Dynamic Modelling of a Falling-film Evaporator for Model Predictive Control

A thesis presented in partial fulfilment of the requirements
for the degree of Doctor of Philosophy in Technology
at Massey University

Nigel Thomas Russell

October 1997

Abstract

Fundamentally this thesis provides a study into dynamic modelling of a pilot-scale falling-film evaporator. Its main aim, however, is to research and demonstrate artificial neural network (ANN) modelling for model predictive control as applied to an evaporator system.

As a prerequisite to testing an advanced control strategy such as model predictive control one must have available a suitable dynamic simulation of the process. To this end the thesis initially presents the formulation of a dynamic model of the evaporator system developed from first-principles.

A novel approach to developing a dynamic ANN representation of a process is presented and applied to the evaporator. This approach incorporates prior knowledge of the system into the network topology to attain a model with a flexible, modular structure. A dynamic, recurrent training methodology is devised to enable the ANN model to predict over a future horizon of arbitrary length.

The performance of the modular ANN model is compared with a linear model of a similar form identified through conventional linear regression methods. It was found that the nonlinear ANN model and subsequent nonlinear MPC scheme exhibited no improvement in performance to that of their linear counterparts.

Errata

p. 31	paragraph 4, line 1 should read:	"...with an evaporation rate..."
p. 42	paragraph 4, line 2 should read:	"...shows how the energy..."
p. 46	paragraph 1, line 1 should read:	"...A_l is a function of L..."
p. 47	paragraph 7, line 2 should include:	"...l_e/d..." and not "...L_e/d..."
p. 83	paragraph 4, line 2 should read:	"...by the model as a function of..."
p. 104	paragraph 1, line 1 should read:	"...then the neuron will turn on..."
p. 105	paragraph 2, line 2 should read: paragraph 3, line 2 should read:	"...are adapted it is said that..." "...on opposite sides of a hyperplane..."
p. 118	paragraph 3, line 5 should read:	"...a number of times..."
p. 157	paragraph 2, line 5 should read:	"...foolhardy..."
p. 166	paragraph 4, line 4 should read:	"...was selected for study..."
p. 168	paragraph 2, line 1 should read:	"...of the effect requires four input variables..."
p. 169	paragraph 8, line 2 should read:	"...but this was considered to short."
p. 190	paragraph 1, line 3 should read:	"...output of neuron 2 has an inverse..."
p. 199	paragraph 2, line 7 should read:	"...individual errors are not as good..."
p. 231	paragraph 1, line 3 should read:	"...occur at a time of 1 minute and eight minutes on the plots."
p. 234	Figure 8-6:	in the legend the "PI-control" and "NN-MPC" labels have been interchanged.

Thesis Contribution

This thesis has made the following contributions:

Analytical modelling

1. The development of an analytically-derived, dynamic model of a falling-film evaporator pilot-plant which is an extension of earlier modelling work.

   The major extensions to the evaporator model include:

   - the re-formulating of the model for a feedforward configuration,
   - the deriving of equations for the preheater system,
   - the deriving of equations for the venturi condenser sub-systems, and
   - the inclusion of an additional product temperature state variable.

2. The combining of the analytical sub-models for three evaporation effects and the additional sub-systems to describe the complete evaporator system.
3. The validation of the complete analytical model against data from the actual plant.

Artificial neural networks

4. The conception of a modular, artificial neural network modelling approach for the development of models to be used within a predictive control strategy.
5. The demonstration of neural network structure selection based upon prior knowledge of the system to create locally-connected, modular models.
6. The application of a dynamic, recurrent training methodology combining backpropagation through time and the Levenberg-Marquardt optimisation method to train the neural networks.
7. The comparison of a modular, artificial neural network model and an equivalent linear model identified through conventional regression techniques.

Model Predictive Control

8. The demonstration of the modular artificial neural network within a model predictive control strategy applied to the evaporator simulation.

Acknowledgements

I would like to sincerely thank my supervisors Dr Huub Bakker and Dr Bob Chaplin for their expert guidance and input during my PhD study. I am very appreciative of the time, encouragement and many thoughtful comments they offered to assist me in my research.

I would also like to thank Huub for his willingness to assist in funding my trip to attend the 1996 conference on Engineering Applications of Neural Networks in London. This excursion proved to be a very rewarding experience and I am very appreciative of Huub's generosity.

Special mention must also be made of Professors Mary and Dick Earle for the generous support they provided in the initial stages of my PhD degree. This and their on-going interest in my progress has been very much appreciated.

I would like to thank the many staff and postgrads with whom I have had the privilege to become acquainted during my study within the Production Technology Department. Together they have made my time at Massey very enjoyable and I will have many fond memories of a variety of departmental academic, social and sporting activities.

Thanks also to my family for all their support and encouragement, particularly my mother and father who have given me the opportunity to achieve what I have to date. I'd like to thank them for all their generosity to me during my university days.

I am extremely grateful to my wife Claire for all that she has been for me. I am very pleased that she was with me during my PhD 'journey' and I love her very much. Her unfailing support and continual belief in me has been an invaluable source of strength. This thesis truly belongs to her also.

Finally, I would like to thank God for everything he has provided me and enabled me to experience. He is the source of all good things. These last few years, although at times a struggle, have certainly been a 'good thing' – thank you.

Table of Contents

1

Introduction

1.1 Background

1.1.1 Advanced process control

During the last two decades significant progress has been made in the area of process control. Traditionally, linear single-input controllers have been used in industrial process control and they are still very common. But these often fail to provide adequate control over a wide operating range and do not adequately handle multivariable systems.

Processes are becoming larger and increasingly complex and are demanding more sophisticated controllers to regulate them. International competitiveness has also required that companies make more efficient use of resources and energy within their plants. *Advanced control* has become the term applied to the modern direction in control systems development. Advanced control strategies have been predominantly applied in the military, aerospace, automotive and some critical process industries where commercial and technical competition is intense [1]. In the future, as global competitiveness increases, more process industries will need to adopt the same attitude towards control as is experienced in these 'high-tech' fields. Rather than employing advanced control to only safety-critical processes, the emphasis will become one of routinely applying it to most plants in order to remain competitive. In such an environment, energy and resource efficiency will become just as important as the quality of the product.

Advanced control generally refers to controllers which employ an element of intelligent processing within their structures; predominantly this is realised through the use of dynamic process models. In many cases these models have been linear approximations of theoretical engineering models as in optimal control and robust control.

Empirical models based on the plant data have also become popular within model-based control strategies such as; model reference adaptive control, inference control, internal

model control and model predictive control. Such schemes generally have made use of linear step-response or impulse-response models derived from on-line experiments with the actual process.

The emergence of advanced control has combined the fields of system identification and process control so that the modern control engineer requires a sound understanding of the principles of system identification in order to be successful.

The control strategies may have advanced but the challenges facing control engineers remain the same. The main difficulties involved with controlling industrial processes are the inherent nonlinearities and the multivariable nature of the systems. The emphasis for advanced control is on developing strategies which are multivariable in nature and are robust in the face of modelling errors and unmodelled disturbances.

Through the use of nonlinear models nonlinear, advanced control is realisable. However it has been generally acknowledged that the lack of suitable nonlinear process models has been a stumbling block to the rapid spread of the application of advanced control to real industrial processes [2]. With the advent of artificial neural network techniques (§1.1.2) the cause of nonlinear model-based control systems has been significantly advanced.

Model Predictive Control

Model Predictive Control (MPC) has become one of the more popular and successful model-based control strategies. In terms of implementation in real industrial processes advanced control had little impact prior to the emergence of MPC. Developed initially within the petrochemical industry [3], MPC has proved to be very successful for a variety of applications [4, 5].

MPC is a receding-horizon control strategy and aims to optimise the control moves over a future horizon based on a desired control objective function. The optimisation routine interacts with a dynamic model of the plant to select the set of future control moves.

Using a constrained optimisation approach enables the plant constraints to be easily included into the control design. This is one of the main benefits of MPC.

Linear MPC is rapidly becoming a standard control methodology in the petrochemical industry and is finding successful application in other industries. The control benefits achieved with linear MPC have proved to be very attractive economically with pay back periods in terms of months [4, 5]. It is believed that nonlinear MPC will provide further control improvements. Artificial neural networks are ideally suited for incorporation with MPC to produce constrained, nonlinear control. As yet there are relatively few industrial implementations of artificial neural network-based MPC.

1.1.2 Artificial neural networks

Artificial neural networks (ANNs) use a parallel processing paradigm and attempt to emulate the example data to which they are presented. They are black-box models in which the parameters can be adapted to model a wide class of nonlinear systems. They are given the name *artificial neural networks* since the method of adaptation has been inspired by the learning processes used in biological nervous systems.

Over the last decade there has been a flurry of research activity into ANNs and their applications. This was initiated by the work published by Rumelhart *et al* [6] in the area of connectionist learning which revived the earlier work of Rosenblatt [7] and Widrow & Hoff [8] in the 1960's.

Since then interest in the technology has grown exponentially and the range of applications for which they have been considered is wide and diverse. This is not surprising considering most of the real-world engineering systems are inherently nonlinear and that in the past there has been a shortage of effective tools to tackle many problems. The huge growth in the ANN field is also due in part to the increasing accessibility of powerful computing capabilities.

Artificial neural networks have been applied to problems in fields as diverse as; image recognition, fraud detection, process control and optimisation, and economic forecasting [9]. The large number of conferences, journals and publications dedicated to the theory and applications of ANNs that have emerged over the last few years is a testament to the growth in popularity of the technique.

Along with their nonlinear capabilities their apparent ease of development over conventional modelling methods and the lack of a reliance on theoretical knowledge are among the main attractions of the technology.

Initially there appeared to be an element of 'hype' surrounding ANNs and their capabilities due to the belief that they could be a panacea to many engineering problems [10]. Certainly the label 'artificial neural networks' helped to spark massive interest as people imagined the prospect of developing artificial brains. This type of concept is largely unhelpful to the serious advancement of the technique.

With the initial interest in ANNs many interested companies quickly attempted to use them without much knowledge of what really was involved. It rapidly became apparent that many of the much-vaunted abilities of the technology were not as great as had been proclaimed. Much of this was due to the unrealistic expectations of many who did not fully appreciate how ANNs worked and expected amazing results. There is very little that is remotely human-like in ANNs; they have their limits and problems just like other black-box modelling techniques do. Many companies became disillusioned as the

technology failed to live up to its reputation and it was quickly discarded. "We've tried them and they don't work" would be a common response [11].

However much has been learnt about the strengths and weaknesses of ANNs since their initial appearance. Many new methods of developing ANNs have been researched and faster and more powerful computing capabilities are now available making their application more viable.

Realistically, ANNs should not be viewed as magical black-boxes but as important engineering tools on a par with other intelligent technologies such as Fuzzy Logic, Genetic Algorithms and the like.

The understanding that ANNs by themselves cannot solve problems has emerged and the use of neural networks as modules embedded within a larger system is common [9]. This emphasises the point that ANNs appear to perform smaller tasks well and can be used as building blocks within a larger solution to replace conventional modelling methods.

ANNs are an important tool for practising engineers and it is believed that as computing capabilities become more advanced, they will continue to have an important role to play in a wide range of applications. However only through successful industrial application will the true value of the technology be realised.

The development of nonlinear process control is one of the logical areas to which ANNs have made a contribution. Although the application of ANNs to control has yet to become widely accepted in practice, much has been written in engineering literature [12, 13, 14, 15] (also see §4.5). What has been reported is very promising but the studies are predominantly based on simulations or small-scale equipment and it is not known whether the success will carry over into real industrial applications. Reports of truly industrial applications of neural control have been slow to emerge. Despite the current lack of widespread industrial application, ANNs have nevertheless provided an enormous boost to the research of nonlinear control systems.

ANNs have a number of attractive features but they also produce additional problems. For example, the fact that ANNs are data-driven models and can be developed without the need for accurate theoretical knowledge raises the problem of obtaining a sufficient quantity of representative process data.

The collection of a large amount of data that spans an acceptable operating range can be particularly difficult to obtain from a production process. Large variations in inputs may not be feasible without disrupting the operation of the process and subsequently affecting the quality of the product. This problem can be overcome to some extent

through the use of historical data collection facilities which can store large amounts of plant data.

Theoretical issues such as model stability, robust model validation and formal guidelines for structure determination remain as obstacles to their acceptance by many engineers both within academia and industry.

Once a network is developed there still exists a lack of understanding of just how they work. Exactly how the networks arrive at their outputs and to what extent these outputs can be relied upon are questions that need to be addressed.

Much emphasis has been placed on resolving these theoretical issues. Some progress has been made but, because of the nonlinear, flexible and adaptive characteristics of these models, sound theoretical answers may never be formulated. It is likely that ANNs will always require a significant amount of trial and error in their design and development and doubts over their robustness may always remain.

1.1.3 Process of study - a falling-film evaporator

The process system which is the subject of this thesis is a pilot-scale, three-effect, falling-film evaporator resident in the Department of Production Technology, Massey University. This plant is a scaled version of the type of process found in the dairy industry and was installed to provide a platform for research into process identification and advanced control within the Department.

The evaporation process is a good example of a nonlinear process that requires accurate control. A large proportion of energy used in industry is given to drying processes and evaporators play a large role in the industrial drying of a number of products. It is a very relevant process to study due to its widespread use, particularly in New Zealand with its large number of dairy processing sites.

Evaporators in the dairy industry are a key process in the production of milk powders. Good evaporator control is particularly important since evaporators are usually located directly upstream from an energy-intensive spray drying process. Tight control on the evaporator leads directly to better control in the dryer which results in better energy efficiency and a more consistent product.

Although MPC has become common in the petrochemical industry it has yet to become prominent in the 'lighter' process industries like food and dairy plants. However it is known that some New Zealand dairy companies are investigating the use of MPC for evaporator control. This highlights the relevance and importance of research into nonlinear MPC for evaporators.

1.1.4 Analytical, dynamic modelling

In developing advanced control a process simulation should be used to test and refine the performance of the strategy. This is an important step in remedying any problems before the control is implemented on the real system. Additionally a process simulation allows the performance of different control schemes to be compared easily.

The key reason for using a simulation is to refine the control scheme as much as possible without disrupting production on the real process and jeopardising equipment or safety.

For this reason the first task in the process of forming a nonlinear MPC strategy for the pilot-plant evaporator would be the development of an analytically-derived dynamic model of the system.

The ultimate application of the dynamic model largely determines the method of analysis of the system and the complexity of the final model. Models for the purpose of design or analysis of the process mechanisms often require distributed models containing partial derivative information. On the other hand models for control system design do not require as much complexity and lumped-parameter analyses usually suffice.

The dynamic model developed for the evaporator pilot-plant is mainly for use as a control design tool whereby the plant model can be used to simulate and compare various control strategies. For this reason the model contains only ordinary differential equations and algebraic relationships and parameters have been lumped where appropriate.

Other reasons to form a process simulation include:

- Process optimisation. Through investigation of the process simulation one could determine the optimal settings that should be applied on the actual plant
- Testing operating scenarios. Various operating scenarios could be run without danger of disrupting the actual process. This could benefit the process engineer who wishes to determine the effects of applying different operating strategies, using different raw materials or making physical alterations to the plant.
- Operator training. Simulation models could provide off-line training to operators and an understanding of how the characteristics and responses of the process can be gained.

- Process monitoring tool. Generally a model represents a perfect system. Any deviations the plant makes from the model may indicate areas where the process needs attention, for example it may indicate when fouling is becoming a problem.

Currently an evaporator modelling project is being undertaken within a New Zealand dairy company [16]. It aims to develop a simulation tool of a large multi-effect evaporation system in order to give the process engineers and operators a better understanding of the characteristics of the evaporator. The research project is partially funded by industry which again highlights the importance placed on evaporation processes within industry. The approach taken to modelling the industrial evaporator is similar to that used in this thesis.

1.2 Objectives

This thesis seeks to develop an ANN development methodology for MPC models which addresses some of the issues raised in the previous section.

In this respect the thesis does not attempt to provide any new optimisation algorithms for creating ANNs which may be faster or more likely to converge to a solution. Instead this work attempts to combine a number of established ANN techniques to provide a framework for obtaining good ANN models for use in MPC. It is hoped that this framework would assist an engineer to develop a meaningful and accurate model of a process system. The approach demonstrated seeks to both simplify the development of the ANN and enhance one's understanding of how the completed network functions. It is hoped that this approach will lead to ANNs which are not merely black-boxes but rather models which have a certain amount of transparency built into their structure.

In the area of ANNs this work has the following aims:

- To develop a modelling methodology which finds a compromise between industrial practitioners who tend to want quick and relevant solutions to problems (especially modelling ones) and academic circles where the focus tends to be on developing solutions for which the integrity and functionality of the solutions can be analysed.
- To combine the processing and development advantages of ANNs with the conventional theoretical engineering knowledge.
- To demonstrate a method of ANN development specifically for use in Model Predictive Control strategies where output predictions are required over a future horizon of arbitrary length.
- To demonstrate nonlinear Model Predictive Control with a modular ANN.

Before setting about achieving the above objectives, this work also aims to:

- Formulate an analytically-derived model and simulation of the evaporator pilot-plant for use in control system development. This model will be used in the simulation of the ANN-based MPC.

To perform a comparison of the ANN model with a conventional modelling technique, a linear regression model is developed with a similar input/output structure to the ANN. The prediction and control performance of both models are compared.

1.3 Overview of thesis

The research undertaken for this thesis spans three broad, engineering subject areas; process engineering, artificial neural networks and control engineering.

In this work the topics relating to these subject areas are individually developed before being combined in the final stages of the thesis.

The first two chapters of the thesis focus on the evaporation process that is under study and the development of a theoretically-derived simulation.

The following two chapters introduce the relevant theory and background of ANNs and MPC which forms the necessary foundation for the development of an ANN representation of the evaporator. The development of the ANN evaporator model is the subject of the subsequent two chapters.

The penultimate chapter combines the work of the preceding chapters to demonstrate a simulated MPC evaporator control scheme before relevant conclusions are drawn in the final chapter.

The Appendices located after the main body of the thesis contain additional information regarding the work presented. Details of all the model development are given, as are comparison plots and results for the validation of each model. The software package MATLAB and its dynamic simulation environment SIMULINK [17] were the platforms used for the majority of the work. All the MATLAB programming files and SIMULINK block diagrams that were developed can be found in the Appendices.

1.3.1 Thesis by chapter

Chapter 2

Chapter 2 introduces industrial evaporators and provides a brief overview of their function and operation. Falling-film evaporators in particular are discussed. The energy considerations for evaporators are also outlined along with the key objectives in controlling such processes. The final section of the chapter introduces the Department

of Production Technology's pilot-plant falling-film evaporator. Its design, operation and instrumentation is described.

Chapter 3

Chapter 3 presents a detailed explanation of the analytical model that has been formulated for the pilot-plant evaporator. After the theoretical discussion the model implementation within MATLAB and SIMULINK is briefly described. The chapter then discusses the task of validating the model against the actual plant. The chapter concludes with a discussion of the performance of the analytical simulation model in characterising the pilot-plant.

Chapter 4

Chapter 4 presents an introduction to artificial neural networks, their development and their use in system identification and control. Feedforward neural network techniques in particular are discussed and some of the common algorithms are outlined. Issues relating to the design of neural networks and techniques for applying them to dynamic system identification problems are discussed. The chapter concludes with an overview of the current methods of applying neural networks to nonlinear process control.

Chapter 5

Chapter 5 gives a brief overview of the Model Predictive Control methodology. The concepts, theory and applications of Model Predictive Control are covered. It focuses on two particular methods, namely, Dynamic Matrix Control and Model Algorithmic Control. The chapter concludes with a discussion of the use and implementation of neural network models to produce nonlinear Model Predictive Control systems.

Chapter 6

Chapter 6 presents the development of a neural network model of the pilot-plant evaporator. The approach taken utilises prior knowledge to form a model with a modular structure. An overview of the use of prior knowledge, and of modular modelling techniques as applied to neural networks, is given before embarking on a description of the model formation. The potential benefits of applying such techniques are discussed and details of the training methodology used are also outlined. The modelling section of the chapter deals with the development of a model for a sub-section of the evaporator. The decisions made as to model variables and structure are included. Comparisons are made of neural networks with alternative structures and a multivariable, linear regression model.

Chapter 7

Chapter 7 extends the modular modelling approach devised in Chapter 6 to produce a neural network model of the complete evaporator system. The chapter includes analysis of the full evaporator model and comparisons with both an equivalent linear regression model and the analytical model of Chapter 3.

Chapter 8

Chapter 8 draws together the modelling work of the previous chapters to demonstrate the application of Model Predictive Control to the pilot-plant evaporator system. To simulate the MPC strategy the analytical model is used to represent the actual plant. The artificial neural network and linear regression models which were developed are embedded in the MPC scheme to perform the predictions. The performance of both MPC controllers are demonstrated for setpoint tracking and disturbance rejection.

Chapter 9

The ninth and final chapter summarises the main conclusions that can be drawn from the research and outlines recommendations for future work.

1.4 References

[1] **Benson, R.**, Process control-the future. *IEE Computing & Control Engineering Journal*, August 1997, pp. 161-166.

[2] **Leigh, J.R.**, Neural networks in process control. *IEE Computing & Control Engineering Journal*, May 1992, p. 104.

[3] **Culter, C.R. & Ramaker, B.L.**, Dynamic matrix control – a computer control algorithm. *Proceedings of AIChE National Meeting*, 1979, Houston, Texas, USA.

[4] **Richalet, J.**, Industrial applications of model based predictive control. *Automatica*, 1993, vol. 29, pp. 1251-1274

[5] **Austin, P.C. & Bozin, A.S**, Applying dynamic matrix control in the process industries. *Transactions of the Institute of Measurement & Control*. 1996, vol. 18, pp. 32-41.

[6] **Rumelhart, D.E., Hinton, G.E. & Williams, R.J.**, Learning internal representations by error propagation. In: *Parallel Distributed Processing*, vol. 1, Rumelhart, D.E. & McClelland, J.L. (Eds.), MIT Press, Cambridge, MA, USA, 1986, pp. 318-362.

[7] **Rosenblatt, F.**, The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958, vol. 65, pp. 386-408.

[8] **Widrow, B.& Hoff, M.E.**, Adaptive switching circuits. *1960 IRE Western Electric Show and Convention Record*, August 1960, part 4, pp. 96-104.

[9] **Hammerstrom, D.**, Neural networks at work. *IEEE Spectrum*, June 1993, vol. 30, pp. 26-32.

[10] **Willis, M.J.**, **Montague, G.A.** & **Morris, A.J**, Modelling of industrial processes using artificial neural networks. *IEE Computing & Control Engineering Journal*, May 1992, pp. 113-117.

[11] Discussions during an open forum session at the *International Conference on Engineering Applications of Neural Networks*, King's College, London, UK, 17-19 June 1996.

[12] Special issues on neural networks. *IEEE Control Systems Magazine*, April 1988, vol. 8; April 1989, vol. 9; April 1990, vol. 10.

[13] **Bhat, N.** & **McAvoy, T.J.**, Use of neural nets for dynamic modelling and control of chemical process systems. *Computers & Chemical Engineering.*, 1990, vol. 14, pp. 573-583.

[14] **Hunt, K.J.**, **Sbarbaro, D.**, **Zbikowski, R.** & **Gawthrop, P.J.**, Neural networks for control systems—a survey. *Automatica*, 1992, vol. 28, pp. 1083-1112.

[15] **Morris, A.J.**, **Montague, G.A.** & **Willis, M.J.**, Artificial neural networks: Studies in process modelling and control. *Chemical Engineering Research & Design (Transactions of the Institution of Chemical Engineers)*, 1994, vol. 72, pp. 3-19.

[16] **Winchester, J.**, **Marsh, C.** & **Bakker, H.**, Analytical dynamic modelling of falling-film evaporators. *Proceedings of Chemica '97 Conference*, 29 September - 1 October 1997, Rotorua, New Zealand, pp. PC49 - PC56.

[17] MATLAB and SIMULINK, *The MathWorks Inc.*, 24 Prime Park Way, Natick, MA., USA.

2

Industrial Evaporation and Falling-Film Evaporators

OVERVIEW

This chapter introduces industrial evaporators and provides a brief overview of their function and operation. Falling-film evaporators in particular are discussed. The energy considerations for evaporators are also outlined along with the key objectives in controlling such processes. The final section of the chapter introduces the Department of Production Technology's evaporator pilot-plant. This evaporator is a three-effect falling-film evaporator and its design, operation and instrumentation is described.

2.1 Industrial Evaporation

2.1.1 Introduction

Evaporation is the process of concentrating a solution containing dissolved or suspended solids by boiling off the solvent. There are a variety of evaporation methods and a wide range of industrial applications, however the predominant users of evaporators are the food industry.

The process has four major applications within the food industry [1]:

(i) Pre-concentration of a liquor prior to further processing, for example before spray drying, crystallisation, etc.

(ii) Reduction of liquid volume prior to transportation or storage in order to reduce costs.

(iii) To reduce 'water activity' in certain foods by increasing the concentration of soluble solids in order to aid preservation.

(iv) For the utilisation and reduction of effluent.

The number of food processes that use evaporators are extensive and include milk powder, fruit juice, salt and sugar processing [2]. In New Zealand evaporators are most commonly found within the dairy industry. These are used to pre-concentrate milk before spray drying it to produce milk powder.

Evaporators are also used in non-food related applications [2]. Here they treat effluent streams to recover useful resources (the processing of black liquor in wood pulp processing), remove major contaminants from a product (removing excess salt in caustic soda production), concentrate waste streams for easier disposal (nuclear reactor wastes), and are an important component in desalination plants (where seawater is evaporated to produce fresh water in the form of condensed vapour).

Heat transfer principles

An evaporation stage is referred to as an *effect*. Figure 2-1 illustrates the input and output flows for a general evaporation effect. An effect is fed a flow of dilute solution and heating steam. It brings these two flows into contact via a heating surface transferring heat from the hot steam to the cooler liquid. If heat losses are ignored then all the energy given up by the steam is transferred to the liquid solution as sensible heat and latent heat of vaporisation. The vapour produced as the liquid boils is separated from the liquid phase and drawn out of the effect. Generally the concentrated liquid is the desired product of the process.

The overall rate of heat transfer, Q, from the heating medium to the boiling liquid across the intervening wall and surface films is given by

$$Q = U A \Delta T \tag{2.1}$$

where U is the overall heat transfer coefficient of the boundary between the two media. This is dependent upon the physical characteristics of the wall, the presence of boundary layer films and any scaling or fouling present on the wall.

A is the total heat transfer surface area.

ΔT is the temperature difference between the heating medium and the boiling liquid.

An understanding of this equation is important for the design, selection and operation of evaporators. Designers of evaporation systems have to continually revisit this equation in order to optimise the heat transfer given constraints on the plant size, physical properties of the product and the temperatures used.

Greater heat transfer area incurs greater capital costs but allows smaller temperature differences and residence times for a given heat flow. Conversely a smaller area, while

cheaper to construct, requires a larger temperature difference and increased energy input, as well as the likelihood of damage to heat-sensitive products.

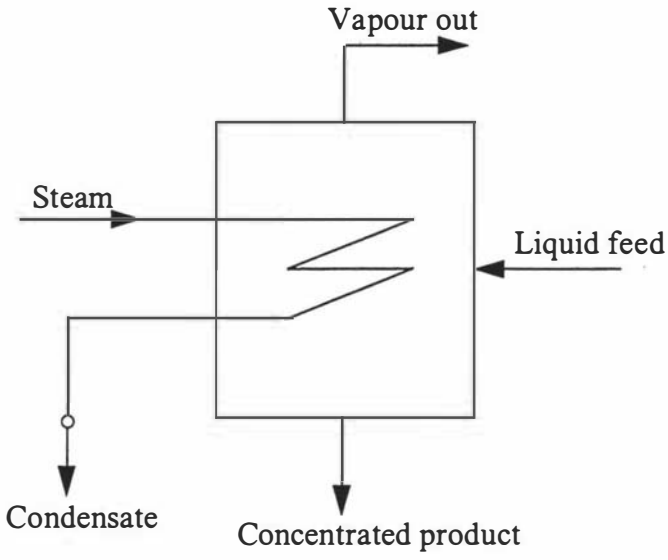


Figure 2-1: Inputs and outputs for an evaporation effect.

2.1.2 Types of evaporators

Before the advent of modern evaporation systems batch processing was widely used. Batch evaporation involved a single vessel in which the liquid was boiled until the product had the required consistency. Many undesirable changes in the properties of the product were experienced with these systems as they operated at atmospheric pressure and had long residence times. The sugar industry as the major user of evaporators during the 19th Century developed steam-heated continuous vacuum evaporators. The vacuum operation allowed lower boiling temperatures. In the latter part of the 19th Century the first multi-stage and vapour recompression evaporators were built improving the energy efficiency of the processes [2]. Today multi-stage evaporators with vapour recompression are widely used.

Evaporation processes involve a number of necessary elements which are together designed and constructed for a specific application. The essential components of continuous industrial evaporation systems are [1]:

(i) A heat exchanger to supply sensible heat and latent heat of evaporation to the liquor, this is usually via saturated steam.

(ii) A separator in which the vapour and liquid phases of the product are separated.

(iii) A condenser to condense the vapour and remove it from the system. This can be omitted if the process is operated at atmospheric pressure.

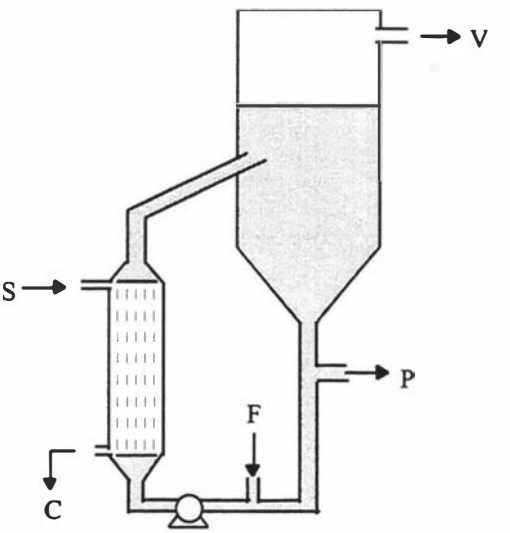
(iv) A preheater system to bring the product to the required boiling temperature.

These elements are usually of stainless steel construction which are easily cleaned and non-corroding.

Evaporators can be classified according to the size and shape of the heating surfaces and mechanisms used to pass the product over these surfaces. Most evaporators have a bundle of tubular heating surfaces called a calandria. Flat and conical heat exchange surfaces are also common. The circulation of the product past these surfaces can be induced through boiling, due to gravity (natural circulation) or by mechanical means. Which of the evaporation methods is used depends on the properties of the product, the physical size limitations on the process and the volume of product required to be processed. The main types of evaporators and their applications are briefly described below.

Forced-circulation evaporators

Forced-circulation evaporators use a pump to ensure the circulation of the product past the heating surfaces. Circulation is maintained regardless of the evaporation rate and so is well suited for crystallisation as the solids remain in suspension throughout. The heating tubes can either be horizontal or vertical, depending on the available headroom, and connect to a chamber where the heated liquid flashes off. The pump withdraws the liquor from the flash chamber and forces it through the calandria back to the flash chamber. If the heating tubes are situated far enough below the liquid level, they are then fully submerged and this prevents boiling in the tubes (Figure 2-2). Tube-boiling increases the possibility of solid deposits forming on the walls. Most forced-circulation evaporators are therefore submerged-tube type.



C = condensate, F = feed, P = product, S = steam, V = vapour.

Figure 2-2: Submerged tube forced-circulation evaporator.

Forced-circulation evaporators give high heat transfer coefficients, positive circulation and relative freedom from scaling at the expense of high operating costs and long residence times. They are best applied to crystalline products and corrosive or viscous solutions.

Horizontal tube evaporators

These are simple units which are used for small capacity operations. The steam flows through the inside of a series of horizontal tubes and the liquor is contained within a cylindrical shell. The liquor either submerges the steam tubes or more commonly is sprayed over them from above (Figure 2-3).

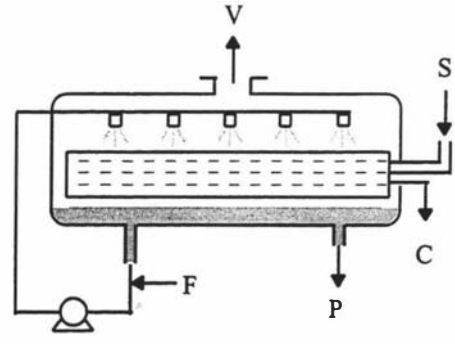


Figure 2-3: Horizontal-tube evaporator.

These types of evaporators require very little headroom, are low cost and provide good heat transfer. Horizontal tube evaporators are unsuitable for severely scaling or salting liquids, however bent-tubes can be used to reduce scaling problems. This type of evaporator is primarily used for seawater evaporation to produce fresh water where the condensed vapour constitutes the product.

Short-tube vertical evaporators

Short-tube vertical evaporators consist of a cylindrical evaporation chamber containing short vertical tubes near the base (Figure 2-4). The tubes are usually around 1.2 to 1.8 metres in length. The heating steam flows through the tubes and the liquor is fed into the chamber and flows upwards as it is heated. Often the steam tubes are placed in a ring around the chamber in order to produce a space in the centre for the liquor to circulate more freely. This allows the liquor to move up through the coils as it is heated and the colder product to pass down through the centre. If salting or scaling solutions are being processed it is best if the steam coils were fully submerged by the liquid level. Circulation of the liquor is entirely dependent on boiling although an impeller can be installed in the chamber to aid the circulation.

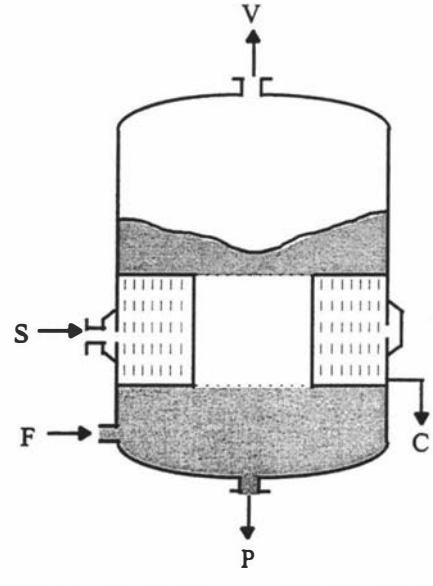


Figure 2-4: Short-tube vertical evaporator.

Short-tube evaporators are relatively inexpensive and do not require a lot of headroom. At high temperatures they offer good heat transfer but not so good at lower temperatures. The hold-up time of the product is relatively high. They are suitable for processing clear or non-corrosive solutions and crystalline solutions if an impeller is used.

Long-tube vertical evaporators

Long-tube vertical evaporators operate with the liquid product located inside long vertical tubes which are heated by the steam in a surrounding shell. The tube lengths are usually between 4 and 20 metres. There are two main types of long-tube vertical evaporators, they are *climbing-* (or *rising-*) *film* evaporators and *falling-film* evaporators. These are also often classified as film-type evaporators.

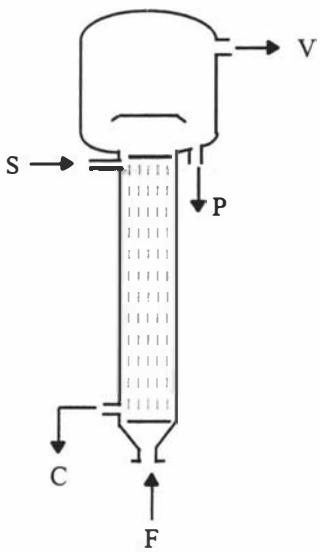


Figure 2-5: Long-tube vertical evaporator.

In climbing-film evaporators (Figure 2-5) the feed enters at the bottom of the calandria and starts to boil part-way up. The expansion on evaporation causes the vapour to lift a thin film of concentrated liquid up the tubes. The mixture of liquid and vapour leaves the top of the tubes and enters the separator and the concentrated product is drawn off. If the ratio of feed to the evaporation rate is low then it is desirable to provide recirculation of the product back through the evaporator.

Falling-film evaporators are essentially an upside down version of the climbing-film evaporator. Liquid is fed to the top of the evaporation tubes and flows down the walls as a film. As the liquid moves down it boils off and the separation of vapour and liquid takes place at the bottom. Some evaporators are arranged for the vapour to rise through the tubes counter-currently to the liquid.

The principle problem is the distribution of the liquid across all the tubes. If the tubes are not fully and evenly wetted then problems with scaling and reduced heat transfer can occur. This problem may require recirculation of the product.

Both climbing-film and falling-film evaporators offer large heating surfaces, which give good heat transfer with small temperature differences between the liquid and the steam. The falling-film evaporator has an added advantage that it has good heat transfer at both high and low operating temperatures. The simple design and compactness in floor space has made long-tube evaporators very popular in industry. Their short residence times (if recirculation is not used) are ideal for heat-sensitive products like orange juice and milk and as a consequence the food industries almost exclusively use falling-film evaporators in their processes. However, these evaporators require a large amount of headroom and are unsuitable for processing salting or severely scaling liquids.

Other evaporators

There are a number of other types of evaporators which include:

Plate evaporators

Plate evaporators consist of a series of gasketed plates mounted together similar to a plate heat exchanger. The evaporator operates a single pass with alternate plates containing steam and the liquid product. The product moves into the plates from the bottom of the unit and moves up between the plates operating on much the same principle as the rising-film evaporator. Alternatively the plate evaporator can be operated as a series of rising-film and then falling-film sections where the liquid moves up one plate and then down the next and so on with a steam layer between each section.

Centrifugal evaporators

Centrifugal evaporators, also known as *centri-therm* evaporators, consist of a stack of hollow, rotating conical elements. The steam is fed into each cone and the liquid product is sprayed over the cones as they rotate. The centrifugal force spreads the liquid thinly over the heating surfaces to provide rapid heat transfer. The concentrate accumulates around the outer edge of the cones and is displaced upwards.

Agitated thin-film evaporators

These evaporators employ a heating surface consisting of a large diameter tube in which the liquid product is spread over the inside wall by a series of rotating blades. These blades produce a thin film of liquid which is heated by steam surrounding the vessel. Similar to a falling-film evaporator the liquid film flows down due to gravity, becoming concentrated as it falls. This type of evaporator is especially suitable for viscous or heat sensitive products. The action of the blades also suppresses the formation of scale on the heating surfaces.

2.1.3 Multi-effect evaporation

The vapour driven off the liquid during evaporation contains considerable energy. In an evaporator with just a single effect this vapour is usually passed through a condenser and removed from the system. The energy efficiency of the process can be dramatically increased by re-using this vapour to evaporate the product further in another effect. In order to do this the boiling temperature in the next effect must be sufficiently low to maintain an adequate temperature difference for the transfer of heat. This can be achieved by operating the second effect at a lower pressure. The re-use of the vapour in this way can extend to a number of effects and is called *multi-effect* evaporation. For a

multi-effect system the first effect is at the highest temperature and pressure and the last effect is at the lowest temperature and pressure.

The first multi-effect evaporator was built in 1844 and used in a cane-sugar plant in Louisiana. Multi-effect evaporators arrived in Europe soon after, however before 1910 most evaporators in use were still single-effect operations. During the 1920's and 1930's double-effect evaporators dominated the market. Development continued on industrial evaporation and with improved materials and manufacturing costs, the first 6- and 7-effect evaporators were commissioned during the 1970's [3].

The purpose of multiple effects is to achieve better energy economy not increased plant capacity. The improvement in energy economy must be set against the capital costs of installing extra effects. The number of effects is also limited by the available overall temperature difference between the steam in the first effect and the condenser cooling water. The optimum number of effects must be arrived at by an economic analysis. A single-effect system requires approximately 1.0 to 1.3 kg of steam to evaporate 1 kg of water. Whereas a multi-effect system requires less than one kg of steam per kg of water as shown in Figure 2-6. These values of specific steam consumption can be considered as a rule-of-thumb [4].

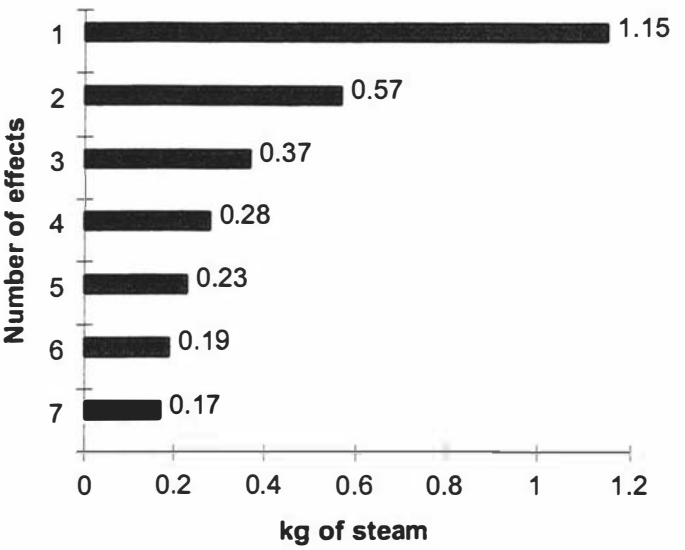


Figure 2-6: Average quantity of steam required to vaporise 1 kg of water in an evaporator.

Evaporators today normally have five to seven effects, which is considered optimum for most large-scale operations, although evaporators with as many as 17 effects have been built [5]. As more effects are included within a system the temperature and pressure differences between the effects decreases. Most of the evaporators described in the previous section can be used as multiple-effect systems.

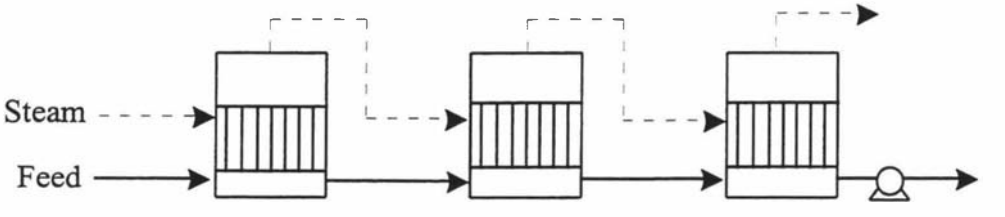
There are four methods of feeding the product to a multi-effect evaporator, they are forward, backward, parallel and mixed feeding systems.

Forward feed

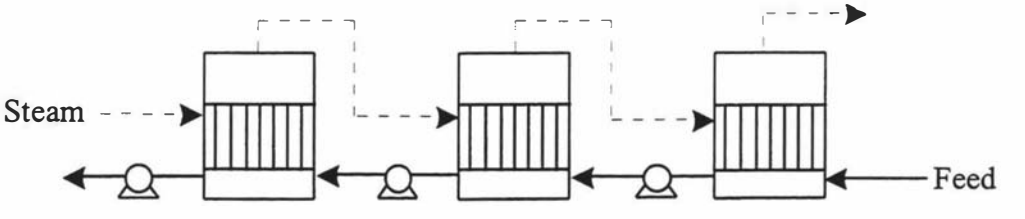
This is the simplest and most common method of feeding (Figure 2-7(a)). The product feed passes forward in the same direction as the vapours, from the first effect, to the second and so on. An extraction pump is only needed after the last effect as the feed can be transferred by the pressure difference between each effect. With this arrangement the viscosity of the product increases through the plant due to both an increasing concentration and a reduction in temperature. Thus the overall heat transfer coefficients are lower in the later effects. The lower temperatures are less likely to cause heat damage to the more viscous liquid however. If the initial feed is not at its boiling point then some of the raw steam will go toward raising the feed temperature and less vapour will be produced for the subsequent effects.

Backward feed

In a backward feed system the product is fed into the last effect and passes through the plant in the opposite direction to the vapours (Figure 2-7(b)). The raw steam is supplied to the first effect so the hottest and most concentrated product is evaporated by the hottest steam. This arrangement is advantageous in that less liquid needs to be heated to the higher temperatures which results in better steam economy. The increase in viscosity due to concentration is offset by the increasing temperature maintaining high heat transfer coefficients. Inter-effect pumps are needed to transfer the liquid. This arrangement is more common with viscous products.



(a) Forward feeding



(b) Backward feeding

Figure 2-7: Two methods of feeding a multi-effect evaporator.

Parallel feed

With this arrangement the product is fed into each effect in parallel. The vapour passes through each effect as in the previous configurations. This arrangement is commonly used with crystallising evaporators and avoids the pumping of dense slurries of product.

Mixed feed

Mixed feed systems are common with systems with a larger number of effects. It represents a compromise between the simplicity of forward feeding and the steam economy of backward feeding. In these arrangements the product is fed into a middle effect and passes down to the last effect before being transferred to the earlier effects for final concentration. They can also be used when a liquid product at an intermediate concentration and temperature is needed for further processing.

2.1.4 Vapour recompression

If the heat for the first effect is supplied via an external steam supply, this is known as direct heating of the system. For small-capacity evaporators direct heating is normally used. For larger systems, however, there are more efficient methods of using steam to heat the effects, namely *vapour recompression* methods.

Vapour recompression involves compressing the vapour generated in an effect to increase its temperature before returning it to the same effect as the heating steam. There are two methods of performing vapour recompression:

Thermal vapour recompression (*TVR*) - This uses a steam jet similar to a steam-jet ejector to increase the pressure of the vapour. This method can only compress part of the vapour from the effect so the remainder of the vapour is passed on to the next effects. The thermal compressor is relatively simple and inexpensive, with no moving parts and therefore has a long useful life.

Mechanical vapour recompression (*MVR*) - Recompression of the vapour is achieved via a mechanical compressor which is either powered by a combustion engine, electric motor or steam turbine. The exhaust vapour from an effect is fed to the compressor and then returned to the system. Only a small amount of make-up steam is required. Cooling water is used to condense some of the exhaust vapour to stabilise the plant. Evaporators using mechanical recompression have by far the lowest operating costs but have the highest capital and maintenance costs.

2.1.5 Auxiliary processes

Apart from the actual heat transfer mechanism evaporators require a number of additional components in order to operate. These elements will now be discussed.

Preheating

Unless the product has been heated in a previous process the first step is to raise the temperature of the liquid before feeding to the evaporator. The liquid should be raised to at least the boiling temperature of the first effect so that evaporation can begin immediately after entering the calandria. If it is not at the boiling temperature some of the heating steam for evaporation is wasted in heating the liquid to the boiling point.

The number of preheaters used depends on the scale of the process. The heat sources that can be utilised for preheating include the exhaust vapour from the final effect, the concentrated product outflow, the cooling water outflow from the condenser, the condensate flow from the shells of each effect and a raw steam supply. The order in which the product stream comes into contact with these heat sources depends on the quantity and heat capacity of each.

Four types of preheaters are used; plate heat exchangers, spiral heat exchangers, straight-tube heat exchangers and direct-contact heat exchangers. Which are used depend upon the product being processed and the balance between heat transfer area, temperature differences, residence time and costs of construction and operation. Straight-tube preheaters are commonly used however direct steam injection (DSI) systems have recently become popular. With direct steam injection raw steam is injected into the product stream to give a rapid increase in temperature as the vapour condenses in the liquid. One drawback with DSI is that the food-quality of the steam must be ensured.

Liquid-vapour separation

The separation of the two-phase mixture of vapour and liquid at the base of the calandria is another important operation as performance can alter the conditions in the next effect. The separator should have:

- high separation efficiency - the vapour should, as far as possible, be free from entrained liquid droplets so that the condensate flow in the next effect is free from product.
- small pressure loss - if the pressure loss is high the condensation temperature in the next effect is reduced, hence reducing the heat transfer in that effect.

- short residence time - the liquid volume in the separator should be as small as possible to minimise the product processing time.

The many different separators in use can be classified into three operating principles:

gravity separator - Separates the free-falling droplets from the vapour by gravity only. This method requires large separators as the vapour velocity has to be reduced until free-fall of the droplets occurs.

baffle separator - The vapour-liquid mixture rebounds off a vertical plate or baffle where the liquid droplets are separated by the impact. The less dense vapour is easily re-directed towards the next effect with the baffle. This method produces satisfactory separation.

centrifugal separator - This method achieves the best separation. The vapour-liquid mixture is led into the separator tangentially and the mixture spirals upwards. The acceleration of the rising vapour causes the centrifugal separation of the liquid droplets against the walls. Greater acceleration of the vapour provides better separation however acceleration causes pressure losses. Therefore the design of the separator is a compromise between good separation and small pressure drop.

Condensing and vacuum supply

The vapours produced in the evaporator are re-used where possible either to heat other effects or for preheating the feed. Surplus vapours must be condensed and the heat removed from the system. Condensers operate at the end of the evaporation process to maintain the heat balance in the system, stabilise the temperatures of the effects and to provide a vacuum for evaporator operation. Most condensers are either water-cooled surface condensers or mixing condensers.

Surface condensers have the cooling water stream completely separate from the condensing vapour, usually as a shell and tube or plate heat exchanger.

Mixing condensers have a spray of cooling water which passes through the exhaust vapours to condense them. This method requires less cooling water and is simple in design however, if any product has been entrained in the vapours, the cooling water may become contaminated and, if cooling towers are used, could cause fouling problems. For this reason surface condensers are more commonly used in industry.

Modern evaporation systems in the food industry are also operated under vacuum to reduce the boiling temperatures and so protect the food products from heat damage. This further improves the energy consumption of the plant.

The vacuum in the system is created by the condensing vapour in the condenser. Air may flow into the plant via leaks at flanges and connections and non-condensable gases may enter with the product. Additional measures are needed to remove these non-condensables from the system. Wherever gases may accumulate de-aeration pipes are provided to draw them off to the condenser. If not removed these gases interfere with the performance of the process by reducing the vacuum.

Alternatively, liquid-ring vacuum pumps and steam- or water-jet ejectors can be used as vacuum pumps. Jet ejectors are of simple design and reliable in operation as there are no moving parts. Water-jet ejectors can double as a mixing condenser, while the steam exhaust from a steam-jet ejector can be used to preheat the liquid feed. Liquid ring pumps while using less energy require more maintenance.

2.2 Falling-film evaporators

Since a falling-film evaporator is the subject of this thesis, a more detailed outline of the uses, operation and characteristics of falling-film evaporators is presented in this section.

As mentioned previously falling-film evaporators are in widespread use within the food and dairy industries; so much so that they have almost replaced all other types.

It is the

- relatively simple design,
- large heat transfer area,
- short residence times and
- good heat transfer capabilities

of falling-film evaporators that make them so popular.

2.2.1 Process description

The major elements of a falling-film evaporator effect is the vertical tube bundle with surrounding heating jacket (calandria), the distribution device at the top of the calandria and a separation vessel at the base (Figure 2-8).

Evaporation calandria

The liquid to be evaporated is pumped to the top of the effect where it is distributed evenly across all the evaporation tubes. The liquid flows down the tube walls as a thin boiling film. Ideally the product should be concentrated equally in all of the heating tubes and without the loss of product quality due to localised hot spots. The biggest

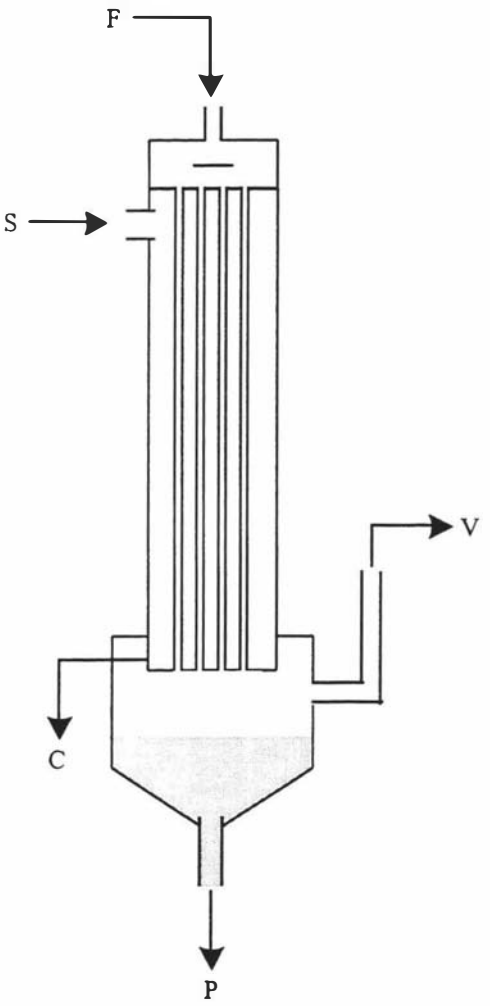


Figure 2-8: Falling-film evaporator effect.

obstacle to trouble-free operation of a falling-film evaporator is the liquid coverage of the heating tubes. If there is insufficient flow entering the tubes the liquid film is likely to break near the base of the tubes. If the distribution of liquid is not even across the tubes the liquid film may not even be formed at the top. If dry spots develop in the tubes the heating surface becomes coated with deposit and eventually the tube may become clogged with burned-on product (fouling). The minimum coverage of product is dependent on the type of product being processed. The length and diameter of the tubes must be designed to achieve a specific film thickness for a given flowrate. If necessary, recirculation of the product around the heating tubes can alleviate the problems of liquid coverage, however this is usually avoided as it can dramatically increase the average residence time of the product.

As well as tube coverage the velocity of the liquid-vapour flow is important. High velocities due to long, narrow tubes cause high pressure losses in the effect. This increases the boiling temperature and reduces the heat transfer rate. Moreover high velocity causes the vapour and liquid to spray off the end of the tubes making their separation more difficult.

Short tubes with large diameters would improve the velocity problems but would prove more difficult to fully cover with product. Hence the evaporation tube dimensions are a compromise between adequate coverage and pressure drop. Experience has shown that tubes between eight and twelve metres long with a diameter between 30 and 50 mm are successful [3]. These dimensions allow for many tubes to be packed together within the calandrias. This gives a large heat transfer area for each effect and hence temperature differences between the steam and liquid can be small.

Liquid distribution devices

As already discussed, the distribution of the product at the top of the evaporation tubes is critical. Two types of methods are employed in order to achieve liquid distribution. These are *dynamic* and *static* distribution.

An example of dynamic distribution is a cone nozzle in which the kinetic energy of the fluid is used to atomise the product into droplets and distributes them over the tube openings. This device is simple and inexpensive but does not guarantee an even coverage. To operate effectively the feed flow and pressure need to be steady in order to generate a uniform spray pattern from the nozzle.

Static distribution methods are generally preferred although they are more expensive to design and construct. Even distribution is achieved by forming a level of fluid above a plate over the tubes (Figure 2-9). Perforations in this plate allow the liquid to flow evenly into each of the tubes giving uniform wetting since the static pressure is equal over each hole. This method adapts to changes in feed flowrate by varying the height of the liquid level. The size of the distribution holes are designed in order to allow this level to form, given specified input flowrates. Otherwise the level may overflow or be emptied and starve the evaporation tubes of liquid.

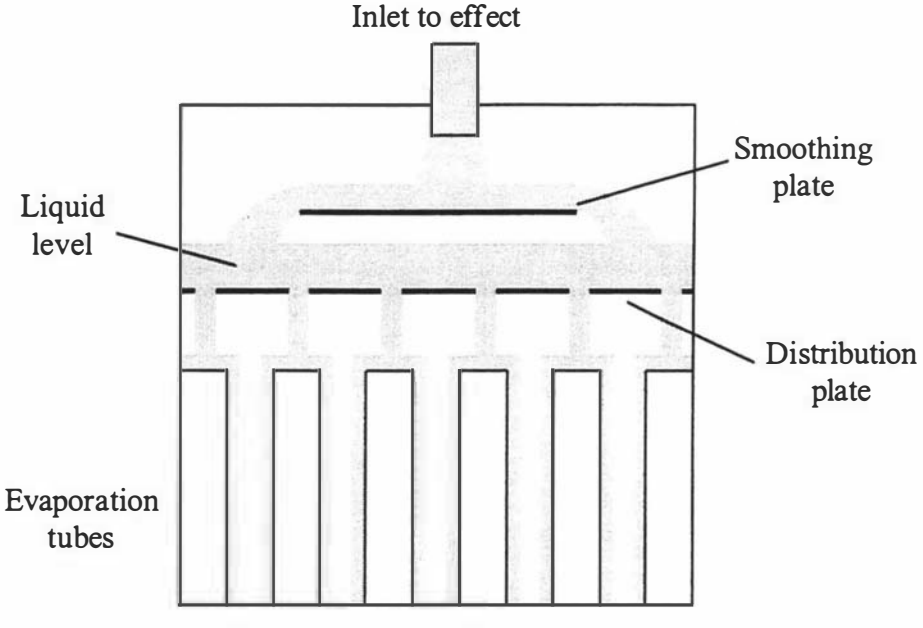


Figure 2-9: Static distribution device

Separators

In falling-film evaporation the liquid-vapour separation occurs at the base of the heating tubes. All three methods of separation (gravity, baffle and centrifugal) are applicable to

this operation and even a combination may be employed. In general however, centrifugal separators are most commonly used.

2.2.2 Industrial operation

As previously mentioned falling-film evaporators are widely used in industry. This section will concentrate on the application of falling-film evaporators in the Dairy Industry since in New Zealand they are the principle users of the technology.

Falling-film evaporators are used predominantly as pre-concentrators of milk before a drying process. Not all the water can be removed by the evaporator since, at a certain point, the total solids content of the product will make the viscosity too high. In dairy plants, for example, spray dryers are commonly employed to produce milk powder. In this type of dryer, concentrated milk of about 50% water is sprayed into a large chamber with a stream of hot air. This forces the remaining moisture to evaporate as the droplets fall. The powder exiting the spray dryer has a water content of around 11% and about 3% after the fluidised beds. Compared to the evaporation process the dryer consumes a lot of energy, about 5 to 10 times as much for the same amount of water removal [4]. So, from an energy-saving point of view, as much moisture as possible should be removed by the evaporator. In producing milk powder about 90% of the water removal takes place in the evaporation process and around 10% in the drying stages.

Modern large dairy evaporators process about 50,000 litres of milk an hour and have up to seven effects. The calandrias are generally about 20 metres high, have a diameter of two metres and contain in the order of 500 tubes each. To minimise energy usage it is now common for multi-effect evaporators to incorporate both thermal and mechanical vapour recompression on one or more effects. With this it is possible to achieve a specific steam consumption of around 0.08-0.13 kg steam per kg of water [1]. A typical process for concentrating milk has a steam usage in the order of 1000 kg/hr.

2.3 Evaporator control

2.3.1 Control objectives

Evaporators generally operate before a major drying process, which requires more energy and is more difficult to control. It is therefore important to achieve good control in the evaporation stage so that the drying process is operating with steady inputs. It is their influence on downstream processes which make evaporators vital to control. Considering the downstream process the two primary control objectives for an evaporation process are [7]:

i) to accurately regulate the concentration of the out-flowing product liquor

ii) to maintain the flow of product through the process at a desired level.

Both these variables have a large bearing on any downstream process. These objectives should be achieved while minimising the energy used and maintaining the product quality.

It is important to make good use of the energy that is put into the evaporator system. Most of the energy is from the external steam supply and, as has already been discussed, the energy efficiency of an evaporation system can be greatly improved by re-using the heat through multi-effect operation and vapour recompression. Operating under a vacuum also lessens the amount of heat required for the process. Other factors such as poor vacuum, air venting rates, air and water leaks, radiation and convection losses, fouling, and separator efficiency are major contributors to heat loss in an evaporator and therefore should be minimised.

The quality of the product has a lot to do with the design of the process as well as the way in which it is operated. It is the over-heating of product which causes the most damage. Generally, damage to the product will be avoided if a well-designed process is maintained in a steady operating condition without large fluctuations in product flowrate or heat input.

The conclusion, therefore, is that any control strategy implemented on an evaporator should simultaneously seek to optimise the energy consumption and maintain the product quality while achieving regulation of the product concentration and flow.

The input variables which can be manipulated in order to regulate the product concentration are the flow of steam to the first effect and the flow of cooling water through the condenser after the last effect. If the liquor flow through the evaporator is constant these two variables can regulate the out-flowing product concentration. One disadvantage though is the long delay between a change in output concentration in response to a change in steam input.

With the solids concentration of the feed and output at fixed levels and the mass of liquid in the evaporator constant then the product flowrate is only a function of the feed flow into the system.

Therefore to achieve control of the product concentration and flowrate the main manipulated variables are:

- steam flow
- feed flow
- cooling water flow

There are however many interactions which occur within the system which make control complicated. Some of the interactions which take place are:

- Changes in the feed flow or feed temperature create changes in the rate of concentration and will affect the steam requirement. Since the temperature difference between the vapour and liquid in an effect is usually only 10°C or less, a change in temperature of only 0.1°C would cause a change in evaporation rate of 1%.
- The pressure in an effect, which is related to its temperature, also strongly influences the flow into and out of the effect.
- Changes in the condenser cooling-water flow cause changes in the vacuum and hence the operating temperatures throughout the evaporator.
- The feed concentration of the product can be considered as a disturbance to the system. If it changes then the steam requirement will need to be altered.
- The steam pressure and quality are also disturbances to the system. These can cause significant fluctuations in evaporation rate.

2.3.2 Evaporator control development

In practice, as with many industrial processes, simple PID (Proportional-Integral-Derivative) control loops are implemented to control evaporators. These may be simple feedback loops, loops which also include feedforward elements or cascade control loops. Stapper [7] discusses an approach to controlling an evaporator system using cascade PID control with feedforward loops to reduce the impact of disturbances. In particular he suggests controlling the steam flowrate by feeding forward measurements of the product feed flow and concentration.

