint residuals), as well as 12 output nodes (one for the variance of each of the 9 variables, as well as the three covariance elements).

After training, the net is presented with the sample variances of the residuals, from which the 9 variances and 3 covariances are estimated. The estimated and the actual covariances of the process variables are shown in table 2.10. (Since estimates of the variances are very similar to those shown earlier on, only the estimates of $cov(f_{1,1}|f_{1,2})$, $cov(f_{2,1}|f_{2,2})$ and $cov(f_{3,1}|f_{3,2})$ are shown in table 2.10.) The weights of the trained net used to estimate these covariances are shown in table 2.11. It is clear that the estimates made by the neural net are more accurate than the estimates based on direct computation of the sample covariances (equation 2.2).

2.5 CONCLUSIONS AND SIGNIFICANCE

As was mentioned previously, the relationship between the covariance matrix of the measurement errors and the covariance matrix of the constraint residuals is not uniquely defined, and further restrictions need to be introduced before the covariances of the measurement errors can be estimated from the covariances of the constraint residuals. This heuristic information can be directly accounted for in the structure of the neural net, or it can be embedded in the exemplars used to train the net. When the neural net is presented with the ambiguous data relating the variances of the measurement errors ($\mathbf{V}_{|e}$) with those of the constraint residuals ($\mathbf{V}_{|r}$), it relates $\mathbf{V}_{|r}$ with the average $\mathbf{V}_{|e}$ of the system.

The neural net forms an internal representation of the relation between the covariances of the constraint residuals and the covariances of the state variables (similar to the representation of the variances), which allows estimates of the covariances of variables in the quasi-steady state process system that are generally more accurate than when these covariances are computed directly from the observed measurements. The method is not dependent on the linearity of the system and once a net is trained for a particular system configuration, no retraining needs to be done to estimate the covariance matrix of the state variables of the system. This feature together with their parallel architectures (Hecht-Nielsen, 1990) make neural nets very attractive in on-line process monitoring systems.

In this dissertation only linear and bilinear systems (typical of many process engineering systems) have been considered. It was shown that these systems can be accounted for by simple single layer back propagation neural nets with sigmoidal computational elements. To summarize:

- Neural nets can be used to estimate the covariance matrices of the variables of quasi-steady state process systems more accurately than can be derived from direct observation of these variables.
- Contrary to more traditional procedures, the use of neural nets is not restricted by the nature of the relationship between the covariances of the constraint residuals and those of the measurement errors.
- Due to their parallel architecture and the capability of trained nets to provide direct estimates of the covariance matrices of the state variables of process systems, neural nets show considerable potential for use in on-line process monitoring systems.

2.6 TABLES REFERRED TO IN CHAPTER 2

TABLE 2.1 Expected values of process variables* (example 2.1)

TIME	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
t_0	1	4	***4.0***	***3.0***	***1.0***	1	1	3	1
t_1	1	4	***4.2***	***3.2***	***1.2***	1	1	3	1
t_2	1	4	***4.4***	***3.4***	***1.4***	1	1	3	1
t_3	1	4	***4.6***	***3.6***	***1.6***	1	1	3	1
t_4	1	4	***4.8***	***3.8***	***1.8***	1	1	3	1
t_5	1	4	***5.0***	***4.0***	***2.0***	1	1	3	1
t_6	1	4	***5.2***	***4.2***	***2.2***	1	1	3	1
t_7	1	4	***5.4***	***4.4***	***2.4***	1	1	3	1
t_8	1	4	***5.6***	***4.6***	***2.6***	1	1	3	1
t_9	1	4	***5.8***	***4.8***	***2.8***	1	1	3	1
t_{10}	1	4	***6.0***	***5.0***	***3.0***	1	1	3	1

*Quasi-steady state variables are distinguished from steady state variables by ***bold italics***

TABLE 2.2 Measurements of process variables (example 2.1)

	F_1'	F_2'	F_3'	F_4'	F_5'	F_6'	F_7'	F_8'	F_9'
t_0	1.058	4.646	3.342	3.394	0.810	1.196	1.043	3.461	0.842
t_1	1.106	3.280	4.402	2.448	0.921	0.929	0.922	3.009	0.803
t_2	0.802	3.387	4.610	3.143	0.903	0.921	1.109	2.723	0.868
t_3	0.948	4.741	3.500	3.685	1.185	1.199	0.918	2.513	0.826
t_4	0.962	3.783	3.828	3.041	1.079	1.033	0.991	3.128	1.124
t_5	1.185	4.631	3.657	3.959	1.633	1.118	0.916	2.465	0.969
t_6	0.907	4.632	4.934	3.183	1.679	1.162	0.936	2.978	0.914
t_7	0.956	4.194	4.922	3.328	2.181	1.140	0.998	2.600	0.873
t_8	1.033	4.389	4.340	4.502	1.870	1.143	0.873	3.136	1.025
t_9	0.877	4.373	5.387	5.013	2.995	0.960	0.984	2.801	0.941
t_{10}	1.197	4.507	6.282	4.329	3.464	0.872	1.137	2.696	1.043

TABLE 2.3 Weight matrix of neural net used to estimate variances of flow variables (example 2.1)

	$v_{o,1}$	$v_{o,2}$	$v_{o,3}$	$v_{o,4}$	$v_{o,5}$	$v_{o,6}$	$v_{o,7}$	$v_{o,8}$	$v_{o,9}$
BIAS	-1.375	-1.267	-1.314	-1.283	-1.219	-1.154	-1.357	-1.316	-1.289
$v_{i,1}$	0.738	0.717	0.715	0.751	0.621	0.589	0.681	0.725	0.620
$v_{i,2}$	0.693	0.792	0.701	0.659	0.645	0.701	0.680	0.593	0.688
$v_{i,3}$	0.641	0.565	0.719	0.717	0.472	0.601	0.697	0.567	0.645
$v_{i,4}$	0.689	0.650	0.611	0.676	0.632	0.589	0.725	0.677	0.616

TABLE 2.4 Estimated and actual variances of process variables (example 2.1)

Actual variances

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
0.013	0.208	0.208	0.117	0.013	0.013	0.013	0.117	0.013

Estimates based on direct method (equation 2.1)

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
0.014	0.245	0.715	0.510	0.703	0.013	0.006	0.085	0.009

Estimates made by neural net

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
0.013	0.280	0.382	0.221	0.021	0.021	0.015	0.092	0.008

TABLE 2.5 Expected values of process variables (example 2.2)

TIME	F_1	F_2	F_3	$f_{1,1}$	$f_{1,2}$	$f_{2,1}$	$f_{2,2}$	$f_{3,1}$	$f_{3,2}$
t_0	1.000	0.350	0.650	0.260	0.740	0.297	0.703	0.240	0.760
t_1	1.100	0.370	0.730	0.280	0.720	0.398	0.602	0.220	0.780
t_2	1.050	0.380	0.670	0.290	0.710	0.431	0.569	0.210	0.790
t_3	1.150	0.360	0.790	0.250	0.750	0.360	0.640	0.200	0.800
t_4	1.170	0.390	0.780	0.240	0.760	0.300	0.700	0.210	0.790
t_5	1.220	0.410	0.810	0.270	0.730	0.448	0.552	0.180	0.820
t_6	1.180	0.400	0.780	0.290	0.710	0.524	0.476	0.170	0.830
t_7	1.210	0.420	0.790	0.310	0.690	0.555	0.445	0.180	0.820
t_8	1.270	0.405	0.865	0.340	0.660	0.746	0.254	0.150	0.850
t_9	1.280	0.440	0.840	0.350	0.650	0.770	0.230	0.130	0.870
t_{10}	1.300	0.430	0.870	0.360	0.640	0.805	0.195	0.140	0.860

TABLE 2.6 Measurements of process variables (example 2.2)

TIME	F_1'	F_2'	F_3'	$f_{1,1}'$	$f_{1,2}'$	$f_{2,1}'$	$f_{2,2}'$	$f_{3,1}'$	$f_{3,2}'$
t_0	0.988	0.361	0.548	0.232	0.599	0.249	0.747	0.217	0.734
t_1	0.944	0.346	0.690	0.235	0.745	0.352	0.672	0.195	0.816
t_2	1.162	0.374	0.775	0.238	0.581	0.473	0.455	0.191	0.783
t_3	1.272	0.382	0.852	0.257	0.631	0.316	0.732	0.168	0.855
t_4	1.146	0.466	0.917	0.282	0.745	0.302	0.659	0.188	0.768
t_5	1.184	0.365	0.796	0.233	0.808	0.466	0.509	0.215	0.786
t_6	1.021	0.459	0.889	0.236	0.792	0.506	0.552	0.149	0.695
t_7	1.438	0.389	0.730	0.316	0.681	0.563	0.518	0.192	0.776
t_8	1.509	0.359	1.032	0.354	0.621	0.753	0.212	0.145	0.790
t_9	1.328	0.368	1.006	0.346	0.552	0.864	0.204	0.155	0.846
t_{10}	1.459	0.419	0.721	0.296	0.724	0.673	0.181	0.167	0.778

TABLE 2.7 Estimated and actual variances of process variables (example 2.2)

Actual variances

F_1	F_2	F_3	$f_{1,1}$	$f_{1,2}$	$f_{2,1}$	$f_{2,2}$	$f_{3,1}$	$f_{3,2}$
0.0134	0.0016	0.0057	0.0009	0.0073	0.0012	0.0066	0.0008	0.0077

Estimates based on direct method (equation 2.1)

F_1	F_2	F_3	$f_{1,1}$	$f_{1,2}$	$f_{2,1}$	$f_{2,2}$	$f_{3,1}$	$f_{3,2}$
0.0123	0.0032	0.0178	0.0039	0.0065	0.0422	0.0412	0.0011	0.0118

Estimates made by neural net

F_1	F_2	F_3	$f_{1,1}$	$f_{1,2}$	$f_{2,1}$	$f_{2,2}$	$f_{3,1}$	$f_{3,2}$
0.0120	0.0015	0.0051	0.0008	0.0069	0.0011	0.0061	0.0007	0.0071

TABLE 2.8 Weight matrix of neural net used to estimate variances of flow variables in non-linear system (example 2.2)

	$v_{o,1}$	$v_{o,2}$	$v_{o,3}$	$v_{o,4}$	$v_{o,5}$	$v_{o,6}$	$v_{o,7}$	$v_{o,8}$	$v_{o,9}$
BIAS	-1.203	-1.220	-1.207	-0.532	-0.514	-0.531	-0.507	-0.487	-0.529
$v_{i,1}$	2.197	2.263	2.217	-1.361	-1.482	-1.444	-1.413	-1.560	-1.438
$v_{i,2}$	0.141	0.211	0.091	1.426	1.411	1.413	1.438	1.565	1.514
$v_{i,3}$	0.232	0.153	0.255	1.258	1.455	1.397	1.303	1.384	1.307

TABLE 2.9 Measurements of process variables (covariances) (example 2.2 - non-zero covariances)

TIME	F_1'	F_2'	F_3'	$f_{1,1}'$	$f_{1,2}'$	$f_{2,1}'$	$f_{2,2}'$	$f_{3,1}'$	$f_{3,2}'$
t_0	1.080	0.301	0.779	0.215	0.785	0.319	0.215	0.286	0.714
t_1	0.892	0.334	0.558	0.325	0.675	0.443	0.325	0.182	0.818
t_2	1.027	0.420	0.608	0.261	0.739	0.386	0.261	0.220	0.780
t_3	1.015	0.302	0.713	0.257	0.743	0.345	0.257	0.168	0.832
t_4	1.106	0.337	0.768	0.249	0.751	0.327	0.249	0.200	0.800
t_5	1.376	0.463	0.913	0.322	0.678	0.457	0.322	0.177	0.823
t_6	1.308	0.378	0.930	0.300	0.700	0.521	0.300	0.142	0.858
t_7	1.334	0.489	0.845	0.303	0.697	0.535	0.303	0.159	0.841
t_8	1.020	0.389	0.631	0.293	0.707	0.867	0.293	0.176	0.824
t_9	1.157	0.361	0.795	0.305	0.695	0.771	0.305	0.150	0.850
t_{10}	1.225	0.500	0.725	0.298	0.702	0.766	0.298	0.142	0.858

TABLE 2.10 Estimated and actual covariances of process variables (example 2.2 - non-zero covariances)

Actual covariances

$cov(f_{1,1}\|f_{1,2})$	$cov(f_{2,1}\|f_{2,2})$	$cov(f_{3,1}\|f_{3,2})$
0.00090	0.00118	0.00077

Estimates based on direct method (equation 2.2)

$cov(f_{1,1}\|f_{1,2})$	$cov(f_{2,1}\|f_{2,2})$	$cov(f_{3,1}\|f_{3,2})$
0.00202	0.03633	0.00124

Estimates made by neural net

$cov(f_{1,1}\|f_{1,2})$	$cov(f_{2,1}\|f_{2,2})$	$cov(f_{3,1}\|f_{3,2})$
0.00117	0.00138	0.00096

TABLE 2.11 Weight matrix of neural net used to estimate covariances of flow variables in non-linear system (example 2.2)

	$v_o(f_{1,1}\|f_{1,2})$	$v_o(f_{2,1}\|f_{2,2})$	$v_o(f_{3,1}\|f_{3,2})$
BIAS	-0.5916	-0.4638	-0.5210
$v_{i,1}$	-0.2772	-0.3210	-0.3081
$v_{i,2}$	-0.3396	-0.1790	-0.3701
$v_{i,3}$	-0.3855	-0.2423	-0.3322

FIGURE 2.1 GENERALIZED PROCESS CIRCUIT (EXAMPLE 2.1)

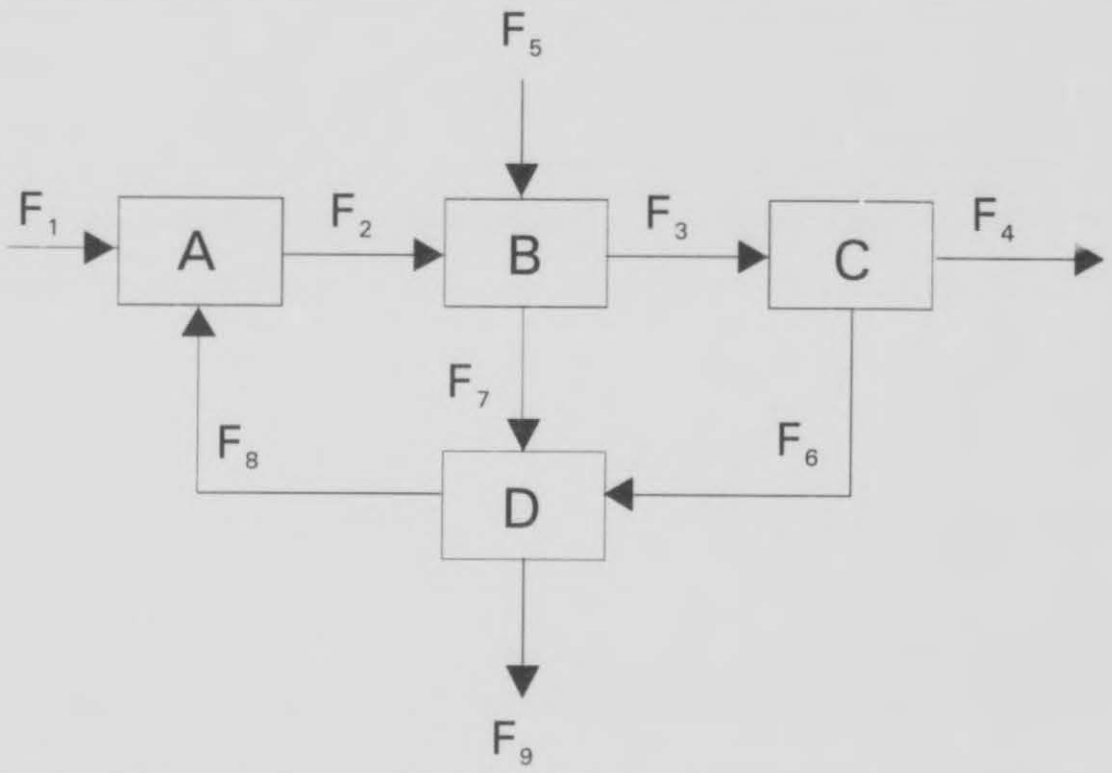


FIGURE 2.2 STRUCTURE OF NEURAL NET USED TO ESTIMATE VARIANCES IN PROCESS CIRCUITS IN EXAMPLES 2.1 & 2.2

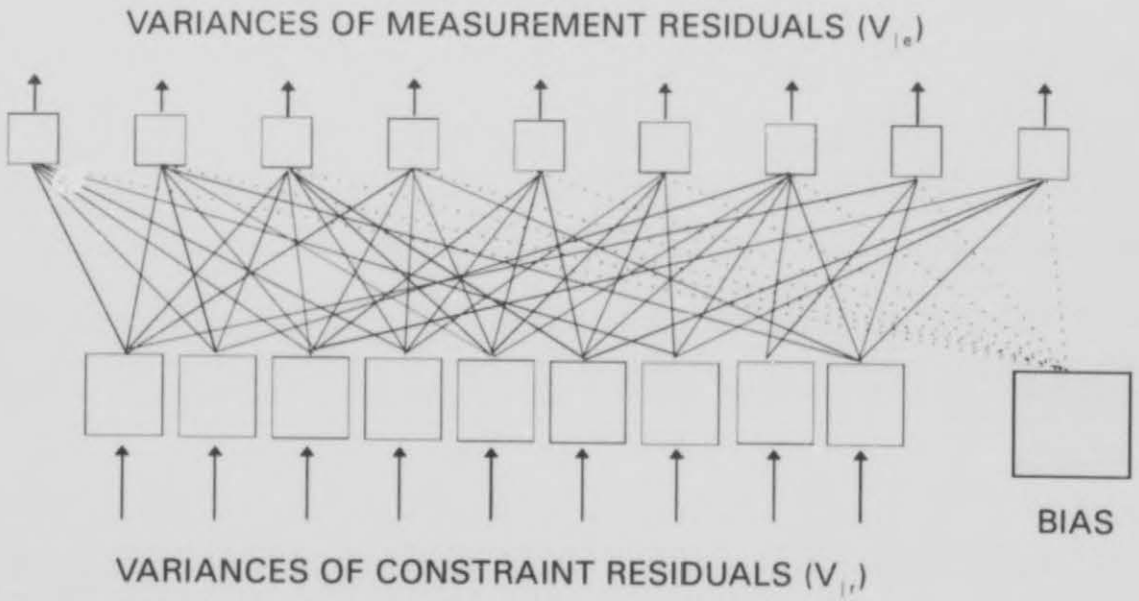


FIGURE 2.3 NEURAL NET REPRESENTATION OF RELATION BETWEEN CONSTRAINT AND MEASUREMENT VARIANCES IN EXAMPLE 2.1

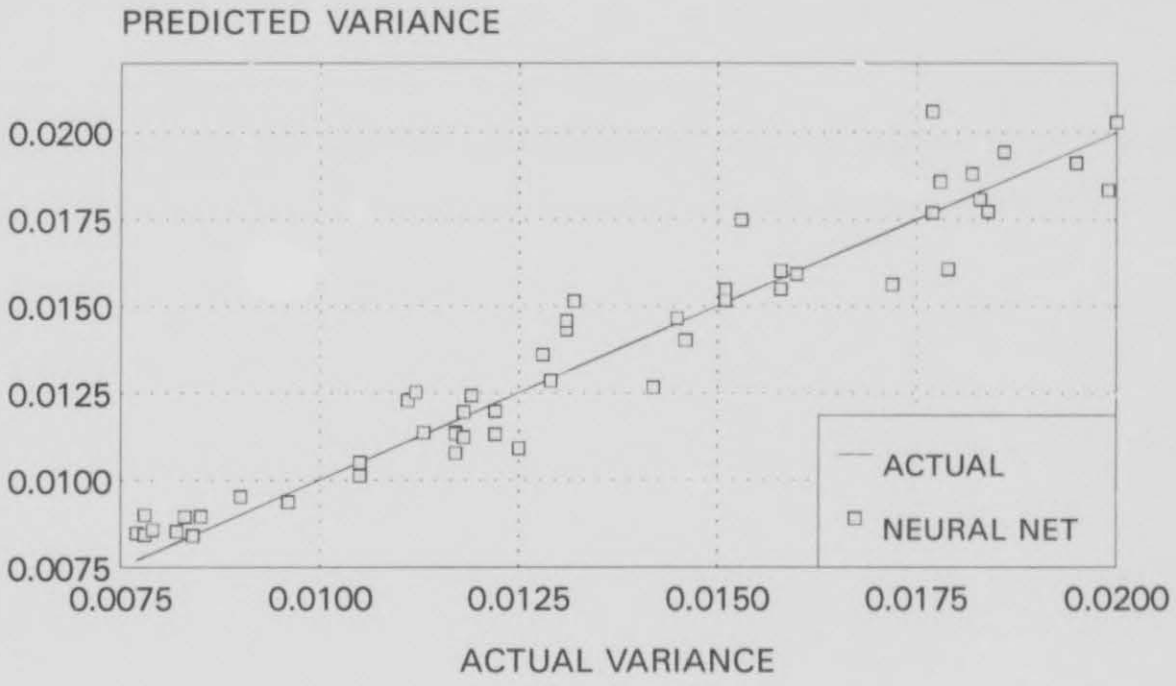


FIGURE 2.4 HIGH TENSION ROLL SEPARATOR USED IN EXAMPLE 2.2

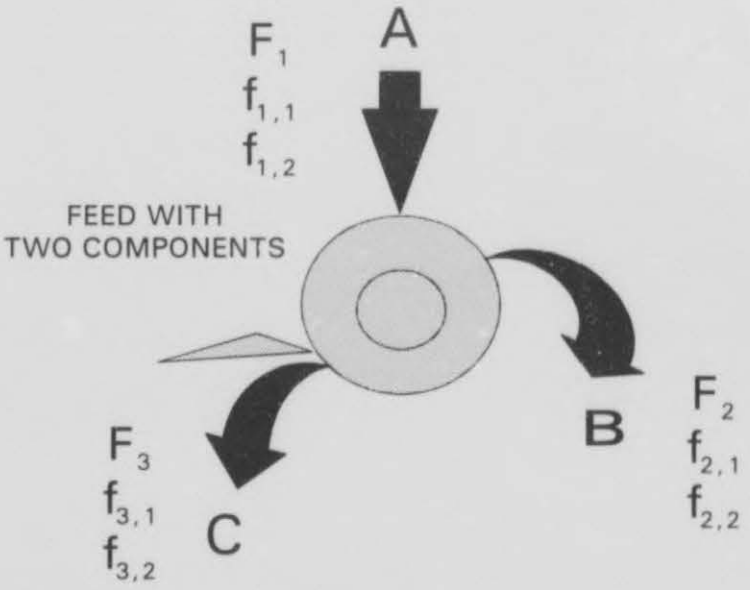
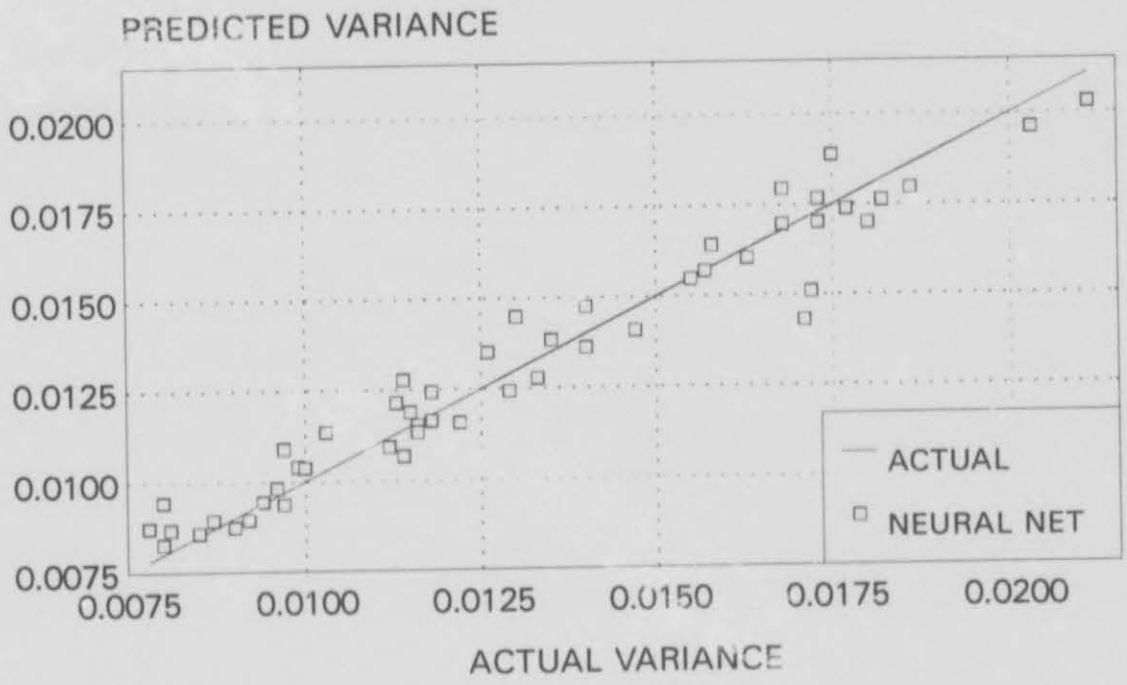


FIGURE 2.5 NEURAL NET REPRESENTATION OF RELATION BETWEEN CONSTRAINT AND MEASUREMENT VARIANCES IN EXAMPLE 2.2



CHAPTER 3

The detection and isolation of systematic errors in steady state systems

Summary

The monitoring of plants and the verification of process models depend crucially on reliable sets of steady state component and total flow rate data. These measurement data are generally subject to random noise (and possibly systematic errors) and typically violate the process constraints of the system. It is consequently necessary to adjust the data, and also to account for systematic or gross errors in the data prior to this reconciliation procedure, or as part of it, in order to avoid severe impairment of the adjustment process. This can be accomplished by using a neural net to form an internal representation of the relationship between the residuals of the measurements or the process constraints, and the error categories represented by these residuals. When presented with other residuals generated by the process model, the trained net can then classify the residuals to the categories it had previously been trained to recognize. The major advantage of using neural nets instead of conventional statistical methods is that neural nets can be used more effectively for the detection of systematic errors in process systems subject to non-linear process constraints (a situation common in the chemical and mineral processing industry), as well as for errors with arbitrary or ill-defined distributions.

3.1 OBJECTIVES OF THIS CHAPTER

Despite extensive research over several decades, no method is as yet available to satisfactorily identify systematic errors in process systems subject to an arbitrary set of constraints. The reason for this is the ill-definedness of the distributions of the residuals of non-linear process constraints, even when the distributions of the variable measurement residuals are known, as well as the difficulty of generating measurement residuals that can be analyzed with standard statistical methods. The main objectives of this chapter are consequently to explore the powerful pattern recognition capabilities of neural networks to

- model the relationship between the residuals of the process variables and constraints and the statistical parameters of these residuals for typical chemical and metallurgical engineering processes, and to

- use these residual distribution models to detect gross or systematic errors in the abovementioned systems.

3.2 BACKGROUND THEORY

The acquisition of reliable plant data is fundamental to a clear understanding of the operational behaviour of a plant, the modelling and optimization of process circuits, as well as the identification of other phenomena peculiar to the process (Verneuil et al., 1992). These data are generally subject to random noise, or even gross errors, owing to inadequate instrumentation, failure or miscalibration of measuring instruments, the departure of the process from steady state due to malfunctioning process equipment, or significant changes in the environment (Hlavacek, 1977). Typical process data will consequently violate mass and energy conservation requirements, as well as other physical constraints pertaining to the process and will have to be adjusted in order to satisfy these constraints. Under these circumstances it is essential that gross errors are detected and accounted for prior to, or during reconciliation of the data, since failure to do so could result in a severely distorted picture of the process. (Hodouin & Vaz Coelho, 1987). Since repeated measurement of a variable is not an effective means for the detection of a systematic error, virtually all gross error detection schemes involve statistical tests based on the characteristics of the constraint residuals of the measurement errors. Unfortunately these tests are generally only useful as far as systems subject to linear constraints are concerned (Serth et al., 1987; Tjoa & Biegler, 1991). In the chemical and mineral processing industries this is a major drawback, since most process systems in these industries are non-linear. In this dissertation a new method is consequently proposed for the detection of systematic errors in constrained measurement data. This method is based on the powerful capability of neural nets to classify measurement errors, and is not hindered by the nature of the system constraints.

Although a systematic error can indicate an error that differs from a random error only with regard to its expected value, while a gross error can indicate an error that differs from a random error with regard to its distribution function in general, the use of these terms in the literature is not consistent. For the purpose of this dissertation the terms systematic errors and gross errors are

thus used interchangeably to denote errors that do not have the same distributions or distribution parameters as random measurement errors.

3.2.1 Problem statement

To detect and isolate systematic errors in a measured set of constrained variables, two conditions have to be met. First is a knowledge of the distributions of the measurement residuals and second is the existence of analytical redundancy (where measured variables are overdetermined).

The process constraints of a typical process system are described by Crowe (1989),

$$\mathbf{C}.\mathbf{x} = \mathbf{0} \tag{3.23}$$

where **C** is an (m x n) constraint matrix of full row rank m (n > m) and **x** is the (n x 1) vector of true values of the state variables. If

$$\mathbf{x}' = \mathbf{x} + \mathbf{e} \tag{3.24}$$

where **x'** constitutes the (n x 1) vector of measurements of the true values **x**, with an (n x 1) error vector **e**, and covariance matrix $\mathbf{V}_{|x}$, then the measured values of the process variables generally violate the process constraints

$$\mathbf{C}.\mathbf{x}' = \mathbf{r} \neq \mathbf{0}$$

or in terms of the true values and error components

$$\mathbf{C}.(\mathbf{x} + \mathbf{e}) = \mathbf{r} \tag{3.25}$$

and assuming the constraints to be linear

$$\mathbf{C}.(\mathbf{x} + \mathbf{e}) = \mathbf{C}.\mathbf{x} + \mathbf{C}.\mathbf{e} = \mathbf{r}, \text{ i.e.}$$

$$\mathbf{C}.\mathbf{e} = \mathbf{r} \tag{3.26}$$

If it is assumed that the error vector **e** has a Gaussian distribution and that no systematic errors are present (the null hypothesis), **r** is a multivariate normal with a zero mean (Madron et al., 1977; Romagnoli & Stephanopoulos, 1981; Mah & Tamhane, 1982; Tamhane & Mah, 1985), i.e.

$$E(\mathbf{r}) = E(\mathbf{C}.\mathbf{e}) = \mathbf{C}.E(\mathbf{e}) = \mathbf{0} \qquad (3.27)$$

Otherwise the expected value of **r** is not zero (the alternative hypothesis), i.e.

$$E(\mathbf{r}) = \mathbf{b} \neq \mathbf{0} \qquad (3.28)$$

which indicates the presence of a systematic error with a bias of magnitude **b**. By making use of standard statistical criteria, or variants of these statistics, the two hypotheses can subsequently be evaluated and rejected or accepted, and the presence of gross errors be determined.

Alternatively the measurement residuals **e** can be tested directly, making the need for additional procedures to isolate errors unnecessary. Since the true value of a state variable is generally unknown, the adjustments **x'-x** are evaluated after reconciliation of the data. If use is made of a least squares method for data reconciliation, the process constraints have to be linear in order to ensure a known distribution for the measurement residual **x'-x**, assuming a known distribution for **x'** (Madron, 1985).

3.2.2 Type I and type II errors

When statistical hypotheses of populations are tested, two types of errors (referred to in the statistical literature as type I or type II errors) are possible (Walpole & Myers, 1978). A type I error is committed when the null hypothesis is valid, but erroneously rejected (i.e when a random error is incorrectly identified as a systematic error), and a type II error is committed when the null hypothesis is accepted when it is false (i.e. when a systematic error is not detected). The probability of committing a type I error is known as the level of significance or the size of the critical region of the test, and is usually denoted by α, while the probability of a type II error being committed is usually denoted by ß. In efficient measurement error detection schemes, the probability of both these errors occurring should be as small as possible.

Similar to statistical tests, the performance of a neural net error detection scheme can also be constructed to minimize the probability of the occurrence of type I or type II errors, through the appropriate labelling of training exemplars. To reduce the occurrence of type I errors, only the residuals in region CD in figure 3.1 are designated as gross errors. When the distributions of random and systematic errors overlap (shown schematically in figure 3.1),

one critical region can only be adjusted at the expense of another. Stated differently, to ensure that the probability of type II errors occurring is as low as possible, the probability of type I errors occurring has to be increased, as is shown diagrammatically in figure 3.1. In the neural net error detection schemes discussed in this chapter, all the nets were trained to significance levels of less than 2%.

3.2.3 Existing methods

Generally speaking, approaches to fault detection and isolation can be divided into two major groups, depending on whether or not the methods are based on the use of a plant model. Methods not based on the use of a model include limit checking, the use of special sensors or multiple sensors (physical redundancy), frequency analysis, as well as the use of knowledge-based methods where rules derived from theory or experience are used to detect the presence of system failures or gross errors (Gertler, 1988). Methods in which mathematical models are used to detect gross errors emanate from the concept of analytical redundancy, and has received considerable attention during the last two or three decades.

Early methods making use of mathematical models were based on iterative data adjustment procedures, whereby measurements were successively deleted from the measurement data set. Gross errors could subsequently be identified through association with the maximum effect of such a deletion on a least squares objective function, but the method was cumbersome, especially when applied to large sets of measurements (Romagnoli & Stephanopoulos, 1981). These methods for detecting the presence of systematic errors were later validated statistically, based on the relation between the residuals of the constraints and the measurement errors. Further advances followed with the proposal of univariate and multivariate statistical criteria for detection not only of the presence of gross errors in the data set as a whole, but also of the locations of these errors (Romagnoli & Stephanopoulos, 1981; Crowe et al., 1983; Knepper & Gorman, 1980; Madron, et al., 1977). These methods were only applicable to measurement data subject to linear constraints and non-linear constraints had to be linearized, typically by retaining first order terms in a Taylor series expansion (Crowe et al., 1986; Romagnoli & Stephanopoulos, 1980, 1981; Stephenson & Shewchuck, 1986). An alternative method of studentized

residuals not dependent on knowledge of the variances of the error measurements and purported to discriminate more accurately against outliers was similarly proposed by Jongelen et al. (1988).

Ragot et al. (1991) explored the application of parity space techniques for the detection of gross errors in analytically redundant process data and demonstrated the equivalence of this method to methods based on the use of normalized residuals.

Although the principle on which all these tests was based remained essentially the same, many refinements to these tests were proposed in subsequent years. Serth and Heenan (1986) for example, proposed a screened combinatorial test, as well as a modified iterative measurement test which they applied to measurement data subject to bilinear constraints. Iordache et al. (1985) similarly proposed a modified test for the identification of multiple gross errors. Narasimhan and Mah (1987, 1989) recommended the use of a generalized likelihood ratio test, which could accommodate errors not only attributable to erroneous measurements, but also to actual deviations in the process itself, while Rollins and Davies (1992) suggested the use of an unbiased method to detect systematic errors. The sophistication of current statistical procedures not withstanding, these methods all suffer from a serious shortcoming. They are all inherently limited in their applicability to data restricted by linear process constraints. That is not to say these techniques can not be applied to data subject to non-linear process constraints at all. Serth & Heenan (1986) and others have proved that under certain circumstances (such as where the process constraints can be linearized successfully) the application of these statistical methods yields reasonably satisfactory results (Crowe et al., 1986; Romagnoli & Stephanopoulos, 1980, 1981).

In a new approach Kramer (1992) has recently shown that autoassociative neural networks can be implemented to detect and eliminate gross errors in measurement data subject to non-linear constraints. The disadvantage of these nets is that they depend on a large degree of redundancy in the measurement data, and are therefore not suitable for the detection and elimination of gross errors in singular variable measurements, or measurements characterized by small sample sizes, such as those frequently encountered in the metallurgical industry, where the independent measurement of process variables is often

difficult and expensive. Error classification by autoassociative neural nets also depends on the relative distribution of errors in the samples. If for example two out of three variable measurements contain biased or gross errors, the autoassociative net will incorrectly characterize the unbiased error as biased, since it does not have an absolute reference regarding the features of a gross error. The methods discussed in this chapter differ from those proposed by Kramer (1992) in that they are based on the conservation equations or other process constraints imposed on the measurement data. Thus unlike Kramer's approach they depend on a mathematical model and do not need large sets of measurements to detect or isolate gross errors (a distinct advantage especially as far as mineral processing or metallurgical systems are concerned).

It is consequently the aim of this investigation to highlight the use of neural net methods to detect gross errors in measurement data. These nets make use of the constraint and measurement residuals of the process system, and like autoassociative neural nets (but contrary to classical statistical methods), they also have a powerful ability to detect gross errors in the presence of non-linear constraints.

3.3 THE DETECTION OF SYSTEMATIC ERRORS BY MEANS OF NEURAL NETS

By presenting a feedforward neural net with examples of process measurement and constraint residuals as input, and appropriate classes indicating the presence of different types of errors as output, the net can be trained to generalize the relationship between residuals and the types of errors giving rise to these residuals. When presented with test vectors consisting of constraint and/or measurement residuals, it is then able to assign the input to the error categories it had been trained to recognize (analogous to the statistical hypothesis tests traditionally used to categorize errors). Since neural nets are not limited by the nature of the process constraints (unlike many statistical methods), they can be used to considerable advantage in different error detection schemes. Two such strategies are outlined in this chapter.

3.3.1 Global detection of gross errors in sets of constrained variables

The first strategy is the simplest and can be used to detect gross errors in sets of variables associated with nodes in the process circuit, similar to the global

test in statistics (Romagnoli & Stephanopoulos, 1980). Note that this strategy as such does not allow the location of errors beyond the sets of variables associated with nodes in the process circuits. Despite this drawback the strategy is useful in that no information regarding the true values of the process variables is required. The method is based on the effect that measurement residuals have on the process constraints of the system. The measurement and constraint residuals are directly proportional, i.e. zero-valued measurement residual vectors (associated with the true or reconciled values of the process variables) generate zero-valued constraint residual vectors, while a monotonous increase in the measurement residuals also results in a corresponding monotonous increase in the constraint residuals. Systematic errors (which are usually significantly larger than random errors) generally result in constraint residuals that are larger than normal, and which can be distinguished from smaller constraint residuals which are usually associated with smaller random errors.

In traditional statistical test methods (assuming that the distribution functions of the variable measurements are known) the detection of gross errors is limited to linear or linearized process constraints which have essentially the same types of distribution functions as the process variables. When the constraints are non-linear their distributions are generally unknown and the constraint residuals can consequently not be subjected to statistical hypothesis testing. By making use of neural nets, this restriction is obviated, since the net can learn arbitrary distributions of the constraint residuals *a priori*, as is explained below.

The general detection strategy involves training a neural net with examples of constraint residuals generated by measurement residuals of a known class. No mathematical models or explicit parameter specifications are involved in the process - the data used to train the net are the standard from which the net learns the distinction between residuals considered to be normal and those considered to be indicative of a bias in the process data. Plant data can be used for training the net, but artificial data are also convenient, since they are easy to generate and there is no uncertainty as far as the classes to which the residuals belong are concerned. During the training process, the net constructs an internal model of the relationship between the constraint residuals and the classes associated with the germane measurement residuals. This model can

subsequently be used to detect gross errors in measurements not encountered previously.

As far as the strategies proposed in this dissertation are concerned, only two classes are recognized, viz. a class associated with random errors and a class associated with one or more gross errors in the observations. Extension of the number of classes (e.g. to discern between gross errors of various types) is a trivial matter that does not merit in-depth discussion. Once trained, the net can be presented with any set of constraint residuals generated by actual plant data, and it will assign these residuals to the classes it had been trained to recognize. The use of this strategy is clarified by way of the following example.

Example 3.1. Detection of gross errors in a two-product classifier.

Consider a two-product classifier such as a hydrocyclone where a feed stream (F_1) is split into two product streams, e.g. an overflow (F_2) and an underflow stream (F_3). If only two components are present, the mass balance equations can be expressed as

$$F_1 - F_2 - F_3 = 0 \tag{3.29}$$

$$F_1 f_{1,1} - F_2 f_{1,2} - F_3 f_{1,3} = 0 \tag{3.30}$$

$$F_1 f_{2,1} - F_2 f_{2,2} - F_3 f_{2,3} = 0 \tag{3.31}$$

where $f_{i,j}$ denotes the fraction of the i'th component in flow stream F_j.

For the neural net to be able to detect systematic errors in the variables F_1, F_2, F_3 and $f_{1,1}$, $f_{1,2}$, $f_{1,3}$, $f_{2,1}$, $f_{2,2}$ and $f_{2,3}$, it has to be presented with examples of what are considered to be such errors. As with any other procedure, this implies a knowledge of the distributions of the residuals of the variable measurements. For the purpose of this example, normal probability distributions were assumed and embodied in the training data.

i) Training and test data

Training and test sets were generated from a set of true values (i.e. an arbitrary set of values that satisfies the process constraints represented by equations

3.29 to 3.31). These training and test sets were derived by corrupting the sets of consistent data with random and gross errors of the form

$$\mathbf{x}' = \mathbf{x} + \mathbf{e} + \mathbf{b} \qquad (3.32)$$

where $\mathbf{x}'$ is the observed value of the process variable with a true value of $\mathbf{x}$, $\mathbf{e}$ is a random measurement residual with a normal distribution with a zero mean and a known variance, and $\mathbf{b}$ is a bias or systematic error component which is per definition zero in random errors and non-zero in gross or systematic errors. The magnitude of the bias was allowed to vary randomly between 10% and 100% of the random variable measurement x_i', i.e. $0.1*(x_i + e_i) \leq |b_i| \leq x_i + e_i$ for gross errors, with a small standard deviation of 2.5%. The training set consisted of 200 feature vectors of the form $\{|r_p|/\sigma_p;CLASS_p\}$,

where r_p denotes the p'th constraint residual of the circuit, σ_p the standard deviation of the constraint residuals r_p and the binary output $CLASS_p$ the type of error (0 for random and 1 for gross) associated with the p'th constraint residual. By considering the normalized magnitudes of the residuals, the relationship between the error classes and the constraint residuals is simplified, since the net does not need to account for the sign of the residuals. In more sophisticated error detection schemes (requiring more sophisticated neural net models) the actual values instead of the magnitudes of the residuals can be used in order to distinguish between different types of gross errors. Gross and random errors were present in approximately equal proportions in the training sets, to allow the net to construct representative models of each class. The test set was comprized of 100 feature vectors similar to those in the training sets.

ii) Structure of neural nets

Four different neural nets which consisted of simple one-layer configurations (not counting the input layer which serves only to distribute the inputs to the rest of the net) were trained for error detection. The nets were structured as showed in figure 3.2, and differed only with respect to the process units used in each net. Four of the most popular types of units described in the literature (Lippmann, 1987; Wasserman, 1990) were investigated, namely linear, sine, hyperbolic tangent and sigmoidal process units.

iii) Training of the nets

The nets were trained by means of the generalized delta rule (Rumelhart et al., 1986) and typically required less than 20 000 iterations to converge. After convergence several runs on different test sets were made with each net. The weights of the trained nets are shown in tables 3.1-3.4.

iv) Results

The ability of the trained nets to classify measurement errors as random or systematic is summarized in table 3.5 and 3.6. As can be seen, the performance of the nets does not appear to be particularly influenced by the type of transfer function implemented. The nets could detect approximately 90% of the gross errors in the data they were trained on (as shown in table 3.5) and approximately 85% of the gross errors in data not encountered before (as shown in table 3.6). Judging from these small differences between the abilities of the nets to classify errors in the training and test data, it can be concluded that the nets generalize the training data well. Similar experiments were conducted with neural nets containing one or more hidden layers, but this did not result in any significant improvement in performance.