Today PID-type control systems are becoming increasingly inadequate for controlling large industrial processes. This is especially the case where the plant capacities are very large and the need for tight control is becoming an important consideration in order to remain competitive in the marketplace. More recently research has focussed on multivariable control approaches.

One method [8] tests a predictive control strategy for a multi-effect evaporator in a pulp mill. This strategy uses a linear impulse-response model within the controller to predict future plant outputs. The control inputs are determined through constrained optimisation of a cost function relating to desired future target outputs. This compared favourably with a PI control system.

Another method [9] makes use of linear process models internal to the controller for a semi-industrial scale milk evaporator. The type of control used is known as Internal

Model Control (see §4.5.2) where the feedback and feedforward compensators used are inverses of the linear process models. Comparisons are made between two controllers, one using a analytical process model and the other a model identified from process input-output data. Both controllers performed well with control using the identified model proving superior.

Much of the modern control research applied to industrial processes focuses on the use of process models within the control scheme. These models are mostly linear models, but work is being undertaken in applying nonlinear models to control [10].

2.4 The Production Technology falling-film evaporator

2.4.1 Plant description

The falling-film evaporator pilot-plant in the Department of Production Technology (Figure 2-10) is a three-effect evaporator with two preheater stages and a water-jet condenser. The plant has been constructed to replicate the type of process found within the dairy industry but at a significantly reduced scale. The whole plant stands five metres in height and takes up about 4 m^2 in floor space thus indicating its small scale compared with its industrial counterparts. Direct heating of the second preheater and first effect is achieved using an external steam supply. The vapour produced in the third effect is used in the first preheater stage. No vapour recompression is used in the process. The role of the water-jet condenser (also known as a *jet-ejector* or *venturi condenser*) is three-fold; (i) to produce the vacuum for the process, (ii) to remove condensed vapours and incondensable gases and (iii) to condense any remaining exhaust vapours from the system. The design of the evaporator allows it be used either in a straight-through configuration or with recirculation around each effect. The recirculation configuration (indicated by the dashed lines in Figure 2-10) makes use of pneumatic valves to control the flow of liquid passed onto the next effect. Centrifugal pumps are used to lift the liquid to the top of the calandrias and are necessary even when operating in a feedforward configuration.

2.4.2 Operating conditions

The design operating flowrate for the evaporator is 180-200 l/hr with a evaporation rate of 30 l/hr. In practice around 40 to 60 l/hr is achieved over the whole process. The product enters from the feed tank at ambient temperature and is heated to 35°C by the first preheater and then to 70°C in the second preheater. The designed steady-state operating temperatures for the effects are 70°C for the first effect, 60°C for the second

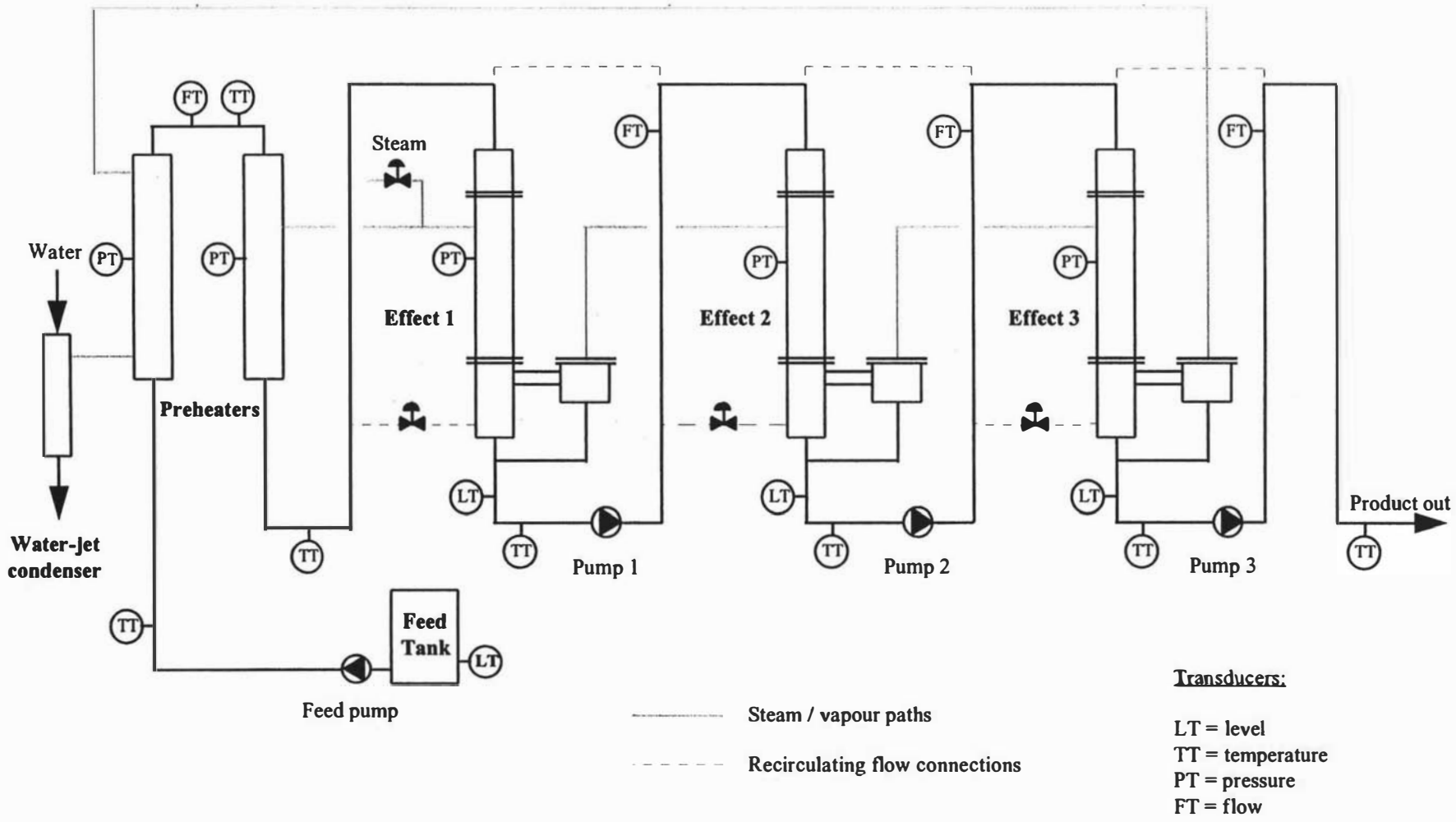


Figure 2-10: Schematic of the Production Technology falling-film evaporator

and 45°C for the third. To achieve this a steam pressure of around 50 kPa absolute (-50 kPa gauge) is required. This corresponds to a saturated steam temperature of 80°C. Table 2-1 summarises the design temperature and pressure operating conditions in the evaporator at steady-state.

Table 2-1: Designed operating temperatures and pressures.

Stage	Product temperature (°C)	Pressure of vapour in shell (kPa abs.)
Effect 1	70	50
Effect 2	60	32
Effect 3	45	20
Preheater 1	35	5
Preheater 2	70	50

The highest boiling temperature for the product is 70°C (32 kPa) and the coolest is 45°C (10 kPa). Cold water at approximately 600 kPa and flowrate of around 1080 l/hr is used as the driving fluid in the water-jet condenser. The condenser is able to produce and maintain a pressure of around 5 kPa in the shell of the first preheater.

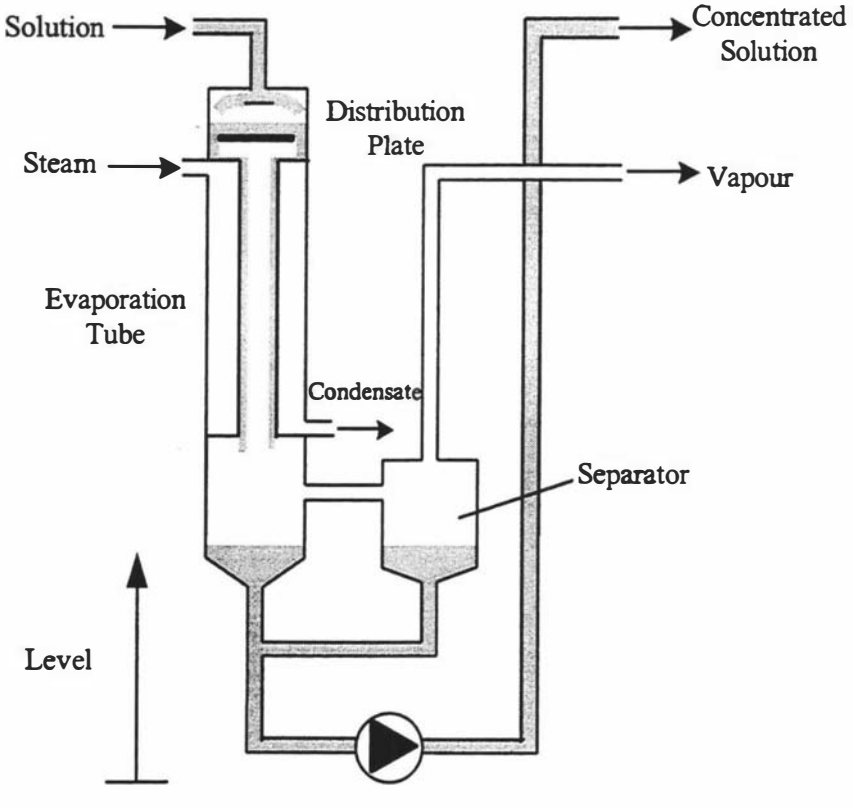


Figure 2-11: An effect of the pilot-plant falling-film evaporator.

2.4.3 Evaporator design

A simple schematic of a single effect of the pilot-plant is shown in Figure 2-11. Each effect has only one evaporation tube which is 2.9 metres in length. The tube diameters are 38 mm for the first two effects and 25 mm for the third. The liquid enters the effect through a simple nozzle and is distributed around the tube by way of a static head over a perforated distribution plate. Once the evaporation has taken place the vapour and liquid are separated in a centrifugal-type separator chamber. The concentrated liquid product forms a level at the base of the effect. This level protects the centrifugal pump below from running dry which may be caused by flow disturbances.

The preheaters are both straight-tube heat exchangers with three tube passes in each. A diagram of the jet-condenser is illustrated in Figure 2-12. High pressure water is forced through a 3 mm diameter venturi and then enters a wider chamber.

In the process the liquid pressure energy is converted into kinetic energy with a resultant decrease in pressure. Upon entering the wider chamber the water-jet entrains the air, condensate and vapour from the converging streams. The mixture then passes through a convergent-divergent outlet in order to decrease the velocity and return the fluid to atmospheric pressure. The condensing of vapour within the venturi condenser will also aid in the production of a vacuum. The performance of the condenser is dependent on the pressure and flow of the driving fluid and the temperature relative to the entrained fluid.

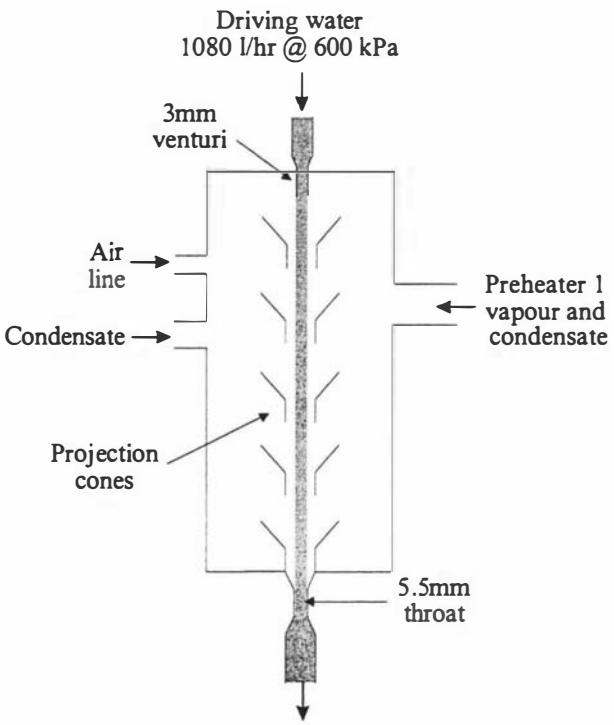


Figure 2-12: Venturi condenser

2.4.4 Instrumentation and controls

The pilot-plant has 20 variables that are measured electronically. These include seven temperatures, five pressures, four flows and four liquid levels. Their positions on the plant are indicated in Figure 2-10 and are as follows:

Temperatures - The temperature of the liquid is measured at base of each effect, at the outlet of each preheater and in the evaporator feed and outlet streams.

Pressures - The pressures are measured in the shell side of each effect and preheater.

Flows - The flowrates for the liquid streams are measured between each effect and at the inlet and outlet of the evaporator. They are positioned such a way that, if recirculation is used, they then measure the recirculated flows around each effect.

Levels - The height of the liquid in the feed tank and at the base of each effect is measured.

No transducers are present for on-line measurement of concentration or density.

The actuators include four centrifugal pumps and a pneumatic steam valve. If recirculation is used four additional pneumatic control valves are available. The pumps are controlled via variable speed drives. All the transducers and actuators are wired into a Fisher & Paykel Programmable State Controller (PSC) Series 2. This is akin to a Programmable Logic Controller (PLC) but allows the programmer to use a state machine programming language instead of ladder logic. The PSC performs all the low-level control functions; signal input and output, error handling and analogue-to-digital and digital-to-analogue conversion.

The PSC also communicates with a desktop computer via a serial link for the supervisory control function. The software package FixDMACS (ver.5.0) is used to carry out the Supervisory Control and Data Acquisition (SCADA) tasks. This allows the plant information to be displayed to an operator. It also performs the tasks of alarm monitoring, data storage and historical trending.

Proportional-Integral (PI) control has been used to regulate the concentrate levels in each effect. This ensures that the pumps are protected and that there is a constant mass of liquid in the plant. The effect pumps are used to control the levels in the straight-through configuration, if recirculation is used the control valves are manipulated instead. The controller gains have been determined through a computer simulation of the level systems. PID control is also implemented on the steam valve in order to regulate the steam pressure into the plant. The measurement of the steam pressure in the first effect shell is taken as the controlled variable. The control loops have proven to be vital in helping to settle the plant during start-up. These control loops are realised within the PSC rather than in the SCADA software in order to optimise the response time.

The PSC is the robust element in the system, it alone runs the plant. If the computer crashes for some reason the evaporator will continue to operate and control will be

maintained by the PSC although the data storage tasks of the SCADA package will cease. Figure 2-13 illustrates the information flow for the evaporator control system.

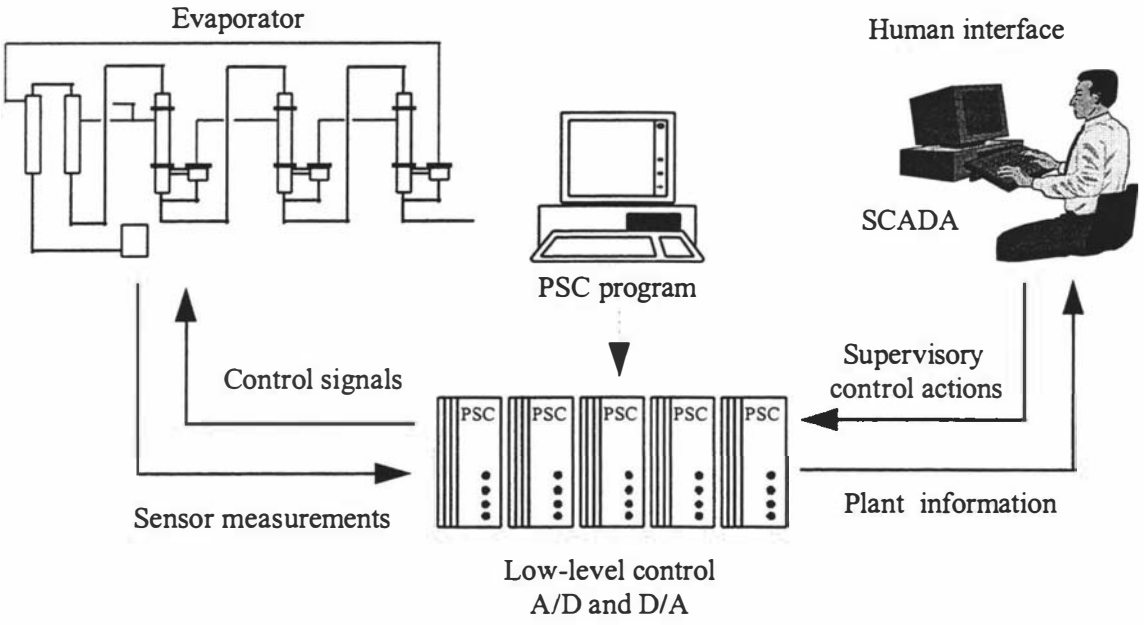


Figure 2-13: Evaporator control system

2.5 References

[1] **Brennan, J.G., Butters, J.R., Cowell, N.D. & Lilley, A.E.V.** *Food Engineering Operations*, 3rd Edition, Elsevier Applied Science, London, 1990, p. 367.

[2] **Kroschwitz, J.I & Howe-Grant, M.** (Eds.), *Kirk-Othmer Encyclopedia of Chemical Technology*, 4th Edition, vol. 9, John Wiley & Sons, New York, USA, 1994, p. 960.

[3] **Hahn, G.**, Evaporator Design. In: *Concentration and Drying of Foods*, MacCarthy, D. (Ed.), Elsevier Applied Science, London, 1986, pp. 113-131.

[4] **Hall, C.W., Farrall, A.W. & Rippen, A.L.** (Eds.), *Encyclopedia of Food Engineering*, 2nd Edition, AVI Publishing Co., Westport, Connecticut, USA. 1986, p. 348.

[5] **Perry, R.H., Green, D.W. & Maloney, J.O.** (Eds.), *Perry's Chemical Engineers' Handbook*, 6th Edition, McGraw-Hill Book Co., New York, USA, 1984, p 11-34.

[6] **Quaak, P., van Wijck M.P.C.M. & van Haren, J.J.**, Comparison of process identification and physical modelling for falling-film evaporators. *Food Control*, 1994, vol. 5, pp. 73-82.

[7] **Stapper, H.L.**, Control of an evaporator and spray drier using feedback and ratio feedforward controllers. *New Zealand Journal of Dairy Science and Technology*, 1979, vol. 14, pp. 241-257.

[8] **Ricker, N.L., Sim, T. & Cheng, C.-M.**, Predictive control of a multi-effect evaporation system. *Proceedings of the American Control Conference*, 1986, Seattle, USA, pp. 355-359.

[9] **van Wijck, M.P.C.M., Quaak, P. & van Haren, J.J.**, Multivariable control of a four-effect falling-film evaporator. *Food Control*, 1994, vol. 5, pp. 83-89.

[10] **Bequette, B.W.**, Nonlinear control of chemical processes: a review. *Industrial Engineering Chemistry Research*, 1991, vol. 30, pp. 1391-1413.

3

Evaporator Analytical Model and Simulation

OVERVIEW

This chapter presents a detailed explanation of the analytical model that has been formulated for the pilot-plant evaporator. After the theoretical discussion the model implementation within MATLAB and SIMULINK is briefly described. The chapter then covers the important task of validating the model against the actual plant. Through the validation process some of the model parameters are adjusted in order to obtain reasonable model estimates when compared with the actual evaporator data. The chapter concludes with a discussion of the performance of the analytical simulation model in characterising the pilot-plant.

3.1 Model development

3.1.1 Introduction

Before embarking on a task of model development one must first specify the purpose for which the model will be used. The purpose of the model influences the method of analysing the system and the complexity of the model.

The analytical evaporator model in this project is primarily to be used as a simulation tool for the development and testing of process control schemes. For this purpose the model is required to mimic the behaviour of the actual system in order to generate accurate estimates of the measurable outputs in response to the various input and disturbance signals. In this respect the model is not required to be overly complex and a dynamic lumped-parameter analysis is satisfactory. A systems approach is taken and therefore a distributed analysis of heat transfer, flow regimes and localised variations of

stream properties are not considered. The model is based on energy and mass balances through each part of the system. The analysis is simplified and hence a number of physical assumptions are required. These are outlined throughout the discussion of the model development. Once the system's basic characteristics have been described by the model, the model is tuned using a variety of coefficients and constants in order to quantitatively validate the model. The final model may not be the most accurate possible but, for the purpose of control system design, too much complexity can be a hindrance rather than adding to the model's performance.

3.1.2 Sub-system modelling approach

The basis for the approach used for the evaporator model development was taken from the work of Quaak and Gerritsen [1]. They developed a dynamic model for a multi-effect evaporator and used a systems approach in which the process was divided into sub-systems for the purpose of analysis.

Initial development of the model for the Production Technology evaporator pilot-plant was carried out by Illingworth [2]. He developed a dynamic model for the effects of the evaporator in the recirculating configuration (refer to §2.4.1) and qualitatively verified it with some simple simulations. (The author aided in many of the plant trials to determine plant constants and operating characteristics for this model). A full three-effect model for the recirculating system was not linked together however nor was any formal model validation carried out.

This recirculating model has since been developed further by the author so that the effect models are connected together. The recirculation model was then altered to model the evaporator in the feedforward configuration. New equations relating to the preheaters and venturi condenser have also been included. The development of the feedforward evaporator model is the subject of this chapter.

Illingworth's recirculating evaporator model followed the approach of Quaak and Gerritsen and included descriptions for the distribution plate, evaporation tube, product transport and energy systems for each effect. A similar analysis is used for the feedforward model with the addition of the temperature in the concentrate level as a state variable. This enables a direct comparison with the actual plant temperature measurements taken at the base of each effect.

The analysis of each effect of the evaporator therefore consists of the following sub-systems (Figure -3-1):

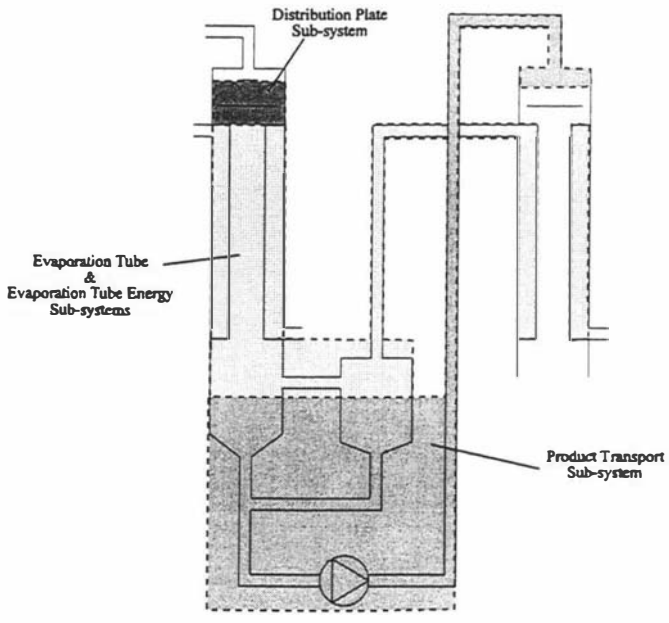


Figure -3-1: Sub-system boundaries in an effect.

- *Distribution plate sub-system*
  This models the flow of liquid through the distribution plate and into the evaporation tube.
- *Evaporation tube sub-system*
  This models the mass transfer occurring in the evaporation tube which results in a flow of concentrated product and a flow of vapour.
- *Product transport sub-system*
  This models the flow of product from one effect to the following effect downstream. Involves the reservoir of liquid concentrate, the pipework connecting the effects and the distribution nozzle in the downstream effect. The temperature changes that occur through the product transport system are also modelled.
- *Evaporation tube energy sub-system*
  This models the heat flows involved in the evaporation of the product. It includes the heat into the evaporation tube and the condensation of the vapour in the downstream effect.

The above sub-systems relating to each effect are combined with equations relating to the following sub-systems:

- *Feed sub-system*
  This models the flow of product from the feed tank through the preheater tubes and into the first effect.

- *Preheater sub-system*
  This models the heat transfer in the two preheaters.
- *Venturi condenser sub-system*
  This models the suction pressure produced by the venturi condenser and the rise in cooling water temperature due to the condensate and vapour exhaust streams.
- *External steam input sub-system*
  This models the temperature of the external steam input which is regulated via a PID controller.

A block diagram representation of the evaporator model is depicted in Figure 3-2. This shows that how the energy sub-systems underlie the whole model and have a strong influence over the other sub-systems.

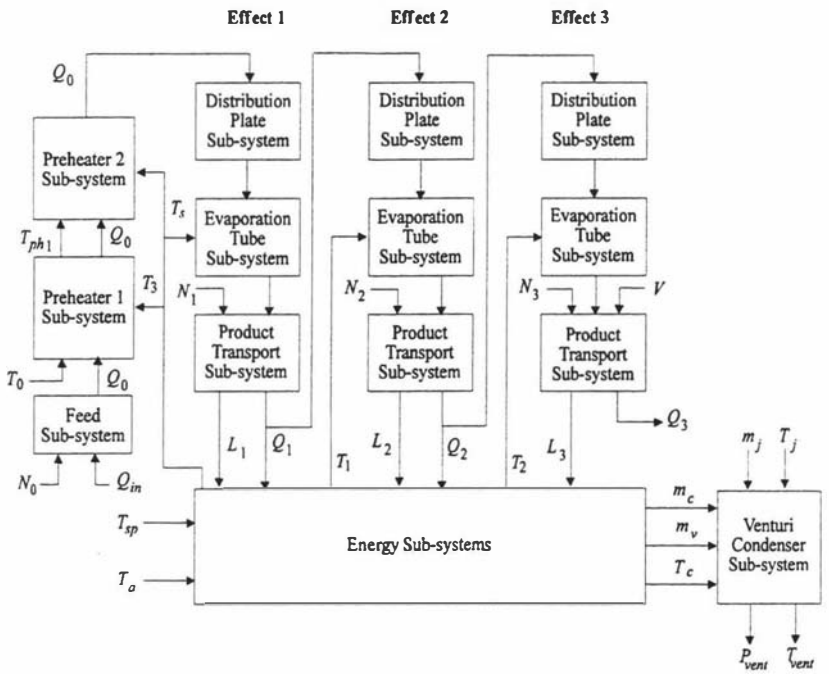


Figure 3-2: Sub-system representation of the evaporator model

For the feedforward model the equations relating to the distribution plate and evaporation tube system for the pilot-plant evaporator are equivalent to those developed by Illingworth. The equations for the product transport and feed systems have been altered from Illingworth's model to account for the feedforward flow configuration. Also a differential equation has been added to calculate the temperature of the liquid concentrate at the base of each effect. This enables a direct comparison with the actual plant temperature measurements taken at these points on the plant. The equations relating to the preheaters, the venturi condenser and the external steam input have been developed by the author. All the sub-systems have been linked together to fully define the feedforward evaporator system. Some of what Illingworth determined in terms of

constants can be transferred directly to the feedforward model, however most constants have been re-calculated to account for the different flow configuration used.

The pilot-plant evaporator has yet to be operated with concentrated solutions nor can the product density be measured on-line. Therefore, as a result, concentration equations have not been implemented in the final model. Throughout the discussion of the sub-systems however, equations relating to the concentration of the product stream are given for completeness. The fluid densities used in the equations can be assumed to be functions of the fluid temperature but, if concentrations are considered in the model, this also influences the density and the relationship between temperature, concentration and density must be determined.

Each sub-system in the evaporator will now be briefly discussed and their equations derived. The description of the variables are given during the discussion of the model and a glossary of the model variables and constants can be found in Appendix 1.

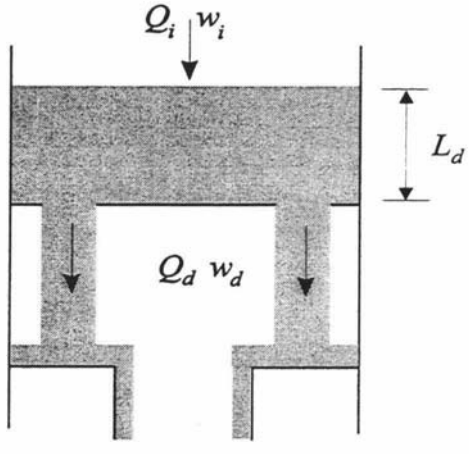


Figure 3-3: Distribution plate sub-system.

3.1.3 Distribution plate sub-system

The distribution plate is assumed to behave as a perfectly mixed tank with a variable level. A simplified diagram of the system is given in Figure 3-3.

The flow through the distribution plate, Q_d, can be determined using the following equation.

$$Q_d(t) = A_h\sqrt{\frac{2gL_d(t)}{K_h}} \tag{3.1}$$

where L_d is the height of liquid above the plate, A_h is the area of the holes in the plate, g is the gravitational acceleration constant and K_h is a friction factor dependent on the dimensions of the holes.

The height of liquid above the plate is a state variable and is given by the following differential equation.

$$\frac{dL_d(t)}{dt} = \frac{Q_i(t) - Q_d(t)}{A_d} \tag{3.2}$$

where Q_i is the flow into the distribution plate system of the effect and A_d is the area of the liquid surface.

The concentration (mass fraction of dry matter) of the flow through the distribution plate, w_d, is given by a mass balance.

$$\frac{dw_d}{dt} = \frac{\rho_i Q_i(t) w_i(t) - \rho_d Q_d(t) w_d(t)}{\rho_d(t) A_d L_d(t)}, \tag{3.3}$$

where ρ_i and ρ_d are the densities of the input and output flow respectively and w_i is the input concentration.

3.1.4 Evaporation tube sub-system

Figure 3-4 shows a generalised diagram of the evaporation tube sub-system. To model this system two simplifying assumptions are made.

i) The fall velocity of the liquid film is assumed to be constant and uniform.

ii) The heat flow across the tube is assumed to be uniform.

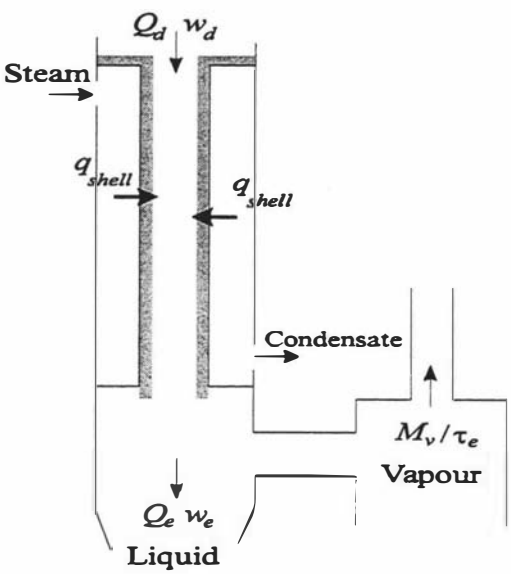


Figure 3-4: Evaporation tube sub-system.

The first assumption can be considered reasonable since the velocity of the falling-film is predominantly due to gravity and an increase in flowrate increases the film thickness. The implication of this assumption is that there is a uniform residence time, τ_e, through the tube. This was confirmed through tests which found times for the flow through the tube were approximately constant for a variety of flowrates (see section 3.1.12)

The second assumption implies a linear decrease in product concentration down the tube length. In reality the heat transfer across the tube wall varies in a nonlinear fashion down the tube. Generally the heat transfer is better at the top of the tube since the flow is more turbulent and the condensate film on the vapour side of the wall is thin. Moving down the tube the resistance to heat transfer increases as the condensate film gets thicker and the falling-film becomes less turbulent. Another influence on the rate of heat transfer is the case when the incoming liquid may be cooler than the effect temperature and energy must be used up to heat the liquid before evaporation can take place.

Despite the nonlinear relationships the assumption of uniform heat flow allows a simple lumped analysis of the heat transfer in the tube. An allowance is made by using mean

heat transfer coefficients not local coefficients in the heat flow equations. The coefficients have been derived assuming a lumped analysis. Additionally, with the small temperature differences occurring in the pilot-plant the assumption of uniform heat transfer should not cause significant problems.

Since the residence time is assumed constant the mass of liquid out of the tube at time t is equal to the mass flow into the tube at $t - \tau_e$ minus the amount of product vaporised during the residence time.

$$\rho_e(t)Q_e(t) = \rho_d(t-\tau_e)Q_d(t-\tau_e) - \frac{1}{\tau_e}\int_{t-\tau_e}^{t} \frac{q_{shell}(k)}{\lambda_e(k)}dk \tag{3.4}$$

where Q_e is the volumetric product flow from the evaporation tube, q_{shell} is the heat supplied to the product and λ_e is the product's latent heat of vaporisation.

The integral term in equation (3.4) calculates the mass of liquid evaporated over the residence time. This integral can be replaced by M_v to give

$$Q_e(t) = \frac{\rho_d(t-\tau_e)}{\rho_e(t)}Q_d(t-\tau_e) - \frac{1}{\rho_e(t)\tau_e}M_v(t). \tag{3.5}$$

M_v is the total mass of vapour produced in the evaporation tube during the previous τ_e seconds and is estimated by

$$\frac{dM_v(t)}{dt} = \frac{q_{shell}(t)}{\lambda_e(t)} - \frac{q_{shell}(t-\tau_e)}{\lambda_e(t-\tau_e)}. \tag{3.6}$$

The product concentration exiting the evaporation tube is calculated via a mass balance involving only pure delay;

$$w_e(t) = \frac{\rho_d(t-\tau_e)Q_d(t-\tau_e)w_d(t-\tau_e)}{\rho_e(t)Q_e(t)}. \tag{3.7}$$

3.1.5 Product transport sub-system

Product level and flowrate

The product transport system is concerned with the flow of product from one effect to the next. The boundaries of the system are the top of the concentrate level in the upstream effect and the distribution nozzle in the following effect. Quaak and Gerritsen made the assumption that the concentrate level remains constant. However, for this model a differential equation with the height of the concentrate level, L, as the state variable was used;

$$\frac{dL(t)}{dt} = \frac{Q_e(t) - Q_p(t)}{A_l} \tag{3.8}$$

where Q_p is the flowrate out of the effect and A_l is the area of the level surface.

A_l is a function L since the geometry of the pipework changes over the operating range of the level.

To calculate Q_p an energy balance between points ① and ③ in Figure 3-5 is required.

Using Bernoulli's equation one arrives at:

$$P_e(t) + \rho_e(t)gL(t) + \rho_e(t)\frac{v_e(t)^2}{2} + \Delta P_P(t)$$

$$= P_f(t) + \rho_p(t)gh_N + \rho_p(t)\frac{v_p(t)^2}{2} + \Delta P_N(t) + \Delta P_{fric}(t) \tag{3.9}$$

where P_e and P_f are the tube pressure of the upstream and downstream effects,

ΔP_P is the pressure increase across the pump,

ΔP_N is the pressure drop across the distribution nozzle of the next effect,

ΔP_{fric} is the pressure drop due to the friction in the pipe line,

h_N is the height of the distribution nozzle,

ρ_p is the discharge density, and

v_e and v_p are the flow velocities at the input and discharge points.

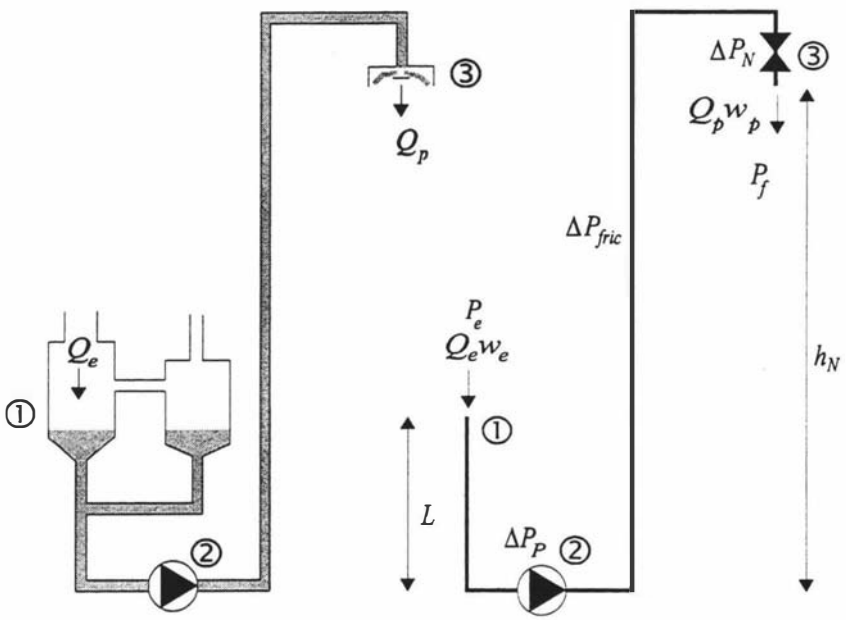


Figure 3-5: Product transport sub-system.

The pressures P_e and P_f are taken to be the saturation pressures at temperatures T_e and T_{tube} inside the evaporation tubes.

The velocity terms are given by the volumetric flowrate divided by the cross-sectional area of the pipe at the particular point. The input velocity, v_e, is the velocity at the surface of the liquid level at the base of the effect and therefore is given by

$$v_e = \frac{Q_p}{A_l}. \tag{3.10}$$

The discharge velocity into the next effect is also calculated using Q_p with the cross-sectional area of the pipe before the distribution nozzle. The high velocity through the nozzle is not used as this is accounted for in the nozzle pressure loss term. Therefore

$$v_p = \frac{Q_p}{A_p} \tag{3.11}$$

where A_p is the cross-sectional area of the inter-effect pipe.

Centrifugal pumps are employed on the pilot-plant. In theory the pressure increase across a pump is a function of the pump speed squared, N^2, and the discharge flowrate, Q_p.

According to the supplied pump operating curves the pressure-speed relationship is approximately linear for the small operating flows used on the evaporator. The following equation was therefore proposed [2]:

$$\Delta P_P(t) = a_P N(t)^2 + b_P Q_p(t)$$

It was found from experiments (§3.1.12) that the pump pressure was independent of the flowrate for the range of operating flows tested. Therefore the b_P term can be assumed to be zero for the pilot-plant and the pump pressure is estimated as a function of the pump speed squared.

$$\Delta P_P(t) = a_P N(t)^2 \tag{3.12}$$

The a_P term was determined through experiment (§3.1.12).

A general formula [3] using the Fanning friction factor, f, pipe length to diameter ratio, L_e/d, and velocity head, is commonly used to calculate line friction pressure losses;

$$\Delta P_{fric} = 4f\frac{l_e}{d}\rho\frac{v^2}{2}.$$

Converting the velocity to a volumetric flowrate gives

$$\Delta P_{fric}(t) = \frac{32 f l_e \bar{\rho}(t) Q_p(t)^2}{\pi^2 d^5} \tag{3.13}$$

where $\bar{\rho}$ is the average density of the liquid (taken as the mean of ρ_e and ρ_i),

f is the Fanning friction factor, obtained from a Moody chart [3],

l_e is the equivalent tube length, which accounts for any pipe bends or connections, and

d is the internal pipe diameter.

The pressure drop across the distribution nozzle can be described using the general flow relationship $Q = k\sqrt{\Delta P}$. A characteristic value, C_N, is determined experimentally (§3.1.12) to give

$$\Delta P_N(t) = \left(\frac{Q_p(t)}{C_N} \right)^2 . \tag{3.14}$$

Each nozzle consists of three flow orifices through which the liquid passes. The output flow from the third effect passes through a hand valve and hence a characteristic constant for this valve (C_V) would be used in place of the nozzle constant.

Substituting equations (3.10) to (3.14) into equation (3.9) and solving for Q_p we get

$$Q_p(t) = \sqrt{\frac{P_e(t) + \rho_e(t) g L(t) + a_P N(t)^2 - P_f(t) - \rho_p(t) g h_N}{\frac{32 f l_e \bar{\rho}(t)}{\pi^2 d^5} + \frac{1}{C_N{}^2} + \frac{1}{2}\left(\frac{\rho_e(t)}{A_l{}^2} - \frac{\rho_p(t)}{A_p{}^2} \right)}} . \tag{3.15}$$

In practice the velocity head terms can be omitted as they are negligible in comparison with the other terms for this system due to the small operating flowrates in the evaporator. Therefore the $\frac{1}{2}\left(\frac{\rho_e}{A_l{}^2} - \frac{\rho_p}{A_p{}^2} \right)$ term could be removed from equation (3.15).

Concentration

To calculate the concentration through the sub-system one can assume plug flow conditions throughout which gives

$$w_p(t) = w_e(t - \tau_p) \tag{3.16}$$

where τ_p is the time delay between the input to the level and the output of the distribution nozzle (point ① to ③ of Figure 3-5).

This time delay is that which satisfies the integral;

$$V_p = \int_{t-t_p}^{t} Q_p(t)\,dt \tag{3.17}$$

where V_p is the volume of pipe the flow has travelled through in τ_p seconds.

An iterative optimisation technique is needed to calculate the best estimate of τ_p from equation (3.17) which can be a slow approach. Alternatively, if it is assumed that Q_p was constant over the previous τ_p seconds ($Q_p(t) = Q_p(t-\tau_p)$), then the time delay can be approximated with an instantaneous estimate using

$$\tau_p = \frac{V_p}{Q_p(t)} \quad \text{for } Q_p(t) > 0, \tag{3.18}$$

where V_p is the total pipe volume.

An alternative approach to assuming plug flow conditions from point ① to point ③ is to assume perfect mixing in the reservoir at the base of the effect and plug flow only in the pipe from the pump to the next effect (point ② to ③). Then

$$w_p(t) = w_l(t - \tau_p) \tag{3.19}$$

where τ_p is the time delay between the pump and the next effect and w_l, the concentration of the liquid out of the reservoir, is determined using

$$\frac{dw_l(t)}{dt} = \frac{\rho_e(t)Q_e(t)w_e(t) - \rho_l(t)Q_p(t)w_l(t)}{\rho_l(t)A_l L(t)} \tag{3.20}$$

where ρ_l is the density of the flowrate out of the level.

Temperature

An energy equation is defined for the liquid level at the base of the effect. A transducer is located in this region on the pilot-plant to measure the temperature of the product. It is necessary to have as part of the model a calculation of the product temperature at this point for comparison with the measured temperature.

The assumptions made are that the liquid in the reservoir is perfectly mixed and that plug flow conditions exist from the pump onwards.

The product level (Figure 3-6) has two heat flows, the heat from the product flowing into the

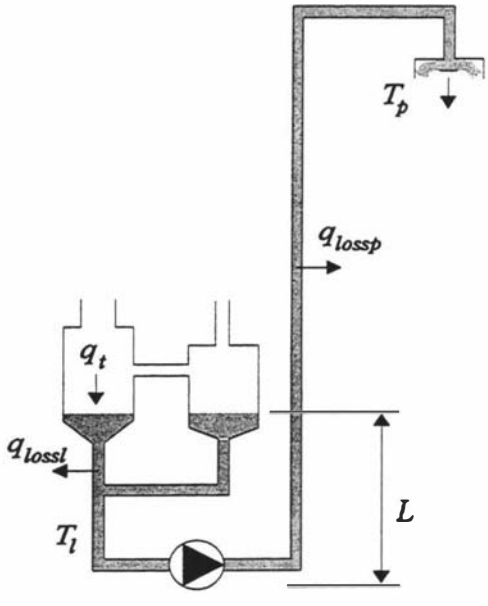


Figure 3-6: Product transport heat flows.

reservoir, q_t, and the heat loss to the surroundings, q_{lossl}.

The change in temperature of the product in the level is given by

$$\frac{dT_l(t)}{dt} = \frac{q_t(t) - q_{lossl}(t)}{M_l(t)c_p} \tag{3.21}$$

where M_l is the mass of liquid in the level and is given by

$$M_l(t) = \rho_l(t)V_l, \tag{3.22}$$

and c_p is the specific heat capacity of the product in the level and V_l is the volume of liquid which is a function of the liquid height.

The two heat flows are calculated by

$$q_t(t) = Q_e(t)\rho_e(t)c_p(T_e(t) - T_l(t)) \tag{3.23}$$

and

$$q_{lossl}(t) = U_{lossl}A_{sl}(T_l(t) - T_a(t) \tag{3.24}$$

where A_{sl} is the surface area of the level sub-system and U_{lossl} is the heat transfer coefficient for the convective heat loss.

The temperature of the product flowing into the next effect can then be estimated by applying the transport delay, τ_p, to T_l in a similar manner to calculating the output concentration (equation (3.19)). If an estimate of heat loss is included then the output temperature of the effect is

$$T_p(t) = T_l(t - \tau_p) - \frac{U_{lossp}A_{sp}(T_l(t) - T_a(t))}{\overline{\rho}(t)Q_p(t)c_p} \tag{3.25}$$

where A_{sp} is the surface area of the pipework, U_{lossp} is the heat transfer coefficient for the convective heat loss and the density is estimated using $\overline{\rho}$.

3.1.6 Evaporation tube energy sub-system

Figure 3-7 illustrates the energy sub-system relating to the evaporation tube. The system is characterised by four heat flows, the combination of which determine the change in temperature of the product through the evaporation tube. It is assumed that the temperature is the same over the length of the tube at any one time.

The energy balance defining this system is;

$$\frac{dT_e(t)}{dt} = \frac{q_{shell}(t) + q_i(t) - q_{tube}(t) - q_{loss}(t)}{M(t)c_p} \tag{3.26}$$

where T_e is the product temperature in the tube, M is the mass of liquid in the evaporation tube, and c_p is the specific heat capacity of the product.

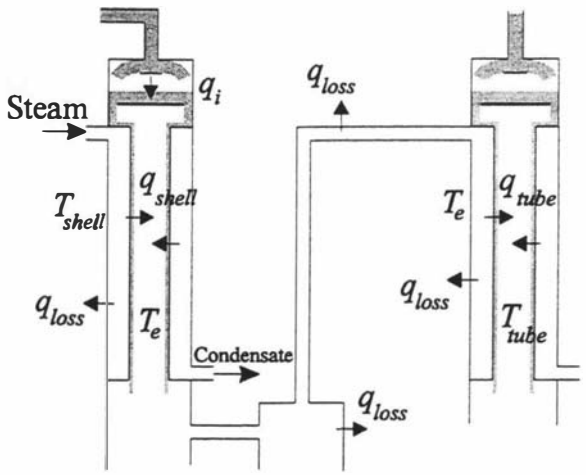


Figure 3-7: Evaporation tube energy sub-system.

The specific heat capacity, c_p, can either be assumed constant or a function of temperature. The mass of liquid in the tube, M, at any point in time can be estimated by the instantaneous mean flowrate through the tube:

$$M(t) = \left(\frac{Q_d(t)\rho_d(t) + Q_e(t)\rho_e(t)}{2} \right) \tau_e \tag{3.27}$$

The four heat flows of equation (3.26) and shown in Figure 3-7 are defined as follows;

- q_{shell} is the heat supplied from the vapour in the shell current effect,
- q_{tube} is the energy given up to heat the product in the tube of the downstream effect,
- q_i is the heat supplied from the input product stream,
- q_{loss} is the heat loss to the surrounds via natural convection.

The driving force for each of these flows are the temperature differences between the two streams involved. The equation defining these heat flows are given below.

The energy flow from the heating vapour is given by

$$q_{shell}(t) = U_e A_e (T_{shell}(t) - T_e(t)) \tag{3.28}$$

where T_{shell} is the temperature of the vapour in the shell.

A_e is the heat transfer area of the evaporation tube, and

U_e is the overall heat transfer coefficient for the heat transfer from the bulk of one stream to the bulk of the other.

T_{shell} is the temperature of the vapour generated in the upstream effect or in the case of first effect is the external steam supply temperature.

Similarly for the next effect downstream

$$q_{tube}(t) = U_e A_e (T_e(t) - T_{tube}(t)) \tag{3.29}$$

where U_e and A_e are the heat transfer area and heat transfer coefficient of the downstream effect and T_{tube} is the temperature of the product in the tube.

For the third effect T_{tube} is the temperature of the product in the first preheater which is downstream from the effect.

The heat from the input stream is represented by a flash term

$$q_i(t) = \rho_d(t) Q_d(t) c_p (T_i(t) - T_e(t)), \tag{3.30}$$

where T_i is the temperature of the incoming product stream.

If the input flow is at a higher temperature than that of the effect tube ($T_i > T_e$), then the incoming liquid flashes off as it enters the region of lower pressure through the distribution plate.

If the input flow is cooler than the effect temperature ($T_i < T_e$) then some heat energy from the heating vapour will be used up to raise the temperature of the incoming product and $q_i < 0$.

The heat loss to the surroundings is given by

$$q_{loss}(t) = U_{loss} A_s (T_e(t) - T_a(t)) \tag{3.31}$$

where T_a is the ambient temperature,

A_s is the surface area of the sub-system, and

U_{loss} is the overall heat transfer coefficient for the convective heat transfer.

The vapour created in the third effect is used to preheat the product in the first preheater. Unlike the shells of the other stages the first preheater shell is attached directly to the venturi condenser without a constriction to ensure the vapour is fully condensed. Because of this the vapour from the third effect is only partially condensed in the preheater shell and an additional negative heat flow term is required in equation (3.26). This term, q_v is for the heat dissipated into the venturi condenser from the uncondensed preheater 1 vapour.

$$q_v = \lambda_{ph1} m_{vph1} \tag{3.32}$$

where λ_{ph1} is the enthalpy of condensation for the preheater vapour and m_{vph1} is the vapour mass flowrate into the condenser (see equation (3.60)).

3.1.7 Feed flow sub-system

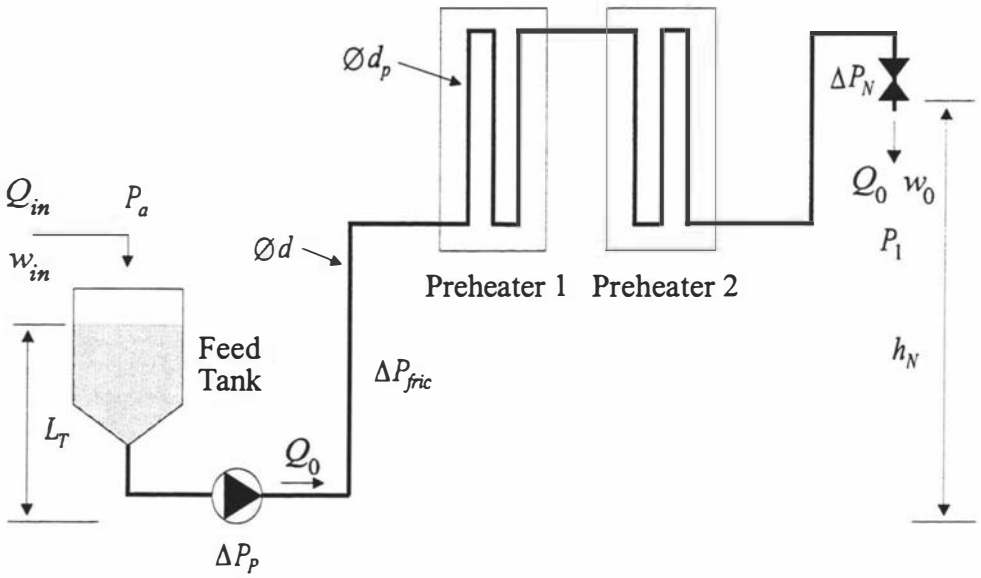


Figure 3-8: Evaporator feed sub-system

The feed flow system consists of the feed tank, feed pump and the pipe work from the tank to the first effect via the two preheaters (Figure 3-8). The model of this sub-system attempts to estimate the feed flowrate into the evaporator.

The state variable of this sub-system is the liquid level in the feed tank, L_T

$$\frac{dL_T(t)}{dt} = \frac{Q_{in} - Q_0(t)}{A_T} \tag{3.33}$$

where A_T is the tank cross-sectional area, Q_0 is the feed flowrate into the evaporator and Q_{in} is the input flow into the feed tank which is assumed to be constant.

The feed flowrate can be determined with the same analysis as for the product transport sub-system using Bernoulli's equation;

$$P_a + \rho_T g L_T(t) + \Delta P_P(t) + \frac{\rho_T v_T^2}{2} = P_1(t) + \rho_{ph2} g h_N + \frac{\rho_{ph2} v_0^2}{2} + \Delta P_N(t) + \Delta P_{fric}(t)$$

...(3.34)

where P_a is the atmospheric pressure,

P_1 is the first effect tube pressure (taken as the saturation pressure at T_1),

ρ_T and ρ_{ph2} are the liquor density at the feed tank and preheater 2,

v_T and v_0 are the fluid velocities at the tank and entering effect 1,

ΔP_{fric} is the line pressure loss, and

ΔP_P and ΔP_N are the pressure losses due to the feed pump and the first effect distribution nozzle respectively. These are calculated using equation (3.12) and (3.14).

The velocity head terms in equation (3.34) can be omitted since they are a factor of 10^3 smaller than the other pressure terms under the standard operating conditions of the evaporator.

The tubes inside the preheaters are of smaller diameter than the pipes which make up the rest of the feed line. The two tube dimensions need to be included in the calculation of the line losses; therefore based on equation (3.13):

$$\Delta P_{fric}(t) = \frac{32 f \bar{\rho}_f(t) Q_0(t)^2}{\pi^2}\left(\frac{l_p}{d_p{}^5} + \frac{l_f}{d^5}\right) \tag{3.35}$$

where Q_0 is the feed flowrate,

$\bar{\rho}_f$ is the mean product density between the feed tank and the first effect,

l_p and l_f are the equivalent pipe length for the preheater tubes and the standard pipework, respectively and

d_p and d are the internal diameters of the preheater tubes and the standard pipework.

By rearranging equation (3.34) (with the velocity heads removed) to make the feed flow, Q_0, the subject (remembering that ΔP_N and ΔP_{fric} are functions of the feed flow) we get

$$Q_0(t) = \sqrt{\frac{P_a(t) + \rho_T(t) g L_T(t) + a_{P0} N_0(t)^2 - P_1(t) - \rho_{ph2}(t) g h_N}{\frac{32 f \bar{\rho}_f(t)}{\pi^2}\left(\frac{l_p}{d_p{}^5} + \frac{l_f}{d^5}\right) + \frac{1}{C_N{}^2}}} \tag{3.36}$$

which is of similar form to the equation for the effect output flowrate (equation (3.15)).

If plug flow conditions are assumed then the concentration entering the first effect is simply equal to a delayed version of the feed tank concentration.

$$w_0(t) = w_T(t - \tau_f) \tag{3.37}$$

where w_T is the density in the feed tank and τ_f is the transport delay between the feed tank and the first effect.

The delay can be approximated by using a similar approach to that used in the product transport system with

$$\tau_f = \frac{V_f}{Q_0(t)} \quad \text{for } Q_0(t) > 0, \tag{3.38}$$

where V_f is the volume of the feed system pipework.

The feed tank temperature can be considered as a state variable and if perfect mixing and negligible heat losses are assumed then;

$$\frac{dT_0(t)}{dt} = \frac{\rho_{in}(t)Q_{in}(t)(T_{in}(t) - T_0(t))}{\rho_T(t)A_T L_T(t)}. \tag{3.39}$$

However the feed temperature, T_0, has been considered as an input variable for the simulation model and equation (3.39) has not been included.

If the product concentration is required then the concentration in the feed tank can also be considered a state variable;

$$\frac{dw_T(t)}{dt} = \frac{\rho_{in}(t)Q_{in}w_{in}(t) - \rho_T(t)Q_0(t)w_T(t)}{\rho_T(t)A_T L_T(t)}. \tag{3.40}$$

3.1.8 Preheater sub-systems

The preheater sub-system equations relate to the heat transfer occurring within the preheaters.

Considering a small length, δx, of the preheater tube of total length, l, (Figure 3-9) then the total heat flow into the section, δx, is

$$q = U_{ph}A_x(T_v - T(x)) - q_{lossx} \tag{3.41}$$

where $A_x = \pi D_p \delta x$, is the heat transfer area for section δx,

T_v is the temperature of the heating vapour,

$T(x)$ is the temperature of the product at point x in the pipe, and

q_{lossx} is the heat loss to the surroundings from section δx.

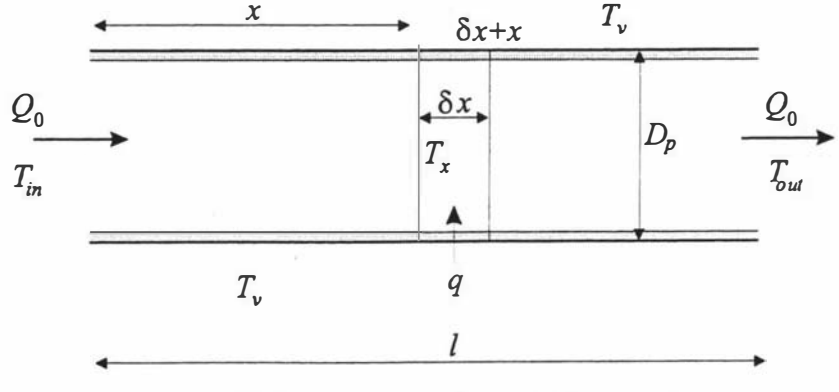


Figure 3-9: A section of a preheater tube

Additionally, we can describe the accumulation of heat across section δx with

$$q_x + q = q_{x+\delta x}$$

to give
$$\rho_{ph} Q_0 c_{ph} T(x) + q = \rho_{ph} Q_0 c_{ph} T(x+\delta x) \tag{3.42}$$

Combining equations (3.41) and (3.42):

$$\frac{T(x+\delta x) - T(x)}{\delta x} = \frac{U_{ph} \pi D_p}{\rho_{ph} c_{ph} Q_0} (T_v - T(x)) - \frac{q_{lossx}}{\rho_{ph} c_{ph} Q_0} \tag{3.43}$$

Then as $\delta x \to 0$ we get

$$\frac{dT(x)}{dx} = \frac{U_{ph} \pi D_p}{\rho_{ph} c_{ph} Q_0} (T_v - T(x)) - \frac{q_{lossx}}{\rho_{ph} c_{ph} Q_0}. \tag{3.44}$$

By Laplace transforming, rearranging and then performing the inverse Laplace transform one arrives at

$$T_v - T(x) = \left(T_v - T(0) + \frac{q_{lossph}}{\rho_{ph} c_{ph} Q_0} \right) e^{\frac{-U_{ph} \pi D_p x}{\rho_{ph} c_{ph} Q_0}}. \tag{3.45}$$

We want an expression for the output temperature, that is, the temperature when x equals the length of preheater tube, l. If the temperature at $x = 0$ is T_{in}, and noting that the heat transfer area is given by, $A_{ph} = \pi D_p l$,

and the heat loss; $q_{lossph} = U_{lossph} A_{sph} (T_v - T_a)$,

then using equation (3.45) the output temperature, T_{out}, is

$$T_{out} = T_v - \left(T_v - T_{in} + \frac{U_{lossph} A_{sph} (T_v - T_a)}{\rho_{ph} c_{ph} Q_0} \right) e^{\frac{-U_{ph} A_{ph}}{\rho_{ph} c_{ph} Q_0}}. \tag{3.46}$$

Equation (3.46) is a steady-state equation for the preheater output temperature.

Inserting the preheater system variables (Figure 3-10) into equation (3.46), the steady-state temperature for the first preheater, $\overline{T}_{ph1}$, is

$$\overline{T}_{ph1} = T_{e3} - \left(T_{e3} - T_0 + \frac{U_{lossph1} A_{sph1} (T_3 - T_a)}{\rho_{ph1} c_{ph} Q_0} \right) e^{\frac{-U_{ph1} A_{ph}}{\rho_{ph1} c_{ph} Q_0}} \tag{3.47}$$

and for preheater 2;

$$\overline{T}_{ph2} = T_s - \left(T_s - T_{ph1} + \frac{U_{lossph2} A_{sph2} (T_s - T_a)}{\rho_{ph2} c_{ph} Q_0} \right) e^{\frac{-U_{ph2} A_{ph}}{\rho_{ph2} c_{ph} Q_0}} \tag{3.48}$$

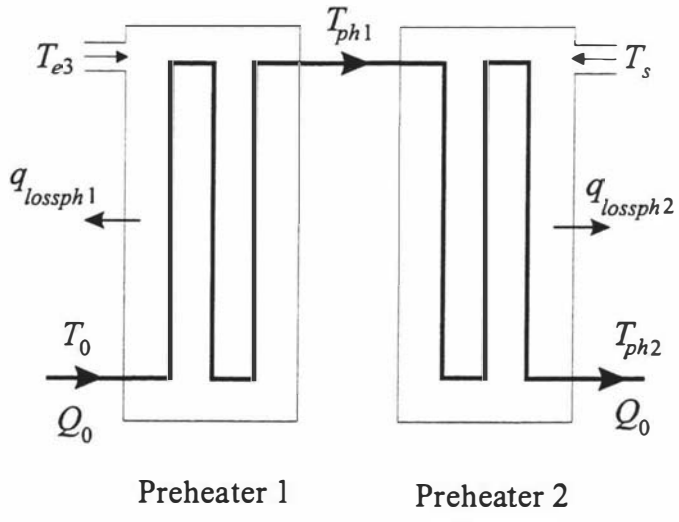


Figure 3-10: Preheater system

To obtain a dynamic relationship a first order response is assumed. Using a discrete first order representation the dynamic function can be written as

$$T_{ph}(z) = \left(\frac{1-\eta_{ph}}{1-\eta_{ph}z^{-1}}\right)\overline{T}_{ph}(z) \tag{3.49}$$

where $\overline{T}_{ph}$ is the steady-state temperature from equation and $\eta_{ph} < 1$.

The term η_{ph} is a function of the system time constant, $\tau_{c(ph)}$;

$$\eta_{ph} = e^{-\left(t_s/\tau_{c(ph)}\right)} \tag{3.50}$$

where t_s is the sampling time.

A sampling time of 1 second is used for the discrete system and the calculation of the preheater time constants is discussed in Section 3.1.12.

3.1.9 Venturi condenser sub-system

The venturi condenser (Figure 3-11) involves a complex process. No simple theory can accurately describe the manner or means by which the jet of water produces entrainment and mixing.

In the evaporator condenser four processes are occurring [4]:

i) acceleration of the particles of the entrained fluid by the impact of particles from the primary jet;

ii) entrainment of the secondary fluid by viscous friction at the periphery of the jet;

iii) overexpansion of the jet to a pressure below that of the secondary fluid, with a consequent flow of the secondary fluid towards the axis of the jet;

iv) change of state occurring as a result of the vapour being condensed by the water jet.

The analysis of the evaporator condenser involved developing equations relating to the entrainment rates of the vapour and condensate, the accumulation of heat in the condenser and the suction pressure produced.

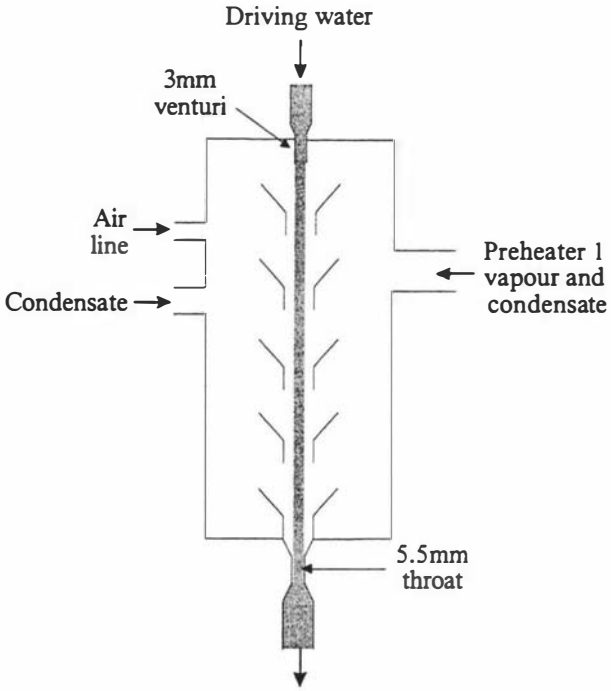


Figure 3-11: Venturi condenser

Entrainment rates

The venturi condenser draws condensate, vapour and any incondensables from the evaporator system. The shells of each effect and the second preheater have small orifices through which the condensate flows and is passed into the condensate line connected to the jet condenser. These orifices ensure that the vapour condenses fully within these shells. At the base of the effect shell a level of condensate forms and the liquid flows out through the orifice (Figure 3-12). The first preheater shell is directly attached to the condenser and so its vapour is not necessarily fully condensed and a mixture of condensate and vapour is drawn into the condenser.

The amount of vapour and hence condensate produced for each effect can be estimated assuming that the rate of vaporisation of product is equal to the rate of condensation of the heating steam. Therefore the mass of condensate produced per residence time in an effect is equal to M_v.

If we assume that the condensate level always covers the orifice and that the effect of the level head is negligible then the mass flow of

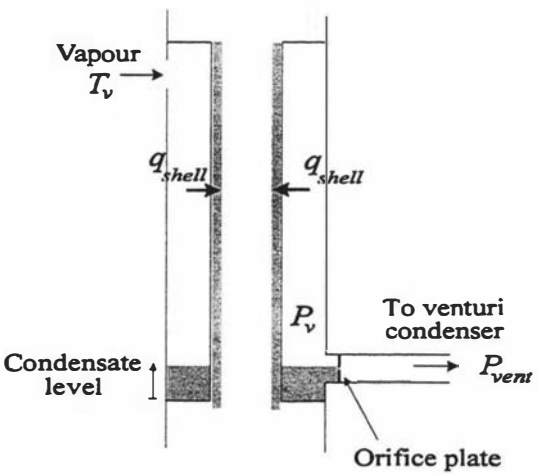


Figure 3-12: Collection of condensate in an effect.

condensate from the shell, m_c, is a function of the pressure drop through the orifice.

$$m_c(t) = C_{vc}\sqrt{\Delta P_c(t)} \tag{3.51}$$

where ΔP_c is the pressure difference between the pressure in shell and the venturi suction pressure, ie. $\Delta P_c = P_v - P_{vent}$, and

C_{vc} is a flow-pressure constant for the condensate outlet orifice.

An estimate of C_{vc} can be arrived at by considering the velocity pressure through the orifice.

$$\Delta P_c(t) = \frac{\rho_c(t)v_c(t)^2}{2} \tag{3.52}$$

where ρ_c is the condensate density and v_c is the condensate velocity through the orifice.

Taking A_c as the cross-sectional area of the orifice then

$$v_c(t) = \frac{m_c(t)}{\rho_c(t)A_c},$$

and one obtains

$$C_{vc} = K_c\sqrt{2\rho_c}A_c, \tag{3.53}$$

where K_c is included as a constant of proportionality.

The K_c term makes allowances for the stream constriction and any frictional effects. It can be considered as the ratio between the actual and the theoretical flowrate through the orifice. This form of equation is equivalent to that recommended in Perry *et al.* [5].

If we assume the changes in the condensate level are negligible then the mass flow of condensate leaving effect i is limited by the mass flow of vapour entering it from the previous effect (i–1) one residence time before.

$$m_{ci}(t) \le \frac{M_{vi-1}(t-\tau_e)}{\tau_e} \tag{3.54}$$

The condensate leaves the second preheater shell in a similar manner to the condensate from the evaporation effects. In the first preheater, however, complete condensation of the vapour is not guaranteed because there is no constriction in the stream leaving the shell. An estimate of the mass flows of vapour and condensate is therefore obtained by assuming that flow of condensate is dependent on the heat transfer in the preheater similar to the calculation of the vapour produced in the evaporation effects (equation (3.6)).

$$\frac{dM_{cph1}(t)}{dt} = \frac{q_{ph1}(t)}{\lambda_{ph1}(t)} - \frac{q_{ph1}(t-\tau_{ph})}{\lambda_{ph1}(t-\tau_{ph})} \tag{3.55}$$

where τ_{ph} is the transport delay time through the preheater tube,

λ_{ph1} is the latent heat of condensation for the preheater vapour,

q_{ph1} is the overall heat transfer in the preheater given by

$$q_{ph1}(t) = U_{ph1} A_{ph} \Delta T_{ph1}(t) \tag{3.56}$$

and ΔT_{ph1} is the log-mean temperature difference for the first preheater, given by

$$\Delta T_{ph1}(t) = \frac{T_{ph1}(t) - T_0(t)}{\ln\left(\frac{T_{e3}(t) - T_0(t)}{T_{e3}(t) - T_{ph1}(t)}\right)}. \tag{3.57}$$

The residence time of the product through the preheater can be estimated by dividing the preheater tube volume by the feed flowrate;

$$\tau_{ph} = \frac{V_{ph}}{Q_0(t)} \quad \text{for } Q_0(t) > 0. \tag{3.58}$$

The mass flow of condensate from the first preheater can then be estimated as follows;

$$m_{cph1}(t) = \frac{M_{cph1}(t)}{\tau_{ph}}. \tag{3.59}$$

The vapour mass flowrate is then estimated by subtracting the condensate flow from the third effect vapour which entered the preheater τ_{ph} seconds before.

$$m_{vph1}(t) = \frac{M_{v3}(t-\tau_{ph})}{\tau_e} - m_{cph1}(t) \tag{3.60}$$

The temperature of the vapour and condensate mixture flowing from the first preheater is assumed to be equal to the temperature of the third effect, T_{e3}.

Condenser temperature

It is expected that the suction pressure of the condenser is limited by the temperature of the condenser outlet water. The minimum suction pressure achievable would be equal to the saturation pressure at the outlet temperature, T_{vent}. As this saturation pressure nears the suction pressure the efficiency of the system to produce a vacuum would decrease.

The temperature of the entrained condensate into the venturi can be calculated based on the relative temperatures and flowrates of each stream.

$$T_c(t) = \frac{[m_{cph2}(t) + m_{c1}(t)]T_s(t) + m_{c2}(t)T_{e1}(t) + m_{c3}(t)T_{e2}(t) + m_{cph1}(t)T_{e3}(t)}{m_{cph2}(t) + m_{c1}(t) + m_{c2}(t) + m_{c3}(t) + m_{cph1}(t)} \tag{3.61}$$

where T_{ei} is the temperature of effect i and is assumed to be the temperature of the condensate from the shell of effect i+1; T_s is the steam supply temperature and the temperature of the condensate from the second preheater and first effect shells.

The temperature of the driving fluid exiting the venturi, T_{vent}, can be calculated considering the heat flows in and out of the venturi system;

$$T_{vent} = T_j + \frac{q_c + q_v - q_{lossv}}{m_j c_j} \tag{3.62}$$

where T_j is the temperature of the in-flowing venturi driving water,

q_c is the heat supplied by the condensate,

q_v is the heat supplied by the vapour (equation (3.32)),

q_{lossv} is the heat lost to the surroundings, and

c_j is the specific heat capacity of the water jet.

Expanding equation (3.62) we obtain

$$T_{vent} = T_j + \frac{m_c c_j (T_c - T_j) + m_{vph1}\lambda_{ph1} - U_{lossv}A_{sv}(T_{vent} - T_a)}{m_j c_j} \tag{3.63}$$

where λ_{ph1} is the latent heat of condensation of the preheater 1 vapour,

U_{lossv} is the overall heat transfer coefficient for the heat loss, and

A_{sv} is the surface area of the venturi.

Suction pressure

The usual method of analysing a jet pump in order to calculate the suction pressure produced is in terms of total pressure. Smith [6] developed a simple theory which was based on the pressure rise through the ejector and was verified experimentally. His results are valid for a water or steam-jet ejector entraining a fluid. The simplified approach of Smith has been used by the author to model the evaporator venturi condenser.

Referring to Figure 3-13 the mixing region of the evaporator ejector is assumed to be the section of the diffuser between points ① and ②. The projection cones do not appear to perform a major function since the jet of water remains well within the cones. The

lowest cone however, rests on the base of the venturi and effectively forms the initial section of the diffuser.

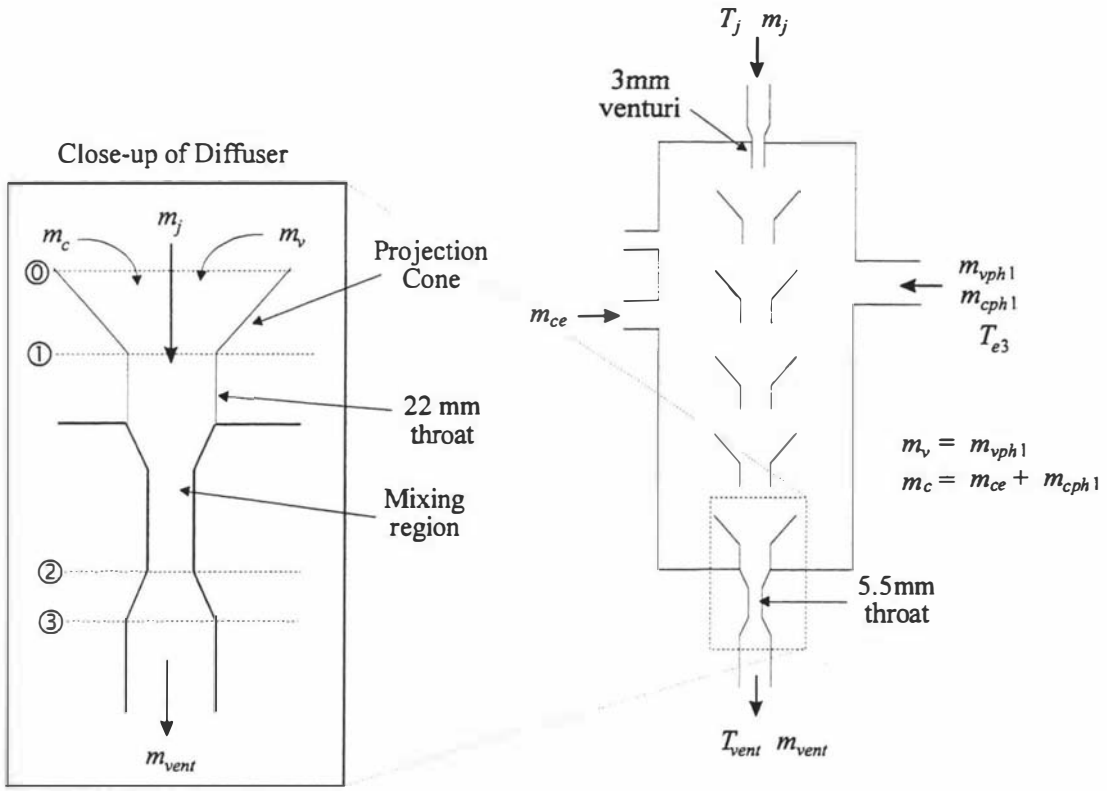


Figure 3-13: Venturi condenser system

The analysis of the venturi condenser follows the approach of Smith by calculating the overall pressure rise in the condenser. The pressure rise equation developed below however differs considerably from that of Smith due to the different condenser geometry and the fact that both vapour and condensate is drawn from the evaporator. Throughout the analysis it is assumed that the amount of air entrained by the condenser is negligible once the evaporator is operating normally.

The pressure drop of the flow entering the diffuser section is obtained from the energy equation below (equation (3.64)). Since the entrained stream is a mixture of vapour and condensate both flows are considered. If changes in density and the vertical height difference are neglected and it is assumed that the vapour and condensate are initially at rest, we can write

$$p_0 - p_1 = \frac{1}{2}(\rho_v v_v^2 + \rho_c v_c^2) \tag{3.64}$$

where p_i is the pressure at point i (in Figure 3-13),

ρ is the density, and

v the velocity of the entrained fluid, with

the v and c subscripts denoting the vapour and condensate properties respectively.

The pressure rise in the diffuser is obtained by applying the momentum equation to the cross sections at points ① and ②.

$$p_1 A_v + m_v v_v + m_c v_c + m_j v_j = p_2 A_t + (m_v + m_c + m_j) v_2 \tag{3.65}$$

where ρ_j and v_j are the density and velocity of the jet, respectively.

m_v, m_c and m_j are the mass flowrates,

A_v is the cross-sectional area of the cone at point ①, and

A_t is area of the diffuser throat at point ②.

The pressure rise in the divergent section is obtained from the energy equation, again neglecting changes in density and height and neglecting the final kinetic pressure.

$$p_3 - p_2 = \frac{1}{2}\rho_2 {v_2}^2 \tag{3.66}$$

where ρ_2 and v_2 are the density and velocity of the mixed fluid.

Smith [6] assumed that the loss of pressure due to wall friction can be expressed in the form;

$$\frac{K_f}{2}\rho_2 {v_2}^2 \tag{3.67}$$

where $0 < K_f < 1$.

Point ③ in Figure 3-13 is at a height, h_c, above the ground, therefore the pressure drop between point ③ and the discharge outlet to atmosphere is given by

$$P_a - p_3 = \rho_3 g h_c \tag{3.68}$$

where P_a is the ambient atmospheric pressure.

Combining equations (3.64) to (3.68) we obtain an expression for the condenser pressure at point ⓪;

$$p_0 = \frac{A_t}{A_v}(P_a - \rho_3 g h_c - \rho_2 v_2) + \frac{(\rho_v {v_v}^2 + \rho_c {v_c}^2)}{2} - \frac{(m_j v_j + m_v v_v + m_c v_c - m_{vent} v_2)}{A_v} + \frac{K_f}{2}\rho_2 {v_2}^2 \tag{3.69}$$

where $m_{vent} = m_j + m_v + m_c$.

It can be assumed that the entrained fluids occupy an area of $A_v(1-\alpha)$ at the entrance to the diffuser, where α is the ratio of the cross-sectional areas of the jet stream and the projection cone.

$$\alpha = \frac{A_j}{A_v} \tag{3.70}$$

Similarly we can define β as being the ratio between the diffuser throat cross-sectional area and projection cone area.

$$\beta = \frac{A_t}{A_v} \tag{3.71}$$

We can also assume, that after mixing, $\rho_2 = \rho_3 = \rho_j$, since the driving fluid flowrate is much higher than the entrained fluid flowrate.

Given these definitions the velocities at the various points can be expressed as follows:

$$v_j = \frac{m_j}{\rho_j \alpha A_v} \tag{3.72}$$

$$v_2 = \frac{m_j + m_v + m_c}{\rho_j \beta A_v} \tag{3.73}$$

For the entrained mixture of condensate and vapour it is assumed that both streams have an equal velocity-induced pressure, that is

$$\frac{1}{2}\rho_c v_c^2 = \frac{1}{2}\rho_v v_v^2 .$$

Hence the ratio of their velocities is

$$\frac{v_v}{v_c} = \sqrt{\frac{\rho_c}{\rho_v}} .$$

Therefore, the following velocity equations can be determined.

$$v_c = \frac{m_c}{\rho_c (1-\alpha) A_v}(R+1) \tag{3.74}$$

$$v_v = \frac{m_v}{\rho_v (1-\alpha) A_v}\frac{(R+1)}{R} \tag{3.75}$$

with R defined as the ratio of the cross-sectional areas for the two flows and is given by

$$R = \frac{A_{vapour}}{A_{condensate}} = \frac{m_v}{m_c}\sqrt{\frac{\rho_c}{\rho_v}} . \tag{3.76}$$

Substituting these terms into equation (3.69) and defining $P_{vent} = p_0$, we get

$$P_{vent} = \beta(P_a - \rho_j g h_c) - \frac{m_j^2}{\rho_j \alpha A_v^2} + \frac{(K_f + \beta) m_{vent}^2}{2\rho_j \beta^2 A_v^2} + \frac{R+1}{(1-\alpha)A_v^2}\left(\frac{m_c^2(\alpha+R)}{\rho_c(1-\alpha)} - \frac{m_v^2}{\rho_v R}\right). \quad ...(3.77)$$

Equation (3.77) represents the suction pressure generated by the jet condenser. This is theoretically assumed to be the minimum achievable pressure inside the condenser vessel. However the thermal effects also need to be taken into account. As mentioned previously, it can be assumed that the minimum pressure is limited by the saturation pressure of the water at temperature T_{vent}. That is,

$$P_{vent} \geq P_{sat}\big|_{T_{vent}}. \quad (3.78)$$

The steps involved in arriving at equation (3.77) from the derived pressure balance expression (equation (3.69)) are detailed in Appendix 2.

Using P_{vent} the pressure increase through the condenser (between points ⓪ and ③) can be easily expressed as

$$\Delta P_r = P_a - \rho_j g h_c - P_{vent}. \quad (3.79)$$

Example calculation

To verify equation (3.77), a calculation of the suction pressure under normal operation of the condenser is shown in Appendix 2 and is discussed below.

We have assumed a driving fluid flowrate (m_j) of 0.3 kg/s (at 18°C), a condensate flow (m_c) of 16.7 × 10^{-3} kg/s (at 70°C) and a vapour flow (m_v) of 2.80 × 10^{-3} kg/s (at 45°C) into the condenser system.

The geometry of the condenser gives the following constants;

$\alpha = 0.0189$ $\quad A_v = 3.73 \times 10^{-4}\ \text{m}^2$

$\beta = 0.0637$ $\quad h_c = 1.8\ \text{m}$

Smith [6] states that generally the friction constant, K_f, has a value between 0.3 and 0.5. For this example we have taken $K_f = 0.3$.

Using equation (3.77) with the above information, the resulting suction pressure (P_{vent}) for the evaporator venturi condenser is calculated to be 4.1 kPa and the increase in pressure, ΔP_r, is 79.5 kPa. This has good agreement with what occurs in practice and can be considered a satisfactory approximation of the condenser operating point.

Pressure characteristics

A plot of the pressure rise in the condenser as a function of the entrainment ratio, m_{ent}/m_j, using equation (3.77) is shown in Figure 3-14. For this plot it is assumed that the mass flow of condensate is 6 times that of the vapour flow which is an estimation of what occurs in reality. This plot shows that as the entrainment ratio increases the pressure rise decreases. This operating curve is characteristic of general jet ejectors [4] and confirms that equation (3.77) is a plausible definition of the condenser suction pressure.

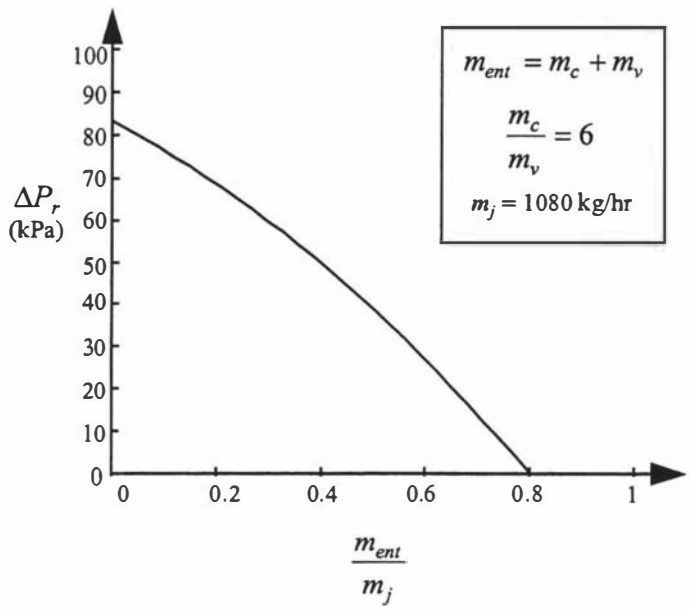


Figure 3-14: Pressure rise against entrainment ratio for the venturi condenser (with m_j constant).

Figure 3-15 is a close-up plot of the curve and shows that for the low entrainment rates experienced in the evaporator (m_{ent} < 100 kg/hr) there is little variation in the suction pressure and ΔP_r is between about 78 and 83 kPa. This agrees with practical experience in that there seems to be little variation in the condenser suction pressure for variations in the entrainment rate.

If the jet mass flowrate is varied the condenser pressure remains low, as can be seen in Figure 3-16. For a constant entrainment flowrate the condenser pressure does not go much above 6 kPa. The asterisk in Figure 3-16 is the operating point calculated above. The suction pressure within the shaded regions are likely to be quite different than shown in the plot. This is due to the limiting saturation pressure at the temperature of the condenser (equation (3.78)). For very small entrainment ratios the plot shows the condenser pressure dropping away to zero however in practice this would level out to a

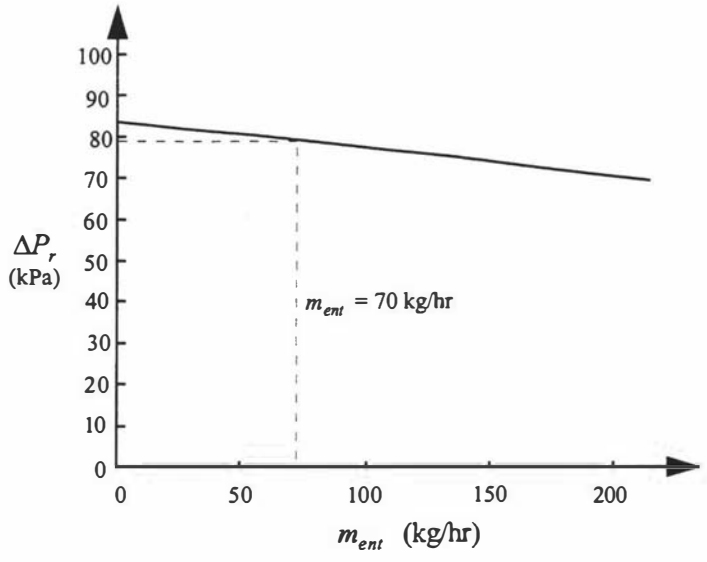


Figure 3-15: Pressure rise against entrainment rate for the venturi condenser (with m_j constant).

constant limited by the water-jet's inlet temperature (eg. T_{vent} = 18°C → min. P_{vent} = 2 kPa). At the higher entrainment ratios the temperature increase of the water-jet would be significantly higher thus raising the saturation pressure limit to above 6 kPa (ie. if T_{vent} > 36°C → min. P_{vent} > 6 kPa).

Again this plot seems to confirm what occurs in practice; that variations in the water-jet flowrate produce small changes in the condenser pressure.

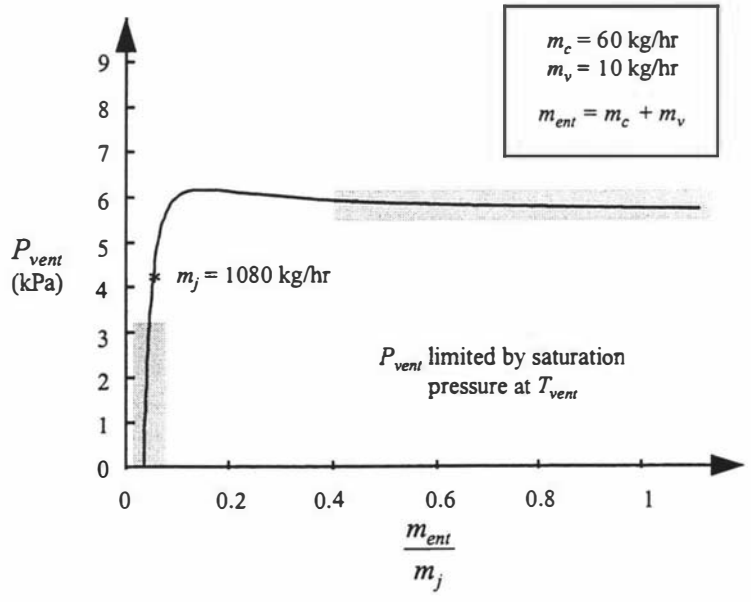


Figure 3-16: Condenser pressure against entrainment ratio (with m_{ent} constant).

3.1.10 External steam supply

So far we have considered the steam temperature as an input variable for the evaporator model. However it is the steam valve position which is the actual input since it determines the flow of steam into the system and hence its pressure and temperature. Additionally the steam pressure is influenced by the evaporation taking place in the first effect. That is, if the product feed flowrate increases then the first effect temperature decreases which causes a drop in the steam pressure. A decrease in the feed flow has an opposite effect. So in practice the steam temperature cannot be considered as a truly independent input variable. The model is required to calculate the steam temperature given the steam valve position and first effect evaporation conditions. The steam valve is difficult to model accurately as its characteristic is dependent on the pressure of the steam supply which can vary from day to day.

The pilot-plant is operated with a PID controller on the steam valve which adjusts the valve position to regulate the steam pressure in the shell of the first effect. It was decided to incorporate the PID controller as part of the evaporator model rather than relying solely on a steam valve characteristic equation. The setpoint to the controller therefore becomes the input to the model.

It must noted that on the actual plant only the steam pressure can be measured whereas the model uses the steam temperature. When comparing the model with the actual plant data, the steam temperature is estimated using the saturated temperature at the measured steam pressure.

The controlled variable for the PID controller is the steam temperature in the first effect shell, T_s, and the manipulated variable is the steam valve position, V_s. Using a transfer function description we can describe the controller as

$$\frac{V_s(s)}{E(s)} = k_p + \frac{k_i}{s} + k_d s \tag{3.80}$$

where E is the deviation from the setpoint, T_{sp} : $E(t) = T_{sp}(t) - T_s(t)$,

k_p, k_i and k_d are the proportional, integral and derivative gains for the PID controller.

It was decided to use a simple first order transfer function to approximate the relationship between the valve position and the steam temperature as

$$\frac{T_s(s)}{V_s(s)} = \frac{1}{\tau_c s + 1} \tag{3.81}$$

with the time constant, τ_c, determined experimentally (§3.1.12).

Combining equations (3.80) and (3.81) we get

$$\frac{T_s(s)}{E(s)} = \frac{k_d s^2 + k_p s + k_i}{s(\tau_c s + 1)}. \tag{3.82}$$

The influence of the evaporation in the first effect can be considered as a disturbance (ΔT_s) to this system so that the deviation from the setpoint becomes

$$E(s) = T_{sp}(s) - T_s(s) + \Delta T_s(s). \tag{3.83}$$

The disturbance on the steam temperature can be estimated by considering the change in the heat given up in the first effect per unit mass flow of liquid.

$$\Delta T_s(t) = \frac{1}{c_{pl}} \frac{d}{dt}\left(\frac{q_{el}(t)}{m_{el}(t)}\right) = \frac{U_{el} A_{el}}{c_{pl}} \frac{d}{dt}\left(\frac{T_s(t) - T_{el}(t)}{Q_{dl}(t)\rho_{dl}(t)}\right) \tag{3.84}$$

Which in the s-domain is

$$\Delta T_s(s) = \frac{U_{el} A_{el}}{c_{pl}} \left(\frac{T_s(s) - T_{el}(s)}{Q_{dl}(s)\rho_{dl}(s)}\right) s. \tag{3.85}$$

This can also be approximated with a discrete representation;

$$\Delta T_s(z) = \frac{U_{el} A_{el}}{c_{pl}} \left(\frac{T_s(z) - T_{el}(z)}{Q_{dl}(z)\rho_{dl}(z)}\right)(1 - z^{-1}). \tag{3.86}$$

3.1.11 Summary of model variables

The evaporator model has ten input variables (five of which are considered to be constant), 19 state variables (two of which are discrete) and 16 output variables. The following summarises the variables used in the evaporator model.

Inputs

The evaporator model has five main input variables:

T_{sp}	Steam temperature setpoint	(°C)
N_0	Feed pump speed	(rpm)
N_1	Effect 1 pump speed	(rpm)
N_2	Effect 2 pump speed	(rpm)
N_3	Effect 3 pump speed	(rpm)

There are five additional inputs which are considered to be constants for the evaporator model:

T_a	Ambient temperature	(°C)
T_0	Feed temperature	(°C)
P_a	Ambient pressure	(Pa)
V	Outlet valve position	(%)
Q_{in}	Flowrate into feed tank	(l/hr)

State variables

The evaporator model has 19 state variables.

For each effect the model has five state variables:

T_e	Evaporation tube product temperature	(°C)
T_l	Concentrate level temperature	(°C)
L	Concentrate level	(m)
L_d	Distribution plate level	(m)
M_v	Mass of vapour produced per residence time	(kg)

Additional state variables relating to the preheater and feed systems are:

T_{ph1}	Preheater 1 product temperature	(°C)
T_{ph2}	Preheater 2 product temperature	(°C)
L_T	Feed tank level	(m)
M_{vph1}	Mass of vapour produced in first preheater per residence time	(kg)

The two preheater temperatures have been implemented as discrete state variables.

Outputs

The model has 16 output variables which are used to compare with measurements from the actual plant.

For each effect there are 3 outputs:

T_l	Concentrate level temperature	(°C)
L	Concentrate level	(m)
Q_p	Product flowrate leaving effect	(l/hr)

Outputs for the feed and preheater systems are:

T_{ph1}	Preheater 1 product temperature	(°C)
T_{ph2}	Preheater 2 product temperature	(°C)
T_s	Steam temperature	(°C)
Q_0	Feed flowrate of product	(l/hr)

Additional outputs for the venturi condenser system are:

T_{vent}	Temperature of condenser outlet	(°C)
P_{vent}	Condenser suction pressure	(Pa)
m_{vent}	Mass flowrate of condenser outlet	(kg/s)

3.1.12 Model parameters

The model parameters that have been calculated are discussed below. Further detail of the calculations that were required and a summary of their values is given in Appendix 3.

Evaporator geometry

All the parameters relating to the physical geometry of the evaporator were calculated using measurements quoted on the design drawings or actual measurements taken from the plant. The surface area of the liquid levels above the distribution plate and at the base of the effects are functions of the height of the levels due to the varying evaporator geometry. Hence these areas were calculated as piecewise functions of the liquid height. The detail of the calculations for the geometric parameters is given in Appendix 3(*a*). Much of the calculation of the geometric properties was undertaken by Illingworth [2].

Heat transfer coefficients

The following discussion outlines the equations used to estimate the heat transfer coefficients. Steady-state conditions are assumed in estimating the coefficients. More detail on the actual calculations is provided in Appendix 3(*b*). Since the estimates of the heat transfer coefficients are based on theoretical equations the specific values obtained were altered during the model validation to support observations of the plant's actual thermal characteristics. These values however gave a starting point for the model validation task. The heat transfer coefficient estimates have been re-calculated since Illingworth's [2] work.

Heat transfer across the evaporation tube walls

The heat transfer through the tube walls in a falling-film evaporator involves a condensing vapour on the outer surface, conduction through the wall and evaporation of the liquid falling-film inside the tube.

These heat transfer processes are in series so the overall heat transfer coefficient, U, for this situation is given by

$$\frac{1}{U} = \frac{1}{h_v} + \frac{1}{h_w} + \frac{1}{h_l} \tag{3.87}$$

where h_v is the coefficient for the heat transfer from the condensing vapour to the wall,

h_w is the heat transfer coefficient for the conduction through the wall,

and h_l is the heat transfer coefficient from the wall to the liquid film.

The wall conduction coefficient is calculated with equation (3.88)

$$h_w = \frac{k_w}{x} \tag{3.88}$$

where k_w is conductivity of the stainless steel wall and x is the wall thickness.

The average heat transfer coefficient for vapour condensing on vertical tubes can be estimated from the following equation [7, 8, 9, 10]

$$h_v = 0.943\left(\frac{\rho_f^{\,2} k_f^{\,3} g\lambda}{\mu_f \Delta T_v L}\right)^{1/4} \tag{3.89}$$

where λ is the condensation enthalpy,

L is the heated length of tube,

ΔT_v is the difference between the bulk vapour temperature and the temperature of the wall on the vapour side, i.e. $\Delta T_v = T_v - T_{wv}$,

g is the acceleration due to gravity,

and ρ_f, k_f and μ_f are the density, thermal conductivity and dynamic viscosity, respectively, of the condensate film which are determined at mean film temperature, T_f.

T_f is determined as the mean temperature between the bulk vapour and the vapour side of the wall.

$$T_f = \frac{T_v + T_{wv}}{2} \tag{3.90}$$

The mean heat transfer coefficient for heating a liquid stream inside a tube requires calculating the coefficient from the Nusselt number, N_{Nu}

$$h_l = N_{Nu}\left(\frac{k^3\rho^2 g}{\mu^2}\right)^{1/3} \tag{3.91}$$

The Nusselt number for the evaporation of a turbulent liquid falling-film in a vertical tube, is a function of the Reynolds number, N_{Re}, and Prandtl number, N_{Pr}, of the stream and is given by [7, 8]

$$N_{Nu} = 0.01 N_{Re}^{1/3} N_{Pr}^{1/3} \tag{3.92}$$

where the Prandtl number is

$$N_{Pr} = \frac{\mu c}{k} \tag{3.93}$$

and the Reynolds number for film flow is

$$N_{Re} = \frac{4\Gamma}{\mu} \tag{3.94}$$

where the specific heat capacity, c, viscosity, μ, and the thermal conductivity, k, of the liquid are calculated at the film temperature.

Γ is the mass flow of the film per wetted perimeter and is given by

$$\Gamma = \frac{m}{\pi d} \tag{3.95}$$

where m is the mass flow of liquid in the falling-film and d is the internal diameter of the tube.

The falling-film would be within the turbulent flow region if $N_{Re} \geq 2100$. For the standard operating conditions of the evaporator the falling-films for each effect are in this range and therefore exhibit turbulent flow.

Heat transfer in the preheater tubes

The heat transfer through the tubes in the preheaters involves condensing vapour on the outer surface, conduction through the wall and heating of the liquid flowing inside the tube The calculations for the heat transfer coefficients for conduction through the wall and for the condensing vapours are the same as used for the evaporation tubes but the calculation for the liquid coefficient is different.

The heat transfer coefficient for heating liquid in the tube firstly involves determining which flow regime exists, calculating the coefficient from the Nusselt number and then inserting the Nusselt result into equation (3.91).

For turbulent flow of liquid through vertical tubes the Nusselt number is [7]

$$N_{Nu} = 0.023 N_{Re}^{0.8} N_{Pr}^{1/3} \left(\frac{\mu}{\mu_w}\right)^{0.14} \tag{3.96}$$

and for transition flow the Nusselt number is

$$N_{Nu} = 0.116(N_{Re}^{2/3} - 125) N_{Pr}^{1/3} \left[1 + \left(\frac{d}{l}\right)^{2/3}\right] \left(\frac{\mu}{\mu_w}\right)^{0.14} \tag{3.97}$$

where μ is the liquid viscosity at the bulk temperature

μ_w is the viscosity at the temperature on the liquid side of the wall, and

l is the length of each tube pass.

The Reynolds number, N_{Re} is calculated as

$$N_{Re} = \frac{\rho v d}{\mu} \tag{3.98}$$

where v is the velocity of the liquid stream and the density, ρ, and viscosity, μ, of the liquid are calculated at the bulk temperature.

For the preheaters the bulk liquid temperature is taken as the mean temperature between the preheater input and output points.

The Reynolds number again determines the flow regime of the liquid. If $N_{Re} > 10{,}000$ then the flow is turbulent, if $2{,}100 < N_{Re} < 10{,}000$ then it is a transitional flow.

Heat loss to the surroundings

Heat loss to the surroundings occurs primarily through condensing vapours on the inside wall, conduction through the wall and convective heat transfer from the outer wall of the vessel. Since the heat transfer coefficients for natural convection are several orders of magnitude lower than those for conduction and condensing vapour, the overall coefficient for the heat loss process, U_{loss}, is approximately equal to the convective coefficient, h_o. For this it will also be assumed that the outer side of the wall is equal to the internal bulk temperature.

For the conditions that exist in the evaporator McAdams [8] suggests the use of a dimensional equation for the heat loss coefficients which, when converted to SI units is

$$h_o = 1.3123(\Delta T_o)^{1/3} \tag{3.99}$$

where ΔT_o is the temperature difference between the bulk ambient air and the hot surface.

Residence times and delays

Quaak and Gerritsen [1] suggest the estimate of the residence time within the evaporation tube, τ_e, can be obtained through measurements. Experiments were carried out by Illingworth [2] which involved injecting dye at the top of the effect and timing its travel down the effect tube. This was carried out for various flowrates and for both sizes of evaporation tubes. All the times were found to be in the region of four seconds, indicating that the residence time was reasonably independent of the flowrate and tube diameter.

The transport delay between the effects, τ_p, is estimated using the tube volume and flowrate (equation (3.18)). This method of estimation was tested on the plant using dye. The dye was injected at the base of the effect when the level, L, was negligible and the delay for the flow to reach the top of the next effect was measured. Various flowrates used under normal operation of the evaporator were tested and the times compared with those calculated using equation (3.18). The test measurements gave reasonably good agreement with the calculated times indicating that equation (3.18) is appropriate to use in the model. The experimental results are given in Appendix 3(c).

The transport delay for the feed system, τ_f, can be tested in a similar manner with similar successful results (Appendix 3(c)) showing that the use of the pipe volume and flowrate is a satisfactory method of estimating the time delays.

Time constants

Time constants are required for the both preheaters and the steam supply equations. These are determined from simple step changes on the plant. The step changes are applied to the setpoint of the steam pressure controller and the responses of the steam temperature and the second preheater temperature are analysed. To calculate the time constant for the first preheater a step change in the third effect temperature is required. This is difficult to achieve in practice and so for the model it is assumed that both the preheaters have the same time constant. Details of the step tests and the calculations can be found in Appendix 3(d).

Pressure-flow characteristic constants

Pump characteristic constants

Equation (3.12) proposes a relationship between the pump speed and the pressure drop across the pump which requires the identification of the term a_P. Illingworth reports the experiments carried out to determine this constant. The experiments involved measuring the pump speed and discharge pressure for various flowrates and a_P was

determined by a linear regression of ΔP versus N^2. A valve was used to vary the flowrate and it was found that the pump pressure was independent of the flowrate. Considering the small flows which the evaporator operates under, this was not unexpected. The third effect pump has a larger impeller than the other three pumps in order to deliver the larger force required to discharge the fluid to atmospheric pressure. As expected it gave different results. The experimental measurements and the regression results from Illingworth are reproduced in Appendix 3(*e*).

Nozzle characteristic constants

The experimental set-up used to determine the distribution nozzle characteristic, C_N, (equation (3.14)) was similar to that used for the pumps with the valve replaced by a nozzle. The effect 1 and 2 distribution nozzles have holes of 1 mm diameter and the third effect has larger 1.5 mm diameter holes so give a different flow characteristic constant. The flowrate was varied through the nozzle and the pressure drop across measured. The flow was regressed with the square root of the pressure drop according to equation (3.14). The results from Illingworth are reproduced in Appendix 3(*e*).

Condensate flow constants

From equations (3.51) and (3.53) we get

$$m_c = C_{vc}\sqrt{\Delta P_c} = \left(K_c\sqrt{2\rho_c}A\right)\sqrt{\Delta P_c}\,. \tag{3.100}$$

An estimate of the flow constant, C_{vc}, can be obtained if we assume that the flow of condensate from an effect is approximately equal to the amount of vapour generated within the evaporator tubes.

$$\left(K_c\sqrt{2\rho_c}A_c\right)\sqrt{\Delta P_c} \approx \frac{q_{shell}}{\lambda} \tag{3.101}$$

The only unknown in the equation (3.101) is the constant, K_c. Therefore we can use this equation to estimate K_c and hence give a value for C_{vc}.

Rearranging equation (3.101) and substituting equation (3.28) for q_{shell} we get

$$K_c \cong \frac{U_e A_e (T_v - T_e)}{\sqrt{2\rho_c \Delta P_c} A_c \lambda}\,. \tag{3.102}$$

This equation has been used to estimate the friction factor and the subsequent flow constant for each condensate orifice. The results of these calculations are given in Appendix 3(*e*).

Thermophysical properties

The effect pressures were taken as the saturated water pressure at the liquid temperature. The pressures, along with the density and enthalpy of the liquid flows, were estimated within the model using polynomial approximations as functions of the liquid temperature and were based on the figures given in Cooper & Le Fevre [11]. This approach was taken to avoid the use of look-up tables. The polynomials used to estimate the three properties are given in Appendix 3(*f*).

The heat capacities (c_p, c_{ph} and c_j) are assumed constant and are estimated using the heat capacity of water at the design steady-state operating temperatures of the respective flows.

Quaak and Gerritsen [1] comment that a system with a vapour in equilibrium with a liquid has a rather large heat capacity since some of the fluid has to be evaporated in order to raise the density of the vapour. However the small mass of vapour generated in the pilot-plant relative to the liquid mass means that the assumption used is satisfactory.

Other constants

The Fanning friction factor, f, (equation (3.13)) for stainless steel was estimated using typical steel roughness properties from Perry *et al* [3]. The value of f was taken as 0.009 for each effect.

Distribution plate friction coefficient, K_h, (equation (3.1)) was not investigated and was assumed to be unity, that is, zero friction. The role of the distribution plate sub-system equation within the total model is small and this assumption does not pose a significant problem.

The flow characteristic constant for the manual outlet valve (C_V) has not been determined from any experimentation on the particular hand valve. When the evaporator is operated this valve remains fully open and is not altered. Pneumatic control valves which have been used for the recirculating evaporator and which are of a similar size to the manual valve have a characteristic constant of the order of 5×10^{-7} when fully open. It is expected that a similar value to this is likely for the manual valve. An appropriate value for C_V was determined as part of the model validation.

3.2 Model implementation

3.2.1 Introduction to S-functions and M-files

The evaporator model was implemented using the software package MATLAB along with its dynamic simulation environment SIMULINK [12]. MATLAB commands are carried out by script or function files known as *M-files*. These files are written in MATLAB language and have a *.m* filename extension. In MATLAB, dynamic systems are represented by *S-functions*. An S-function is a file or function which SIMULINK interacts with for simulation and analysis. S-functions can either be generated graphically by creating a SIMULINK block diagram or as text in M-files or coded as C or FORTRAN subroutines. S-functions can be called from the MATLAB command line or be included within a block diagram.

The differential equations for the evaporator model have been written as S-function M-files and have been embedded in a block diagram representation of the evaporator system. Coding the differential equations in M-files is a much simpler approach than graphically defining the equations. It also allows the easy inclusion of any algebraic equations. One disadvantage is that an S-function implemented as an M-file runs considerably slower than an equivalent SIMULINK block diagram or C routine.

An S-function written as an M-file requires a specific structure for SIMULINK to correctly perform the differential calculations and generate the system outputs. It must provide SIMULINK with clear definitions of the variables involved and how the rates of change and system outputs are determined. The S-function handles the various requests from SIMULINK to perform the dynamic calculations through the use of a flag variable. The S-function can either return the initial conditions, the state derivatives or an output vector depending on the value of this flag variable, as in Table 3-1;

Table 3-1: Flag options for S-functions

Flag	S-Function output
0	Size of parameters and initial conditions
1	State derivative (dx/dt)
2	Discrete state $x(n + 1)$
3	Output vector
4	Next time interval for discrete system

A flag call of 1 is used for continuous states, while for discrete states flag calls of 2 and 4 are used.

3.2.2 The evaporator model

For the evaporator model there is an S-function file for each effect and each preheater. These S-functions contain all the differential equations and associated algebraic calculations required for each section of the evaporator. Each S-function is embedded into the SIMULINK block diagram for the evaporator model (*ffevap.m*) through S-function blocks. The S-function files for each effect are reproduced in Appendix 4(*a*). The plant simulation is initialised by defining the initial conditions within global variables *INIT1*, *INIT2* etc. which are referred to at the beginning of the S-functions.

M-files used to perform additional calculations within the model are given in Appendix 4(*b*) these functions include:

calca.m Calculates the cross-sectional area and volume of the effect level,

ffcalcq0.m Feed flow (Q_0) calculation based on current plant conditions,

calctp.m Calculates an estimate of the time delay between effects (τ_p) which is implemented using a variable time delay block in SIMULINK.

calctph.m Estimates the residence time through the preheaters, τ_{ph}.

convert.m Estimates the thermophysical properties of water for a given temperature (or pressure).

The SIMULINK block diagrams in the figures on the following pages show the flow of information in the model and how the S-functions are linked together. There are three levels to the evaporator SIMULINK model *ffevap.m*.

i) The top level (Figure 3-17) shows the main inputs and outputs of the model. It is in this level that various inputs can be applied or controllers can be introduced.

ii) The middle level (Figure 3-19) is where the main structure of the model can be observed. This shows the preheater, effect and venturi blocks linked together and the information paths between them.

iii) The lower level is where the information is passed into the S-functions M-files from the Simulink environment. An example of this is shown in Figure 3-18 for the first effect system.

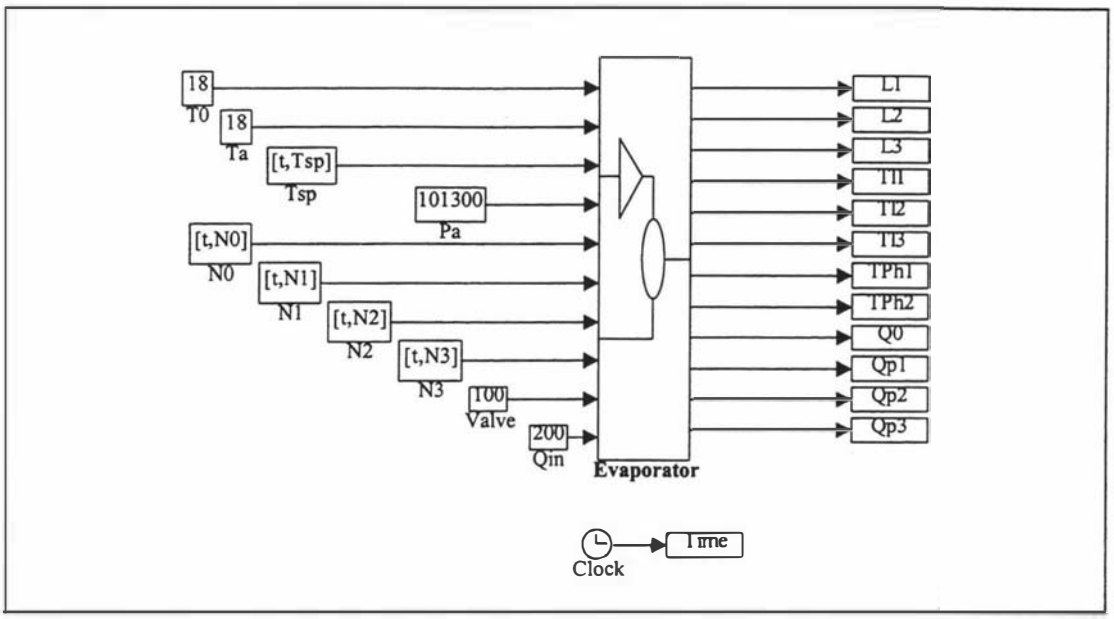


Figure 3-17: Top level of the *ffevap.m* SIMULINK block diagram

The venturi condenser system, although not containing any differential equations, is written as a separate M-file (*venturi.m*) with its own block within the block diagram. The calculation of the feed flow (Q_0) and the steam supply temperature (T_s) are also performed separately from the S-functions. The remainder of the lower level blocks making up the evaporator SIMULINK model are shown in Appendix 4(*c*). The central parts of these blocks are S-functions or M-files. These figures show the arrangement of the inputs and outputs to and from the M-files.

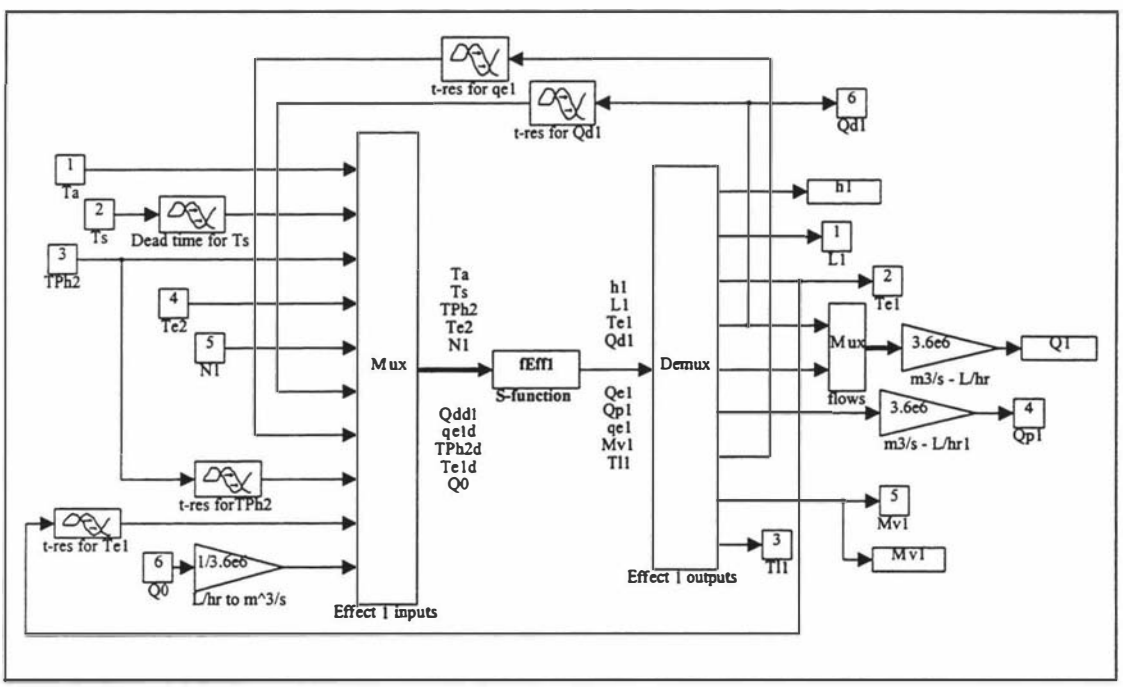


Figure 3-18: An example of the lower level of the *ffevap.m* SIMULINK block diagram - *Effect 1*

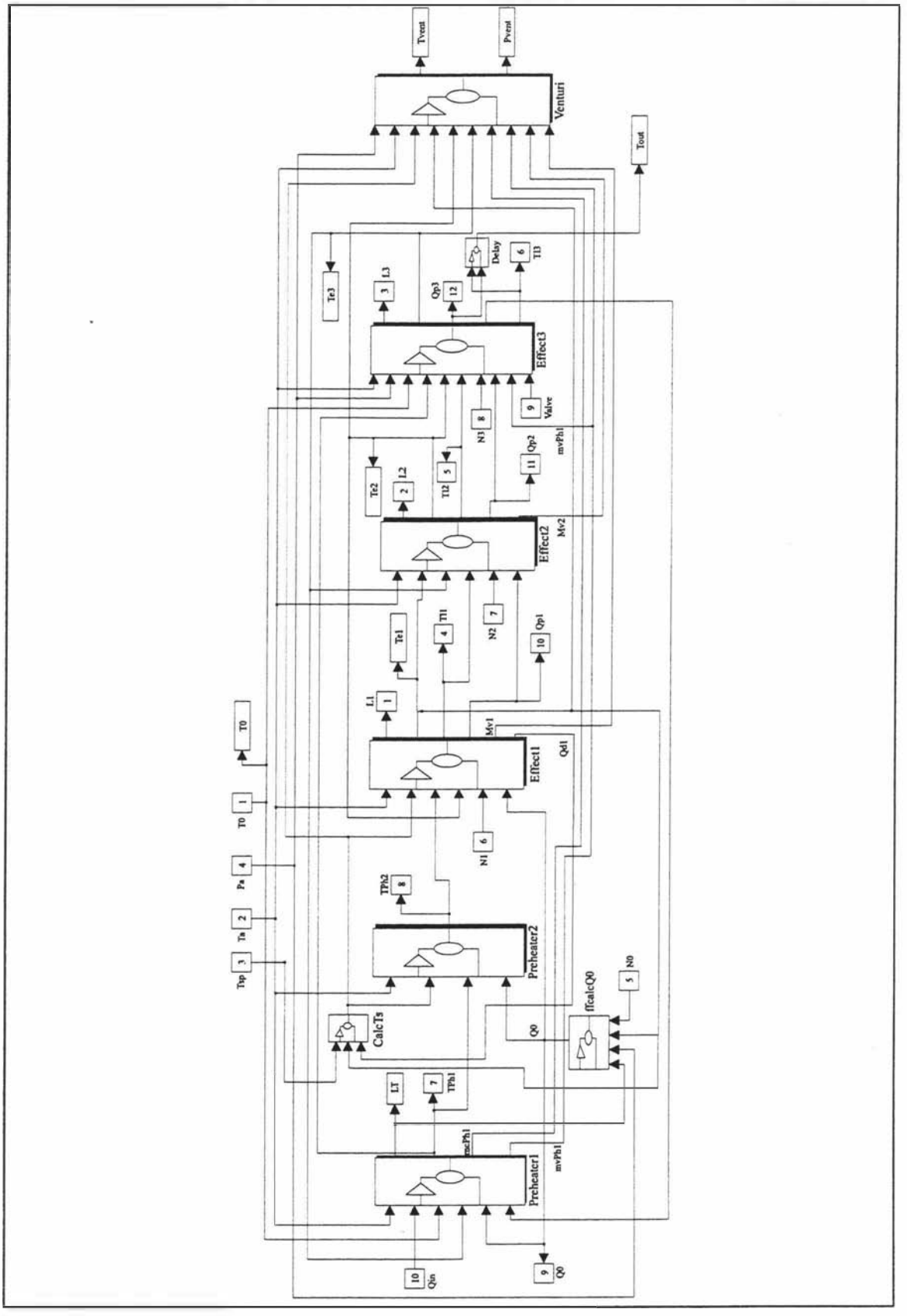


Figure 3-19: Middle level of *ffevap.m* SIMULINK block diagram

3.3 Model validation

The aim of the model validation is to determine whether the model is an adequate representation of the physical process. Some of the model parameters are likely to need adjusting from the theoretical or experimental values in order to improve the model performance.

It is impossible to obtain a perfect model with no error so the question is: How good does the model need to be? As discussed at the beginning of this chapter the model is essentially required for use as a simulation tool for control system design. Considering this, the model should correctly represent the dynamic behaviour of the evaporator system and indicate when the plant is going unstable. Therefore a small amount of error within the model may be acceptable as long as the model estimates follow the dynamics of the system and remain within the vicinity of the actual values. This is especially important for critical control-variables like the product temperatures.

The parameters that were used to adjust the model were the heat transfer coefficients for both the heating of the product (U) and the heat losses to the surroundings (U_{loss}), and the flow characteristic constants for the pumps (a_P), nozzles (C_N) and outlet valve (C_V). All other parameters were assumed to be fixed.

The heat transfer coefficients (HTCs) were calculated initially using general theoretical equations. These equations have a number of assumptions about the plant conditions and flow characteristics. These theoretical values are considered to be approximations which may need adjustment to improve the model estimates.

The pump and nozzle flow constants were obtained experimentally from tests performed on the actual plant and should give reasonable approximations. However there is no guarantee that the resulting constants are valid during normal operation of the whole system. This is because the tests were carried out under experimental conditions when only an isolated portion of the system was used and there was no influence of the normal operating temperatures and pressures.

It must be remembered that the equations themselves are only approximations to reality therefore there is a limit to how accurately we can produce plant estimates without having to re-formulate parts of the model. This is especially the case for a lumped-parameter system such as the evaporator model where many assumptions are made.

3.3.1 Plant trials

The validation process involves applying the same set of inputs to the evaporator system and to the simulation model and comparing the results.

Initially steady-state plant data was used to set the model parameters. These parameters were further adjusted by comparing the model data with the plant data when step changes and random sequences are applied on the system inputs.

An iterative procedure of running the simulation, comparing the model results with the plant data and adjusting the model parameters was carried out until an adequate representation was achieved.

The inputs which can be altered on the plant are the feed pump, the three effect pumps and the steam valve. The measurable outputs which are of interest are the preheater and effect temperatures, the product flowrates and the concentrate levels.

The external steam temperature (T_s) is technically an input variable, however as previously discussed the steam temperature is calculated by the model as function of the steam controller setpoint. On the actual plant the steam pressure is measured in three places: on the steam line after the steam valve, in the shell of the second preheater and in the shell of the first effect. The saturated temperature was used to estimate the steam temperature at the measured steam pressure using the *convert.m* routine. The steam pressure measured in the shell of the first effect was used to calculate T_s since it was this pressure which the steam valve controller was regulating. The gains of the PID steam pressure controller used for the trials were $k_p = 4.0$, $k_i = 0.1$ and $k_d = 1.0$. These had been determined on previous plant runs and had been adjusted on a trial-and-error basis.

On the pilot-plant the product temperature is measured at the base of each effect and the pressures are measured in the shell of each effect. The temperature in the evaporation tubes can be estimated by assuming that the tube pressure is equal to the shell pressure of the following effect downstream and that the vapours are saturated. Then the tube temperature is equal to the saturation temperature at the measured pressure in the downstream effect.

One would expect that the temperature in the evaporation tube would be greater than the measured temperature at the base of the effect. However, when we convert the measured pressures to their respective saturation temperatures these are lower than the measured temperatures. This indicates that the assumption of saturated vapour in the shell is invalid and so the measured pressures do not necessarily provide good estimates of the tube temperatures.

For the remainder of the discussion the product temperature in the reservoir at the base of the effect (T_l) will be referred to as T_n where n is the effect number. The 'l' subscript is dropped to simplify the notation since the only product temperatures of interest during the validation are the level temperatures. Similarly the 'p' subscript is dropped from the effect flowrates for the validation discussion and so Q_{pn} becomes just Q_n.

Steady-state operating points

Steady-state data was collected from the plant at various levels of steam temperature and feed flowrate. The effect level controllers remained active throughout the trial to maintain constant product levels in the effects. After making a change in the operating point the plant was allowed to settle down. An average measurement over a 4 or 5 minute period was then taken as the steady-state value at that operating point. Nine different operating points were obtained by adjusting the steam temperature and the feed flowrate. This trial is referred to as *Trial 0*. A table of the steady-state data which resulted from this trial is shown in Appendix 5(*a*).

Step response

The next step in the validation process was to obtain plant data which contains the dynamic characteristics of the plant. A trial with step changes to the steam temperature setpoint (*Trial 1*) was applied to both the simulation model and the actual plant. The plant data from this trial was used in combination with the data from the random input trials to determine the best values for the heat transfer coefficients.

Random input sequences

It was desired that a variety of combinations of inputs were applied to the system in order that all the possible modes of the system were excited and to reveal a complete picture of the dynamics.

Using an experimental design strategy one can adjust the plant inputs so that each possible combination of high and low levels on the inputs occurs during the plant trial. To avoid having to perform a formal experimental design, Pseudo-Random Binary Sequences (PRBS) can be applied to each input in order to achieve all the possible input combinations. Each input is varied between a high and a low value. At a constant sampling period there is a random chance that a particular input value may change. The probability of the change can also be adjusted so that there is less than 50% chance of a particular input changing levels. This type of random input scheme enables each input to remain at each level for both long and short periods of time. The random nature of the signal ensures that over a sufficiently long time-period each combination of input level is experienced. Applying a PRBS to a system is often easier than determining a formal experimental design scheme. This is especially the case when more than two levels are required for each input. The data used for this validation was required to demonstrate the nonlinear dynamic nature of the plant and so more than two levels were used for each input variable so that a more detailed set of data could be obtained. In

effect the input sequences were not binary sequences but instead could be named Pseudo-Random Multi-level Sequences (PRMS)

The random input data that was generated was also used in the task of identifying nonlinear artificial neural network models which require system data having the inputs varying between more than two levels. This will be discussed further in Chapter 6.