3.3.2 Location of gross errors in sets of constrained variables based on variable measurement residuals

The second strategy differs from the previous one, in that the measurement residuals instead of the constraint residuals are used to locate systematic errors. Traditional statistical methods make use of analogous approaches, by reconciling the variable measurements and testing the residuals generated by reconciliation (Madron, 1985). Since these test procedures can not be separated from the data reconciliation problem (which have to be solved first to generate a set of variable adjustments which can be tested for gross errors), they are affected by the ability of the reconciliation procedure to yield unbiased reconciled measurement values in the presence of gross errors. It is especially the least squares procedures that are sometimes vulnerable to this type of problem when the constraints of the process are non-linear (Madron, 1985) and as can be expected, the problem is aggravated by the presence of multiple gross errors.

A similar approach is possible with a neural net, where a set of reconciled process variables is also used as a basis for the location of systematic errors. The important distinction between the neural network approach and conventional statistical methods, is that the net can form an internal representation of the distributions of residuals and is consequently not restricted by the nature of, or the types of distribution functions associated with the process system. Like statistical methods, the neural net approach is also dependent on the generation of reconciled variable measurements and if the variable adjustments are compromised by the presence of gross errors in the data, the performance of the net can also be expected to deteriorate owing to the lower quality of the inputs.

The use of a neural net to locate systematic errors in process data entails the corruption of a consistent set of variable measurements (in effect assumed to be the true values of the process variables) with various types of errors, and training a neural net to classify these errors based on the measurement residuals $|x_i'-x_i|$ or $|x_i'-x_i''|$. The following examples illustrate the technique.

a) Example 3.2: Location of multiple gross errors in an industrial flotation circuit

The flotation circuit depicted in figure 3.3 has previously been described in the literature (Cutting, 1976) and consists of 12 process units, viz. 6 flotation banks (R_1, R_2 & C_1-C_4), 5 hydrocyclones and a mill. Since only the total flow rates of the process streams, F_1, F_2, .. F_{19} are considered, the effect of the mill can be ignored. The circuit is thus subject to 11 linear process constraints (equations 3.33-3.43) and since measurements of the flow variables generally violate these constraints, they have to be adjusted prior to further use. As part of the reconciliation procedure, it is necessary to detect and eliminate gross errors in the flow variable measurements, as the presence of these errors can lead to large distortions in the reconciled values of the variables. Knowledge of the variances of these measurements is furthermore a prerequisite to the detection of systematic errors, as it is used to differentiate between the different classes of errors. As was pointed out in chapter 2, these variances are often not available and can then be estimated by some of the methods described in the preceding chapter. In order not to complicate this

demonstration unduly however, arbitrary variances are assumed for the measurement errors.

Process constraints:

Flotation banks

$$F_1 - F_2 - F_3 = 0 \quad (3.33)$$

$$F_4 - F_5 - F_6 = 0 \quad (3.34)$$

$$F_9 - F_{10} - F_{11} = 0 \quad (3.35)$$

$$F_{12} - F_{13} - F_{14} = 0 \quad (3.36)$$

$$F_{15} - F_{16} - F_{17} = 0 \quad (3.37)$$

$$F_{16} - F_{18} - F_{19} = 0 \quad (3.38)$$

Hydrocyclones

$$F_3 + F_{11} - F_4 = 0 \quad (3.39)$$

$$F_6 - F_7 - F_8 = 0 \quad (3.40)$$

$$F_5 + F_{14} - F_9 = 0 \quad (3.41)$$

$$F_{10} + F_{17} - F_{12} = 0 \quad (3.42)$$

$$F_{13} + F_{18} - F_{15} = 0 \quad (3.43)$$

The adjusted data, shown in table 3.7, were used as a basis for generating artificial measurements, by corrupting the consistent set of measurements by various random and systematic errors. In this investigation all errors had a standard deviation of approximately 10%, so that random and systematic errors were differentiated sc'ely in terms of their expected values, as shown in figure 3.4 where a normal error is compared with a gross error with a 15% bias.

Since the constraints are linear, a traditional statistical method, such as the popular measurement test could also have been used to determine the presence of gross errors in the process variables (Iordache et al., 1985).

A single layer back propagation net (shown diagrammatically in figure 3.2) consisting of an input and output layer with 19 computational elements each (corresponding to the 19 process variables of the system) was constructed to identify gross errors in the measured values of the flow streams F_1, F_2, .. F_{19}. The states of the computational elements in the output layer of the net (one element for each measured variable) indicate the presence (output value = 1) or the absence (output value = 0) of a gross error. It is in principle also possible to distinguish between systematic errors with different biases or expected values, by expanding the domains of the states of these elements, or perhaps by assigning errors to three different categories, viz. random, gross and indeterminate.

The set of exemplars consists of feature vectors of the type $T_k = \{|F_i'-F_i|/\sigma_i;CLASS_i\}$, where i = 1,2,..19, i.e. the inputs consist of the normalized magnitudes of the measurement residuals of the flow variables in the system (F_i-F_i'), as well as an indication of the type of error associated with a particular measurement value ($CLASS_i$).

Approximately 50 artificially generated exemplars were needed to adequately train the neural net, as indicated in figure 3.5. After approximately 10 000 training cycles (presentations of each vector in the training set to the net) very little improvement in the root mean square error (difference between actual and the desired output of the net) occurred, as shown in figure 3.6. The performance of the net could consequently be evaluated against the test set of vectors and is depicted graphically in figure 3.7. The labels A-G shown in figure 3.7 denote the corruption of the measurement data with errors with different biases as explained in table 3.8. Biases are shown relative to the measurements. In figure 3.7 it can be seen that the net classified most gross errors correctly when the relative bias of the systematic error was larger than approximately 70%. These values are not absolute, since the performance of the net is also determined by the variance of the errors (approximately 0.013 in this case). For systematic errors with expected values not much different from those of the actual measurements themselves (less than 40%), the discriminatory power of the net dropped markedly, as could be expected.

b) Example 3.3: Location of multiple gross errors in the measurement data of a three-stage backfill circuit subject to non-linear process constraints

This example is an extension of example 3.1 discussed in section 3.3.1, and is based on a backfill circuit which consists of three hydrocyclones connected as indicated in figure 3.8 and which is used for the preparation of backfill material in a South African mine (Woollacott et al., 1992). Although the plant data both before and after reconciliation are provided in table 3.9, the measured data could only be tested for gross errors in an arbitrary way, since no knowledge of the covariance matrices of these measurements was available. As before the adjusted data were corrupted with known errors, which enabled accurate evaluation of the nets.

The material balance of the circuit is expressed by equations (3.44-3.53). These equations constitute the constraints on the process system, the residuals of which are incorporated in the training data set of the neural net.

Process constraints:

$$F_1 + F_5 - F_6 = 0 \tag{3.44}$$

$$F_6 + F_7 - F_9 = 0 \tag{3.45}$$

$$F_9 - F_2 - F_3 = 0 \tag{3.46}$$

$$F_3 - F_8 - F_7 = 0 \tag{3.47}$$

$$F_2 - F_4 - F_5 = 0 \tag{3.48}$$

and for i = 1 to 6

$$F_1.f_{1,i} + F_5.f_{5,i} - F_6.f_{6,i} = 0 \tag{3.49}$$

$$F_6.f_{6,i} + F_7.f_{7,i} - F_9.f_{9,i} = 0 \tag{3.50}$$

$$F_9.f_{9,i} - F_2.f_{2,i} - F_3.f_{3,i} = 0 \tag{3.51}$$

$$F_3.f_{3,i} - F_8.f_{8,i} - F_7.f_{7,i} = 0 \tag{3.52}$$

$$F_2.f_{2,i} - F_4.f_{4,i} - F_5.f_{5,i} = 0 \tag{3.53}$$

where

$$\sum_j \sum_i f_{i,j} = 1,\ i = 1,2,..9,\ j = 1,2,3$$

As in the previous example, the reconciled values of all the flow variables F_i and $f_{i,j}$ are used as a basis for the demonstration of gross error detection by means of neural nets. In order not to unduly complicate computational procedures, the reconciled data have once again been corrupted artificially with only two classes of errors (i.e. random and systematic).

i) Training and test data

In order to further evaluate the ability of neural nets to discriminate between error classes, eight different test and training data sets were generated from the reconciled data shown in table 3.9. All errors had standard deviations of 12% and biases as shown in table 3.8. As before, the training vectors T_i consisted of the normalized measurement residuals, as well as the classes associated with these residuals $\{|x_i'-x_i|/\sigma_i;CLASS_i\}$.

ii) Structure of neural nets

To determine the presence of systematic errors, a back propagation neural net with sigmoidal process units was used to categorize the two types of errors. The input layer of the net was comprized of 19 input units, corresponding to each of the measurement residuals of the circuit, while the output layer similarly consisted of 19 units (one for each normalized measurement residual), for assigning the residuals to the appropriate error classes.

iii) Results

The net converged in less than 10 000 iterations, typically as shown in figure 3.9, where the root mean square (RMS) value of the difference between the desired and the actual output of the net is shown during training. The ability of the net to isolate gross errors in the circuit is shown in figure 3.10. This figure shows a sharp decline in the ability of the net to detect gross errors as the difference between the expected values of the normal and the gross error becomes less than approximately 20%. Similar trends were observed for other variances.

c) *Example 3.4: Detection of gross errors in metallurgical grinding circuit (Serth et al., 1987)*

In the previous examples the ability of neural nets to detect gross errors in data subject to non-linear constraints has been demonstrated. In this example the performance of a neural net is compared to that of a statistical method described in the literature (Serth et al., 1987, 1989). The specific method has been selected because it deals explicitly with non-linear constraints, and is furthermore reported to be an efficient means of detecting multiple gross errors in measured data.

With the modified iterative measurement test (MIMT) technique process data are first reconciled (based on linear or linearized process constraints) and from these data a test statistic (z_j) is computed for each measured variable, i.e. $p_j = e_j/\mu_{j,j}^{½}$. Since the process constraints have been linearized, p_j is a standard normal deviate under the null hypothesis that z_j contains no systematic error. Each test statistic p_j is compared with a critical test value, $p_c = z_{1-ß/2}$, the 1-ß/2 point of the standard normal distribution. The variable corresponding to the largest value of $|p_j| > p_c$ is then deleted from the vector of measured variables and new reconciled measurement values are computed from the compressed measurement vector z_m. This step is followed by a limit check on the new reconciled measurements. If the limits are violated, the previously deleted variable is returned to the measurement set and the next largest value of $|p_j| > p_c$ is selected and the whole process repeated. If the upper and lower limits of the reconciled variables are not violated, the variable is not returned to the measurement set, since it is considered to contain a bias.

The circuit consisted of a ball and rod mill connected to a cyclone classifier as shown in figure 3.11. Based on the constraints that the mass fractions have to sum to unity, the mass fractions of water in streams F_5 to F_9 have been eliminated, and the following set of equations was obtained to describe mass flow in the circuit (Mular et al., 1976; Serth et al., 1987). The following set of equations is identical to that used by Serth et al. (1987) with $R_i = F_i$ and $W_{i,j} = f_{i,j}$.

Rod mill (Node 1)

$$F_1 - F_5 + F_5.\Sigma^3{}_i f_{5,j} = 0 \qquad (3.54)$$

$$F_4 - F_5.\Sigma^3_j f_{5,j} = 0 \quad (3.55)$$

Pump (Node 2)

$$F_2 + F_5 + F_6 - F_7 - F_5.\Sigma^3_j f_{5,j} - F_6.\Sigma^3_j f_{6,j}\, 0 + F_7.\Sigma^3_j f_{7,j} = 0 \quad (3.56)$$

$$F_5 f_{5,j} + F_6 f_{6,j} - F_7 f_{7,j} = 0, \ (j = 1,2,3) \quad (3.57)$$

Ball mill (Node 3)

$$F_3 + F_8 - F_6 + F_6.\Sigma^3_j f_{6,j} - F_8.\Sigma^3_j f_{8,j} = 0 \quad (3.58)$$

$$F_8.\Sigma^3_j f_{8,j} - F_6.\Sigma^3_j f_{6,j} = 0 \quad (3.59)$$

Cyclone (Node 4)

$$F_7 - F_8 - F_9 - F_7.\Sigma^3_j f_{7,j} + F_8.\Sigma^3_j f_{8,j} + F_9.\Sigma^3_j f_{9,j} = 0 \quad (3.60)$$

$$F_7 f_{7,j} - F_8.f_{8,j} - F_9.f_{9,j} = 0, \ (j = 1,2,3) \quad (3.61)$$

i) Training and test data

As before, training and test data of the form $\{|x_i'-x_i|/\sigma_i;\ CLASS_i\}$ were generated from a consistent set of process data (shown in table 3.10).

The errors introduced into the measurements were of the form $\mathbf{x} + \mathbf{e} + \mathbf{b}$, where $\mathbf{e}$ had a zero mean and a standard deviation of 2.5%, as investigated by Serth et al. (1987). Biases varied randomly between 10% and 100% of the values of the corresponding variable measurements, i.e. $0.1*(x_i + e_i) \leq |b_i| \leq x_i + e_i$, and were assigned randomly in equal proportions to a set of simulated measurements, so that approximately half of the measurement vectors contained random variables only, while the other half contained gross errors ranging from 10 to 100%. The proportions of the gross errors of various magnitudes were approximately equal as well, i.e. there were just as many gross errors with magnitudes ranging from 10-20%, as there were errors with magnitudes ranging from 20-30% or 60-70%, etc. These errors are depicted schematically in figure 3.11.

ii) Structure of neural nets

In order to detect the presence of gross errors in the variables associated with particular nodes in the process circuit, a neural net with 24 input elements and 24 hyperbolic tangent output elements (one for each of the 24 corresponding variables) was constructed and trained on the circuit as a whole.

iii) Training and testing

The nets were trained with the normalized cumulative delta rule (Rumelhart et al., 1986) and tested against the test data after convergence. Training proceeded rapidly and as before all the nets converged in less than 20 000 iterations.

iv) Results

The results of the tests are shown in tables 3.11 and 3.12. In table 3.11 the ability of the net to detect gross errors with different biases ranging in magnitude from 10-100% is shown and compared with the ability of the MIMT method. From these data it is clear that the net is more successful than the MIMT method. In table 3.12 the ability of the net to detect systematic errors in the various variables is shown. The net managed to detect virtually all gross errors in the variables, regardless of the magnitudes of the variables in the circuit. This is not so much due to the superiority of the net as an error classifier, but can probably be attributed to an inability in the MIMT method to yield unbiased adjustments to the variables prior to testing for gross errors. The neural net in contrast, was evaluated with unbiased measurement residuals, hence the better performance. Other statistical methods such as the one proposed by Tjoa and Biegler (1991) could be expected to detect systematic errors in the relatively smaller variables with a high degree of accuracy as well, as will be discussed in more detail in example 3.5 of this section.

d) Example 3.5: Detection of gross errors in an arbitrary non-linear system

This example has been used previously by Pai and Fisher (1988), Ramamurthi and Bequette (1990), as well as Tjoa and Biegler (1991), and comprizes five measured variables x_i' (i = 1, 2, .. 5) and three unmeasured variables x_i' (i = 6, 7, 8), subject to six non-linear constraints.

$$\tfrac{1}{2}(x_1)^2 - 0.7x_2 + x_3x_6 + (x_2)^2x_6x_7 + 2x_3(x_8)^2 - 255.8 = 0$$

$$x_1 - 2x_2 + 3x_1x_3 - 2x_2x_6 - x_2x_7x_8 + 111.2 = 0$$

$$x_3x_6 - x_1 + 3x_2 + x_1x_7 - x_3(x_8)^{\frac{1}{2}} - 33.57 = 0$$

$$x_4 - x_1 - (x_3)^2 + x_7 + 3x_8 = 0$$

$$x_5 - 2x_3x_7x_8 = 0$$

$$2x_1 + x_2x_3x_6 + x_7 - x_8 - 126.6 = 0 \qquad (3.62)$$

The exact values of these variables are $\mathbf{x} = \{4.5124, 5.5819, 1.9260, 1.4560, 4.8545, 11.070, 0.61647, 2.0504\}^T$ (Tjoa & Biegler, 1991). Tjoa and Biegler (1991) considered the reconciliation of these variables with a hybrid successive quadratic programming (SQP) method which was used to minimize an objective function based on a joint probability distribution of both random and gross errors. The performance of the algorithm is considered in more depth in the following chapter, when data reconciliation with neural nets is investigated. At convergence of the SQP procedure, they tested each measurement against the combined distribution. Since the presence of gross errors are taken into account during the minimization of the bivariate objective function, the variable adjustments are unbiased, which facilitates the isolation of gross errors considerably (regardless of the test method used to identify these errors). The method is still dependent on the explicit specification of a joint distribution model however, which may not be an accurate reflection of the process model.

In order to compare the error detection capability of a neural net with the method proposed by these authors, a 100 data sets were corrupted with 10% Gaussian noise to simulate random errors. In case 1 the 100-vector set is further corrupted with gross errors with a bias equal to four times the standard deviation (σ_i) of the random errors. The gross errors were distributed equally among the five measured variables (x_i') and in all 20% of the measurement vectors were corrupted (one gross error per measurement vector only). In case 2 every fifth variable set was completely corrupted with gross errors (i.e. five gross errors per variable set) and in case 3 a gross error was placed in each data set for measurements x_1' to x_5' in rotation. Since the success of the method is to a large extent ascribable to the ability of the reconciliation algorithm to generate unbiased estimates of the true values of the process

variables, an exact comparison of the method with a neural net is not possible unless the same reconciliation procedure is used in both cases. Nonetheless if the adjustments or residuals arising from the reconciliation of the measurements by the SQP method are considered to be unbiased as concluded by Tjoa and Biegler (1991), a reasonable comparison can be made by evaluating the response of the neural net to artificially generated unbiased residuals.

The neural net consisted of an input layer with five process elements (corresponding to the five measured values $\mathbf{x}'$ only, as it was not necessary to take the unmeasured variables into account) and an output layer with five sigmoidal process elements (one again for each variable x_i').

As with previous examples, training proceeded with the use of the normalized cumulative delta rule and after convergence of the net after approximately 20 000 iterations, the net was used to detect errors in test data sets 1, 2 and 3. The method proposed by Tjoa and Biegler detected approximately 73% of the gross errors in case 1, 60% of the gross errors in case 2 and 69% of the gross errors in case 3. The neural net detected approximately 72%-74% of the gross errors in all cases. These results should only be regarded as an indication of the capability of a neural net however, since especially in cases 2 and 3 the method used by Tjoa and Biegler might have had to contend with some bias in the measurement residuals prior to evaluation, not taken into account when testing the net. The weights of the trained net are shown in table 3.13.

3.3.3 Location of gross errors in sets of constrained variables based on measurement and constraint residuals

In somewhat more sophisticated approaches, error detection and isolation can be based on both measurement and constraint residuals, by training the neural net with vectors of the form $\{|x_i'-x_i|/\sigma_i, |r_p|/\sigma_p; CLASS_i\}$, similar to an approach discussed by Aldrich and Van Deventer (1993), which was based on the measurements and the constraint residuals as such $\{x_i', r_p; CLASS_i\}$. These techniques would have to be complemented by supervisory routines which would be able to interpret ambiguous output (a constraint residual might indicate the presence of a gross error, while the relevant measurement residuals appear to be unbiased or vice versa). In critical systems for example, only data unambiguously classified as unbiased could be considered as such.

3.4 DISCUSSION OF RESULTS AND CONCLUSIONS

The use of neural nets for the detection of gross errors in process data is best explained in analogy with statistical tests presently used for the same purpose. These tests are based either on the constraint residuals generated by the inconsistent data, or directly on the measurement residuals generated when the observed values of the variables are reconciled with the constraints.

The neural net techniques proposed in this chapter follow much the same approach and the prerequisites to the application of these methods are the same as for statistical tests - analytical redundancy of the process data and knowledge of the distribution of random errors in the data. An important point is the fact that the knowledge concerning the random errors does not need to be explicit when neural nets are used (i.e. the usual assumption of a normal distribution with a certain variance and a zero mean), since the net can derive its own representation of this distribution when provided with sufficient suitable process data. Once the net has constructed a model of the random errors in the measurements (from examples represented to it in the training phase), it uses this representation as the exclusive standard against which errors are classified.

A robust method to detect errors in constrained sets of measurements is based on the constraint residuals of the conservation equations or other constraints of the process. When statistical methods are used, the distribution of the constraint residuals has to be known and since this distribution is a function of the distributions of the individual variable measurements, statistical methods are generally limited to linear systems. (In non-linear systems the distribution of the constraint residuals is no longer the same as that of the measurement residuals and generally unknown). In neural nets this limitation does not apply, since the net learns the distribution of the constraint residuals for the particular system prior to classification of these residuals. Having learned this relationship for a particular set of constraint equations and a random error distribution, the net can be used to detect the presence of gross errors in the set of variable measurements. Once detected, other techniques can then be used to isolate these systematic errors.

Alternatively (and in order to isolate gross errors directly) tests can be based on measurement residuals. In contrast with the constraint residuals, these measurement residuals can unfortunately not be determined directly from the

variable measurements themselves, since the true values of the variables at the time of measurement are not known (assuming the process model to be correct, the true values of the constraint residuals at the time of measurement are known to zero). As a result it is not possible to make use of the actual measurement residuals (the differences between the true and the observed values of the variables) to locate gross errors in the system. Use can be made of the estimated measurement residuals however (the differences between the reconciled and the observed values of the variables). As a consequence the detection of gross errors based on measurement residuals is dependent on the accuracy with which the true values of the variables can be estimated, i.e. the reconciliation procedure used to filter the data.

The relation between the measurement residuals is typically relatively simple and training of the net uncomplicated, so that accurate classification of the errors is possible provided that the residuals presented to the net during training and testing are reasonably close estimates of the actual residuals of the measurements.

The examples considered in this dissertation were intended to demonstrate the use of back propagation neural nets to detect gross errors in measurement data subject to process constraints. In practice more sophisticated training procedures could be adopted, which could be used in conjunction with more sophisticated error classification schemes. It would also be desirable to incorporate additional information in the net (such as equipment and instrument failure histories, previous knowledge about measurement covariances, etc.), either through direct modification of the architecture of the net, or by using the net in conjunction with a knowledge base or another neural system.

In the light of these comments the following can be concluded:

- neural nets constitute a powerful means of detecting gross errors in sets of constrained variables, regardless of the nature of the constraints;
- explicit knowledge of the distribution of random errors in the variables is not a prerequisite to the use of neural net methods to detect gross errors, since the nets can learn the distributions a priori;

- neural nets can be used to isolate gross errors in variable measurements, regardless of the constraints of the system
- neural nets are able to discriminate between different errors at least as well as standard statistical methods; and
- the nets used to detect gross errors are relatively simple (no hidden layers are required) and the performance of these nets is moreover not affected significantly by the type of transfer function used.

3.5 TABLES REFERRED TO IN CHAPTER 3

TABLE 3.1 Weights of trained hyperbolic tangent neural net used to detect gross errors in example 3.1.

	$v_{o,1}$	$v_{o,2}$	$v_{o,3}$
BIAS	7.2773	-0.9801	-0.5564
$v_{i,1}$	9.1197	-2.6379	-0.0008
$v_{i,2}$	-0.2414	2.7840	-0.1008
$v_{i,3}$	0.0637	1.3763	0.2227

TABLE 3.2 Weights of trained sigmoidal neural net used to detect gross errors in example 3.1.

	$v_{o,1}$	$v_{o,2}$	$v_{o,3}$
BIAS	-1.1830	-1.0019	-0.7812
$v_{i,1}$	2.8595	-0.6192	0.0164
$v_{i,2}$	0.4902	1.2586	-0.2684
$v_{i,3}$	-0.3364	0.1607	0.1536

TABLE 3.3 Weights of trained sinusoidal neural net used to detect gross errors in example 3.1.

	$v_{o,1}$	$v_{o,2}$	$v_{o,3}$
BIAS	0.9655	-0.5244	-0.5777
$v_{i,1}$	1.7577	-1.0830	0.0549
$v_{i,2}$	0.1078	1.4312	-0.0467
$v_{i,3}$	-0.0592	0.6478	0.2131

TABLE 3.4 Weights of trained linear neural net used to detect gross errors in example 3.1.

	$v_{o,1}$	$v_{o,2}$	$v_{o,3}$
BIAS	0.1936	-0.2874	-0.5502
$v_{i,1}$	0.9029	-0.2629	0.0577
$v_{i,2}$	0.0498	0.6560	-0.0311
$v_{i,3}$	-0.0594	0.2738	0.1853

TABLE 3.5 Detection (%) of gross errors (bias 10-100%, standard deviation 2.5%) in training data (example 3.1).

Process units:	SIG[1]	LIN[2]	TANH[3]	SIN[4]
Average (%) errors detected	89.33	89.31	90.83	90.00
Standard deviation (%)	5.44	5.45	6.47	4.97

[1]sigmoidal: $v(u) = 1/(1+e^{-u})$, [2]linear: $v(u) = k_1 + k_2.u$, [3]hyperbolic tangent: $v(u) = (e^{u}-e^{-u})/(e^{u}+e^{-u})$, [4]sine: $v(u) = \sin(u)$

TABLE 3.6 Detection (%) of gross errors (bias 10-100%, standard deviation 2.5%) in test data (example 3.1).

Process units:	SIG[1]	LIN[2]	TANH[3]	SIN[4]
Average (%) errors detected	88.00	82.33	86.56	85.67
Standard deviation (%)	7.79	1.80	7.21	6.45

[1]sigmoidal: $v(u) = 1/(1+e^{-u})$, [2]linear: $v(u) = k_1 + k_2.u$, [3]hyperbolic tangent: $v(u) = (e^{u}-e^{-u})/(e^{u}+e^{-u})$, [4]sine: $v(u) = \sin(u)$

TABLE 3.7 Adjusted values of flotation circuit flow streams

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
3.418	2.950	0.468	3.492	1.808	1.684	0.134	1.549	4.882	3.407

F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}
1.475	3.660	3.557	0.123	4.523	4.251	0.273	3.284	0.966

TABLE 3.8 Ratios of the expected values of the uncorrupted measurements ($\mathbf{x}'$) to those of corrupted measurements ($\mathbf{x}' + \mathbf{b}$)*

CASE	A	B	C	D	E	F	G	H
$E(\mathbf{x}')/E(\mathbf{x}' + \mathbf{b})$	1	1.225	1.25	1.3	1.4	1.5	1.7	2

*All measurements had standard deviations of approximately 12%

TABLE 3.9 Plant and adjusted data from backfill circuit

Cyclone	Size or % solids	F_9 Exp	F_9 Calc	F_2 Exp	F_2 Calc	F_3 Exp	F_3 Calc
I	150	5.4	5.6	15.9	16.5	1.0	1.4
	106	16.4	15.4	33.5	32.5	8.7	8.8
	75	14.8	13.9	16.4	17.1	12.7	12.7
	53	9.1	9.7	9.2	8.2	12.4	10.2
	38	4.0	5.9	3.8	3.9	6.2	6.7
	0	50.3	50.5	21.2	21.8	59.0	60.2
	solids	49.3	47.5	61.8	62.2	42.3	43.5

Cyclone	Size or % solids	F_2 Exp	F_2 Calc	F_4 Exp	F_4 Calc	F_5 Exp	F_5 Calc
II	150	15.9	16.5	18.6	17.2	16.5	15.9
	106	33.5	32.5	33.5	34.7	29.5	30.4
	75	16.4	17.1	15.2	15.2	19.1	18.8
	53	9.2	8.2	8.2	8.5	7.6	7.9
	38	3.8	3.9	2.5	2.3	5.9	5.5
	0	21.2	21.8	22.0	22.0	21.4	21.6
	solids	61.8	62.0	61.7	62.2	61.2	61.2

Cyclone	Size or % solids	F_3 Exp	F_3 Calc	F_7 Exp	F_7 Calc	F_8 Exp	F_8 Calc
III	150	1.0	1.4	12.4	12.0	0.5	0.0
	106	8.7	8.8	35.9	36.2	5.0	5.1
	75	12.7	12.7	17.0	17.1	11.4	12.2
	53	12.4	10.2	9.5	9.6	9.8	10.3
	38	6.2	6.7	3.7	3.5	10.1	7.1
	0	59.0	39.8	21.5	21.6	63.2	65.4
	solids	42.3	43.5	- 23.0	40.6	41.5	

TABLE 3.10 Consistent values of variables in metallurgical grinding circuit used for the generation of random and gross errors

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
99.7	320.0	25.0	257.9	357.6	1109.4	1787.0	1084.3	702.7

$f_{5,1}$	$f_{5,2}$	$f_{5,3}$	$f_{6,1}$	$f_{6,2}$	$f_{6,3}$	$f_{7,1}$	$f_{7,2}$	$f_{7,3}$
0.0679	0.3964	0.2570	0.0103	0.4622	0.2826	0.0200	0.3663	0.2268

$f_{8,1}$	$f_{8,2}$	$f_{8,3}$	$f_{9,1}$	$f_{9,2}$	$f_{9,3}$
0.0328	0.5566	0.1831	0.0002	0.0721	0.2947

TABLE 3.11 Gross errors detected (%) by neural net and MIMT method

BIAS(%)	MIMT[1]	BPNN[2]
<20	63	98.2
20-30	80	100
30-40	76	100
40-50	81	100
50-60	87	100
60-70	84	100
70-80	89	100
80-90	92	100
>90	78	100

[1]Modified iterative measurement test
[2]Back propagation neural net

TABLE 3.12 Comparison of neural net with MIMT method to detect gross errors in selected variables

VARIABLE NUMBER	DESCRIPTION	% OF GROSS ERRORS DETECTED	
		MIMT[1]	BPNN[2]
7	Largest flow	94	99+
20	Largest composition	93	99+
1	2nd smallest flow	80	99+
13	2nd smallest composition	64	99+
3	Smallest flow	8	99+
22	Smallest composition	0	99+

[1]Modified iterative measurement test
[2]Detection of errors based on distribution of measurement residuals (see section 3.3.2)

TABLE 3.13 Weights of trained neural net used in example 3.5

	$v_{o,1}$	$v_{o,2}$	$v_{o,3}$	$v_{o,4}$	$v_{o,5}$
BIAS	0.4997	0.8768	0.5418	0.8929	0.6777
$v_{i,1}$	3.1778	0.0780	-0.0365	-0.1163	-0.0924
$v_{i,2}$	0.0744	3.3703	-0.0785	0.2559	-0.2869
$v_{i,3}$	0.2166	0.5721	2.7970	0.0545	-0.1915
$v_{i,4}$	-0.6865	0.3314	-0.0248	4.7152	0.2778
$v_{i,5}$	-0.2422	-0.1645	0.3275	-0.0207	4.2026

FIGURE 3.1 PROBABILITY DISTRIBUTIONS OF RESIDUALS

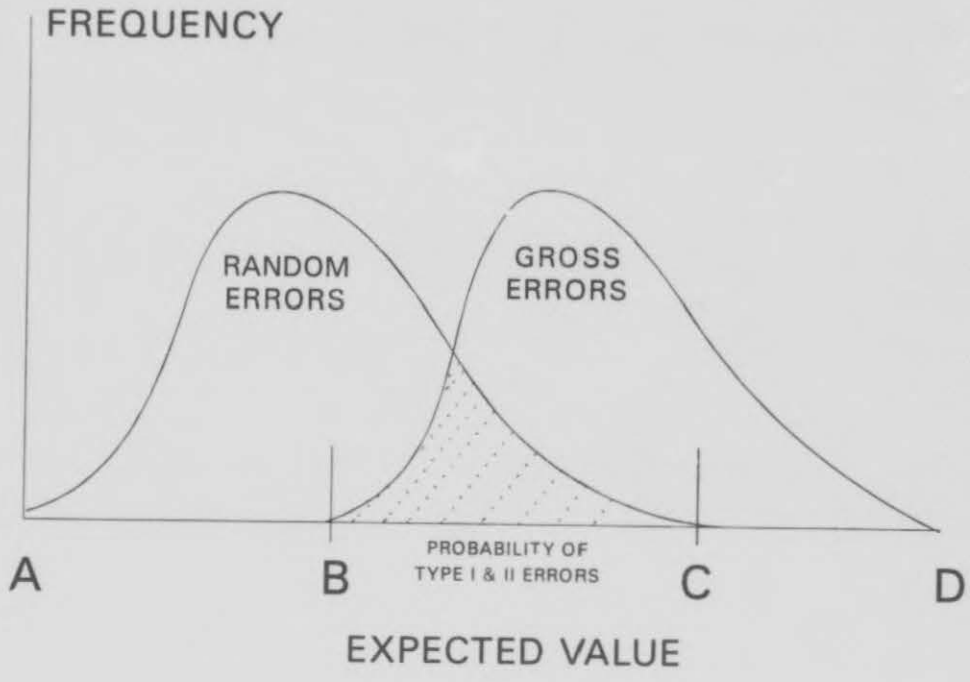


FIGURE 3.2 GENERAL STRUCTURE OF NEURAL NETS USED TO DETECT GROSS ERRORS IN PROCESS CIRCUITS

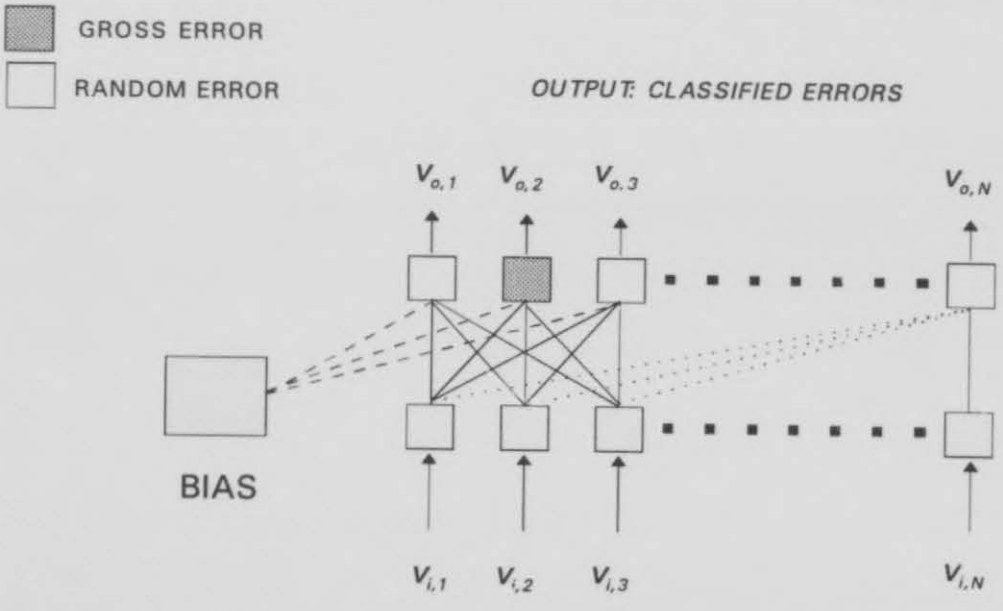


FIGURE 3.3 A FLOTATION CIRCUIT WITH LINEAR PROCESS CONSTRAINTS (Example 3.2)