During the plant trials the pump controllers were turned off but the steam valve controller remained active since it was included as part of the analytical model. In generating the data, random input sequences were applied to the four pumps and to the setpoint of the steam pressure controller.

Four input levels were used for the three effect pumps. The feed pump and steam pressure setpoint are able to be varied over wider operating ranges without causing instability in the process. They were therefore excited between sixteen different input levels. The design steady-state operating point for the evaporator was achieved when the steam pressure was 50 kPa and the feed flowrate was 200-220 l/hr. Each of the random sequences were centred around this operating point for the random sequences. The sampling time for the random signals was 15 seconds for the pumps and 30 seconds for the steam pressure.

The ranges or the levels used for the random inputs are given in the following tables:

Table 3-2: Input ranges for the feed pump and steam pressure setpoint.

Input	Input range
Feed pump (N_0)	400-990 rpm
Steam pressure setpoint (P_{sp})	40 - 60 kPa

Table 3-3: Levels for the effect pump random inputs.

Input	Low (rpm)	Mid-low (rpm)	Mid-high (rpm)	High (rpm)
Pump 1 (N_1)	1170	1234	1298	1360
Pump 2 (N_2)	1522	1613	1673	1760
Pump 3 (N_3)	1808	1828	1848	1868

Some examples of the random input sequences applied on the evaporator can be seen in Figure 3-20.

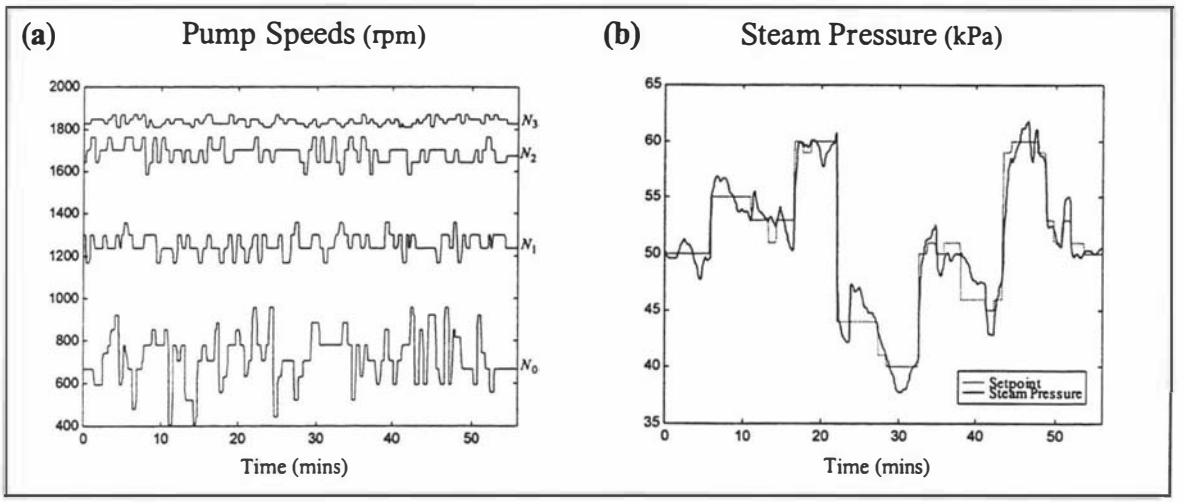


Figure 3-20: Example of random inputs applied to the evaporator

Three trials were carried out using different sets of random inputs, each set having similar characteristics. The three random input trials were named *Trial 2*, *Trial 3* and *Trial 4*.

Table 3-4: Plant data sets for model validation

Trial data	Length	Role
Trial 2	2 hr 6 min	Parameter selection
Trial 3	1 hr 33 min	Parameter selection
Trial 4	53 min	Model validation

The first two random input data sets were used to adjust and select the model parameters. The third set was used to perform an unbiased test of the simulation model after the model had been fit to the other data sets.

3.3.2 Comparison of the simulation and plant data

Steady-state analysis

Estimates of the heat transfer coefficients (HTCs) for the model can be gained from using the steady-state data with the model equations. With the derivative terms set to zero the model can be re-arranged to make the HTCs the subject of the equations. To determine the mean HTCs using the steady-state data it was assumed that the theoretical heat loss coefficients in Table A3-8 of Appendix 3(*b*) were correct. A comparison of the steady-state HTCs with 95% confidence limits and the theoretical values is given in Table 3-5.

Table 3-5: Comparison of theoretical and steady-state HTCs

HTC	Theoretical value ($W/(m^2K)$)	Mean steady - state value ($W/(m^2K)$)
U_{ph1}	1700	1935 ± 288
U_{ph2}	1770	1809 ± 252
U_1	2410	3358± 1743
U_2	2310	3097± 992
U_3	1930	1786± 153

The theoretical HTCs are in good agreement with those obtained from the steady-state plant data. The plant data values tend to be slightly higher with the exception of the third effect HTC (U_3).

The steady-state data was also used to estimate values for the flow characteristic constants. An optimisation routine was used with the model flow equations to determine the optimal values of the parameters. These will be discussed more in the following section.

Flow characteristics constants

Optimal values of the pump, nozzle and valve constants were calculated using an optimisation routine and the model flow equations. The optimisation was carried out using the data sets from Trials 0, 2, and 3 . Tables A5-4 and A5-5 in Appendix 5(*b*) summarise the parameter values. The mean values and the 95% confidence limits for each parameter from these optimisation calculations are given below in Table 3-6. The flow constants derived from the plant data are also compared with the values estimated through experiments on the plant which are calculated in Appendix 3(*e*). These mean values were implemented within the simulation with successful results.

Observing the two sets of values one can see that the model constants are not too dissimilar from the experimental values. The pump, nozzle and valve constants determined experimentally by Illingworth [2] are within or close to the 95% confidence intervals of the values determined from the plant trial data. The error bounds of the experimental values are small suggesting that the conditions encountered during the experiments were fairly steady. The flow constants calculated from the optimisation are effectively mean values over a range of operating conditions, hence the wide bounds on the limits.

The feed and the first two effect pumps are physically identical and hence their pump constants should be similar. However the feed pump constant is considerably higher than that of the other two pumps. This is probably because the feed pump operates with a negative differential pressure and therefore requires a considerably lower pump speed. The third effect pump has to pump against the partial vacuum therefore is required to operate at a faster speed. However, its speed requirement is reduced as it has a larger impeller than the other pumps. These differences are also reflected in the third effect pump constant.

Table 3-6: Pump, nozzle and valve characteristic flow parameters before and after model validation.

Constant	Units	Experimental value ($\times 10^{-2}$)	Model value ± 95% CI ($\times 10^{-2}$)
a_{P0}	Pa/rpm^2	2.267 ± 0.02	3.71 ± 1.42
a_{P1}	Pa/rpm^2	2.267 ± 0.02	2.14 ± 0.24
a_{P2}	Pa/rpm^2	2.267 ± 0.02	1.92 ± 1.53
a_{P3}	Pa/rpm^2	2.666 ± 0.04	3.00 ± 0.16
Constant	**Units**	**Experimental value ($\times 10^{-7}$)**	**Model value ± 95% CI ($\times 10^{-7}$)**
C_{N1}	m^3(Pa^{½}s)	6.65 ± 0.04	4.91 ± 1.04
C_{N2}	m^3(Pa^{½}s)	6.65± 0.04	5.45 ± 1.37
C_{N3}	m^3(Pa^{½}s)	2.97± 0.10	2.89 ± 1.51
C_V	m^3(Pa^{½}s)	~ 5	5.90 ± 1.86

The nozzle constants for the first two effects are similar since they have the same size holes. The third effect nozzle has larger holes hence its nozzle constant is smaller.

The model pump, nozzle and valve constants in Table 3-6 were used in the following simulations. They proved to give satisfactory flowrate estimates when compared with the actual data.

Simulation initialisation

For each plant trial the input sequences for the steam temperature setpoint (T_{sp}) and the pumps (N_0, N_1, N_2, N_3) were applied to the analytical simulation. The simulation used a third order Runge-Kutta integration algorithm with second order step size control.

Initially the simulation used the mean values for the pump, nozzle and valve constants rounded to two decimal places and the HTCs obtained from the steady-state analysis discussed previously.

The initial conditions for the simulations were set as being equal to the first data value for each variable of the actual plant data. The initial values for the vapour flows (M_v) were obtained after optimising their value based on the product flow and level information given in the random trial data. After trialing different values and observing the resulting level and product flowrate estimates the initial effect vapour flows were set to 10 kg/hr for the first and second effects and 20 kg/hr for the third. The initial value for the first preheater condensate flow (m_{cph1}) was set at 8 kg/hr and the mass flow of uncondensed vapour from the first preheater was set to be constant at 12 kg/hr. If this vapour flow was varied according to equation (3.60) it was found that the third effect temperature fluctuated too much.

The effect tube temperatures (T_e) for each effect were initialised at a temperature greater than the effect level temperature (T_l).

$$T_e(0) = T_l(0) + \Delta T$$

with ΔT equal to 0.5 °C for effect 1; 0.3 °C for effect 2 and 0.2 °C for effect 3.

These values were used after the initial few simulations revealed these to be the mean temperature differences.

It also became apparent that two dead-times needed inclusion into the model for the steam temperature entering the first effect and second preheater systems. This eliminated any time lags apparent in the model temperature estimates. The two dead-times were estimated as 25 seconds for the steam temperature entering the second preheater and 35 seconds for the first effect.

Heat transfer coefficients

For the initial simulations using the random input and step change trial data, some of the model estimates were not in good agreement with the actual data. After a certain amount of trial-and-error adjustment of the HTCs the values in Table 3-7 (model column) were found to give satisfactory results.

The model heat transfer coefficients are not too dissimilar from that determined from the steady-state analysis and have good agreement with the theoretical values. The coefficients used in the model are all within 12% of the theoretical values. Generally the model HTCs appear to represent a compromise between the steady-state values and the theoretical values. The heat loss coefficients used in the model remained unchanged from the figures calculated theoretically. The HTCs reported here represent a best

guess. Other combinations of coefficients might be found which give similar or even better results than those presented here.

Table 3-7: Heat transfer coefficients before and after model validation

Constant	Theoretical value ($W/(m^2K)$)	Mean steady-state value ($W/(m^2K)$)	Model value ($W/(m^2K)$)
U_{ph1}	1700	1935	1900
U_{ph2}	1770	1809	1800
U_1	2410	3358	2600
U_2	2310	3097	2500
U_3	1930	1786	1800

The data from the simulation using the above model parameters was compared with the actual plant data for the step change trial and each of the random input trials.

Step response trial

We will firstly consider the performance of the simulation in modelling the step response data. The step change was applied to the steam temperature setpoint. Figure 3-21 shows this step change, the measured response and the model output for the steam temperature.

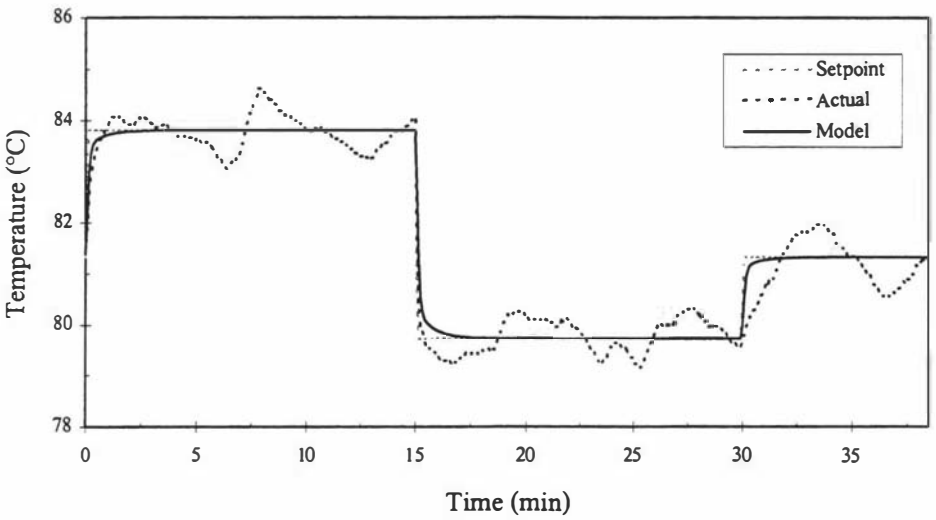


Figure 3-21: Trial 1 step changes and steam temperature response.

Overall the model performs satisfactorily when simulated with the step change data. The simulation estimates are compared with the actual data in plots given in Appendix 5(*c*). Figure 3-22 to Figure 3-24 shows some of the model predictions compared with

the measured plant data of Trial 1. Generally the temperature outputs from the simulation perform satisfactorily, capturing the basic dynamics of the variables. The concentrate level estimates are generally good although some offset is apparent. In estimating the flowrates the model effectively produces a mean value. The Q_0 and Q_3 estimates are good however there is some bias in the first and second effect flow estimates. The model produces flowrate estimates which have a smooth response whereas the data from the plant exhibit considerable measurement noise.

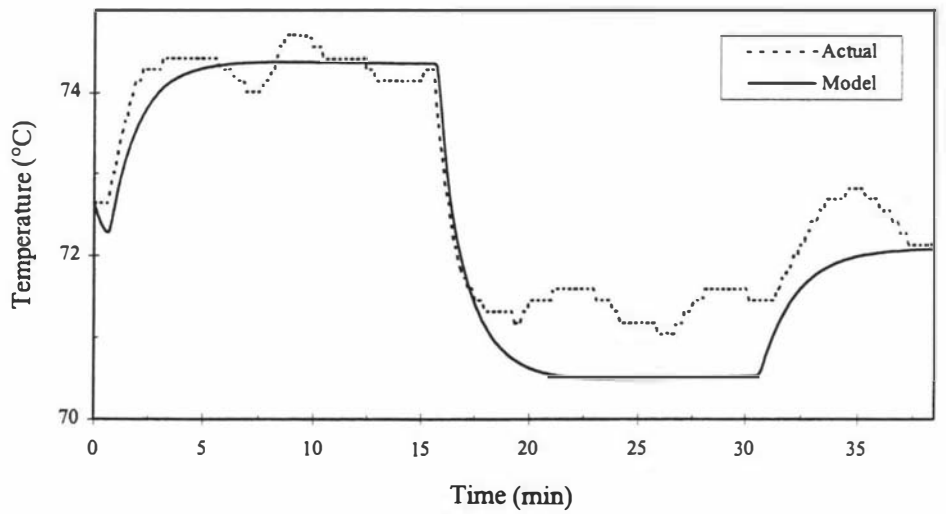


Figure 3-22: Comparison of T_1 model estimate with Trial 1 data.

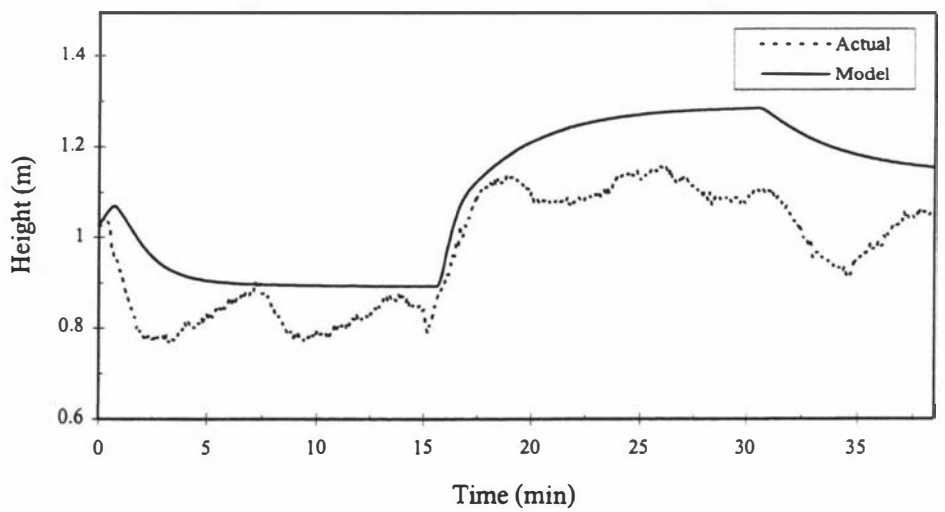


Figure 3-23: Comparison of L_1 model estimate with Trial 1 data.

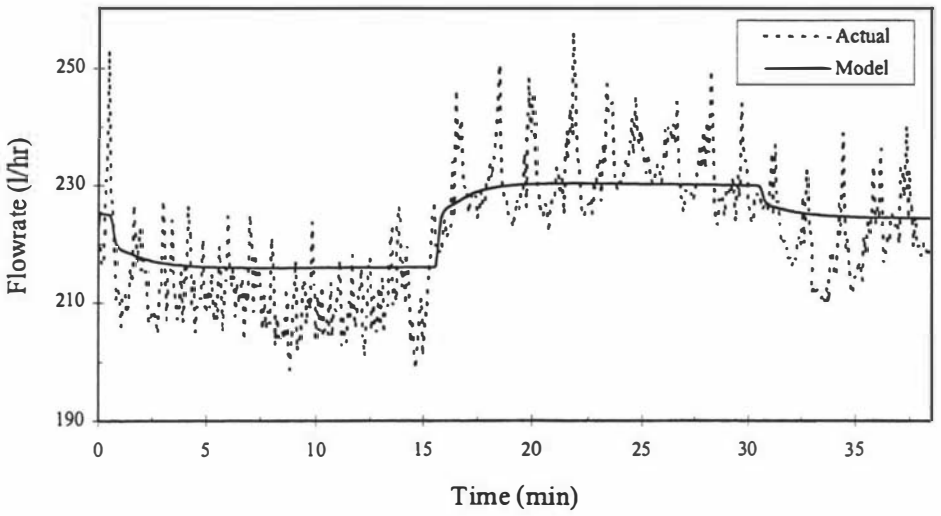


Figure 3-24: Comparison of Q_0 model estimates with Trial 1 data.

Steam temperature

The model's estimate of the external steam temperature, as described in Section 3.1.10, does not always mimic the steam temperature from the plant trials. The model calculation was based on a linear transfer function whereas the valve exhibits a deadband characteristic which is difficult to model. Comparing the actual steam measurement and the model estimate for Trial 1, (Figure 3-21) it is obvious that the model fails to successfully characterise the nonlinear behaviour of the steam valve when the pressure is settling down after a change. The current model does not adequately model the fluctuations causing a poorer estimation of the other variables, especially the temperatures.

Applying the actual steam measurement to the model as an input, rather than modelling the steam system, improves the performance of the model across all the variables, especially the temperatures. For the random input trials the steam temperature estimation also adversely effects the model performance however these simulations were carried out using the steam temperature calculation as part of the model.

Random input trials

Each of the three random input trials were run using the simulation model. Since the Trial 2 and Trial 3 data sets were used to select the HTCs and flow constants for the simulation model, the Trial 4 data set was used to perform an unbiased validation of the model. The model parameters used for the step response simulation were retained for the random input trials.

All the plots of the model estimates for Trial 4 for each variable of are shown in Appendix 5(*c*) and a few example plots are given below in Figure 3-25 to 3-28.

A plot of the steam temperature estimate compared to the actual data is shown in Figure 3-25. Again the model fails to capture the nonlinear fluctuations in the actual data. Instead the model estimates follow the steam temperature setpoint quite closely.

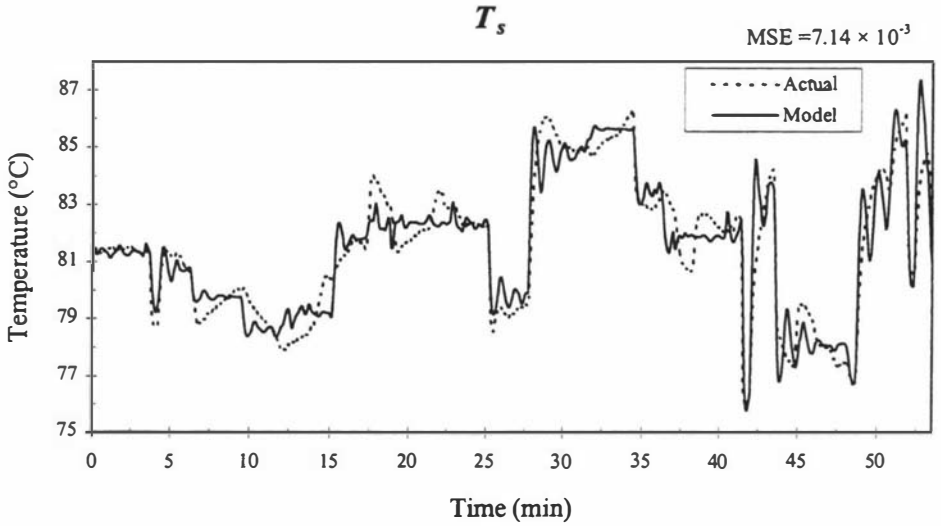


Figure 3-25: Comparison of steam temperature model estimate with Trial 4 data.

The performance of the model in estimating the plant variables based on the Trial 4 inputs is discussed below.

Product temperatures

In general the model estimates of the temperatures are quite good considering the difficulty in obtaining accurate estimates for the steam temperature. A plot of the T_1 estimate can be seen in Figure 3-26. All the product temperatures follow the dynamic characteristics of the actual temperatures reasonably well. There are instances where the model estimates fail to capture some of the peaks and troughs in the data and instead cut across these fluctuations. This is caused by the poor steam temperature estimation by the model. However these errors are still within 1°C of the actual value. The best temperature estimates are obtained for the second preheater and first effect but these get worse in the downstream sections since the errors accumulate through the plant model. Therefore the third effect and first preheater generally have the worst temperature estimates. This can be seen in the plots of the model estimates in Appendix 5(*c*).

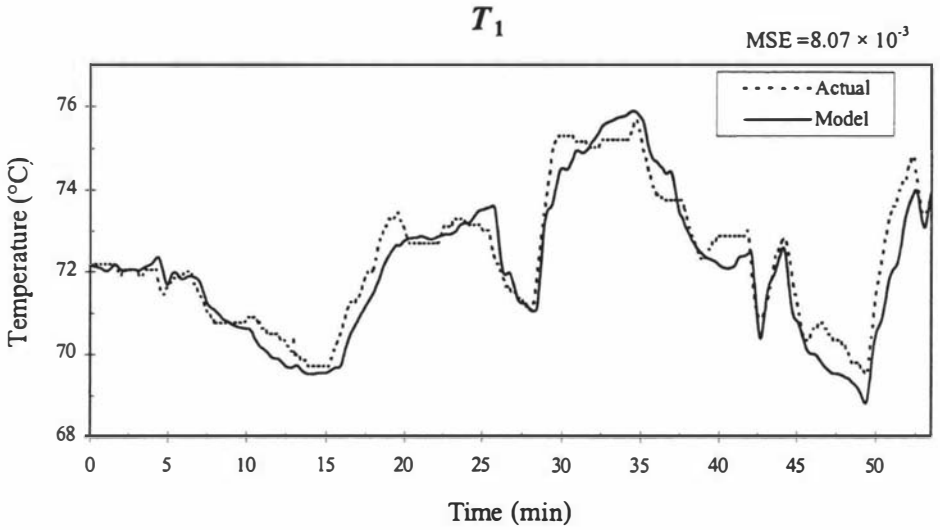


Figure 3-26: Comparison of T_1 model estimate with Trial 4 data.

Product flowrates

The flowrate estimates generally have a smooth path following the actual data since they do not contain the noise which occurs in the flowrate measurements. The model therefore does not always accurately estimate the amplitude of all the peaks and troughs of the flowrate. As an example the feed flow (Q_0) comparison is illustrated in Figure 3-27. The model estimates for the first and second effect flowrates tend to exhibit a positive bias for the Trial 4 data.

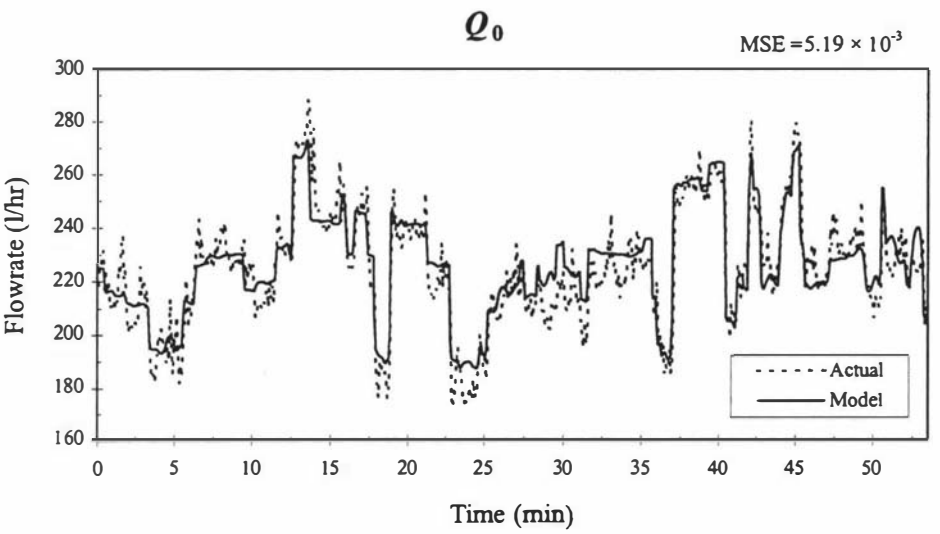


Figure 3-27: Comparison of feed flow model estimate with Trial 4 data.

Concentrate levels

The levels are the most difficult variables for the model to estimate correctly. This is because the level system acts essentially as an integrator of the difference between the

effect input and output flowrates. Any errors in the flow estimates will affect the level and these errors accumulate throughout the model simulation. This produces level estimates which generally capture the correct shape of the actual level measurements but which display an offset in amplitude. This is evident in the plot of the concentrate level for effect 2 in Figure 3-28. Therefore the MSE values for the levels are usually higher than for the other variables. Although the level estimates often do exhibit an offset it is encouraging that their trends do follow that of the actual data.

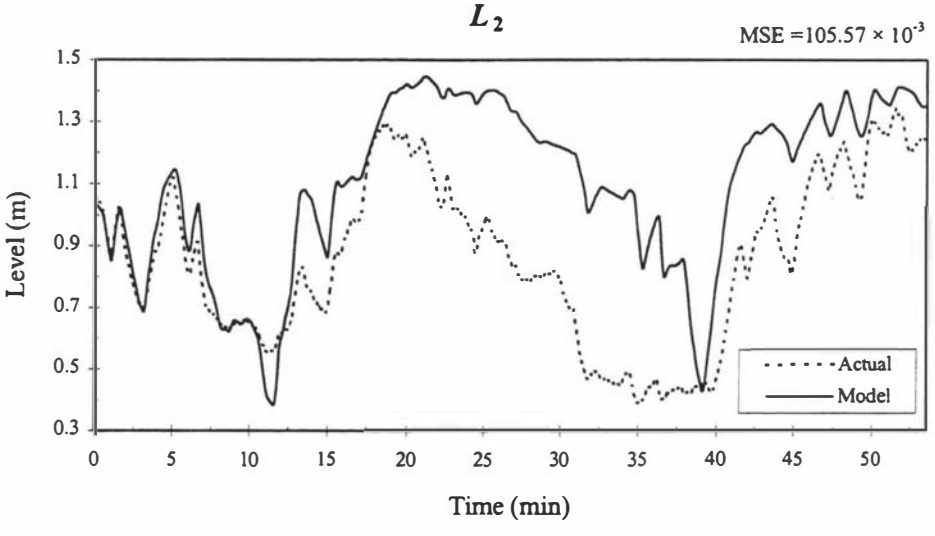


Figure 3-28: Comparison of L_2 model estimate with Trial 4 data.

Vapour flows

The initial conditions of the vapour flows were 10 kg/hr for the first and second effects and 20 kg/hr for the third. During the simulations the flows oscillated about these initial values. Figure 3-29 shows a plot of the three effect vapour flows as calculated by the simulation model for Trial 4. The evaporation rate (the sum of these three flows) was calculated to be between 34 and 46 kg/hr. This evaporation rate cannot be measured

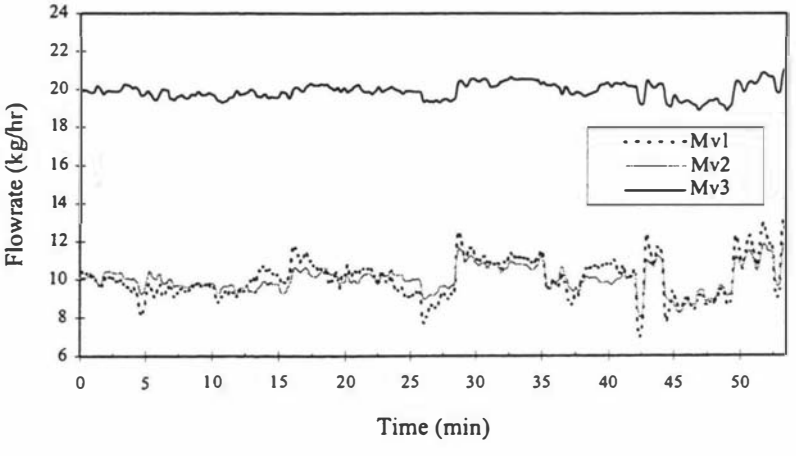


Figure 3-29: Model estimates of the effect vapour flowrates for Trial 4.

directly on the plant but the model estimate is in agreement with what is observed (usually between 35-60 kg/hr) during a period of steady operation when the flowrates and levels are held constant. For example, if we calculate the difference between the feed flowrate (Q_0) and the output flowrate (Q_3) for the nine steady-state operating points from the plant data given in Appendix 5(*a*) we can estimate the evaporation rate at those points. By doing so the nine evaporation rates obtained are between 33 and 60 l/hr with a mean of 46 l/hr, indicating that the evaporation rate estimated by the model is indeed in agreement with what occurs in practice.

Venturi condenser

The temperature of the discharge from the venturi condenser is not measured directly on the pilot-plant. However readings were recorded by hand periodically during the plant trial to compare with the model estimates of the temperature. The temperature measurements were all within a small range of 24.8 - 26.4 °C. A plot of the model estimate of the venturi outlet temperature is shown in Figure 3-30 with the measured temperatures indicated with dots. The model estimates oscillate between 27.8 °C and 30.4 °C with a mean of 29.5 °C. The measured temperatures are a lot lower than that estimated by the model. A possible reason for this discrepancy could be due to the measuring of the outlet temperature approximately 3 metres downstream from the venturi outlet although this is likely to only account for a small portion of the disagreement. The difference could also indicate that the amount of uncondensed vapour entering the condenser from preheater 1 is over-estimated by the model.

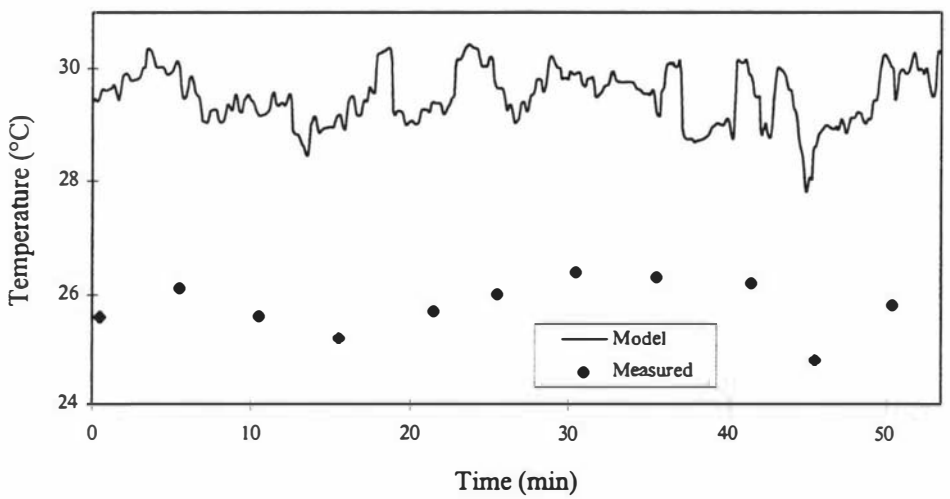


Figure 3-30: Model estimate of condenser outlet temperature (T_{vent}) for Trial 4

A plot of the venturi suction pressure estimated by the model is shown in Figure 3-31. It is compared with the actual pressure measured in the shell of the first preheater. This pressure sensor has the closest physical location in relation to the condenser. The

venturi pressure estimated by the model has a reasonable agreement with this measured pressure. The model suction pressure fluctuates within the range of 3.8 - 4.7 kPa while the measured preheater pressure varies between 3.7 and 5.2 kPa. The fact that the measured preheater pressure is close to the condenser pressure and not equal to the saturated pressure of the third effect vapour confirms that the assumption of saturated vapour in the evaporator shells is not valid.

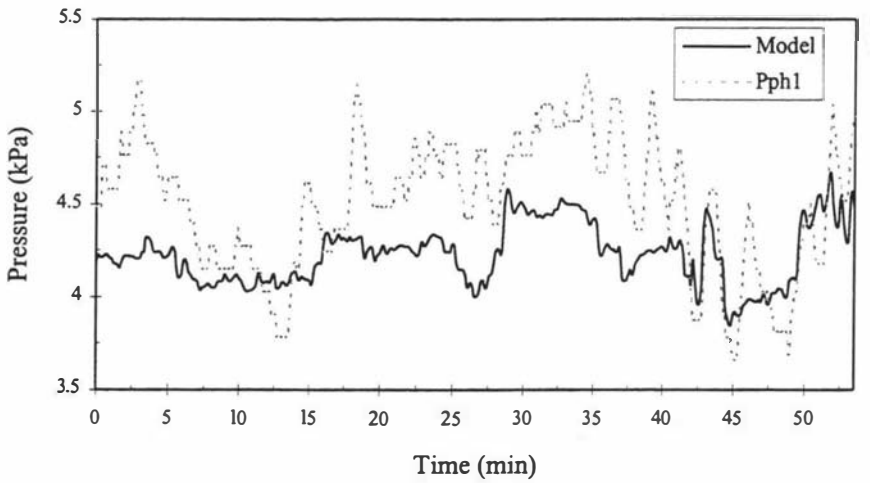


Figure 3-31: Model estimate of the condenser suction pressure (P_{vent}) for Trial 4

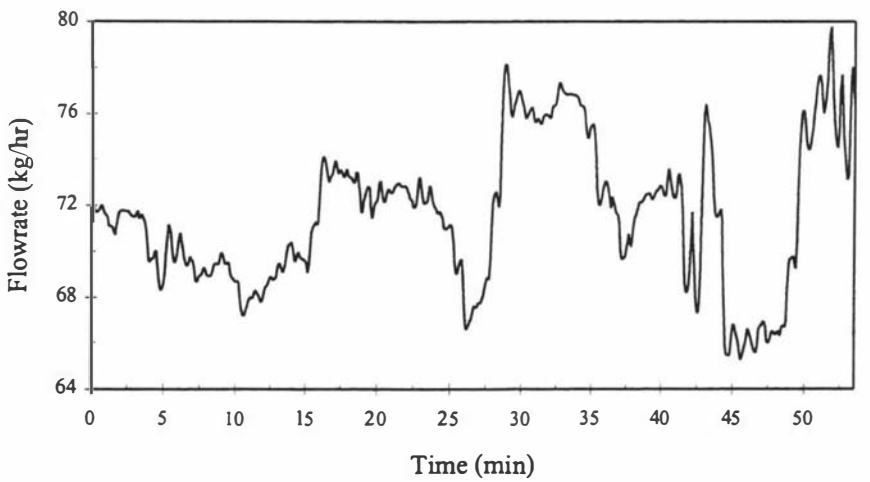


Figure 3-32: Model estimate of the total condensate and vapour flow into the condenser for Trial 4

The flows of condensate and vapour into the venturi condenser was also estimated by the model. A plot of the total mass flow of condensate and vapour is shown in Figure 3-32. This flowrate of condensate and vapour has a mean of 71.6 kg/hr and fluctuates between 65.3 and 79.8 kg/hr. During the running of Trial 4 measurements of the condenser outlet flowrate (m_{vent}) were taken using a bucket and stopwatch. Later the condenser driving fluid flowrate was measured when the plant was not operating. Six

flow measurements were taken for each situation and the mean flows were calculated. These are given below.

	Mean condenser outlet flow (kg/hr)
Plant operating	1122
Plant not operating	1054
Difference =	68

This gives an average difference in condenser outlet flow of 68 kg/hr. This increase in flow is made up of the entrained condensate and vapour. The measured flow is within the range of that calculated by the model and confirms that the model estimates of the total condensate and vapour flows are in agreement with the actual plant.

3.3.3 Conclusions

Summary tables for the parameters used in the model are given in Appendix 5(*b*).

The parameters determined during this validation process represent one particular solution for the model. Another set of parameters could be found which give improved model estimates for some of the output variables but perhaps not others.

The parameters that have been chosen here seem to produce reasonable temperature estimates while the level estimates are not optimal. It was felt that it was more important to get the best temperature predictions since the levels are always going to be problematical to estimate accurately. This is mainly due to the accumulation of error in the level estimates due to the integrating nature of the level system. The model generally produces flowrate estimates which are a moving average of the noisy measurements. The model does perform well in estimating the dynamic nature of all the variables even if there may sometimes be offsets in the magnitudes.

The model estimation of the condensate and vapour flows and the conditions in the venturi condenser all seem to be in agreement with what is observed on the plant apart from the condenser temperature which was estimated to be higher than the measurements.

One major disadvantage of the current model is the estimation of the steam temperature. Improving this part of the model would greatly improve the temperature estimates and as a consequence the other variables also. However despite this the model was shown to perform quite satisfactorily when used to estimate step response and random input data sets.

Another improvement that is necessary is the formulation of the model into C or a similar language in order to improve the speed of simulation. Presently the simulation model is cumbersome and runs at around one fifth of real time during simulations. The use of S-function M-files was good for the development of the model but they are significantly slower to run than C code.

Despite these limitations the model is a suitable tool for use in analysing the application of process control to the evaporator.

3.4 References

[1] **Quaak, P. & Gerritsen, J.B.M.**, Modelling dynamic behaviour of multiple-effect falling-film evaporators. In: *Computer Applications in Chemical Engineering*, Bussemaker, H.T. & Iedema, P.D. (Eds.), Elsevier, Amsterdam, 1990, pp. 59-64.

[2] **Illingworth, M.I.**, *Dynamic modelling of a three-effect falling-film evaporator.* Final Year Undergraduate Research Project, Dept. Production Technology, Massey University, New Zealand, 1993.

[3] **Perry, R.H., Green, D.W. & Maloney, J.O** (Eds.). *Perry's Chemical Engineers' Handbook*, 6th Edition, McGraw-Hill Book Co., New York, USA, 1984, p. **5**-24.

[4] **Kroll, A.E.**, The design of jet pumps. *Chemical Engineering Progress*, 1947, vol. 1, pp. 21-24.

[5] **Perry, R.H., Green, D.W. & Maloney, J.O** (Eds.). *Perry's Chemical Engineers' Handbook*, 6th Edition, McGraw-Hill Book Co., New York, USA, 1984, pp. **5**-12 – **5**-13.

[6] **Smith, R. A.**, Theory and design of simple ejectors. Chpt. 13 of *Some Aspects of Fluid Flow*, Edward Arnold & Co., 1951, pp. 229-241.

[7] **Perry, R.H., Green, D.W. & Maloney, J.O** (Eds.). *Perry's Chemical Engineers' Handbook*, 6th Edition, McGraw-Hill Book Co., New York, USA, 1984, pp. **10**-16 – **10**-20.

[8] **McAdams, W.H.**, *Heat Transmission*, McGraw Hill, New York, 1954.

[9] **Stephan, K.**, *Heat Transfer in Condensation and Boiling*, Springer-Verlag, Berlin, Germany 1992.

[10] **Billet, R.**, *Evaporation Technology*, VCH Publishers, Weinheim, Germany, 1989.

[11] **Cooper, J.R. & Le Fevre, E.J.**, *Thermophysical Properties of Water Substance*, Edward Arnold Ltd., London, 1986.

[12] MATLAB and SIMULINK, *The MathWorks Inc.*, 24 Prime Park Way, Natick, MA., USA.

4

Artificial Neural Networks

OVERVIEW

This chapter presents an introduction to artificial neural networks, their development and their use in system identification and control. Feedforward neural network techniques in particular are discussed and some of the common algorithms are outlined. Issues relating to the design of neural networks and techniques for applying them to dynamic system identification problems are discussed. The chapter concludes with an overview of the current methods of applying neural networks to nonlinear process control.

4.1 Introduction

4.1.1 Background

Artificial neural networks use a parallel processing paradigm inspired by the principles of learning and information processing in biological nervous systems. The field of artificial neural networks attempts to mimic the learning process that is believed to take place in real neural networks in order to develop tools for classification, prediction and modelling.

The human brain consists of many (in the order of 10^{11}) simple processing elements called neurons which are connected together in complex networks. Each neuron receives a number of input signals but has only a single output. The links between the neurons are stimulated via chemical or electrical impulses and represent the strength of relationship between the neurons that they connect. As the brain is stimulated the connections that receive positive reinforcement are strengthened and those that receive negative responses are weakened. Over the years the neural networks in our brains have been repeatedly presented with large amounts of information and various forms of

stimulation and through this experience we have learnt how to behave and respond to certain conditions which we face.

The manner in which artificial neural networks operate is an extremely simplistic representation of what occurs in biological nervous systems. Conceptually, an artificial neural network is made up of many simple processing elements (as in a biological brain) and provides a method for parallel processing a large quantity of data. They have proven to be effective for a range of practical problems where conventional computation techniques have not succeeded. Artificial neural networks are best suited for tasks for which humans are able to achieve well and for which conventional modelling approaches fail. Applications for which neural networks have been employed include pattern recognition, classification, prediction and forecasting, function evaluation and nonlinear system identification and control [1].

As a form of artificial intelligence, artificial neural networks differ from other techniques in that they do not rely solely on human expertise to gain the understanding of a particular problem or system. Instead they are able to be used to infer trends and relationships within the information to which they are repeatedly exposed.

This chapter will primarily concentrate on the application of artificial neural networks to function estimation or system identification and control as this is the topic of interest in this thesis. Hence this chapter will not be offering an overview of all the types of artificial neural networks in existence and the discussion assumes one is dealing with the task of forming a model of a real system. The reader should be aware however of the broad range of fields in which artificial neural networks are commonly applied.

Artificial neural networks can be used to obtain a model using actual data from a system under study. The network models the trends and relationships within the data and effectively generates a mapping of the system inputs and outputs based upon the relationships encountered within the data. After a period of repeated presentation of the data it is able to mimic the behaviour of the system or function which generated the data.

At times an artificial neural network will be referred to as just a 'neural network' within this work and should not be considered as a reference to a biological neural network.

4.1.2 Neuron models - perceptron, adaline and the Delta Rule

Neural networks are varied with many different types being explored in research or being developed for applications. Although they differ in their structures and the algorithms incorporated into their elements they are all networks of highly interconnected processing elements known as *neurons* or *nodes*. To understand how

neural networks work one must first investigate the structure and function of the neurons which form them.

The *perceptron* was conceived by Rosenblatt [2] in 1958 and has become the basic building block of many neural networks. Most neural networks make use of perceptron-type models as processing elements.

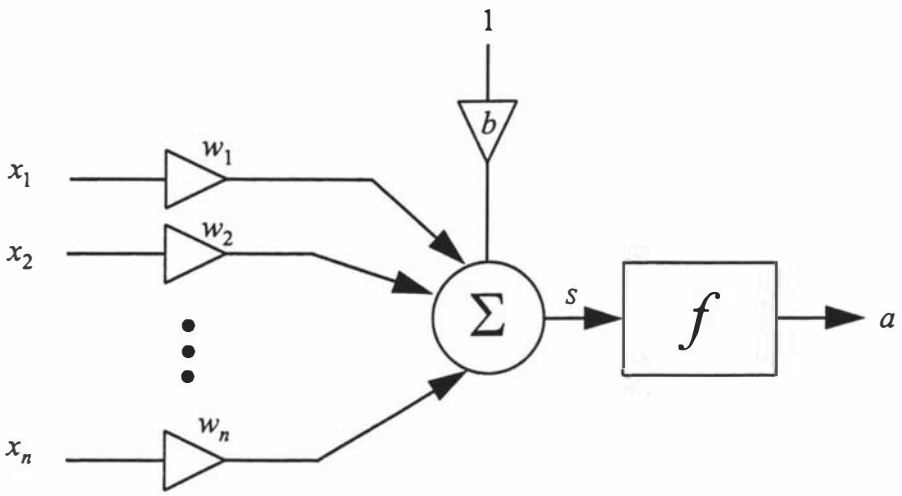


Figure 4-1: The perceptron model.

The perceptron (Figure 4-1) has a number of inputs and a single output. It forms a weighted sum of the elements of the *n*-dimensional input vector:

$$s = \sum_{j=1}^{n} w_j x_j \tag{4.1}$$

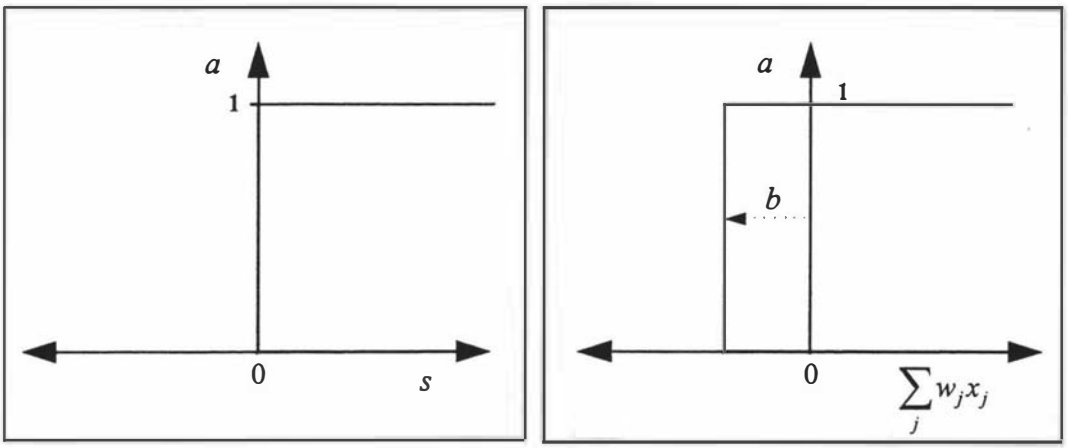
where x_j is the j^{th} element of the n x 1 input vector, X, and w_j is the connection weight on the j^{th} input.

The result is passed through a threshold or activation function to produce the output, a, which is called the *activation* of the neuron. Rosenblatt's original model used a hard-limiting nonlinearity as the activation function (Figure 4-2(a)), whereby:

$$a = f(s) = \begin{cases} 1, & s > 0 \\ 0, & s \le 0 \end{cases} \tag{4.2}$$

A bias input can be included in the summation to translate the nonlinear function along the horizontal input axis (Figure 4-2(b)). The bias input is held at 1 and so the bias weighing, b controls the horizontal position of the activation function.

$$s = \sum_{j=1}^{n} w_j x_j + b \tag{4.3}$$



(a) **(b)**

Figure 4-2: (a) Hard-limiter function, (b) Hard-limiter with bias.

If $\sum_{j=1}^{n} w_j x_j > -b$ then neuron will turn on ($a = 1$); if $\sum_{j=1}^{n} w_j x_j \leq -b$ then the neuron will turn off ($a = 0$).

The perceptron is able to recognise simple input patterns and classify them according to their values. The weights represent the strength of the relationship between the input vector and the neuron output. These weights can be set to values which produce the correct mapping of inputs to outputs.

The weights and the bias can be adapted using a number of different algorithms. The original perceptron convergence procedure for adjusting the weight was developed by Rosenblatt. Initially the weights and bias are set to small random non-zero values. Patterns of n-dimensional input vectors and their corresponding target outputs are then presented to the neuron.

The perceptron activation, a, for input pattern, p, is given by:

$$a^p = f\left(\sum_j x_j^p w_j + b\right) \tag{4.4}$$

After presentation of each vector the neuron's activation is compared with the target value. The weight on a particular input connection is updated according to the error between the neuron output and the desired output, d, multiplied by the input through the connection. Therefore the weight at iteration $k + 1$ is:

$$w_j(k+1) = w_j(k) + \Delta w_j(k) \tag{4.5}$$

where:

$$\Delta w_j(k) = \mu\left(d^p - a^p\right)x_j^p \tag{4.6}$$

$0 < \mu < 1$ is a constant known as the learning rate which controls the rate of change of the weights.

The process of presenting example data to a neuron and determining the correct weights to represent the data is referred to as *training*. As the weights are adapted it said that the neuron 'learns' to model the data.

Rosenblatt showed that a single perceptron could correctly classify inputs only if the input vectors were linearly separable, that is they fall on opposite sides if a hyperplane (a line in 2-dimensional space). If this is the case the perceptron convergence procedure converges and positions the decision boundary between the n classes of inputs. If the inputs are not linearly separable the procedure oscillates continuously. An example of a set of linearly inseparable vectors is the XOR function (Figure 4-3). No single straight line can be drawn to separate the two classes.

x_1	x_2	$x_1 \otimes x_2$
0	0	0
0	1	1
1	0	1
1	1	0

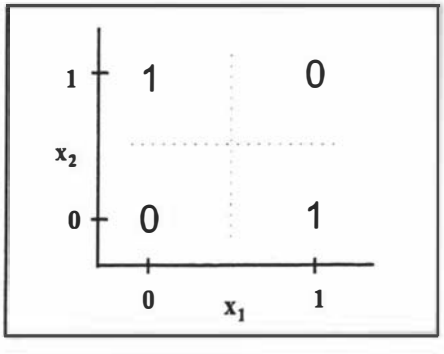


Figure 4-3: Truth table and plot for the XOR function.

At about the same time Widrow and Hoff were developing similar ideas [3]. They had conceived a processing element similar to the perceptron which they named an ADALINE (Adaptive Linear Element). A modification to the perceptron convergence procedure forms the adaptation algorithm Widrow and Hoff used. Their algorithm finds the Least Mean Square solution of the decision boundary which minimises the mean square error between the desired and actual outputs and hence is called the Least Mean Square (LMS) algorithm but is also often referred to as the Delta Rule or Widrow-Hoff algorithm [4].

The LMS algorithm is identical to the perceptron convergence procedure described above except the hard limiter nonlinearity is replaced with a linear activation function which gives continuous valued outputs rather than binary outputs. The activation therefore is:

$$a^p = \sum_j x_j^p w_j + b \tag{4.7}$$

The sum-of-squared error term for a neuron output with input vector p is defined as:

$$E^p = \frac{1}{2}\left(d^p - a^p\right)^2$$

$$= \frac{1}{2}\left(d^p - \sum_j x_j^p w_j + b\right)^2 \tag{4.8}$$

The objective is to find a set of weights to minimise E^p. The LMS algorithm sets the change in the weights to be proportional to the negative of the derivative of error with respect to each weight. That is;

$$\Delta w_j = -c\frac{\partial E^p}{\partial w_j} \qquad c > 0 \tag{4.9}$$

By partial differentiation of equation (4.8) with respect to w_j, and substitution into equation (4.9), one can obtain:

$$\Delta w_j = \mu\left(d^p - a^p\right)x_j^p \tag{4.10}$$

where $\mu = c$.

This is equivalent to the perceptron convergence equation (4.6). The algorithm searches over the weight-space for the point of minimum error and does this by moving in the direction of greatest improvement in error, that is the largest value of $-\partial E^p/\partial w$. This gradient descent method of adaptation is known as a steepest descent optimisation where the movement is in the direction opposite to the error gradient.

There are many other algorithms which have been developed for the perceptron, most of them in the 1960's [5]. The LMS algorithm however is a special case of the *backpropagation* method which will be discussed shortly and has gained widespread application.

4.1.3 The multi-layer perceptron

The limitation that convergence occurs only for linearly separable vectors remains true for neurons using the LMS algorithm. However when perceptrons are cascaded together in layers, the limitations of single layer networks are overcome. Such a network is known as a *multi-layer perceptron* (MLP) (Figure 4-4). An MLP is an example of a *feedforward* network as all nodes are fully connected to the nodes in the next layer and there are no feedback connections. (In contrast a network which consists of connections between nodes in the same layer or previous layers is referred to as a *recurrent*

network.) The neuron-layers not connected directly to either an input or an output node are termed *hidden* layers.

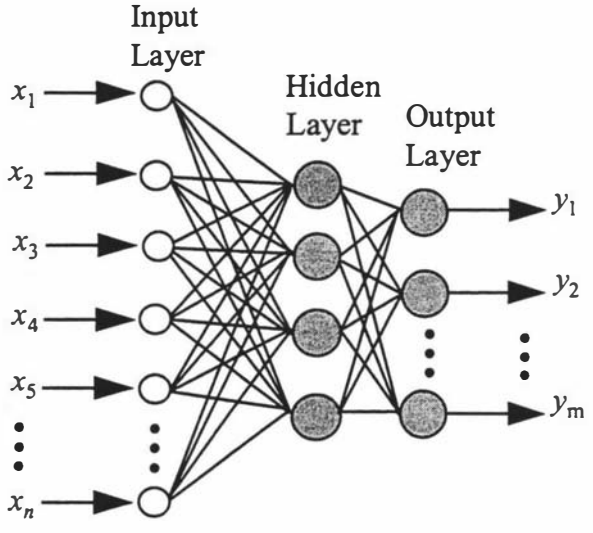


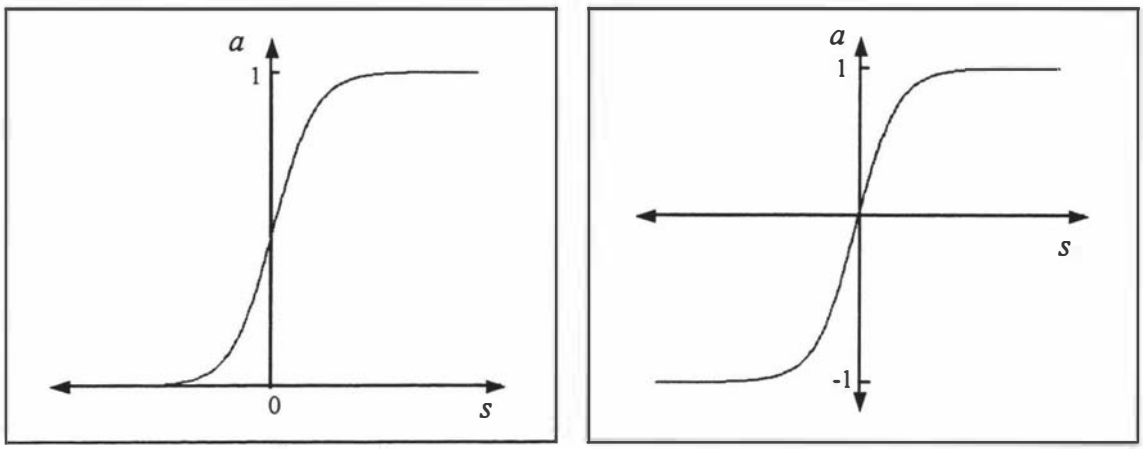
Figure 4-4: A two-layered feedforward MLP.

As for a single perceptron, each connection carries a weighting which reflects the strength of the relationship between the neurons it connects and can be chosen to achieve a desired input/output relationship. A particular weight is raised to positively reinforce the relationship between the neurons it connects but is reduced to negatively reinforce the relationship. The aim is to drive the network weights to values which minimise the difference between the predicted and desired outputs.

Nonlinear activation functions are commonly used within MLPs. If linear functions were used then an equivalent network made up of a single-layer of perceptrons, with appropriate weights, could be found [5] which defeats the purpose of using a layered network. A variety of nonlinear activation functions can be employed but commonly used functions are the sigmoid function or hyperbolic tangent function (Figure 4-5).

The activation function is the component of the node which determines the nonlinear nature of the network model. Layered networks with continuous nonlinear activation functions are able to form complex nonlinear decision boundaries for classification problems and smooth nonlinear curves in function approximation rather than solutions based on straight line segments.

Individual layers of neurons need not have the same activation functions. It is quite common for a net to employ sigmoidal neurons in the hidden layers together with linear output neurons. The hidden sigmoidal layers offer sufficient nonlinear functionality for most problems and with linear outputs the training is simplified.



$$f(s) = \frac{1}{1+e^{-s}}$$

$$f(s) = \tanh\left(\frac{s}{2}\right) = \frac{1-e^{-s}}{1+e^{-s}}$$

(a) **(b)**

Figure 4-5: (a) Sigmoid function, (b) Hyperbolic tangent function.

The nodes in the input layer are in fact not perceptrons at all. They represent the point at which the inputs enter the network and are distributed to each neuron in the first hidden layer. For the purpose of this thesis the convention will be used that the number of layers in a network will not include the input layer. A two-layer network therefore, consists of a layer of input ports, a hidden layer and an output layer and hence only contains *two* perceptron layers. The convention used for describing the topology of a net is as follows; a two-layer feedforward network with five inputs, eight hidden nodes and three output nodes would be referred to as a 5-8-3 network.

4.2 Feedforward network training methods

The main problem with MLPs was a lack of a good learning algorithm as the delta rule could not deal adequately with layered nodes. It wasn't until Rumelhart *et al.* [6] reported the development of the backpropagation algorithm in the mid 1980's that research into, and application of, neural networks advanced significantly. Since then many other methods for training feedforward neural networks have been developed. The backpropagation method, as one of the pioneering algorithms, will be described in the following section. An appreciation of this algorithm enables a better understanding of some of the other methods which will be briefly discussed in subsequent sections. These methods are *supervised learning* methods since the desired output results are provided for the network during the training process. In *unsupervised learning* the network moves to a solution without the assistance of target values. Only supervised

learning will be discussed in this thesis since it is most applicable to system identification problems.

4.2.1 The method of error backpropagation

The backpropagation algorithm has become the most common weight-update algorithm in use for feedforward neural network modelling. The algorithm is a generalisation of the Delta Rule and hence is sometimes referred to as the Generalised Delta Rule. Since the network we are dealing with is layered, the Delta Rule has to be generalised to enable the output error for any node in the network to be found.

The requirement for this algorithm is that the nonlinear activation function must be differentiable over its range. The most common nonlinear function used in backpropagation nets is the sigmoidal function, although any differentiable function can be used. The backpropagation algorithm enables the derivative of the error function to be calculated with respect to any weight within the net and updates the weight according to:

$$\Delta w_{ij} = -c\frac{\partial E}{\partial w_{ij}} \tag{4.11}$$

where w_{ij} is the weight connecting the i^{th} neuron to the j^{th} neuron and the error term, E, for a network with M output neurons is defined as either:

$$\textit{batch training:} \qquad E = \sum_p E^p = \frac{1}{2}\sum_p \sum_{i=1}^{M} \left(d_i^p - a_i^p\right)^2 \tag{4.12}$$

or

$$\textit{pattern training:} \qquad E = E^p = \frac{1}{2}\sum_{i=1}^{M} \left(d_i - a_i\right)^2 \tag{4.13}$$

In batch training the error summed over all the p input patterns is used to update the network weights. In contrast, pattern training updates the weights after each individual pattern has been presented. From this point on in the thesis all the p indices will be dropped to simplify the notation, however, the remainder of the discussion is equally applicable to both batch and pattern learning. Of the two approaches batch training is usually the better method to employ [7].

By extending the delta rule to nodes connected in an L-layer MLP the change of weight at iteration $k + 1$ becomes:

$$\Delta w_{ij}^l(k) = \mu\left(d_i^l - a_i^l\right)a_j^{l-1}\Big|_{w(k)} \tag{4.14}$$

where w_{ij}^l is the connection weight between node j (layer l–1) and node i (layer l),

a_i^l is the activation (output) of the i^{th} node in layer l,

d_i^l is the desired output from node i,

μ is the learning rate,

$l = 1,..., L$.

This can be written as $$\Delta w_{ij}^l(k) = \mu \delta_i^l a_j^{l-1}\Big|_{w(k)} \tag{4.15}$$

where $\delta_i^l = \left(d_i^l - a_i^l\right)$ using the Delta Rule.

However for a neuron in a hidden layer ($l < L$) the desired output (d_i^l) is not usually known. Therefore δ_i has to be calculated for hidden layer nodes according to the error gradient given in equation (4.11). δ_i is called the square-error derivative (or delta) associated with node i.

The full derivation of the backpropagation algorithm is given in Appendix 4. The main results for the algorithm are summarised below.

Equation (4.15) holds for any neuron in the network, such that:

for an output neuron ($l = L$): $$\delta_i^L = \left(d_i^L - a_i^L\right) f_L'\left(s_i\right) \tag{4.16}$$

and for a hidden-layer neuron ($l < L$): $$\delta_i^l = \sum_{h=1}^{m} \left(\delta_h^{l+1} w_{hi}\right) f_l'\left(s_i\right) \tag{4.17}$$

where δ_h^{l+1} is the delta associated with node h in the downstream layer (layer $l + 1$),

m is the number of neurons in layer $l + 1$,

$f_l'\left(s_i\right)$ is the derivative of the activation function of node i in layer l.