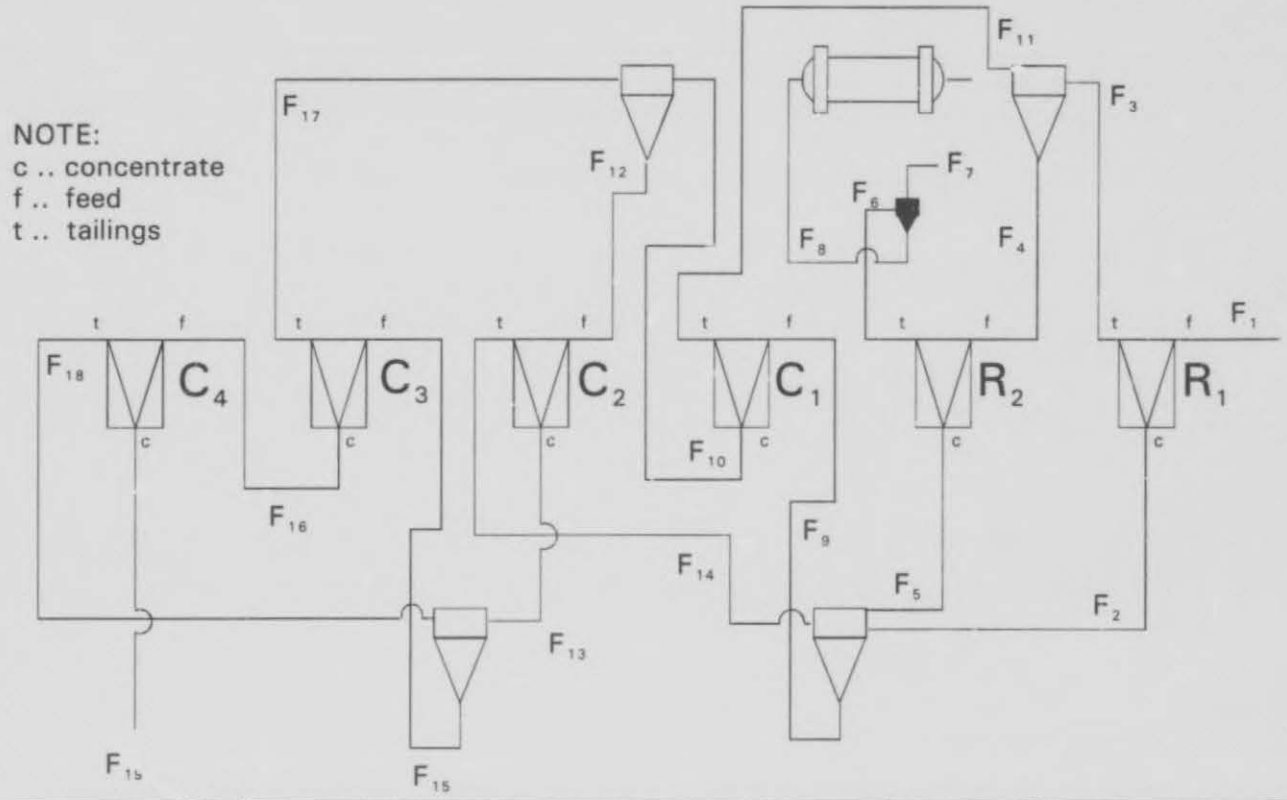


FIGURE 3.4 ARTIFICIAL ERRORS INTRODUCED INTO THE TRAINING AND TEST SETS OF EXEMPLARS

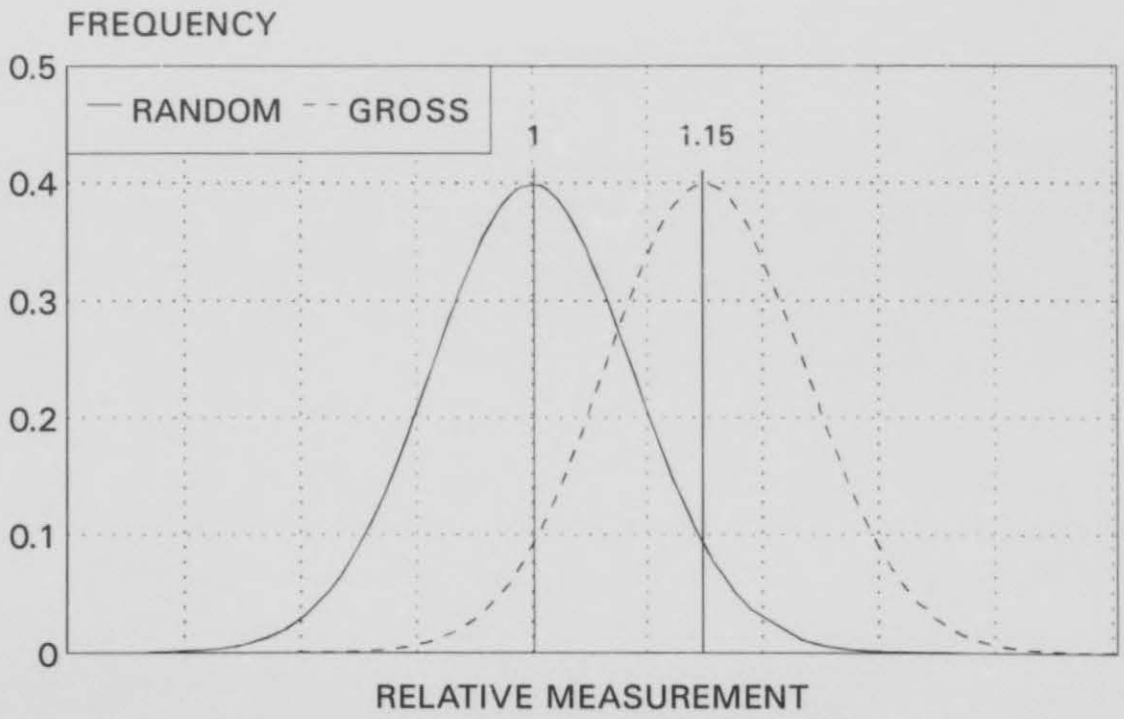


FIGURE 3.5 RELATION BETWEEN PERFORMANCE OF NET AND SIZE OF EXEMPLAR SET

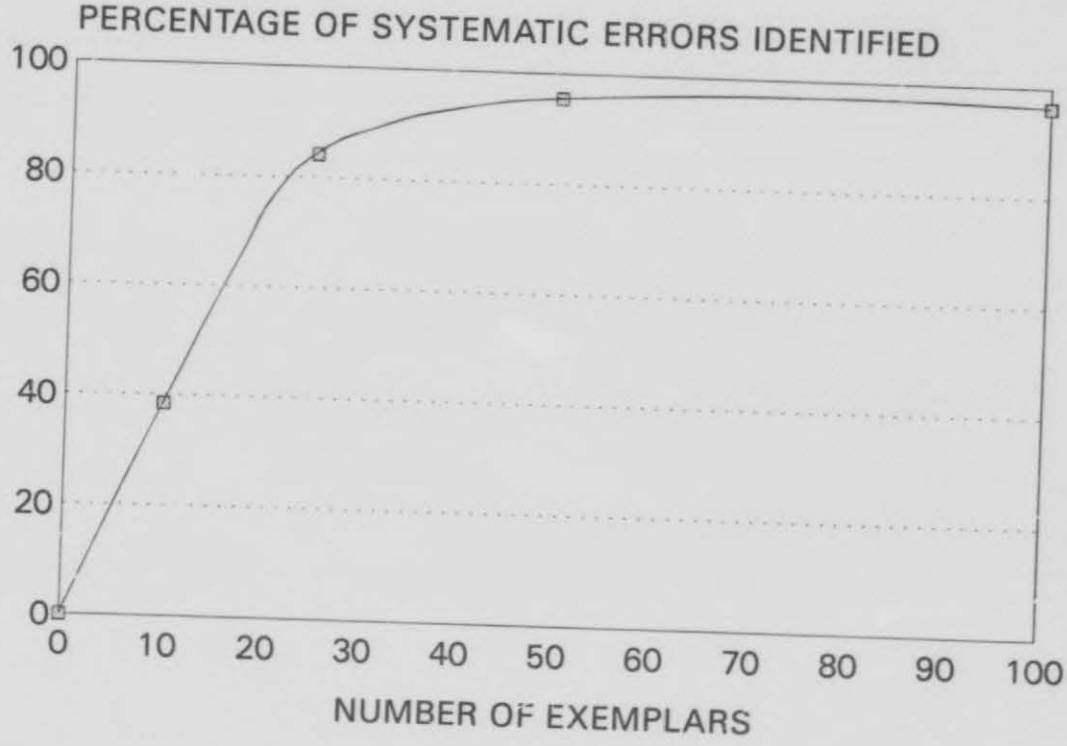


FIGURE 3.6 PERFORMANCE OF BACK PROPAGATION NEURAL NET DURING TRAINING

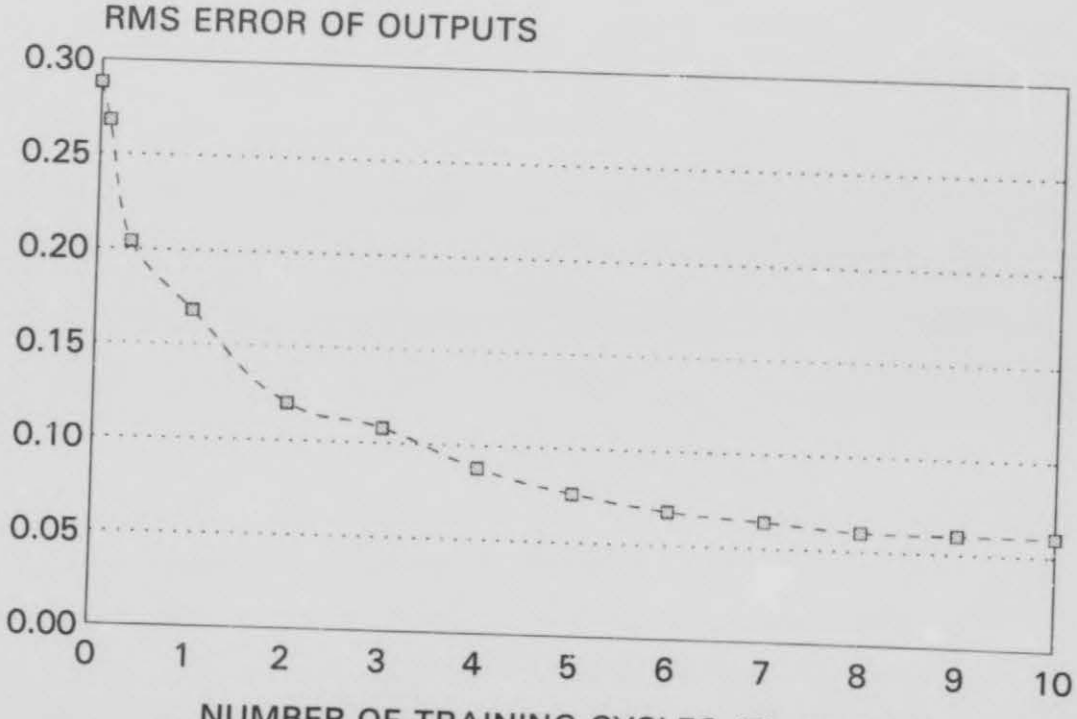


FIGURE 3.7 CLASSIFICATION OF SYSTEMATIC ERRORS IN MEASUREMENT DATA

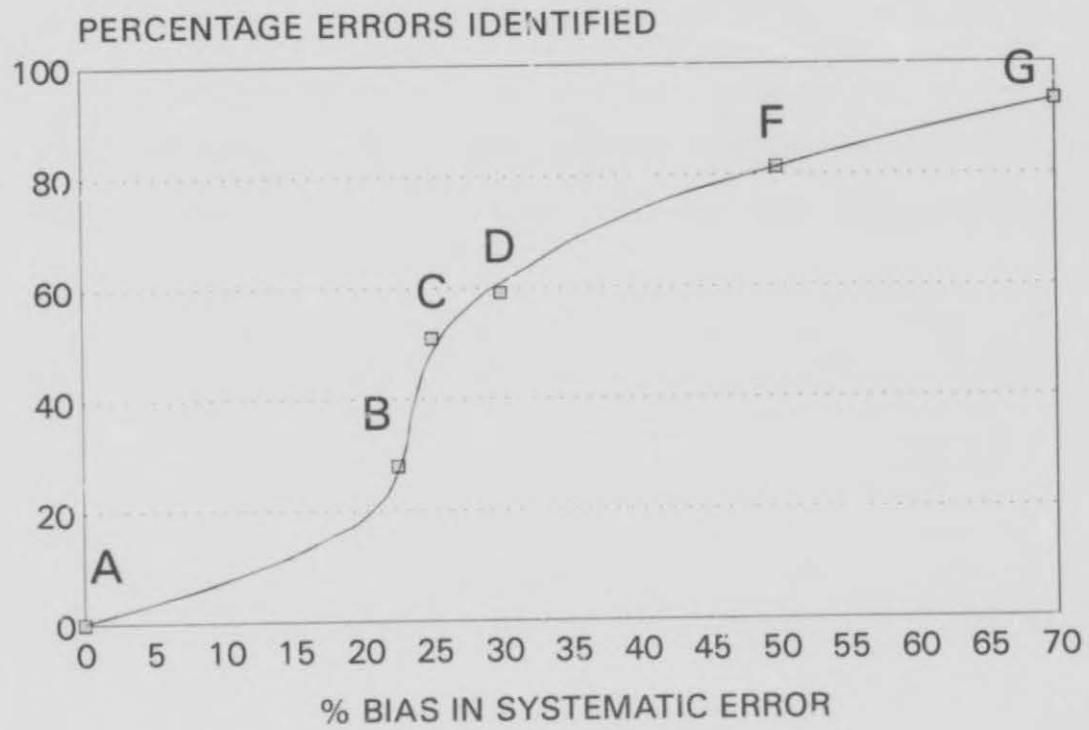


FIGURE 3.8 SYSTEM OF NEURAL NETS TO DETECT ERRORS IN HYDROCYCLONE CIRCUIT (Example 3.3)

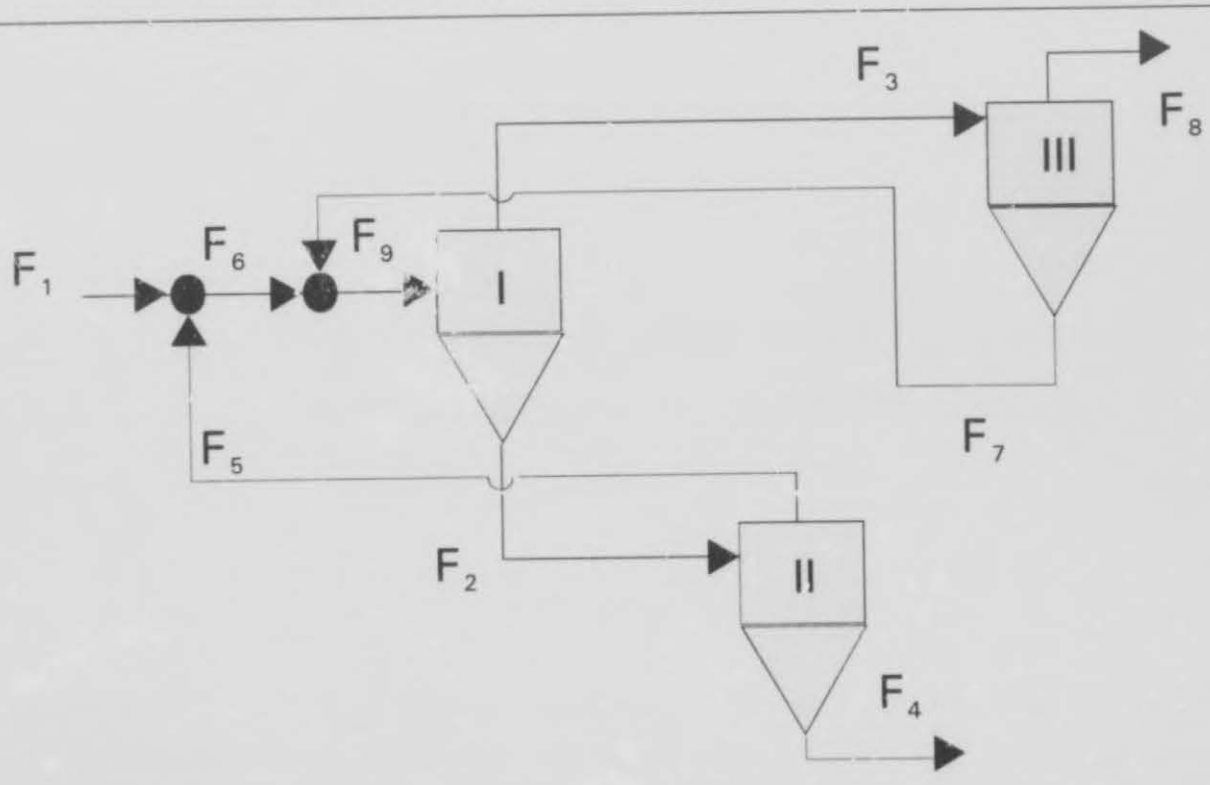


FIGURE 3.9 PERFORMANCE OF BACK PROPAGATION NEURAL NET DURING TRAINING

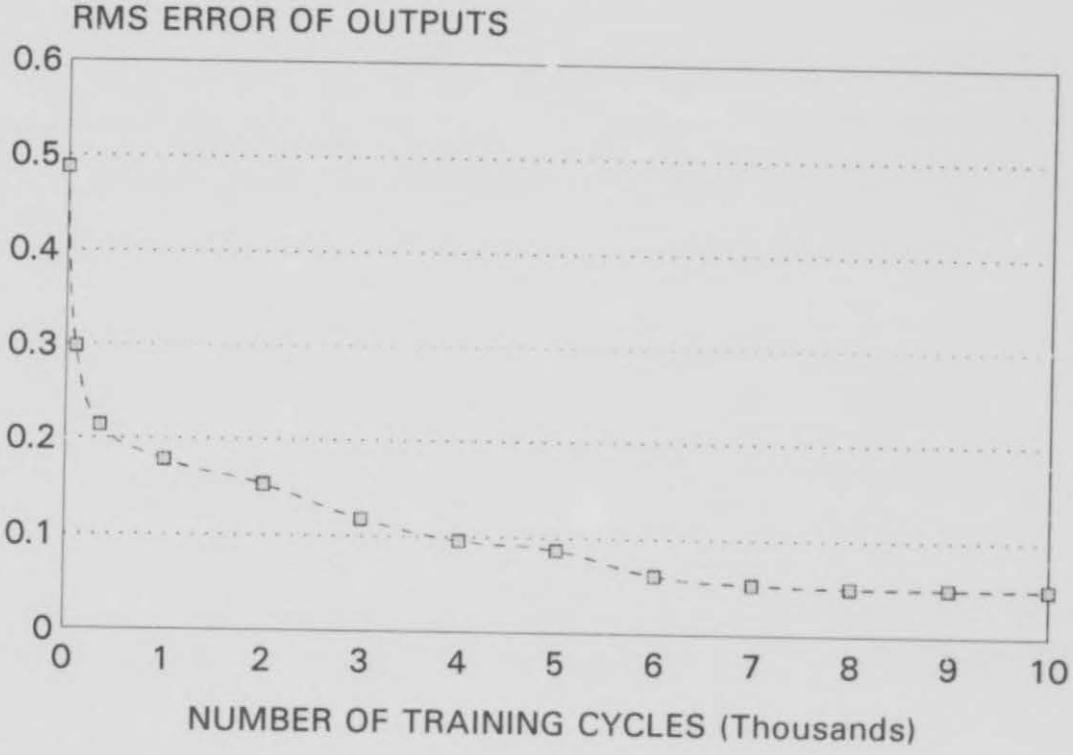


FIGURE 3.10 IDENTIFICATION OF SYSTEMATIC ERRORS IN MEASUREMENT DATA (EXAMPLE 3.3)

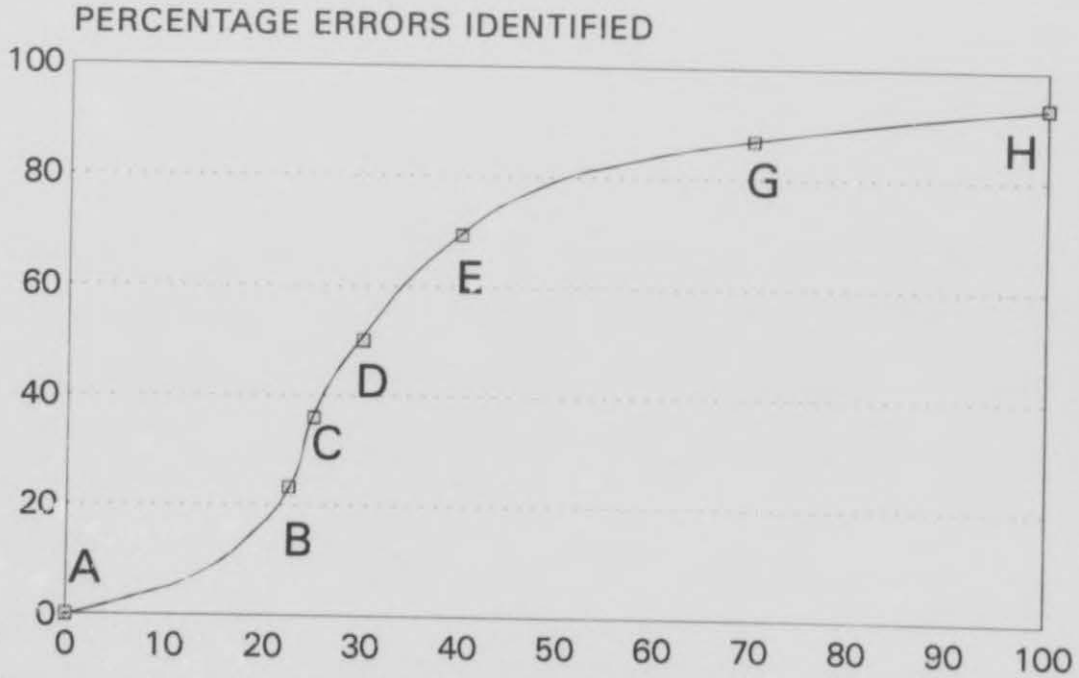


FIGURE 3.11 METALLURGICAL GRINDING CIRCUIT (Example 3.4)

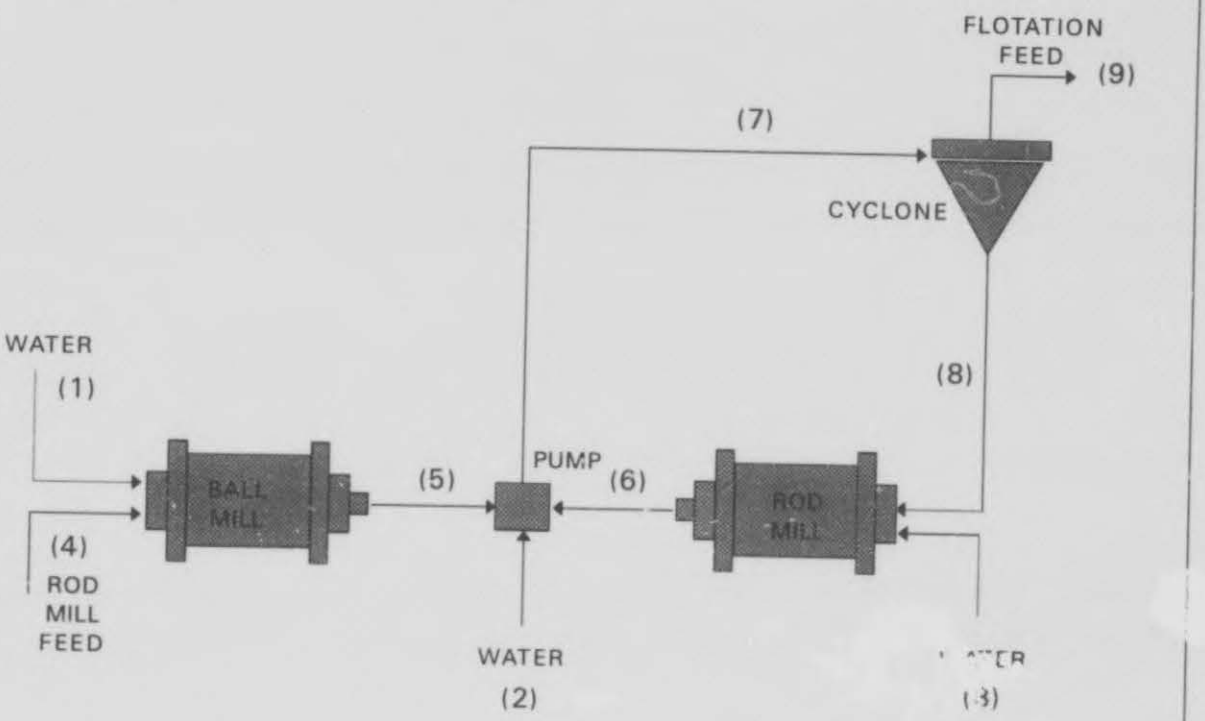
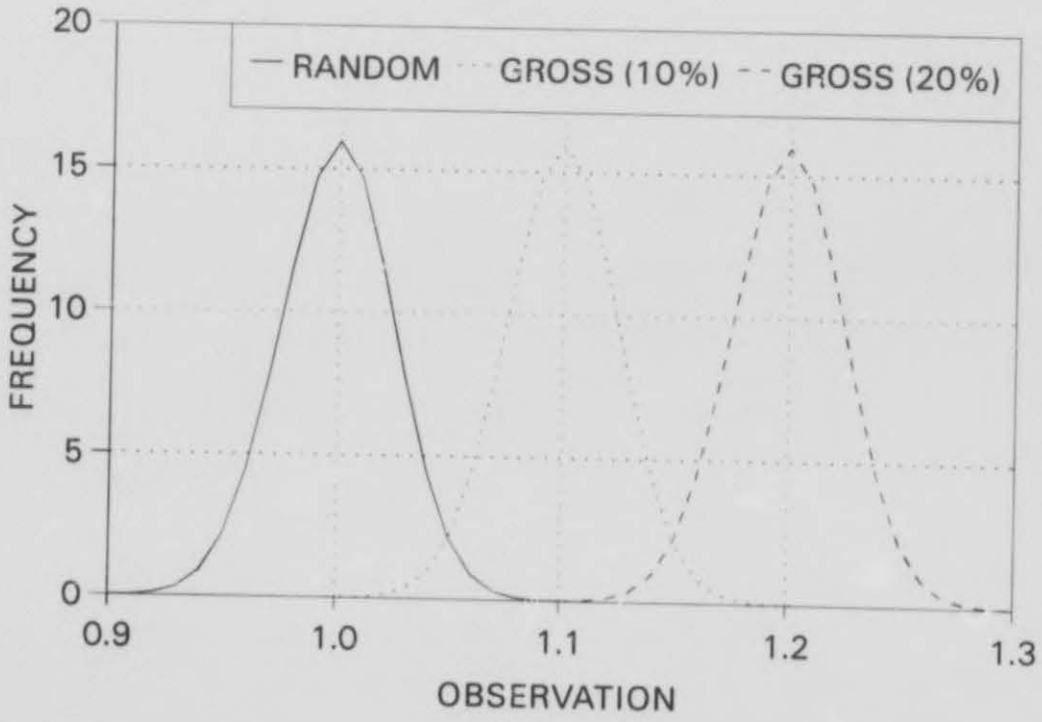


FIGURE 3.12 NORMAL DISTRIBUTION OF RANDOM AND GROSS ERRORS

Standard deviation of all errors: 2.5%

Serth & Heenan (1987)



CHAPTER 4

The reconciliation of inconsistent process data

Summary

Since measurements of variables in chemical and metallurgical plants generally violate the conservation and other constraints of these systems due to random measurement errors, these data have to be reconciled with the constraints prior to further use. In multicomponent systems the reconciliation of process data normally results in a non-linear constrained optimization problem, which can constitute a formidable computational burden when large systems have to be solved by conventional techniques. Connectionist systems, such as artificial neural networks can be implemented to considerable advantage for the solution of optimization problems such as these and in this dissertation their use is explored. Three variants of crossbar feedback connectionist systems have been investigated, two of which are based on gradient descent techniques and one based on a direct search method. The results of simulations, as well as a comparison with traditional computational procedures indicate that systems such as these based on gradient descent techniques can be used to solve large systems efficiently

---

4.1 OBJECTIVES OF CHAPTER 4

Robust and accurate procedures are currently available for the reconciliation of plant data not consistent with related plant models. These procedures are generally expensive in terms of computational requirements and given the size and complexity of some plant models, these techniques are not adequate for the solution of large-scale problems, or small-scale problems in on-line applications. The objectives of this chapter are thus:

- The examination of connectionist structures for use in data reconciliation problems through the incorporation of standard search methods;
- Evaluation of the dynamics of these systems;
- Comparison of the connectionist systems with conventional efficient non-linear procedures.

4.2 BACKGROUND THEORY

A clear understanding of the operational behaviour of a plant is essential for the identification of process phenomena, as well as for the optimization and control of the plant or process circuit. The collection and analysis of data from processes are therefore an important means for evaluating the performance of a plant or an individual process unit. The available data are generally subject to random noise, or even gross errors, which can among others be attributed to failure or miscalibration of measuring instruments, the departure of the process from steady state owing to malfunctioning process equipment, or significant changes in the environment. It is thus vital that the inconsistent data are reconciled with the process constraints prior to further use.

4.2.1 General material and energy balance problem

The usual approach to the reconciliation of measured variables is aimed at the minimization of the weighted sum of the squares of the measurement residuals, subject to conservation and other constraints of the process (Hodouin & Everell, 1980), i.e.

$$\min (\mathbf{x}'-\mathbf{x}'')^T\mathbf{V}^{-1}(\mathbf{x}'-\mathbf{x}''), \text{ subject to} \qquad (4.1)$$

$$\mathbf{d}(\mathbf{x}'') = \mathbf{0}$$

where $\mathbf{V}^{-1}$ is a residual weighting matrix, usually the inverse of the variance-covariance matrix of the measurements $\mathbf{x}'$, and $\mathbf{x}''$ the vector of reconciled measurements. Where an estimate of this variance-covariance matrix is not available, a numerical weighting system can also be used by defining $\mathbf{V}^{-1}$ as the inverse of the elements of the measurements of the variables $\{\mathbf{x}\}$.

Process circuits are often described in terms of a network consisting of m branches and n nodes, usually so that the nodes correspond with process units in the circuit, and branches correspond with connections or flow streams between the units (Vaclavek & Loucka, 1976; Vaclavek et al., 1979; Romagnoli & Stephanopoulos, 1981; Cutting, 1976; Hodouin et al., 1982). The topology of the circuit can then be described with the use of a

Boolean incidence matrix **A**(*m*,*n*), resulting in a set of material balance equations of the form

$$\mathbf{d(x)} = \mathbf{A.x} = \mathbf{0} \qquad (4.2)$$

where **x** is the vector representing the material flow parameters of the system. It should be noted that although these parameters could represent any desirable physical entity, such as particle size fractions, chemical species or specific gravity, the choice of the particular parameter is related to the structure of the incidence matrix **A**(*m*,*n*). A mill for example, would have a profound effect on a flow parameter representing a particle size fraction, but would merely serve as a conduit for a parameter representing a chemical species. Although the general data reconciliation problem is also concerned with variable classification and the determination of unmeasured variables, this dissertation deals only with measured variables subject to small random errors, in order not to unduly complicate the evaluation of the parallel systems investigated.

The solution of the problem in effect ensures that the flow parameters are adjusted as little as possible, that all the conservation constraints are satisfied, and that the more reliable variable measurements are adjusted less than the less reliable variable measurements.

4.2.2 Conventional optimization procedures

Most conventional optimization procedures involve the identification of the overdetermined measurement errors, followed by rectification of these errors, and then the determination of determinable unmeasured flow parameters. These procedures are used in conjunction with schemes for the identification and elimination of systematic errors, and the re-adjustment of flow variables where necessary (Tamhane & Mah, 1985; Romagnoli & Stephanopoulos, 1981).

Some of the optimization strategies for multicomponent balances include direct substitution of the variables in the constraints into the objective function, optimization by means of Lagrange multipliers, direct solution by means of linear and non-linear programming techniques and the use of the Chebyshev minimax criterion (Hlavacek, 1977). Generalized least squares

techniques have been applied to cement clinker and uranium phosphate grinding processes (Hodouin, et al., 1982; Hodouin & Vaz Coelho, 1987), flotation plants (Hodouin et al., 1988; Hodouin & Everell, 1980) and complex mineral beneficiation plants (Cutting, 1976), for example. A somewhat different method employed by Crowe et al. (1983, 1986) concerned the construction of projection matrices to decompose the problem into two or three subproblems, depending on the linearity of the problem. Univariable search techniques have been employed with great success with strongly unimodal response surfaces (White, et al., 1977), while non-linear problems have been solved by Newton-Raphson techniques, such as pertaining to grinding and classification plants (Cutting, 1976), as well as Gauss-Newton iterative algorithms (Pai & Fisher, 1988). Tjoa & Biegler (1991) made use of a hybrid successive quadratic programming (HSQP) method to rectify material balances around heat exchanger networks, while Sanchez et al. (1992) also used a successive quadratic programming (SQP) method to reconcile material and enthalpy measurements in a demethanization plant.

These techniques usually involve iterative procedures and are computationally demanding, especially as far as large complex plants are concerned, or where on-line material balancing is required. These disadvantages associated with the use of traditional methods make the use of connectionist systems or neural nets an attractive alternative for the optimization of mass balance problems. Due to their massively parallel structures, and recent advances in very large scale integration (VLSI) and ultra-large scale integration (ULSI) electronic circuits (Del Corso et al., 1989; Goser et al., 1989; Murray, 1989), neural nets show great potential for the solution of computational problems of high dimension in processing times several orders of magnitude less than what could be achieved with sequential computational devices (Verleysen & Jespers, 1989; Best, 1990).

In this dissertation the use of connectionist systems for the rectification of inconsistent redundant variable measurements is proposed and demonstrated by way of two elementary examples.

4.3 TYPES OF CONNECTIONIST SYSTEMS

Recurrent or feedback nets, especially those known as crossbar or Hopfield nets, have been used for a wide range of optimization problems, ranging from the solution of non-polynomial (NP) complete combinatorial problems, such as the travelling salesperson problem (Tank & Hopfield, 1986), combinatorial optimization problems subject to inequality constraints (Abe et al., 1992), assignment problems (Wang, 1992a), systems of complex-valued linear equations (Wang, 1992c), the four colour mapping problem (Takefuji & Lee, 1991), the identification and recognition of visual images (Nasrabadi & Choo, 1992; Yuille, 1989), as well as the solution of linear (Tank & Hopfield, 1986; Cichocki & Unbehauen, 1992, Wang, 1992b), non-linear (Kennedy & Chua, 1988; Wang & Tsang, 1991) and dynamic programming (Chiu, et al., 1991) problems. These nets differ from feedforward systems (such as back propagation nets) in that information is not only passed forward through the layers of the net, but backwards or laterally as well. The performance of three different connectionist systems explained below and referred to as CS-I, CS-II and CS-III were investigated.

4.3.1 Connectionist system I (CS-I)

The architecture of CS-I corresponds to that of a crossbar or Hopfield neural net, as shown in figure 4.1. The system consists of three layers, viz. an input layer, a hidden layer with full lateral connections, as well as an output layer. All layers have the same number (N) of elements, and all are provided with linear transfer functions, of the form

$$g(u) = k_1.u + k_2, \quad (k_1, k_2 \text{ constant})$$

The exact number of elements in each layer is determined by the number of process variables to be reconciled (i.e. one element for each process variable).

a) Neurodynamics

When these networks are viewed as dynamic systems, the network computation process can be seen as a system moving in a state space[2] through the constant application of some transition rules. These transition rules are procedures for updating the state of the system, depending on its current state. The system dynamics or neurodynamics of the net are determined by the transition rule, as well as the order in which the system variables or node outputs are updated. If application of the transition rule ceases to affect the current state of the system, the system is said to have converged to a fixed point or attractor in the state space. The set of all initial states or points leading to this fixed point is known as the attractor basin of the particular attractor (Masson & Wang, 1990).

In order to analyze the dynamics of the system it is usually convenient to define a scalar function, which depends on the state of the system and has a definite value for each point in the state space. If the value of this energy or cost function (E) does not increase with a change in the state (**v**) of the system (i.e. $dE/d\mathbf{v} \leq 0$, for all possible **v**) and is bounded from below, it is also a Lyapunov function, and an indication that the system is unconditionally stable.

By mapping the objective function and the constraints of an optimization problem onto this energy function, these problems can be solved in that the optimal solution to the problem is forced to coincide with the minimum energy of the system. The dynamics of the net amount to a constraint relaxation process, where the energy measure is defined by the degree of constraint violation of the system.

To use crossbar or Hopfield nets for material balance reconciliation problems, it is necessary to map the objective function (F_{OBJ}) (which incorporates the process system constraints **d**) onto the net, such as by defining the energy function (E) of the net in terms of the objective function (F_{OBJ})

---

[2]The state space of a set of variables $\mathbf{x} = \{x_i / i = 1,2,3,..n\}$ is the Cartesian product of the domains of the variables, i.e. $\mathbf{D}^* = a_1.a_2.a_3...a_n$. The state space $\mathbf{D}_\mathbf{x}$ of the set of n variables (x) is thus n-dimensional.

$$F_{OBJ} = (\mathbf{x}'-\mathbf{x}'')^T \mathbf{V}_{|x}^{-1} (\mathbf{x}'-\mathbf{x}'') + [\mathbf{d}(\mathbf{x}'')]^T \mathbf{V}_{|d}^{-1} [\mathbf{d}(\mathbf{x}'')]$$

i.e. equivalent to

$$E = (\mathbf{v}^0-\mathbf{v})^T \mathbf{V}_{|x}^{-1} (\mathbf{v}^0-\mathbf{v}) + [\mathbf{d}(\mathbf{v}')]^T \mathbf{V}_{|d}^{-1} [\mathbf{d}(\mathbf{v}')] \qquad (4.3)$$

where the scalar function E represents the energy of the net, $\mathbf{x}'$ the vector of variable measurements, $\mathbf{x}''$ the vector of reconciled measurements, $\mathbf{d}(.)$ the (equality) constraints of the system, $\mathbf{v}^0$ the initial output state of the connectionist system, $\mathbf{v}'$ the current output state of the connectionist system, and $\mathbf{V}_{|x}$ and $\mathbf{V}_{|d}$ some weighting matrices.

By defining the neurodynamics of the net by means of the Newton equations, i.e.

$$du_i/dt = -dE/dv_i \qquad (4.4)$$

the computational energy function E is forced to decrease monotonically, regardless of the nature of this function.

Proof (Takefuji & Lee, 1991):

$$\begin{aligned} dE/dt &= \Sigma_i \, dv_i/dt . dE/dv_i \\ &= \Sigma_i \, dv_i/dt . (-du_i/dv_i) \\ &= -\Sigma_i \, (du_i/dt . dv_i/du_i) . du_i/dt \\ &= -\Sigma_i \, (dv_i/du_i) . (du_i/dt)^2 \leq 0 \end{aligned} \qquad (4.5)$$

As long as the transfer function $v_i = g(u_i)$ is continuous and non-decreasing, dv_i/du_i is always positive and dE/dt always negative or zero. The resultant state of the system can consequently be related to a solution of the problem.

The neurodynamics of the net are thus defined by a set of ordinary differential or difference equations, which have to be integrated at each time increment to determine the output states of the neurons after each change of state.

$$du_i/dt = -dE(\mathbf{v})/d(v_i) \qquad (4.6)$$

$$u_i^{t+1} = u_i^t + -dE(v)/dv_i. \quad t \tag{4.7}$$

Integration of these equations continues until the system has reached a point of stability (i.e. its energy has been reduced to a minimum, so that $du_i/dt = 0$, for all i). In practical terms the system is considered to be stable when

$$\Sigma_i \mid dui/dt \mid \leq \epsilon \tag{4.8}$$

where $\epsilon > 0$ is an arbitrary small convergence criterion. The solution of this system of non-linear equations is based on the use of a gradient descent technique, with constant step size lengths.

b) Scaling of data

Before the data are presented to the net, it is important that they are scaled to ranges that are useful with regard to the neurodynamic function being used. Without proper scaling, process elements could become saturated, which could eventually have a severe effect on the movement of the system through state space. Scaling is usually effected by normalizing the input data. After the network has processed the data, the results are descaled to the original units.

c) Connection weights

The weights of the net are defined by the variance-covariance matrix of the measurements, as well as the weights associated with the process constraint residuals, as presented in equation (4.3). Since estimates of the variance-covariance matrix elements are often not available, a weight matrix based on the actual values of the variable measurements will be defined. This ensures that the values of small variables are not adjusted by increments that are unduly large in relation to the value of the variables themselves.

The scaled input (measured values of the process variables) of the net is clamped to the input layer, and the states of the elements in the hidden layer are updated repeatedly and asynchronously (simulated by a random updating procedure) through numeric integration of the potential of each element. The system is allowed to settle into a minimum point, and the

output of the hidden layers (the solution) is passed forward to the process elements in the output layer, from where it is descaled to yield a solution to the optimization problem.

4.3.2 Connectionist system II (CS-II)

This system is essentially a generalized version of the first one, in that instead of having a single hidden layer, it has a P-dimensional array of hidden layers (If P = 1, the system reduces to CS-I), each containing N elements in general, as shown in figure 4.2. The input section of the system consists of a single input layer, each element of which is connected to a corresponding element in each of the P hidden layers. The elements in the hidden layers are similarly connected to corresponding elements in the output layer. The input and output layers do not process the data, but merely serve as distribution points for data input and output.

The same neurodynamic principles concerning CS-I are applicable, except that once the measurement vector has been fed to each of the different layers in the hidden array, P different sets of initial conditions are generated in the array prior to the commencement of relaxation of the energy of the net. Cycles of state changes are allowed to take place independently in each layer in the array, and when necessary the state of layer is compared with those of its neighbours and the neighbouring state associated with the lowest energy is assumed by the particular layer in the array. Each layer is then again allowed to relax from a stochastically reinitialized condition close to the previous lowest energy state. Communication with a particular element and other elements in the network is allowed to take place only after a particular element has become trapped in an energy minimum. This ensures that the movement of the hidden layer active in the deepest attractor basin in the array is not slowed down unnecessarily by frequent polling to assess the states of its neighbours.

4.3.3 Connectionist system III (CS-III)

Direct search procedures are attractive for the solution of sets of non-linear equations, since they are easy to use and computationally efficient. A direct random search procedure with systematic search space

contraction, such as proposed by Luus and Jaakola (1973) and Luus and Wang (1978), has been incorporated in the neurodynamics of the third system, shown schematically in figure 4.3. CS-III is equivalent to CS-II, with the difference that instead of a gradient-based search, use is made of a direct method with a systematic reduction in the search space associated with each interval. The reduction in the search intervals associated with each of the search variables leads to a more efficient search procedure, since unless the search domain is in the immediate vicinity of the optimum, convergence by means of a random search can be very ineffective (Sarma, 1990). The procedure is implemented as follows:

1. Set the time increment counter $j = 1$.

2. Set up the system, so that the initial states ($\mathbf{v}^0$) of the artificial neurons in all P hidden layers correspond to the measured values (x') of the process variables.

3. Define an initial search range R^0_i for each of the system states v^0_i of the neural net.

4. Determine P sets of values, so that $v^j_i = v^{j-1}_i + \phi^{j-1}_i . R^{j-1}_i$, where ϕ^{j-1}_i is a random number associated with the state of artificial neuron i at time j-1, and $0 \leq \phi^{j-1}_i \leq 1$, for all i and j.

5. Of these P sets, determine the set which minimizes $\Sigma_i \mid du_i/dt \mid$.

6. If $\Sigma_i \mid du_i/dt \mid \leq \epsilon$, terminate the search, if not, reduce the search ranges R^j_i by an amount δ, i.e. $R^j_i = (1 - \delta).R^{j-1}_i$. If $\delta = 1$, terminate the search, if $\delta = 1$, repeat the procedure.

After convergence a set of values **v** will remain, which corresponds to a minimum in the energy of the system, i.e. where $du_i/dt = -dE/dv_i \approx 0$, for all i. This minimum will be the one closest to the initial state of the system, and in multimodal systems it might be necessary to incorporate stochastic procedures which would allow the system to find a global minimum point. The incorporation of procedures such as these was not pursued in this investigation.

4.4 EXAMPLES

4.4.1 Example 4.1: Two-product classifier

In this example a two-product classifier (such as a hydrocyclone or a screw classifier) is considered, which classifies a feed stream (F_1) with n components into two output streams (F_2 and F_3). Measurements of the flow rates (F_i) and component concentrations ($f_{i,j}$) typicaliy violate the mass conservation equations pertaining to the classifier, viz.

$$F_1 - F_2 - F_3 = 0$$

$$F_1.f_{1,i} - F_2.f_{2,i} - F_3.f_{3,i} = 0, \text{ for } i = 1,2, .. n \qquad (4.7)$$

The simulated output of connectionist systems CS-I, CS-II and CS-III is summarized in table 4.1 for a two-component system. As can be seen from figure 4.4, which portrays the performance of the CS-I system, the value of the objective function (energy of the net) decreases rapidly at first for step sizes smaller than 0.2, after which diminishing progress is made with further computation. (A constant step size was used for all the variables throughout the optimization procedure.) Step sizes larger than 0.2 resulted in unstable behaviour of the system. The iteration steps referred to in figure 4.4 comprize cycles through which each variable is updated once on average. For the two-product classifier, with 21 process variables F_i, $f_{i,j}$ ($i = 1, 2, 3$ & $j = 1, 6$), an iteration step thus consisted of a series of 21 random variable selections and subsequent adjustments of the selected variables. The reconciled values of the flow rates F_i and concentrations $f_{i,j}$ resulted in a threefold order of magnitude decrease in the objective function (energy function of the net), which is more or less comparable to results obtainable with other optimization techniques.

The performance of CS-II (number of layers = 10 & 100) is compared with that of CS-I (number of layers = 1) in figure 4.5. It is clear that the additional layers in the system do not lead to a significant improvement in performance. This is not surprizing, since the process system considered is subject to bilinear constraints only, and does not have a highly non-linear character.

The connectionist system based on direct search techniques (CS-III) did not perform very well compared to those based on gradient descent techniques (CS-I and CS-II), as can be seen from figure 4.6. The system used in this case consisted of 200 layers and had an initial range of 0.1 for each search variable. This range was contracted to zero as the search progressed, but only resulted in a decrease of about 60%-70% in the initial energy of the system. Other initial search ranges and contraction procedures did not lead to significantly better results.

4.4.2 Example 4.2 (Pai & Fisher, 1988)

This example (also used in chapter 3) is based on the one used by Pai & Fisher (1988), as well as Tjoa and Biegler (1991). It is thus possible to make a rough comparison of the performance of the neural net with the computational procedures used by these authors. The example involves five measured variables x_1, x_2, x_3, x_4 and x_5 and three unmeasured variables x_6, x_7 and x_8 subject to six non-linear constraints.

$$\tfrac{1}{2}(x_1)^2 - 0.7x_2 + x_3x_6 + (x_2)^2x_6x_7 + 2x_3(x_8)^2 - 255.8 = 0$$

$$x_1 - 2x_2 + 3x_1x_3 - 2x_2x_6 - x_2x_7x_8 + 111.2 = 0$$

$$x_3x_6 - x_1 + 3x_2 + x_1x_7 - x_3(x_8)^{\frac{1}{2}} - 33.57 = 0$$

$$x_4 - x_1 - (x_3)^2 + x_7 + 3x_8 = 0$$

$$x_5 - 2x_3x_6x_8 = 0$$

$$2x_1 + x_2x_3x_6 + x_7 - x_8 - 126.6 = 0 \qquad (3.62)$$

As was mentioned in chapter 3 (example 3.5), the exact values of these variables are $\mathbf{x} = \{4.5124, 5.5819, 1.9260, 1.4560, 4.8545, 11.070, 0.61647, 2.0504\}^T$ (Tjoa & Biegler, 1991). Tjoa and Biegler (1991) corrupted 100 sets of these data with Gaussian noise in order to conduct a statistical evaluation of a tailored objective function in a non-linear computational routine, as well as a hybrid successive quadratic programming (SQP) routine.

In order to evaluate the use of a neural net to reconcile inconsistent constrained data, the exact values of the variables are similarly corrupted

by Gaussian noise of 10% and 30%. The errors of a single set of variables resulting from the errors of corruption are shown in tables 4.2 (10% noise) and 4.3 (30% noise).

One of the salient features of the system is the highly irregular response surface of the energy function of the net. The consequence of this highly non-linear character of the system is that the energy function is extremely sensitive to adjustment of the variables, especially at points where the derivative of the energy or objective function with regard to the adjustable variable (E/ x_i' or E/ x_i'') is very large (positive or negative). As a result very small time steps had to be used in order to ensure that the adjustment of a variable does not lead to overshooting of a local optimum in the energy function surface.

In the case of network CS-I, the optimal step size for each variable is determined by a subroutine which systematically decreases the value of the initial time step if at first it does not result in a decrease in the system energy, until an improvement in the objective function is found. In this way relatively large time steps can be taken initially, which can be adjusted near troublesome spots on the surface of the energy function when necessary.

The results which compare favourably with those obtained by other non-linear methods (Tjoa & Biegler, 1991) are shown in tables 4.2 and 4.3, and typically led to a reduction of three orders of magnitude in the energy of the system after approximately 40 iteration steps. The percentage errors in the values of the variables before and after reconciliation (compared to the exact values of the variables) are also shown in tables 4.2 and 4.3. Figure 4.7 depicts some of these results graphically. Note that step sizes larger than approximately 10^{-5} lead to an unstable search procedure (compare with values larger than 0.2-0.3 in the previous example). The use of different step sizes for the different search variables (MULTISTEP) instead of a constant step size for all variables, resulted in considerable improvements in the performance of the system.

In contrast to the situation highlighted by example 4.1, much is to be gained by using a multilayer system such as CS-II. In figure 4.8, the significant improvement in convergence based on the use of 10 layers,

versus 1 (CS-I) is illustrated. This can be attributed to the non-linear character of the response surface of the energy function. By making use of CS-II, movement through the state space of the system is accelerated along steeper attractor basin gradients, than is the case when CS-I is used.