The process of calculating the square-error derivative for a given hidden node, i involves multiplying each δ associated with each node in the layer downstream ($l + 1$), by their respective weights on the connections from node, i. These weighted square-error derivatives are then summed together and multiplied by the derivative of node i's activation function at its current operating point.

The process that occurs during training is as follows;

On presentation of an input pattern predicted outputs are produced from each node in output layer. The errors between the model output and the targets are used to calculate the square-error derivatives for each node in the output layer and to update the output

layer weights. The deltas are propagated back to the previous layer to calculate the error derivatives for each node in that layer and to update their weights. The process continues until the deltas are fed back to the first layer and all the weights in the network have been updated. The input data is fed forward through the network and the error information is propagated back through the same network connections, hence the reason the backpropagation algorithm derives its name.

The sigmoidal and hyperbolic tangent functions have the advantage that their derivatives can be expressed in terms of the dependent variable. This allows for easy inclusion within the backpropagation equations.

Function		**Derivative**
Sigmoid function:	$f(x) = \dfrac{1}{1+e^{-x}}$	$f'(x) = f(x)[1-f(x)]$
Tanh function:	$g(x) = \dfrac{1-e^{-x}}{1+e^{-x}}$	$g'(x) = \frac{1}{2}[1-g(x)][1+g(x)]$

To summarise, if the sigmoidal nonlinearity is used in the neurons then equation (4.16) and (4.17) become:

$$\delta_i^L = \left(d_i^L - a_i^L\right)a_i^L\left(1-a_i^L\right) \quad \text{for output nodes} \tag{4.18}$$

$$\delta_i^l = \sum_h \left(\delta_h^{l+1} w_{hi}^{l+1}\right)a_i^l\left(1-a_i^l\right) \quad \text{for hidden nodes} \tag{4.19}$$

and are used in $$\Delta w_{ij}^l(k) = \mu\delta_i^l a_j^{l-1}\Big|_{w(k)} \tag{4.20}$$

If linear output nodes are used then δ_i in equation (4.18) is simply given by:

$$\delta_i^L = \left(d_i^L - a_i^L\right) \tag{4.21}$$

Note that equation (4.20) has the same form as the delta rule for a single perceptron in equation (4.10).

The update equation for the bias weights is given by

$$\Delta b_i^l = -\mu\frac{\partial E}{\partial b_i^l}, \tag{4.22}$$

and since an input to a bias is always unity it can be easily shown that

$$\Delta b_i^l = \mu\delta_i^l. \tag{4.23}$$

Extensions to the backpropagation algorithm

Momentum

A commonly used method to improve the speed of convergence of the backpropagation algorithm is to make the current weight update a function of past weight changes. This is achieved by using:

$$\Delta w_{ij}^{l}(k) = \alpha \Delta w_{ij}^{l}(k-1) + (1-\alpha)\mu \delta_i^l a_j^{l-1}\Big|_{w(k)} \tag{4.24}$$

where $0 < \alpha < 1$, is a constant.

Introducing past weight changes in this way produces an exponentially weighted average of past search directions over the weight space. This helps keep the weights moving across flat portions of the error surface, hence α is known as the momentum constant. Using momentum also effectively filters out high frequency variations of the error surface in the weights.

Adaptive learning rate

The learning rate, μ, is the step size for the gradient descent search. For practical purposes one would like μ to be as large as possible without causing oscillations in the search directions and possible instability. One method of achieving this is by varying the learning rate according to the error that is produced by the network. If the error is decreasing μ is increased to encourage learning in that direction, however, as soon as the error increases μ is decreased.

Performance of backpropagation training

Training a neural network can be viewed as a nonlinear optimisation problem which is formulated as follows:

Given: the set of input vectors $\boldsymbol{X}$ and a set of desired outputs $\boldsymbol{Y}$,

find: the set of network weights, $\boldsymbol{w}$,

to: minimise the error cost function, $E(\boldsymbol{w})$.

Standard backpropagation training is based on steepest descent optimisation where the optimum network weights are found by moving through the weight-space in a direction opposite to the maximum error gradient. Steepest descent methods are guaranteed to converge to a solution if the step size (μ) is constant and sufficiently small. However typical error surfaces for practical problems contain localised minima or long, gradual 'valleys' with small gradients and this causes slow convergence for steepest descent optimisation. Although the use of a momentum term and adaptive learning rate violates

the requirement of the proof for convergence, experience has shown that they often improve the performance of the backpropagation method.

More powerful optimisation algorithms are now commonly applied to neural network training and are out-performing backpropagation. These optimisation methods are not necessarily new but have been used in many other areas for a number of years; it is just that they have recently found application in the field of neural networks. These methods offer improved speed of convergence and improved handling of localised minima. The techniques include other gradient methods, quasi-Newtonian search methods and random searches.

4.2.2 Conjugate gradient method

The method of conjugate gradients [8, 9] is based on the concept of *conjugate directions*. If we wished to find the minimum point of an n-dimensional surface and we minimised along some direction $\boldsymbol{u}$, then the gradient of the function must be perpendicular to $\boldsymbol{u}$ at the minimum point on $\boldsymbol{u}$. If we now wanted to move in a new direction, $\boldsymbol{v}$, in order not to spoil our minimisation along direction $\boldsymbol{u}$ we must ensure that the new direction, $\boldsymbol{v}$, is also perpendicular to the gradient. If this is so then vectors $\boldsymbol{u}$ and $\mathbf{v}$ are said to be conjugate. A set of n possible conjugate vectors is said to be a conjugate set. If successive line minimisations of a function are performed along a conjugate set of directions, then those directions do not need to be re-traversed. For functions that are quadratic forms then n line minimisations along a set of linearly independent conjugate directions will locate exactly at the minimum point of the function. In practice the functions are not of a quadratic form and repeated cycles of line minimisations in conjugate directions will eventually reach the minimum point. After each n steps the search direction should be reset to be the gradient of the function.

The basic philosophy is to generate a conjugate search direction as a linear combination of the current steepest descent direction and the previous search direction. The following is an outline of the conjugate gradient method.

The aim is to minimise an n-dimensional function $E(\boldsymbol{w})$, where $\boldsymbol{w}$ is a n-element vector,

1) Starting at point $\mathbf{w}_0$, the initial search direction, $\mathbf{d}_0$, is in the direction of steepest descent: $\boldsymbol{d}_0 = -E'(\boldsymbol{w}_0) = \boldsymbol{r}_0$

2) A line search is performed in the search direction

i.e. find the step size, μ_i, which minimises: $E(\boldsymbol{w}_i + \mu_i \boldsymbol{d}_i)$

3) Update position: $\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \mu_i \boldsymbol{d}_i$

4) Find steepest descent direction: $\boldsymbol{r}_{i+1} = -E'(\boldsymbol{w}_{i+1})$

5) Calculate new conjugate search direction as a linear combination of the steepest descent direction and the previous search direction:

$$\boldsymbol{d}_{i+1} = \boldsymbol{r}_{i+1} + \beta_{i+1}\boldsymbol{d}_i$$

where: $\beta_{i+1} = \dfrac{\boldsymbol{r}_{i+1}^{\mathrm{T}}\boldsymbol{r}_{i+1}}{\boldsymbol{r}_i^{\mathrm{T}}\boldsymbol{r}_i}$ is commonly used.

After every n iterations the algorithm is reset by making $\boldsymbol{d}_i = -E'(\boldsymbol{w}_i)$

6) Return to step 2.

The β term used in step 5 is equivalent to the momentum term in the backpropagation method [10]. The Newton-Raphson or Secant methods are often used for the line search in step 2. These methods require calculation of the Hessian matrix, $E''(\boldsymbol{w}_i)$, which increases the computation requirements of the algorithm. Other methods have been devised which remove the need for such calculations by using approximation methods to estimate the correct step size [11].

The conjugate gradient method when applied to weight training in neural networks has been shown to give better performance than for the backpropagation method [10].

4.2.3 Levenberg-Marquardt method

The Levenberg-Marquardt optimisation method is an approximation of Newton's method and has become the standard for nonlinear least squares routines [8]. The algorithm was first put forward by Marquardt [12] related to an earlier piece of work by Levenberg [13]. The algorithm provides an elegant method of varying smoothly between the Gauss-Newton method (based on expanding the function in a Taylor series) and the steepest descent method. The latter method is used far from the minimum point and as the minimum is approached switches to the former.

If we have a function to minimise $E(\boldsymbol{w})$ with respect to $\boldsymbol{w}$ then using Newton's method the change in $\boldsymbol{w}$ at the i^{th} iteration would be:

$$\Delta \boldsymbol{w}_i = -\frac{E'(\boldsymbol{w}_i)}{E''(\boldsymbol{w}_i)} \tag{4.25}$$

where $E''(\boldsymbol{w}_i)$ is the Hessian matrix and $E'(\boldsymbol{w}_i)$ is the gradient. If $E(\boldsymbol{w})$ is a sum of squares function

$$E(\boldsymbol{w}) = \sum_{j=1}^{N} e_j^2(\boldsymbol{w}) \tag{4.26}$$

then it can be shown that

$$E'(\boldsymbol{w}) = \boldsymbol{J}^T(\boldsymbol{w})e(\boldsymbol{w}) \tag{4.27}$$

$$E''(\boldsymbol{w}) = \boldsymbol{J}^T(\boldsymbol{w})\boldsymbol{J}(\boldsymbol{w}) + \boldsymbol{S}(\boldsymbol{w}) \tag{4.28}$$

where $\boldsymbol{J}(\boldsymbol{w})$ is the Jacobian matrix

$$\boldsymbol{J}(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial e_1(\boldsymbol{w})}{\partial w_1} & \frac{\partial e_1(\boldsymbol{w})}{\partial w_2} & \cdots & \frac{\partial e_1(\boldsymbol{w})}{\partial w_n} \\ \frac{\partial e_2(\boldsymbol{w})}{\partial w_1} & \frac{\partial e_2(\boldsymbol{w})}{\partial w_2} & \cdots & \frac{\partial e_2(\boldsymbol{w})}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\boldsymbol{w})}{\partial w_1} & \frac{\partial e_N(\boldsymbol{w})}{\partial w_2} & \cdots & \frac{\partial e_N(\boldsymbol{w})}{\partial w_n} \end{bmatrix} \tag{4.29}$$

and

$$\boldsymbol{S}(\boldsymbol{w}) = \sum_{j=1}^{N} e_j(\boldsymbol{w})e_j''(\boldsymbol{w}). \tag{4.30}$$

For the Gauss-Newton method it is assumed that $\boldsymbol{S}(\boldsymbol{w}) \approx 0$, and equation (4.25) becomes

$$\Delta \boldsymbol{w}_i = -\frac{\boldsymbol{J}^T(\boldsymbol{w}_i)e(\boldsymbol{w}_i)}{\boldsymbol{J}^T(\boldsymbol{w}_i)\boldsymbol{J}(\boldsymbol{w}_i)}. \tag{4.31}$$

The Levenberg-Marquardt modification to the method is:

$$\Delta \boldsymbol{w}_i = -\frac{\boldsymbol{J}^T(\boldsymbol{w}_i)e(\boldsymbol{w}_i)}{[\boldsymbol{J}^T(\boldsymbol{w}_i)\boldsymbol{J}(\boldsymbol{w}_i) + \lambda \boldsymbol{I}]} \tag{4.32}$$

where λ is a scalar.

If λ is very large then the denominator in equation (4.32) becomes diagonally dominant and the method approximates gradient descent optimisation (with step size of $1/\lambda$). On the other hand if λ approaches zero the method approximates the Gauss-Newton method. The Gauss-Newton method is faster and more accurate near the minimum, so the aim is to shift towards the Gauss-Newton method as quickly as possible. Thus, λ is adaptive and is decreased after each step that improves the solution and is increased only when a step moves away from the minimum. Commonly λ is set to a small value, say 0.001, and is adapted by a factor of 10.

The algorithm is assumed to have converged when the norm of the gradient (equation (4.27)) is less than some predetermined value, or when the sum of squares has been reduced to some error goal.

In practice the Levenberg-Marquardt method works very well, giving fast convergence times and is less likely to get trapped in local minima. Hagan & Menhaj [14] demonstrate the superior performance of this method over steepest descent and conjugate gradient learning in neural networks. However, its main disadvantage is that significantly more processing memory is required for the algorithm compared with other methods. This is because the Jacobian matrix must be calculated and stored for each iteration of the algorithm.

The Levenberg-Marquardt optimisation can easily be incorporated into the backpropagation method replacing steepest descent minimisation [14].

Firstly the vector $\boldsymbol{w}$, is made to contain all the network weights and biases and is of the following form:

$$\boldsymbol{w} = [w_{11}^1, w_{12}^1, \ldots, w_{S_1 R}^1, b_1^1, \ldots, b_{S_1}^1, w_{11}^2, \ldots, b_{S_L}^L]^T$$

where S_i is the number of neurons in the i^{th} layer,

R is the number of inputs to the network, and

L is the number of network layers.

$E(\boldsymbol{w})$ is the error sum of squares term where $\boldsymbol{e}(\boldsymbol{w}) = \boldsymbol{d} - \boldsymbol{a}(\boldsymbol{w})$, is the output prediction error of the network ($\boldsymbol{d}$ = target output vector and $\boldsymbol{a}(\boldsymbol{w})$ = network output vector).

The modification to the standard backpropagation algorithm lies in the calculation of the delta from the output layer of the network. Instead of equation (4.16) the following equation is used:

$$\delta_i^L = f_L'(s_i) \tag{4.33}$$

Each column of the δ_i^L matrix is a sensitivity vector which must be backpropagated through the network using equation (4.17); the following relationships from the backpropagation method;

$$\frac{\partial E}{\partial w_{ij}^l} = \delta_i^l a_j^{l-1} \tag{4.34}$$

$$\frac{\partial E}{\partial b_i^l} = \delta_i^l \tag{4.35}$$

and equation (4.29) to produce one row of the Jacobian matrix , $\boldsymbol{J}(\boldsymbol{w})$.

4.2.4 Random search methods

Random search methods offer a global search of the weight space for neural network training. Such methods are easy to implement and attempt to avoid problems with localised minima.

A random optimisation method devised by Solis & Wets [15] has been shown to successfully train feedforward neural networks and perform better than the backpropagation method [16]. The algorithm used is similar to the *Chemotaxis* algorithm [17, 18] which is so named because of parallels drawn with the bacteria 'chemotaxis'. This method has also shown encouraging results.

Both these methods postulate that weight adjustments occur in a random manner and that weight changes follow a zero mean Gaussian distribution. The algorithm adjusts weights by adding Gaussian distributed random values to the old weights. The new weights are accepted if there is an improvement in the resulting output error. If no error improvement is made a new set of random values is generated and the process is repeated. The variance of the random increments can be adjusted during the training in order to assist the network convergence.

4.2.5 Other types of feedforward networks

Radial Basis Function networks

Radial Basis Function (RBF) networks [19] are feedforward networks with the Gaussian function, given by equation (4.36), used as the neuron activation functions.

$$g(x) = \mathrm{e}^{\frac{-(x-c)^2}{\sigma^2}} \tag{4.36}$$

where: x is the input to the neuron, c is the function centre and σ is the function width.

The way in which a RBF network discriminates is quite different to a MLP as the Gaussian neurons do not perform a weighted sum of the inputs but partition the input-space into regions with each neuron defining a region or cluster of the input space. Whether a certain input vector belongs to a particular cluster is dependent on its euclidean distance from the cluster centre, c, and the cluster width, σ. The cluster centres and widths can be considered adjustable parameters and the network can be trained using any of the previous feedforward methods.

If the number of hidden nodes and their centres are predefined and linear output neurons are used then training simply involves determining the weights in the linear output layer

using a least squares method which gives a fast one-pass training method. Alternatively, and more commonly, a clustering algorithm is used to position the cluster centres in the input space and determines the width of each. In this case additional neurons are added during training in order to fully define the input-space. The number of hidden neurons then is not known prior to training and is dependent on the quality of the training data. This method involves more computation and may generate large networks.

Cascade-correlation networks

The cascade-correlation [20, 21] method is similar to backpropagation but adds hidden neurons during training. Training starts with only the input and output nodes. If this network cannot become accurate enough, then a set of candidate hidden nodes are added which are not connected to the output layer. These candidate nodes are trained using a type of gradient descent to maximise their correlation between each candidate and the output error. When the correlation stops improving the network selects the best hidden node, freezes its weights and connects it to the output layer. This hidden node becomes a permanent part of the network. This constructive process continues until the network becomes sufficiently accurate. Each new hidden node receives the network inputs plus the outputs of all the previous hidden nodes thereby creating a cascading structure rather than a single hidden layer. Cascade-correlation networks determine their own structure and size which eliminates the need determine the optimum number of hidden nodes.

4.3 Neural network practical design issues

4.3.1 Setting of initial weights

As with any optimisation problem the initial values of the function parameters can have a large effect on the solution obtained by restricting the search within a certain region of the solution space. In order to ensure no bias in the search the initial network weights are usually set to small random values between –1 and 1. The training is then carried out a number times using different sets of initial weights to ensure a good solution is achieved.

Research has been carried out to determine other methods of setting initial weights for certain networks. Nguyen and Widrow [22] showed that by using random weights with a certain method of normalisation, a 2-layered sigmoidal network will be better able to form a function approximation of an arbitrary function and shorten the training time.

4.3.2 Network topology - numbers of layers and nodes

The number of system inputs and outputs determines the number of nodes in the input and output layers respectively. For MLPs the number of hidden layers and the number of nodes within them is determined by the network designer and is dependent on the complexity of the relationships one is wanting to model. Increasing the number of layers in the network increases the complexity of the classifying regions or the complexity of nonlinear mappings in solution space. It has been shown that an MLP consisting of a single hidden layer of continuous smooth monotonically increasing neurons is able to approximate any continuous function arbitrarily well [23, 24]. However for some problems a small three-layer net is able to achieve the same results as a two-layer net with a large number of hidden nodes [5].

Another issue is that the number of nodes required in the hidden layers of the network is not obvious. Too many nodes result in long training times and memorising or over-fitting of the specific input patterns presented to the network, whereas too few nodes results in poor characterisation of the trends within the data. The best arrangement of nodes to use has to be determined by trial and error. In practice one should start with a small number of hidden nodes and increase them until there is no obvious improvement in training performance.

4.3.3 Training data

To generate accurate neural network models of a system one must have sufficient reliable data to use in the training process. Since neural network models are generated from process data, the quality of a model will be limited by the quantity and quality of the data it is provided with. Neural networks are able to interpolate satisfactorily but are poor at extrapolating beyond what they have been trained on. Therefore, in gathering training data, one must ensure that there is sufficient data which covers the modes of the system being modelled.

In order to give equal weighting to each input and target variable during the training of the network, the training data is standardised. This is achieved by scaling the data for each variable between two limits, usually between 0 and 1 or –1 and 1. This ensures that there is no bias towards those variables whose values have larger magnitudes. In practice the bounds are set at 0.1 and 0.9 or –0.9 and 0.9 respectively, to prevent the saturation of the neurons. An alternative to scaling is normalisation of the input patterns by subtracting the mean and dividing by the standard deviation of each variable.

4.3.4 Weight Scaling

When implementing a network trained on scaled data the new inputs have to be scaled in a similar manner to the training data in order for the network to calculate properly. The resulting outputs also have to be unscaled in order to obtain the real output values. Alternatively, the network weights can be scaled in order to remove the need for scaling and unscaling during implementation. This approach makes the implementation task a lot simpler as the method of scaling the data does not need to be remembered each time the neural network is used; instead, the scaling information is incorporated in the network weights.

If the inputs to a neural network were scaled between two values ϕ_{lo} and ϕ_{hi}, where $\phi_{lo} < \phi_{hi}$, then the scaled value of the j^{th} input is given by:

$$x_j = (\phi_{hi} - \phi_{lo})\frac{X_j - \min(X_j)}{\max(X_j) - \min(X_j)} + \phi_{lo} \tag{4.37}$$

where: X_j is the unscaled value.

It is easy to show that the first layer of network weights can be scaled using the following general equations:

$$w'_{ij} = \frac{(\phi_{hi} - \phi_{lo})w_{ij}}{\max(X_j) - \min(X_j)} \tag{4.38}$$

$$b'_i = b_i - \sum_j \left(\frac{(\phi_{hi} - \phi_{lo})w_{ij}\min(X_j)}{\max(X_j) - \min(X_j)} - \phi_{lo}w_{ij} \right) \tag{4.39}$$

where: w'_{ij} and b'_i are the weights and biases that are used for simulating the network with unscaled inputs.

Similarly the target outputs can also be scaled using :

$$d_i = (\phi_{hi} - \phi_{lo})\frac{D_i - \min(D_i)}{\max(D_i) - \min(D_i)} + \phi_{lo} \tag{4.40}$$

where D_i is the unscaled target for the i^{th} output neuron.

If the output neurons are linear then the output layer weights can be scaled and the network used to predict unscaled outputs.

In general for the output layer weights it can be shown that:

$$w'_{ij} = \frac{w_{ij}(\max(D_i) - \min(D_i))}{\phi_{hi} - \phi_{lo}} \tag{4.41}$$

$$b_i' = \frac{(b_i - \phi_{lo})(\max(D_i) - \min(D_i))}{\phi_{hi} - \phi_{lo}} + \min(D_i) \tag{4.42}$$

where: w_{ij}' and b_i' are the weights and biases that are used for simulating the network to produce unscaled outputs.

Only the input and output layer weights need to be scaled in this way; any additional hidden layer weights do not need to be altered. Similar equations can be easily developed for networks where the training data has been normalised using the mean and standard deviations of the data.

If nonlinear output neurons are used then the targets are no longer linear functions of the output weights and the scaling of the output layer weights can prove difficult.

Once a network has been satisfactorily trained then the weight scaling equations are applied to the network and from then on the network can be used with data which has not been scaled. If further training is necessary after the weights have been scaled then the inverses of the above equations need to be applied.

4.3.5 Performance measures

There are a variety of measures by which one can analyse the performance of a trained network, they are usually functions of the output prediction error.

Possible measures for use during training and validation are the sum-of-squared error (SSE), mean-squared-error (MSE) or root-mean-squared (RMS) error terms. These are usually standardised by calculation of the scaled output values so that one can compare between various output variables with different magnitudes.

Other measures which can be applied when comparing the performance of networks with differing structures are Akaike's error criteria [25]. These consider the size of the network as well as the output error and are useful when selecting a model structure. The two Akaike measures are as follows:

$$\text{Final Prediction Error (FPE)} = \frac{E}{2N} \times \frac{(N + N_w)}{(N - N_w)} \tag{4.43}$$

$$\text{Information Theoretic Criterion (AIC)} = \ln\left(\frac{E}{2N}\right) + 2\frac{N_w}{N} \tag{4.44}$$

where E is the sum of square error, N is the number of data points and N_w is the number of adjustable parameters (weights) in the network.

Sometimes the prediction error may not give a good indication of the performance of a network since the network output may recreate the correct shape but have an offset or

exhibit a time-shift. In these cases spectral analysis or correlation analysis could be used as an indication of the network response.

For this study the MSE and FPE measures were used to analyse network performance.

4.3.6 Training, testing and validation

Three sets of data should be used during the development of a neural network model. These are a training set, testing set and a validation set. All the data should be characteristic of the system in question. The training set is used to present to the network during training and should therefore be of the required quantity and quality. The other two sets are independent of the training data and used to test the performance of the neural network. The ability of a neural network to correctly model data which it has previously not been trained on is referred to as the ability to *generalise.*

It is recommended practice that during training the network be periodically tested with the independent testing data in order to track the generalisation capabilities of the network. This enables the developer to observe the point where the network begins to model information which is specific to the training data and the generalisation performance degrades. A typical plot of training and testing error is shown in Figure 4-6. In this example the most appropriate set of network weights would be those at iteration k since it is at the minimum point of the testing error curve. After this point the testing error rises as the network overfits the training data.

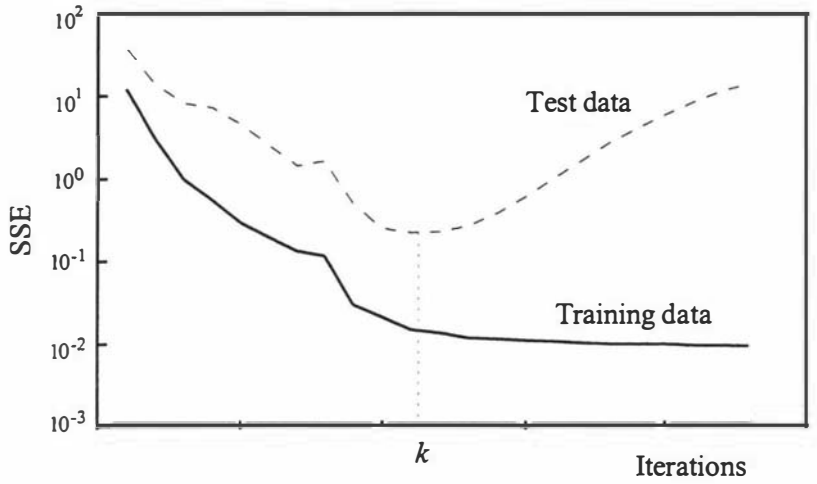


Figure 4-6: Training and testing errors during network training.

Once the best network weights have been selected it is necessary to validate the neural network on a further set (or sets) of data which has not been used in any way during the development process. This is an unbiased test of whether the network has captured the characteristics inherent in the system.

More formal validation methods can be used and involve using correlation tests. According to Billings & Voon [26] the validity of linear models can be determined using two simple covariance tests which detect unmodelled linear terms. If the model is adequate then the prediction errors or residuals (ε) will be uncorrelated to the model inputs (u) and the autocorrelation of the residuals is an impulse function. Therefore for a single input-single output model:

$$\left.\begin{aligned} R_{\varepsilon\varepsilon}(k) &= \text{an impulse function} \\ R_{u\varepsilon}(k) &= 0 \quad \text{for all } k \end{aligned}\right\} \tag{4.45}$$

where: $R_{xy}(k)$ indicates the cross-correlation function between $x(t)$ and $y(t)$.

For nonlinear models these tests are not sufficient and three further cross-correlations must be used to determine the model adequacy [26, 27]. These are:

$$\left.\begin{aligned} R_{\varepsilon(\varepsilon u)}(k) &= 0 \quad k \geq 0 \\ R_{u^{2'}\varepsilon}(k) &= 0 \quad \text{for all } k \\ R_{u^{2'}\varepsilon^2}(k) &= 0 \quad \text{for all } k \end{aligned}\right\} \tag{4.46}$$

where $\varepsilon u(t) = \varepsilon(t+1)u(t+1)$,

$u^{2'}(t) = u^2(t) - \bar{u}^2(t)$, and

$\bar{u}^2(t)$ represents the time average or mean value of $u^2(t)$.

A practical model, which will always contain some bias can be regarded as acceptable if the above five correlation tests are contained within a 95% confidence interval of $\pm 1.96/\sqrt{N}$, where N is the number of data points. Both Chen *et al* [27] and Doherty *et al* [28] have demonstrated the application of these correlation tests to validate neural network models.

4.4 Neural networks for system identification

4.4.1 Introduction

Neural networks have become useful tools for system identification problems. Their application to complex nonlinear systems, in particular, has proven popular for the following reasons:

- Neural networks offer nonlinear characteristics,
- The time and effort required for neural network development is decreased in comparison with conventional analytical modelling techniques,

- Neural networks are constructed using actual data collected from the system and so offer a solution where the theoretical knowledge of a system is lacking.

As universal approximators [23, 24], neural networks have found wide application in industrial process modelling. Much has been reported on the subject of applying neural networks to process modelling. Particular chemical engineering problems that neural networks have been applied to include fault diagnosis [29], dynamic modelling [30, 31, 32], on-line process estimation, [19] and system identification and control [33, 34, 35] (and also see §4.5).

There are many aspects of system identification using neural networks for which different methods and different types of networks need to be employed and as with any field there is more than one way of achieving a certain task. With system identification the question of characterising the process dynamics is particularly important for neural networks specially those that are used as part of a control strategy.

Other issues have also received attention by researchers such as; making use of prior knowledge of the system, making the neural networks more physically meaningful with respect to the actual system and simplifying the training task. These issues are raised in Chapter 6 in the discussion of the development of the neural network model for the falling-film evaporator.

4.4.2 Dynamic modelling

Neural networks produce a static nonlinear mapping between inputs and outputs, however, system identification problems require the dynamics of the system to be captured. The dynamics of the system must therefore be incorporated into the training of the neural network. There are a number of ways this can be achieved including the use of time delays on network inputs, using output or state feedback, or using neurons with dynamic processing capabilities.

Time delay neural networks

The simplest and most common method of producing dynamic neural network models is by using past information from the system as inputs to the network.

Past system inputs can be used to train the network giving a finite impulse response (FIR) representation of the system (equation (4.47)).

$$\text{FIR model:} \qquad \hat{y}(k+1) = \Gamma\left[u(k), u(k-1), \ldots, u(k-K_u+1)\right] \tag{4.47}$$

where: K_u is the number of system inputs fed to the neural network and $\Gamma[.]$ is the nonlinear function equivalent of the neural network.

Neural networks with output feedback

More commonly time-delay networks are developed with the system outputs fed back as inputs to the network. This gives a model with a representation of a NARX (nonlinear autoregressive with exogenous inputs) model. Figure 4-7 illustrates a block diagram for a method of identifying a dynamic system which produces a neural network, discrete-time model with output feedback. This method is known as a *series-parallel* modelling approach [36] and a feedforward training algorithm is used. The neural network can be trained on-line as the system is operated or alternatively a historical database of system data can be used for the training.

A *parallel* training approach [36] could also be used whereby the model outputs are fed back to the input layer of the network rather than the actual system outputs. This is illustrated in Figure 4-7 with the switch positioned at point B. Feedforward training can be used for this method of identification, however, in order to properly account for the external feedback connection, dynamic recurrent training methods are used. These are discussed in more detail in Section 5.5.2.

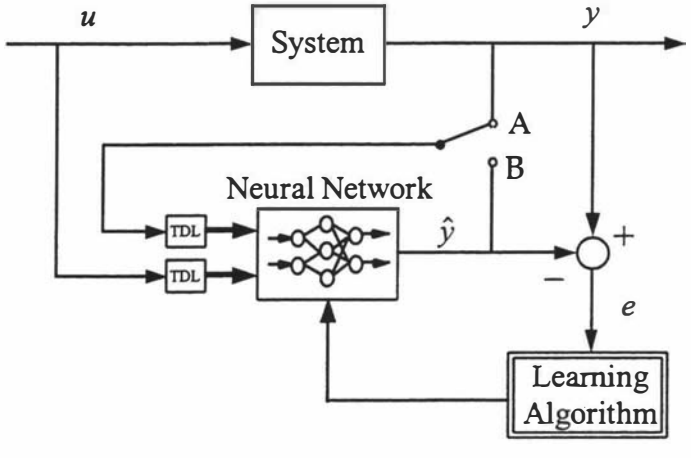


Figure 4-7: Identifying a dynamic neural network model.

With the switch at 'A' = series-parallel model, with switch at 'B' = parallel model with output feedback

These discrete-time NARX models can be described by the following equations:

Series-parallel model:

$$\hat{y}(k+1) = \Gamma\left[u(k),u(k-1),\ldots,u(k-K_u+1);y(k),y(k-1),\ldots,y(k-K_y+1)\right] \quad (4.48)$$

Parallel model:

$$\hat{y}(k+1) = \Gamma\left[u(k),u(k-1),\ldots,u(k-K_u+1);\hat{y}(k),\hat{y}(k-1),\ldots,\hat{y}(k-K_y+1)\right] \quad (4.49)$$

where K_u and K_y are the number of system inputs and outputs fed to the neural network respectively and $\Gamma[.]$ is the nonlinear function equivalent of the neural network.

The drawback with using time-delay networks with output feedback is that the dimensions of the network can become large, especially for systems with a large number of variables.

Neural networks with state feedback

Networks which incorporate state feedback are typically single layer networks with feedback connections between each of the neurons. In the general case all the neurons are completely interconnected, even with themselves. Some or all of the neurons may receive an external input and any or all of the neurons may be viewed as outputs. This class of networks are known as *recurrent* networks due to the feedback connections between the neurons. The recurrent connections store the state information of the system. Fully recurrent networks (Figure 4-8) are the most general class of networks of which many other network structures are simplifications.

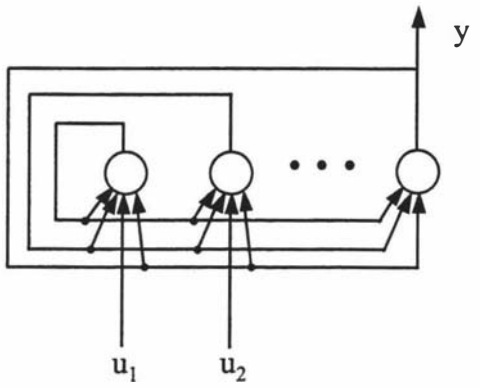
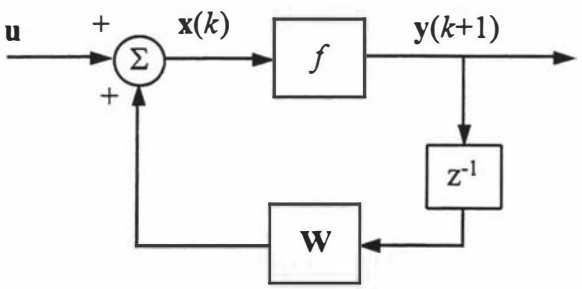


Figure 4-8: Fully recurrent neural network

The Hopfield network [37] is the best known recurrent network. It is a fully recurrent, single layer network. The Hopfield network can be viewed as a representation of a nonlinear dynamical system. A discrete version of a Hopfield net with input vector, **u**, state vector, $\mathbf{x}(k)$, and output vector, $\mathbf{y}(k)$, is shown in Figure 4-9. **W** is the weight matrix and $f(.)$ is the nonlinear activation function.



$$\mathbf{x}(k) = \mathbf{W}\mathbf{y}(k) + \mathbf{u}$$

$$\mathbf{y}(k+1) = f(\mathbf{x}(k))$$

Figure 4-9: Discrete-time Hopfield network

The internal structure of recurrent networks capture the system dynamics thereby requiring smaller networks than those used in feedforward models. However the training times are longer for recurrent networks. A comparison of modelling a chemical

process with recurrent and feedforward networks is given in [31] which highlights these points.

Networks with dynamic neurons

A method of producing dynamic neural networks by incorporating dynamic behaviour within the neuron itself is an alternative to developing a specialised dynamic network structure. The simplest approach is to incorporate a first-order transfer function within each neuron [17, 38]. Networks using this *filter-based* approach (Figure 4-10) have the output, a, of the neurons transformed in the following manner:

$$a(t+1) = \alpha a(t) + (1-\alpha)a(t) \qquad 0 \le \alpha \le 1 \qquad (4.50)$$

The α terms which relate to the time constant of the first-order filter are not likely to be known and so the problem becomes one of adapting these filter coefficients during the training process. In effect they become extra parameters in the optimisation problem. There may still be a requirement for past information to be used as inputs to filter-based networks if pure time delays occur within the system being modelled.

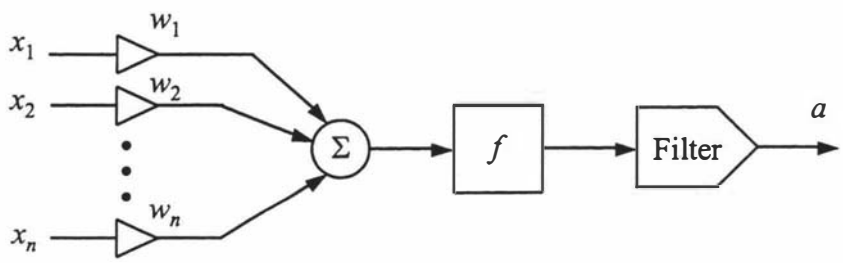


Figure 4-10: Dynamic neuron processing

4.5 Neural networks in process control

Since neural networks have been found to offer promising solutions to process modelling tasks it is only natural that these models be used in process control. Many of the control strategies that are in place in industrial plants are single loop linear controllers. Nonlinear control methods have been very slow to make an impact on actual processes. But with the nonlinear capabilities of neural networks, and their reliance on process data rather than extensive process knowledge, nonlinear multivariable control is now becoming increasingly realisable.

Process control has become one of the major research and application areas of neural network techniques and much has been published on the subject [39, 33, 34]. This section seeks to give a brief overview of the main types of control to which neural

networks have been applied. These strategies include feedback control applications, model-based control and adaptive control.

4.5.1 Feedback control applications

Neural networks and PID control

The majority of the controllers used in industrial applications are of the PID type. Since these controllers are so prevalent, and need to be correctly tuned, the application of neural networks to enhance the existing control loops is a possibility. Two ways in which neural networks could be used with PID controllers is as auto-tuning mechanisms or within a gain scheduling system.

An auto-tuning scheme would use the neural network to provide nonlinear models for the controller tuning. This would be better than the traditional approach of using linear models [33]. This could be extended to include a neural network model in parallel with the actual process which supervises the selection of the controller gains.

A gain scheduling method where different controller parameters are used as the process moves into different operating regions could also employ neural networks. Here a neural network could be trained on-line with the controller and process or, alternatively, with an already identified neural network model of the process. The output error would be used to update the weights in the network in order to improve its outputs, which are the controller gains. This approach is equivalent to that of a identifying a serial semi-parametric model (discussed in Section 6.2.2) where the controller is the parametric model. The network would be able to adjust the controller gains based on the current process operating position.

Inferential control

Often process measurements of primary importance can only be achieved by the use of an on-line analyser or through off-line analysis, both of which involve a time delay. Inferential estimation schemes can be used to assist process operators by providing real-time estimates from measurements of related variables to infer the value of the primary variable. Neural networks offer an ideal tool for this type of application. The possibility of using the inferred estimates within a feedback control strategy could then be easily achieved. The use of a suitable secondary variable as input to the neural network could provide a method of feedforward regulation since disturbance effects often manifest themselves first in the response of other variables.

The application of neural networks to inferential estimation of a fermentation process and an inferential PI controller for a distillation column is reported in [17].

4.5.2 Model-based control

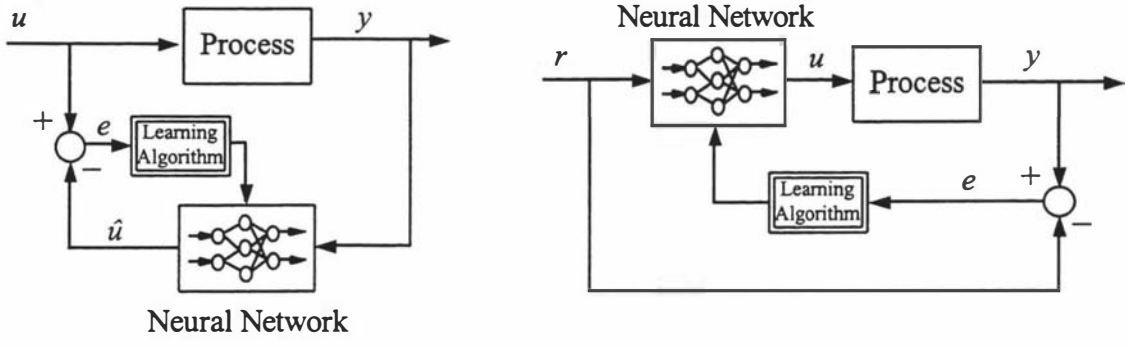
Model-based control systems are becoming increasingly popular in research and industry as computing power and modelling techniques advance. Model-based controllers incorporate an explicit dynamic model of the process into their structure. Most model-based control systems which are implemented employ linear process models. Neural networks applied to model-based control give rise to nonlinear control structures. One problem is that the nonlinear and non-transparent characteristics of neural networks make it difficult to analyse their robustness or stability and hence the robustness of the controllers they are used in. However, despite this yet-to-be-solved theoretical issue, neural networks are being widely applied to model-based controllers.

The major model-based control structures to which neural networks have been applied are Inverse Control, Internal Model Control, Model Predictive Control and Model Reference Control.

Inverse control

Inverse model control makes use of an inverse process model to act as the controller. The model receives the process setpoint and outputs the required control input to the process. An inverse neural network model can be trained to mimic the inverse dynamics of the system. In *direct inverse modelling* [33, 40] the neural network is fed the actual process output in order to predict the given system input (Figure 4-11(a)). The error between the network output and the input to the system is used to train the network to represent the inverse plant.

There are however a number of drawbacks to this approach relating to the operational range of the training and the possibility that an incorrect or unstable inverse might be modelled. An alternative approach which aims to overcome these problems is known as



(a) Direct method **(b) Specialised method**

Figure 4-11: Structures for inverse modelling.

specialised inverse modelling [33, 40] which uses the setpoint as an input to the network in order to model the control input. The error signal to train the network is obtained by comparing the setpoint with the actual output from the process (Figure 4-11(b)). A forward neural network model of the plant could be used instead of the actual plant during training in order to remove the need for the real system in the training process [41].

Inverse control is applied by simply cascading the inverse system model with the process as in Figure 4-12. Clearly this approach relies heavily on the accuracy of the inverse model. This can be overcome to some degree by using on-line learning where the parameters of the model are adjusted on-line.

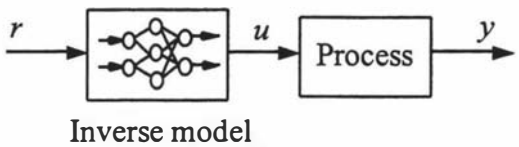


Figure 4-12: Direct inverse control

Model Reference Control

In model reference control the desired performance of the controlled system is specified through a stable reference model. The control system attempts to make the process output match that of the reference model. If a neural network is used as the controller, the error between the reference model output and the process output is used to train the neural network [36]. This approach is closely related to the training of inverse models described above. The training will force the network controller to be a de-tuned inverse model controller resulting from the dynamics of the reference model.

Internal Model Control

In Internal Model Control (IMC) both forward and inverse system models are used as elements within the feedback loop. The roles of the inverse and forward models and the appealing properties of the approach have been discussed at length by Garcia and Morari [42]. The models used can be either linear or nonlinear. If using neural networks, the inverse model can be identified using the approaches outlined above. Implementation of neural networks into the IMC structure is straightforward. The forward model is located parallel with the process and the inverse models in series (Figure 4-13). The difference between the actual plant output and the forward model output is used for feedback control. A linear filter (F) can be used before the inverse model in order to provide adequate robustness and tracking response to the closed-loop system.

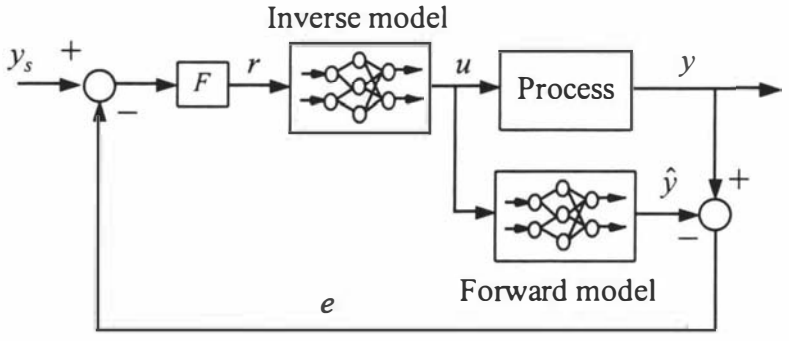


Figure 4-13: Structure for internal model control.

The successful use of neural networks in IMC, as applied to nonlinear chemical process simulations is reported in [43, 44, 45, 46]. Hunt & Sbarbaro [45] use RBF networks and investigate issues relating to the model invertability. Nahas *et al* [46] demonstrate favourable comparisons when compared with PID control.

Model Predictive Control

Model Predictive Control (MPC) systems employ a dynamic model to generate process predictions over a future time horizon. The control inputs are optimised using the process model in order achieve a desired output trajectory. MPC and the implementation of neural networks in MPC are discussed in detail in Chapter 5.

4.5.3 Adaptive control

Adaptive control involves the on-line adjustment of the controller parameters in order to improve the control of a system as the operating conditions and environment changes. Examples of adaptive controllers include model reference adaptive systems, self-tuning regulators and gain scheduling controllers [47].

Since neural networks are themselves easily adaptable they are ideal for implementing within adaptive control schemes. The algorithms used to update the parameters in conventional adaptive control systems are similar to the learning algorithms used to train neural networks. The control system can therefore be adapted by using on-line training of the neural network to account for changes occurring in the process.

Adaptive versions of model-based control systems are an active area of research. Model Reference Adaptive Control (MRAC) is discussed and demonstrated in [36] and [48] where the inverse network controller is continually adapted by the output error signal. The MRAC scheme can either involve direct or indirect control [36]. With direct neural network-based MRAC the neural network controller is adjusted to minimise the error between the plant and the reference model output. In the indirect scheme the neural network controller is designed based on a second neural network that has been trained to

approximate the plant. Both neural networks are adjusted on-line to minimise the error between the plant and the reference model output.

Neural networks in adaptive MPC and IMC require on-line, real-time dynamic training methods such as real-time recurrent learning [49] or the use of an on-line history of past process data to feed to the network [50]. The neural networks are usually trained using historical data and standard training techniques before implementation into an adaptive control scheme. For adaptive IMC both the forward and inverse models can be adaptive [51] and trained on different error signals (Figure 4-14). On-line training methods are computationally expensive tasks to perform and therefore are often only applicable to slow time-varying processes.

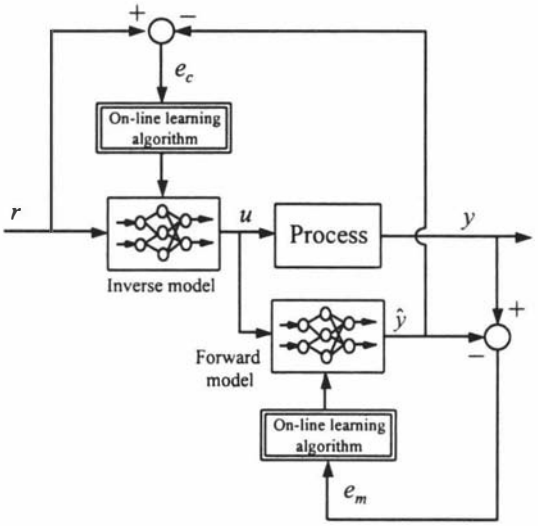


Figure 4-14: An adaptive IMC strategy

The advantages of using adaptive controllers is that changes in the process characteristics and operating environment can be accounted for without having to manually update the controller. However the difficulty in determining the stability of nonlinear adaptive controllers does pose a problem. Lyapunov methods and hyperstability theory have been applied to prove the stability of adaptive systems where the underlying plant is assumed linear and time-invariant. For adaptive systems consisting of nonlinear plants and neural networks new concepts and theory are still being explored. Adaptive controllers will become more common as computational capabilities are advanced and nonlinear adaptive stability theory is developed.

4.6 References

[1] **Hammerstrom, D.**, Neural networks at work. *IEEE Spectrum*, June 1993, vol. 30, pp. 26-32.

[2] **Rosenblatt, F.**, The perceptron: A probabilistic model for information storage and organisation in the brain. *Psychological Review*, 1958, vol. 65, pp. 386-408.

[3] **Widrow, B.& Hoff, M.E.**, Adaptive switching circuits. *1960 IRE Western Electric Show and Convention Record*, August 1960, part 4, pp. 96-104.

[4] **Widrow, B & Lehr, M.A.**, 30 years of adaptive neural networks: perceptron, madaline and backpropagation. *Proceedings of the IEEE*, 1990, vol. 78, pp. 1415-1442.

[5] **Hush, D.R. & Horne, B.G.**, Progress in supervised neural networks. *IEEE Signal Processing Magazine*, January 1993, vol. 10, pp. 8-39.

[6] **Rumelhart, D.E., Hinton, G.E. & Williams, R.J.**, Learning internal representations by error propagation. In: *Parallel Distributed Processing*, vol. 1, Rumelhart, D.E. & McClelland, J.L. (Eds.), MIT Press, Cambridge, MA, USA, 1986, pp. 318-362.

[7] **Qin, S.-Z., Su, H.-T. & McAvoy, T.J.**, Comparison of four neural net learning methods for dynamic system identification. *IEEE Transactions on Neural Networks*, 1992, vol. 3, pp. 122-130.

[8] **Press, W. H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P.**, *Numerical recipes in FORTRAN,* 2nd Edition, Cambridge University Press, Cambridge, UK, 1992.

[9] **Shewchuk, J.R.**, *An introduction to the conjugate gradient method without the agonizing pain*, Edition 1¼, Carnegie Mellon University, Pittsburgh, PA, USA, 1994.

[10] **Leonard, J. & Kramer, M.A.**, Improvement of the backpropagation algorithm for training neural networks. *Computers and Chemical Engineering*, 1990, vol. 14, pp. 337-341.

[11] **Møller, M.F.**, A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 1993, vol. 6, pp. 525-533.

[12] **Marquardt, D.W.**, An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 1963, vol. 11, pp. 431-441.

[13] **Levenberg, K.**, A method for the solution of certain nonlinear problems in least squares. *Quarterly of Applied Mathematics*, 1944, vol. 2, pp. 164-168.

[14] **Hagan, M.T. & Menhaj, M.B.**, Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 1994, vol. 5, pp. 989-993.

[15] **Solis, F.J. & Wets, J.B.**, Minimization by random search techniques. *Mathematics of Operations Research*, 1981, vol. 6, pp. 19-30.

[16] **Baba, N.**, A new approach for finding the global minimum of error function of neural networks. *Neural Networks*, 1989, vol. 2, pp. 367-373.

[17] **Willis, M.J., Di Massimo, C., Montague, G.A., Tham, M.T. & Morris, A.J.** Artificial neural networks in process engineering. *IEE Proceedings - D*, 1991, vol. 138, pp. 256-266.

[18] **Montague, G.A., Morris, A.J. & Willis, M.J.**, Artificial neural networks: methodologies and applications in process control. In: *Neural Networks for Control and Systems*, Warwick, K., Irwin, G.W. & Hunt, K.J. (Eds.), Peter Peregrinus, Stevenage, UK, 1992, pp. 123-150.

[19] **Chen, S., Cowan, C.F.N. & Grant, P.M.**, Orthogonal least squares learning algorithm for Radial Basis Function networks. *IEEE Transactions on Neural Networks*, 1991, vol. 2, pp. 302-309.

[20] **Fahlman, S. & Lebiere, C.**, The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, PA, 1991.

[21] **Hammerstrom, D.**, Working with neural networks. *IEEE Spectrum*, July 1993, vol. 30, pp. 46-53.

[22] **Nguyen, D & Widrow, B.**, Improving the learning speed of 2-layered neural networks by choosing initial values of the of the adaptive weights. *Proceedings of the International Joint Conference of Neural Networks*, July 1990, San Diego, CA., vol. 3, pp. 21-26.

[23] **Hornik, K., Stinchcombe, M. & White, H.**, Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989, vol. 2, pp. 359-366.

[24] **Leshno, M., Lin, V.Ya., Pinkus, A. & Schocken, S.**, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 1993, vol. 6, pp. 861-867.

[25] **Akaike, H.**, A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 1974, vol. AC-19, pp. 716-723.

[26] **Billings, S.A. & Voon, W.S.F.**, Correlation based model validity tests for nonlinear models. *International Journal of Control*, 1986, vol. 44, pp. 235-244.

[27] **Chen, S., Billings, S.A. & Grant, P.M.**, Nonlinear system identification using neural networks. . *International Journal of Control*, 1990, vol. 51, pp. 1191-1214.

[28] **Doherty, S.K., Williams, D. & Gomm, J.B.**, Modelling a highly nonlinear process using an artificial neural network. *Proceedings of the Irish DSP and Control Colloquium*, July 1994, Dublin, Ireland, pp. 45-52.

[29] **Hoskins, J.C. & Himmelblau, D.M.**, Artificial neural network models of knowledge representation in chemical engineering. *Computers & Chemical Engineering*, 1988, vol. 12, pp. 881-890.

[30] **Chen, S., & Billings, S.A.**, Neural networks for nonlinear dynamic system modeling and identification. *International Journal of Control*, 1992, vol. 56, pp. 319-346.

[31] **You, Y. & Nikolaou, M.**, Dynamic process modeling with recurrent neural networks. *AIChE Journal*, 1993, vol. 39 pp. 1654-1667.

[32] **Bhat, N., Minderman, P., McAvoy, T. & Wang, N.**, Modeling chemical process systems via neural computation. *IEEE Control Systems Magazine*, April 1990, vol. 10, pp. 24-30.

[33] **Hunt, K.J., Sbarbaro, D., Zbikowski, R. & Gawthrop, P.J.**, Neural networks for control systems—a survey. *Automatica*, 1992, vol. 28, pp. 1083-1112.

[34] **Morris, A.J., Montague, G.A. & Willis, M.J.**, Artificial neural networks: Studies in process modelling and control. *Chemical Engineering Research & Design (Transactions of the Institution of Chemical Engineers)*, 1994, vol. 72, pp. 3-19.

[35] **Bhat, N. & McAvoy, T.J.**, Use of neural nets for dynamic modelling and control of chemical process systems. *Computers & Chemical Engineering.*, 1990, vol. 14, pp. 573-583.

[36] **Narendra, K.S. & Parthasarathy, K.**, Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1990, vol. 1, pp. 4-27.

[37] **Hopfield, J.J.**, Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 1982, vol. 79, pp. 2554-2558.

[38] **Turner, P., Montague, G. & Morris, J.**, Nonlinear and direction-dependent dynamic process modelling using neural networks. *IEE Proceedings - D, Control Theory Appl.*, 1996, vol. 143, pp. 44-48.

[39] **Warwick, K., Irwin, G.W. & Hunt, K.J.**, (Eds.), *Neural Networks for Control and Systems*, Peter Peregrinus, Stevenage, UK, 1992.

[40] **Psaltis, D., Sideris, A. & Yamamura, A.A.**, A multilayered neural network 0controller. *IEEE Control Systems Magazine*, April 1988, vol. 8, pp. 17-21.

[41] **Nguyen, D.H. & Widrow, B.**, Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, April 1990, vol. 10, pp. 18-23.

[42] **Garcia, C.E. & Morari, M.**, Internal model control – 1. A unifying review and some new results. *Industrial and Engineering Chemistry. Process Design and Development*, 1982, vol. 21, pp. 308-323.

[43] **Psichogios, D.C. & Ungar, L.H.**, Nonlinear internal model control and model predictive control using neural networks. *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, 1990, pp. 1082-1087.

[44] **Psichogios, D.C. & Ungar, L.H.**, Direct and indirect model based control using artificial neural networks. *Industrial Engineering Chemistry Research*, 1991, vol. 30, pp. 2564-2573.

[45] **Hunt, K.J. & Sbarbaro, D.**, Neural networks for nonlinear internal model control. *IEE Proceedings - D, Control Theory Appl.*, 1991, vol. 138, pp. 431-438.

[46] **Nahas, E.P., Henson, M.A. & Seborg, D.E.**, Nonlinear internal model control strategy for neural network models. *Computers & Chemical Engineering*, 1992, vol. 16, pp. 1039-1057.

[47] **Åström, K.J.**, Adaptive feedback control. *Proceedings of IEEE*, 1991, vol. 75, pp. 185-217.

[48] **Lightbody, G. & Irwin, G.W.**, Direct neural model reference adaptive control. *IEE Proceedings - D, Control Theory Appl.*, 1995, vol. 142, pp. 31-43.

[49] **Williams, R.J. & Zipser, D.**, A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989, vol. 1, pp. 270-280.

[50] **Mills, P.M., Zomaya, A.Y. & Tadé, M.O.**, Adaptive model-based control using neural networks. *International Journal of Control*, 1994, vol. 60, pp. 1163-1192.

[51] **Yang, Y.Y. & Linkens, D. A.**, Adaptive neural-network-based approach for the control of continuously stirred tank reactor. *IEE Proceedings - D, Control Theory Appl.*, 1994, vol. 141, pp. 341-349.

5

Model Predictive Control

OVERVIEW

This chapter gives a brief overview of the Model Predictive Control methodology. The concepts, theory and applications of Model Predictive Control are covered. It focuses on two particular methods, namely, Dynamic Matrix Control and Model Algorithmic Control. The chapter concludes with a discussion of the use and implementation of neural network models to produce nonlinear Model Predictive Control systems.

5.1 Introduction

While simple control schemes are in common use throughout industry and can perform satisfactorily there is still much effort given to developing new control techniques to improve the quality and consistency of industrial processing.

The main difficulty in controlling many chemical processes are the inherent nonlinear and multivariable nature of the processes. PID controllers are extremely common but often fail to provide adequate control of a process over a wide operating range and are not designed to handle multi-input, multi-output systems. PID control techniques can be used satisfactorily for many processes with localised control objectives, however more complex processes, which are difficult to decouple and involve constraints, require advanced control approaches.

Advanced multivariable control systems had little impact in the process industries until the development of Model Predictive Control (MPC), sometimes referred to as Model-Based Predictive Control (MBPC), systems [1]. MPC systems offer a multivariable control methodology and include flexible constraint handling capabilities. It is these capabilities which have proven MPC to be successful in many practical applications.

MPC was initially developed within the oil and petrochemical industry and as a consequence of its success has raised much interest in academia.

Many advanced control systems have concentrated on including an explicit dynamic model of the process within the controller design (see §4.5.2.). In MPC systems this model is used to make long-range predictions of the process outputs for given control inputs. This predictive capability is used to obtain a set of control inputs which drive the process outputs to their desired reference levels. Different MPC design procedures have been reported which vary in the types of models used, methods for selecting the process inputs and procedures for establishing the reference levels.

MPC gets its name from the manner in which the control law is computed. At the present time step, t, the plant responses over a specified prediction horizon, N_p, are considered. The objective is to select a set of N_c future control moves, known as the control horizon, to minimise a cost function based on a desired output trajectory over the prediction horizon (Figure 5-1). The optimisation variables are the control actions N_c time steps into the future. After N_c time steps it is assumed the control action remains constant. Although future control inputs are calculated, only the first one is actually implemented in the process. Plant output measurements are then obtained and at time $t+1$ the computation is repeated with the horizon moved by one time interval. This is known as a receding horizon control strategy.

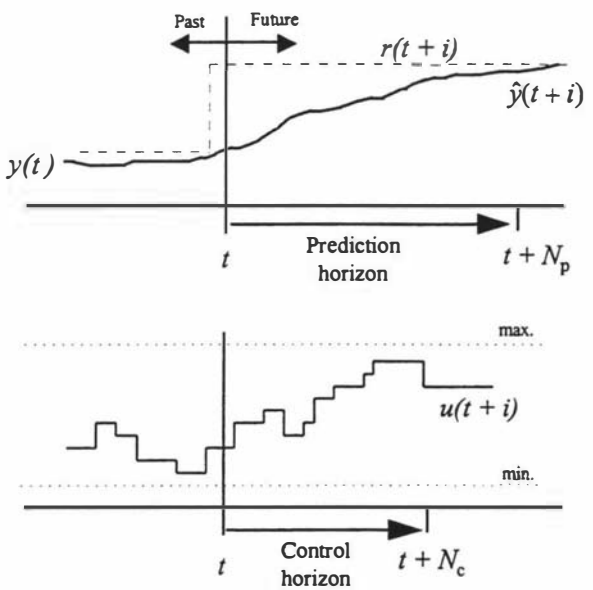


Figure 5-1: Receding horizon approach of MPC.

A model predictive controller consists of the following basic elements:

- A dynamic model from which future plant output predictions, $\hat{y}(t+i), i > 0$, can be generated based upon past information and a future set of control input moves, $\boldsymbol{u}(t+i), i > 0$.
- Knowledge of the future reference signal, $\boldsymbol{r}(t+i), i = 1,\ldots,N_p$, for the plant outputs. These may be predefined or determined through the use of a separate process optimisation procedure.
- A cost function, $\Phi(\boldsymbol{e},\boldsymbol{u})$, which is a function of the future output errors, $\boldsymbol{e}$, and the set of proposed control inputs, $\boldsymbol{u}$.
- An optimisation routine which minimises $\Phi(\boldsymbol{e},\boldsymbol{u})$ either unconditionally or subject to various constraints placed upon control and process operation.
- Various filters which help to broaden the range of the performance and provide robustness against unmeasured disturbances or plant-model mismatch.

5.2 Types of MPC systems

The interest during the past decade in Model Predictive Control techniques can be traced back to a set of papers which appeared in the late 1970's regarding Dynamic Matrix Control [2] and Model Algorithmic Control [3, 4]. These control methodologies have since become well known and have been applied successfully to various industrial processes. Other MPC algorithms which have been proposed are very similar to these.

5.2.1 Dynamic Matrix Control (DMC)

Dynamic Matrix Control (DMC), one of the most popular MPC strategies, was devised within the petrochemical industry and had initial applications to a fluid catalytic cracker [2]. DMC uses a linear step response representation of the process to model the system. The general form of the models used are:

$$\hat{\boldsymbol{y}}(t+1) = \boldsymbol{y}(t) + \boldsymbol{A}\Delta\boldsymbol{u} \tag{5.1}$$

where $\hat{\boldsymbol{y}}(t+1)$ is the future model output predictions,

$\boldsymbol{y}(t)$ is the measured plant outputs,

$\boldsymbol{A}$ is the dynamic matrix of step response coefficients and

$\Delta\boldsymbol{u}$ is the matrix of past and current control moves.