Connectionist system CS-III displays the same less favourable convergence behaviour as was the case in the previous example, as shown in figure 4.9. Exponential contraction of the range of the system results in somewhat better performance, compared to a linear reduction.

4.5 DISCUSSION OF RESULTS

Judging from the reduction in the objective function (equation 4.1) of the reconciliation problem, the results obtained with neural nets simulated on a computer appear to be comparable to those normally associated with traditional non-linear optimization methods, even though the neurodynamics used in these nets are relatively basic. If necessary the results can be improved upon by making use of more sophisticated neurodynamic functions. These functions could incorporate other stochastic procedures such as simulated annealing and its variants (Jeffrey & Rosner, 1986; Kirk et al., 1983; Kirkpatrick, 1992) or hill climbing terms (Takefuji & Lee, 1991) to avoid entrapment in local minima, while moving the system through state space.

The real advantage of using neural nets for data reconciliation problems is the fact that they can be implemented in electronic hardware which could fully exploit the massively parallel architectures of the nets. By making use of analog devices (Verleysen & Jespers, 1989), which typically converge in the characteristic time of the artificial neurons (in the order of 10^{-6} to 10^{-3} seconds), rapid computation is possible (Kamgar-Parsi & Kamgar-Parsi, 1990).

Since this investigation was based on the use of simulated neural nets, and not actual analog nets implemented in electronic circuits, no direct conclusions can be made with regard to the temporal aspects of the computational procedures. A rough estimate of the speedup is provided by Amdahl's law

$$S = P/[P(1-\Omega)+\Omega] \qquad (4.8)$$

where S is the speedup factor, P the number of processors working on the task, and Ω the fraction of the programming code which can be executed in parallel. The time consumed by computational overheads was estimated to be not more than approximately 5% for all three connectionist systems, and on this basis and the results of the optimization experiments, it was possible to estimate the speedup factors for the solution of the data reconciliation problems outlined in examples 4.1 and 4.2. These estimates are summarized in table 4.4. Two different situations are highlighted in the table, namely the location of a solution (local minimum) of the problem, and secondly the location of a global solution or minimum to the problem (by combining a stochastic procedure with a gradient descent or direct search method). The gradient descent methods (CS-I and CS-II) performed significantly better than the direct search procedure (CS-III). As can be expected, the larger the problem, the more is gained by making use of these parallel strategies. According to equation 4.8 the speedup factor is also quite sensitive to the fraction of computer code that can be executed in parallel (estimated to be 95% in this investigation).

The quality of the solutions obtained with the simulated nets indicates however, that analog nets could be employed to considerable advantage to solve data reconciliation problems.

The problems posed in example 4.2 presented non-linearities of a higher degree than the problem discussed in example 4.1. This meant that smaller time steps had to be implemented to ensure a monotonic decrease in the energy of these systems, and as a result these systems took longer to converge than the bilinear two-product classification system. After approximately 10 iteration steps or cycles (see figure 4.4) the energy of the bilinear system discussed in example 4.1 (9 variables) did not show further significant decreases for step sizes larger than 0.3. The energy or objective function of the system considered in example 4.2 (8 variables) on the other hand, decreased by approximately two orders of magnitude after 25 iterations (and showed a decrease of approximately three orders of magnitude after 40 iteration steps).

In figure 4.10 the CS-II system is compared with two other non-linear procedures used for the solution of the problem posed in example 4.2, viz. that of Broyden (Broyden, 1965; Pai & Fisher, 1988) and the constant derivative approach (Knepper & Gorman, 1980). From this graph it can be seen that the CS-II system initially (steps 1 to 3) decreases the value of the energy or penalty function faster than the other two methods. In subsequent iteration steps it loses ground, but in the end (steps 11 and 12) the advantage gained by the methods of Broyden and constant derivatives is largely eradicated. It should be borne in mind that this comparison can serve as a rough guideline only, since the central processing unit (CPU) times associated with the execution of the iteration steps in the different algorithms can not be compared directly. If anything, a comparison of actual CPU times could only be to the advantage of the CS-II system with its relatively simple computational procedures.

Another important factor that should not be overlooked is that in principle the efficiency of the CS-II system is not affected significantly by an increase in the dimensionality of the process system, while other non-parallel procedures such as those depicted in figure 4.10 are usually sensitive to increases in the size of the problem. In large systems consisting of hundreds or even thousands of variables, the CS-II system can consequently be expected to perform significantly better than any other traditional procedure.

4.6 CONCLUSIONS

In this chapter the use of connectionist systems (which were simulated on a digital computer) for reconciling inconsistent measurement data is discussed. It has been shown that

- The measurements of flow streams and assays inconsistent with process models can be reconciled accurately by procedures based on the use of connectionist systems;
- The use of connectionist systems can lead to a significant reduction in the computational effort needed to optimize data reconciliation procedures;

- Even with small problems the performance of connectionist systems is at least comparable to that of conventional procedures;
- In problems of high dimensionality, procedures based on the use of connectionist systems appear to be more efficient than those based on conventional strategies.

4.7 TABLES REFERRED TO IN CHAPTER 4

TABLE 4.1 Reconciled and measured values of the process variables in the two-product classifier (example 4.1)

TWO-COMPONENT SYSTEM[1],[2]

Measured

F_1	F_2	F_3	$f_{1,1}$	$f_{1,2}$	$f_{2,1}$	$f_{2,2}$	$f_{3,1}$	$f_{3,2}$
0.961	0.602	0.347	0.198	0.768	0.127	0.817	0.354	0.608

Reconciled (CS-I)

F_1	F_2	F_3	$f_{1,1}$	$f_{1,2}$	$f_{2,1}$	$f_{2,2}$	$f_{3,1}$	$f_{3,2}$
0.956	0.608	0.348	0.205	0.752	0.123	0.828	0.351	0.613

[1] Ratio of initial energy of system to that of final energy: $E_0/E_f \approx 11000$

[2] The percentage error values e_i' and e_i'' were calculated as $100.[(x_i') - (x_i')_{exact}]/(x_i')_{exact}$ and $100.[(x_i'') - (x_i'')_{exact}]/(x_i'')_{exact}$ respectively

TABLE 4.2 Reconciled and corrupted values (10% Gaussian noise) of the process variables used in example 4.2[1],[2]

CORRUPTED VALUES

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
4.786	5.564	1.917	1.365	5.307	10.225	0.617	2.064
e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
6.06	-0.32	-0.47	-6.25	9.32	-7.63	0.09	0.66

RECONCILED VALUES (CS-I)

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
4.742	5.694	1.903	1.398	4.927	10.942	0.597	2.041
e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
5.09	2.01	-1.19	-3.98	1.49	-1.16	-3.16	-0.46

[1] Ratio of initial energy of system to that of final energy: E_0/E_f = 1000 (Euler); E_0/E_f = 1000 (DSRI)

[2] The percentage error values e_i were calculated as $100.[(x_i') - (x_i)]/(x_i')$

TABLE 4.3 Reconciled and corrupted values (30% Gaussian noise) of the process variables used in example 4.2[1],[2]

CORRUPTED VALUES

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
3.713	4.699	1.365	1.528	3.680	9.080	0.622	2.658
e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
-17.72	-15.82	-29.13	4.95	-24.19	-17.98	0.90	29.63

RECONCILED VALUES (CS-I)

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
5.163	5.367	1.860	1.078	5.041	11.833	0.593	2.451
e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
14.42	-3.85	-3.43	-25.96	3.84	6.89	-3.81	19.54

[1] Ratio of initial energy of system to that of final energy: $E_0/E_f = 1000$ (Euler); $E_0/E_f = 1000$ (DSRI)

[2] The percentage error values e_i were calculated as $100.[(x_i') - (x_i)]/(x_i')$

TABLE 4.4 Estimated speedup factors for examples 4.1 & 4.2

EXAMPLE	SYSTEM	NO OF PRO-CESSORS	SPEEDUP FACTOR
(Search for first local minimum)			
1	CS-I	21	10.50
2	CS-I	8	5.93
(Search for global minimum)			
1	CS-IIa	4200	19.91
1	CS-IIb	4200	19.91
2	CS-IIa	1600	19.77
2	CS-IIb	1600	19.77
1	CS-III	4200	0.42
2	CS-III	4200	0.42

FIGURE 4.1 STRUCTURE OF CONNECTIONIST SYSTEM CS-I

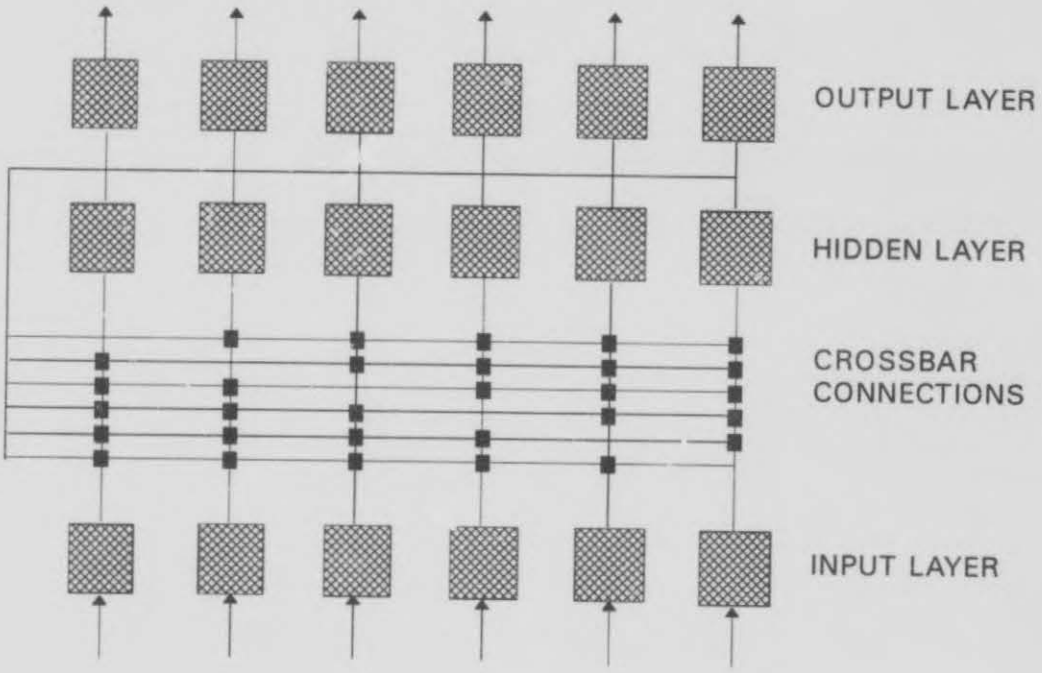


FIGURE 4.2 STRUCTURE OF CONNECTIONIST SYSTEM CS-II

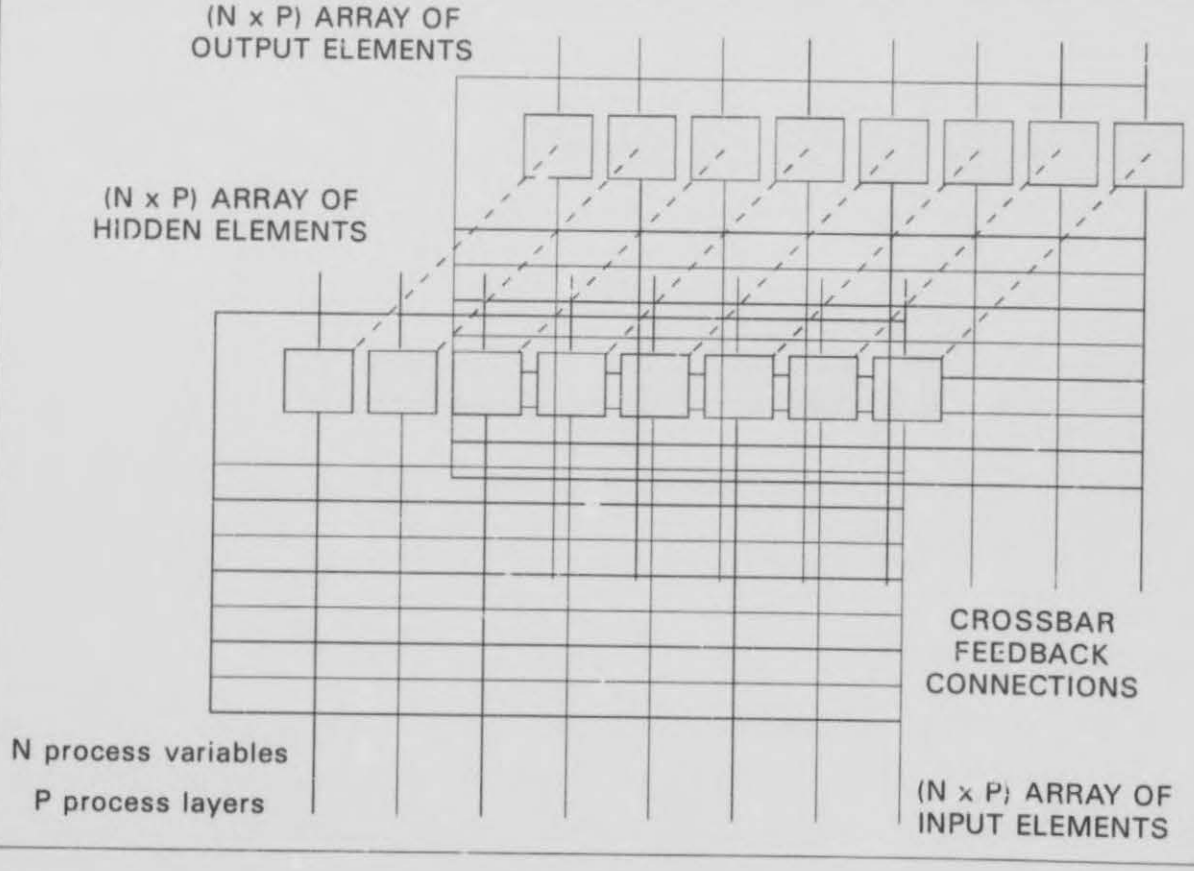


FIGURE 4.3 STRUCTURE OF CONNECTIONIST SYSTEM CS-III

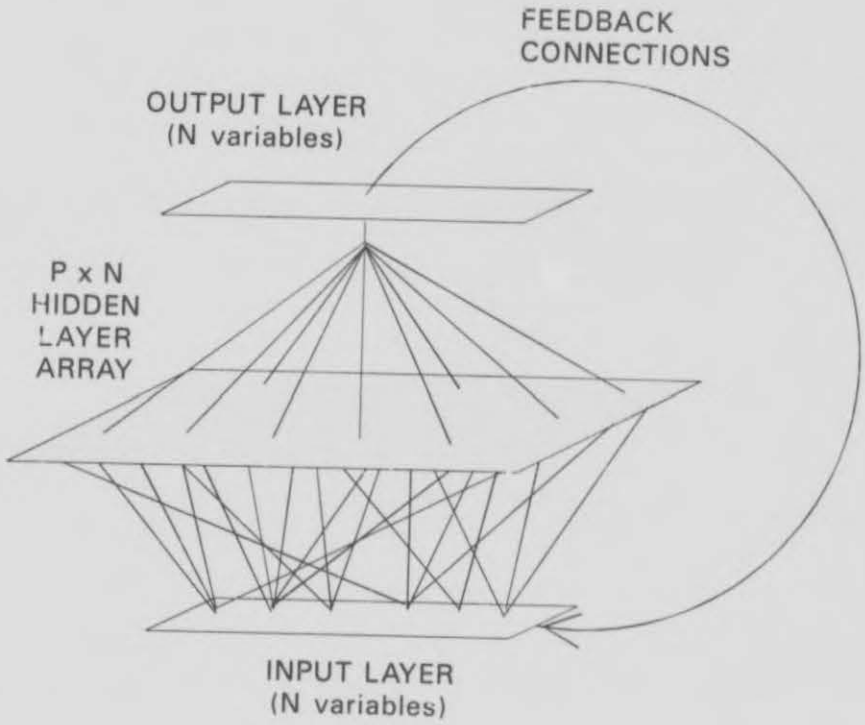


FIGURE 4.4 PERFORMANCE OF CS-I
EFFECT OF STEP SIZE ON DECREASE IN ENERGY
EXAMPLE 4.1

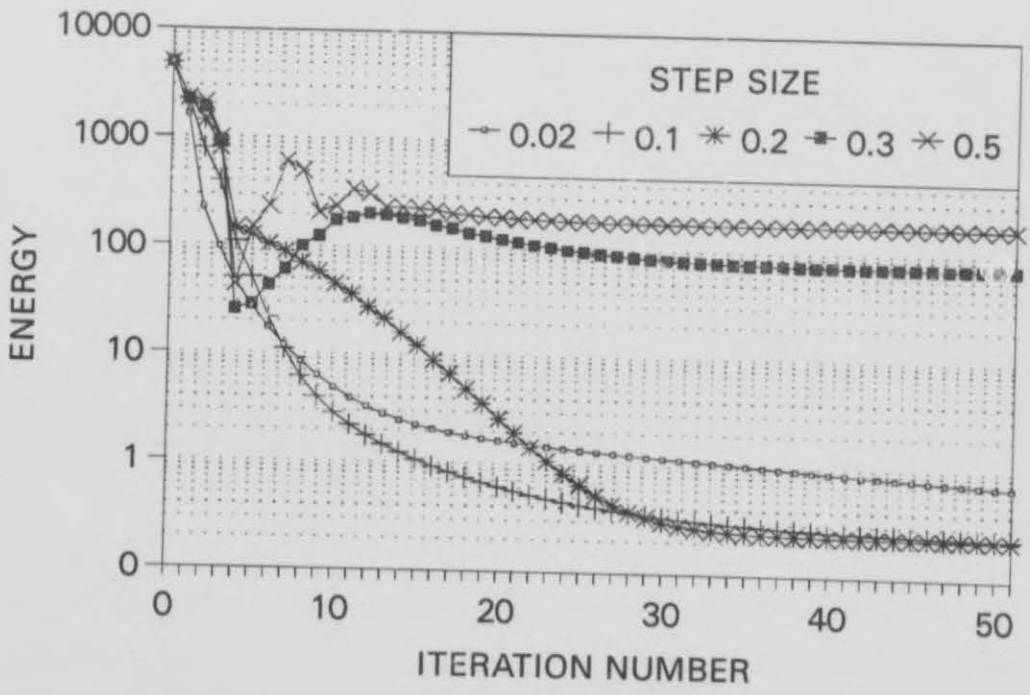
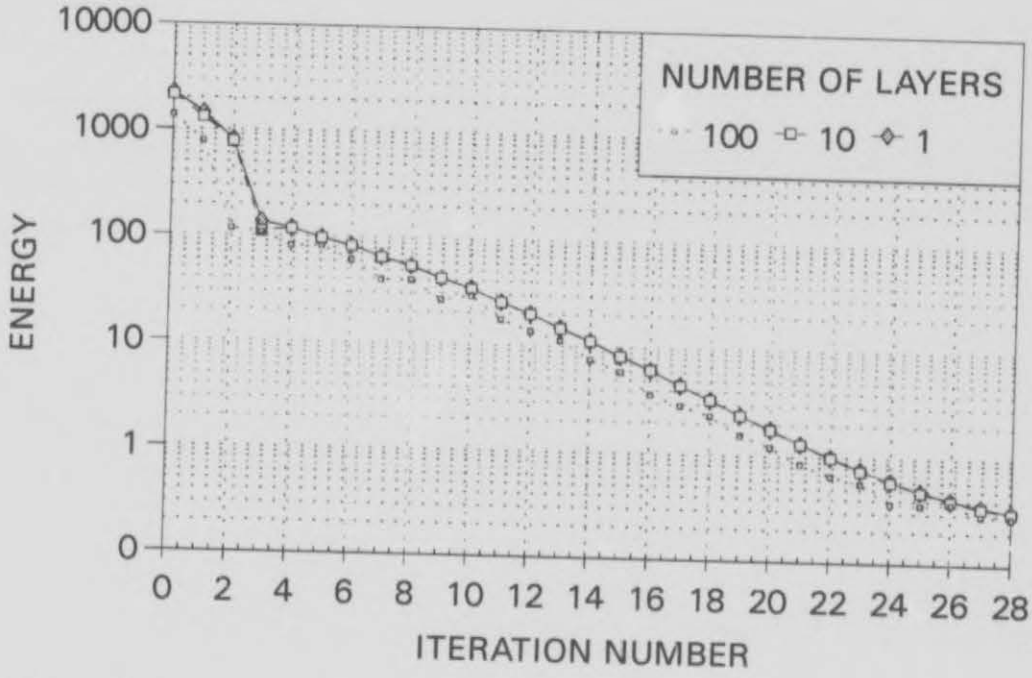
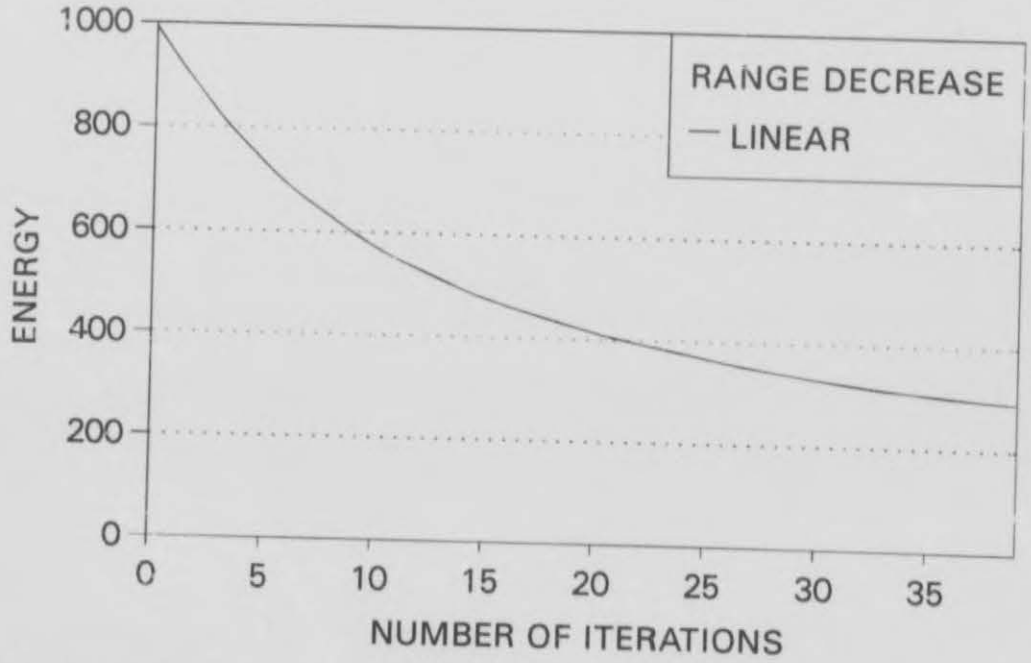


FIGURE 4.5 PERFORMANCE OF CS-II
EFFECT OF NUMBER OF LAYERS ON DECREASE IN ENERGY
EXAMPLE 4.1



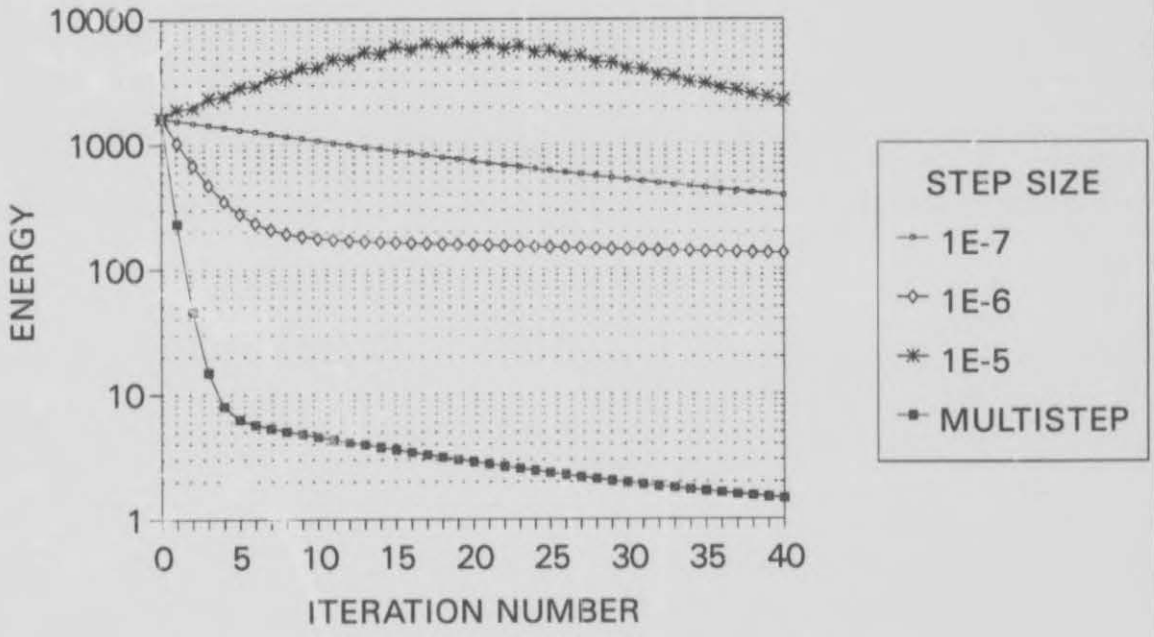
STEP SIZE = 0.2

FIGURE 4.6 PERFORMANCE OF CS-III
EXAMPLE 4.1



200 HIDDEN LAYERS
INITIAL STEP SIZE MAXIMUM: 0.1

FIGURE 4.7 PERFORMANCE OF CS-I EFFECT OF STEP SIZE ON DECREASE IN ENERGY EXAMPLE 4.2



Step sizes constant for all variables, except MULTISTEP

FIGURE 4.8 PERFORMANCE OF CS-II EFFECT OF NUMBER OF LAYERS ON DECREASE IN ENERGY EXAMPLE 4.2

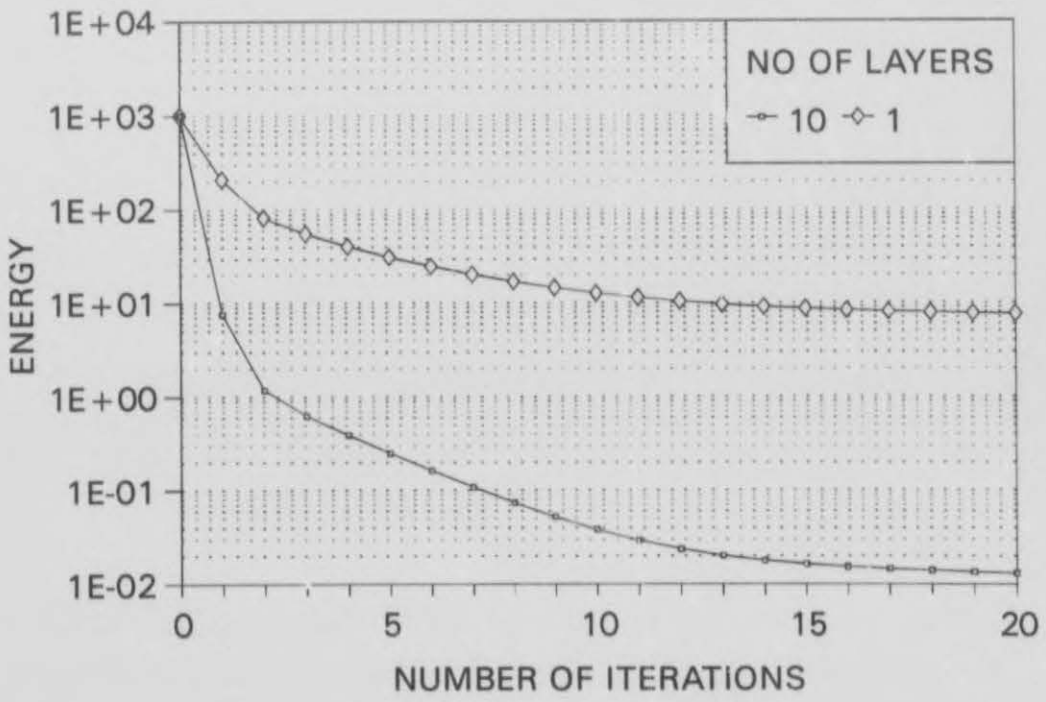
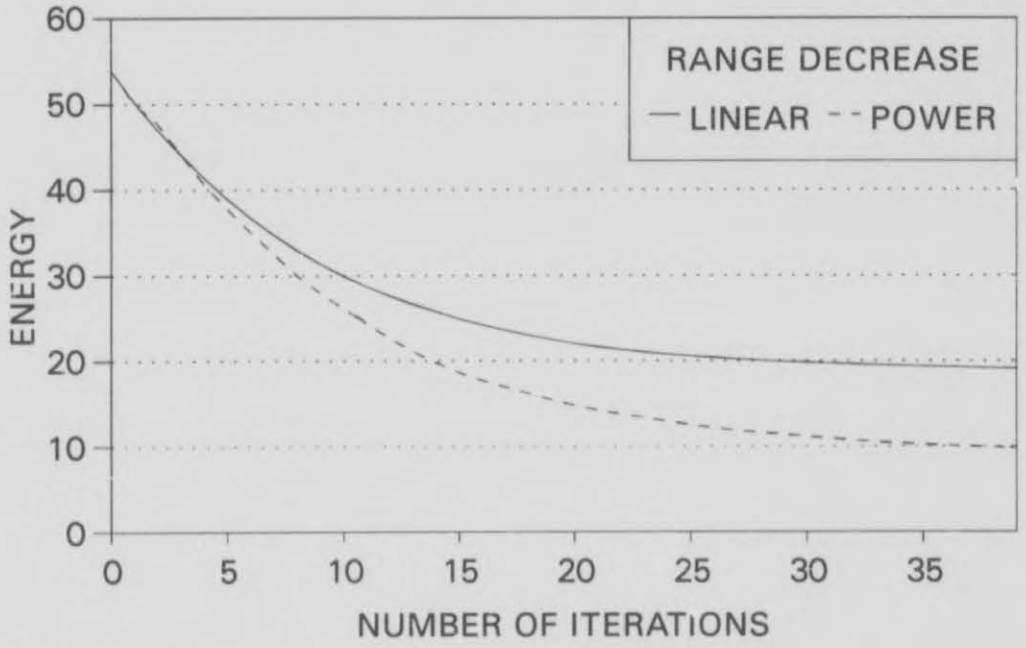
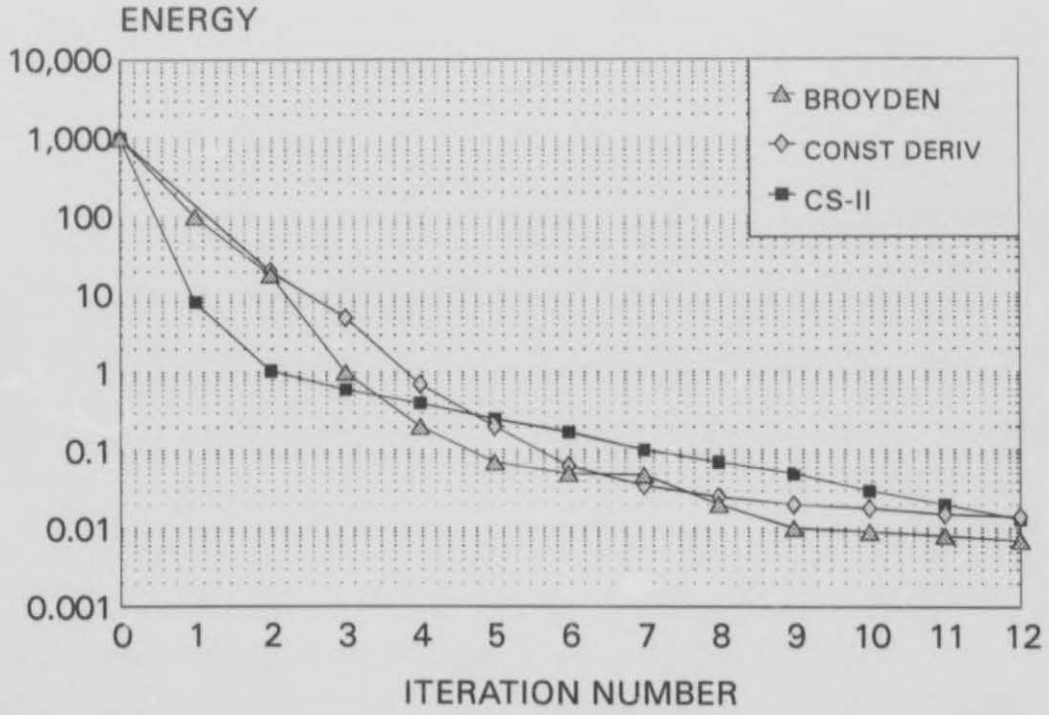


FIGURE 4.9 PERFORMANCE OF CS-III EXAMPLE 4.2



200 HIDDEN LAYERS
INITIAL STEP SIZE MAXIMUM: 0.01

FIGURE 4.10 COMPARISON OF CS-II WITH OTHER NON-LINEAR PROCEDURES



CHAPTER 5

Connectionist plant models

Summary

Although the potential of new techniques for the construction of accurate plant models, such as those based on connectionist methods, is generally acknowledged, little on their practical application can be found in the chemical and metallurgical engineering literature. In this dissertation the use of neural nets to model a gold reduction, a gold leach and a phosphate flotation plant is discussed. The models performed better than the linear regression models used on the plants, even where relatively few data were available. The use of a neural net in conjunction with a linear programming model of a gravity concentration circuit is also explored. Use of the net enables optimization of the circuit through an iterative procedure in which the neural net forces the linear programming models to consider only feasible states of the system.

5.1 OBJECTIVES

The objectives of chapter 5 are:

- The construction of connectionist models for metallurgical process plants, viz a gold reduction plant, a gold leach plant and a phosphate flotation plant;
- Comparison of these models with the models used on the plants;
- The use of a neural net in conjunction with two linear programming models to optimize non-linear process systems (exemplified by a gravity concentration circuit in this case). By linearizing these processes the powerful capabilities of linear programming techniques can be exploited, while incorporation of the neural net model ensures that the non-linear character of the process is retained.
- Investigation of the feasibility of the use of connectionist models on metallurgical plants.

5.2 BACKGROUND

The majority of chemical and mineral processing plants are burdened with copious amounts of process data, which makes it difficult to identify the

essential features of the processes involved in plant operations. The development of process models based on these data is usually not cost effective and the data are usually analyzed by means of multiple linear or non-linear regression techniques. Since these techniques require explicit process models, they are not always suitable for modelling of the complex behaviour that industrial plants so often exhibit. In contrast, neural nets do not suffer from this drawback and (provided they are presented with enough representative data) constitute an efficient means for the construction of implicit models of ill-defined processes. In spite of these well-known attributes (Venkatasubramanian & McAvoy, 1992), very little has been published in the chemical engineering literature with regard to the use of neural nets in this way (Bhat & McAvoy, 1990; Bhat, et al., 1990). In this chapter the use of neural nets for the prediction of gold losses on a gold reduction plant and the consumption of various additives on a gold leach and phosphate flotation plant is described and compared with regression models in use on the plants.

The generalized plant modelling problem consists of two parts, namely the decomposition of the plant into sets of acyclic process circuits if necessary, followed by modelling of these irreducible subsystems. The decomposition of large or complex plants can be accomplished by various means which can among others be incorporated in connectionist structures (see for example appendix C) in order to take advantage of parallel processing strategies. Assuming the process system to be modelled to be acyclic, the problem concerned with the construction of a circuit or plant model can be expressed as follows:

$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & y_{1,2} & .. & y_{1,p} \\ y_{2,1} & y_{2,2} & .. & y_{2,p} \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ y_{n,1} & y_{n,2} & .. & y_{n,p} \end{bmatrix} \quad \epsilon \; R^{nxp} \qquad (5.63)$$

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & .. & x_{1,m} \\ x_{2,1} & x_{2,2} & .. & x_{2,m} \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ x_{n,1} & x_{n,2} & .. & x_{n,m} \end{bmatrix} \quad \epsilon \; R^{nxm} \qquad (5.64)$$

where $y_{i,k}$ (i = 1,2, .. p) represent p variables dependent on m causal or independent variables $x_{j,k}$ (j = 1,2, .. m), based on n observations (k = 1,2, .. n). The variables $y_{i,k}$ are usually parameters which provide a measure of the performance of the plant, while the $x_{j,k}$ variables are the plant parameters on which these performance variables are thought to depend.

The problem is then to relate the matrix **Y** to some function of matrix **X**, in order to predict **Y** from **X**. The simplest approach, and a method often used on mineral processing plants, is to assume a linear relationship between **X** and **Y**, i.e. $\mathbf{Y} = \mathbf{X}.\mathbf{k}_1 + \mathbf{k}_2$ and to find the coefficient vectors $\mathbf{k}_1$ and $\mathbf{k}_2$ by ordinary least squares methods, that is $\mathbf{k}_1 = (\mathbf{X}^T\mathbf{X})^{-1}.\mathbf{X}^T\mathbf{Y}$ and $\mathbf{k}_2 = \mathbf{Y} - \mathbf{X}.\mathbf{k}_1$, provided that the elements of the columns $\mathbf{X}_j$ of matrix **X** are not correlated and that the number of observations is larger than the number of coefficients that has to be estimated (i.e. n > m). If not, other techniques, such as partial least square methods (Qin & McAvoy, 1992) can be used to obviate the problem. Should the assumption of multilinear relationships between the variables prove to be inadequate, they can be extended by the addition of suitable non-linear terms (Loveday & Marchant, 1972), the incorporation of spline methods (Whiten, 1972), or replaced by non-linear regression methods (Britton & Van Vuuren, 1973).

The main advantage of modelling techniques based on the use of neural nets, is that *a priori* assumptions with regard to the functional relationship between **x** and **y** are not required. The net learns this relationship instead, on the basis of examples of related **x**-**y** vector pairs or exemplars.

5.3 MODELLING OF LOSSES ON A GOLD REDUCTION PLANT

The efficiency of gold reduction plants is often assessed in terms of the gold lost during the recovery process, since the recovery of gold (which commonly exceeds 97%) is too insensitive a parameter to use (Britton & Van Vuuren, 1973; MacKay & Lloyd, 1975). Gold losses are generally comprized of the gold lost in a dissolved form, as well as the gold lost in solid residues. These losses can not be explained in terms of a fundamental model of the plant (a typical design is shown in figure 5.1 and is often predicted in practice by means of linear regression models. These models relate the dissolved gold losses (y_1) and the undissolved

gold losses (y_2) to a number of empirical parameters, namely the head grade of the ore (x_1), residual grade of the ore (x_2), solution tonnage (x_3), treated tonnage (x_4), filter feed rate (x_5), filter wash (x_6), solids duty (x_7), filter ARLA (x_8), solution duty (x_9), entering solution (x_{10}), filter flocculation (x_{11}), filter vacuum (x_{12}), sodium cyanide agitator I (x_{13}), and sodium cyanide agitator II (x_{14}).

A set of [x_1, x_2, .. x_{14}; y_1, y_2] data consisting of a total of 76 vectors was obtained from a gold plant in South Africa, randomized and subdivided into a training set (60 vectors) and a test set (16 vectors). The training set consisted of exemplars presented to the neural nets during training (weight adjustment of the nets), while the test set was used to monitor the performance of the nets subsequent to training. This procedure is essential to ensure that the net generalizes the relationships between parameters correctly, instead of just learning to reproduce the data presented to it.

A back propagation net with an input layer and one hidden layer, both comprized of fourteen processing elements or artificial neurons, and an output layer comprized of two processing elements was used to model the gold losses, as shown in figure 5.2. The fourteen elements in the input layer corresponded to the fourteen input parameters used to correlate the gold losses (x_1, x_2, .. x_{14}), while the number of elements in the hidden layer was chosen arbitrarily. The input layer did not process the data, but merely served to distribute the data to the hidden layer. The output of the two processing elements in the output layer corresponded with the predicted values of the two output variables, namely the dissolved (y_1) and undissolved gold loss (y_2).

The layers were connected in a feedforward manner, i.e. no layer was connected to any layer preceding it, and all layers consisted of elements with hyperbolic tangent translation functions, to ensure that low-valued and high-valued outputs were treated equally, i.e.

$$g(u) = (e^u - e^{-u})/(e^u + e^{-u}) \qquad (5.65)$$

The output of the net after training with the generalized delta rule (Rumelhart et al., 1986; Leonard & Kramer, 1990) is compared with the predicted outputs based on a linear regression analysis used on the plant. The results are depicted graphically in figures 5.3 and 5.4. Based on the root mean square values of the correlation errors, the nets performed

significantly better than the existing plant models (approximately 51% for the undissolved gold losses and 87% for the dissolved gold losses).

Neural nets containing more processing elements, either in the same hidden layer or in multiple hidden layer versions, did not substantially improve predictions. Nets with too many processing elements relative to the size of the data training set, have in fact shown a tendency to learn the data, rather than the relationships between parameters (much like fitting a polynomial of too high a degree to too few data).

5.4 MODELLING OF THE CONSUMPTION OF AN ADDITIVE TO A GOLD LEACH PLANT

The consumption of an additive to a gold leach plant depends in a complex way on plant design and operation, and is in practice (as in the previous case) modelled by linear regression models based on empirical parameters considered to influence consumption significantly. In all, seven parameters were used to predict the consumption of the additive (y_1), namely percentage extraction (x_1), residual grade of ore (x_2), cyanide flow rate (x_3), head grade of gold ore (x_4), type of ore (x_5), agitation rate (x_6) and temperature (x_7).

The neural net shown in figure 5.5 consisted of an input layer with seven processing elements corresponding to the seven input parameters x_1, x_2, ... x_7, one hidden layer comprized of seven processing elements as well, and an output layer comprized of a single processing element, corresponding to the consumption of the additive (y_1). All processing elements were provided with hyperbolic tangent transfer functions and were trained by means of the generalized delta rule (a common choice for these types of transfer functions). Training was once again effected by presenting the net repeatedly with exemplars of the experimental data contained in the training set (60 vectors in all). As before, the set of plant data was randomized prior to subdivision into a test and a training data set. After training the net was tested against the data test set (10 vectors in all). The results for both the training and test sets are depicted in figure 5.6. Despite a relatively small training set, the net was able to generalize the relationship between the input and output variables, and predicted the consumption of the additive significantly better (approximately 83%, based on the average root mean square error in prediction) than the multiple linear regression model used on the plant.

5.5 MODELLING OF RECOVERY AND REAGENT CONSUMPTION ON A PHOSPHATE FLOTATION PLANT

The ore feed to a phosphate flotation plant (shown schematically in figure 5.7) is analyzed hourly and these data, as well as those representing other parameters in the plant are averaged on a daily basis and used to predict the consumption of three reagents in the plant, viz. water glass (y_1), polyglycol ether (y_2) and fatty acid (y_3). The water glass or sodium silicate serves as a dispersant and depresses diopside, iron silicates and olivine. The fatty acid acts as a collector for apatite and contributes to the frothing characteristics of the flotation cells, while the polyglycol ether (nonyl phenol tetraglycol ether) is a non-ionic surfactant and emulsifier, serving as a froth modifier and a depressant for iron minerals and calcite. These reagents are expensive (totalling approximately 87% of the direct operating costs of the plant) and inadequate control of their consumption can have a major impact on plant economics (Fourie, 1981). More specifically, the variables (y_1, y_2 and y_3) are related to the mass fractions of apatite (x_1), phlogopite (x_2), lizardite (x_3), magnetite (x_4), diopside (x_5), calcite (x_6), dolomite (x_7) and forsterite (x_8) in the feed, as well as the feed pulp density (x_9), feed flow rate (x_{10}), the phosphate (P_2O_5) concentration in the feed (x_{11}), iron content of the feed (x_{12}) and the tailings (x_{13}) and concentrate (x_{14}) flow rates.