Using this model, the optimisation problem consists of finding, at each time t, the set of N_c future control moves, $\Delta \boldsymbol{u}(t+i)$, $i = 1,...,N_c$ which minimises:

$$\Phi = \sum_{i=1}^{N_p} \boldsymbol{e}(t+i)\boldsymbol{Q}\,\boldsymbol{e}(t+i)^T + \sum_{i=1}^{N_c} \Delta \boldsymbol{u}(t+i)\,\boldsymbol{R}\,\Delta \boldsymbol{u}(t+i)^T \tag{5.2}$$

where $\boldsymbol{e}(t) = \boldsymbol{r}(t) - \hat{\boldsymbol{y}}(t)$,

$\Delta \boldsymbol{u}(t) = \boldsymbol{u}(t) - \boldsymbol{u}(t-1)$ and

$\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive weighting matrices.

The objective function acts on the vector of output errors from the process $\boldsymbol{e}$, and penalises large changes in the control inputs, $\boldsymbol{u}$.

The DMC optimisation strategy is carried out subject to defined process constraints and offers flexible constraint handling within the control strategy.

The control problem can be formulated to minimise the objective function subject to the following constraints:

- Constraints on the control inputs:

$$\boldsymbol{u}_{min} \le \boldsymbol{u}(i) \le \boldsymbol{u}_{max} \tag{5.3}$$
$$i = t,..., t + N_c$$

- Constraints on the deviation from the previous control input:

$$\boldsymbol{u}(i-1) - \Delta \boldsymbol{u}_{max} \le \boldsymbol{u}(i) \le \boldsymbol{u}(i-1) + \Delta \boldsymbol{u}_{max} \tag{5.4}$$
$$i = t,..., t + N_c$$

- Constraints on the model outputs:

$$\boldsymbol{y}_{min} \le \hat{\boldsymbol{y}}(i) \le \boldsymbol{y}_{max} \tag{5.5}$$
$$i = t,..., t + N_p$$

- Constant control action assumed for time steps beyond the control horizon, N_c:

$$\boldsymbol{u}(i) = \boldsymbol{u}(t + N_c - 1) \tag{5.6}$$
$$\forall\, i > t + N_c - 1$$

The optimisation outlined above constitutes a standard quadratic program which can be solved on-line in a finite number of steps.

The algorithm provides a wide choice of control response through the use of the various tuning parameters available. The parameters which can be adjusted are N_p, N_c, $\boldsymbol{Q}$ and $\boldsymbol{R}$.

N_p is usually chosen to be equal to the step response settling time of the system. From experience it has been found [5] that:

- the control action is more aggressive,
- the system response is faster and
- the closed-loop system is less robust to plant-model mismatch

when

- N_p is decreased and/or
- N_c is increased and/or
- $\boldsymbol{R}$ is decreased.

To add robustness to the control scheme by accounting for any unmodelled disturbances or plant/model mismatch, DMC applies a correction to obtain a better prediction. This correction is based on the difference between the actual value of the controlled output at the current sampling instant and that predicted by the model:

$$\boldsymbol{d}(t) = \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t) \tag{5.7}$$

It is assumed that this value remains constant over the prediction horizon, t to $t + N_p$.

The error signal can therefore be rewritten as:

$$\boldsymbol{e}(t+i) = \boldsymbol{r}(t+i) - \hat{\boldsymbol{y}}(t+i) - \boldsymbol{d}(t), \quad i = 1,\ldots, N_p \tag{5.8}$$

The reference signal, $\boldsymbol{r}(t)$, is not necessarily a constant setpoint but could be a smooth trajectory leading to a desired setpoint, r^*. An example of such a reference trajectory is a first order relationship:

$$\boldsymbol{r}(t+i) = \beta^i \boldsymbol{r}(t) + (1-\beta^i) r^* \tag{5.9}$$

where: $\boldsymbol{r}(t) = \boldsymbol{y}(t)$, $i = 1,\ldots, N_p$ and $0 \le \beta \le 1$.

Further extensions could include a higher order relationship or r^* could be made to be time varying.

A block diagram representation of a general MPC system is given in Figure 5-2.

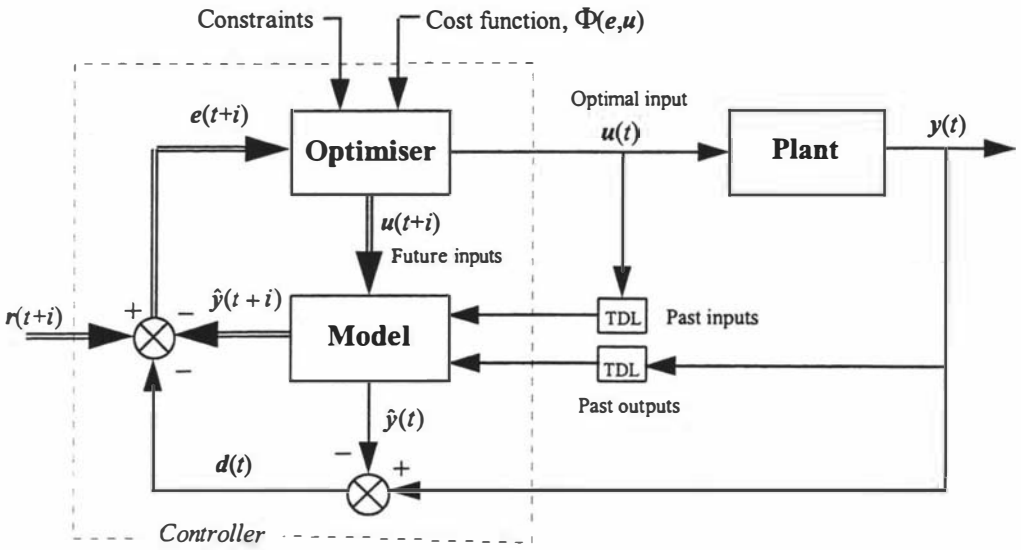


Figure 5-2: MPC block diagram

5.2.2 Model Algorithmic Control (MAC)

Richalet *et al* [3] initially discussed the development of a Model Predictive Heuristic Control methodology. They reported the successfully application of their control software system IDCOM (Identification-Command) to a PVC plant, a steam generator and an oil refinery distillation column. Later the term Model Algorithmic Control was given to the same approach [4].

Generally MAC is very similar to DMC apart from three distinct aspects [1].

1. Instead of using a step response model involving $\Delta \boldsymbol{u}$ an impulse response model involving $\boldsymbol{u}$ is used.
2. There is no separate parameter for the control horizon, N_c. Therefore $N_c = N_p$.
3. The disturbance estimate of equation (5.7) is filtered. The disturbance becomes:

$$\boldsymbol{d}(t+i) = \alpha \boldsymbol{d}(t+i-1) + (1-\alpha)[\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)] \qquad (5.10)$$

with $i = 1$ to N_p, $\boldsymbol{d}(t) = 0$ and $0 \le \alpha \le 1$.

Equation (5.10) adds a first order exponential filter in the feedback path. The filter parameter, α, is a much more direct and convenient tuning parameter than the cost function weights or horizon length in the general MPC formulation.

5.3 The benefits and disadvantages of MPC

The benefits of a MPC strategy include:

- Flexible constraint handling is built into the control strategy. This includes both explicit constraints on the actuators and implicit constraints such as limits on the velocity of input changes. This feature is very attractive for practical applications. Instead of ignoring the operating constraints at the design stage and dealing with the constraints in an ad hoc manner during implementation, MPC allows the designer to build constraints into the control methodology during the design and implementation of the controller. If any additional constraints become apparent during operation, for example due to equipment problems, these can be included in the optimisation routine in a flexible manner and the control actions will be adjusted accordingly. Additionally, during the course of operation one can examine the constraint boundaries on which the optimal solution lies and hence gain an understanding of the critical constraints. This may lead to improvements to the process in order to improve the control.
- A multivariable control strategy. MPC optimises process inputs to achieve a number of control objectives.
- MPC can be easily incorporated as a global control scheme with existing PID control loops. In fact in large systems MPC is generally implemented as a master global controller which regulates the setpoints of localised single input control loops. This enables the control to realise shorter sampling times than that of the actual MPC controller.
- The possibility of extension to nonlinear control through the use of nonlinear models.

The disadvantages of MPC include:

- The need for a dynamic black-box model of the process, which requires development time and effort. To obtain the model test signals have to be applied to the plant during operation which may be in conflict with production objectives.
- Generally the control should be tested on a simulation prior to implementation on the actual plant which requires the development of a simulator.
- Due to the optimisation calculations required at each sampling time the computing demands are very high. The computation time may not be a problem for slow processes where long sampling times can be used but MPC may be impossible to implement for some systems with faster dynamics.

As computing power, communication links and software becomes more advanced MPC will become realisable for many systems. Generally it has been found that the benefits of MPC far outweighs the costs of implementation. As will be discussed in the next section the MPC has proven not only to be beneficial in terms of improved control but also financially very attractive.

5.4 Applications and advances of MPC

The application of MPC systems to industrial processes is becoming increasingly popular. As previously mentioned DMC was originally developed within the oil and petrochemical industry and IDCOM had its first applications in the chemical and oil industries. Today the petrochemical industry remains the main user and promoter of advanced control such as MPC, the most common application being the control of crude oil distillation columns where it has proved successful [6].

The technology is spreading into other areas such as the pulp and paper industry [5, 7]. The benefits of applying DMC to a fibre recycle plant were reported [5] as being:

- an increase in production,
- a decrease in variability of product quality,
- a reduction in resources and
- a reduction in rejected product.

As well as these benefits the process constraints can be highlighted in real time through the use of the constrained control scheme. This enables improvements to the process for increased productivity or product quality to be identified quickly.

This application of MPC was very successful and the pay back period associated with the project was measured in months [5].

Richalet *et al* [3] report similar attractive pay backs for a two distillation columns and a catalytic cracking tower. The economic gain achieved from reducing energy usage and increasing the rate of production due to MPC was estimated to be over $100,000 per annum for each process.

The model plays a vital role in the operation of the controller and in effect, tuning of the controller mainly amounts to determining an appropriate model to use. Most of the development effort is spent in identifying a process model and relatively little effort is necessary for tuning the control strategy [6]. Much effort therefore has been placed on obtaining better plant descriptions as this directly affects the achievable quality of the control. MPC systems have traditionally been based on linear models but it is possible

for nonlinear models to be used. Linear MPC is simpler to implement than the nonlinear version as linear models are easier to identify and use in computation and also allow the use of algebraic optimisation. Processes having linear MPC systems in place have been shown to have produced significant control improvements as discussed above. If a nonlinear model was used however, these improvements could be even greater since the processes in question are inherently nonlinear in nature. If a nonlinear model is used then a more complex nonlinear programming approach is required to solve the optimisation problem.

An example of using a nonlinear analytical model within MPC [8] is the implementation of nonlinear MPC to control a fixed-bed water-gas shift reactor. Lee *et al* [9] have also generalised many of the MPC techniques into a generic structure giving Generic Model Control (GMC). They report the application of GMC using a nonlinear model to control a forced circulation evaporator with a performance improvement over linear DMC. Other types of nonlinear models such as fuzzy models [10] and neural networks (see next section) have also been used with MPC.

5.5 Model Predictive Control and neural networks

The effort necessary to identify an analytical nonlinear model, implement it within an MPC strategy and overcome obstacles such as estimating unmeasured states has prompted researchers to investigate the use of artificial neural network models in MPC. The nonlinear characteristic, short development time compared to conventional methods and the ease of inclusion into a nonlinear optimisation program, make neural networks ideal for use within nonlinear MPC.

Recently there has been much reported on the application of neural networks to MPC systems. Encouraging results have been achieved from a number of MPC simulation studies. These include CSTR systems [11, 12, 13, 14, 15], distillation columns, [13, 16], fermentation [15], evaporation [17], a steel rolling mill [18] and a lab-scale neutralisation plant [19].

All report positive results from the studies and favourable comparisons with linear MPC or classical control techniques.

Most included the neural network within the MPC system as the internal model. Lee & Park [13], however, used the neural network in parallel to a standard DMC controller to act as a feedforward controller to counteract the effect of disturbances. The network is trained on-line using disturbance error learning. The network is adapted through the use of the portion of the control input signal due to the unmodelled effects of the system emanating from the DMC controller.

Mills *et al* [17] developed an adaptive MPC scheme for an evaporator simulation. The neural network was adapted on-line using the backpropagation method with a history-stack of past process data.

A further method of using neural networks with MPC is to use one neural network as the internal model and then train a second network to mimic the action of the optimisation routine while the system is operating. This network is trained to produce the same control input for a given process output as the optimisation routine. The second network can then replace the optimiser and plant model once it has been satisfactorily trained [20].

5.5.1 *n*-step-ahead predictions using neural networks

Most commonly the neural networks used in the above studies were time-delayed networks that were trained off-line using historical data.

It is a requirement for models used within MPC that they are able to produce predictions of future outputs given only certain past information. This ability is referred to as *n*-step-ahead prediction where *n* is the length of the prediction horizon (N_p).

For a time-delay net to achieve N_p-step-ahead predictions it must either

- have N_p output neurons each predicting for a different future time-step or
- act as a one-step-ahead predictor and be iterated N_p times.

The first approach is usually not practical since it is common for N_p to be in the order of 30 time steps, seriously enlarging the dimensions of the network required.

The latter approach implements the network as an unfolded recurrent network. Therefore time-delayed neural networks that are used for *n*-step-ahead prediction are required to have an externally recurrent structure (see Figure 5-3) and should be trained using a dynamic recurrent training approach which are discussed in the next section.

If the network was trained using a standard feedforward method then it would be expected that the length of the prediction horizon would be limited by the number of delays used in the input tapped delay lines. Any predictions beyond the order of the inputs to the network could possibly lead to unstable behaviour.

The studies referred to in the previous section mostly used time-delayed neural networks within the MPC scheme and iterated the networks during implementation to predict into the future. The training seemed to use standard feedforward methods without consideration for the fact the networks would be used in a recurrent fashion. This could possibly lead to poor prediction results.

Dynamic recurrent training methods enable a prediction horizon which is independent of the order of the delayed lines on the inputs to the network.

5.5.2 Dynamic recurrent training

Time-delayed neural networks that have a feedback connection from the output layer of the network to the input layer are trained using a parallel training approach (Figure 4-7). This type of network structure (Figure 5-3) can be referred to as either an *externally recurrent* [21], a *semi-recurrent* [22] or a *NARX recurrent* [23] network structure.

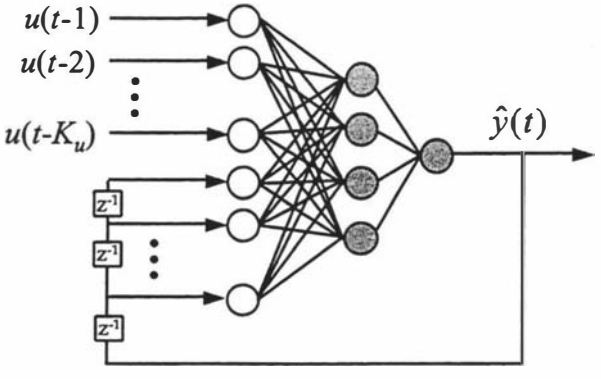


Figure 5-3: An example of a externally recurrent neural network

When an externally recurrent network is trained using a standard feedforward method then the weight updates do not take into account the output errors of previous time instances. If it is then employed in a recurrent fashion the net outputs can become unstable as prediction errors are propagated through the network.

One way to train an externally recurrent network is to view it as a feedforward network that is unfolded over time. If the system data is N time steps long then we create N copies of the network (Figure 5-4). The recurrent connections are now feedforward connections from one network to the subsequent network and the network can be trained

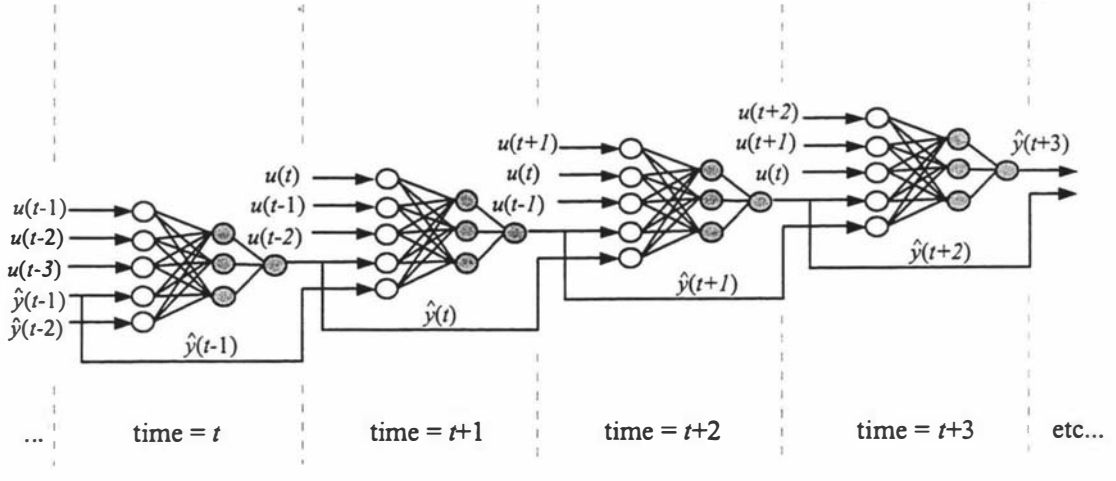


Figure 5-4: An externally recurrent network unfolded over time.

as one large feedforward network while ensuring that the weights for each network remain the same. This method of training is called *backpropagation through time* (BPTT) [20, 24, 25].

Extending the standard backpropagation algorithm to account for time one has to calculate the deltas for the output layer at time t, by considering the deltas for the input neurons at time $t + 1$ to $t + K_y$. All of these input neurons are connected to the corresponding output neurons at time t. According to equation (4.17) the delta for a sigmoidal hidden layer neuron at time t is given by:

$$\delta_j^l(t) = \sum_i \left(\delta_i^{l+1}(t) w_{ij}^{l+1}\right) f_l'(s_j(t)) \tag{5.11}$$

For a recurrent network the input layer neurons must also have deltas. It is easy to show that the delta for a input layer neuron is:

$$\delta_j^0(t) = \sum_i \left(\delta_i^1(t) w_{ij}^1\right) \tag{5.12}$$

since the input neurons are linear. The delta for a output neuron must include a term for the input layer deltas propagated back from the future time steps. Therefore the delta for an output layer neuron becomes:

$$\delta_j^L(t) = \left(d_j^L(t) - a_j^L(t) + \sum_{\tau=1}^{K_y} \delta^0(t+\tau) \right) f_L'(s_j(t)) \tag{5.13}$$

where: K_y is the number of delayed outputs used, $\delta^0(t + \tau)$ is the corresponding delta from the input layer at time $t + \tau$ and $f_L'(s_j(t))$ is the derivative of the output neuron activation function.

In equation (5.13) the delta propagates the required information from $t = N$ back to the current time instant. The BPTT method can easily be implemented by altering the algorithm for standard backpropagation. The method has a relatively low computational cost but high memory requirements. In using the BPTT method convergence can still be difficult to achieve as the error surface for recurrent networks can contain significantly more local minima than training for a 1-step-ahead network. Alternatively the algorithm can be truncated by unfolding the network over only the last n time steps rather over the whole length of the data, N. This reduces the computation and memory requirements of the training but information more than n time steps in the past will be sacrificed.

The BPTT method has been successfully used for externally recurrent networks in modelling a wastewater treatment plant and a catalytic reformer [21] where they were used to attain long-range predictions over an arbitrary length horizon. The conjugate

gradient method together with a random search algorithm were used within a backpropagation through time training scheme to adapt the weights.

There are many other approaches to training recurrent networks in the literature [26] most of which have been developed for fully recurrent networks. These alternative recurrent training methods are more efficient than BPTT since they are able to train networks without the need for unfolding the network. One such algorithm which can be used to train recurrent networks of any form is known as *real-time recurrent learning* (RTRL) [27]. This method is an inherently on-line algorithm which is well suited for real-time learning. This approach calculates the gradient recursively and, although it has a higher computational cost than BPTT, it is more memory efficient [20, 28, 29].

5.5.3 Input spread-encoding

An alternative approach to using time-delayed networks to perform long-range predictions is given in Gomm *et al* [30]. Their approach uses a spread encoding technique applied to the network inputs and allows a network trained using a standard feedforward algorithm to be used recursively. Spread encoding involves mapping individual input values to several input nodes, each one active over a limited excitation range. The technique is based upon an interpretation of these nodes as forming a discrete probability distribution with mean equal to the coded value. They found that using spread encoding aided in improving the prediction accuracy of the neural network when used as a recurrent model. The major drawback of a spread encoding scheme is that with an increasing number of variables the number of inputs into the network rises exponentially.

5.6 Stability issues of MPC

Research into moving-horizon control has been carried out within two frameworks. In the chemical process control literature, moving horizon control is popularly called MPC and in the systems theory literature it is referred to as receding-horizon control (RHC).

The early MPC strategies [2, 3] were developed within the industrial sector and no explicit provision was made in the problem formulation to ensure close-loop stability.

In the linear case, closed loop stability of MPC is assured by proper choice of the prediction and control horizons and the weighting matrices in the objective function. However once constraints are introduced in the control law the proof of stability and robustness become difficult.

With the successful application of MPC in industry the academic control community took an interest in the technique. It was only then that serious investigation into the

stability of moving-horizon control was undertaken. Industrial practitioners continued to successfully implement MPC despite the theoretical reservations of their academic counterparts who had yet to develop satisfactory stability criteria for constrained MPC. Generally the practitioners are satisfied simply by the fact that MPC has been successful in many applications and do not necessarily see that the proof of stability is vital.

Recently much effort from academic circles has been given to finding stability conditions of RHC and these are still being investigated. Such research into RHC [31, 32, 33] has provided the missing guarantees of stability.

The key factor with RHC stability involves imposing terminal constraints on the predicted tracking errors over some future time range. In other words, the output errors are constrained to zero beyond the prediction horizon and the control moves themselves should become zero beyond the control horizon.

$$\boldsymbol{y}(i) = r, \forall\, i > t + N_p \tag{5.14}$$

and

$$\Delta\boldsymbol{u}(i) = 0, \forall\, i \geq t + N_c \tag{5.15}$$

These stability constraints are the fundamental difference between the MPC and RHC literature. Such terminal constraints guarantee asymptotic stability on the finite horizon. However the terminal constraints may cause a high degree of input activity and for constrained MPC have the potential to cause conflicts with inequality constraints on the system inputs and subsequently introduce infeasibility problems in the optimisation. Possible remedies could be to use longer horizons or introduce setpoint conditioning [34]. Additionally, solving a nonlinear optimal control problem with terminal constraints requires, in principle, an infinite number of iterations.

An alternative approach to specifying the above constraints (equations (5.14) and (5.15)) is to introduce infinite weights on the relevant end-points within the objective function [35]. Infinite weights in the objective function however cause similar problems to terminal constraints in the presence of input constraints. By reducing the size of the terminal weights the optimisation can become more realisable but stability is no longer guaranteed.

A recent paper by Sistu & Bequette [36] analyses the closed-loop stability for a system that does not require any explicit terminal constraints in the control formulation. Instead the stability is based on computable criteria for a given system using a convergence theory of iterative processes.

In a different context, Michalska & Mayne [37] devised a dual-mode control strategy in which a neighbourhood encompassing the desired operating point for the system is

defined. The control strategy employs a linear, locally stabilising control law inside the neighbourhood and a receding horizon controller outside it. This allows the requirement of an exact solution to the optimal control problem to be relaxed.

Little has been reported on the stability requirements of neural network-based MPC. However a recent paper [38] demonstrates that a receding horizon control strategy based on a class of recurrent networks can stabilise nonlinear systems. The analysis follows the approach of that of Mayne & Michalska [33].

Many questions still remain unresolved regarding the stability and robustness of constrained MPC even for linear systems. Despite the theoretical issues that remain unanswered both linear and nonlinear MPC have proven to be extremely successful when applied to a range of industrial control problems both in academia and within industry.

The stability analysis of neural network-based MPC is beyond the scope of this research.

5.7 References

[1] **Garcia, C.E., Prett, D.M. & Morari, M.**, Model predictive control: theory and practice–a survey. *Automatica*, 1989, vol. 25, pp. 335-348.

[2] **Cutler, C.R. & Ramaker, B.L.**, Dynamic matrix control–a computer control algorithm. *Proceedings of AIChE National Meeting*, 1979, Houston, Texas.

[3] **Richalet, J., Rault, A., Testud, J.L. & Papon, J.**, Model predictive heuristic control: applications to industrial processes. *Automatica*, 1978, vol. 14, pp 413-428.

[4] **Rouhani, R. & Mehra, R.K.**, Model algorithmic control (MAC); basic theoretical properties. *Automatica*, 1982, vol. 18, pp 401-414.

[5] **Austin, P.C. & Bozin, A.S**, Applying dynamic matrix control in the process industries. *Transactions of the Institute of Measurement & Control*. 1996, vol. 18, pp. 32-41.

[6] **Richalet, J.**, Industrial applications of model based predictive control. *Automatica*, 1993, vol. 29, pp. 1251-1274.

[7] **Ricker, N.L., Sim, T. & Cheng, C.-M.**, Predictive control of a multi-effect evaporation system. *Proceedings of the American Control Conference*, 1986, Seattle, USA, pp. 355-359.

[8] **Wright, G.T. & Edgar, T.F.**, Nonlinear model predictive control of a fixed-bed water-gas shift reactor: an experimental study. *Computers & Chemical Engineering*, 1994, vol. 18, pp. 83-102.

[9] **Lee, P.L., Newell, R.B. & Sullivan, G.R.**, Generic model control – a case study. *The Canadian Journal of Chemical Engineering*, 1989, vol. 67, pp. 478-484.

[10] **Skrjanc, I. & Matko, D.**, Fuzzy predictive controller with adaptive gain. *Proceedings of Advances in Model-Based Predictive Control Conference*, 1993, Oxford, UK.

[11] **Saint-Donat, J., Bhat, N. & McAvoy, T.J.**, Neural net based model predictive control. *International Journal of Control*, 1991, vol. 54, pp. 1453-1468.

[12] **Willis, M.J., Di Massimo, C., Montague, G.A., Tham, M.T. & Morris, A.J.** Artificial neural networks in process engineering. *IEE Proceedings - D, Control Theory Appl.*, 1991, vol. 138, pp. 256-266.

[13] **Lee, M. & Park, S.**, A new scheme combining neural feedforward control with model predictive control. *AIChE Journal*, 1992, vol. 38, pp. 193-200.

[14] **Psichogios, D.C. & Ungar, L.H.**, Direct and indirect model based control using artificial neural networks. *Industrial Engineering Chemistry Research*, 1991, vol. 30, pp. 2564-2573.

[15] **Chen, Q. & Weigand, W.A.**, Dynamic optimization of nonlinear processes by combining neural net model with UDMC. *AIChE Journal*, 1994, pp. 1488-1497.

[16] **Morris, A.J., Montague, G.A. & Willis, M.J.**, Artificial neural networks: Studies in process modelling and control. *Chemical Engineering Research & Design (Transactions of the Institution of Chemical Engineers)*, 1994, vol. 72, pp. 3-19.

[17] **Mills, P.M., Zomaya, A.Y. & Tadé, M.O.**, Adaptive model-based control using neural networks. *International Journal of Control*, 1994, vol. 60, pp. 1163-1192.

[18] **Sbarbaro-Hofer, D., Neumerkel, D. & Hunt, K.**, Neural control of a steel rolling mill. *IEEE Control Systems Magazine*, June 1993, vol. 13, pp. 69-75.

[19] **Draeger, A., Engell, S. & Ranke, H.**, Model predictive control using neural networks. *IEEE Control Systems Magazine*, October 1995, vol. 15, pp. 61-66.

[20] **Hunt, K.J., Sbarbaro, D., Zbikowski, R. & Gawthrop, P.J.**, Neural networks for control systems—a survey. *Automatica*, 1992, vol. 28, pp. 1083-1112.

[21] **Su, H.-T., McAvoy, J. & Werbos, P.**, Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial Engineering Chemistry Research*, 1992, vol. 31, pp. 1338-1352.

[22] **Turner, P., Montague, G. & Morris, J.**, Nonlinear and direction-dependent dynamic process modelling using neural networks. *IEE Proceedings - D, Control Theory Appl.*, 1996, vol. 143, pp. 44- 48.

[23] **Lin, T., Horne, B.G., Tino, P. & Giles, C.L.**, Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 1996, vol. 7, pp. 1329-1338.

[24] **Rumelhart, D.E., Hinton, G.E. & Williams, R.J.**, Learning internal representations by error propagation. In: *Parallel Distributed Processing*, vol. 1, Rumelhart, D.E. & McClelland, J.L. (Eds.), MIT Press, Cambridge, MA, USA, 1986, pp. 318-362.

[25] **Werbos, P.J.**, Backpropagation through time: what it is and how to do it. *Proceedings of the IEEE*, 1990, vol. 78, pp. 1550-1560.

[26] **Pearlmutter, B.A.**, Dynamic recurrent neural networks. Technical Report CMU-CS-90-196, Carnegie Mellon University, Pittsburgh, PA, 1990.

[27] **Williams, R.J. & Zipser, D.**, A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989, vol. 1, pp. 270-280.

[28] **Hush, D.R. & Horne, B.G.**, Progress in supervised neural networks. *IEEE Signal Processing Magazine*, January 1993, vol. 10, pp. 8-39.

[29] **Piché, S.W.**, Steepest descent algorithms for neural network controllers and filters. *IEEE Transactions on Neural Networks*, 1994, vol. 5, pp. 198-212.

[30] **Gomm, J.B., Lisboa, P.J.G., Williams, D. & Evans, J.T.**, Accurate multi-step-ahead prediction of non-linear systems using the MLP neural network with spread encoding. *Transactions of the Institute of Measurement & Control*. 1994, vol. 16, pp. 203-213.

[31] **Clarke, D.W. & Scattolini, R.**, Constrained receding horizon predictive control. *IEE Proceedings - D, Control Theory Appl.*, 1991, vol. 138, pp. 347-354.

[32] **Rawlings, J.B & Muske, K.R,** The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control,* 1993, vol. 38, pp. 1512-1516.

[33] **Mayne, D.Q. & Michalska, H.,** Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control,* 1990, vol. 35, pp. 814-824.

[34] **Scokaert, P.O.M. & Clarke, D.W.,** Stabilising properties of constrained predictive control. *IEE Proceedings - D, Control Theory Appl.,* 1994, vol. 141, pp. 295-304.

[35] **Demircioglu, H. & Clarke, D.W.,** Generalised predictive control with end-point state weighting. *IEE Proceedings - D, Control Theory Appl.,* 1993, vol. 140, pp. 275-282.

[36] **Sistu, P.B. & Bequette, B.W.,** Nonlinear Model-Predictive Control: Closed-loop stability analysis. *AIChE Journal*, 1996, vol. 42, pp. 3388-3402.

[37] **Michalska, H. & Mayne, D.Q.,** Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control,* 1993, vol. 38, pp. 1623-1633.

[38] **Kambhampati, C., Delgado, A., Mason, J.D & Warwick, K.,** Stable receding horizon control based on recurrent networks. *IEE Proceedings - D, Control Theory Appl.,* 1997, vol. 144, pp. 249-254.

6

The Evaporator Neural Network Modelling Approach

OVERVIEW

This chapter presents an approach for the development of an artificial neural network model of the pilot-plant evaporator and is broken up into the sections outlined below.

Neural network modelling approach

The approach taken utilises prior knowledge to form a model with a modular structure. An overview of the use of prior knowledge, and of modular modelling techniques as applied to neural networks, is given before embarking on a description of the model formation. The potential benefits of applying such techniques are discussed.

Sub-network training

The training methodology used to train the neural networks is described.

Application of the approach to an evaporation effect

To demonstrate the modelling approach and training methodology a neural network is developed for a sub-section of the evaporator. The decisions made as to model variables and structure are included in the discussion together with the analysis of the model performance. The sub-networks which make up the model are compared with linear models of similar structures. The sub-networks are combined to form a complete model of the sub-section.

Alternative models

The modular neural network model is compared with neural networks of alternative structures and a multivariable, linear regression model.

6.1 Modelling objectives

The neural network model of the pilot-plant evaporator is to be implemented within a Model Predictive Control strategy.

As discussed in the previous chapter, a model within a MPC strategy is required to be;

- a dynamic representation and
- have *n*-step-ahead prediction capability.

Additionally it is also beneficial if the model is fast, simple to run and easily modified.

Fast running - at each sampling instant the optimisation routine within the controller must run the model several times in order to determine the optimum set of control inputs. The simpler and faster the model the quicker the optimisation can be performed.

Easily modified - it is an advantage if the model can easily be updated to accommodate any change in the process characteristics over time. These could be due to physical plant alterations or variations in the operating conditions.

It would desirable, therefore, if the approach taken in modelling the evaporator using neural networks took these requirements into consideration.

The modelling approach proposed in this work, makes use of prior knowledge within a modular structure in order to improve the training of the network, give the model a meaningful structure and make it modifiable

To produce a dynamic model that could perform long-range prediction a time-delayed externally recurrent network was decided upon.

These aspects will be discussed in the following section.

6.2 Neural network modelling approach

6.2.1 Dynamic modelling

Time-delayed networks with output feedback were used for the evaporator neural network model. The main reason for selecting this method of dynamic modelling is because of its simplicity. Since historical data is used for the modelling it is easy to form the data into a time-delayed format by shifting the columns of data. The output feedback allows the model to perform long-range predictions as would be expected of such an externally recurrent neural network.

Fully recurrent networks, although requiring fewer inputs, can be difficult to train. There is also evidence to suggest that time-delayed networks that are externally

recurrent (NARX-recurrent networks) are better at learning long-term dependencies than a fully recurrent network [1]. Filter-based networks [2] (discussed in §4.4.2) are essentially fully recurrent networks. The filter feeds back the previous neuron output and therefore each neuron contains a recurrent link with the filter constant acting as the connection weighting.

6.2.2 Overview of using prior knowledge

One of the many claims about neural networks is that little theoretical or prior knowledge about a particular system is needed in order to produce a neural network representation of it. This was one of the aspects which made the initial interest in neural networks soar. Although this is partly true, and neural networks are essentially 'black-box' models, it is a very fool-hardy modeller who ignores any available knowledge in pursuit of a purely 'learned' model. A technique which can achieve this would be highly desirable for many problems however, in practice, neural networks are simply not that powerful. For most industrial applications some knowledge is usually readily available and should be utilised in order to obtain better models of systems. This knowledge may take the form of well-known theoretical relationships, information from other modelling exercises, practical understanding from observing the process or just an operator's rule-of-thumb that seems to have some basis. Any reliable information that is available can enhance a "black-box" model and so should, if possible, be made use of. On the other hand one does not want to restrict the neural network's adaptation process and constrain it to characterise only a particular part of the whole problem. One must seek a balance and allow the network to model any relationships within limits that are based on the knowledge available.

Three benefits of including prior information in neural network development are [3];

- it prevents the network from making impossible or implausible predictions,
- it may improve the ability of the network to generalise and
- it may increase the efficiency of the training.

In practice it is very difficult not to use any prior knowledge in developing neural networks but it is the degree to which it is incorporated in the process which can vary. The areas in which the use of prior knowledge can be classified are: input conditioning, design approaches and training approaches [4]. Input conditioning includes the selection of network inputs and data pre-processing. Design approaches include the formation of semi-parametric models and modular modelling. Using prior knowledge in modular designs will be discussed in detail in section 6.2.3.

Selection of network inputs

When modelling a particular system one must decide which input variables should be included in the model in order to correctly estimate the outputs. Usually the necessary output variables are obvious choices but the selection of the input variables is not always straightforward. The choice is either to use all available input variables or just a subset of them. The selection should be based on the available knowledge about the system and which inputs would be most helpful in obtaining an accurate representation. If one simply uses all available information the network might still settle to a good solution but may take a longer time in doing so as it has to determine itself which inputs are important and which are not. If the known cause-and-effect relationships between inputs and outputs are made use of, it can speed up the network training process and would more likely result in a valid model.

Data pre-processing

If particular relationships between inputs and outputs are known to exist then this knowledge can also be used in pre-processing the inputs before they are presented to the network. Examples of how the inputs can be manipulated include; using square root or power terms, combining two inputs as a ratio or product, or even including time-delayed inputs to represent pure delay in the system. These techniques can improve both the training process and the quality of the resulting model.

Semi-parametric models

Prior knowledge can be used by combining neural networks with fixed-form parametric models to create a *hybrid* or *semi-parametric* structure. A parametric model is one which has been derived from first principles, existing empirical correlations or known mathematical transformations. There are many ways in which one can combine neural networks with such parametric models, depending on the problem at hand [4], but the common approaches are either a series- or parallel-hybrid model structure.

Series hybrid models [5] make use of the neural network to estimate the model parameters which are fed to the parametric model. The network is trained using the output error from the combined model.

In parallel hybrid models [4] the outputs of the neural network and the parametric models are combined to form the total model output. The parametric model serves as a best estimate of the real system and the neural network is trained on the error between the parametric model and actual system outputs. The neural network effectively accounts for any uncertainties in the parametric model.

Hybrid modelling approaches such as these have been used to model fermentation processes [4, 5] and a water treatment plant [6].

Training approaches

Using prior knowledge in training can involve imposing conditions on the model during the training phase in order to guarantee obedience to the actual process constraints. This may take the form of inequality constraints on the inputs and outputs of the network as well as the network weights [3]. Another approach is to impose functional conditions on the model by way of an additional penalty or objective function during training [4]. Generally these training approaches are difficult to implement and the form they should take is not easily determined.

Another approach is to concentrate on structuring the learning environment for a neural network in order to enhance the training. A network could be trained to perform simple similar tasks or sub-tasks before learning a more complex task. The learning process could also be enhanced by correctly ordering the presentation of the examples to the network during training. Initially presenting easier tasks and building up more difficult examples after the basic relationships have been mastered. These types of approaches have been studied with application to developing a neural network-based window filter for image processing problems [7].

6.2.3 Overview of modular modelling

Since neural networks are 'black-box' models they have little structure which is apparently relevant to the actual system they represent. It is very difficult to analyse what is occurring within a network and how outputs are calculated. It is important for a model to be analysed in order to determine how reliable it is and to give some idea of its suitability for a certain task. For neural networks it is necessary to be able to verify that the training process has resulted in a network with a plausible characteristic but this is a difficult task to do effectively. In backpropagation-type networks every neuron contributes to the network output and so it is difficult to determine which neurons are playing an important role and which are not. In clustering networks, like the Radial Basis Function networks, not all neurons contribute to the output and therefore the functionality of the network can be more easily analysed. However, since the clusters are determined on spatial criteria the structure would still have little relevance to the physical system unless dealing with spatial data as with pattern recognition problems. In the case of function approximation there is little physical relevance in applying a spatial decomposition of the data [8]. However clustering in an RBF network may have relevance to the underlying function.

In order to give some structure to neural network models *modular modelling* approaches can be used. One motivation for using a modular modelling approach is to give meaning to a neural network so that the model can be analysed and understood with respect to the actual system. An additional and equally appealing motivation is that these modular approaches can provide an efficient method for tackling large modelling problems, resulting in improved training and generalisation.

It is also believed that the nature of information processing in the human brain is modular. Individual functions are broken into sub-processes and are executed within separate modules without mutual interference [9].

A modular network is a network containing localised computation in which each module is an independent system interacting with other modules to perform a more complex function. A definition suggested by Ronco & Gawthrop [8] also includes the notion that each module must have an interpretable and relevant function which is meaningful according to the physical properties of the system being modelled.

As well as increased interpretability, modular modelling approaches also offer the following advantages [8]:

- They give a reduction in the number of network connections and therefore fewer model parameters (weights). The networks are more sparsely connected which means improvements in computational speed, training and generalisation.
- They enable analysis of network behaviour which may shed light on unknown or poorly understood relationships.
- They avoid interference from irrelevant or redundant learning.
- They are easily modifiable and updated as the system or needs change. Only a portion of the whole model would require re-training.
- Hybrid models containing modules of varying characteristics are able to be developed to suit specific applications. Modules may be either nonlinear or linear, or could be dynamic while others are static. This hybrid nature could include different activation functions, network structures or training methods within the one overall model.

Modular modelling involves:

- decomposition of the modelling task;
- organisation of the modular architecture and
- the integration of the modules to form the complete model.

Problem decomposition can be achieved systematically using clustering algorithms. However, as discussed previously this is only physically relevant for data that is spatially distributed. The data can be considered spatially distributed about the input space but in general this will not reflect the structure of the physical system. Therefore the decomposition of a dynamic modelling problem is usually carried out in an *ad hoc* approach using prior knowledge of the system.

The structure of the modules can be determined using a self-organising procedure where the module structure is adapted or constrained during the training or alternatively, using prior knowledge to determine the structures. Self-organising methods take the form of pruning algorithms [10] or incremental methods like cascade correlation networks [11]. For pattern recognition problems Happel & Murre [9] used genetic algorithms to determine the best module structures prior to training.

Approaches to combining of modules are many and varied. Some modules may only receive certain inputs or all the modules may receive the same inputs. The network output may take the form of a decision layer where the model output is a function of all the individual module outputs. Alternatively, each module could receive the same inputs and the model output could be determined by a gating network [12]. Each of the modules are effectively competing against each other. Another option is to arrange the modules sequentially so that the input of a particular module is made up of other module outputs.

The above approaches to model decomposition, structure determination and module interconnection assume that no prior knowledge about the system is available and are most suitable when applied to classification and recognition problems.

For dynamic system modelling, modular approaches that rely on using prior knowledge seem to be most appropriate. These approaches, although difficult to formalise, use prior knowledge to construct interconnected neural network models according the functional and topological structure of the process. The decomposition of the task into smaller elements and re-combination of the outputs is dependent on the system and the intended use of the model. The design usually breaks the process model into groups of related variables or modules representing processing units within the total system. The sub-networks can then be connected according to the structures and functions of the actual process. This approach is analogous to a block diagrammatic description of a process where subsequent blocks do not influence blocks occurring previously. Therefore if feedback occurs within the actual system then feedback links must be made explicit within the structure of the overall network model.

Mavrovouniotis & Chang [13] demonstrate a method of utilising prior knowledge with time-delay networks. They grouped related variables together to form sub-networks

which are loosely coupled and arranged in layers producing a hierarchical modular network. The initial layer of sub-networks directly receive external inputs and their inputs feed into the next layer. Each sub-network uses a particular subset of the input variables but the exact relationships between the variables are determined by training the network as a whole. A single neuron represents the smallest sub-network that can be used in this manner. Mavrovouniotis & Chang demonstrated their idea with a feedforward network in which the neurons (representing sub-nets) were only coupled to certain inputs according to functional principles. This formed a network with localised areas of computation. They found that this network was more accurate, faster to train and easier to analyse than a traditional feedforward network modelling the same data.

6.2.4 Sub-network approach

Using a time-delayed neural network to model the evaporator system can require a large number of inputs due to the number of variables and the time constants of the plant. More complex networks have longer training times as the learning algorithm has to optimise a function of higher dimension.

It was decided to break the modelling task into a set of smaller, more manageable parts in order to reduce the model complexity. A combination of small nets with only one output as a function of a few strongly related inputs may be a better alternative than one large network. After considering the additional benefits of using prior knowledge and modular modelling outlined in the previous sections it was decided to apply prior theoretical knowledge to the model decomposition and that each sub-network or module contain localised connections.

In this study the neural network model uses prior knowledge to create a modular model in two ways:

i) The modelling task is decomposed into smaller elements or modules representing sub-units within the total system. The sub-networks are combined to form the full model according to the structure of the actual process. Each network is used to predict only one output variable. The inputs are chosen to be those variables which are known to strongly influence that particular output. Any secondary effects between the system variables that are not modelled within the sub-networks will still be accounted for by the way they are joined together.

ii) Following the approach of Mavrovouniotis & Chang [13] each sub-network contains localised computation by grouping related inputs together. The structure of the network is determined using the following guidelines:

- The delayed inputs for each variable are connected to two hidden neurons.

- All the inputs relating to a particular time instant are also connected to a single hidden neuron.
- All the hidden neurons are connected to a single output neuron.

Using the guidelines in ii) a single-output, externally-recurrent network with S_i inputs each having d delays has S_h hidden neurons, given by

$$S_h = 2(S_i + 1) + d. \tag{6.1}$$

The above guidelines represent a simple and compact approach to structuring the sub-nets. It is believed that two hidden neurons coupled to each input would adequately characterise each input over the time spanned. These neurons attempt to capture the dynamics inherent within each input variable. Using only one neuron for this task would not be sufficient to describe complex behaviour whereas using more than two neurons is unlikely to significantly improve the characterisation.

The additional hidden neurons attempt to capture how the variables relate to each other at each time instance and do not use any dynamic information. These neurons also prevent the network structure becoming simply a parallel combination of the inputs. Using only one hidden neuron for each time step is believed to be sufficient to characterise the input data and can be considered as effectively a 'snap-shot' of the system inputs at a particular time.

An example of the structure of a sub-network with localised connections and output feedback is illustrated in Figure 6-1.

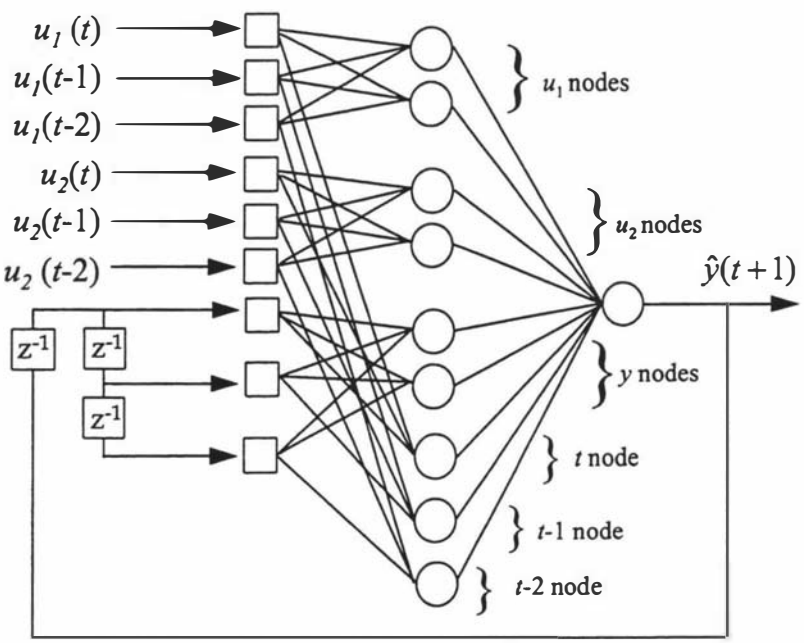


Figure 6-1: An externally recurrent sub-network with localised computation.

6.3 Sub-network training

As discussed in Chapter 5, externally recurrent neural networks should be trained using dynamic recurrent algorithms. One of the simplest and easiest to implement of these is the backpropagation-through-time (BPTT) algorithm. It was decided to use the BPTT algorithm with the Levenberg-Marquardt (LM) method of optimisation for the evaporator sub-networks. This enabled a much faster convergence than would be possible using standard backpropagation or even conjugate gradient optimisation with BPTT. This training method will be denoted as Levenberg-Marquardt Through Time (LMTT) and is outlined in the next section.

6.3.1 Sub-network training algorithm

Backpropagation-through-time with LM optimisation

To combine BPTT with the LM method one must use BPTT when determining the Jacobian matrix. For the LM method the Jacobian matrix is formed from the backpropagated output layer deltas using equation (4.33) which is rewritten below.

$$\delta_i^L = f_L'(s_i(t)) \tag{6.2}$$

where $f_L'(s_j(t))$ is the derivative of the output neuron activation function at time t.

As shown in Chapter 5 (equation (5.13)) for BPTT training, the deltas for the output layer nodes include the deltas from the input nodes at future time steps. Therefore, in the combined LMTT method, the delta for the i^{th} output node at time t becomes

$$\delta_i^L(t) = f_L'(s_i(t))\left(\sum_{\tau=1}^{K_y}\delta^0(t+\tau)\right) \tag{6.3}$$

where K_y is the number of delayed outputs used, and

$\delta^0(t+\tau)$ is the corresponding delta from the input layer at time $t+\tau$

The LMTT method involves the following steps;

1. Presentation of the input data to the network and generation of network outputs.
2. Calculation of the output error vector (all outputs from time, $t = 0,\ldots, N$).
3. Backpropagation of the output node deltas for time, $t = N$ to current time through each time step using equation (6.3) together with the equations relating to the deltas for the hidden and input layer neurons (equation (5.11) and (5.12)).

4. Formation of the Jacobian matrix of the network using equations (4.34), (4.35) and (4.29).
5. The Jacobian matrix and the error vector are used to update the weights using equation(4.32).
6. Repeat steps 1-5 until LM convergence criteria has been met or the maximum number of iterations has been reached.

A detailed outline of the algorithm, complete with all the equations, is given in Appendix 7.

When using a BPTT approach it is necessary during the presentation phase (step *1*) that the input matrix is updated with the network predictions. Therefore the input data matrix which is presented to the network has the following form for a single-input, single-output system with K_u input delays and K_y output delays.

$$\mathbf{X} \equiv \begin{bmatrix} u(0) & \cdots & u(-K_u) & y(0) & \cdots & y(-K_y) \\ u(1) & \cdots & u(1-K_u) & \hat{y}(1) & \cdots & \hat{y}(1-K_y) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u(t) & \cdots & u(t-K_u) & \hat{y}(t) & \cdots & \hat{y}(t-K_y) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u(N-1) & \cdots & u(N-K_u-1) & \hat{y}(N-1) & \cdots & \hat{y}(N-K_y-1) \end{bmatrix} \tag{6.4}$$

Where N is the number of time steps of training data, $t = -K+1, \ldots, 0, 1, 2, \ldots, N$,
$K = \max\{K_u, K_y\}$

$\hat{y}(t)$ is the network prediction at time t, and

$y(t)$, the actual plant output, is used for all $t < 1$ when $\hat{y}(t)$ is not available.

The training data is presented in batch mode however, after the presentation of each row of the input matrix, the model predictions are used to update the input matrix by replacing the $y(t)$ values. This ensures that the network is trained using a parallel method, as illustrated in Figure 4-7 (§4.4.2), and the resulting errors (step 2) are calculated based on a recurrent implementation. Essentially the past outputs from the actual plant after K time steps are not required as the remainder of the output values are generated by the network itself.

Prior to the development of the LMTT training algorithm, a number of trials using a standard feedforward training method was applied to the training of the evaporator sub-networks [14]. These trials involved using data from the analytical evaporator model and were reasonably successful. The LM algorithm was used in a batch feedforward

training method (series identification in Figure 4-7). To approximate the externally-recurrent nature of the nets the input data matrix was updated during training by inserting the network outputs after each training epoch. This resulted in an input matrix of the form given in equation (6.4). The prediction error to be minimised is obtained by a standard feedforward implementation of the network.

These trials worked adequately for the simulation data however it was observed that a descending error gradient during the training did not necessarily relate to a improvement in the externally-recurrent network implementation. This was because the updating of the input matrix only approximated how the network was actually implemented and the externally-recurrent structure is not accounted for in the adaptation of the weights. For this reason this method was not used for modelling with the real plant data.

6.3.2 Neural network MATLAB files

MATLAB was used to perform all the development and simulation of the neural networks. Many of the macros written were adapted from the functions within the MATLAB *Neural Network Toolbox* (*ver. 2.0*) [15]. A brief description and listing of the important MATLAB macros that were developed for the data processing and network training, simulation and analysis is given in Appendix 8.

6.4 Application of sub-network training to an evaporation effect

6.4.1 Introduction

To determine the success of sub-network modelling, an example system was modelled using the approach outlined above. The example system used was a single effect of the falling-film evaporator pilot-plant. We wished to consider this system as a general case for an evaporation effect. Therefore the second effect of the pilot-plant was selected as for study, since it was located between two similar evaporation stages rather than at either the start or end of the process. For convenience throughout the remainder of this chapter the effect 2 system will be given the label Eff2.

6.4.2 Evaporator data

As mentioned in Chapter 3 the random input sequences were not binary sequences but were generated such that several (greater than two) levels of input were used. For linear identification PRBS inputs are adequate but for the purpose of nonlinear identification the data needs to represent the full range of the system's operating modes and therefore

more than two operating levels are necessary on each input variable. Pseudo-random sequences were applied to the inputs of the evaporator simulation in order to excite the model states and demonstrate the dynamics of the evaporator. The same random input trial data used in the validation of the analytical model is used for training, testing and validating the neural model.

For the neural network development the plant data sets were designated as shown below.

Table 6-1: Allocation of plant data for neural network development

Label	Plant Trial Data
Training data set	Trial 2 data
Testing data set	Trial 3 data (*first half*)
Validation data set 1	Trial 3 data (*all*)
Validation data set 2	Trial 4 data

The selection of the training data set for the neural network development is an important task. The training data should cover the full range of the input space for all the variables, so as to create a valid model over a wide range of operating conditions. Also it should be of sufficient quantity. A common rule-of-thumb is that the number of training vectors should be at least ten times the number of network parameters. The Trial 1 data was chosen as the training set since it covers the widest range of the input space for the majority of the variables and also is the largest set available. In practice however it is difficult to get a single set of data that covers the input space over the necessary variables. As will be seen, the predictions for some of the variables do suffer from the fact that the training data does not span the entire input space for all the variables.

6.4.3 Selection of model variables

The Eff2 output variables that are of interest in this example are the product temperature at the base of the effect (T_{t2}), the concentrate level (L_2) and the output flowrate from effect 2 (Q_{p2}).

The output variables of interest are only those that can be readily measured on the actual plant. This is why the temperature in the concentrate level is used and not the evaporation tube temperature since the temperature transducers on the plant are located at the base of each effect.

As was the case during the discussion of the analytical model validation (§3.3.1) the temperature at the base of the effect n will be denoted as T_n with the '*l*' subscript removed and the effect flowrate becomes Q_n with the '*p*' subscript removed.

The neural network model of the effect requires five input variables to predict the three output variables above. The input variables include the flowrate into the effect (Q_1) the temperature of the incoming product (T_1), the temperature of the downstream effect (T_3) and the pump speed (N_2). Note that T_1 would also be approximately equal to the temperature of the heating vapour in the effect shell.

Three sub-networks were created, one to predict each output. The structure of each sub-network was determined from theoretical relationships known through experience with the analytical model but could equally be determined through knowledge of the physical process characteristics or through correlation analysis.

Each network was used to predict only one output variable and its inputs are variables which are known to strongly influence that particular output. The resulting sub-nets have multi-input, single-output structures.

The following equations define the functionality of the three sub-networks which make up the effect model.

The effect 2 temperature was selected to be a function of the upstream and downstream effect temperatures (T_1 and T_3), the flowrate into the effect (Q_1) and previous values of the effect 2 temperature.

$$T_2 = f(T_1, T_3, Q_1, T_2) \tag{6.5}$$

These variables were selected since the upstream and downstream temperatures directly influence the effect temperature and the mass of product in the evaporation tube strongly influences the rate at which the temperature changes.

The effect 2 concentrate level was chosen to be a function of the flow in (Q_1) and out (Q_2) of the effect, the temperature of the effect (T_2) and previous values of the level.

$$L_2 = f(Q_1, Q_2, T_2, L_2) \tag{6.6}$$

These variables were selected since according to the analytical model the change in the level is a function of the product flow into the effect and product and vapour flows out of the effect. The temperature has also been included as a variable to represent the vapour flow since the higher the temperature the greater the evaporation.

The flow out of effect 2 was determined as a function of the differential pressure between effect 2 and 3 which is estimated using the temperature difference $T_2 - T_3$, the pump speed (N_2), the concentrate level (L_2) and the previous value of the flowrate.

$$Q_2 = f(T_2 - T_3, N_2, L_2, Q_2) \tag{6.7}$$

The basis for selecting these variables is the Bernoulli energy balance equation through the product transport system.

The reasons for using the temperature difference in this network as opposed to the pressure difference are two-fold.

Firstly, networks using differential pressure have been tried but it was found that the performance was not significantly better than when using the temperatures.

Secondly, since the pressure variables are outputs additional sub-networks are required in the model to obtain the pressure estimates. This requires more development time and may introduce more errors into the model.

It was felt that no benefit is gained by developing another sub-network while the overall model performance is not significantly improved.

6.4.4 Selection of sampling rate

Since the neural networks are discrete models a sampling rate must be chosen. In selecting the sampling rate one must consider its influence on the structure and accuracy of the model as well as the use of the model within a control strategy.

The data obtained from the evaporator has a sampling rate of five seconds. Therefore the sampling rate selected for the model must be a multiple of five seconds.

Data from the plant could have been obtained with a shorter rate, as low as 2 seconds, but it this was considered too short. No benefit would have been gained through the use of a shorter rate since the concentrate levels have the shortest time constants which are in the region of 30 seconds. It is assumed that time constants for the pumps and flowrates are negligible.

The sampling rate of the model influences the amount of past information the model has available to capture the dynamics of the system. With short time steps the number of delayed inputs needed for a given time history increases which may increase the model order and the dimensions of the neural network.

In a Model Predictive Control implementation the sampling rate of the model determines the rate of the controller. For the evaporator controller a short sampling time is desirable as this would more effectively control the relatively short time constants in the pilot-plant.

A compromise is needed between model complexity and accuracy and controller performance when selecting the sampling rate. For a real-time implementation the

computational requirements of MPC would also be a factor in the selection of the sampling rate. This was not considered here since the implementation is not real-time.

In modelling terms the choice of sampling rate in most cases is not critical because the range between too small and too large is relatively wide. A recommended rule-of-thumb given by Isermann [16] is that the sampling time, t_s, be between 1/15 and 1/5 of the 95% settling time for a system after a disturbance. From experience sampling rates within these limits have been found to work well.

In the evaporator the temperature dynamics are slower than those of the concentrate levels. Generally the settling time for the temperatures are of the order of 2 to 4 minutes whereas the levels are around 1 minute. The flowrates have rapid dynamics and for the neural network model are considered to be static in terms of the inputs and first order for the recurrent output variable.

It was decided to use a sampling rate (t_s) of 10 seconds, which is within Isermann's recommended range for the levels but just below that for the temperature variables. A sampling rate of 15 seconds would place the sampling rate outside the recommended range for the levels. It is believed that a sampling rate of 10 seconds represents a satisfactory compromise for the neural model and the subsequent MPC strategy.

6.4.5 Linear ARX sub-models

As a preliminary study prior to the neural network development an identification of linear sub-models was carried out. A set of ARX (AutoRegressive with eXogenous inputs) regression models were determined using a least squares approach [17]. The process of identifying such models would aid in the structure selection for the neural network sub-nets. The system identification approach of Ljung [17] is used and it is assumed that a basic understanding of this identification technique is well known so no introduction to the theory will be given here. The ARX models were identified using the least squares method given in the *MATLAB System Identification Toolbox* [18] which was developed by Ljung.

Since the formal procedures for ARX modelling are well developed, they would give a quick indication of the order of the models necessary to model the Eff2 variables. They would also prove to be useful for comparing the neural network modelling with a more conventional identification technique.

Three sub-models were created, one each for T_2, L_2 and Q_2. The input variables used for each ARX model were identical to that for the neural network models given above in Section 6.4.3. Each sub-model was of multi-input, single-output structure with the following form,

$$A(z)y(t) = B(z)u(t - n_k) \tag{6.8}$$

where n_k is the number of delays between the input and output and u is a vector of n_u input variables.

In equation (6.8) $A(z)$ represents a polynomial in the delay operator z^{-1}, as below

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_{n_a} z^{-n_a} . \tag{6.9}$$

This polynomial describes how past values of the output affect the current output, where n_a is the order of the polynomial.

Similarly, $B(z)$ is a $n_u \times 1$ vector, given by

$$B(z) = \begin{bmatrix} B_1 & B_2 & \cdots & B_{n_u} \end{bmatrix}^T \tag{6.10}$$

with each $B_i(z)$ representing a polynomial in the delay operator applied to input i

$$B_i(z) = b_{i1} z^{-n_k} + b_{i2} z^{-n_k - 1} + ... + b_{in_b} z^{-n_k - n_b} \tag{6.11}$$

where n_b is the order of input polynomials used. For a multi-input system n_b is a vector whose elements define the order of each input polynomial $B_i(z)$. The $B_i(z)$ polynomials describe how the past values of each input affect the current output.

The structures of the sub-models were determined by comparing the performance of models with different orders. The model structures which minimised Akaike's Final Prediction Error (FPE) criterion (see Section 4.3.5) were selected for use.

The results of the analysis gave the structures listed in Table 6-2.

Table 6-2: ARX sub-model structures

Model Output	Model inputs	Order of A, n_a	Order of B_i, n_b	Input to output delay, n_k	Number of parameters, N_w
T_2	$[T_1\ T_3\ Q_1]$	2	[4 2 6]	1	14
L_2	$[Q_1^f\ Q_2^f\ T_2]$	2	[3 1 3]	1	9
Q_2	$[T_1 - T_2\ L_2\ N_2]$	1	[1 3 3]	0	8

The first element of the B_i vector is the order of the first input variable, the second is the order of the second input etc.

The training data set was used to estimate the A and B polynomials for each model.

The residuals for the models were analysed to test the adequacy of the fit. Both the autocorrelation of the residuals and the cross correlation between the residuals and the inputs were analysed. Generally the results were found to be within the 99% confidence limits. Some of the autocorrelation test results dipped slightly beyond these limits

which indicates that the residuals are not necessarily random. More importance was attached to the independence of the residuals and the inputs which the cross correlation tests account for. These tests were within the confidence limits.

Each model was also tested with the validation data sets to test their generalisation capabilities. The test was to perform a long-range prediction of the actual data therefore each model was implemented with recurrent outputs. The MSE prediction results for each sub-model and each data set are given in Table 6-3. These MSE values are standardised by scaling the outputs to lie between 0 and 1 so that a comparison between the different variables can be made.

Table 6-3: Prediction errors for individual ARX sub-models

	Mean-Squared Error ($\times 10^{-3}$)		
Model	Training data	Validation 1 data	Validation 2 data
T_2	1.27	6.02	2.29
L_2	15.40	79.00	62.60
Q_2	1.67	11.40	3.64

It can be seen from these MSE results and the plots below that the ARX model can predict temperature and flowrate reasonably well, while the level predictions are not so good. As would be expected the prediction results for the validation data are worse than those for the training data set.

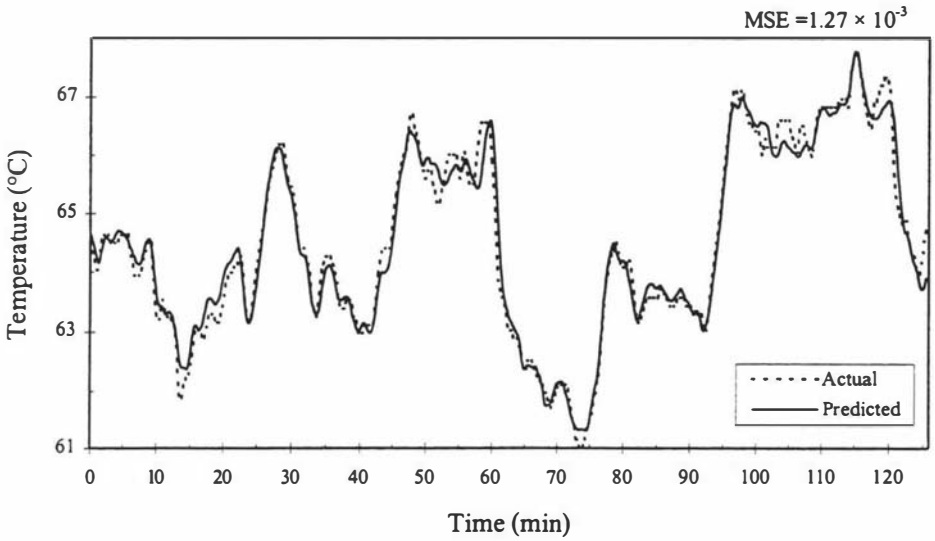


Figure 6-2: Prediction of training data for T_2 ARX model

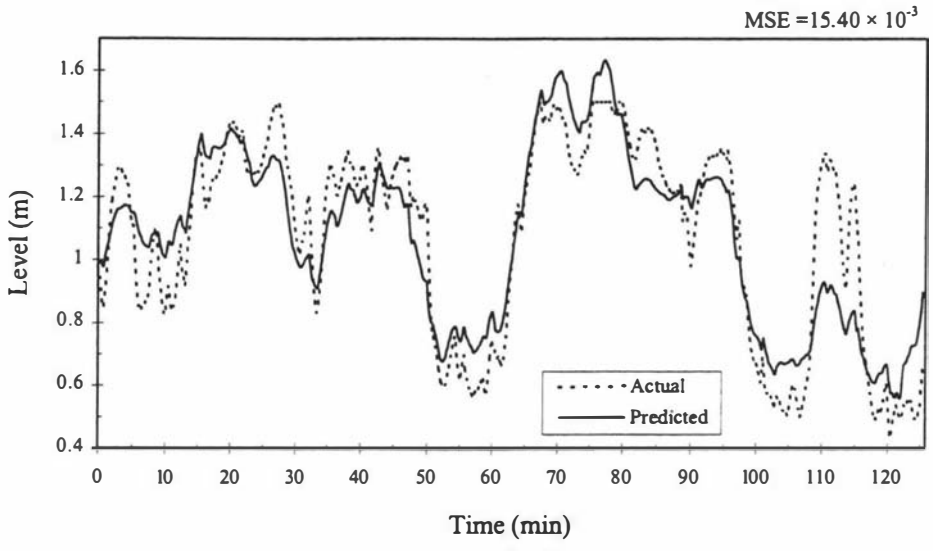


Figure 6-3: Prediction of training data for L_2 ARX model

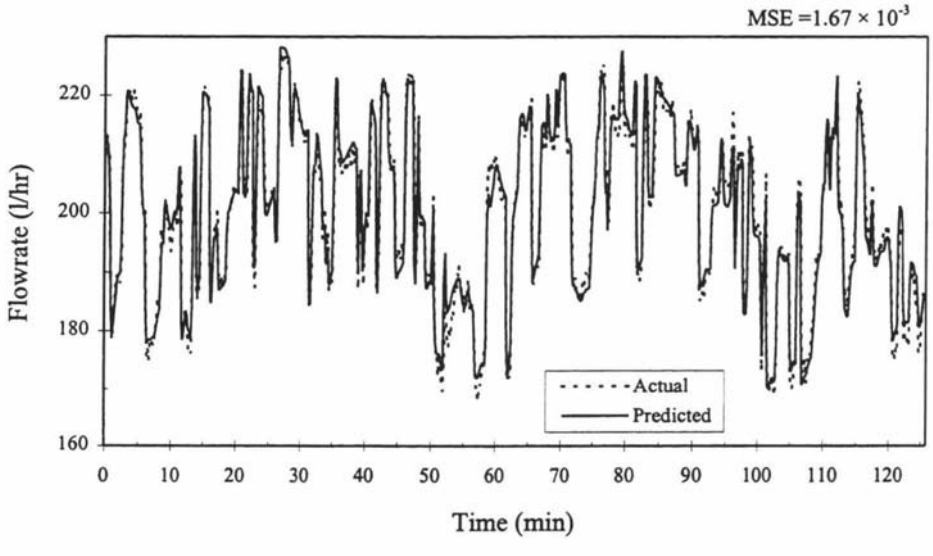


Figure 6-4: Prediction of training data for Q_2 ARX model

6.4.6 Sub-network model orders

As mentioned previously, the flows and levels have faster dynamics than do the temperature variables. This has implications as to the number of delays necessary on the inputs to the networks.

Since the networks have a large number of connections additional inputs can dramatically increase the number of parameters and hence can adversely effect the training process. To minimise the network size it was decided to limit the number of delays on each input to three at most. A longer time history can be used for a particular network by adjusting the time spacing between the delays. This ensures the network size is kept small while the correct time history of information is made available to the networks.

Correlation analysis was used to determine the number of time steps each input delay line should span. For a particular network, the cross-correlation between each input variable and the output variable was calculated over a range of ±20 time delays. The delay on the input variable which produces the largest cross-correlation gives an indication of how many time steps the delay lines need to cover.

An example of how the correlation between T_2 and T_1 varies is shown in Figure 6-5. This plot shows that the highest correlation between T_2 and T_1 is when T_1 is delayed by 3 time steps (which represents 30 seconds). Hence, for the T_2 network, the delays on T_1 should at least extend to time $t - 3$.

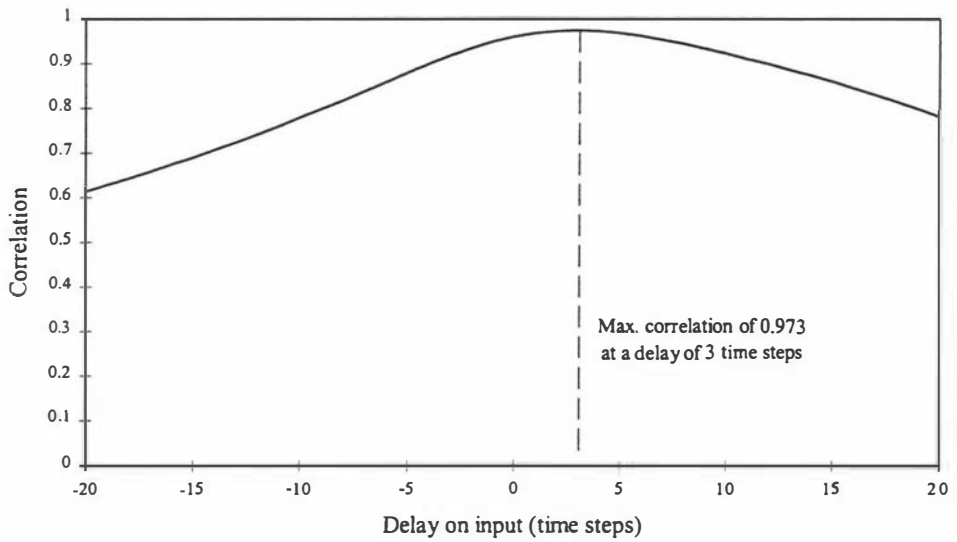


Figure 6-5: Correlation between T_2 (output) and T_1 (input) against delay on T_1

Analysing the correlations in this way is not conclusive as to which delays should be used. For example some inputs which are known to influence a particular output do not produce a significant cross-correlation figure over a range of delays. This may be due to a complex relationship between the two. However this analysis still does aid in giving a general idea of the time history that is required to model the relationships. The structure used for the linear ARX models also aid in the selection of suitable structures for the neural networks.

The level and temperature networks were made third order, while the flow network used only a first order delay applied to the output variable.

The temperature and level models both had three past values of each input passed to the input layer.

The delays used for each sub-net are represented by delay vectors, D_u and D_y. D_u defines the delays applied to the input variables and D_y the delays on the recurrent output. In general, if $D_u = [z_1\ z_2\ z_3]$ then the input variables are delayed by $t - z_1$, $t - z_2$ and $t - z_3$ time steps. D_y is defined in a similar fashion and applied to the network

output. To simplify the description of each network and to correctly create a locally-connected topology each input variable for a particular network undergoes the same delays. Note that the minimum value an element in the D_y vector can take is 1, representing a output delay of one time step. However the D_u vector can include 0 which represents an input which is not delayed. Each delay vector has a maximum of three elements since it was decided to limit each variable to be no more than third order.

This notation differs from that used for the ARX sub-models. As an example, the structure of the T_2 ARX model using the delay vector notation is shown below.

Table 6-4: T_2 ARX model structure in delay vector notation

Output delays	Input delays
D_y = [1 2 3 4 5]	T_1: D_u = [1 2 3 4]
	T_3: D_u = [1 2 3]
	Q_1: D_u = [1 2 3 4]

The inputs to the sub-nets however are restricted to each having the same order.

Temperature sub-network

The maximum correlations between the inputs T_1 and T_3 and the output T_2 are given in Table 6-5.

Table 6-5: Input-output correlations for T_2 sub-net

	Max. Correlation	Delay (time steps)
T_1	0.973	3
T_3	0.961	1

The flowrate Q_1 is not highly correlated with T_2 at any delay but it is known that changes in product flow through the evaporation tube do influence the temperature.

Based on these results the delays used for the T_2 sub-network were set at $D_u = D_y$ = [1 2 4]. This covers a time history of 40 seconds which encompasses the time constant for the effect temperature. This structure also agrees with that used in the T_2 ARX model (see Table 6-4 above).

Level sub-network

The flowrate data from the plant contains considerable measurement noise. This noise does not pose a significant problem for the temperature prediction but does cause

problems for the level predictions. Therefore the flow data was smoothed using an exponential filter:

$$Q_i^{\mathrm{f}}(t) = \alpha Q_i(t) + (1-\alpha)Q_i^{\mathrm{f}}(t-1) \tag{6.12}$$

where Q_i^{f} is the filtered flow and α is set to 0.05.

The filtered flowrate Q_1 gave a maximum correlation of 0.853 with the level, L_2. This occurred at a delay of 2 time steps. The measured output flowrate Q_2 slightly leads the L_2 measurement and therefore was most highly correlated at negative delay times. The leading of the Q_2 measurement is explained by the physical location of the sensors. The pump lies in the line between the base of the effect and the Q_2 flowmeter. Any changes in flowrate due to the variation in pump speed would be registered by the Q_2 flowmeter downstream either at the same to or prior to changes in the concentrate level measurement. The temperature input T_2, was not highly correlated with the level measurement however it is still a necessary input in order to give an indication of the evaporation rate within the calandria.

It was decided to use delays of $D_u = D_y = [1\ 2\ 3]$ for the L_2 sub-network. These agree with the ARX structure used for the L_2 model (see Table 6-2). The delays represent a time history of 30 seconds which is within the region of the time constant for the level.

Flow sub-network

The flow network is essentially only required to be a static model with just the current value of each input used. This is because; a) the flowrate is strongly dependent on the current values of the variables and b) in the analytical model no differential equation is needed to calculate the flowrate. This is also confirmed by the high correlation of the flowrate with the effect pump when there is no delay applied. Therefore $D_u = 0$ was used. To enhance the network performance the network was made first order by using the output at the previous time step as an input to the network, therefore $D_y = 1$. Theoretically the network does not require a recurrent output since in the analytical model no differential equation is used for the flowrate. However it was found that the sub-network performed better with a single recurrent output connection.

In the process of selecting the model structures other possibilities were investigated. This included;

- Varying the time delays on the T_2-net eg. $D_u = [1\ 4\ 8]$.
- Using more prior information for Q_2-net, eg. $D_y = D_u = [1\ 2\ 3]$.
- Varying the time delays on the L_2-net eg. $D_u = [1\ 2\ 5]$.

These alternatives did not perform any better than the structures outlined in the previous sections. The structures selected were accepted as being simple yet representative of the system characteristics and most closely resembling the correlation results.

6.4.7 Sub-network details

Activation functions

Two-layer (ie. single-hidden-layer) networks are used with sigmoidal hidden layer neurons and linear output layer neurons. Multi-layer perceptrons with this structure are most commonly employed in process modelling and have proven to be successful in other applications similar to the evaporator modelling [19].

Selection of hidden nodes

The level and temperature networks are structured according to the locally-connected approach discussed in 6.2.4 and illustrated in Figure 6-1. Therefore the number of hidden neurons used in these sub-nets is given by equation (6.1), that is 11 for both.

For the flowrate network the number of hidden nodes was determined by comparing the network training and generalisation for different-sized hidden layers. Several networks were trained with the number of hidden neurons ranging from 1 to 10. Figure 6-6 shows the variation in training MSEs for the Q_2 sub-network with different numbers of hidden nodes. This plot has had a few outliers removed which had errors way above 10×10^{-3}

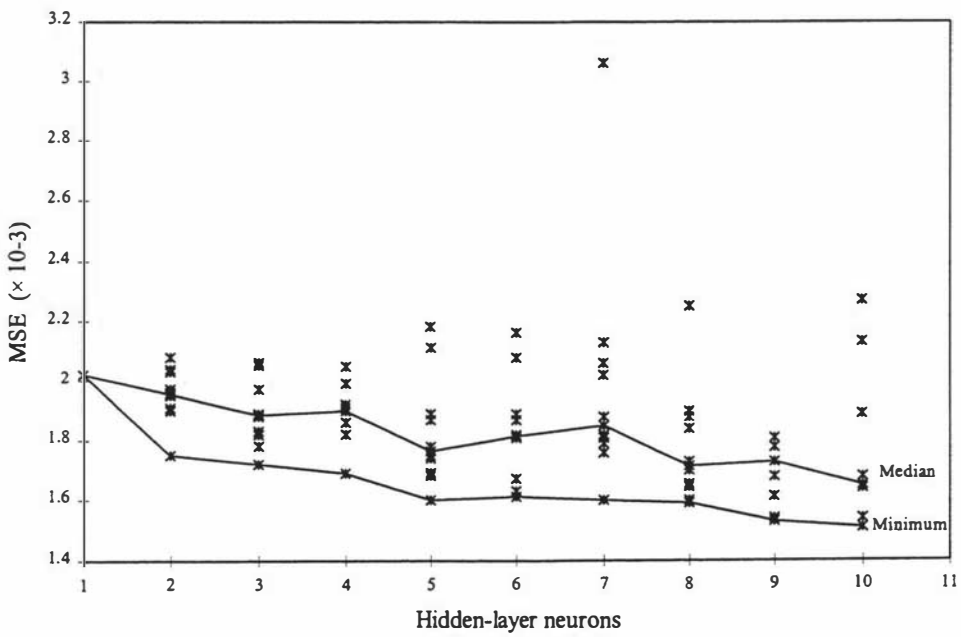


Figure 6-6: Training error against number of hidden neurons for the Q_2 sub-network.

for the networks with 9 or 10 hidden neurons. Lines for the median and minimum training errors have been included. In general as the number of hidden neurons is increased the median and minimum training errors decrease. However at the same time the variability in the error also increases so that good or very poor results are obtained for networks with more hidden nodes.

If these networks are tested by performing long-range prediction of the testing data we get the errors plotted in Figure 6-7. This plot shows that the median testing error remains reasonably constant as the number of hidden neurons increases. The minimum error values however are quite variable and do not appear to follow a particular trend. Again the results tend to have more spread when more hidden neurons are used.

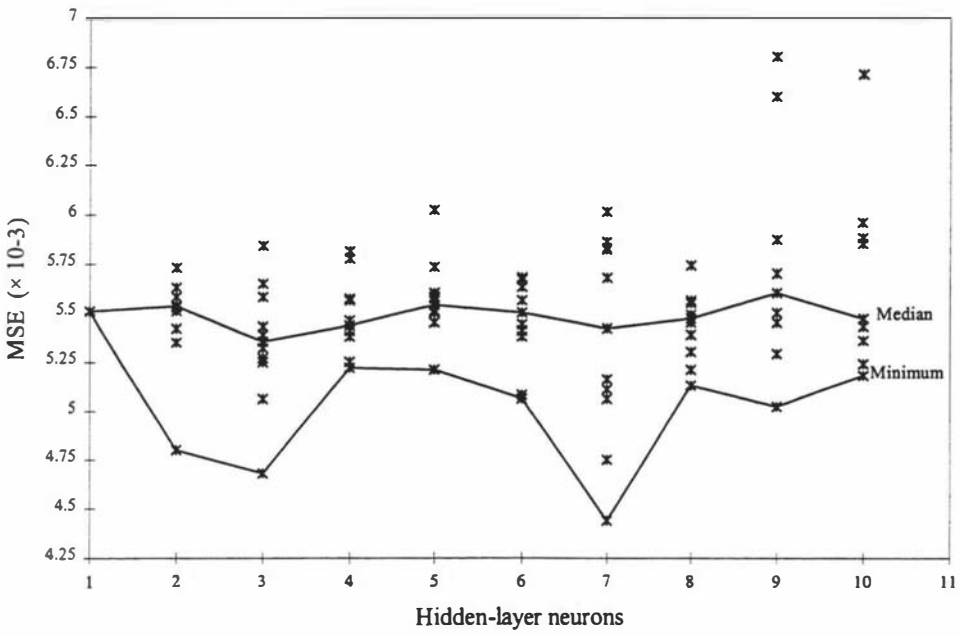


Figure 6-7: Testing error against number of hidden neurons for the Q_2 sub-network.

Based on these observations it was decided to use the sub-network with three hidden neurons which had the smallest testing error. It is felt that this represents a satisfactory compromise between the network size and accuracy.

Summary of network structures

Table 6-6 below summarises the structures of the three sub-networks for the Eff2 system. D_u is the delay vector for the input variables, D_y is the delay vector for the recurrent outputs, S_i is the number of input nodes, S_h is the number of hidden-layer neurons, S_o is the number of output neurons and N_w is the total number of parameters (weights and biases) of the network. Note that Q_i^f represents an exponentially filtered flowrate.

Table 6-6: Summary of sub-network structures for the Eff2 model

Output	Inputs	D_u	D_y	S_i-S_h-S_o	N_w
T_2	$[T_1\ T_3\ Q_1]$	[1 2 4]	[1 2 4]	12-11*-1	59
L_2	$[Q_1^f\ Q_2^f\ T_2]$	[1 2 3]	[1 2 3]	12-11*-1	59
Q_2	$[T_2 - T_3\ L_2\ N_2]$	0	1	4-3-1	19

* Locally connected layer.

Notice that, although the locally-connected temperature and level sub-nets have 11 hidden-layer neurons, the total number of weights and biases is only 59. A fully connected network with the equivalent number of neurons would have 155 weights and biases, more than two and a half times as many.

6.4.8 Sub-network training

The training data was used to individually train each of the sub-networks using the Levenberg-Marquardt with backpropagation through time (LMTT) method. The training data was scaled between –0.9 and 0.9.

Each network was tested during the training phase to determine when training should cease in order not to over-fit the training data. The test was to perform a long-range prediction of the testing data and the network weights were selected when the testing error was at a minimum. Training was deemed to be complete when a rise in the testing error occurred after the minimum point.

The training for these externally-recurrent sub-nets is a much more difficult task than it is for a feedforward training method. Recurrent networks have a more complex error surface than a feedforward network with many areas of local minima which can cause the optimisation to become 'stuck'.

Typically the sub-nets required less than 50 epochs each before the LMTT training terminated. In the LM optimisation method the algorithm stops if the step size or the error gradient become smaller than the user defined limits

In other work dealing with training externally recurrent networks using BPTT [19] the problems with local minima were overcome by applying a global random search technique to 'jog' the training and guide the search away from the local minimum. In the training of the evaporator sub-nets such a technique did not result in any improvements when problems occurred in the training.

The training was repeated a number of times (at least ten) for each sub-network to allow a variety of random initial weights to be used. For identical sub-nets approximately half of the training runs converged to solutions which gave prediction errors of similar

magnitudes for both training and testing sets. It was therefore assumed that the net with the lowest prediction error for the testing set was a satisfactory approximation to an optimal solution.

Once training was completed the selected network weights were scaled using the weight scaling equations given in Section 4.3.4 to permit their use with unscaled plant data. This ensures that the method of scaling the training data is not required to be known when using the networks.

It must be remembered that the following training, testing and validation results are for long-range predictions. A network performing long-range prediction is acting as a pure simulation model whose output is based on the inputs only. In others words it does not use any of the actual plant outputs to form its output (other than as initial conditions) and so is required to generate the past outputs itself over the entire length of the data record. If however the network was configured as a predictor whereby it predicts only n steps ahead then the network has available to it all the actual plant outputs up to the current time.

Table 6-7 summarises the time horizon and number of time steps ahead the networks are predicting for each data set. During training the sub-networks are attempting to perform predictions over 750 steps ahead and are tested with a 280-step-ahead prediction.

Table 6-7: Prediction horizon for each data set

	Prediction horizon	
Data set	Time (min:sec)	Time steps (10 sec)
Training	126:30	759
Testing	46:40	280
Validation 1	93:30	561
Validation 2	53:30	321

Table 6-8 shows the training and testing mean-squared error (MSE) associated with each of the sub-networks. These results are for the networks which had the best prediction error for the test data. Generally for each variable there are a number of networks with training and generalisation errors of similar magnitude.

The MSE results are calculated for output data that has been scaled between 0 and 1 to standardise the values so that a comparison can be made between the different variables.

Table 6-8: Long-range prediction errors for the individual sub-nets

	Mean-Squared Error ($\times 10^{-3}$)	
Sub-network	Training Data	Testing Data
T_2	1.50	3.80
L_2	9.73	28.43
Q_2	1.82	4.68

The temperature sub-net gives the best prediction errors and the levels the worst. This is because the temperature in the effect is highly correlated with the temperatures in the other effects whereas those inputs into the level sub-net are not highly correlated with the level measurement. Figure 6-8 to 6-10 show the long-range prediction plots for the training data for each of the individual sub-nets.

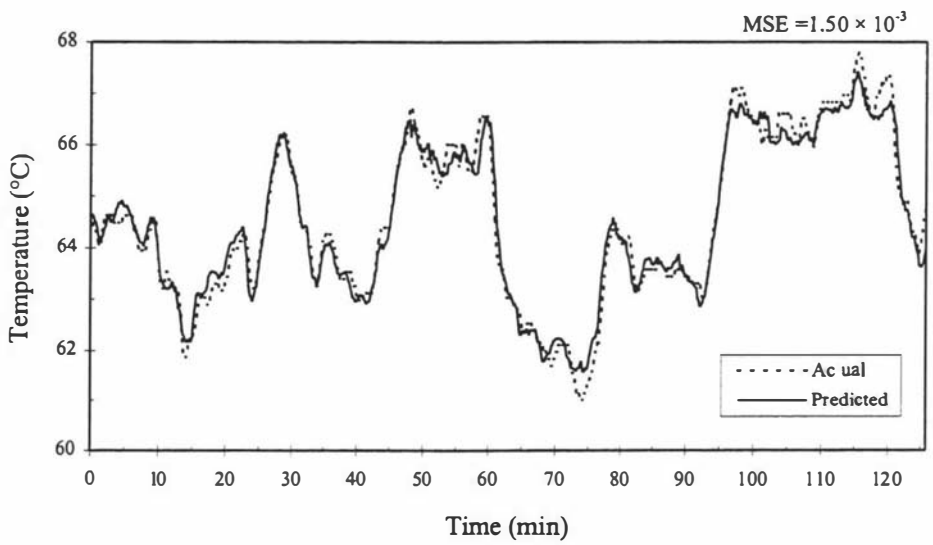


Figure 6-8: Prediction of training data for T_2 sub-net

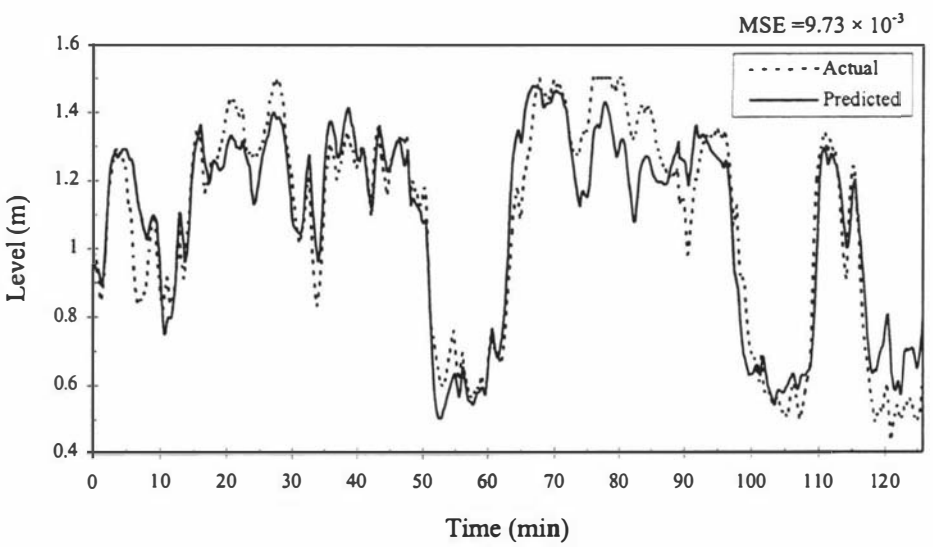


Figure 6-9: Prediction of training data for L_2 sub-net

In general the sub-nets do well in capturing the dynamics of the output variables as they attempt to learn to predict up to 750 steps ahead. The temperature predictions are not so good at the higher and lower extremes nor are the level predictions on occasions where there is some offset between the actual and predicted outputs.

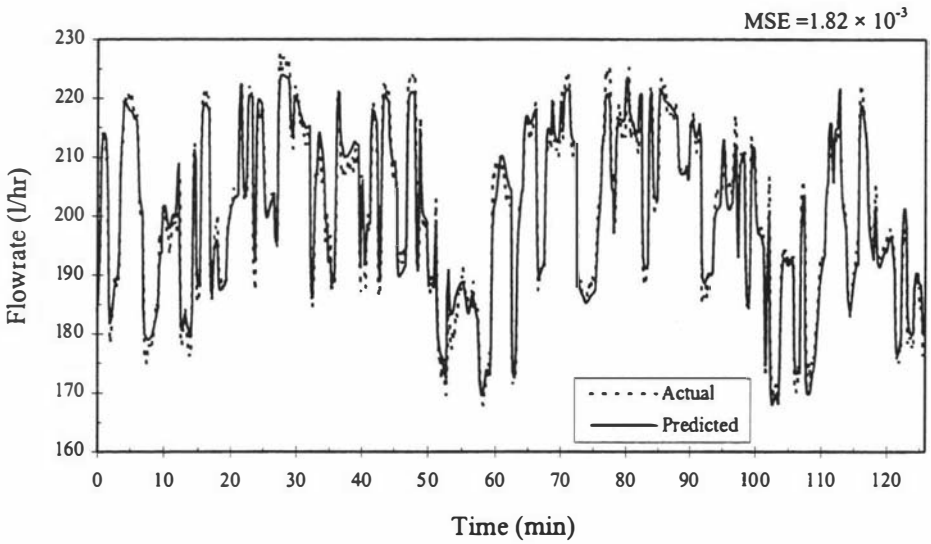


Figure 6-10: Prediction of training data for Q_2 sub-net

6.4.9 Sub-network validation

The sub-networks were validated using both correlation-based tests and cross-validation using the validation data sets.

Correlation-based validation tests

The correlation tests, outlined in Section 4.3.6, are used to assess the adequacy of the model fit by testing the correlation between combinations of the model inputs and the prediction residuals. For nonlinear models there are five tests which are used which test the randomness of the residuals and whether they are independent of the input variables.

The sub-nets described in the previous section were examined with the five correlation tests. The model fit can be considered satisfactory if the model complies with each test within the 95% confidence limits. Figure 6-11 shows typical plots of the correlations obtained for the sub-networks when performing 1-step-ahead predictions. This clearly shows that both the autocorrelation and cross-correlations are violating the confidence limits (dotted lines) for test 1, test 4 and test 5. This indicates that the residuals are not random and that they are correlated with the inputs.

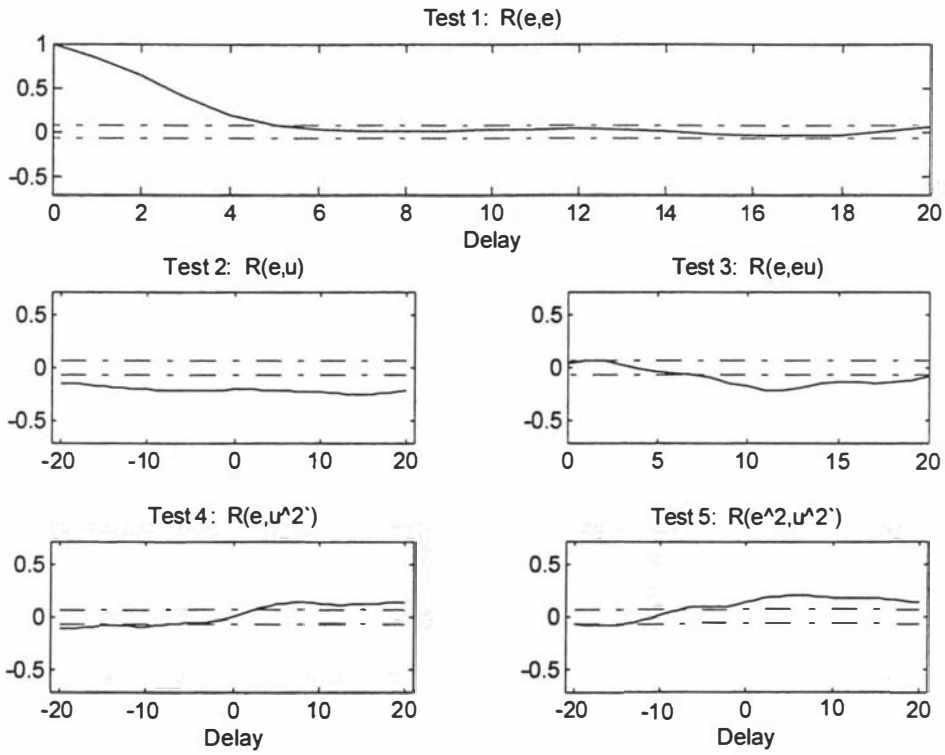


Figure 6-11: Correlation-based validation tests for LMTT-trained network

For a comparison three sub-networks with identical structures as the previous sub-networks were trained but this time using a standard feedforward LM training method (no recurrent connections). The same correlation tests were carried out for the residuals of these networks. The plots in Figure 6-12 were typical of the results achieved. Here it is clear that the networks performed satisfactorily in fitting the data.

This comparison indicates that the input variables used for the sub-networks are sufficient to model the data. However the recurrent LMTT training methodology does not obtain as good a model fit as would be desired and it appears that using a feedforward training method would suffice.

However in implementing the networks for long-range prediction of the data sets, the networks trained with the feedforward method do not perform as well as the LMTT trained networks.

This suggests that the LMTT training finds it difficult to obtain an adequate solution, however networks trained with a feedforward method are less suited to being implemented as externally-recurrent networks in order to perform *n*-step-ahead predictions.

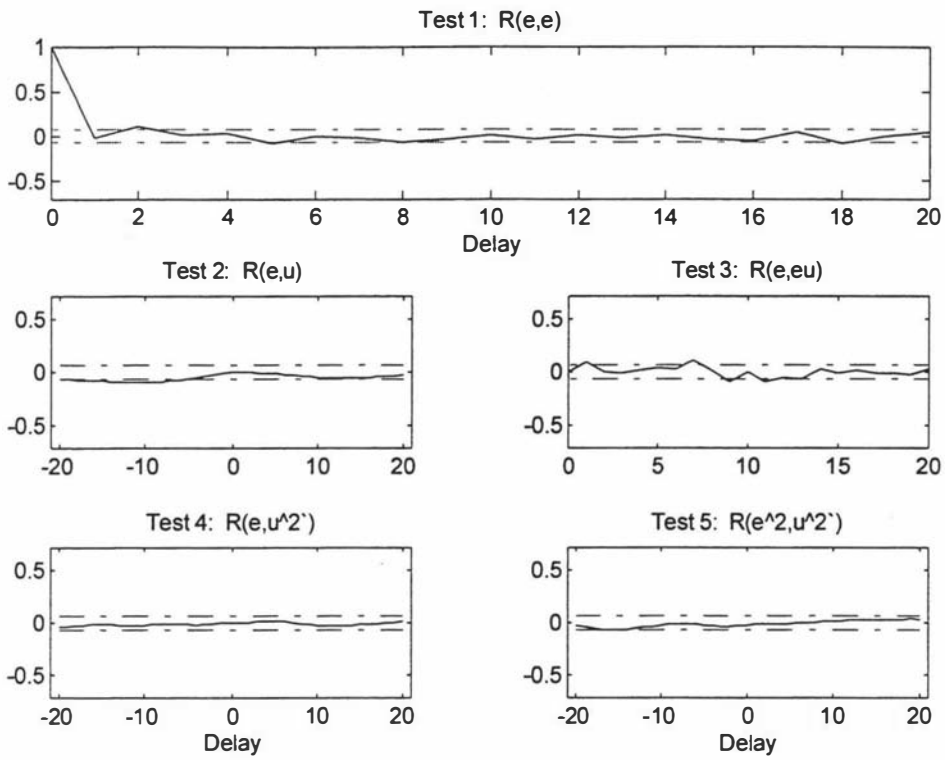


Figure 6-12: Correlation-based validation tests for feedforward-trained network

Cross-validation

Two sets of validation data are used to test the generalisation capabilities of the networks. The first half of one data set (validation 1 data) is made up of the testing data set with the remaining half being data that was previously 'unseen' by the networks. The second validation data set (validation 2 data) is a smaller set of data from a completely separate trial. Table 6-9 lists the long-range prediction errors of the individual sub-nets for the two validation data sets.

Table 6-9: Long-range prediction errors for individual sub-nets

	Mean-Squared Error ($\times 10^{-3}$)	
Sub-network	Validation 1 Data	Validation 2 Data
T_2	4.50	2.69
L_2	32.11	18.20
Q_2	9.71	3.77

The poor performance for Q_2 when predicting the Validation 1 data is mainly due to the flowrate data falling below 160 l/hr a number of times whereas in the training data the minimum flowrate is approximately 170 l/hr. So the sub-network was being asked to extrapolate beyond its training limits (see Figure 6-17). This situation is undesirable since the training data should cover the full range of the input space. However it is

difficult to get a single set of data that does this for all the variables. Overall the training data covers the widest range of the input space for the majority of the variables, however Q_2 is not one of them.

Comparing the sub-net results with those of the ARX models (Table 6-3) one can observe that compared with the ARX models:

- the T_2 results are similar,
- the L_2 results are better and
- the Q_2 results are generally similar or better.

The long-range predictions of the validation data for each variable are plotted below. The temperature predictions (Figure 6-13 and Figure 6-14) are quite good although as with the training data predictions the model output misses some of the peaks and troughs present in the actual data.

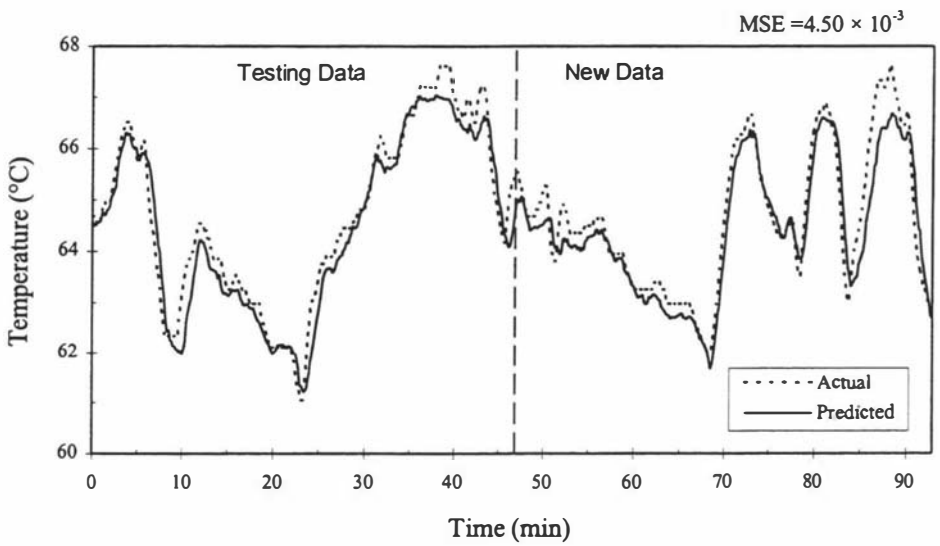


Figure 6-13: Prediction of validation 1 data for T_2 sub-net

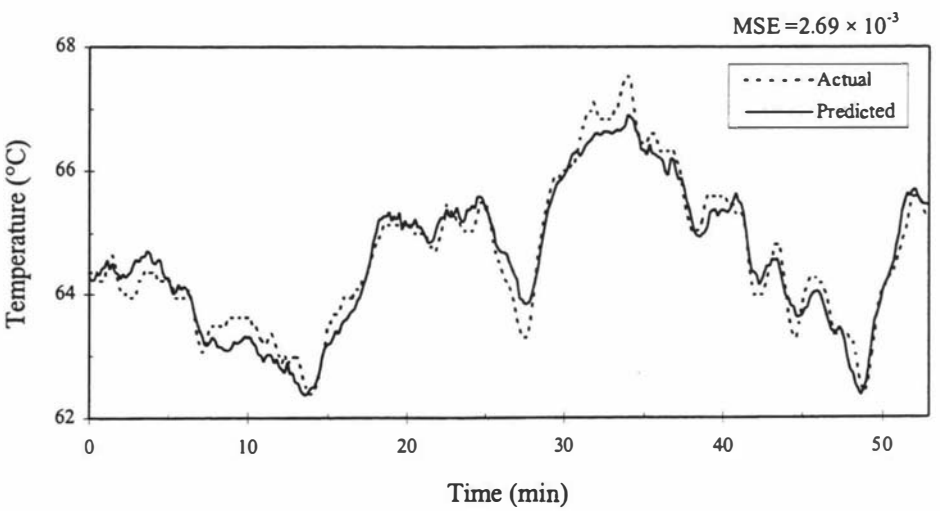


Figure 6-14: Prediction of validation 2 data for T_2 sub-net. (MSE = 2.69×10^{-3})

The level predictions (Figure 6-15 and Figure 6-16) experience a positive offset to the actual data, although generally the shape information is captured by the model. At low values of the level the prediction suffers by extrapolating beyond the training range.

The offset observed between the concentrate level estimates and the actual data are difficult to avoid. As has been previously discussed (Chapter 3), due to the integrating nature of the level system, errors tend to accumulate in the model. The resulting estimates exhibit the correct shape of the data but not necessarily the correct magnitudes.

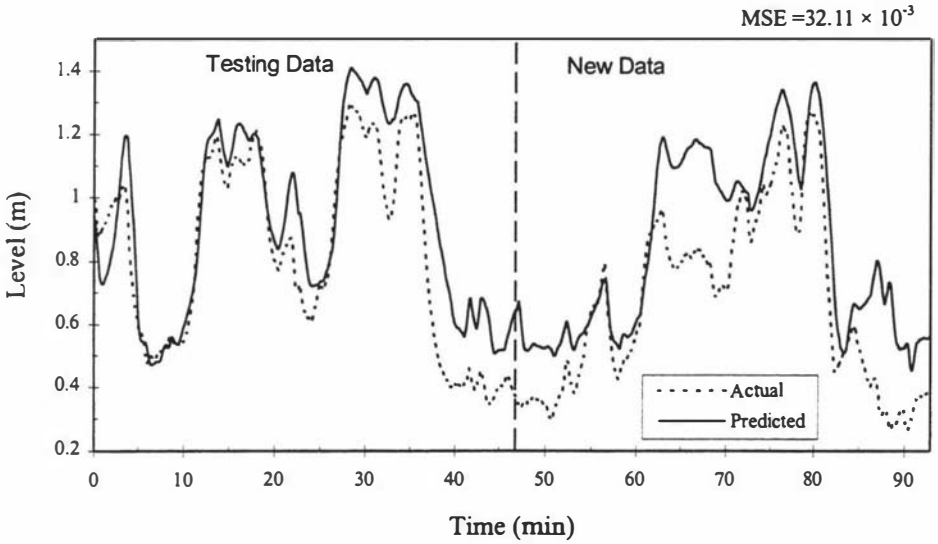


Figure 6-15: Prediction of validation 1 data for L_2 sub-net

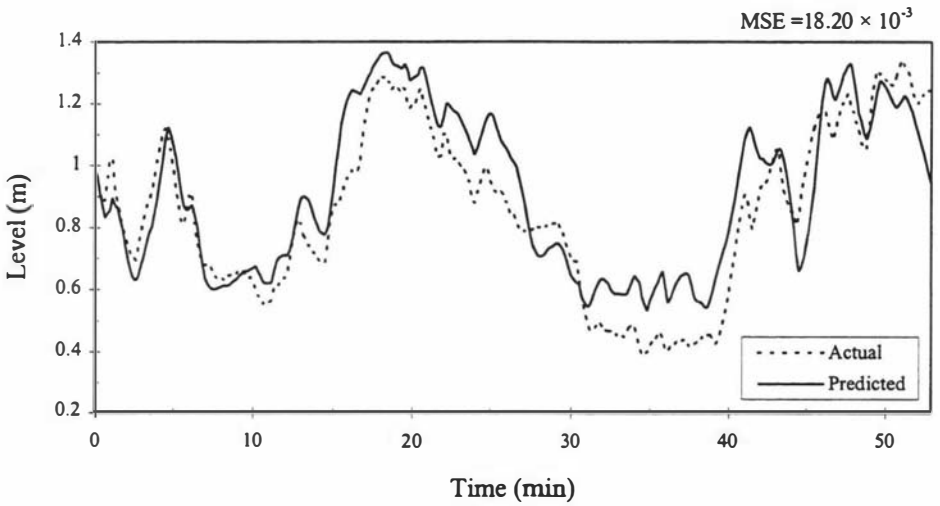


Figure 6-16: Prediction of validation 2 data for L_2 sub-net

The flowrate predictions (Figure 6-17 and Figure 6-18) again suffer due to extrapolation errors for the validation 1 data, however the model does well with the second data set which is more in the range of the training data.

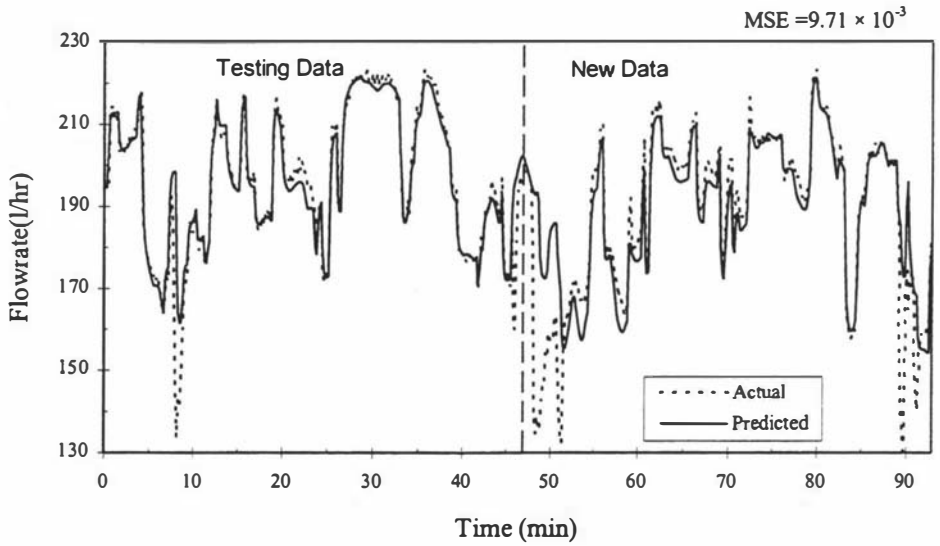


Figure 6-17: Prediction of validation 1 data for Q_2 sub-net

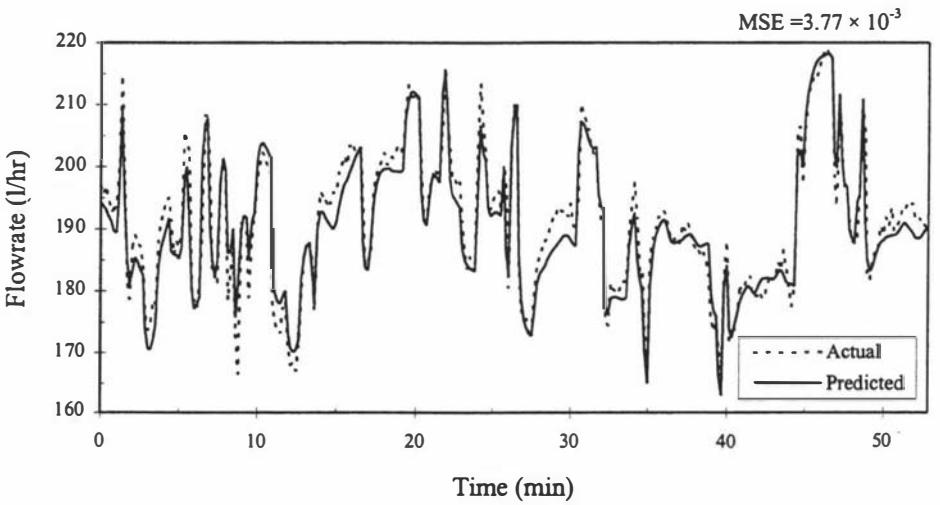


Figure 6-18: Prediction of validation 2 data for Q_2 sub-net

6.4.10 Fully connected sub-networks

As a comparison, another set of sub-networks were developed which did not contain localised connections, but had a fully-connected structure. The number of hidden-layer neurons used in these networks was determined so that the number of connections was approximately equal to those in the previously developed sub-networks. Both the T_2-net and L_2-net have 59 weights and biases. For the fully-connected sub-networks four hidden layer neurons were used to give a total 57 weights and biases. Since the Q_2-net is already fully connected no alternative sub-network was developed for comparison.

The structure of the alternative networks are given in Table 6-10. These sub-networks are effectively a fully-connected representation of the previously developed locally-connected sub-nets.

Table 6-10: The structures of the fully-connected sub-networks

Output	S_i-S_h-S_o	N_w
T_2	12-4-1	57
L_2	12-4-1	57

These networks were trained in a similar manner to the previous locally connected networks. The networks with the best testing errors were selected for comparison with the previous sub-nets. Table 6-11 lists the predictions errors obtained for the best fully-connected sub-networks for each of the data sets. Remember that the Validation 1 data set contains the testing data.

Table 6-11: Long-range prediction errors for individual fully-connected sub-nets (four hidden-layer neurons)

	Mean-Squared Error ($\times 10^{-3}$)		
Sub-network	Training Data	Validation 1 Data	Validation 2 Data
T_2	2.59	5.12	4.03
L_2	11.79	36.94	33.45

The performance of training for the fully-connected networks was quite different than with the previous sub-nets. The training was less likely to converge to a satisfactory solution and the training rarely lasted for longer than 20 epochs. About three quarters of the training runs gave MSE values many times larger than those obtained with the locally-connected networks. These networks seemed to get stuck in local minima causing the LM optimisation to stop at poor solutions. Both fully-connected network predictions were worse than those of the locally-connected sub-nets. Figure 6-19 and Figure 6-20 show the long-range prediction plots for the validation 2 data set.

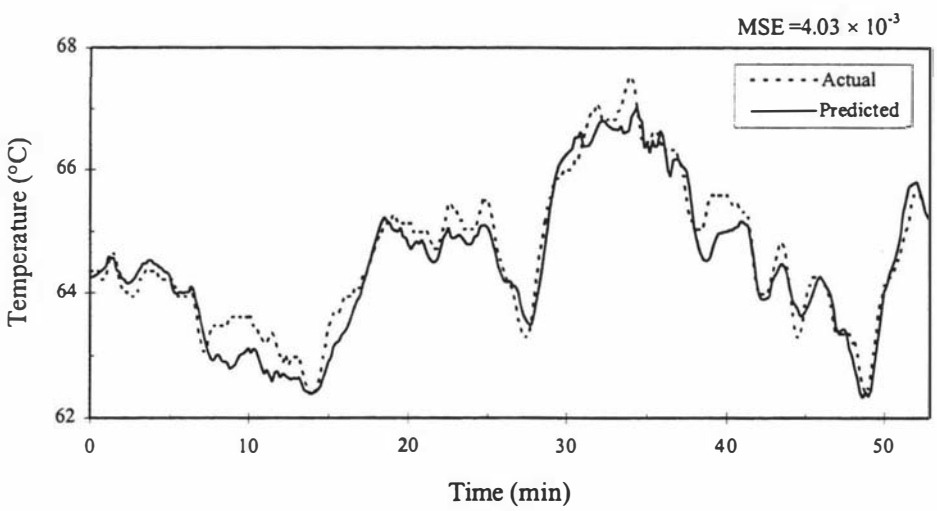


Figure 6-19: Prediction of validation 2 data for fully-connected T_2 sub-net

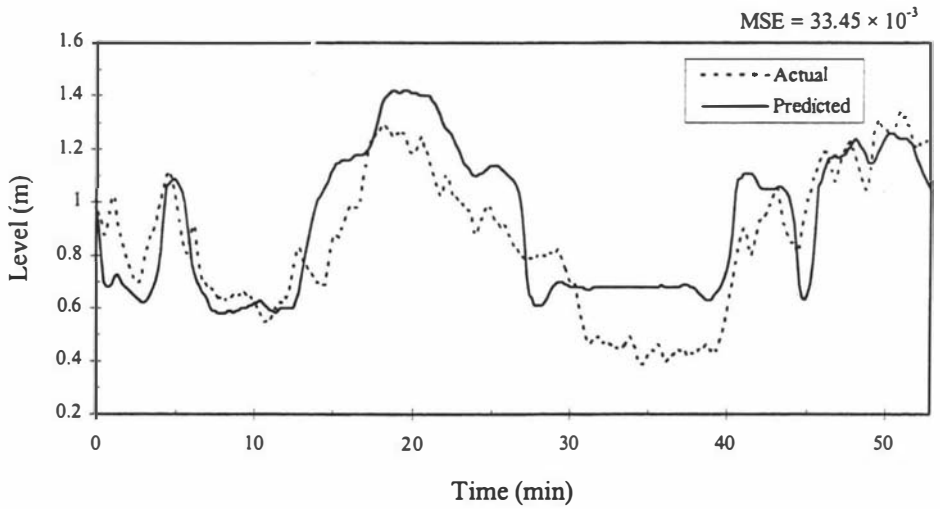


Figure 6-20: Prediction of validation 2 data for fully-connected L_2 sub-net

The number of hidden-layer neurons was increased to eight to investigate whether this would improve the training and generalisation performance of the networks. Eight hidden neurons gives 113 network parameters. As can been seen in Table 6-12 the prediction errors for the best networks are not better than for the networks with four hidden neurons; in fact the T_2 predictions are worse.

As well as better prediction capabilities locally-connected sub-networks have an additional advantage in that they can be readily analysed to determine the role each neuron is playing in producing the network output. By inspecting some of the outputs

Table 6-12: Long-range prediction errors for fully-connected sub-nets (eight hidden-layer neurons)

	Mean-Squared Error ($\times 10^{-3}$)		
Sub-network	Training Data	Validation 1 Data	Validation 2 Data
T_2	3.55	6.73	7.67
L_2	12.02	35.17	21.97

from the hidden layers of the networks it was apparent that the two neurons receiving inputs from the same variable would characterise a different region of the variable's input space. For example one neuron might model the variable's mean operating region and the other the occurrence of variations from this region. The outputs of each hidden neuron and how they form the network output can be readily analysed for each network to gain a better understanding of the functionality of the network.

Some examples from the T_2 locally-connected network are shown in the following figures.

In Figure 6-21 the outputs of the first two hidden-layer neurons are plotted. These neurons have the T_1 data as inputs (which is plotted in Figure 6-22). It appears that

neuron 1 is characterising the small variations in the temperature input while the second neuron has modelled the larger fluctuations. By comparing Figure 6-21 with Figure 6-22 it can be seen that the output of neuron 2 is has an inverse relationship with the temperature input.

A very similar characteristic is found with the third and fourth hidden neurons and the T_3 input data.

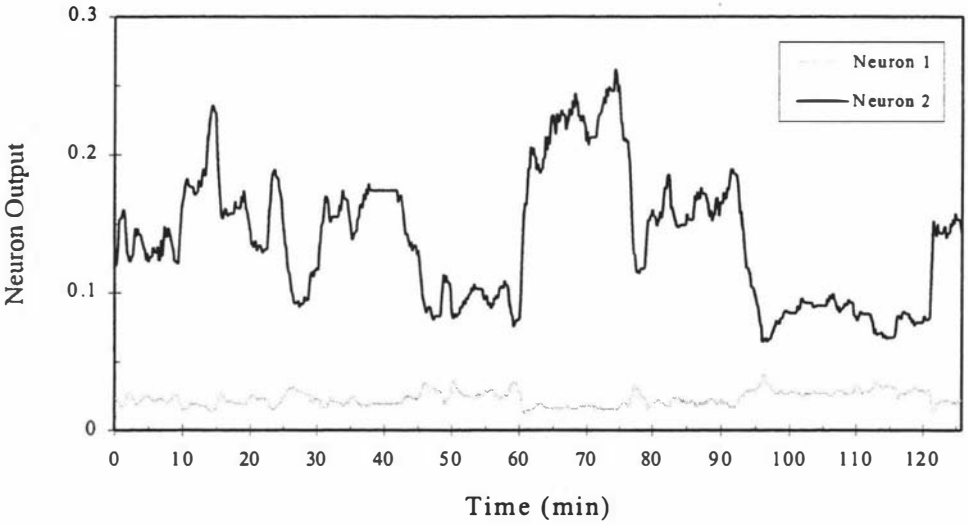


Figure 6-21: Outputs of two hidden-layer neurons for T_2-net

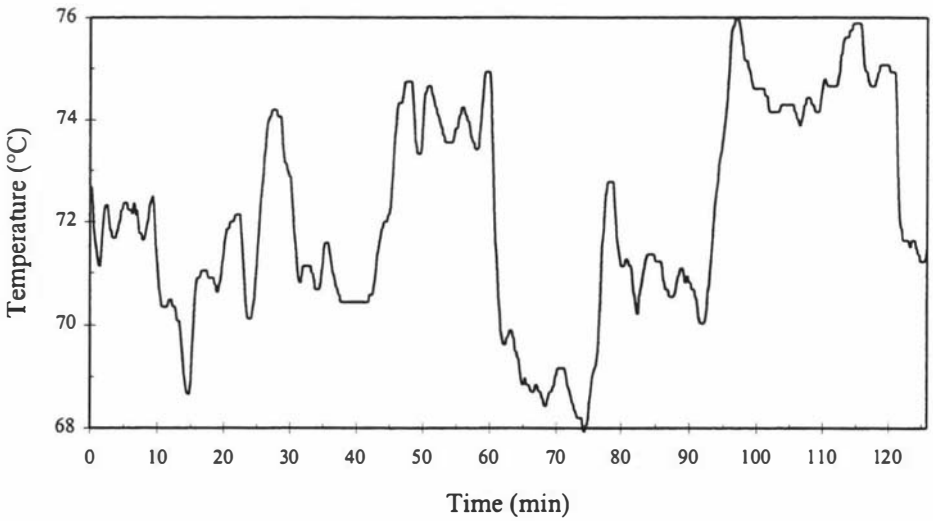


Figure 6-22: T_1 input to hidden neurons 1 and 2

The outputs of the time-based hidden neurons (neurons 9, 10 and 11) can also be viewed in a similar manner (Figure 6-23). The inputs of neuron 9 are all the input variables at time $t-1$, neuron 10 connects with the variables at time $t-2$ and neuron 11 at time $t-4$. From this plot it appears that the inputs at $t-2$ produce a fairly constant output from neuron 10 and the variations in the output, T_2, are modelled by the inputs at $t-1$ and $t-4$.

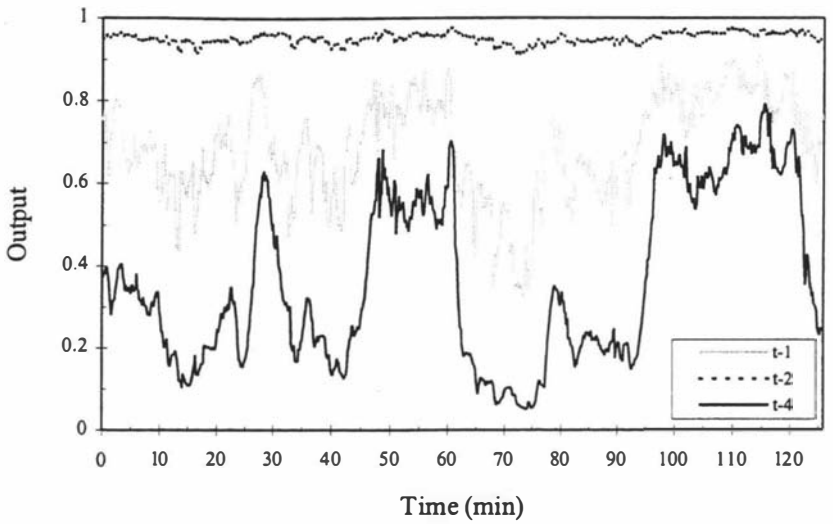


Figure 6-23: Outputs from the three time-input hidden neurons

6.4.11 Summary of sub-model results

From examination of the previous result tables and plots it is obvious that the temperature and flowrate models performed better than the level models.

The temperature and flowrate predictions from both the neural network and ARX models were reasonably similar in accuracy whereas the level predictions from the ARX model were not so good. This indicates that the temperature and flowrate data for the Eff2 system is not highly nonlinear.

The comparison of the different models is clearly shown in the Figure 6-24 and Figure 6-25. These plots compare the ARX models, the locally-connect networks and the fully-connected networks with 4 hidden neurons.

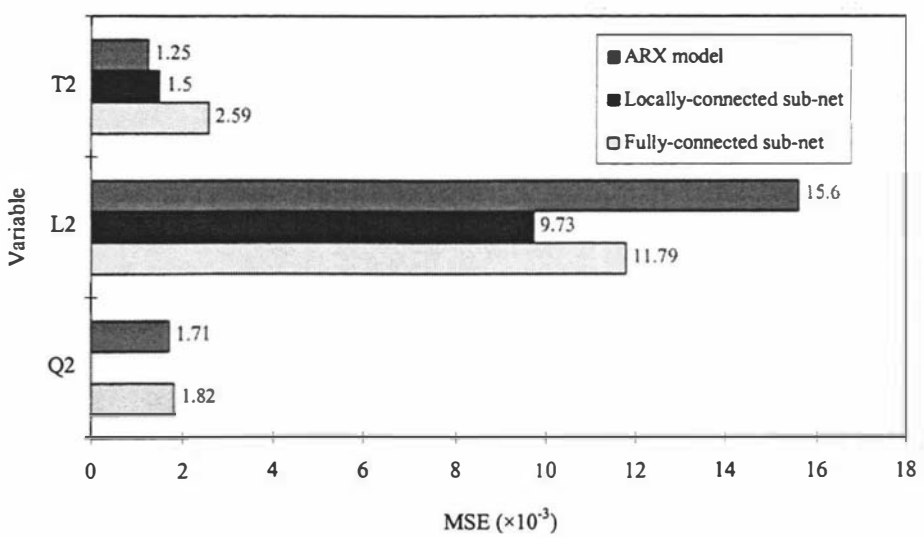


Figure 6-24: Prediction errors for training data set

It is obvious from these plots, that the L_2 models have most difficulty while theT_2 and Q_2 models have a similar performance. The ARX L_2 model is the poorest level model, while the T_2 ARX model is the best temperature model. The locally connected network performs significantly better with the level than the other models.