The data used in the investigation consisted of 438 sets of [x_1, x_2, .. x_{14}; y_1, y_2, y_3] vectors, which were subdivided into a training set consisting of 408 vectors, and a test set consisting of 30 vectors. The original data set was not randomized before subdivision into the training and test data sets. This meant that the training data was related to an earlier period of plant operation, while the test data were related to a subsequent period, during which possible changes in the process could have taken place. This is in contrast to the procedures followed in the modelling of the previously discussed gold reduction and leach plants, and is perhaps a more realistic approach to plant modelling, where models are not maintained regularly (or at least not on a daily basis). Instead of using a single neural net with three output nodes, three separate nets with one output each were used - an approach also used by Lucas et al. (1993) that yielded slightly better results than one based on the use of a single net with three outputs. Where more than one workstation is available, this strategy can also be used to reduce the

training times of large nets, since each net could be assigned to a separate machine. If needed the three separate models can be recomposed by placing them side by side connected to the same input vector. The nets used to model the consumption of the different additives were all back propagation neural nets with sigmoidal processing elements, and as before the normalized cumulative delta rule was used to train the nets.

5.5.1 Water glass consumption (y_1)

A single hidden layer consisting of six hidden units ($v_{h,1}$-$v_{h,6}$) was used between the input and the output layers of the net, as illustrated in figure 5.8. The input layer was fully connected to both the hidden layer and the output layer of the net. Instead of training the net to a certain output error tolerance on the training exemplars, use was made of a cross validation method in which the performance of the net was periodically checked against the test data set, until improvement in the performance of the net became marginal. Note that this approach has no effect on the adjustment of the weights of the net during training, but merely serves as a guide to an appropriate neural net structure.

5.5.2 Polyglycol ether (y_2) and fatty acid (y_3) consumption

The nets used to model the polyglycol ether and fatty acid consumption had identical configurations and were comprized of two hidden layers each, as shown in figure 5.9. The first hidden layer consisted of six hidden elements ($v_{h1,1}$, $v_{h1,2}$, .. $v_{h1,6}$), while the second had three ($v_{h2,1}$, $v_{h2,2}$ and $v_{h2,3}$). The input layer was fully connected to the first hidden layer only, while the first hidden layer was fully connected to both the second hidden layer and the output element. As before, a cross validation technique was also used to determine the convergence of the nets.

The performance of the three nets is summarized in table 5.1, where the average percentage errors of the various nets are compared with those of the multilinear regression models. The ability of the nets to generalize the trends in the data is also depicted in figures 5.10 to 5.15, where the predictions corresponding to the training and test data sets are highlighted. As can be seen from these results, as well as those in table 5.1, the nets did not perform better than the regression models on the training data. They were able to generalize the data better than the

regression models however, as shown by the average errors in the test sets.

5.6 STEADY STATE SIMULATION AND OPTIMIZATION OF A GRAVITY SEPARATION CIRCUIT BY MEANS OF LINEAR PROGRAMMING AND ARTIFICIAL NEURAL NETS

Owing to the empirical nature of gravity concentration technology, fundamental modelling of gravity separation circuits is not feasible at present, and as a result most models are of an empirical or semi-empirical nature (Jowett & Sutherland, 1985; Laplante & Shu, 1988). Spiral gravity concentrator circuits can be modelled and optimized by making use of neural nets (representing the requisite empirical knowledge of the system) embedded in conventional computational procedures. In this example simulation is based on two linear programming models and an artificial neural net representing the performance characteristics of the separators under various operating conditions. These concentrators (or in general, the i'th concentrator) each separates a feed ($F^{f}_{i,k}$, k = g,v,w) stream composed of a valuable element (v), gangue (g) and water (w), into a concentrate ($F^{c}_{i,k}$), middlings ($F^{m}_{i,k}$) and tailings stream ($F^{t}_{i,k}$).

The neural net is trained to generalize the relation between the process conditions, viz. the total flow rate (F^{TOT}), the dry solids flow rate (F^{DS}) and the feed grade (Φ), and the concentrate-tailings ($\Gamma^{c,t}_{(i,)k} = F^{c}_{(i,)k}/F^{t}_{(i,)k}$) and middlings-tailings ($\Gamma^{m,t}_{(i,)k} = F^{m}_{(i,)k}/F^{t}_{(i,)k}$) separation factors for each of the three elements *k* in the circuit. (The operating characteristics of the separators are assumed to be identical, so that the subscript i in the separation factors can be omitted.)

This distributed representation of the experimental data can then be used in conjunction with the two linear programming models to simulate and optimize the gravity separation circuit.

5.6.1 Simulation procedure

The strategy used to simulate the gravity separation circuit entails linearization of the model equations (reflecting the material conservation requirements of the system) for each concentrator in the circuit, which facilitates optimization by means of linear programming techniques. The highly non-linear character of the process is retained through the incorporation of an artificial neural net previously trained to represent the

separation of a feed stream $F^{f}_{i,k}$ with a given composition of elements *g*, *v* and *w* into three product streams $F^{c}_{i,k}$, $F^{m}_{i,k}$ and $F^{t}_{i,k}$. The global optimization scheme is iterative and optimization is guided by the neural net in terms of an ill-defined constraint relaxation process, whereby the results obtained by the linear program models are forced to satisfy the process constraints represented by the neural net.

By using the two linear programming models (Anthony, et al., 1991) sequentially (the one a subset of the other), the flow rates of the valuable element and flow paths of the separation circuit are optimized first in order to maximize the recovery of the valuable element, followed by optimization of the concentrate grade by minimizing the gangue in the concentrate streams.

The circuit configuration on which the mass balance equations are based, is shown in figure 5.16, which illustrates the steady state flow of the valuable element (v) between two concentrator units. The flow of the gangue (g) and water (w) is similar to that of the valuable element. Recycle streams are indicated by $ry^{k}_{i,j}$ (concentrate), $rz^{k}_{i,j}$ (middlings) and $rm^{k}_{i,j}$ (tailings). Both linear programming models are derived from a material balance around the general circuit model depicted in figure 5.16. No explicit restrictions are specified and all constraints are derived from experimental data.

5.6.2 Linear programming model I

The model which is described in more detail elsewhere (Anthony, et al., 1991; Reuter et al., 1988; Reuter & Van Deventer, 1990) is formulated by considering all possible process constraints imposed on the material conservation equations of the system shown in figure 5.16.

Mass balance constraints:

$$F^{f}_{j,k} + \Sigma rm_{ji}{}^{k} + \Sigma rz_{ji}{}^{k} + \Sigma ry_{ji}{}^{k} - F^{t}_{j,k} - F^{m}_{j,k} - F^{c}_{j,k} = 0 \qquad (5.26)$$

$$F^{t}_{j,k} - F^{b}_{j,k} - \Sigma rm_{i,j}{}^{k} = 0$$

$$F^{m}_{j,k} - \Sigma rz_{i,j}{}^{k} = 0$$

$$F^{c}_{j,k} - F^{a}_{j,k} - \Sigma ry_{i,j}{}^{k} = 0, \text{ for } (i = 1,2,3;\ j = 1,2,..N \text{ and } k = v, g, w)$$

Separator, external and recycle constraints:

The separation factors used in the model are specified in terms of upper and lower bounds, for each component or element *k* as follows:

$$\Gamma^{c,t,L}_{i,k}.F^{m}_{i,k} - F^{c}_{i,k} \leq 0 \qquad (5.27)$$

$$F^{c}_{i,k} - \Gamma^{c,t,U}_{i,k}.F^{t}_{i,k} \leq 0$$

$$\Gamma^{m,t,L}_{i,k}.F^{t}_{i,k} - F^{m}_{i,k} \leq 0$$

$$F^{m}_{i,k} - \Gamma^{m,t,U}_{i,k}.F^{t}_{i,k} \leq 0,$$

with (k = v, g, w)

In accordance with the plant being modelled, certain non-feasible recycle streams can be eliminated by setting $rm_{i,j}^{k}$ to zero, for the appropriate i,j,k-values. All process variables are furthermore bounded by limits derived from the operational characteristics of the plant.

Objective function

The aim of the objective function of model I is to maximize the recovery of valuable elements, subject to the constraints derived from the mass balance streams, i.e.

$$\text{Max: OBJ} = F^{a}_{1,v}\, F^{a}_{2,v} + .. + F^{a}_{N,v} \qquad (5.28)$$

where $F^{a}_{i,v}$ represents the recovery of the valuable element *v* from concentrator unit *i* in the circuit.

5.6.3 Linear programming model II

Model II minimizes the flow of the gangue and water in the concentrate recovery streams $F^{a}_{i,k}$ and is constrained by the flow configuration determined by model I, i.e.

$$\text{Min: GRADE} = F^{a}_{1,k} + F^{a}_{2,k} + .. + F^{a}_{N,k} \qquad (5.29)$$

with k = g & w, and N the number of spiral concentrators in the circuit.

The separation factors applicable to the gangue and the water are appropriately restricted to reflect the operability limits of the plant:

$$\Gamma^{c,t,L}_{i,g}.F^{t}_{i,g} - F^{c}_{i,g} \leq 0 \qquad (5.30)$$

$$F^{c}_{i,g} - \Gamma^{c,t,U}_{i,g}.F^{t}_{i,g} \leq 0$$

$$\Gamma^{c,t,L}_{i,w}.F^{t}_{i,w} - F^{c}_{i,w} \leq 0$$

$$F^{c}_{i,w} - \Gamma^{c,t,U}_{i,w}.F^{t}_{i,w} \leq 0$$

$$\Gamma^{m,t,L}_{i,g}.F^{t}_{i,w} - F^{m}_{i,g} \leq 0$$

$$F^{m}_{i,g} - \Gamma^{m,t,U}_{i,g}.F^{t}_{i,g} \leq 0$$

$$\Gamma^{m,t,L}_{i,w}.F^{t}_{i,w} - F^{m}_{i,w} \leq 0$$

$$F^{m}_{i,w} - \Gamma^{m,t,U}_{i,w}.F^{t}_{i,w} \leq 0$$

Realistic values are assigned to the flow variables, based on the operational limitations of the circuit.

5.6.4 Neural net representation of separation process

A back propagation neural net with an input layer with three computational elements (one for F^{TOT}, F^{DS} and Φ), a hidden layer with twelve computational elements and an output layer with six computational elements (one for each $\Gamma^{c,m}_{k}$ and $\Gamma^{m,t}_{k}$, k = g,v,w) is used, similar to the structure shown in figure A.1 in appendix A. The net is subsequently trained with a set of exemplars of the form $\{F^{TOT},F^{DS},\Phi,\Gamma^{c,m}_{k},\Gamma^{m,t}_{k}\}$. The exemplars are generated from experimental data obtained from a commercial plant, based on the assumption that the only factors influencing the separation factors are the process conditions F^{TOT}, F^{DS} and Φ. The behaviour of all gravity concentrators is thus considered to be identical. In more sophisticated analyses these assumptions can be modified to take the behaviour of individual process units into account. Presentation of these data enables the net to learn the ill-defined relationship between the separation factors and the process conditions.

5.6.5 Optimized flow circuit

The results of the optimization of a flow circuit containing four gravity concentrator banks are shown in figure 5.17, while the corresponding values of the flow streams are summarized in table 5.2. The simulated separation factors which are modelled with the neural net, satisfy the

experimental data as shown in figure 5.18. Discrepancies between the simulated and experimental data can be attributed to experimental errors, the influence of other less significant parameters not accounted for in the model, as well as the somewhat uneven distribution of the plant data. This modelling methodology can be applied to many other mineralogical separation processes which are difficult to describe fundamentally, such as hydrocyclone classification, heavy medium separation and flotation. As with any model, the success of the procedure depends on the accuracy of the assumptions on which the linear programming models (or other numerical computational routines) are based, as well as the availability of a large body of reliable process data.

5.7 DISCUSSION OF RESULTS

The use of neural nets to model metallurgical plants can lead to significant improvements in the prediction of the behaviour of the plants modelled. Non-linear regression techniques would probably also have led to an improvement on the multilinear regression models, but the identification of suitable explicit models can often be a cumbersome procedure of trial and error, especially where process changes over longer periods necessitate the continuous development of new models to best fit the data over a given period. Neural nets would have to be retrained[3] to accommodate periodic changes in the behaviour of the plant - a relatively inexpensive process with the use of commercial neural network software such as *Brainmaker*™ or *NeuralWorks Professional II/Plus*™.

A surprizing aspect of the investigation is the satisfactory results obtained with relatively few data, especially in the prediction of the consumption of additives in the gold leach plant, as well as those in the phosphate flotation plant. A general rule of thumb for the construction of neural nets

$$N_h = \frac{n}{k.(m + p)}, \qquad 2 \leq k \leq 10 \tag{5.66}$$

where n is the number of exemplars in the training data set, N_h the number of hidden units, m the number of units in the input layer and p the number of units in the output layer, suggests the use of no more than

[3]Progress is being made with the adaptive training of neural nets, which would greatly reduce the expense related to the maintenance of connectionist plant models.

between three and fifteen processing elements in the hidden layers of the net. The factor k depends on the noisiness of the data; k-values of around 2 can be used for noise-free data, while noisy data warrant factors of up to 10 or even 50-100 in extreme cases. In the neural net models of the polyglycol ether and fatty acid consumption, an equivalent of approximately 27 hidden units were used, without detriment to the capability of the nets to generalize the trends exemplified by the training data.

One of the strategies often followed when dealing with non-linear (i.e. mathematically intractable) process systems, is to linearize these systems (usually through Taylor expansions around points of interest). Although these approaches enable the use of well established and powerful mathematical techniques for simulation and optimization, they are often severely limited by their inability to capture the essential characteristics of the system (Kim et al., 1990). In section 5.6 it was shown that the non-linear character of a process system (exemplified by a gravity concentration circuit) can be modelled separately with a neural net, which can then be used in conjunction with linearized models of the system for simulation and optimization. The neural net in effect forces the linear programming models to consider only feasible system states during the search procedure.

A further advantage in the use of connectionist plant models is the potential gain in computational power, due to the parallel architectures of neural nets. Standard procedures for the incorporation of neural nets (which can be trained off-line as often as necessary) into electronic circuits are available and can be used in on-line applications of processing plants (Goser et al., 1989; Murray, 1989; Rosetto et al., 1989; Zurada, 1992).

5.8 CONCLUSIONS

- Ill-defined metallurgical or chemical processing plants can be modelled effectively with neural nets, even when plant data are comparatively sparse;

- The most effective way of modelling plants in the presence of noisy data is through the use of cross-validating methods in which the ability of the net to generalize the relationship between the input and output data is monitored during training;

- A technique based on the use of hybrid neural net linear programming models for the modelling and optimization of a gravity concentration circuit was demonstrated. The technique is sufficiently general for application to other similar separation circuits in the metallurgical industry.

- These neural net models have the additional advantage that they can be implemented in on-line applications with ease.

5.10 TABLES REFERRED TO IN CHAPTER 5

TABLE 5.1 Average % errors in the prediction of additive consumption in the phosphate flotation plant

Training data	y_1	y_2	y_3
MLR	10.05	10.23	11.10
BPNN	10.81	13.79	12.42
Test data	y_1	y_2	y_3
MLR	12.93	23.09	14.35
BPNN	10.78	16.01	9.93

TABLE 5.2 Flow rates [kg/h] in optimized gravity separator bank (see figure 5.17)

$F^a_{1,g} = 125.53$
$F^a_{2,g} = 122.41$
$F^a_{3,g} = 136.59$
$F^a_{4,g} = 55.51$
$F^a_{1,h} = 63.20$
$F^a_{2,h} = 93.87$
$F^a_{3,h} = 81.34$
$F^a_{4,h} = 201.80$
$F^a_{1,w} = 31.89$
$F^a_{2,w} = 36.04$
$F^a_{3,w} = 62.87$
$F^a_{4,w} = 29.84$
$F^b_{1,g} = 717.32$
$F^b_{2,g} = 699.46$
$F^b_{3,g} = 942.01$
$F^b_{1,h} = 4.09$
$F^b_{2,h} = 6.07$
$F^b_{3,h} = 5.26$
$F^b_{1,w} = 1594.25$
$F^b_{2,w} = 1801.80$
$F^b_{3,w} = 1343.32$
$F^t_{1,g} = 717.32$
$F^t_{2,g} = 699.46$
$F^t_{3,g} = 942.01$
$F^t_{4,g} = 444.05$
$F^t_{1,v} = 4.09$
$F^t_{2,v} = 6.07$
$F^t_{3,v} = 5.26$
$F^t_{4,v} = 9.35$
$F^t_{1,w} = 1594.25$
$F^t_{2,w} = 1801.80$
$F^t_{3,w} = 1343.32$
$F^t_{4,w} = 1989.09$
$rm^g_{34} = 444.05$
$rm^v_{34} = 9.35$
$rm^w_{34} = 1989.09$
$rz^g_{14} = 333.04$
$rz^g_{41} = 43.04$
$rz^g_{42} = 41.97$
$rz^g_{43} = 56.52$
$rz^v_{14} = 4.67$
$rz^v_{41} = 27.39$
$rz^v_{42} = 40.68$
$rz^v_{43} = 35.25$
$rz^w_{14} = 69.62$
$rz^w_{41} = 143.48$
$rz^w_{42} = 162.16$
$rz^w_{43} = 282.90$
$F^f_{1,g} = 552.85$
$F^f_{2,g} = 863.83$
$F^f_{3,g} = 691.07$
$F^f_{4,g} = 691.07$
$F^f_{1,v} = 90.00$
$F^f_{2,v} = 140.63$
$F^f_{3,v} = 112.50$
$F^f_{4,v} = 112.50$
$F^f_{1,w} = 1700$
$F^f_{2,w} = 2000$
$F^f_{3,w} = 1500$
$F^f_{4,w} = 2000$
$F^c_{1,g} = 125.53$
$F^c_{2,g} = 122.41$
$F^c_{3,g} = 136.59$
$F^c_{4,g} = 55.51$
$F^c_{1,v} = 63.20$
$F^c_{2,v} = 93.87$
$F^c_{3,v} = 81.34$
$F^c_{4,v} = 201.80$
$F^c_{1,w} = 31.89$
$F^c_{2,w} = 36.04$
$F^c_{3,w} = 62.87$
$F^c_{4,w} = 29.84$
$F^m_{1,g} = 43.04$
$F^m_{2,g} = 41.97$
$F^m_{3,g} = 56.52$
$F^m_{4,g} = 333.04$
$F^m_{1,v} = 27.39$
$F^m_{2,v} = 40.68$
$F^m_{3,v} = 35.25$
$F^m_{4,v} = 4.67$
$F^m_{1,w} = 143.48$
$F^m_{2,w} = 162.16$
$F^m_{3,w} = 282.90$
$F^m_{4,w} = 69.62$

FIGURE 5.1 A TYPICAL GOLD REDUCTION PLANT

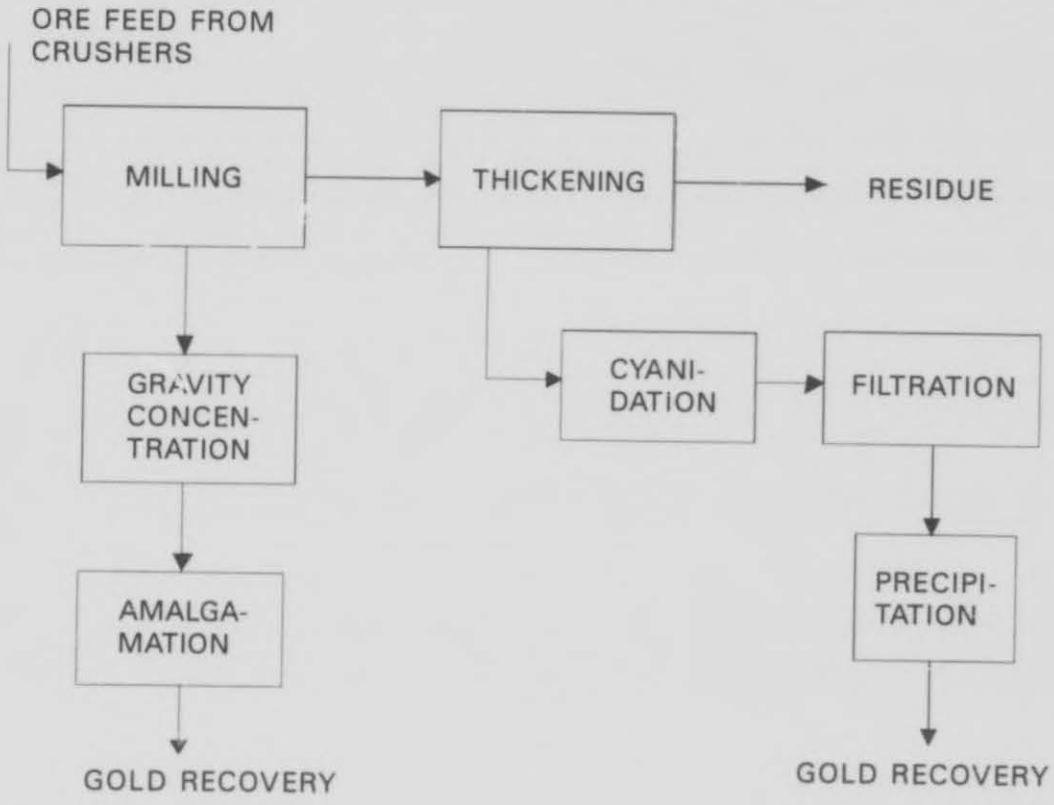


FIGURE 5.2 NEURAL NET MODEL OF GOLD LOSSES

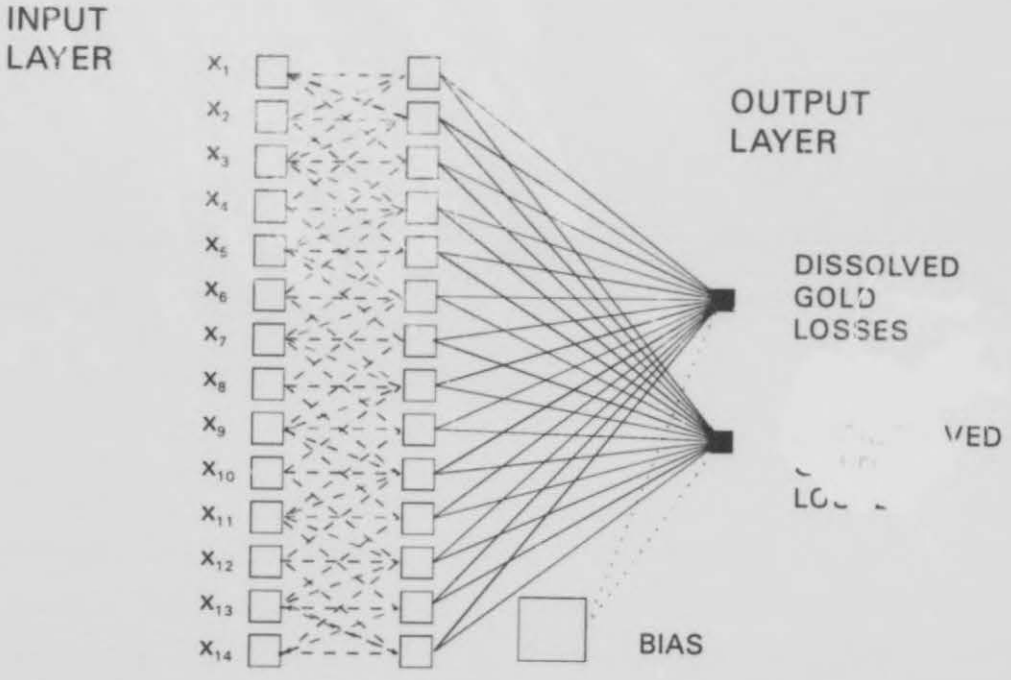


FIGURE 5.3 PREDICTION OF DISSOLVED GOLD LOSSES

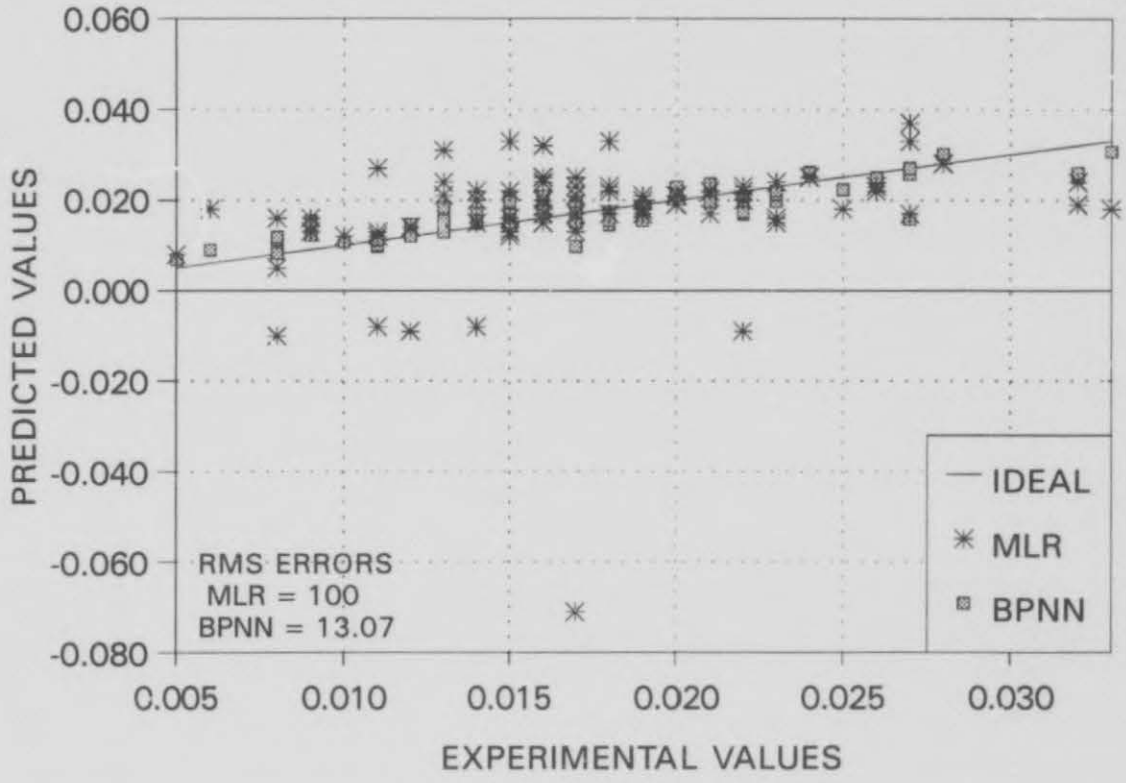


FIGURE 5.4 PREDICTION OF UNDISSOLVED GOLD LOSSES

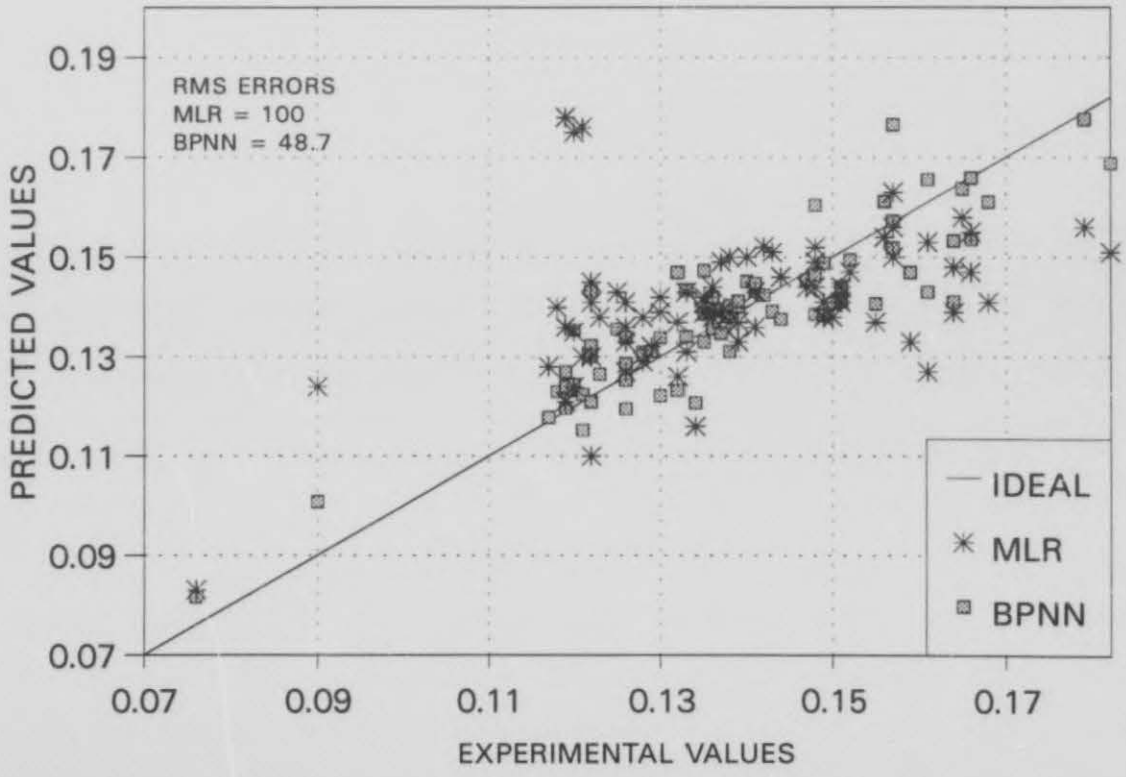


FIGURE 5.5 NEURAL NET MODEL OF ADDITIVE CONSUMPTION IN GOLD PLANT

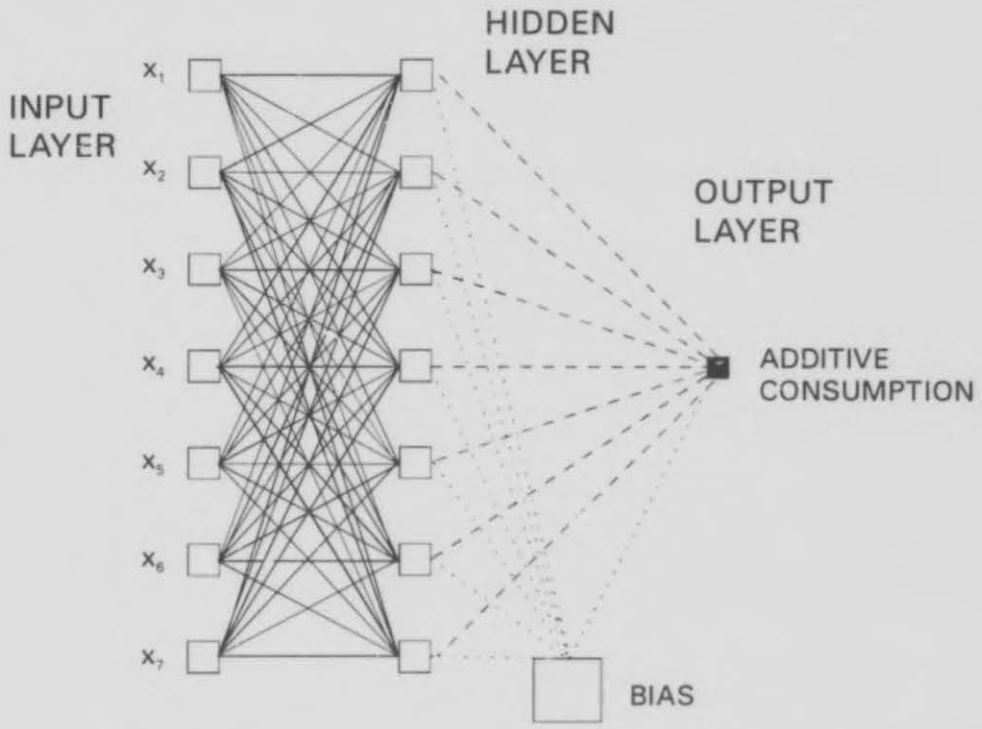


FIGURE 5.6 PREDICTION OF ADDITIVE CONSUMPTION IN GOLD LEACH PLANT

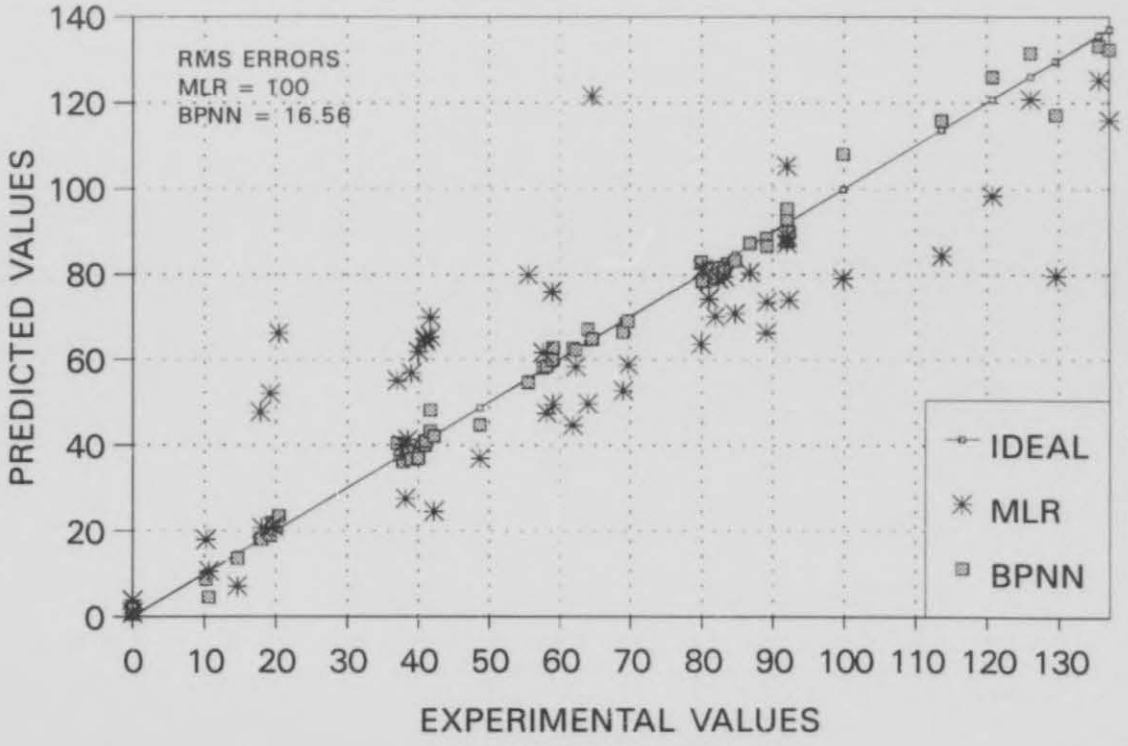


FIGURE 5.7 FLOWSHEET OF PHOSPHATE FLOTATION PLANT

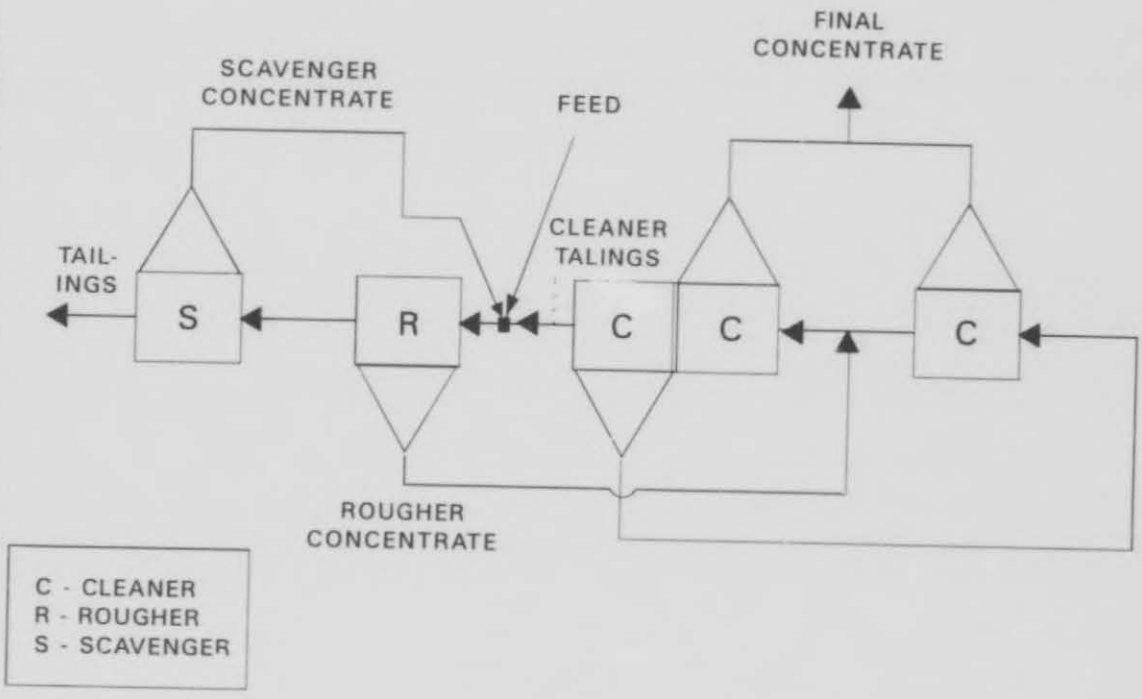


FIGURE 5.8 NEURAL NET MODEL OF WATER GLASS CONSUMPTION

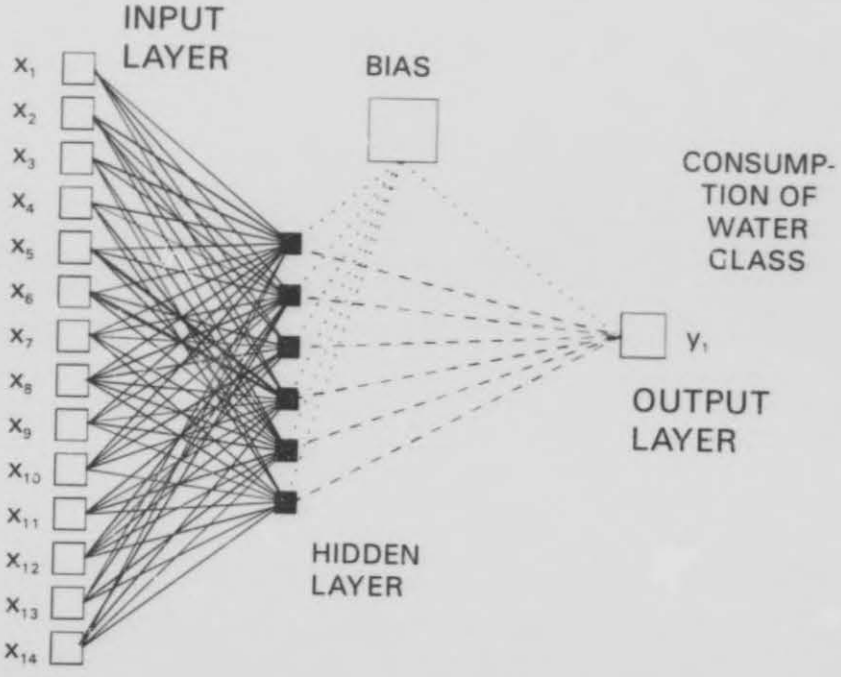


FIGURE 5.9 NEURAL NET MODEL OF POLYGLYCOL ETHER & FATTY ACID CONSUMPTION

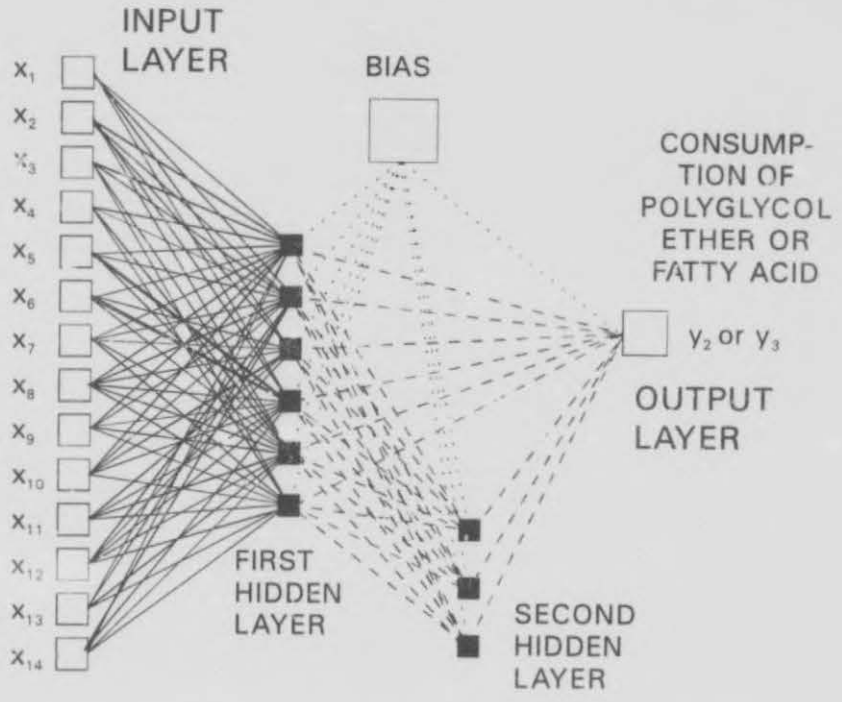
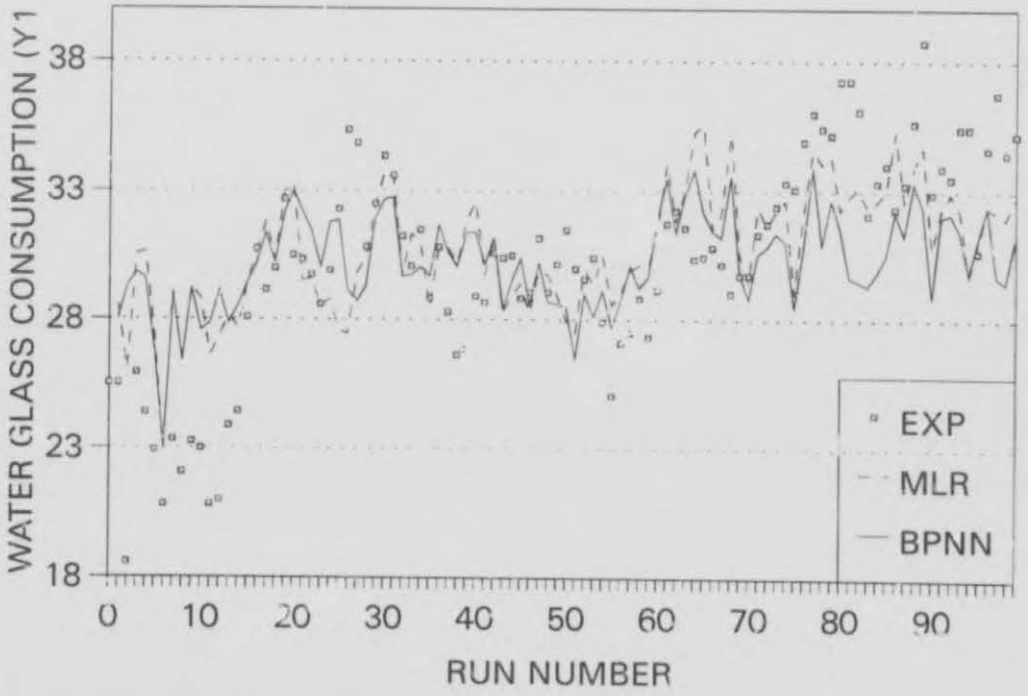
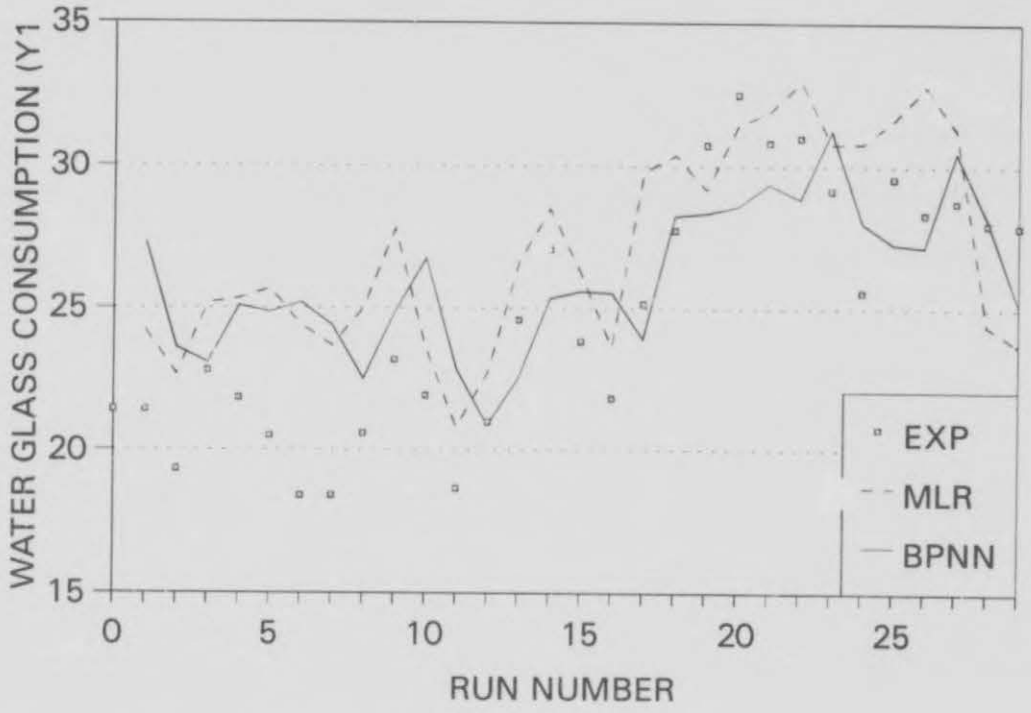


FIGURE 5.10 PREDICTION OF WATER GLASS CONSUMPTION (Y1) TRAINING DATA SET



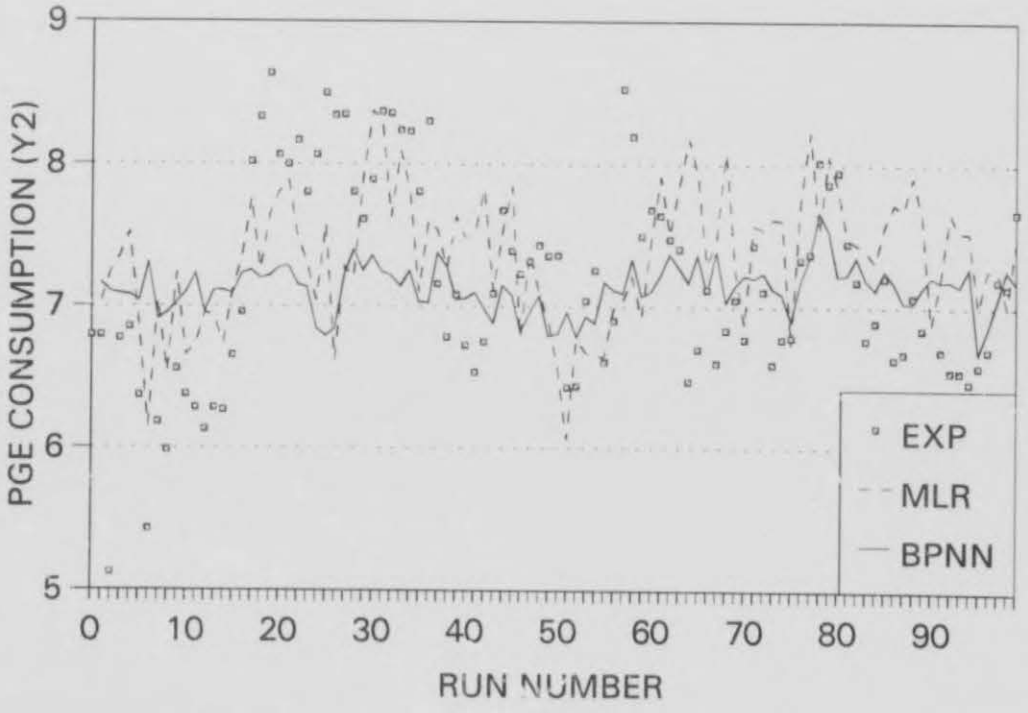
Run numbers are time sequential

FIGURE 5.11 PREDICTION OF WATER GLASS CONSUMPTION (Y1) TEST DATA



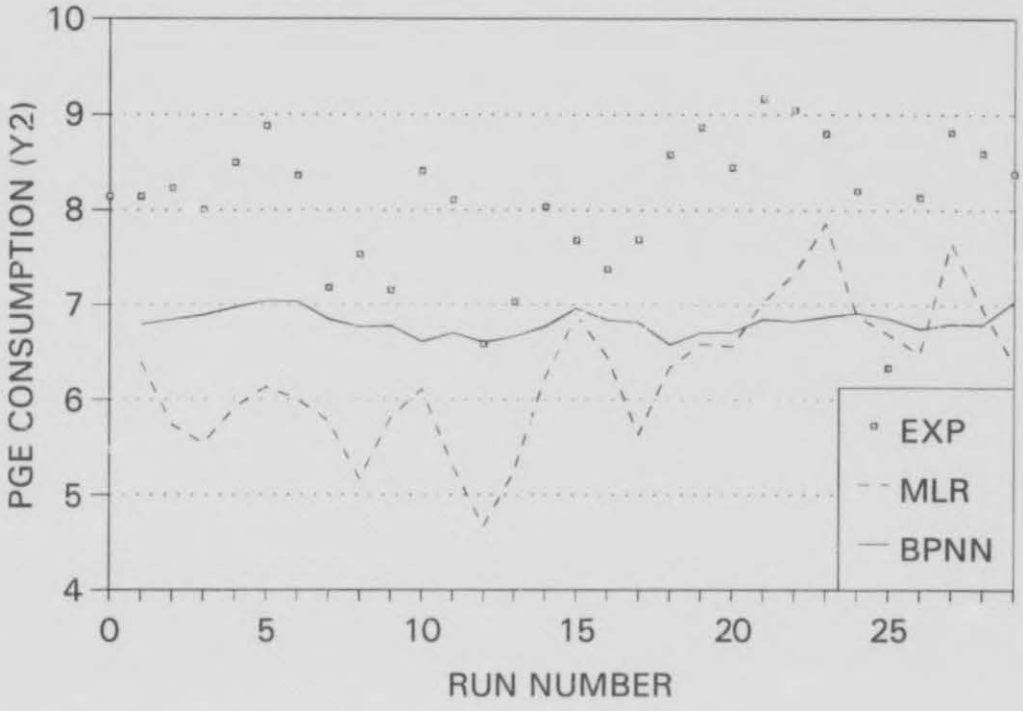
Run numbers are time sequential

FIGURE 5.12 PREDICTION OF POLYGLYCOL ETHER CONSUMPTION (Y2) TRAINING DATA



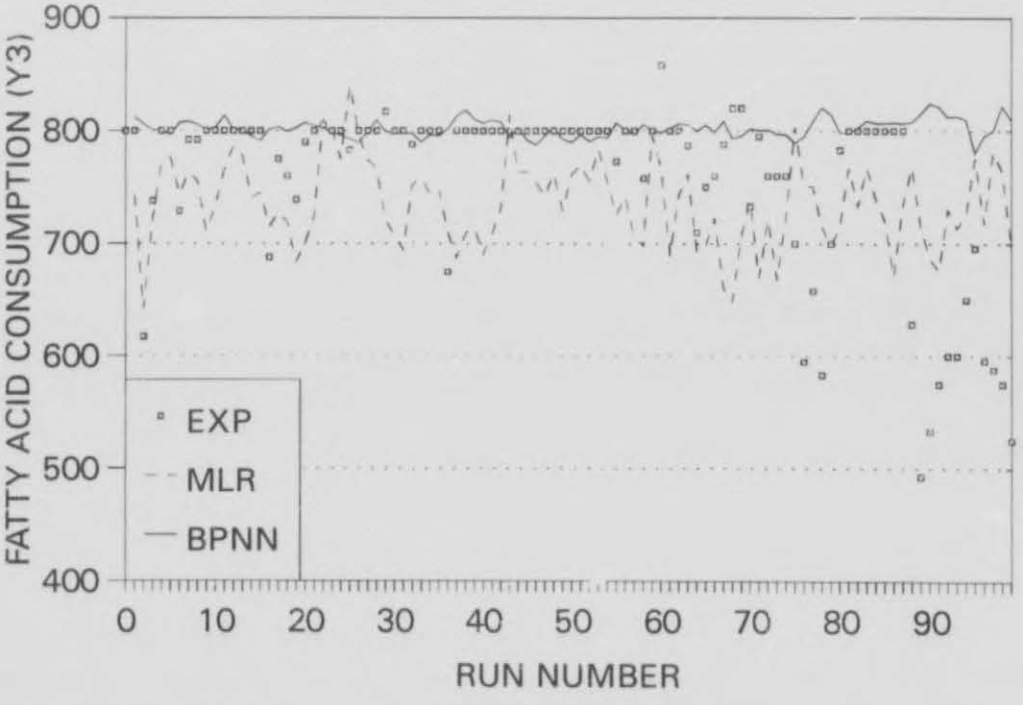
Run numbers are time sequential

FIGURE 5.13 PREDICTION OF POLYGLYCOL ETHER CONSUMPTION (Y2) TEST DATA



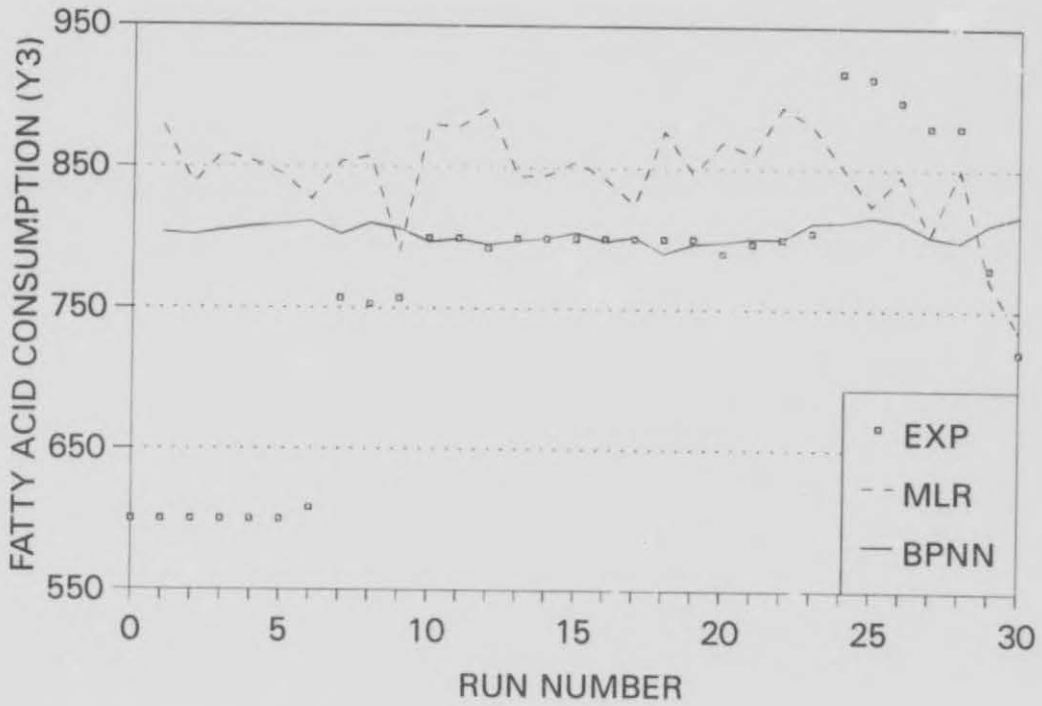
Run numbers are time sequential

FIGURE 5.14 PREDICTION OF FATTY ACID CONSUMPTION (Y3) TRAINING DATA



Run numbers are time sequential

FIGURE 5.15 PREDICTION OF FATTY ACID CONSUMPTION (Y3) TEST DATA



Run numbers are time sequential

FIGURE 5.16 FLOW OF VALUABLE ELEMENT BETWEEN SEPARATOR BANKS i AND j

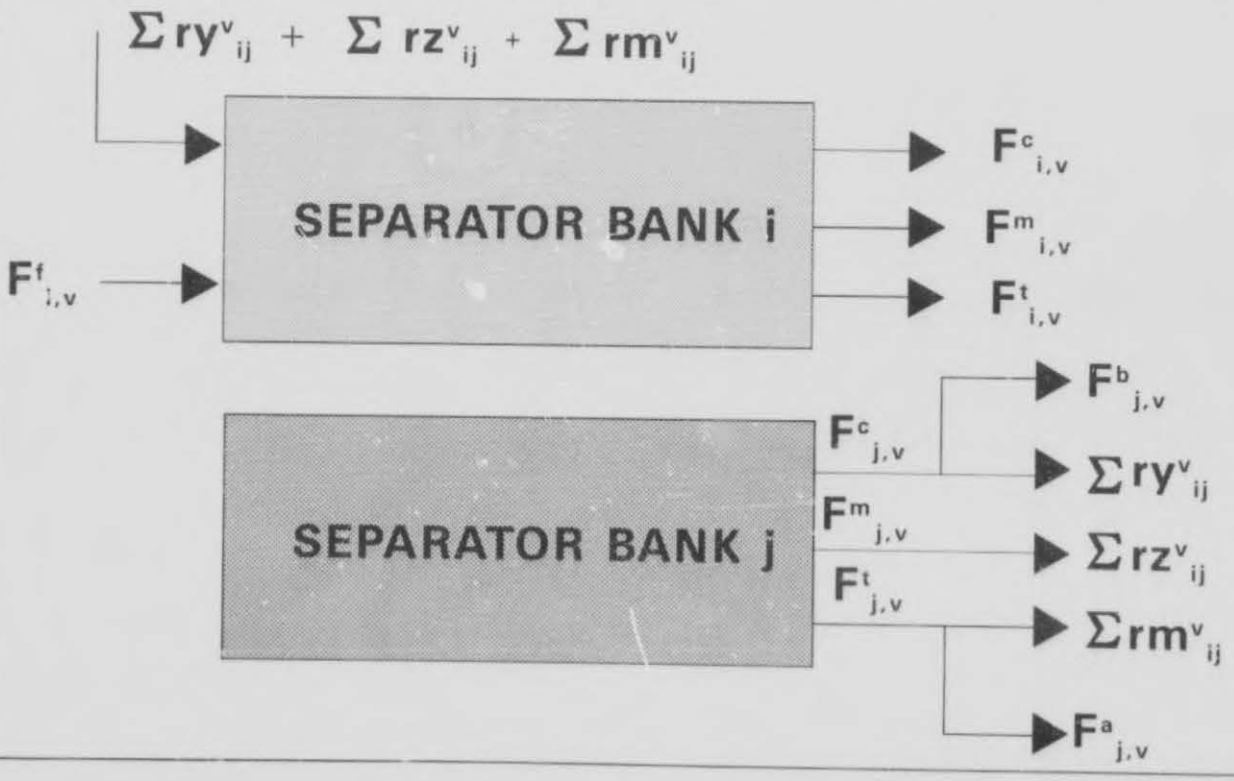


FIGURE 5.17 FOUR-UNIT SEPARATOR BANK AFTER OPTIMIZATION

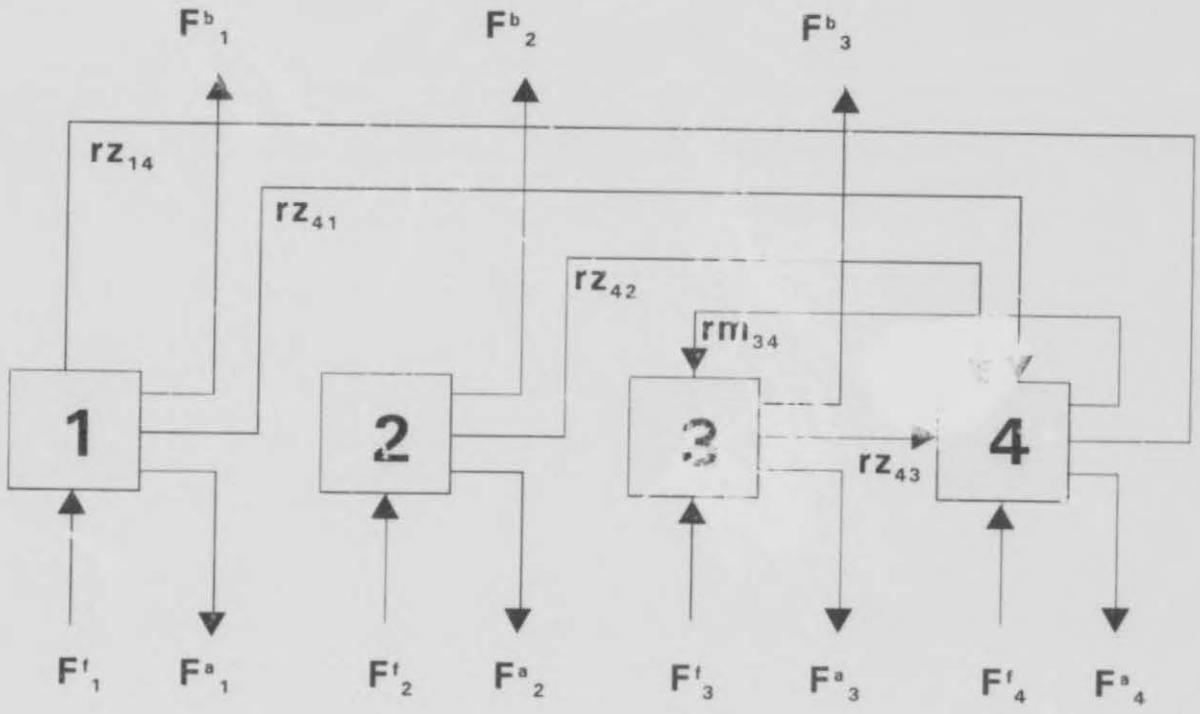
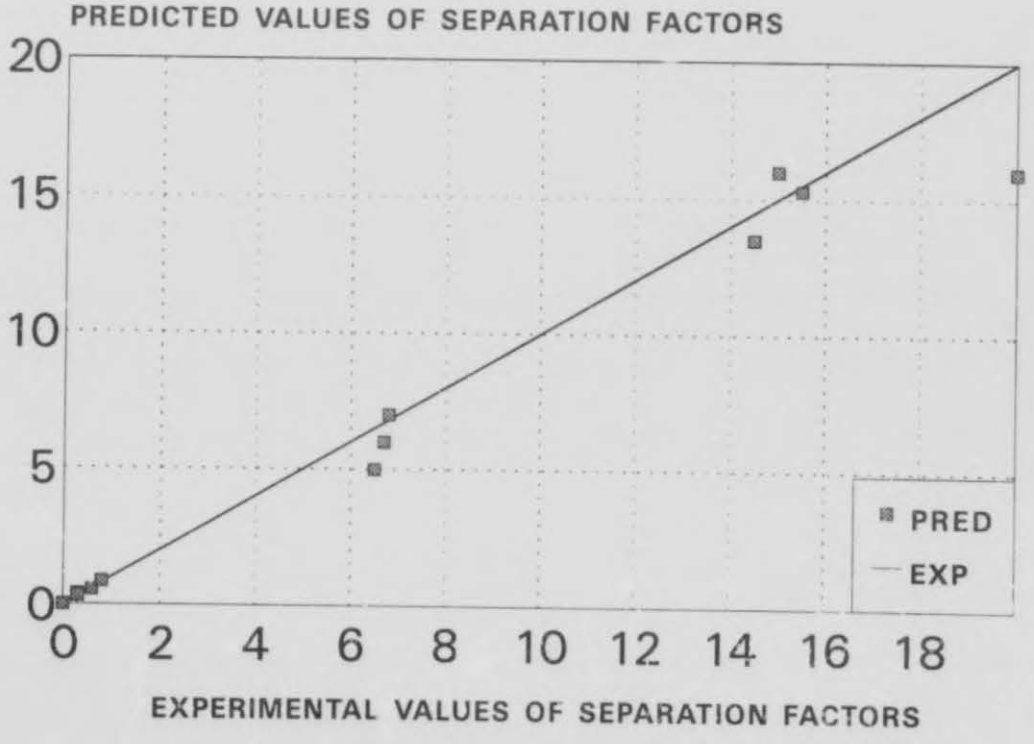


FIGURE 5.18 COMPARISON OF SIMULATED SEPARATION FACTORS WITH EXPERIMENTAL DATA



CHAPTER 6

Conclusions

In this dissertation the use of connectionist systems or neural nets for the modelling of steady state processes has been explored. Attention was given to the processing of plant data prior to modelling, i.e. the estimation of variance-covariance matrices associated with measurements the detection and isolation of gross errors in the plant data, the reconciliation or filtering of the data after elimination of gross errors and finally the construction of plant models based on the use of plant data preprocessed by the foregoing techniques.

The variance-covariance matrices of process variables are generally unknown and often difficult or expensive to measure. Under these circumstances estimates of these parameters can be made with the use of neural nets, as was demonstrated in chapter 2. The methods depend on the modelling of the relationship between the variances and covariances of the residuals of the process constraints and the corresponding variances and covariances of the measurements of the process variables. Since this relationship is not unique, additional information is required to estimate the variances and covariances of the variable measurements. This information can be integrated with a neural net either through modification of the topology of the net, or as distinct patterns in the training examples presented to the net, as was shown in chapter 2.

The detection and diagnosis of faults in complex process plants are one of the most important aspects of the monitoring and control of such plants. Existing methods are usually based on the analytical redundancy afforded by a mathematical model of the system and make use of statistical tests to detect and isolate gross errors in the plant data. These tests depend on the distributions of the residuals of the process variable

measurements, as well as the residuals of the constraints of the process model. Since these distributions are generally unknown and especially difficult to estimate in non-linear systems, standard gross error detection strategies are ineffective for all but simple (linear) process systems.

By making use of neural nets the ill-defined relationship between the residuals of the variable measurements and the residuals of the process constraints can be modelled accurately, regardless of the (non-)linearity of the system and can be used effectively to detect and isolate systematic errors, as was shown in chapter 3. In contrast with conventional statistical methods, explicit knowledge of random error distributions is not required, since the neural net can construct this distribution directly from process data. These techniques are not limited to the process engineering industry, but could find wide application in many other areas dealing with complex technological systems, such as aviation, electronic engineering, aerospace, etc.

In chapter 4 the exploitation of the supercomputing potential of neural nets for the reconciliation of inconsistent process data was investigated. The reconciliation of process data which characterize the behaviour of large or complex plants generally constitutes a large computational burden which can be alleviated considerably by more efficient computing techniques or devices. These types of problems lend themselves well to connectionist computing devices or neurocomputers which consist of large numbers of primitive processing units. In chapter 4 it was shown that by assigning a process unit in the neurocomputer to each of the process variables that has to be reconciled significant gains in computational efficiency can be attained. The use of such systems is particularly attractive for large systems. The performance of these systems is furthermore particularly sensitive to the degree of parallelism and although not explored in any depth, it was shown that an increase in the parallelism of the system could result in considerable enhancement in the performance of the computational device.

In chapter 5 the construction of plant models based on neural nets was discussed. The dissolved and undissolved gold losses in a gold reduction plant, the consumption of an additive in a gold leach plant, as well as the behaviour of a phosphate flotation plant were predicted satisfactorily by

means of neural net models. These models were also compared with the models in use on the plants and found to be more accurate.

The optimization of separation circuits or plants can often be accomplished by partitioning the circuit into a linear(ized) and a non-linear subsystem, which can be optimized with a procedure based on the iterative use of linear programming techniques. The non-linear subsystem which is generally ill-defined can be represented with a neural net, which can be used in a modular fashion with the linear programming superstructure, as was demonstrated with a gravity concentration circuit. The modular nature of these strategies renders them more useful, since the behaviour of specific types of separators can be modelled independently in terms of a generalized connectionist data base that can be used in linear programming optimizers and simulation routines.

SYMBOLS

A	incidence matrix of a process circuit
$\mathbf{A}^*$	recycle or loop matrix with elements $\{a_{i,j}^*\}$
$a_{i,j}^*$	elements of loop matrix
$a_k^\#$	constant input current to amplifiers in objective function section of linear programming neural net
b	bias vector of gross error $\{b_i\}$
$b_k^\#$	constant input current to amplifiers in constraint section of linear programming neural net
C	matrix of process constraint coefficients
$c_k^\#$	input capacitance of k'th amplifier in a neurocomputer
cov(·)	covariance
d(·)	set of process constraints
$\mathbf{D}^*$	Cartesian product of the domains of variables
$\mathbf{D}_j$	j' th column vector in transposed loop matrix $\mathbf{A}^*$
$d_k\#$	the network output corresponding to field element $f_k\#$
e	errors in measurements of process variables $\{e_i\}$
E	scalar energy or cost function serving as a measure of the overall state of a feedback neural net
E(·)	expected value
E_0	initial energy of a neural net system
E_f	energy of a stable neural net system
$\mathbf{e}_f$	random error in measured composition **f** of process stream **F**
$\mathbf{e}_F$	random error in measured process flow streams **F**
$\mathbf{e}_x$	random error in variable **x**
F	flow rates of process streams $\{F_i\}$
f	mass fractions $\{f_{i,j}\}$ of components in flow stream **F**

$\mathbf{f(x)}$	an arbitrary set of functions of a set of variables **x**
$F^{a}_{j,k}$	concentrate recovery of element *k* from concentrator *j*
$F^{b}_{j,k}$	tailings recovery of element *k* from concentrator *j*
$F^{c}_{i,k}$	flow rate of element *k* of concentrate stream from gravity concentrator *i*
F^{DS}_{i}	dry solids flow rate from concentrator *i*
$F^{f}_{i,k}$	flow rate of element *k* of feed stream to gravity concentrator *i*
$f_i\#$	fields in training or test vectors in general
$f_{i,j}$	mass fraction of component j in flow stream F_i
$f_{i,j}''$	adjusted mass fraction of component *j* in flow stream F_i
$f_{i,j}'$	measured mass fraction of component *j* in flow stream F_i
f_j	j'th field element in input vector to neural net
$F^{m}_{i,k}$	flow rate of element *k* of middlings stream from gravity concentrator *i*
F_{OBJ}	an objective function
$F^{t}_{i,k}$	flow rate element *k* of tailings stream from gravity concentrator *i*
F^{TOT}	total flow rate
Δg	incremental change in g between time i and i-1
$\mathbf{G}(\cdot)$	set of (in)equality constraints
$g(\cdot)$	transfer function of neural net process element
$g_c(\cdot)$	transfer function in constraint section of linear programming neural net
$g_k\#$	descaled output of k'th element of neural network
$g_v(\cdot)$	transfer function in objective function section of linear programming neural net
h	step length in numerical integration routine
$\mathbf{H}(\cdot)$	set of equality constraints (design specification constraints)
H_i	approximation to inverse Jacobian $J^{-1}(y)$
$i_j\#$	j'th network input

J	Jacobian
ΔJ^{-1}	incremental change in J^{-1} between time i and i-1
k	a factor associated with the noisiness of neural net training data
$\mathbf{k}$	set of constant parameters
$\mathbf{k}_1$, $\mathbf{k}_2$	coefficient vectors of linear regression models $\mathbf{Y} = \mathbf{X}.\mathbf{k}_1 + \mathbf{k}_2$
l_j	j'th cycle in a process circuit
$\mathbf{M}_i$	block diagonal, block triangular and border matrices
M_k	maximum value that an element in a field can assume
m_k	minimum value that an element in a field can assume
N_h	number of processing elements in the hidden layer(s) of a neural net
$o_k\#$	actual output of k'th element of neural network
$\mathbf{p}$	internal parameters in a sequential modular simulator
P	the number of processors or processing elements in a parallel computational structure
$p^{\#}(s_j)$	weighting factor allocated to stream s_j
p_c	critical value of standard normal deviate
p_j	j'th standard normal deviate
$q_{j,i}$	ratio defined as $q_{j,i} = s_{j,i}/(s_{j,i-1})$
$q_{j,max}$	upper bound of $q_{j,i}$
$q_{j,min}$	lower bound of $q_{j,i}$
R	a bounded state variable domain
$\mathbf{R}$	a set of bounded state variable domains
$\mathbf{r}$	vector of process constraint residuals resulting from errors in measurements of process variables $\{r_p\}$
$\mathbf{R}(\cdot)$	set of equality constraints (flowsheet equations)
$r_D\#$	lower limit of range allowed for output of neural net
$R_D\#$	upper limit of range allowed for output of neural net
$r_I\#$	lower limit of range allowed for input of neural net
$R_I\#$	upper limit of range allowed for input of neural net

R^j_i	search range associated with variable *i* at time *j*
$r_k^{\#}$	resistance of k'th amplifier in a neurocomputer
rm_{ji}^k	recycle of element *k* in tailings stream from concentrator *j* to *i*
ry_{ji}^k	recycle of element *k* in concentrate stream from concentrator *j* to *i*
rz_{ji}^k	recycle of element *k* in middlings stream from concentrator *j* to *i*
S	speedup factor
s_j	j'th stream in a generic process circuit
$s_{j,i}$	gradient function defined as $[f_{j,i}(x_i) - f_{j,i-1}(x_{i-1})]/[x_{j,i} - x_{j,i-1}]$
t	time
Δt	time increment, equivalent to an iteration step in terms of the computational algorithm
t_j	time at which observation j of a state variable takes place
T_k	k'th training vector of a neural net
$T_{o,j}$	desired value of j'th output of a neural net
$u_i(t)$	the potential of processing element *i* at time *t*, i.e. the sum of the products of the weights and outputs of all other elements feeding into the processing element
$\mathbf{v}$	set of output states of computational elements in neural net $\{v_j\}$
$\mathbf{V}_{\mid d}$	weights associated with the material balance constraints incorporated in the energy function *E*
$\mathbf{V}_{\mid e}$	variance-covariance matrix of measurement residuals **e**
$\mathbf{V}_{\mid r}$	variance-covariance matrix of constraint residuals **r**
$\mathbf{V}_{\mid x}$	variance-covariance matrix of the process variable **x**
$\mathbf{v}^0$	initial output states of computational elements in the neural net
$var(\cdot)$	variance
$v_{h(i),j}$	j'th processing element in the (i'th) hidden layer of a neural net

$v_{i,j}$	j'th node in input layer of neural net
$v_{o,k}$	k'th node in output layer of neural net
$w_{i,j}$	connection strength or weight between process units *i* and *j* in neural net
X	(n x m) matrix of observations of m independent variables $x_{i,k}$
x	vector of independent variables $[x_1, x_2, .. x_p]$ (true values)
x + e	uncorrupted measurement; unbiased error
x"	adjusted values of process variables $\{x_i"\}$
x'	measured values of process variables $\{x_i'\}$
$x_{i(,j)}$	j'th observation of variable x_i
$x_{i,AVG}$	average value of variable x_i
X_j	j'th state variable
Y	(n x p) matrix of observations of p dependent variables $y_{i,k}$
Δy	incremental change in y between time i and i-1
y	vector of dependent variables $[y_1, y_2, .. y_m]$ (true values)
$y_{i(,k)}$	(k'th observation of the) j'th dependent variable
z	vector of decision variables
z_{max}	upper bound of vector of decision variables **z**
z_{min}	lower bound of vector of decision variables **z**

Greek letters

α	probability of the occurrence of a type I error
α_i	domain of i'th variable in a set
ß	probability of the occurrence of a type II error
$\Gamma^{c,t,L}{}_{i,k}$	lower limit of concentrate-tailings separation factor $\Gamma^{c,t}{}_{i,k}$
$\Gamma^{c,t,U}{}_{i,k}$	upper limit of concentrate-tailings separation factor $\Gamma^{c,t}{}_{i,k}$

$\Gamma^{c,t}_{i,k}$	concentrate-tailings separation factor for element k in concentrator i
$\Gamma^{m,t,L}_{i,k}$	lower limit of middlings-tailings separation factor $\Gamma^{m,t}_{i,k}$
$\Gamma^{m,t,U}_{i,k}$	upper limit of middlings-tailings separation factor $\Gamma^{m,t}_{i,k}$
$\Gamma^{m,t}_{i,k}$	middlings-tailings separation factor for element k in concentrator j
δ	an arbitrary small value
ϵ	convergence criterion
ϕ^j_i	random number associated with variable i at time j ($0 \leq \phi^j_i \leq 1$)
$\mu_{j,j}$	variance of multivariate normal deviate p_j
Φ_j	feed grade
Θ_i	bias of process element i in a neural net
τ	learning rate
Ω	the fraction of computer code that can be executed in parallel

Subscripts

g	gangue
h	an only hidden layer
h1	first hidden layer
h2	second hidden layer
i	input
o	output
v	valuable element
w	water

REFERENCES

ABE, S., KAWAKAMI, J. & HIRASAWA, K. 1992. Solving inequality constrained combinatorial optimization problems by the Hopfield neural networks. *Neural Networks*, vol 5, no 4, pp 663-670.

ABU-MOSTAFA, Y.S. & PSALTIS, D. 1987. Optical neurocomputers. *Scientific American*, vol 256, no 3, pp 88-95.

ALDRICH, C. & VAN DEVENTER, J.S.J. 1993. The use of neural nets to detect systematic errors in process systems. *International Journal of Mineral Processing* (to be published).

ALLOTT, K. 1991. Untapped expertise?. *Process Engineering*, vol 72, no 3, pp 30-31.

ALMASY, G.A. & MAH, R.S.H. 1984. Estimation of measurement error variances from process data. *Industrial & Engineering Chemical Process Design & Development*, vol 23, no 4, pp 779-784.

ANTHONY, K.R., VAN DEVENTER, J.S.J. & REUTER, M.A. 1991. Steady-state simulation and optimization of gravity separation circuits by use of linear programming and expert systems. *Minerals Engineering*, vol 24, no 3/4, pp 311-327.

BANG, W.L. & BING J.S. 1991. Modified Hopfield neural networks for retrieving the optimal solution. *IEEE Transactions on Neural Networks*, vol 2, no 1, pp 137-142.

BARNWELL, J. & ERTL, B. 1987. Expert systems and the chemical engineer. *The Chemical Engineer*, no 440, pp 41-43.

BATTITI, R. 1992. First and second order methods for learning: Between steepest descent and Newton methods (Review). *Neural Computation*, vol 4 no 2, pp 141-166.

BERARDINIS, L.A. 1992. Clear thinking on fuzzy logic. *Machine Design*, vol 64, no 8, pp 46-53.

BEST, R.J. 1990. Process simulation using parallel computers. *Transactions of the Institution of Chemical Engineering*, vol 68, part A, pp 419-429.

BHAT, N.V. & McAVOY, T.J. 1990. Use of neural nets for dynamic modelling and control of chemical process systems. *Computers & Chemical Engineering*, vol 14, no 4/5, pp 573-583.

BHAT, N.V., MINDERMAN, P.A. Jr, McAVOY, T.J. & WANG, N.S. 1990. Modeling chemical process systems via neural computation. *IEEE Control Systems Magazine*, vol 10, no 3, pp 24-30.

BIEGLER, L.T. & HUGHES, R.R. 1982. Infeasible path optimization with sequential modular simulators. *AIChE Journal*, vol 28, no 6, pp 994-1002.

BIEGLER, L.T. & HUGHES, R.R. 1985. Feasible path optimization with sequential modular simulators. *Computers & Chemical Engineering*, vol 9, no 4, pp 379-394.

BIEGLER, L.T. 1989. Chemical process simulation. *Chemical Engineering Progress*, vol 85, no 10, pp 50-61.

BRADLEY, D. 1993. Will future computers be all wet? *Science*, vol 259, no 5097, pp 890-892.

BRANNOCK, N.F., VERNEUIL, V.S. & WANG, Y.L. 1979. PROCESS^{SM} simulation program a comprehensive flowsheeting tool for chemical engineers. *Computers and Chemical Engineering*, vol 3, Paper 6A.4, pp 329-352.

BRITTON, M.I. & VAN VUUREN, E.J.J. 1973. Computer analysis, modelling and optimisation of gold recovery plants in the Anglo-American group. *Journal of the South African Institute of Mining and Metallurgy*, vol 73, no 7, pp 211-222.

BROYDEN, C.G. 1965. A class of methods for solving non-linear simultaneous equations. *Mathematical Computation*, vol 19, p 577.

CANHAM, L. 1993. A glowing future for silicon. *New Scientist*, vol 138, no 1868, pp 23-27.

CHAN, W.K. & PRINCE, R.G.H. 1986. Application of the chain rule of differentiation to sequential modular flowsheet optimization. *Computers & Chemical Engineering*, vol 10, no 3, pp 223-240.

CHIU, C., MAA, C.-Y. & SHANBLATT, M.A. 1991. Energy function analysis of dynamic programming neural networks. *IEEE Transactions on Neural Networks*, vol 2, no 4, pp 418-425.

CICHOCKI, A. & UNBEHAUEN, R. 1992. Neural networks for solving systems of linear equations and related problems. *IEEE Transactions on circuits and systems - I: Fundamental theory and applications*, vol 39, no 2, pp 124-138.

CLARK, S.M. & REKLAITIS, G.V. 1984. Investigation of strategies for executing sequential modular simulations. *Computers and Chemical Engineering*, vol 8, no 3/4, pp 205-218.

CORDOBA, J.F. 1988. A linear algorithm for nonredundant decompositions. *Computers & Chemical Engineering*, vol 12, no 1, pp 105-107.

CROWE, C.M. 1989. Test of maximum-power for detection of gross errors in process constraints. *AIChE Journal*, vol 35, no 5, pp 869-872.

CROWE, C.M., CAMPOS, Y.A.G. & HRYMAK, A. 1983. Reconciliation of process flow rates by matrix projection. Part I: Linear case. *AIChE Journal*, vol 29, no 6, pp 881-888.

CROWE, C.M., CAMPOS, Y.A.G. & HRYMAK, A. 1986. Reconciliation of process flow rates by matrix projection. Part II: The nonlinear case. *AIChE Journal*, vol 32, no 4, pp 616-623.

CUTTING, G.W. 1976. Estimation of interlocking mass balances on complex mineral beneficiation plants. *International Journal of Mineral Processing*, vol 3, pp 207-218.

DAMBROT, S.M. 13 July 1992. Neural networks challenge fuzzy logic's reign. *Electronics*, p 26.

DEL CORSO, D., GROSSPIETSCH, K.E. & TRELEAVEN, P. 1989. European approaches to VLSI neural networks. *IEEE Micro*, vol 9, no 6, pp 5-7.

DIWEKAR, U.M., GROSSMAN, I.E. & RUBIN, E.S. 1992. An MINLP process synthesizer for a sequential modular simulator. *Industrial Engineering & Chemical Research*, vol 31, no 1, pp 313-322.

EBERHART, R.C. & DOBBINS, R.W. 1990. Early neural network development history: The age of Camelot. *IEEE Engineering in Medicine and Biology*, vol 9, no 3, pp 15-18.

EVANS, L.B. 1987. Computer-aided design (CAD): Advances in process flowsheeting systems. In: *Recent developments in chemical process and plant design*. Edited by Liu, Y.A., McGee (jr), H. & Epperly, W.R., John Wiley and Sons, 1987, pp 269-276.

EVANS, L.B., BOSTON, J.F., BRITT, H.I., GALLIER, P.W., GUPTA, P.K., JOSEPH, B., MAHALEC, V., NG, E., SEIDER, W.D. & YAGI, H. 1979. ASPEN: An advanced system for process engineering. *Computer applications in Chemical Engineering - Proceedings of the 12th European Symposium, 8-11 April 1979, Paper 6A.3,* In: Computers & Chemical Engineering, vol 3, pp 319-327.

FAN, J.Y., NIKOLAOU, M. & WHITE, R.E. 1993. An approach to fault diagnosis of chemical processes via neural networks. *AIChE Journal*, vol 39, no 1, pp 82-88.

FOUCHY, K. 1991. Process simulation gains a new dimension. *Chemical Engineering*, vol 98, no 10, pp 47,49,51,53.

FOURIE, J.H. 1981. Plant practice in the flotation of phosphate. *Journal of the South African Institute of Mining and Metallurgy*, vol 81, no 4, pp 101-108.

GARDNER, W.D. 1990. Neural nets get practical. *High Performance Systems*, pp 68-72. In: ADVANCES IN COMPUTERS, editor Yovitz, M.C. vol 34, Academic Press, San Diego, CA (1992).

GEAK, E. 1993. Computer learns to handle fusion reactor. *New Scientist*, vol 138, no 1879, p 5.

GORCZYNSKI, E.W., HUTCHISON, H.P. & WAJIH, A.R.M. 1979. Development of a modularly organised equation-oriented process simulator. *Computers and Chemical Engineering*, vol 3, Paper 6A.5, pp 353-356.

GORSEK, A., GLAVIC, P. & SENCAR, P. 1992. Optimal process design for specialty products. *European Symposium on Computer Aided Process Engineering - 1*, 24-28 May 1992, Elsinore, Denmark, pp S321-S328.

GOSER, K., HILLERINGMANN, U., RUECKERT, U. & SCHUMACHER, K. 1989. VLSI technologies for artificial neural networks. *IEEE Micro*, vol 9, no 6, pp 28-44.

GRAELLS, M., ESPUÑA, A. & PUIGJANER, L. 1992. Optimization of process operations in the leather industry. *European Symposium on Computer Aided Process Engineering - 1*, 24-28 May 1992, Elsinore, Denmark, pp S221-S228.

HARRISON, B.K. 1992. Computational inefficiencies in sequential modular flowsheeting. *Computers & Chemical Engineering*, vol 16, no 7, pp 637-639.

HECHT-NIELSEN, R. 1988. Neurocomputing: Picking the human brain. *IEEE Spectrum*, vol 25, no 3, pp 36-41.

HECHT-NIELSEN, R. 1990. *NEUROCOMPUTING*. Addison-Wesley Publishing Company.

HEUCKROTH, M.W., GADDY, J.L. & GAINES, A.D. 1976. *AIChE Journal*, vol 22, no 4, pp 744-750.

HILLESTADT, M. & HERZBERG, T. 1986. Dynamic simulation of chemical engineering systems by the sequential modular approach. *Computers & Chemical Engineering*, vol 10, no 4, pp 377-388.

HIMMELBLAU, D.M. & BISCHOFF, K.B. 1968. *Process analysis and simulation (Deterministic systems)*. Part III, Systems analysis, pp 231-279, John Wiley & sons, Inc., N.Y.

HINTON, G.E. 1989. Connectionist learning procedures. *Artificial Intelligence*, vol 40, no 1, pp 185-234.

HLAVACEK, V. 1977. JOURNAL REVIEW: Analysis of a complex plant - steady state and transient behavior. *Computers and Chemical Engineering*, vol 1, no 1, pp 75-100.

HLAVACEK, V. 1983. *Proceedings of the Second International Conference on FOUNDATIONS OF COMPUTER-AIDED PROCESS DESIGN*, Snowmass, Colorado, June 19-24, pp 417-420.

HODOUIN, D. & EVERELL, M.D. 1980. A hierarchical procedure for adjustment and material balancing of mineral processes data. *International Journal of Mineral Processing*, vol 7, pp 91-116.

HODOUIN, D. & VAZ COELHO, S.V. 1987. Mass balance calculations around mineral processing units using composition analyses within particle size classes. *International Journal of Mineral Processing*. vol 21, pp 65-82.

HODOUIN, D., ALLIOT, N. & FLAMENT, F. 1991. Redundancy analysis of complex sets of mineral processing data for mass balance computation. *International Journal of Mineral Processing*, vol 32, pp 213-231.

HODOUIN, D., GARON, M., RéMILLARD, M. & THéRIEN, M. 1988. Assessment of precious metal distribution in Lac Mattagami flotation

plant by computer mass balance calculation. *CIM Bulletin*, vol 81, no 919, pp 62-69.

HODOUIN, D., GELPE, T. & EVERELL, M.D. 1982. Sensitivity analysis of material balance calculations - an application to a cement clinker grinding process. *Powder Technology*, vol 32, pp 139-153.

HOFFMAN, H. 1988. Future trends in chemical engineering. *Computers & Chemical Engineering*, vol 12, no 5, pp 415-420.

HOLLY, W., COOK, R. & CROWE, C.M. 1989. Reconciliation of mass flow rate measurements in a chemical extraction plant. *The Canadian Journal of Chemical Engineering*, vol 67, no 4, pp 595-601.

HOPFIELD, J.J. & TANK, D.W. 1985. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, vol 52, pp 141-152.

HOPFIELD, J.J. & TANK, D.W. 1986. Computing with neural circuits: A model. *Science*, vol 233, no 4764, pp 625-633.

HOPFIELD, J.J. 1984. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Science*, USA, vol 81, pp 3088-3092.

HORNIK, K., STINCHCOMBE, M. & WHITE, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, vol 2, pp 359-366.

HOSKINS, J.C, KALIYUR, K.M. & HIMMELBLAU, D.M. 1991. Fault diagnosis in complex chemical plants using artificial neural networks. *AIChE Journal*, vol 37, no 1, pp 137-141.

HUNT, K.J. & SBARBARO, D. 1991. Neural networks in process control. *Process Engineering*, vol 72, no 5, pp 59-63.

HUNT, K.J., SBARBARO, D., ZBIKOWSKI, R. & GAWTHROP, P.J. 1992. Neural networks for control systems - a survey. *Automatica*, vol 28, no 6, pp 1083-1112.

HUSH, D.R. & HORNE, B.G. 1993. Progress in supervised neural networks - what's new since Lippmann? *IEEE Signal Processing Magazine*, vol 10, no 1, pp 8-39.

HUSH, D.R., HORNE, B.G. & SALAS, J.M. 1992. Error surfaces for multilayer perceptrons. *IEEE Transactions on Systems, Man, and Cybernetics*, vol 22, no 5, pp 1152-1161.

HUTCHISON, H.P., JACKSON, D.J. & MORTON, W. 1986a. The development of an equation-oriented flowsheet simulation and optimization package - I. The QUASILIN program. *Computers & Chemical Engineering*, vol 10, no 1, pp 19-29.

HUTCHISON, H.P., JACKSON, D.J. & MORTON, W. 1986b. The development of an equation-oriented flowsheet simulation and optimization package - II. Examples and results. *Computers & Chemical Engineering*, vol 10, no 1, pp 31-47.

IORDACHE, C., MAH, R. S. H. & TAMHANE, A. C. 1985. Performance studies of the measurement test for detection of gross errors in process data. *AIChE Journal*, vol 31, no 7, pp 1187-1201.

JEFFREY, W. & ROSNER, R. 1986. Optimization algorithms: Simulated annealing and neural network processing. *Astrophysical Journal*, vol 310, pp 473-481.

JONGELEN, E.M., DEN HEIJER, C. & VAN ZEE, G.A. 1988. Detection of gross errors in process data using studentized residuals. *Computers & Chemical Engineering*, vol 12, no 8, pp 845-847.

JOWETT, A. & SUTHERLAND, D.N. 1985. Some theoretical aspects of optimizing complex mineral separation systems. *International Journal of Mineral Processing*, vol 14, pp 85-109.

KAHANER, D.K. 1991. Advances in fuzzy theory and applications. *IEEE Micro*, vol 11, no 4, pp 8-11.

KAMGAR-PARSI, B. & KAMGAR-PARSI, B. 1990. On problem solving with Hopfield neural networks. *Biological Cybernetics*, vol 62, pp 415-423.

KARIM, M.N. & RIVIERA, S.L. 1992. Comparison of feed-forward and recurrent neural networks for bioprocess state estimation. *European Symposium on Computer Aided Process Engineering - 1*, 24-28 May 1992, Elsinore, Denmark, pp S369-S377.

KELLER, J. Y., ZASADZINSKI, M. & DAROUACH, M. 1992. Analytical estimator of measurement error variances in data reconciliation. *Computers & Chemical Engineering*, vol 16, no 3, pp 185-188.

KENNEDY, M.P. & CHUA, L.O. 1986. Unifying the Tank and Hopfield linear programming circuit and the canonical nonlinear programming circuit of Chua and Lin. *IEEE Transactions on Circuits and Systems*, vol CAS-34, no 2, pp 210-214.

KENNEDY, M.P. & CHUA, L.O. 1988. Neural networks for nonlinear programming. *IEEE Transactions on Circuits And Systems*, vol 35, no 5, pp 554-562.

KIM, I.-W., LIEBMAN, M. J. & EDGAR, T. F. 1990. Robust error-in-variables estimation using nonlinear programming techniques. *AIChE Journal*, vol 36, no 7, pp 985-993.

KIRK, S., GELATT, C.D. (jnr) & VECCHI, M.P. 1983. Optimization by simulated annealing. *Science*, vol 220, no 4598, pp 671-680.

KIRKPATRICK, S. 1992. A volatility measure for annealing in feedback neural networks. *Neural Computation*, vol 4, no 1, pp 191-195.

KISALA, T.P., TREVINO-LOZANO, R.A., BOSTON, J.F. & BRITT, H.I. 1987. Sequential modular and simultaneous modular strategies for flowsheet optimization. *Computers & Chemical Engineering*, vol 11, no 6, pp 567-579.

KITO, S., HATTORI, T. & MURAKAMI, Y. 1992. Estimation of acid strength of mixed oxides by neural network. *Industrial Engineering Chemistry Research*, vol 31, no 3, pp 979-981.

KLIMASAUSKAS, C.C. 1992. Neural networks: An engineering perspective. *IEEE Communications Magazine*, vol 30, no 9, pp 50-53.

KNEPPER, J. C. & GORMAN, J. W. 1980. Statistical analysis of constrained data sets. *AIChE Journal*, vol 26, no 2, pp 260-264.

KRAMER, M.A. 1992. Associative neural networks. *Computers & Chemical Engineering*, vol 16, no 4, pp 313-328.

KRAMER. M.A. & LEONARD, J.A. 1990. Diagnosis using backpropagation neural networks - analysis and criticism. *Computers & Chemical Engineering*, vol 14, no 12, pp 1323-1338.

KRASLAWSKI, A., KOIRANEN, T. NYSTRöM, L. & GOURDON, C. 1992. Concurrent engineering: Fuzzy simulation and similarity in quality loss function deployment and applications. *European Symposium on Computer Aided Process Engineering - 1*, 24-28 May 1992, Elsinore, Denmark, pp S321-S328.

LAPLANTE, A.R. & SHU, Y. 1988. Simulating gravity circuits: The compromise between accuracy and simplicity. *Proceedings of the International Symposium on Computer Software in Chemical and Extractive Metallurgy*, Montreal, Canada, Aug 28-31, pp 137-152.

LAU, K.H. & ULRICHSON, D.L. 1992. Effect of local constraints on the convergence behaviour of sequential modular simulators. *Computers & Chemical Engineering*, vol 16, no 9, pp 887-892.

LEDET, W.P. & HIMMELBLAU, D.M. 1970. Decomposition procedures for the solving of large scale systems, from *ADVANCES IN CHEMICAL ENGINEERING*, vol 8, Academic Press, eds Drew, B.T., Cokelet, G.R., Hoopes, J.W., Jnr & Vermeulen, T.

LEONARD, J. & KRAMER, M. A. 1990. Improvement of the backpropagation algorithm for training neural networks. *Computers & Chemical Engineering*, vol 14, no 3, pp 337-341.

LEONE, H.P., MELLI, T.R., MONTAGNA, J.M., VECCHIETTI, A.R., & CERRO, R.L. 1987. SIMBAD: a process simulator linked to a DBMS - 3. The physicochemical properties package. *Computers & Chemical Engineering*, vol 11, no 1, pp 217-226.

LIN, T.D. & MAH, R.S.H. 1978. A sparse computation system for process design and simulation: Part I. Data structures and processing techniques. *AIChE Journal*, vol 24, no 5, pp 830-839.

LIPPMANN, R.P. 1989. Pattern classification using neural networks. *IEEE Communications Magazine*, vol 27, no 11, pp 47-64.

LIPPMANN, R.P. 1987. An introduction to computing with neural nets and data parallel algorithms for massively parallel computing. *IEEE ASSP Magazine*, vol 35, pp 4-21.

LOURI, A. 1991. Three-dimensional optical architecture. *IEEE Micro*, vol 11, no 2, pp 24-27.

LOVEDAY, B.K. & MARCHANT, G.R. 1972. Simulation of multicomponent flotation plants. *Journal of the South African Institute of Mining and Metallurgy*, vol 72, no 11, pp 288-294.

LUCAS, S., ZHAO, Z. CAWLEY, G. & NOAKES, P. 1993. Pattern recognition with the decomposed multilayer perceptron. *Electronics Letters*, vol 29, no 5, pp 442-443.

LUPO, J.C. 1989. Defense applications of neural networks. *IEEE Communications Magazine*, vol 27, no 11, pp 82-88.

LUUS, R. & JAAKOLA, R. 1973. Optimization by direct search and systematic reduction of the size of the search region. *AIChE Journal*, vol 19, no 4, pp 760-766.

LUUS, R. & WANG, B.-C. 1978. Reliability of optimization procedures for obtaining global optimum. *AIChE Journal*, vol 24, no 4, pp 619-626.

MACKAY, J.D. & LLOYD, P.J.D. 1975. Progress in assessing plant operating data. *Journal of the South African Institute of Mining and Metallurgy*, vol 76, no 3, pp 162-170.

MADRON, F. 1985. A new approach to the identification of gross errors in chemical engineering measurements. *Chemical Engineering Science*, vol 40, no 10, pp 1855-1860.

MADRON, F., VEVERKA, V. & VANECEK, V. 1977. Statistical analysis of material balance of a chemical reactor. *AIChE Journal*, vol 23, no 4, pp 482-486.

MAH, R. S. H. & TAMHANE, A. C. 1982. Detection of gross errors in process data. *AIChE Journal*, vol 28, no 5, pp 828-830.

MAH, R.S.H. 1974. A constructive algorithm for computing the reachability matrix. *AIChE Journal*, vol 20, no 6, pp 1227-1228.

MAH, R.S.H. 1983. Application of graph theory to process design and analysis. *Computers & Chemical Engineering*, vol 7, no 4, pp 239-257.

MAHALEC, V., KLUZIK, H. & EVANS, L.B. 1979. Simultaneous modular algorithm for steady state flowsheet simulation and design. *Computers and Chemical Engineering*, vol 3, Paper 6B.4, pp 373-381.

MASSON, E. & WANG, Y.-J. 1990. Introduction to computation and learning in artificial neural networks. *European Journal of Operational Research*, vol 47, no 1, pp 1-28.

McLANE, M., SOOD, M.K. & REKLAITIS, G.V. 1979. A hierarchical strategy for large scale process calculations. *Computers and Chemical Engineering*, vol 3, Paper 6B.3, pp 383-394.

MINSKY, M. 1993. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine*, vol 12, no 2, pp 35-51.

MONTAGNA, J.M., LEONE, H.P., MELLI, T.R., VECCHIETTI, A.R. & CERRO, R.L. 1987. SIMBAD: a process simulator linked to a DBMS - 1. The executive system. *Computers & Chemical Engineering*, vol 11, no 1, pp 63-71.

MOTARD, R.L. & WESTERBERG, A.W. 1981. Exclusive tear sets for flowsheets. *AIChE Journal*, vol 27, no 5, pp 725-732.

MULAR, A., BRADBURN, R., FLINTOFF, B. & LARSEN, C. 1976. Mass balance of a grinding circuit. *CIM Bulletin*, vol 69, pp 124-129.

MURRAY, A.F. 1989. Pulse arithmetic in VLSI neural networks. *IEEE Micro*, vol 9, no 6, pp 64-73.

MURTHY, C.L.N. & HUSAIN, A. 1983. An efficient tearing algorithm based on minimum sum of weights. *Computers & Chemical Engineering*, vol 7, no 2, pp 133-136.

NAIDU, S.R., ZAFIRIOU E. & McAVOY, T.J. 1990. Use of neural networks for sensor failure detection in a control system. *IEEE Control Systems Magazine*, vol 10, no 3, pp 49-55.

NARASIMHAN, S. & MAH, R.S.H. 1987. Generalized likelihood ratio method for gross error identification. *AIChE Journal*, vol 33, no 9, pp 1514-1521.

NARASIMHAN, S. & MAH, R.S.H. 1989. Treatment of general steady state process models in gross error identification. *Computers & Chemical Engineering*, vol 13, no 7, pp 851-853.

NARASIMHAN, S., MAH, R.S.H., TAMHANE, A.C., WOODWARD, J.W. & HALE, J.C. 1986. A composite statistical test for detecting changes of steady states. *AIChE Journal*, vol 32, no 9, pp 1409-1418.

NASRABADI, N.M. & CHOO, C.Y. 1992. Hopfield network for stereo vision correspondence. *IEEE Transactions on Neural Networks*, vol 3, no 1, pp 5-13.

PAI, C.C.D. & FISHER, G.D. 1988. Application of Broyden's method to reconciliation of nonlinearly constrained data. *AIChE Journal*, vol 34, no 5, pp 873-876.

PERKINS, J.D. 1983. Equation-oriented flowsheeting. *Proceedings of the Second International Conference on FOUNDATIONS OF COMPUTER-AIDED PROCESS DESIGN*, Snowmass, Colorado, June 19-24, pp 309-367.

PERREGAARD, J. & SORENSEN, E.L. 1992. Simulation and optimization of chemical processes: numerical and computational aspects.

European Symposium on Computer Aided Process Engineering - 1, 24-28 May 1992, Elsinore, Denmark, pp S247-S254.

PHO, T.K. & LAPIDUS, L. 1973. Topics in computer-aided design: Part I. Optimum tearing algorithm for recycle streams. *AIChE Journal*, vol 19, no 6, pp 1170-1181.

PSICHOGIOS, D.C. & UNGAR, L.H. 1991. Direct and indirect model based control using neural networks. *Industrial and Engineering Chemistry Research*, vol 30, no 12, pp 2564-2573.

QIN, S.J. & McAVOY, T.J. 1992. Nonlinear PLS modelling using neural networks. *Computers & Chemical Engineering*, vol 16, no 4, pp 379-391.

RAGOT, J., AITOUCHE, A., KRATZ, F. & MAQUIN, D. 1991. Detection and location of gross errors in instruments using parity space techniques. *International Journal of Mineral Processing*, vol 31, pp 281-299.

RAJNIAK, P., KRALIK, M. & ILAVSKY, J. 1992. A steady state equation oriented simulator for separation processes of solution with weak electrolytes. *European Symposium on Computer Aided Process Engineering - 1*, 24-28 May 1992, Elsinore, Denmark, pp S457 - S464.

RANGAIAH, G.P. 1985. Effect of dimension on direct search methods for constrained optimization. *Computers & Chemical Engineering*, vol 9, no 4, pp 405-406.

REUTER, M.A. & VAN DEVENTER, J.S.J. 1990. The use of linear programming in the optimal design of flotation circuits incorporating regrind mills. *International Journal of Mineral Processing*, vol 28, pp 15-43.

REUTER, M.A., VAN DER WALT, T.J. & VAN DEVENTER, J.S.J., Modeling of metal-slag equilibrium processes using neural nets. *Metallurgical Transactions B*, vol. 23B, 1992, pp. 643-650.

REUTER, M.A., VAN DEVENTER, J.S.J. & VAN DER WALT, T.J. 1993. A generalised neural net kinetic rate equation. *Chemical Engineering Science*, vol 48, no 7, pp 1281-1297.

REUTER, M.A., VAN DEVENTER, J.S.J., GREEN, J.C.A. & SINCLAIR, M. 1988. Optimal design of mineral separation circuits by use of linear programming. *Chemical Engineering Science*, vol 43, no 5, pp 1039-1049.

RICO-MARTÍNEZ, R., KRISCHER, K., KEVREKIDIS, I.G., KUBE, M.C. & HUDSON, J.L. 1992. Discrete versus continuous time non-linear signal processing of Cu electrodissolution data. *Chemical Engineering Communications*, vol 118, pp 25-48.

ROLLINS, D.K. & DAVIS, J.F. 1992. Unbiased estimation of gross errors in process measurements. *AIChE Journal*, vol 38, no 4, pp 563-572.

ROMAGNOLI, J.A. & STEPHANOPOULOS, G. 1980. On the rectification of measurement errors for complex chemical plants. *Chemical Engineering Science*, vol 35, no 5, pp 1067-1081.

ROMAGNOLI, J.A. & STEPHANOPOULOS, G. 1981. Rectification of process measurement data in the presence of gross errors. *Chemical Engineering Science*, vol 36, no 11, pp 1849-1863.

ROSEN, E.M., & PAULS, A.C. 1977. Computer-aided chemical process design. *Computers & Chemical Engineering*, vol 1, no 1, pp 11-21.

ROSENBAUM, A. 13 Jul 1992. Europe's fuzzy success. *Electronics*, p 26.

ROSSETTO, O., JUTTEN, C. & HERAULT, J. 1989. Analog VLSI synaptic matrices as building blocks for neural networks. *IEEE Micro*, vol 9, no 6, pp 56-63.

ROTH, M.W. 1990. Survey of neural network technology for automatic target recognition. *IEEE Transactions on Neural Networks*, vol 1, no 1, pp 28-43.

RUMELHART, D. E., HINTON, G. E. & WILLIAMS, R. J. 1986. Learning internal representations by error propagation. *Parallel Distributed Processing*, vol 1, pp 318-362.

SANCHEZ, M., BANDONI, A. & ROMAGNOLI, J. 1992. PLADAT: A package for process variable classification and plant data reconciliation. *European Symposium on Computer Aided Process Engineering* - 1, 24-28 May 1992, Elsinore, Denmark, Special Issue of Computers & Chemical Engineering, pp S499-S506.

SARMA, M.S. 1990. On the convergence of the Baba and Dorea random optimization methods. *Journal of Optimization Theory and Applications*, vol 66, no 2, pp 337-343.

SBARBARO, D. 1991. Neural networks in process control. *Process Engineering*, vol 72, no 5, pp 59-63.

SERTH, R. W. & HEENAN, W. A. 1986. Gross error detection and data reconciliation in steam-metering systems. *AIChE Journal*. vol 32, no 5, pp 733-742.

SERTH, R. W., VALERO, C. M. & HEENAN, W. A. 1987. Detection of gross errors in nonlinearly constrained data: a case study. *Chemical Engineering Communications*, vol 51, pp 89-104.

SHACHAM, M., MACHIETTO, S., STUTZMAN, L. & BABCOCK, P. 1982. Equation oriented approach to process flowsheeting. *Computers & Chemical Engineering*, vol 6, no 2, pp 79-95.

SHANNO, D.F. 1983. Large scale unconstrained optimization. *Computers & Chemical Engineering*, vol 7, no 5, pp 569-574.

SIKORA, R. 1992. Learning control strategies for chemical processes - a distributed approach. *IEEE Expert*, vol 7, no 3, pp 35-43.

SMETS, H.G.M. & BOGAERT, W.F.L. 1992. Deriving corrosion knowledge from case histories: the neural network approach. *Materials & Design*, vol 13, no 3, pp 149-153.

SMITH, G.J. & MORTON, W. 1988. Dynamic simulation using an equation-oriented flowsheeting package. *Computers & Chemical Engineering*, vol 12, no 5, pp 469-473.

SORSA, T., KOIVO, H.N. & KOIVISTO, H. 1991. Neural networks in process fault diagnosis. *IEEE Transactions on Man, Systems, and Cybernetics*, vol 21, no 4, pp 815-825.

STADTHERR, M.A. & HILTON, C.M. 1982. On efficient solution of large-scale Newton-Raphson based flowsheeting problems in limited core. *Computers & Chemical Engineering*, vol 6, no 2, pp 115-120.

STADTHERR, M.A. & VEGEAIS, J.A. 1985. Advantages of super-computers for engineering applications. *Process Engineering*, vol 81, no 9, pp 21-24.

STEPHENSON, G. & SHEWCHUCK, C. 1986. The reconciliation of process data with process simulation. *AIChE Journal*, vol 32, no 2, pp 247-254.

SU, H.-T. & McAVOY, T.J. 1992. Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial Engineering Chemistry & Research*, vol 31, no 5, pp 1338-1352.

TAKAMATSU, T. 1983. The nature and role of process systems engineering. *Computers and Chemical Engineering*, vol 7, no 4, pp 203-218.

TAKEFUJI, Y. & LEE, K.C. 1991. Artificial neural networks for four-coloring map problems and K-colorability problems. *IEEE Transactions on Circuits and Systems*, vol 38, no 3, pp 326-333.

TAMHANE, A. C. & MAH, R. S. H. 1985. Data reconciliation and gross error detection in chemical process networks. *Technometrics*, vol 27, no 4, pp 409-422.

TANK, D.W. & HOPFIELD, J.J. 1986. Simple "neural" optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, vol CAS-33, no 5, pp 533-541.

TJOA, I.B. & BIEGLER, L.T. 1991. Simultaneous strategies for data reconciliation and gross error detection of nonlinear systems. *Computers and Chemical Engineering*, vol 15, no 10, pp 679-690.

UPADHYE, R.S. & GRENS, E.A. 1975. Selection of decompositions for chemical process simulation. *AIChE Journal*, vol 21, no 1, pp 136-143.

VACLAVEK, V. & LOUCKA, M. 1976. Selection of measurements necessary to achieve multicomponent mass balances in chemical plant. *Chemical Engineering Science*, vol 31, no 12, pp 1199-1205.

VACLAVEK, V., RAJNIAK, P. & ILAVSKY, J. 1979. Sensitivity analysis in balance computations of complex chemical processes. *Chemical Engineering Communications*, vol 3, pp 377-386.

VAN DER WALT, T.J. & VAN DEVENTER, J.S.J. 1992. The simulation of ill-defined metallurgical processes using a neural net training program based on conjugate-gradient optimization. *IFAC Workshop on Expert Systems in Mineral and Metal Processing*. [Helsinki University of Technology, Espoo, Finland, 26-28 August].

VAN DER WALT, T.J. & VAN DEVENTER, J.S.J. 1993. Transformation of continuous process data for nonparametric modelling, *The Chemical Engineering Journal*. (submitted).

VAN DER WALT, T.J., BARNARD, E. & VAN DEVENTER, J.S.J. 1993. Process modelling with the regression network. *IEEE Transactions on Neural Networks*, (submitted).

VAN DER WALT, T.J., VAN DEVENTER, J.S.J. & BARNARD, E. 1993a. The estimation of kinematic viscosity of petroleum crude oils and fractions with a neural net. *The Chemical Engineering Journal*, vol 51, pp 151-158.

VAN DER WALT, T.J., VAN DEVENTER, J.S.J. & BARNARD, E. 1993b. The dynamic modelling of ill-defined processing operations using connectionist networks. *Chemical Engineering Science*, vol 48, no 11, pp 1945-1958.

VAN DER WALT, T.J., VAN DEVENTER, J.S.J. & BARNARD, E. 1993c. Neural nets for the simulation of mineral processing operations. Part I. Theoretical principles. *Minerals Engineering*, (accepted for publication).

VAN DER WALT, T.J., VAN DEVENTER, J.S.J. & BARNARD, E. 1993d. Neural nets for the simulation of mineral processing operations. Part II. Applications. *Minerals Engineering*, (accepted for publication).

VAN DER WALT, T.J., VAN DEVENTER, J.S.J. & BARNARD, E. 1993e. The regression network: A new connectionist network for modelling with sparse data. *Chemical Engineering Science*, (submitted).

VAN DER WALT, T.J., VAN DEVENTER, J.S.J., BARNARD, E. & OOSTHUIZEN, G.D. 1992. The simulation of ill-defined processing operations using connectionist networks. *Proceedings of APCOM '92*, SME-AIME., Littleton CO., Editor: Y.C. Kim, Chapter 85, pp. 879-888. [23rd International Symposium on the Application of Computers and Operations Research in the Minerals Industry, The University of Arizona, Tucson, AZ, U.S.A., 7-11 April 1992].

VAN DEVENTER, J.S.J., VAN DER WALT, T.J. & REUTER, M.A. 1992. The simulation of ill-defined hydrometallurgical plants using artificial intelligence. *Proceedings of the 2nd International Conference on Hydrometallurgy (ICHM' 92)*, vol 1, Editors: Chen Jiayong, Yang Songqing & Deng Zuoqing, International Academic Publishers, Beijing, 1992, pp. 132-137. [Changsha, China, 23-26 October 1992].

VECCHIETTI, A.R., LEONE, H.P., MELLI, T.R., MONTAGNA, J.M., & CERRO, R.L. 1987. SIMBAD: a process simulator linked to a DBMS - 2. The structure of memory. *Computers & Chemical Engineering*, vol 11, no 1, pp 519-525.

VEGEAIS, J.A. & STADTHERR, M.A. 1992. Parallel processing strategies for chemical process flowsheeting. *AIChE Journal*, vol 38, no 9, pp 1399-1407.

VENKATASUBRAMANIAN, V. & McAVOY, T.J. 1992. Editorial: Neural network applications in chemical engineering. *Computers & Chemical Engineering*, vol 16, no 4, pp v-vi.

VENKATASUBRAMANIAN, V., VAIDYANATHAN, R. & YAMAMOTO, Y. 1990. Process fault detection and diagnosis using neural networks - I. Steady-state processes. *Computers & Chemical Engineering*, vol 14, no 7, pp 699-712.

VERLEYSEN, M. & JESPERS, P.G.A. 1989. An analog VLSI implementation of Hopfield's neural network. *IEEE Micro*, vol 9, no 6, pp 46-55.

VERNEUIL, V.S. (Jr), YANG, P. & MADRON, F. 1992. Banish bad plant data. *Chemical Engineering Progress*, vol 88, no 10, pp 45-51.

WALPOLE, R.E. & MYERS, R.H. 1978. *Probability and statistics for engineers and scientists*. MacMillan Publishing Co., Inc., N.Y.

WANG, C.J. & TSANG, E.P.K. 1991. Solving constraint satisfaction problems using neural networks. *Second International Conference on Artificial Neural Networks*, 18-21 November 1991, pp 295-299.

WANG, J. 1992a. Analog neural net for solving the assignment problem, *Electronics Letters*, vol 28, no 11, pp 1047-1050.

WANG, J. 1992b. Electronic realization of recurrent neural nets for solving simultaneous linear equations, *Electronics Letters*, vol 28, no 5, pp 593-595.

WANG, J. 1992c. Recurrent neural networks for solving systems of complex-valued linear equations. *Electronics Letters*, vol 28, no 18, pp 1751-1753.

WASSERMAN, P.D. 1989. *NEURAL COMPUTING: THEORY AND PRACTICE*. Von Nostrand-Reinhold.

WESTERBERG, A.W. & BERNA, T.J. 1978. Decomposition of very large-scale Newton-Raphson based flowsheeting problems. *Computers & Chemical Engineering*, vol 2, no 1, pp 61-63.

WESTERBERG, A.W. 1991. Process Engineering. In: *Advances in Chemical Engineering*, vol 16. Eds Colton, C.K. Academic Press, Inc., San Diego, CA, 1991.

WHITE, J.W., WINSLOW, R.L. & ROSSITER, G.J. 1977. A useful technique for metallurgical mass balances - applications in grinding. *International Journal of Mineral Processing*, vol 4, pp 39-49.

WHITEN, W.J. 1972. The simulation of crushing plants with models developed using multiple spline regression. *Journal of the South African Institute of Mining and Metallurgy*, vol 72, no 10, pp 257-264.

WILLIAMS, T. 1987. Optics and neural nets - trying to model the human brain. *Computer Design*, vol 26, pp 47-62.

WOOLLACOTT, L. C., MOYS, M. & HINDE, A. 1992. The use of material balance smoothing in the evaluation of backfill operations. *Journal of the South African Institute of Mining and Metallurgy*, vol 92, no 5, pp 121-129.

YDSTIE, B.E. 1990. Forecasting and control using adaptive connectionist networks. *Computers & Chemical Engineering*, vol 14, no 4/5, pp 583-599.

YUILLE, A.L. 1989. Energy functions for early vision and analog networks. *Biological Cybernetics*, vol 61, pp 115-123.

ZURADA, J.M. 1992. Analog implementation of neural networks. *IEEE Circuits & Devices*, vol 8, no 5, pp 36-41.

APPENDIX A
BRIEF REVIEW OF THE FUNDAMENTALS OF BACK PROPAGATION NEURAL NETWORKS

A.1 STRUCTURE OF BACK PROPAGATION NEURAL NETWORKS

Excellent in-depth discussions on neural nets can be found in the literature and only a very brief overview is provided in this dissertation (Hecht-Nielsen, 1990; Hush & Horne, 1993, Lippmann, 1987, 1989; Rumelhart et al., 1986; Wasserman, 1989).

A neural net is a parallel distributed information processing structure, consisting of an arrangement of interconnected primitive processing elements. Each processing element can have an arbitrary number of input connections, but only one output connection (that can branch or fan out to form a multiple output connection) as shown in figure A.1. These elements or artificial neurons can have local memory and also possess transfer functions that can use or alter this memory, process input signals and produce the output signals of the elements.

The processing elements of a neural net are typically divided into disjoint subsets, called layers, in which all the process units generally possess the same computational characteristics. The layers comprising a neural net are usually categorized as either input, hidden or output layers, to denote the way in which they interact with the information environment of the net.

The back propagation nets used in this study were feedforward networks (see figure A.2) which could be trained by repeatedly presenting them with examples of scaled inputs (see next section) and desired outputs (Bhat et al., 1990; Bhat & McAvoy, 1990; Hecht-Nielsen, 1990; Hinton, 1989; Hornik, et al., 1989; Karim & Rivera, 1992; Leonard & Kramer, 1990; Lippmann, 1987, 1989; Rumelhart et al., 1986; Wasserman,

1989). Training, which entailed the adjustment of the weight matrix of the net, occurs by means of learning algorithms designed to minimize the mean square error between the desired and the actual output of the net (Battiti, 1992; Bhat & McAvoy, 1990). During the learning process information is propagated back through the net in order to update the connection weights of the net, so that the net can form an internal representation of the relationship between the inputs and the outputs presented to it.

A.2 NEURODYNAMICS

Computation in back propagation neural nets is feedforward and synchronous, i.e. the states of the process units in layers nearest to the input layer of the net are updated before units in successive layers further down in the net. The activation rules determine the way in which the process units are updated and are typically of the form

$$v_i(t+1) = g[u_i(t)] \qquad (A.1)$$

where $u_i(t)$ designates the potential of a process unit at time t, i.e. the difference between the weighted sum of all the inputs to the unit and the unit bias

$$u_i(t) = \Sigma_j w_{i,j}.v_j(t) - \Theta_i \qquad (A.2)$$

The form of the transfer function g may vary, but could be a linear, step or sigmoidal transfer function, among others, with a domain typically much smaller than that of the potential of the process unit, such as [0;1] or [-1;1], for example.

The training of back propagation neural nets is an iterative process involving the changing of the weights of the net, typically by means of a gradient descent method, in order to minimize an error criterion, that is

$$w_{i,j}(t+1) = w_{i,j}(t) + \Delta w_{i,j}, \text{ where} \qquad (A.3)$$

$$w_{i,j} = -\tau.\partial\epsilon/\partial w_{i,j} \qquad (A.4)$$

where τ is the learning rate and ϵ the error criterion, i.e.

$$\epsilon = \tfrac{1}{2}.\Sigma(T_{o,j} - v_{o,j})^2 \qquad (A.5)$$

based on the difference between the desired ($T_{o,j}$) and the actual outputs ($v_{o,j}$) of the unit.

A.3 SCALING OF DATA

Before data can be presented to a neural net, it is usually necessary to scale them to ranges which would enable the net to learn. A hyperbolic tangent transfer function produces outputs in the range lying between -1 and 1 and for this type of net to learn effectively, it is necessary to scale the outputs to the same range. This is usually accomplished by mapping the minimum and maximum values of the actual input and output data linearly to the respective minimum and maximum values of the network ranges. If an exemplar presented to the net consists of I input fields and D output fields, i.e. [$f_1\#$, $f_2\#$, .. $f_I\#$, $f_{I+1}\#$, $f_{I+2}\#$, .. $f_{I+D}\#$], two sets of corresponding vectors can be defined [m_1, m_2, .. m_I, m_{I+1}, m_{I+2}, .. m_{I+D}] and [M_1, M_2, .. M_I, M_{I+1}, M_{I+2}, .. M_{I+D}], where m_k and M_k typically correspond to the minimum and maximum values[4] that $f_k\#$ could assume. If the ranges allowed for the input and output layer of the net are respectively defined as ($r_I\#$, $R_I\#$) and ($r_D\#$, $R_D\#$), $i_j\#$ as the network input corresponding to $f_j\#$, $d_k\#$ the network output corresponding to $f_k\#$, $o_k\#$ the actual output of the net and $g_k\#$ the corresponding real world output, then the mappings of the real world data to those of the network can then be described as follows:

Input

$$i_j\# = [(R_I\#-r_I\#).f_j\# + M_j.r_I\# - m_j.R_I\#]/[M_j-m_j] \qquad (A.6)$$

Output

$$d_k\# = [(R_D\#-r_D\#).f_k\# + M_k.r_D\# - m_k.R_D\#]/[M_k-m_k] \qquad (A.7)$$

Mapping from network output to real world

$$g_k\# = [(M_k-m_k).o_k\# + R_D\#.m_k - r_D\#.M_k]/[R_D\#-r_D\#] \qquad (A.8)$$

Non-numeric or missing field values are usually mapped to the middle of the target range, that is $\frac{1}{2}(R_I\# + r_I\#)$ or $\frac{1}{2}(R_D\# + r_D\#)$.

---

[4]These indices can assume any values, as long as $m_k < M_k$.

FIGURE A.1 TYPICAL STRUCTURE OF A FEEDFORWARD NEURAL NET

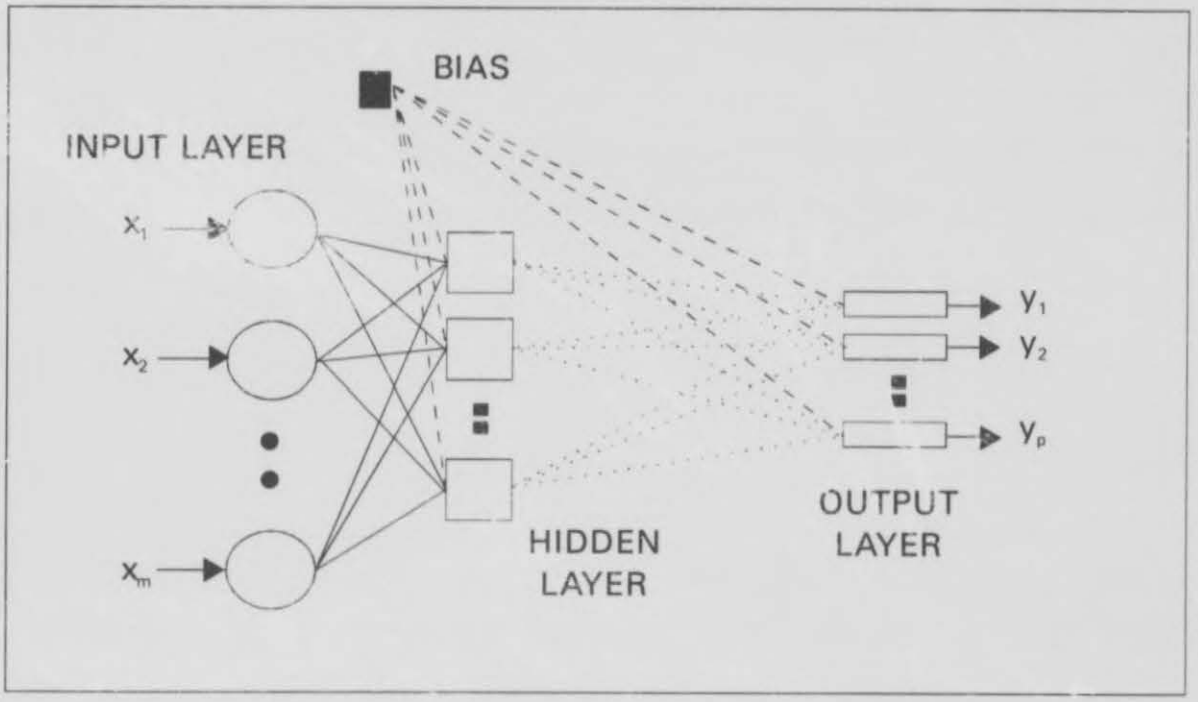
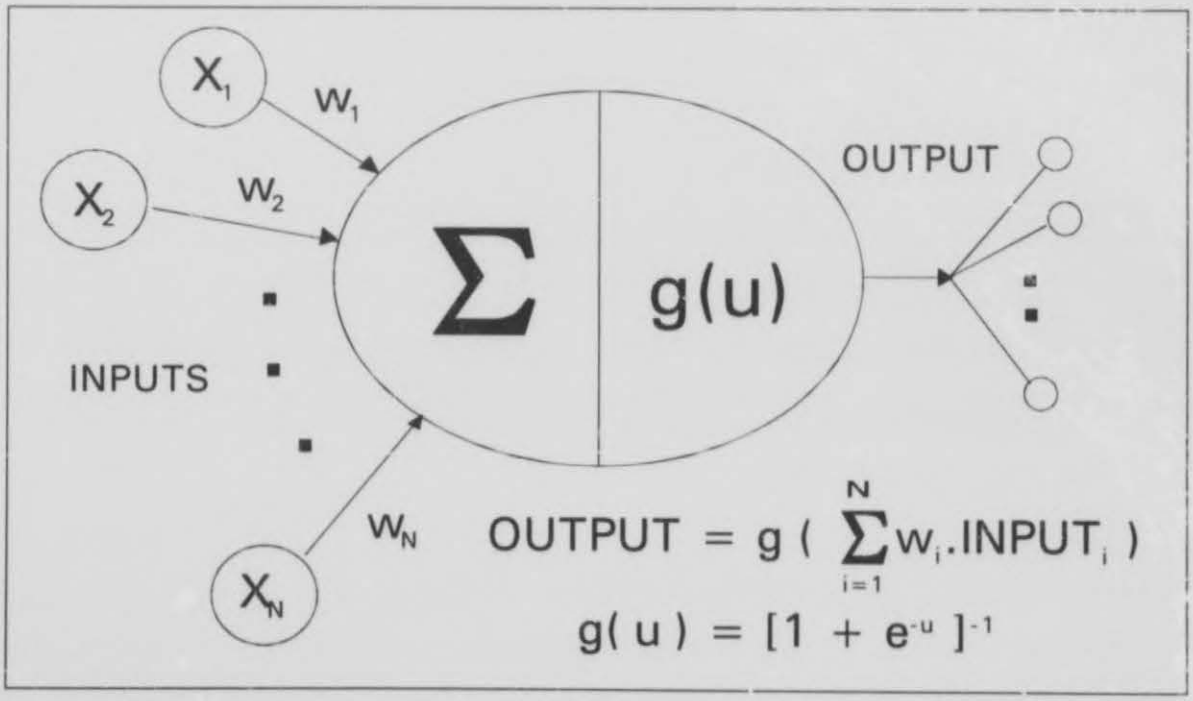


FIGURE A.2 PROCESSING ELEMENT OF A BACK PROPAGATION NEURAL NET



APPENDIX B
BRIEF REVIEW OF PROCESS SIMULATION METHODOLOGY

In the systems technology approach, it is fundamental to discern between systems or process *analysis* which is concerned with the outputs of a system, based on certain inputs, and process *synthesis*, which is concerned with the inputs of a system, based on certain outputs (Takamatsu, 1983). Analysis entails the investigation of the structures of the system and the relations and interactions among its various elements, and it is useful for calculating unknown system outputs from known inputs. In contrast, synthesis is concerned with the design of elements and their complex mode of interaction in order to transform given system inputs into desired outputs. Although distinct concepts, analysis and synthesis are closely entwined. In practice synthesis operations are followed by analyses in which the behaviour of the system is investigated, especially in order to derive a basis from which the future evolution and optimization of the system can be pursued.

Analysis is not limited to a formal decomposition of the system or object; it is a complex and creative process in its own right, and no general consensus exists as to the best strategy for the modelling and simulation of chemical plants or process circuits (Evans, 1987). In essence process simulation starts with a process flow sheet, from which a simulation model is constructed. This conceptual representation typically consists of a model or unit operation block for each processing step or processing unit in the circuit. These unit operation blocks are connected in a specific structure and comprized of sets of equations relating the inputs and outputs pertaining to each block. In the pre-computer era prior to the end of the Second World War no more than comparatively simple (stationary, discrete, deterministic, one-dimensional) models could be solved, and then only by graphical or analytical means (Hofmann, 1988). More

sophisticated models were developed as digital computers became cheaper and more available. Strategies for solving the equations of these models can broadly be classified as sequential modular, non-sequential modular (equation-based), or more recently, two-tier approaches, which are hybrids of the former two methods (Evans, 1987; Perregaard & Sorensen, 1992). Although the sequential modular approach (Diwekar et al., 1992) appears to be the more popular of the two, and the strategy employed in most commercial simulation software packages, equation solving methods are rapidly gaining ground.

B.1 STEADY STATE SIMULATION

Technically speaking, chemical processes are never at steady state, but are always fluctuating or drifting in the state space describing their behaviour. If these process changes take place over comparatively long periods, the process is considered to be in a steady or a quasi-steady state. Three different strategies are followed to model these equilibria, viz. the sequential modular, the equation-based and the simultaneous modular approach.

B.1.1 Sequential modular simulators

The sequential modular approach forms the backbone of most commercial process simulators, and is based on the implementation of process unit blocks as computational subroutines, calculating output as functionally related to input.

The first step towards the modelling of flowsheets with this strategy entails the partitioning of the flowsheet, i.e. the identification and assembly of collections of unit modules forming maximal cyclic subsystems that have to be solved together. Calculations are carried out by a sequential procession from one module to another, generally in the direction of the material flow streams. Recycle loops in the process are accommodated through initial estimates of selected (torn) recycle streams, which are updated in the course of successive passes through the flowsheet (McLane et al., 1979).

The development of sequential modular process simulators began in the late 1950s, when stand-alone programs designed for calculating unit

operations were sequenced, so that the flow of unit calculations corresponded with the flow of material and energy in the actual process (Biegler, 1989). The earliest attempt was made with Flexible Flowsheet, which was followed rapidly by others such as SPECS at Shell, COPE at Exxon and FLOWTRAN at Monsanto. These unit operation modules allowed the construction of large flowsheets with a minimum of effort, and the construction of special solution strategies without altering the overall approach to the flowsheeting programs. Difficulties arose when the recycle structure of the flowsheet resulted in awkward iterations in the calculation sequence, and as a consequence only the simplest convergence algorithms (such as those based on direct substitution, dominant eigenvalues and Wegstein routines), which allow convergence of a single recycle loop at a time, could be implemented. The general architecture of sequential modular process simulators is shown in figure B.1. It is estimated that up to 80% of the computational effort stems from the determination of physical properties of the materials to be processed, and as a result major differences between various sequential modular simulators can be attributed to the sophistication and scope of their physical property calculators. The individual unit or process models can be solved with different degrees of rigor, and can involve tens of thousands of equations to represent the equipment, as well as the physical properties of the materials being processed (Westerberg, 1991). The benefits of implementing a sequential modular simulator include conceptual simplicity, advantage that can be taken from a large number of industrially developed process models, and the possible inclusion of convergence heuristics accumulated over the years. Sequential modular methods used in process simulators such as ASPEN (Evans et al., 1979), PROCESS (Brannock et al., 1979) and SIMBAD (Leone et al., 1987; Montagna et al., 1987; Vecchietti et al., 1987) are well-suited for solving steady state simulation problems which are well-defined, although even these types of systems may involve up to three nested levels of iteration in the solution procedure (Perkins, 1983). These iteration levels include calculations concerned with the estimation of physical properties, the unit or process modules, and the convergence of torn flow streams.

Sequential modular packages are less efficient in solving systems where not all the parameters (feed streams to all the units and process parameters) are defined (Perkins, 1983), and the complications posed by

large highly integrated systems can moreover result in severe problems with convergence (Harrison, 1992). The most important disadvantage of sequential modular strategies might well prove to be their inability to exploit parallel computational strategies which are steadily growing in importance every year (Vegeais & Stadtherr, 1992). As the name suggests, these procedures can not solve different modules simultaneously, unless they are completely independent.

a) Flowsheet convergence methods

i) Unconstrained flowsheeting

Unconstrained flowsheeting models (see figure B.2) are often represented in block structures with external feed and product streams and a recycle loop with tear stream values $\mathbf{x}$. Under the sequential modular architecture the model of the block is structured in an input-output form, so that for specified values of the internal parameters of the model and given values of the process feed and tear streams, the model will provide values for the product streams, as well as a calculated value $\mathbf{f(x)}$ for the tear stream vector. The values of $\mathbf{x}$ and $\mathbf{f(x)}$ are subsequently converged to within suitable tolerances.

ii) Constrained flowsheeting

In the application of sequential modular simulators, it is often desirable to use the simulator to determine the values of design variables or internal parameters $\mathbf{p}$, in order to comply with certain design constraints, $\mathbf{d(x,p)=0}$, as depicted in figure B.3. These problems are referred to as *constrained* flowsheet problems, subject to the constraints $\mathbf{d(x,p)=0}$, and it is consequently necessary to find both $\mathbf{x}$ and $\mathbf{p}$ so that $\mathbf{f(x)=x}$ and $\mathbf{d(x,p)=0}$. In sequential modular simulators, this is traditionally accomplished by use of additional calculation loops, which may be nested with the tear stream loops. In an outer loop strategy for assumed values of $\mathbf{x}$ and $\mathbf{p}$ for example, tear stream calculations are repeated until the equality $\mathbf{f(x)=x}$ is satisfied. The constraints $\mathbf{d(x,p)=0}$ are then evaluated, and if not satisfied, $\mathbf{p}$ is suitably adjusted and the iteration process repeated. These variable partitioning schemes are often ineffective, so that tear equations and design constraints have to be solved simultaneously. Some of the numerical methods which are

commonly used in constrained and unconstrained sequential modular models, are successive substitution, as well as the methods of Wegstein and Broyden.

The simplest approach for the solution of $\mathbf{f(x)} = \mathbf{x}$, is based on successive substitution, i.e. $x_{i+1} = f(x_i)$. Convergence is guaranteed only if all eigenvalues of the Jacobian of $\mathbf{f(x)}$ have a modulus less than unity (Clark & Reklaitis, 1984). This method is often used in a modified form to enhance convergence (Wegstein's method),

$$x_{j,i+1} = (1-q_{j,i}).f_{j,i}(x_i) + q_{j,i}.x_{j,i} \qquad (B.1)$$

where $q_{j,i} = s_{j,i}/(s_{j,i-1})$, and

$$s_{j,i} = [F_{j,i}(x_i) - F_{j,i-1}(x_{i-1})]/[x_{j,i} - x_{j,i-1}]$$

To induce stability $q_{j,i}$ is usually bounded from above and below; $q_{j,min} \leq q_{j,i} \leq q_{j,max}$. A more sophisticated and powerful strategy is based on the method proposed by Broyden (Clark & Reklaitis, 1984; Broyden, 1965), which can be summarized as follows

$$y_{i+1} = y_i - t_i.H_{i+1}.g_i(y_i) \qquad (B.2)$$

where

$$H_{i+1} = H_i - (H_i.\Delta g_i - \Delta y_i).(\Delta y_i)^T.H_i/(\Delta y_i)^T H_i.\Delta g_i), \text{ and}$$

$$(\Delta y_i)^T H_i.\Delta g_i = 0$$

where H is an approximation to the inverse Jacobian of $J^{-1}(y)$, and $\Delta J^{-1}{}_i = J^{-1}{}_i - J^{-1}{}_{i-1}$, $\Delta g_i = g_i - g_{i-1}$ and $\Delta y_i = y_i - y_{i-1}$

B.1.2 Non-sequential modular or equation-based methods

In contrast to the development of sequential modular simulators, which were historically developed mainly by the industrial community, non-sequential process simulators were by and large derived from academic circles (Biegler, 1989) and although these simulators have certain attractive features, their diffusion into the commercial arena remains slow (personal communication, Chimowitz, 1993). These equation solving methods (Perkins, 1983; Rajniak et al., 1992) comprising the so-called

simultaneous or global approach, are concerned with the collection of all equations describing the flowsheet and to solve them as a large system of non-linear equations. In contrast to the sequential modular apprcach, the numerical procedures and the directionality of information flow through the system is completely divorced from the plant description. The flowsheeting problem is essentially reduced to the solution of a very large and sparse set of non-linear equations (Vegeais & Stadtherr, 1992) which can involve tens or even hundreds of thousands of equations (Fouchy, 1991; Hlavacek, 1977; Westerberg, 1991), and can be mathematically summarized as:

$$\text{solve } \mathbf{F}(\mathbf{x},\mathbf{u}) = \mathbf{0} \qquad \text{(B.3)}$$

$$\text{with } \mathbf{G}(\mathbf{x},\mathbf{u}) \leq \mathbf{0}, \text{ where}$$

$\mathbf{x}$ is the vector of state (independent) variables, and $\mathbf{u}$ is the vector of decision (independent) variables, $\mathbf{F}(\mathbf{x},\mathbf{u})$ constitutes the set of process models equations and $\mathbf{G}(\mathbf{x},\mathbf{u})$ the set of inequality or equality constraints. The extra step requiring the calculation of output streams from input streams for constituent process units found in the sequential modular approach is consequently dispensed with in equation-based methods. Two basic approaches can be followed to solve the systems of equations, and the rationale behind both is to obtain a solution strategy that will converge rapidly and reliably for the particular problem (Stadtherr & Hilton, 1982).

The first approach is based on tearing a sufficient number of variables to permit the remaining variables to be calculated as a sequence of smaller problems. The tear variables are calculated by some sort of successive substitution procedure, provided that the tear equations contain the tear variables in an explicit form. If not, standard root finding techniques can be used together with the residuals of the tear equations. This approach effectively amounts to the solution of a large system of non-linear equations by iterating only on a few tear variables, which constitutes a drastic reduction in the dimensions[illegible]oblem. The key step in the tearing approach hinges arou[illegible]elopment of an appropriate solution strategy, i.e. which variab[illegible] which equations to solve for which variables (the output set) as well as the sequence in which these equations should be solved (precedence ordering). Owing to the

problems involved with the efficient selection of a reliable solution strategy, the approach based on tear equations has not been adopted whole-heartedly by the process engineering community.

The second approach to solving the system of equations, the quasi-linear approach, involves the simultaneous linearization of all equations and iteration on all variables, typically using Newton-Raphson and quasi-Newton-Raphson methods or suitable variants thereof. In each iteration a huge set of sparse linear equations (possibly involving several thousand variables) has to be solved, and the use of sparse matrix strategies is essential for all but the smallest problems, due to the computational problems created by fill-in associated with the use of normal matrix methods.

The advantage of the equation-based approach is that it is a convenient and natural method for specifying variables and constraints. It is also the approach with the most potential for exploiting parallel computational structures, provided that the computation of sparse matrices can be parallelized adequately. Disadvantages include the requirement of good initial estimates for variables, difficulties that might be associated with the handling of non-linear and discontinuous relationships between variables, especially those relating physical properties, possible difficulties associated with the diagnosis of problems, as well as not making use of the large number of unit operation models developed by industries.

a) Executive routines

The executive part of the process simulation model is concerned with the management of the flow of information during simulation. The executive accepts input data, determines the topology of the flowsheet and derives and controls the sequence of calculations in the flowsheet. Control is then passed to the unit operation level for the execution of each module, where specialized procedures from a unit operations library calculate material and energy balances for a particular unit. Frequent calls are also made by the executive and unit operations to physical properties libraries for such routine tasks as phase equilibrium, enthalpy and other stream property calculations.

b) Sparse matrix methods

As has been mentioned, one of the approaches to solving a large system of non-linear equations, involves the simultaneous linearization of all equations and iteration on all the variables until the system converges. The repeated solution of large sparse linearized systems of equations of the form $\mathbf{A}.\mathbf{x} = \mathbf{k}$ can easily overburden the available computational resources, and it is therefor desirable to decompose these system into smaller and more manageable blocks of equations and variables. A few decomposition strategies commonly used for this purpose are decomposition into a so-called block triangular form (BTF), bordered block diagonal form (BBDF) and the bordered block triangular form (BBTF).

i) Block triangular form

The system of equations is first reduced to a set of irreducible blocks, after which the blocks are partitioned (Lin and Mah, 1978). The strategy fails for large irreducible blocks, owing to excessive computational requirements. In this case reversing back to a simultaneous modular approach can solve the problem, with each module representing a block with a relatively small number of units that can be handled more conveniently.

ii) Bordered block diagonal form

This technique proposed by Westerberg and Berna (1978), permits efficient use of mass storage and involves permutation of the coefficient matrix **A** into a bordered block diagonal form, i.e.

$$\mathbf{A} = \begin{array}{c|c} \mathbf{M}_1 & \mathbf{M}_2 \\ \hline \mathbf{M}_3 & \mathbf{M}_4 \end{array} \qquad \text{(B.4)}$$

where $\mathbf{M}_1$ is a block diagonal matrix, with each diagonal block corresponding to a single unit. The external variables that describe flows between units are represented by the border matrices $\mathbf{M}_2$ and $\mathbf{M}_4$. The rows constituting $\mathbf{M}_3$ and $\mathbf{M}_4$ represent a set of equations to reduce fill-in, while solving for $\mathbf{M}_1$, as well as a set of connecting equations describing the flowsheet. Ordinary Gaussian elimination is used to reduce

$\mathbf{M}_1$-blocks into the upper triangular form and corresponding areas in $\mathbf{M}_3$ to zero. An important advantage of this approach is that each block or process unit can be accommodated individually in memory, which greatly reduces the burden on computational resources. Since relatively little access (generally of the same order as the number of diagonal blocks) to mass storage is required, the performance of the bordered block diagonal form is not impeded unduly.

iii) Bordered block triangular form (BBTF)

The BBTF is an alternative to the BBDF approach of Westerberg and Berna, and entails the formation of a matrix **A**

$$\mathbf{A} = \begin{array}{c|c} \mathbf{M}_1 & \mathbf{M}_2 \\ \hline \mathbf{M}_3 & \mathbf{M}_4 \end{array} \qquad \text{(B.5)}$$

through appropriate tearing. In this case $\mathbf{M}_1$ is block triangular, and the columns in the borders $\mathbf{M}_2$ and $\mathbf{M}_4$ correspond to design variables, while the rows in $\mathbf{M}_3$ and $\mathbf{M}_4$ represent form equations. Off-diagonal elements in $\mathbf{M}_1$ indicate information flows between units in the system. Gaussian elimination with back substitution can similarly be used to solve the system.

B.1.3 Simultaneous-modular or two-tier approach

The late 1970s and the early 1980s saw the development of a continuum of approaches spanning the gap between sequential and non-sequential modular simulators. Simultaneous modular approaches gradually evolved into simultaneous modular methods where unit operations remained essentially intact, but stream connections were solved simultaneously. Equation-oriented approaches on the other hand, were adapted to incorporate procedures at the lowest levels, e.g. for the calculation of the physical properties of process materials. The simultaneous-modular approach is not as well-defined as the sequential modular or non-sequential modular approaches and broadly consists of strategies that have been developed to exploit the advantages of both the equation-based and the sequential modular approach, especially as far as making use of the large base of existing sequential modular software of the latter (Perkins, 1983, Shacham et al., 1982). Some of these strategies involve

the simultaneous solution of design specifications and torn recycle streams, as well as a two-tier approach where alternate use is made of an approximate equation-based process model and a rigorous procedural model (Mahalec et al., 1979). In the two-tier approach the approximate equation-based model is solved exactly in order to generate new parameter estimates for use in the rigorous model. The rigorous model is used in turn to generate new values for adjustable parameters in the approximate model, the approximations of which could be linear or non-linear (Mahalec et al., 1979).

B.1.4 Calculation of physical properties

Virtually all commercial simulators have separate sections that distinguish between process models and stream connections on the one hand, and the physical properties (data banks and correlations) on the other. Almost all commercial simulators make use of cubic equations of state models for the description of pure components. Local activity coefficient models (such as UNIQUAC, NRTL and Wilson) are supplemented by the use of group contribution methods (such as UNIFAC), while well-known activity coefficient correlations for hydrocarbons, such as Chao-Seader and Grayson-Streed are employed frequently.

Owing to the time consuming nature of physical property models, as well as their separation from the rest of the flowsheet, the use of local property models that can be fully exploited by the simulator has been proposed (Biegler, 1989).

B.2 OPTIMIZATION

The optimization of process flowsheets became feasible for large scale problems with the development of linear programming codes in the late 1950s and gained further momentum with the introduction of mixed integer linear programming packages in the 1960s (Westerberg, 1991). Prior to that, optimization consisted of ad hoc techniques, such as case studies, which could only be applied to small plants.

When a flowsheet is optimized, a system of equations similar to those encountered during simulation, is solved with some process parameters

determined and others free, in order to maximize or minimize a specific objective function, i.e.

$$\max/\min F(\mathbf{x},\mathbf{z}) \qquad (B\ 6)$$

subject to

$$\mathbf{R}(\mathbf{x},\mathbf{z}) = \mathbf{0}$$

$$\mathbf{H}(\mathbf{x},\mathbf{z}) = \mathbf{0}$$

$$\mathbf{G}(\mathbf{x},\mathbf{z}) \geq \mathbf{0}, \text{ and}$$

$$\mathbf{z}_{min} \leq \mathbf{z} \leq \mathbf{z}_{max}$$

where $\mathbf{z}$ is a vector of decision variables, $\mathbf{x}$ a vector of system variables, F the objective function, $\mathbf{R}(\mathbf{x},\mathbf{z})$ the flowsheet equations, $\mathbf{H}(\mathbf{x},\mathbf{z})$ the design specification constraints, $\mathbf{G}(\mathbf{x},\mathbf{z})$ inequality constraints and $\mathbf{z}_{min}$ and $\mathbf{z}_{max}$ the bounds on the decision variables.

Methods for solving flowsheet optimization problems can be divided into two broad categories, namely *feasible* and *infeasible* path methods (Kisala et al., 1987). Feasible path methods require the equality constraints of the problem to be satisfied at every intermediate estimate of the decision variables along the trajectory towards the optimal solution, while with infeasible path methods the equality constraints need only be satisfied at the optimal solution. Since the entire flowsheet has to converge at every time step of the calculation sequence, feasible path methods are very time consuming and only really effective as far as smaller problems are concerned (Biegler, 1989).

Infeasible path methods, which came into their own in the early 1980s, can be subdivided into three further classes, depending on the type of simulation strategy involved, i.e. sequential modular, two-tier simultaneous modular and equation-based methods. In sequential modular methods, the modular architecture of the simulator is used, and tear streams and the optimization problem are converged simultaneously by a convergence block (Biegler, 1989; Biegler & Hughes, 1982). Successive quadratic programming techniques, which require very few function evaluations for convergence, are often used for optimization in

commercial simulators. These techniques are less efficient with regard to equation-based methods, which generally require the optimization of large problems subject to many equality constraints. Equation-based methods solve the optimization problem directly instead, as a large non-linear programming problem. Although these methods are potentially very efficient, more development needs to be done in order to counter many of the numerical problems arising from applications to chemical engineering problems (Kisala, et al., 1987). Simultaneous modular or two-tiered have their origin in attempts to apply the equation-oriented methodology iteratively. These algorithms make use of sequential modular process unit simulators to generate a simplified flowsheet that can be solved as a non-linear programming problem. Hybrid methods which incorporate or combine some of the features of these strategies have been proposed by various authors (Biegler, 1982).

A number of methods is available for solving the non-linear programming problem, of which gradient-based algorithms and especially successive quadratic programming are regarded as one of the most efficient (Chan & Prince, 1986; Westerberg, 1991). In successive quadratic programming procedures a quadratic programming problem is set up at each step, by taking a second order approximation of the Lagrange function and first order approximations of the constraints. The solution of this quadratic programming problem yields a search direction along which a penalty function can be minimized, and the process is repeated at each base point until a solution to the original programming problem is obtained. Constraints are not necessarily satisfied at base points.

Examples of equation oriented packages include QUASILIN (Hutchison et al., 1986a, 1986b; Smith & Morton, 1988), SYMBOL (Gorczynski et al., 1979), GENDER (Gorczynski et al., 1979) and SPEEDUP (Westerberg, 1991).

B.3 DYNAMIC SIMULATION

Despite the availability of several dynamic process simulation tools, no system has yet gained general acceptance as a simulation tool for large scale process plants (Hillestadt & Herzberg, 1986). The dynamic

simulation of processes typically involves systems of stiff non-linear ordinary differential equations (ODEs), as well as algebraic equations, usually of high dimensionality. These equations are typically sparse (often less than 1% of the Jacobian elements are non-zero). In contrast to steady state systems, complications such as discontinuities between time and state events can also occur. Dynamic systems can be simulated by modular as well as equation-based methods.

B.3.1 Modular methods

There are two different approaches to modular dynamic simulation, also referred to as coupled modular methods. The coupled modular approach is also referred to as the simultaneous modular approach in the literature. This is unfortunate, since it leads to confusion with simultaneous modular (two-tier) methods used in steady state simulation, to which it bears little resemblance. In coupled modular methods all modules are integrated by a common routine, such as implemented by the general purpose process simulator DYNSYL). As an alternative, modules are provided with individual integration routines, integrated over a common time horizon. This approach is known as uncoupled modular, independent modular or sequential modular (Hlavacek, 1983). Use of the last term is also not recommended, since it can lead to unnecessary confusion with sequential modular approaches used in steady state simulation, to which it bears little resemblance. Although integration usually refers to routines for solving systems of ordinary differential equations, partial differential equations can be solved as well, provided that these equations can be discretized properly.

B.3.2 Equation-based methods

With equation-based methods all the equations of the system are solved simultaneously, through partitioning (various methods are available) and the use of sparse matrix techniques, to take advantage of the structure of the sets of equations. Some of these methods include decomposition of the system into a block triangular matrix, based on the structure of the occurrence matrix of the system (Himmelblau & Bischoff, 1968). A solution is then obtained by direct or iterative techniques. An alternative strategy is to decompose the flowsheet, so that individual units can be treated separately. Integration of the systems of equations is

accomplished by predictor-corrector methods. Depending on its activity, each unit can then be solved with a different number of Newton-Raphson iterations. Another promising strategy is to decompose the system into two subsystems - a fast one and a slow one. Different integration methods and step lengths (multi-rate methods) can then be applied to each system. The two subsystems are connected by means of a polynomial interpolator or extrapolator.

B.4 FUTURE TRENDS IN PROCESS SIMULATION

Process industries are at present experiencing tremendous changes. Companies of industrialized countries are increasingly focusing on the manufacture of high technology or specialty products with an increased intellectual component, such as those encountered in the fields of biotechnology, pharmaceuticals and material science, while moving away from the manufacturing of commodity goods, which are becoming less attractive (Gorsek et al., 1992). As a consequence producers of commodity goods such as metals, minerals, pulp, food and paper, are compelled to become more efficient, and to rely increasingly on the use of process simulation to improve plant performance. The demand for improved modelling techniques is moreover driven by increased use of batch processing in the manufacture of specialty products (Graells et al, 1992).

Advances in the simulation and modelling of chemical and metallurgical processes is so closely related to the growth of the information processing industry, that it can not really be considered in proper perspective without focusing on the development of computational tools (Evans, 1987; Sbarbaro, 1991). The continuous demand for more sophisticated and powerful computational systems is two-fold. Although conceptually simple to model, a process system may be large, so that the sheer computational burden posed by the dimension or structure of the system causes it to be an intractable problem to solve. In certain cases, a doubling in the dimension of the problem can for instance result in up to a sixteenfold increase in the computational burden placed on conventional methods (Rangaiah, 1985; Heuckroth et al., 1976; Shanno, 1983). Other types of systems, on the other hand, may involve only a few variables, but could be very hard to describe fundamentally. The first type of problem has fueled the demand for so-called supercomputing

devices, while the second has underpinned the necessity for computational structures which can accommodate heuristic knowledge. Both demands have been met by a number of different systems which have been developed or matured in the last decade only. Of these, neural nets are particularly attractive, due to their potential for processing huge amounts of data, as well as for their ability to serve as repositories for heuristic or empirical knowledge.

FIGURE B.1 PROCESS ANALYSIS AND SYNTHESIS

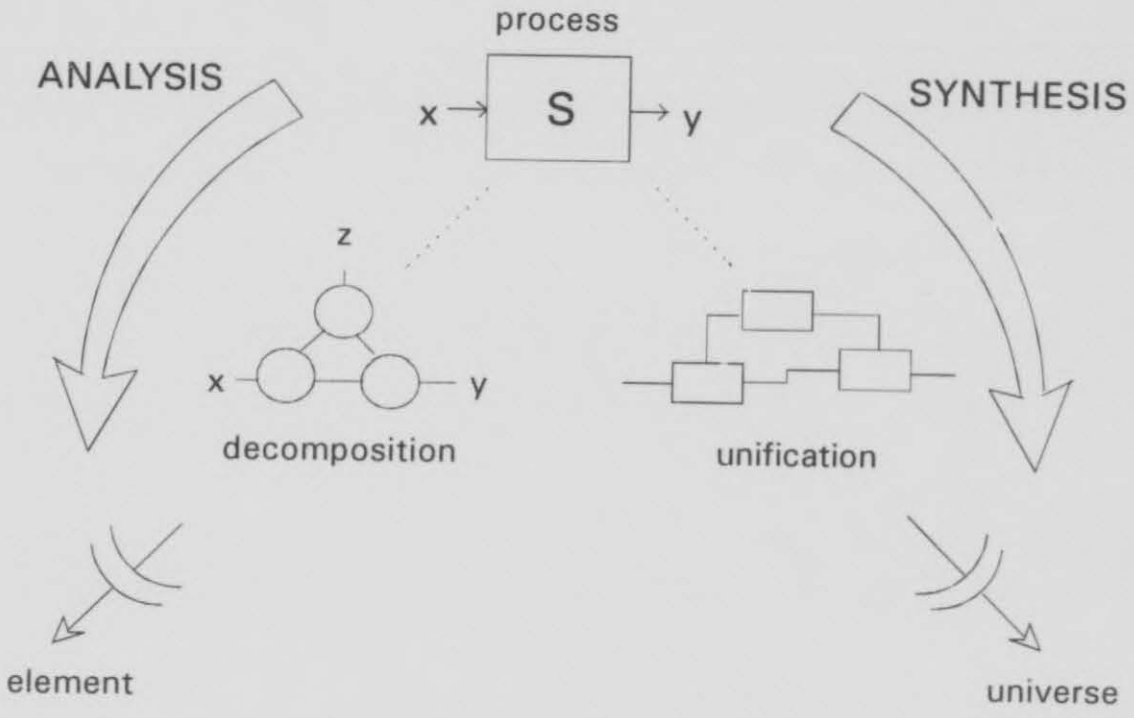


FIGURE B.2 TYPICAL UNCONSTRAINED FLOWSHEET

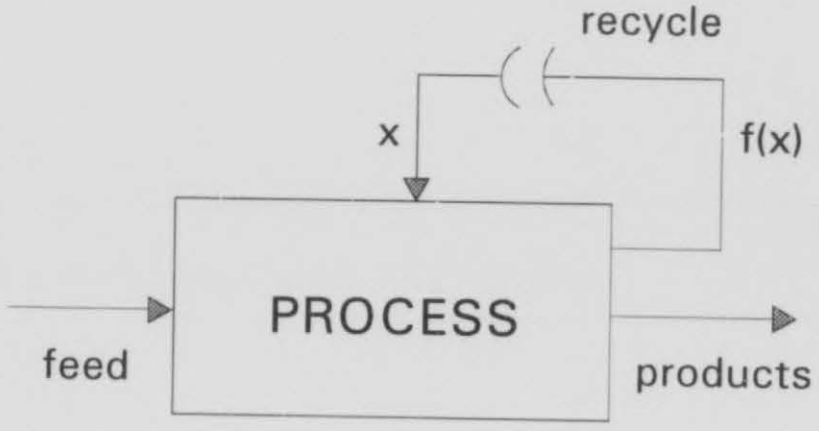
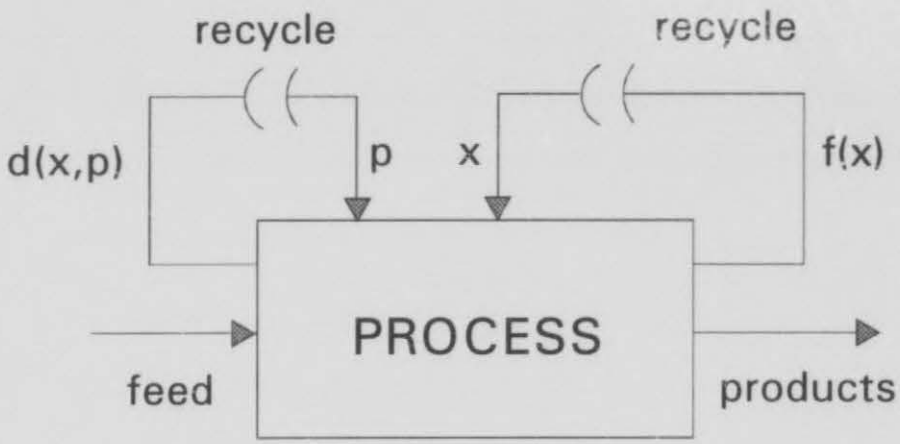


FIGURE B.3 TYPICAL CONSTRAINED FLOWSHEET



APPENDIX C
DECOMPOSITION OF PROCESS CIRCUITS CONTAINING RECYCLE FLOW STREAMS

Before a large process system[5] representing a complex network of interacting elements can be analyzed, it has to be decomposed into smaller subsystems that can be investigated separately in order to decrease the burden on computing facilities. Decomposition is typically implemented in stages (Mah, 1983) and to this end it is convenient to represent the flowsheet in terms of a digraph, the vertices of which represent the process units and the edges of which represent the flow of material or energy between these units (Cordoba, 1988; Pho & Lapidus, 1973). The first stage of decomposition, or *partitioning*, entails the division of the flowsheet into blocks of maximal cyclicity, i.e. the parts of the flowsheet that have to be converged prior to the commencement of any downstream calculations (Evans et al., 1979; Mah, 1983). Process units in each of these maximal cyclic blocks are linked together by material and energy flow streams, and the equations describing these relationships consequently have to be solved together (Evans, et al., 1979; Mah, 1983). Partitioning is unique and can be conducted by a number of different algorithms, such as those based on the use of a reachability matrix or a depth-first search procedure (Mah, 1974; Ledet & Himmelblau, 1970).

As soon as the compiete system of equations describing the process circuit is partitioned into a set of irreducible subsystems of simultaneous equations, it is usually desirable to further decompose these blocks of equations in order to simplify their solution (Himmelblau & Bischoff,

---

[5] 'Large' is not a precisely defined concept. Current methods have, for example, proved effective in solving mildly non-linear systems with up to 10 000 variables, while difficulties have been encountered with badly non-linear systems with as few as 100 variables (Shanno, 1983). For the purposes of this discussion, a *large* system is thus considered to be any type of system that is difficult to solve in practice, either as a result of the dimension or the structure of the system.

1968; Motard & Westerberg, 1981). The decomposition of such an irreducible subsystem is known as *tearing* and in terms of a digraph representation of a process circuit, a tear set is a set of edges whose removal leaves each vertex connected to another at most by paths going only in one direction. The objective of tearing is to reduce the computation time needed to solve the entire set of system equations simultaneously.

C.1 CURRENT METHODS FOR THE SELECTION OF TEAR SETS

One of the simplest criteria for the selection of tear sets, is to minimize the number of streams in the set. Another is to allocate weights to streams in proportion to the number of variables they comprize and to minimize the weighted sum of the tear set, i.e. involve the least number of variables, rather than streams (Mah, 1983).

Murthy and Husain (1983) described a tearing algorithm that works directly on the digraph of the process system. Their algorithm was designed to search for the cut set that would best minimize the sum of weights assigned to the process streams. They proposed the assignment of weights so that the tearing of input streams would affect the least number of output streams.

Upadhye and Grens (1975) argued that a superior alternative entails the requirement that the tear set belongs to a non-redundant family of sets. A tear set is considered to be non-redundant if it does not contain multiple tears. Their claim is supported by their own results, as well as those of Rosen and Pauls (1977). This strategy was developed further by exploiting the geometrical characteristics of non-redundant tear sets (Motard and Westerberg, 1981). When flowsheets are constrained, such as by the imposition of design specifications on process units, the character of the original unconstrained flowsheet may be altered, so that this approach to decomposition is not necessarily the best one to follow (Lau & Ulrichson, 1992).

The decomposition of process circuit by means of neural nets is not discussed in detail here, because the principles are demonstrated in section C.3 by means of a simple example.

C.2 THE USE OF A RECURRENT NEURAL NET TO DECOMPOSE CIRCUITS WITH RECYCLE STREAMS INTO SERIAL UNITS

Process circuits can be represented in several different ways, one of which is by means of a loop or recycle matrix, $\mathbf{A}^*$ in which the elements $a_{i,j}^*$ of the matrix are defined by $a_{i,j}^* = 1$, if stream s_j forms part of the loop l_j, otherwise $a_{i,j}^* = 0$. In this configuration it can be shown that the optimal decomposition of the circuit is equivalent to a zero-one optimal covering problem. These problems can be solved by means of linear programming feedback neural nets.

The construction of a linear programming neural net is depicted in figure C.1. The net is composed of two sections, one of which represents the objective function of the programming problem and the other of which represents the constraints of the system. The objective function section of the analog version of the net contains n amplifiers (one for each independent variable v_k in the objective function), each of which is fed a constant input current of $a_k^{\#}$. The k'th amplifier in this section has an input capacitance of $c_k^{\#}$ and an equivalent resistance $r_k^{\#}$. These amplifiers are furthermore assumed to satisfy the hard limiting non-linear input-output relation

$$v_k = g_v(u_k), \text{ where}$$

$$g_v(u_k) = 1, \text{ if } u_k \geq 0, \text{ and}$$

$$g_v(u_k) = 0, \text{ if } u_k < 0 \qquad \text{(C.1)}$$

The output of these amplifiers can thus only assume the values 0 or 1, and represents the solution to the optimization problem.

The constraint section of the net contains m amplifiers (one for each problem constraint) arranged in the same way as those found in the objective function section. Each of these amplifiers is provided with a constant bias current $b_k^{\#}$ as input, and also receives an input current from the amplifiers in the objective section. The output of the amplifiers $c_j^{\#}$ in the constraint section is an indication of the extent to which the constraints of the system are satisfied, i.e.

$$c_j^{\#} = g_c(u_j), \text{ where}$$

$$u_j = \mathbf{D}_j\mathbf{v} - b_j^{\#}, \text{ and}$$

$$g_c(u_j) = 0, \text{ if } u_j \geq 0$$

$$g_c(u_j) = 1, \text{ if } u_j < 0 \qquad \text{(C.2)}$$

These equations indicate that when the j'th constraint is violated, the j'th amplifier in the constraint section feeds a current proportional to $d_{i,j}$ to the i'th amplifier in the objective function section. This current becomes zero when the constraint is satisfied.

Prior to stabilization of the net, the amplifiers in the objective function section attempt to minimize the values of the output variables v_k, by pushing them to zero. At the same time the amplifiers in the constraint section attempt to satisfy the system constraints by pulling the output of the corresponding amplifiers in the objective function section through the injection of current of opposite polarity into these amplifiers. At equilibrium the output of the amplifiers in the objective function section represents the optimal solution to the problem, subject to the constraints imposed on the system.

The neurodynamics of the net can be expressed by equation C.3,

$$c_i^{\#}du_i/dt = -\varepsilon_i^{\#} - u_i/r_i^{\#} - \Sigma d_{ji}.g_v(\mathbf{D}_j\mathbf{v} - b_j^{\#}) \qquad \text{(C.3)}$$

The energy of the system can moreover be expressed as

$$E = \mathbf{D}.\mathbf{v} + \Sigma G(\mathbf{D}_i\mathbf{v} - b_i^{\#}), \text{ where } g(.) = dG(.)/d(.) \qquad \text{(C.4)}$$

Provided that the transfer function $g_v(u_j)$ is bounded and monotonically increasing, these dynamics minimize the Lyapunov function represented by equation C.4 (Tank & Hopfield, 1986).

C.3 EXAMPLE C.1

In this example the use of a linear programming net to optimally decompose an elementary process circuit into sequential subunits containing no recycle streams is demonstrated. Consider the process circuit depicted in figure C.2, which was also used by Pho and Lapidus (1973) to demonstrate the use of an iterative tearing algorithm. The corresponding loop matrix of the system, which consists of five process units connected by 10 streams, is shown in table C.1. The optimal

decomposition of this system can be formulated in terms of a zero-one optimal covering problem, i.e.

$$\min \Sigma p^{\#}(s_j).x_j, \text{ subject to}$$

$$\Sigma a_{i,j}{}^{*} x_j \geq 1,\ i = 1,2,..M \text{ and } x_j = 0 \text{ or } 1 \qquad \text{(C.5)}$$

The structure of the neural net used to decompose the system is similar to that depicted in figure C.1.

TABLE C.1 LOOP MATRIX OF PROCESS CIRCUIT (EXAMPLE C.1)

(a)	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
l_1	1	1	0	0	0	0	1	1	0	1
l_2	0	0	0	0	1	0	1	1	0	1
l_3	0	1	0	1	0	0	0	1	0	1
l_4	0	1	1	0	0	0	0	1	0	0
l_5	1	1	0	0	0	1	0	1	0	0
l_6	0	0	0	1	1	0	0	1	0	0
l_7	0	1	0	1	0	0	0	0	1	0
l_8	1	1	0	0	0	1	0	1	0	0
l_9	0	0	0	0	1	0	1	0	1	0

(b)	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
l_1	0	1	0	0	0	0	0	0	0	0
l_2	0	0	0	0	1	0	0	0	0	0
l_3	0	1	0	0	0	0	0	0	0	0
l_4	0	1	0	0	0	0	0	0	0	0
l_5	0	1	0	0	0	0	0	0	0	0
l_6	0	0	0	0	1	0	0	0	0	0
l_7	0	1	0	0	0	0	0	0	0	0
l_8	0	1	0	0	0	0	0	0	0	0
l_9	0	0	0	0	1	0	0	0	0	0

The output of the net after four iteration steps is shown in table C.1(b).

The use of linear programming neural nets such as the one described in this appendix can subsequently be used to generate optimal tear sets in partition blocks of arbitrary size and can be used in conjunction with the data reconciliation systems described in chapter 4 to accommodate large systems.

FIGURE C.1 A LINEAR PROGRAMMING NEURAL NET

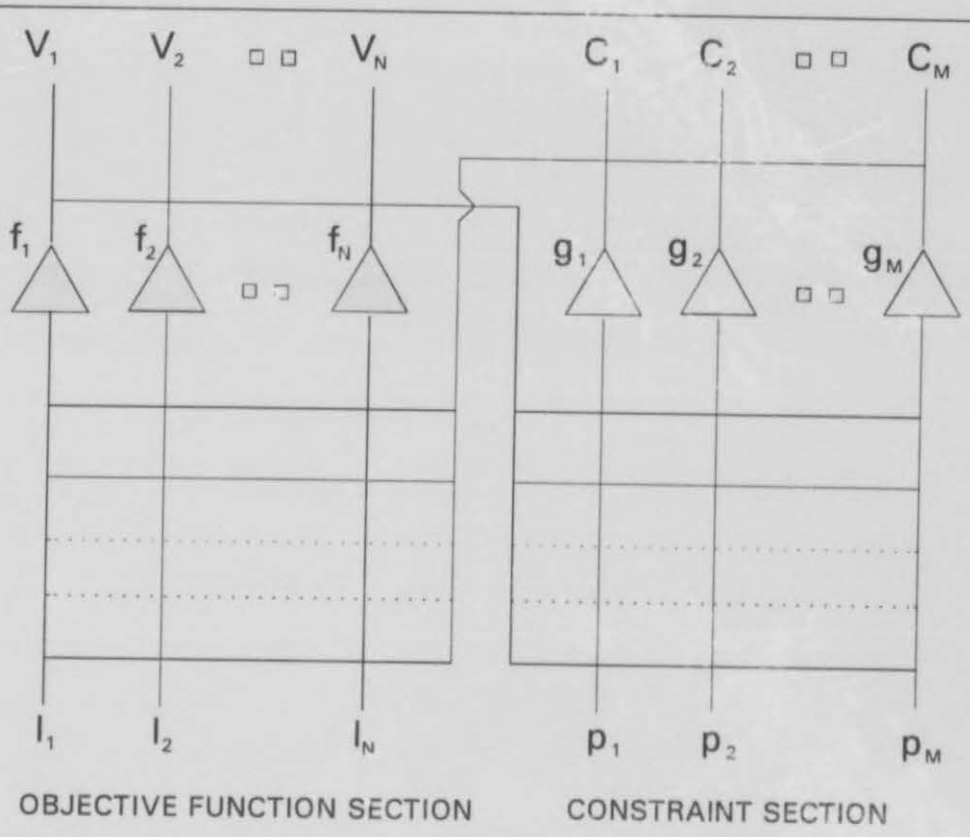
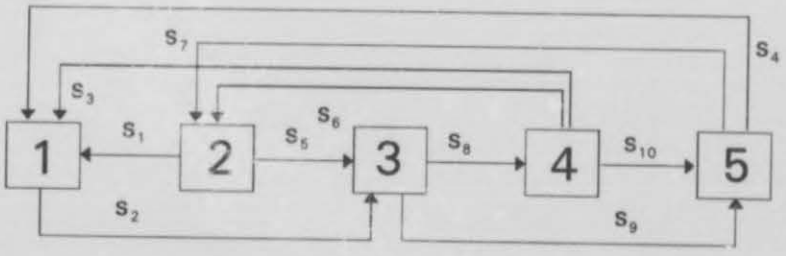


FIGURE C.2 RECYCLE CIRCUIT USED IN EXAMPLE C.1 (Pho & Lapidus, 1973)



APPENDIX D
PAPERS BASED ON THIS DISSERTATION ACCEPTED FOR PUBLICATION

Aldrich, C. & Van Deventer, J.S.J. The identification of systematic errors in steady state mineral processing systems by means of back propagation neural nets. *International Journal of Mineral Processing* (accepted for publication).

Aldrich, C., Van Deventer, J.S.J. 1993 & Reuter, M.A. The application of neural nets in the metallurgical industry. *Minerals Engineering*, (accepted for publication).

Aldrich, C. & Van Deventer, J.S.J. Identification of gross errors in material balance measurements by means of neural nets. *Chemical Engineering Science* (revised version submitted).

Aldrich, C. & Van Deventer, J.S.J. Estimation of measurement error variances in quasi-steady state process systems by means of neural nets. *Computers & Chemical Engineering* (revised version submitted).

Aldrich, C., Van Deventer, J.S.J, Van der Walt, T.J. & Reuter, M.A. 1993. Recent advances in the simulation of mineral processing circuits using artificial intelligence. *Proceedings of the XVII International Mineral Processing Conference*, vol 2, pp 529-537, Sydney, Australia, 23-28 May 1993.

Aldrich, C. & Van Deventer J.S.J. 1993. New computational methods for the application of connectionist networks to mineral processing. *Proceedings of the XXIV Applications of Computers and Operations Research in the Mineral Industries*, Montreal, Quebec, Canada, 31 Oct - 3 Nov 1993 (accepted for publication).