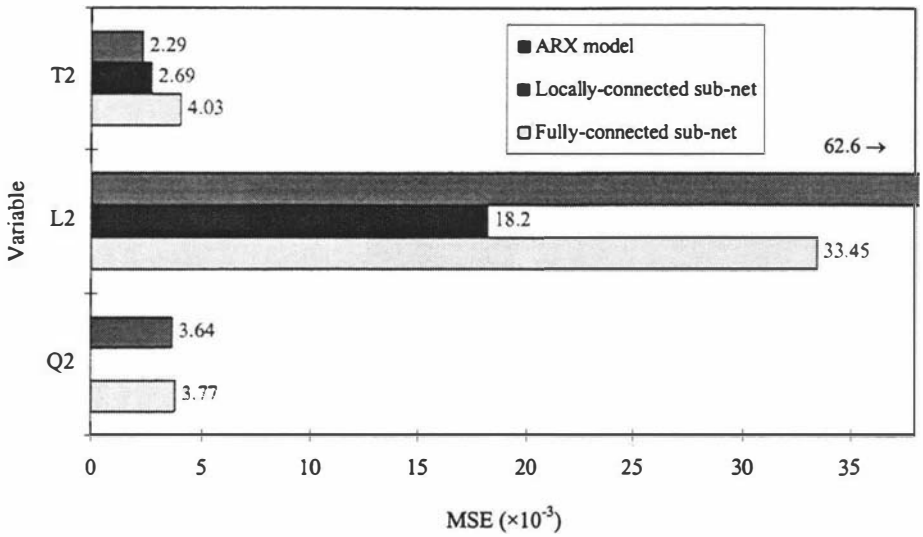


Figure 6-25: Prediction errors for validation 2 data set

We can also compare the different models through the use of Akaike's Final Prediction Error (FPE) measure (equation (4.43)). The FPE criterion considers the size of the models (number of parameters, N_w) along with their prediction errors. The model which minimises the FPE has the most accurate and parsimonious structure. For models with a similar prediction accuracy the FPE criterion penalises the model which has the greater number of parameters.

Table 6-13 summarises the number of parameters used within each individual Eff2 sub-model.

Table 6-13: Number of parameters (N_w) for Eff2 sub-models

Output	ARX models	Locally-connected sub-nets	Fully-connected sub-nets
T_2	14	59	57
L_2	9	59	57
Q_2	8	–	19

The FPE results for the ARX sub-models and both the locally-connected and fully-connected (4 hidden neurons) sub-networks for the training and validation 2 data are shown in Figure 6-26 and Figure 6-27. Note that no locally-connected flow network was created as it is only a first order model.

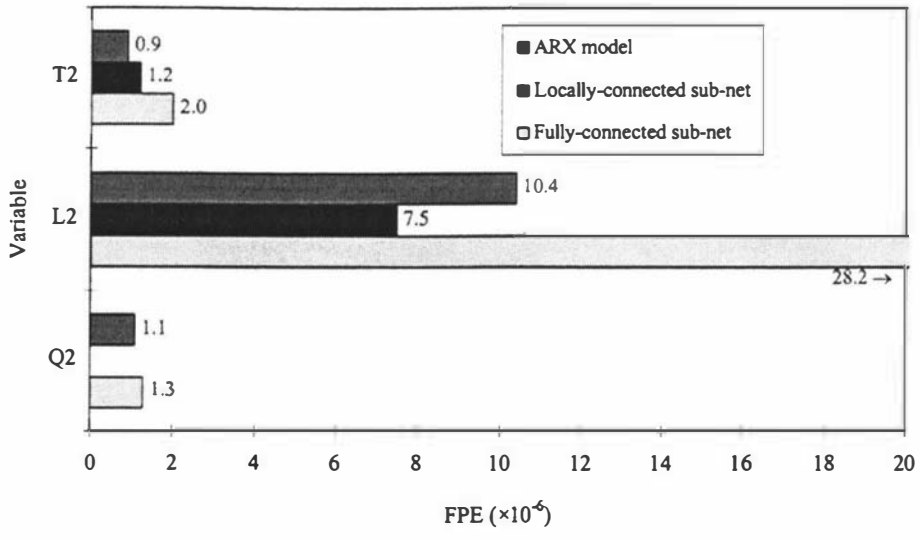


Figure 6-26: FPE results for training data set

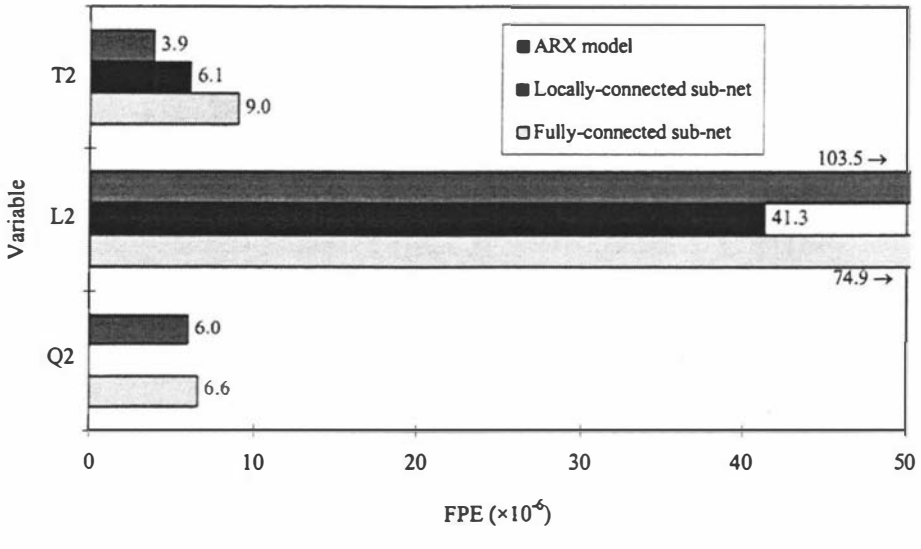


Figure 6-27: FPE results for validation 2 data set

Based on the FPE values the ARX model produced better results than the sub-networks for the temperature while the flowrate was about the same. However, for the level the locally-connected sub-net has the lowest FPE, despite there being significantly fewer parameters in the ARX model.

These results appear to indicate that the temperature and flowrate have an essentially linear characteristic whereas the level is more complex. For the temperature this can be explained by the high correlation between the Eff2 temperature and the temperatures of the surrounding effects. For the flowrate, the output is highly correlated to the pump speed.

In the case of the level, the cross-sectional area of the evaporator changes quite dramatically with height. At the higher points the liquid level includes the lower portion of the separation vessel as well as the base of the effect. These geometrical properties together create a very nonlinear relationship between the flowrates and the change in level. Additionally it is suspected that there may be unobservable modes operating within the level system. One possibility is that pressure fluctuations caused by the vapour flow from the separator may be influencing the level as well as the flowrates. Since the vapour flow is not measured its influence on the levels cannot be fully determined. This pressure influence may even cause the sensor to register a changing level when in fact it may not have changed.

6.4.12 Full system model

A modular model of the Eff2 system was created by combining the three locally connected sub-networks. The combined Eff2 model can be considered as a two-layer network with an extra input layer as illustrated in Figure 6-28. The initial input layer collects the four input variables (T_1, T_3, Q_1 and N_2) and the past output variables (T_2, L_2, Q_2) and passes the data through tapped delay lines (and filters for the flowrates) and feeds into the next input layer. The connections between these two input layers are not adaptive and have a weighting of unity. These connections relate to the selection of the input variables for each sub-net as discussed in Section 6.4.3 and hence this layer is not fully connected. The second layer of input nodes relate to the input nodes for the individual sub-nets and hence there are 28 (ΣS_i) nodes in this layer. The next layers relate to the hidden and output layers of the individual sub-nets hence there are 25 (ΣS_h) and 3 (ΣS_o) neurons in these layers. The total number of adjustable parameters for the

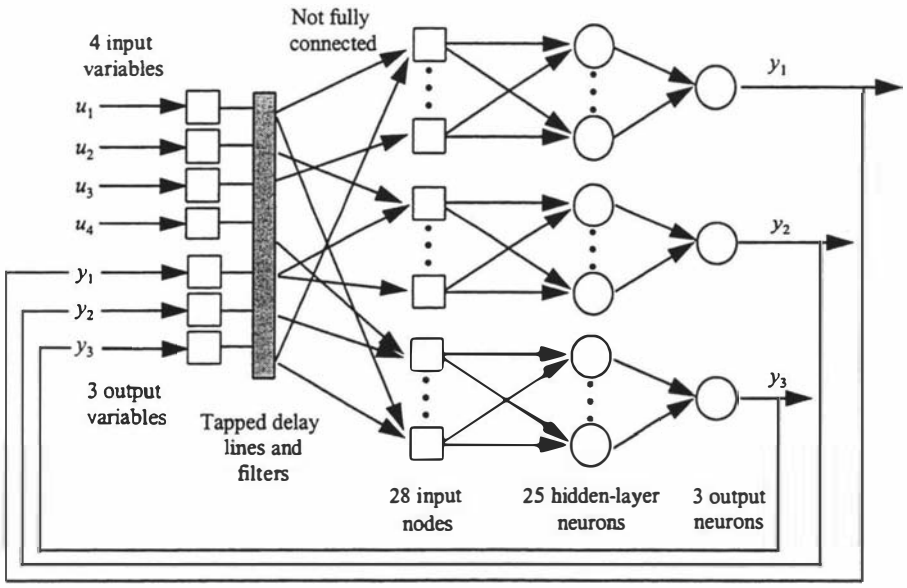


Figure 6-28: General structure of the Eff2 modular, neural model.

combined model is equal to 137 ($\Sigma N_w = 59 + 59 + 19$).

Figure 6-29 illustrates how the combined Eff2 model is implemented within SIMULINK. This block diagram shows clearly the feedback links within the model and how the outputs of some of the sub-networks become inputs for others.

As an example, the block diagram of the T_2 sub-network is shown in Figure 6-30. The blocks on the input lines prior to the 'Mux' block implement the time delays. Here D_u and D_y are equal to [1 2 4].

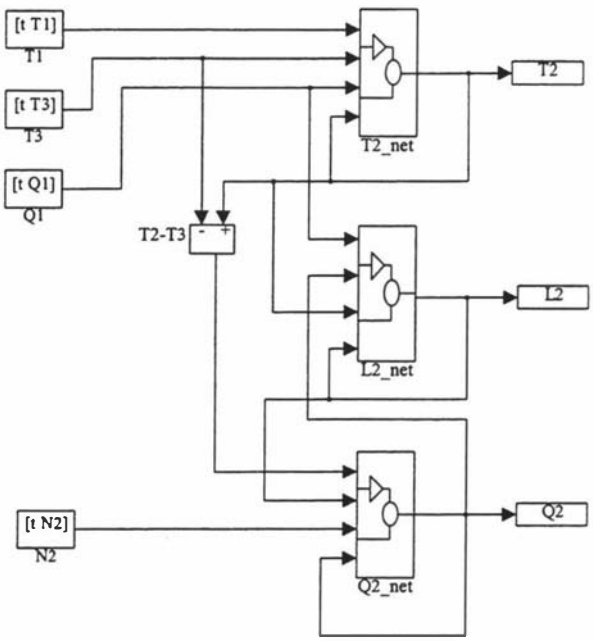


Figure 6-29: SIMULINK block diagram of the Eff2 neural model.

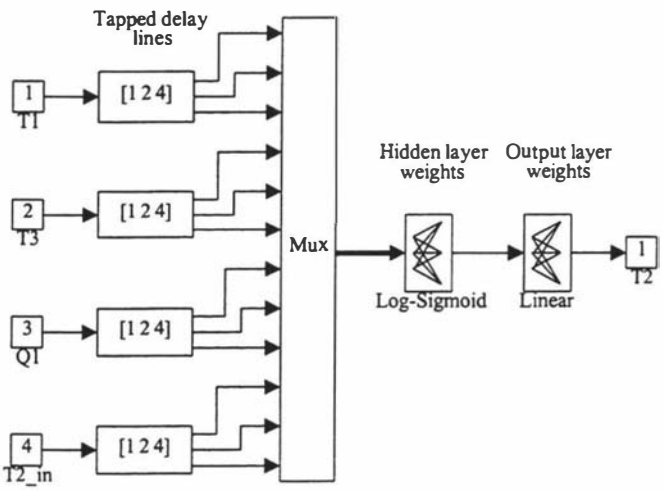


Figure 6-30: SIMULINK block diagram of T_2-net

The combined Eff2 model was used to perform long-range predictions of the plant data. The prediction errors are tabulated in Table 6-14. The results for T_2 are exactly the same as for the individual T_2 sub-net since in the combined model the T_2 network receives no inputs from either of the other sub-nets. As would be expected though the prediction errors for L_2 and Q_2 are higher than those for the individual sub-nets due to the feedback of prediction errors within the model.

Table 6-14: Long-range prediction errors for modular Eff2 network

	Mean-Squared Error ($\times 10^{-3}$)		
Output	Training Data	Validation 1 Data	Validation 2 Data
T_2	1.50	4.50	2.69
L_2	10.20	35.40	20.10
Q_2	4.31	14.90	7.48

The results from estimating the training data have been included for comparison with the validation data sets. Once again the poor performance when predicting the validation 1 data is due somewhat to the validation data extending below the range of the training data. The poor performance of the flowrate prediction causes the level network predictions to deteriorate. However despite this the full model still performs reasonably well. The prediction plots of the combined model for the validation 2 data are given below. The plot of the T_2 response is exactly as is shown in Figure 6-14.

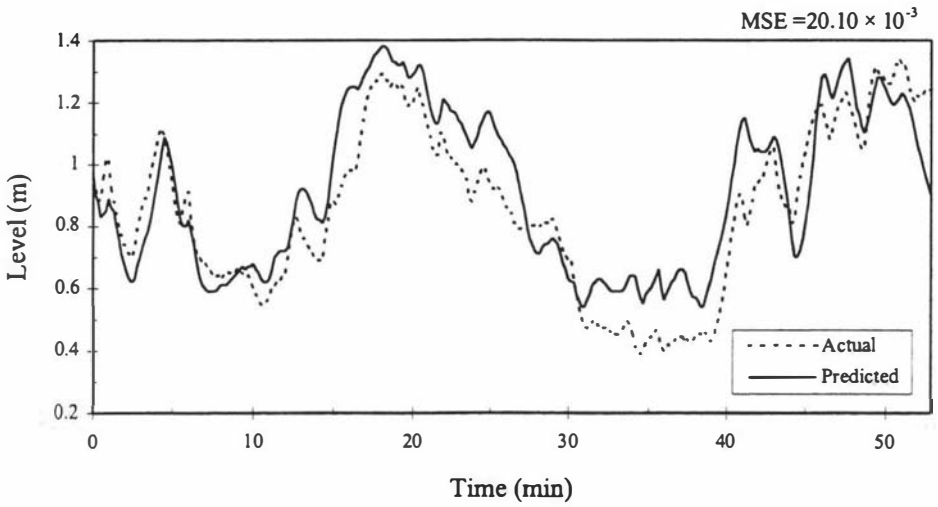


Figure 6-31: Prediction of L_2 from combined Eff2 model using validation 2 data

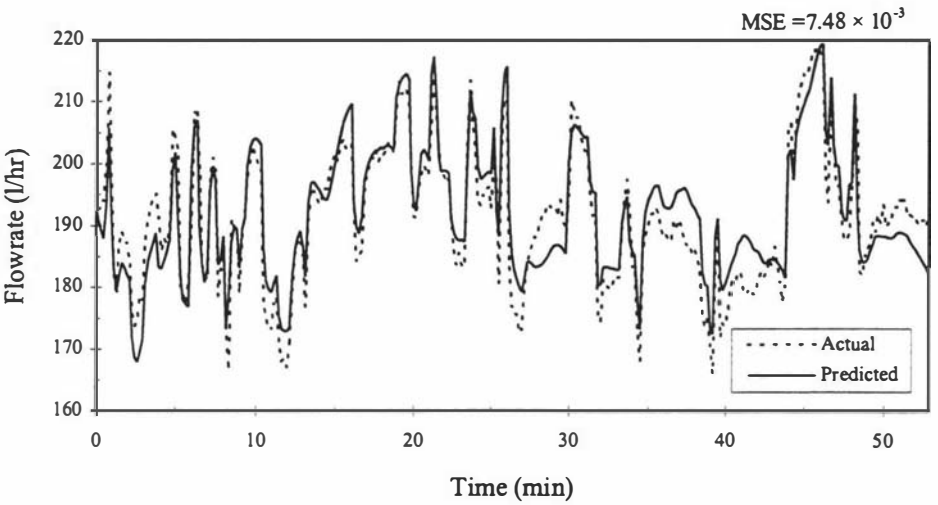


Figure 6-32: Prediction of Q_2 from combined Eff2 model using validation 2 data

6.4.13 Attempts to improve the level sub-network

It was becoming clear that the neural network was struggling to satisfactorily model the concentrate level system. Although the prediction plot (Figure 6-31) looks reasonably good it must be remembered that this is simply with the L_2-net implemented within the Eff2 model. Once it is embedded within a larger model for the whole evaporator system then its performance is likely to significantly degrade further as errors will be more

prevalent. It had already been determined that filtering the flowrates significantly improved the training of the level network.

Some alternative structures were trialed to see if an improved L_2-net could be developed. The two most promising choices were;

i) Using Q_1–Q_2 as an input, rather having the flows separate and

ii) Predicting the change in level (ΔL_2) using Q_1 and Q_2 as inputs.

For the first option, the network functional relationship of the network was

$$L_2 = f(Q_1^f - Q_2^f, T_2, L_2) \tag{6.13}$$

Both flowrates were individually filtered. Training a network with these inputs did not produce results as good as had previously been obtained. The training errors were similar but the testing set predictions were poor.

The training for the second alternative involved feeding back a past value of L_2 to add to the network output (Figure 6-33).

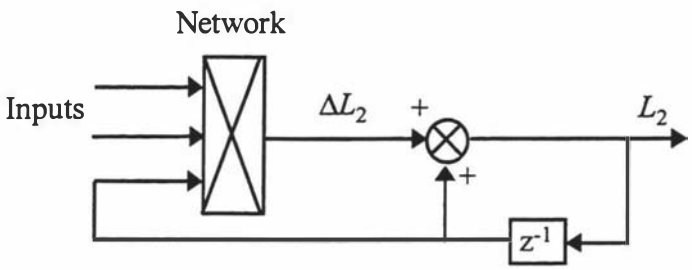


Figure 6-33: Neural network to predict ΔL_2

The network was trained using the L_2 data as the target output. The time delay was initialised with the correct L_2 value. During training, the network output was added to the previous L_2 value and the resulting output compared with the target. Hence over time the network was adapted to model the change in level.

Although this seems like an appropriate solution it unfortunately failed to produce satisfactory results. The training was generally unable to converge to a reasonable error minimum. Models implemented in this manner are often prone to drift and so even if the training was successful it may still have caused problems when combined with the other sub-nets.

6.5 Alternative models

To provide comparisons to the combined modular model of the previous section, three other models were developed, these were:

1. A fully-connected, two-layer, externally-recurrent neural network which is not modular and predicts all three outputs using all the relevant inputs,
2. A neural network as in 1) above but with a locally-connected structure similar to the locally-connected sub-nets developed previously, and
3. A linear ARX model which is of the same structure as the modular neural network model but made up of the sub-models of Section 6.4.5.

The comparison with a single large non-modular neural network provides a means of determining whether there is any benefit in using a modular modelling approach.

A linear ARX model is developed in order that the modular neural network model can be compared with a conventional linear model.

6.5.1 Single fully-connected neural network model

In order to properly compare the performance of a single network model, the inputs used are required to be equivalent to the modular model. The single network model of the Eff2 system has three output neurons, one each to predict T_2, L_2 and Q_2. The inputs used for the network are the same four as used for the combined modular model of the previous section. This gives the following functional relationship for the network.

$$[T_2, L_2, Q_2] = f(T_1, T_3, Q_1^f, N_2, T_2, L_2, Q_2) \tag{6.14}$$

Note that the feed flowrate, Q_1 is filtered using an exponential filter as was done previously. The delay vectors for the inputs and past outputs were set to be similar to those used for the modular model so both D_u and D_y were made to be [1 2 4]. However in order to keep the network similar to the modular model the pump speed (N_2) had no delay applied (ie. $D_u = 0$).

This gave 19 input nodes for the single network model. This network requires six hidden-layer neurons in order to have approximately the same number of weights and biases as the 137 in the modular model. The hidden nodes were also increased to 10, 15 and 20 to investigate whether this improved the training results.

As would be expected it was much more difficult to achieve satisfactory convergence during training than for the sub-networks. Most of the training runs did not reach 20 epochs before the error gradient or step size became too small and the LM algorithm

terminated. The processing time required for each epoch was considerably longer than that for the sub-network training.

The training and testing results for the networks with ten or more hidden neurons were generally better than the networks with only six hidden neurons. The network with the best convergence and generalisation capabilities was chosen to compare with the modular model. This network had ten hidden neurons with a total of 233 weights and biases, significantly more than that used in the modular model. The long-range prediction errors for this network are given in Table 6-15 below. As can be seen the individual errors and the mean values are not as good as for the modular model results (cf. Table 6-14, Figure 6-24 and Figure 6-25).

Table 6-15: Long-range prediction errors for three-output Eff2 network (ten hidden-layer neurons)

	Mean-Squared Error ($\times 10^{-3}$)		
Output	Training Data	Validation 1 Data	Validation 2 Data
T_2	3.93	13.14	11.86
L_2	13.35	71.84	41.82
Q_2	43.41	39.54	85.34

6.5.2 Single neural model with localised connections

The next alternative model that was developed was a three-output network with localised connections. The inputs used were exactly the same as those used for the previous three-output networks. The networks were structured in a similar manner to that used for the locally-connected sub-networks, that is, the inputs for each variable were connected to two hidden neurons and the inputs at each time step were coupled to a single hidden neuron. The exception was the pump input (N_2) which was only connected to a single neuron. This structure gives 16 hidden neurons and a total of only 88 network parameters, less than that for the modular Eff2 model.

The training of these networks was similar to the previous three-output networks in that often the training terminated due to poor convergence. However the training runs that did do well generally produced better training and testing results than the fully-connected three-output networks. The network with the best testing performance was selected to compare with the modular network model. The results of the long-range predictions for this network are listed in Table 6-16.

These results are an improvement upon the those for the fully-connected three-output network but are not as good as the modular model apart from the T_2 prediction for the

validation 1 data which is better. The fact that the results have improved with the localised connections illustrates the advantages that can be obtained by utilising a simple pruning of the network connections.

Table 6-16: Prediction errors for locally-connected, 3-output Eff2 network

	Mean-Squared Error ($\times 10^{-3}$)		
Output	Training Data	Validation 1 Data	Validation 2 Data
T_2	2.03	3.81	4.87
L_2	13.98	56.26	34.61
Q_2	36.90	21.18	43.67

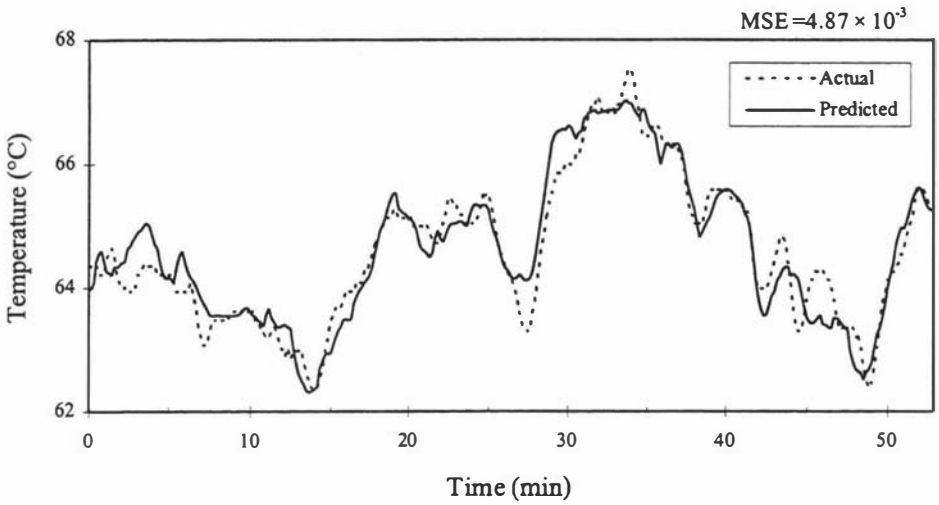


Figure 6-34: T_2 prediction of locally-connected, 3-output network using validation 2 data

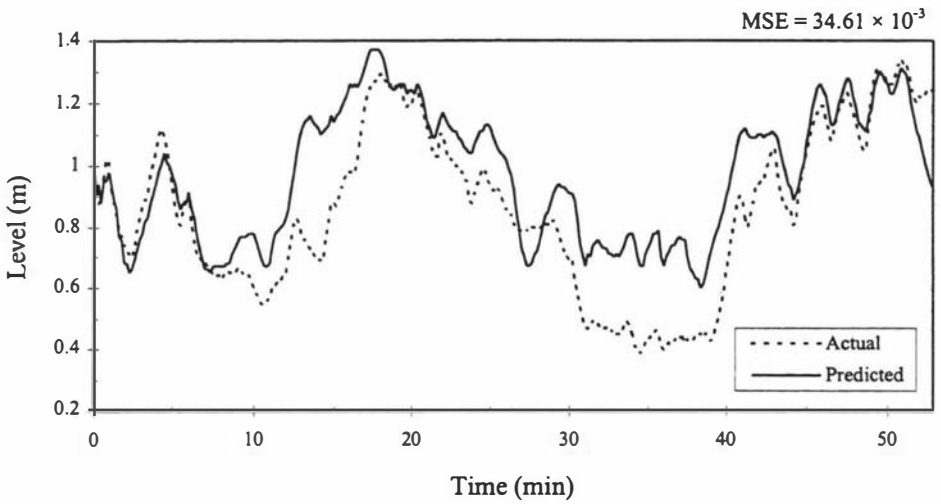


Figure 6-35: L_2 prediction of locally-connected, 3-output network using validation 2 data

The predictions for this network are shown in the following plots (Figure 6-34 to Figure 6-36). The temperature and level predictions are similar to the modular model however the flowrate predictions are unable to follow each rise and fall of the actual data (Figure 6-36). The flowrate output tends to follow an average value of the actual flowrate across the prediction horizon.

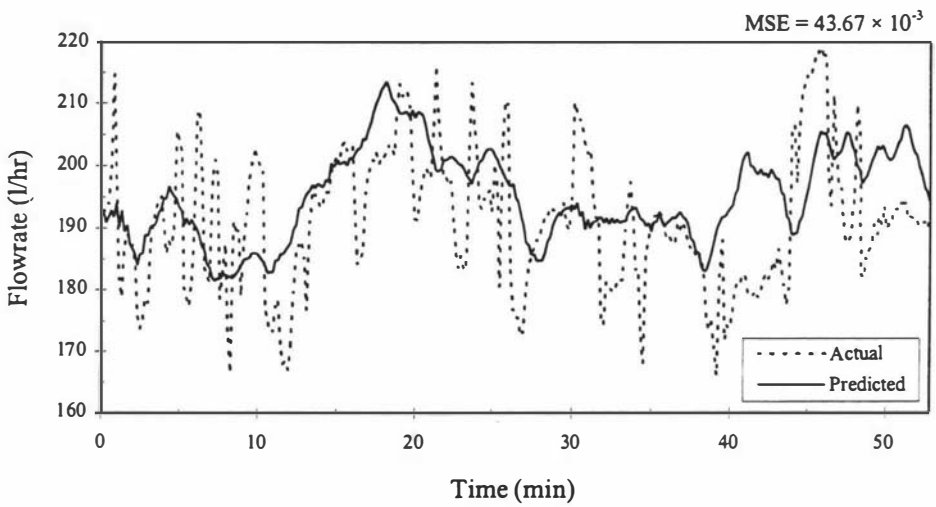


Figure 6-36: Q_2 prediction of locally-connected, 3-output network using validation 2 data

6.5.3 ARX model

The variable structure of the ARX model was equivalent to that of the modular neural network model. It was made up of the three sub-models identified in Section 6.4.5 which were inter-connected in an identical manner to the neural sub-nets. The resulting modular ARX model had a total of 31 parameters.

The modular ARX model was simulated to produce long-range predictions for each of the data sets and the MSE results are given in Table 6-17.

Table 6-17: Prediction errors for the Eff2 ARX model

	Mean-Squared Error ($\times 10^{-3}$)		
Output	Training Data	Validation 1 Data	Validation 2 Data
T_2	1.27	6.02	2.29
L_2	14.30	69.50	52.80
Q_2	5.61	19.90	15.90

The modular ARX model generally produces prediction results of a similar nature to the individual ARX sub-models (Table 6-3). Note that the temperature results are identical to those of the individual T_2 model.

The flowrate and level results are not as good as those of the modular neural network whereas the temperature errors are of a similar magnitude (cf. Table 6-14).

6.5.4 Summary of results for the Eff2 models

The combined Eff2 model predictions have a similar characteristic to that seen with the individual sub-models. The modular neural network model performed considerably better than the three-output networks. Of the three-output networks, the locally-connected network was the better, again showing that the localised structure formed by collecting the relevant inputs together aids in both training and generalisation results. It also greatly reduces the model parameters. The MSE results for the Eff2 models are plotted in Figure 6-37 and Figure 6-38. These bar charts clearly show that the 3-output

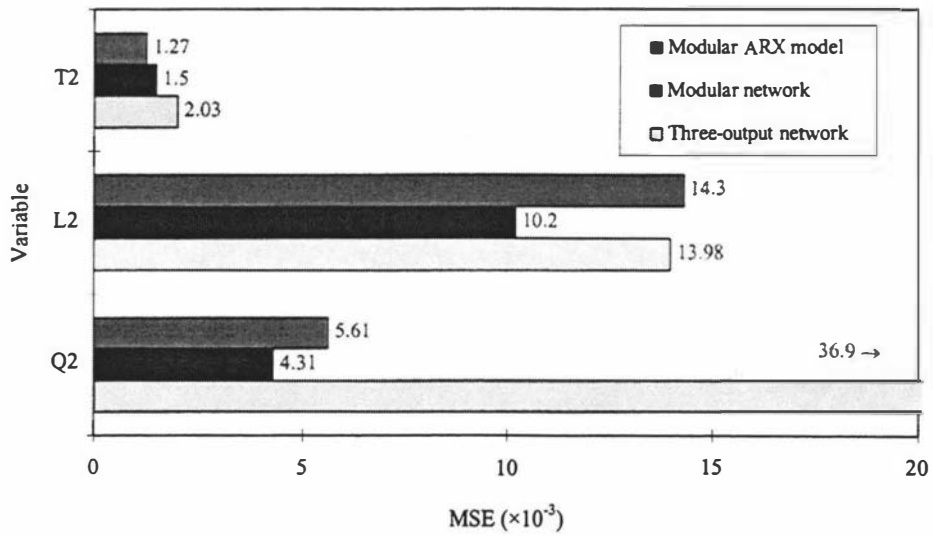


Figure 6-37: Training data prediction errors for Eff2 models

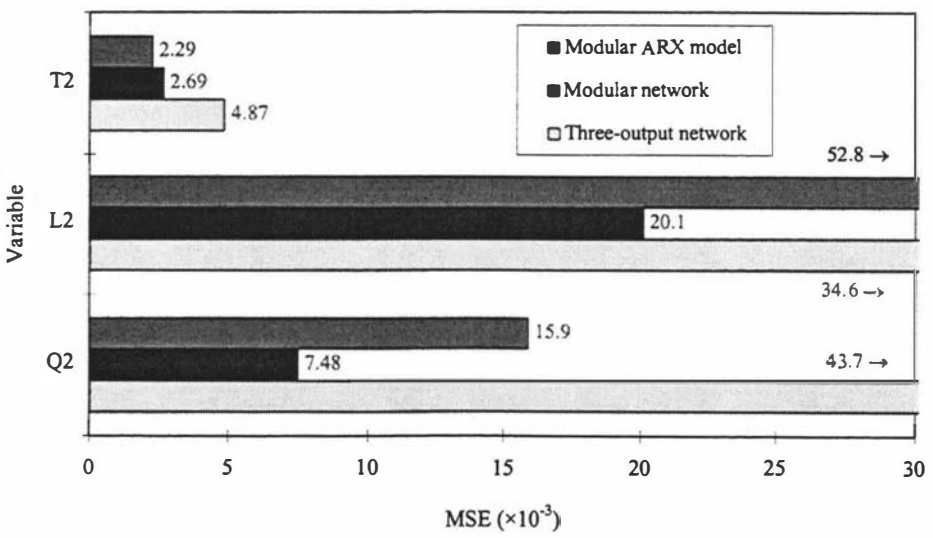


Figure 6-38: Validation 2 data prediction errors for Eff2 models

model results are significantly worse than those of the modular models. The modular neural network appears to perform better than the ARX model for the level and flowrate.

Table 6-18 summarises the number of parameters used in each Eff2 model. These were used to calculate the FPE results for the models.

Table 6-18: Number of parameters (N_w) in the Eff2 models

Modular ARX	Modular Neural network	Locally-connected 3-output network
31	137	88

By inspection of the FPE results (Figure 6-39 and Figure 6-40) it appears that the results for the modular neural network were on a par with that of modular ARX model. The

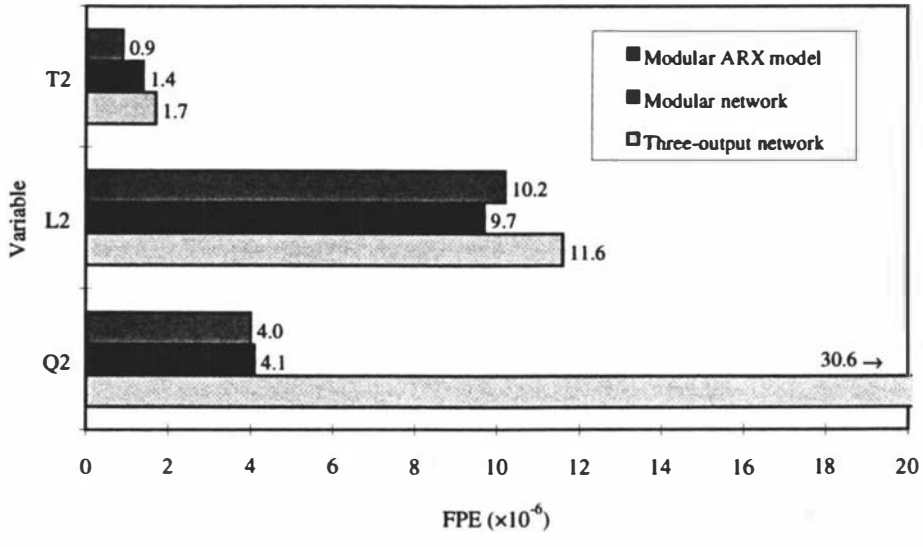


Figure 6-39: Training data FPE results for the Eff2 models

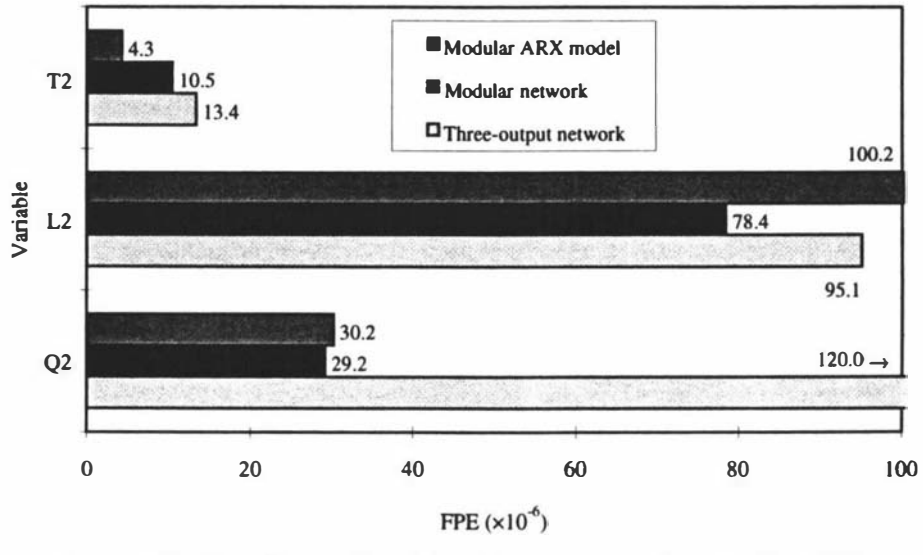


Figure 6-40: Validation 2 data FPE results for the Eff2 models

training results are particularly close while, in generalising for the validation data, the modular network model performs better for L_2 and about the same for Q_2. The locally-connected three-output network although much improved on the fully-connected network still fails to match the modular models over all the variables.

Before the neural network modelling was undertaken it was expected that the performance of the network would surpass that of the linear ARX model. Instead the results show that the linear model was comparable to the nonlinear neural network. According to Hornik *et al* [20] and Leshno *et al* [21], such a neural network model is theoretically able to approximate any continuous function. This is not restricted to either linear or nonlinear functions therefore even for linear data the networks should be the equal of the ARX model. The network training approach used, however, was not guaranteed to find an optimum network solution. Therefore the possibility remains that the sub-networks could have been improved.

Another possibility for the poorer neural network response is the presence of *chaotic behaviour* in the evaporation process or the neural network model itself. Chaos is a complicated behaviour of nonlinear dynamical systems. A chaotic process exhibits the following behaviour:

- A chaotic process oscillates between points in its phase space but without repeating the same trajectory. The areas of attraction in this space are unstable fixed points called *chaotic attractors.*
- A chaotic process exhibits sensitivity to initial conditions. A slight difference in initial values of some parameters will result in quite different responses from the process.

A chaotic process therefore seems random but it is not; it is not periodic but it does repeat almost the same patterns in its phase space; and it is predictable but only over a short time in the future.

There is evidence that neurons in the brain exhibit chaotic behaviour [22]. This has lead to the conclusion that neural network models also can manifest chaotic behaviour under certain conditions [23]. When the neural network model was used to predict the plant data the responses of the neural network were similar to that of the analytical and ARX model predictions and it was therefore concluded that neural network model did not exhibit chaotic behaviour.

There is a possibility that some of the plant measurements may contain an element of chaotic behaviour. This is observable in the step-response plots in Appendix 5(*c*) where the flow measurements appear to be more than just noisy signals.

The poorer-than-expected neural network results may therefore be due to the evaporation process being a chaotic system with a higher dimension than the dimension of the neural network input space. For instance, fewer time-lags than necessary may have been considered in modelling the system [23].

6.6 Conclusions

6.6.1 Summary of neural modelling approach

The development of a modular neural network model for use in MPC involves the steps outlined in the following sections.

Sub-network selection

The first task is the selection of the model variables and the determination of the functional relationships between them. The sub-networks are made up of the variables which are strongly related through knowledge of theoretical principles and observations of plant behaviour. It is also necessary to determine how the sub-nets should be linked together to form the full model.

Structure selection

Decisions need to be made on the topology of the sub-nets, the sampling rate of the model, and the number and length of the delays used on the variables. These all affect the final structure of the sub-networks.

The sampling rate needs to take into account the required precision of the predictions and the final control application of the model.

For dynamic sub-nets of order greater than one, a locally-connected topology should be employed. Generally it was found that the locally-connected sub-nets trained and generalised better than fully connected networks of equivalent size. If fully-connected sub-nets are used then the topology is determined through testing with different numbers of hidden neurons.

The delays applied to the input variables can be determined through correlation analysis between the input and output variables. Use of linear ARX modelling studies can assist in the choice of the order of the sub-networks.

Sub-network training

The individual sub-networks are trained using a backpropagation-through-time algorithm with the Levenberg-Marquardt optimisation (LMTT) to update the weights. During the training phase the networks are regularly tested with a second data set. The

test is a long-range prediction of the testing data set. After a number of training runs the network with the best generalisation performance is selected.

Sub-network validation

The sub-network validation stage can involve correlation-based validation tests and cross-validation tests.

The correlation-based validations involve statistical tests for correlations between combinations of the model inputs and the prediction error. These tests assess the adequacy of the model fit.

Cross-correlation involves testing the network for long-range predictions on different input signals-not used during the training phase-and comparing the response with the actual system output.

Combining of networks

The sub-networks or modules which make up the model of the full system are connected together. For process engineering systems these would generally follow the form of the actual process so that the model would have feed forward and feedback connections to parallel the actual plant.

Validation

Cross-correlation tests are carried out with the full system model in order to analyse its performance for long-range predictions.

6.6.2 Model performance

The use of a locally-connected structure for the neural network modelling was found to improve both the convergence and generalisation performance of the networks over networks with a fully-connected structure. Such an approach also enables the models to be easily analysed to discern how the network forms its output from the input variables.

Combining the sub-nets into a modular model proved to be superior to training a single network to predict a number of the outputs.

The linear ARX model generally produced similar results to the neural network model. If one takes into account the fewer number of parameters in the ARX model, however the linear model performed very well.

Both models had difficulty in predicting the concentrate level variable, L_2.

Some suggestions have been raised as to why level predictions have proved difficult:

- The integrating nature of level system causes modelling errors to accumulate.

- The possibility of unobservable modes within the level system since part of the separator is included as the reservoir. The fluctuations in vapour pressure in the separator, which are unable to be measured, may interact with the level measurements.
- The poor performance could equally be due to the poor selection of training data. Some of the data in the validation sets extend beyond the limits of the training data. This is undesirable but can be difficult to avoid when attempting to model a number of variables simultaneously.
- Measurement problems have been experienced with the level sensors when operating the pilot-plant. From observations it is suspected that the measurements at either the high or low concentrate levels may not be totally accurate. It is difficult to be sure of this since one cannot see inside the evaporator during operation. If the integrity of the measurements are in doubt this could mean that at some instances the models may be correctly estimating the levels when it appears that they are not.

It is believed that the modular modelling approach demonstrated in this chapter remains a valuable approach to developing neural network models of engineering systems. This is despite the poorer-than-expected performance of the modular neural network both in absolute terms and in relation to the linear regression model. It must also be noted that at no time during the testing and validation of the neural network models was there any evidence of the unstable behaviour which is common with incorrectly trained recurrent networks. This gives confidence in the training approach demonstrated here.

The training method developed in this chapter was used to complete the evaporator model. This is discussed in the following chapter.

6.7 References

[1] **Lin, T., Horne, B.G., Tino, P. & Giles, C.L.**, Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 1996, vol. 7, pp. 1329-1338.

[2] **Turner, P., Montague, G. & Morris, J.**, Nonlinear and direction-dependent dynamic process modelling using neural networks. *IEE Proceedings - D, Control Theory Appl.*, 1996, vol. 143, pp. 44- 48.

[3] **Joerding, W.H. & Meador, J.L.**, Encoding a priori information in feedforward neural networks. *Neural Networks*, 1991, vol. 4, pp. 847-856.

[4] **Thompson, M.L. & Kramer, M.A.**, Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*, 1994, vol. 40, pp. 1328-1340.

[5] **Psichogios, D.C. & Ungar, L.H.**, A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 1992, vol. 38, pp. 1499-1511.

[6] **Conlin, J., Peel, C. & Montague, G.A.,** Modelling pressure drop in water treatment. *Artificial Intelligence in Engineering*, 1997, vol. 11, pp. 393-400.

[7] **Pugmire, R.H.,** *A neural network based window filter and its training for image processing tasks.* PhD Thesis, Massey University, New Zealand, 1995.

[8] **Ronco, E. & Gawthrop, P.,** Modular neural networks: a state of the art. Technical Report : CSC-95026, University of Glasgow, UK, 1995.

[9] **Happel, B.L.M. & Murre, J.M.J.,** Design and evolution of modular neural network architectures. *Neural Networks*, 1994, vol. 7, pp. 985-1004.

[10] **Reed, R.,** Pruning algorithms: A survey. *IEEE Transactions on Neural Networks*, 1994, vol. 4, pp. 740-747.

[11] **Fahlman, S. & Lebiere, C.,** The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, PA, 1991.

[12] **Jacobs, R., Jordan, M., Nowlan, S. & Hinton, G.,** Adaptive mixture of local experts. *Neural Computation*, 1991, vol. 3, pp. 79-87.

[13] **Mavrovouniotis, M.L. & Chang, S.,** Hierarchical neural networks. *Computers & Chemical Engineering*, 1992, vol. 16, pp. 347-369.

[14] **Russell, N.T. & Bakker, H.H.C.,** Modular modelling of an evaporator for long-range prediction. *Artificial Intelligence in Engineering*, 1997, vol. 11, pp. 347-355.

[15] *Neural Network Toolbox (ver. 2.0a)*, The MathWorks Inc., Natick, MA, USA, 1994.

[16] **Isermann, R.,** Practical aspects of process identification. *Automatica*, 1980, vol. 16, pp. 575-587.

[17] **Ljung, L.,** *System Identification - Theory for the User*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1987.

[18] *System Identification Toolbox (ver. 3.0a)*, The MathWorks Inc., Natick, MA, USA, 1992

[19] **Su, H.-T., McAvoy, J. & Werbos, P.,** Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial Engineering Chemistry Research*, 1992, vol. 31, pp. 1338-1352.

[20] **Hornik, K., Stinchcombe, M. & White, H.,** Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989, vol. 2, pp. 359-366.

[21] **Leshno, M., Lin, V.Ya., Pinkus, A. & Schocken, S.,** Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 1993, vol. 6, pp. 861-867.

[22] **Freeman, W.,** Simulation of chaotic EEG patterns with a dynamic model of the olfactory system. *Biological Cybernetics*, 1987, vol. 56, pp. 139-150.

[23] **Kasabov, N. K.,** *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*, MIT Press, MA, USA, 1996.

7

A Neural Network Model of the Complete Evaporator System

OVERVIEW

This chapter extends the modular modelling approach devised in Chapter 6 to produce a neural network model of the complete evaporator system. The chapter includes analysis of the full evaporator model and comparisons with both an equivalent linear regression model and the analytical model of Chapter 3.

7.1 Neural network model development and results

To complete the evaporator model, a further ten sub-networks were developed. The first and third effects were modelled in a similar fashion to the Eff2 model with three sub-nets each. The four other sub-nets are used to predict the feed flowrate, the two preheater temperatures and the external steam supply temperature.

7.1.1 Selection of model variables

Temperature sub-nets

The variables used in the effect temperature sub-nets were of a similar form to that of T_2-net. Therefore the following functions were defined:

$$T_1 = f(T_s, T_2, Q_0, T_1) \tag{7.1}$$

$$T_3 = f(T_2, T_{ph1}, Q_3, T_3) \tag{7.2}$$

The second preheater temperature sub-network has the functional relationship:

$$T_{ph2} = f(T_s, T_{ph1}, Q_0, T_{ph2}) \tag{7.3}$$

with the shell temperature (T_s), the incoming temperature (T_{ph1}), the product flowrate through the tubes (Q_0) and previous values of the output (T_{ph2}) as variables. The first preheater temperature is expressed in a similar fashion but without the incoming product temperature as a variable since this is assumed constant.

$$T_{ph1} = f(T_3, Q_0, T_{ph1}) \tag{7.4}$$

The steam temperature sub-net has the functional relationship:

$$T_s = f(T_{sp}, T_1, Q_0, T_s) \tag{7.5}$$

with the steam temperature controller setpoint (T_{sp}), the temperature of the first effect(T_1), and feed flowrate (Q_0) and previous values of the output (T_s). As with the analytical model the steam controller is considered as part of the model.

Level sub-nets

As was the case for the Eff2 system, the concentrate levels are a function of the flow in and out of the effect, the temperature of the effect and previous values of the level.

$$L_1 = f(Q_0, Q_1, T_1, L_1) \tag{7.6}$$

$$L_3 = f(Q_2, Q_3, T_3, L_3) \tag{7.7}$$

Flowrate sub-nets

The flowrate sub-nets are based on the Bernoulli energy balance equation for the flow through the tubes. The pressure differences are represented with the relevant temperatures. For the feed flowrate (Q_0) and third effect flowrate (Q_3) networks the pressure difference is a function of the atmospheric pressure which is assumed constant and so only the relevant effect temperature is included as a variable. For the feed flow sub-net the level in the feed tank is also assumed to be constant and so no level variable is included.

$$Q_1 = f(T_1 - T_2, N_1, L_1, Q_1) \tag{7.8}$$

$$Q_3 = f(T_3, N_3, L_3, Q_3) \tag{7.9}$$

$$Q_0 = f(T_1, N_0, Q_0) \tag{7.10}$$

7.1.2 Input Delays

The correlations between the sub-network outputs and their respective inputs were analysed to determine which input delays should be used.

Generally the delay vectors used for the additional sub-networks are equivalent to those used for the Eff2 nets. Therefore the temperatures have inputs at $t-1$, $t-2$ and $t-4$, the levels have inputs at $t-1$, $t-2$ and $t-3$ and the flowrates use no delays on the input variables with the output delayed by one time step.

The exceptions to the above are the steam temperature, first effect temperature and second preheater temperature nets. The first effect and second preheater temperatures are functions of the steam temperature, T_s. Because of the longer delay involved with the steam reaching the shells of these stages the delay vectors are required to be longer. The maximum correlation between both temperatures and the steam temperature occurs at a delay of 7 time steps. Therefore a D_u vector of [1 5 7] was used for these two sub-nets. The steam temperature sub-net did not require as long a time history therefore a D_u of [1 2 3] was used.

7.1.3 Sub-network structures

All the temperature and level sub-nets were locally connected while the flow sub-nets were fully-connected since they were only first order. As with the Eff2 system the number of hidden-layer neurons used in the remaining flowrate networks was determined through varying the number and comparing the performance. From these trials it was found that either 5 or 6 hidden neurons was best for each of the flow networks which gave each sub-net 31 weights and biases. This is more than that used in the Q_2-net and is probably due to the Q_2 measurement being less noisy than the others.

Table A9-1 in Appendix 9 summarises the structures of the 13 sub-networks for the evaporator system.

7.1.4 Model structure

The general structure of the modular neural network model is illustrated in Figure 7-1. From this diagram it can be seen that the sub-networks are arranged in parallel with the outputs being fed back to an additional input layer. This layer of links which lies between the tapped delay lines and the sub-nets is not fully-connected, is not adaptable and each connection carries a weighting of unity.

The complete modular evaporator model of 13 sub-networks has 5 input variables to predict 13 outputs and consists of 630 adjustable network parameters. Located prior to the sub-nets are the tapped delay lines and signal conditioning layer. Here the delays are applied and some of the inputs are prepared for feeding to the sub-nets (eg. filtering of the flowrates). All the outputs are recurrent and are fed back into the tapped delay line and signal conditioning layer. This gives a total of 18 data streams entering the

tapped delay lines which fan out into the 120 connections that feed the input layers of the sub-nets. The sub-net portion of the model can be considered as a sparsely connected two-layer network with a topology of 120-116-13.

After training, the 13 sub-networks were combined to form the complete evaporator model as illustrated in Figure 7-1. The SIMULINK block diagram for the modular neural model is shown in Figure 7-2. Each of the network blocks used in the SIMULINK model has a similar arrangement as that of the T_2-network shown in Figure 6-30.

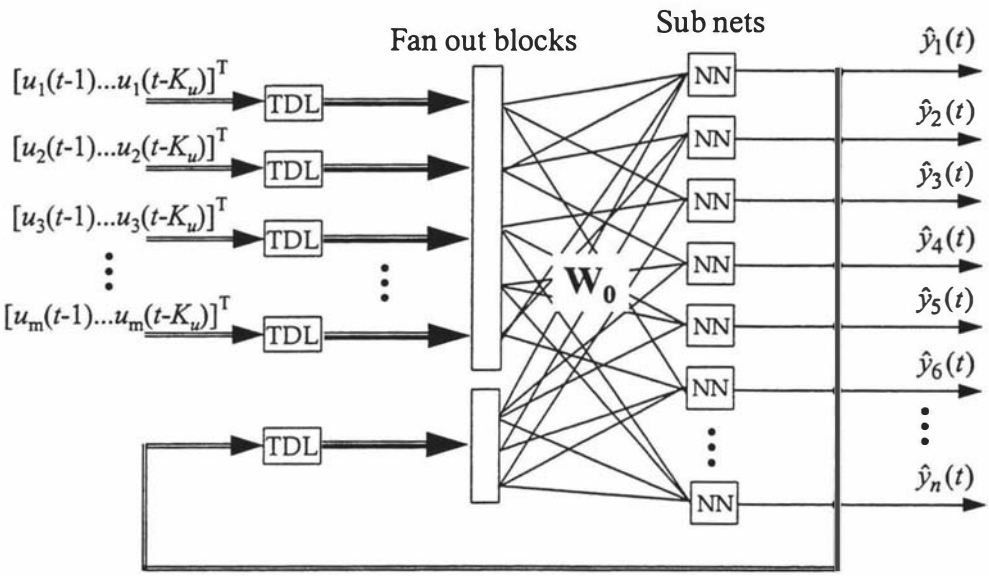


Figure 7-1: General structure of the neural network evaporator model

7.1.5 Training and testing of sub-nets

The training for the additional sub-nets was performed in a similar fashion to the previous sub-net training runs. Each network was trained a number of times and the network with the best testing error selected for use in the full model. As observed previously the temperature and flowrate networks generally performed better than the level networks. The full training and testing results for the best networks are tabulated in Appendix 9.

7.1.6 Validation of the evaporator model

The modular network model was simulated to perform long-range predictions of the data sets. Again the training data set was included to provide a comparison for the validation sets. As would be expected the results for the full model are not as good as those for the individual sub-nets. The prediction errors are listed in Table 7-1 to 7-3 together with some example plots. Plots for all the Validation 2 predictions are given in Appendix 9.

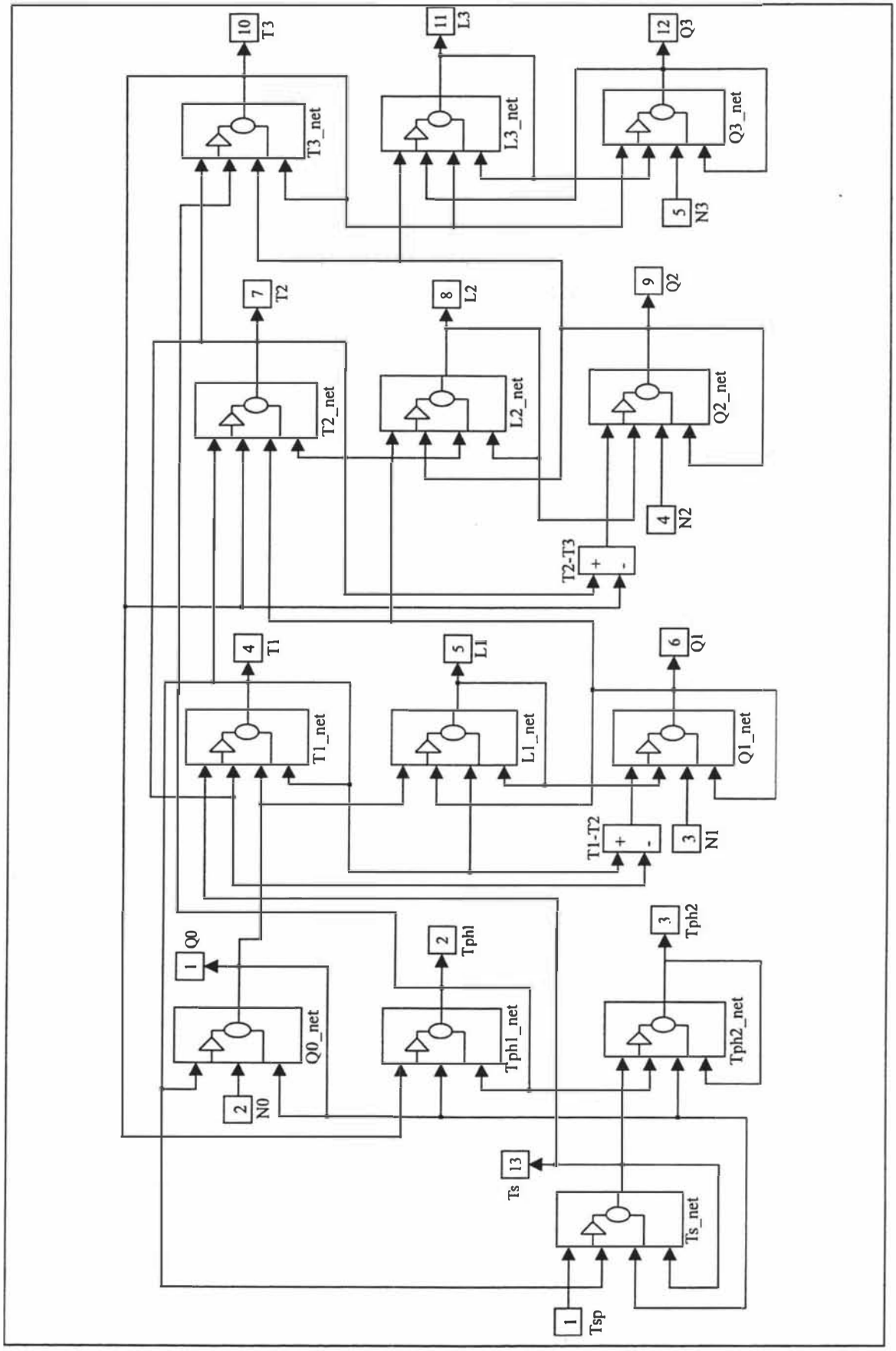


Figure 7-2: SIMULINK block diagram of *nnevap.m* model

Temperatures

Generally the temperature predictions remain satisfactory however the errors do increase through the evaporator. The second preheater and first effect are where the steam first enters the system and these have the best results. At the downstream end the third effect and first preheater predictions are the worst. This trend is most likely due to the accumulation of errors through the temperature variables of the model. The model outputs for the downstream temperatures tend to appear as a smoothed version of the actual data which does not follow every trough or rise (see plots in Appendix 9).

Table 7-1: Temperature prediction errors for the modular neural network.

	Mean-Squared Error ($\times 10^{-3}$)		
Output	Training Data	Validation 1 Data	Validation 2 Data
T_s	2.65	4.71	5.97
T_{ph1}	20.60	26.7	30.20
T_{ph2}	2.05	4.36	6.54
T_1	1.54	5.61	5.90
T_2	3.29	12.6	8.19
T_3	10.90	29.1	21.30

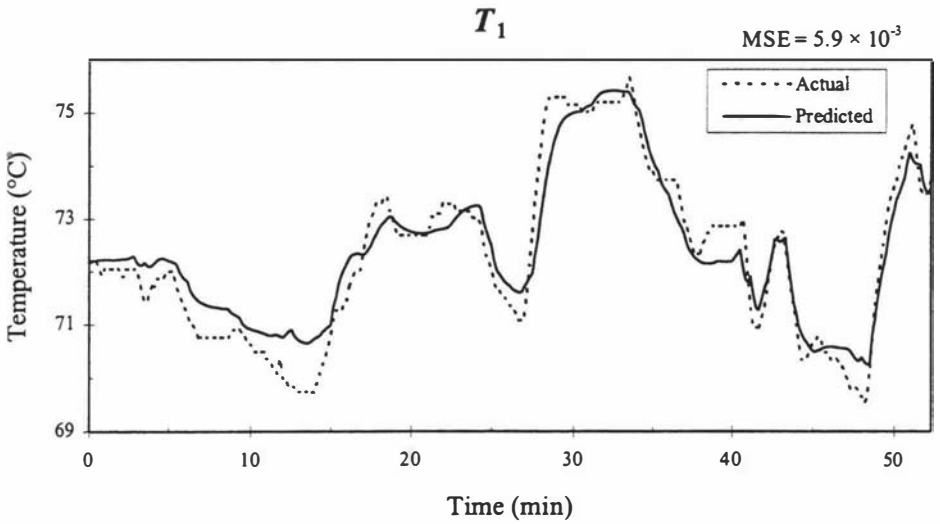


Figure 7-3: Neural model prediction of T_1 for Validation 2 data

Concentrate levels

Once again the level networks produce the poorest predictions of all the variables. The second effect level being the most difficult to accurately model.

Observing Figure 7-4 it appears that the L_2 prediction is reasonable at higher values but performs poorly when estimating the lower values correctly (between 30 and 40 min). This may be due in part to the poor range of the training data.

Table 7-2: Level prediction errors for the modular neural network.

	Mean-Squared Error ($\times 10^{-3}$)		
Output	Training Data	Validation 1 Data	Validation 2 Data
L_1	7.17	14.00	13.00
L_2	17.40	52.60	84.30
L_3	9.29	32.80	13.40

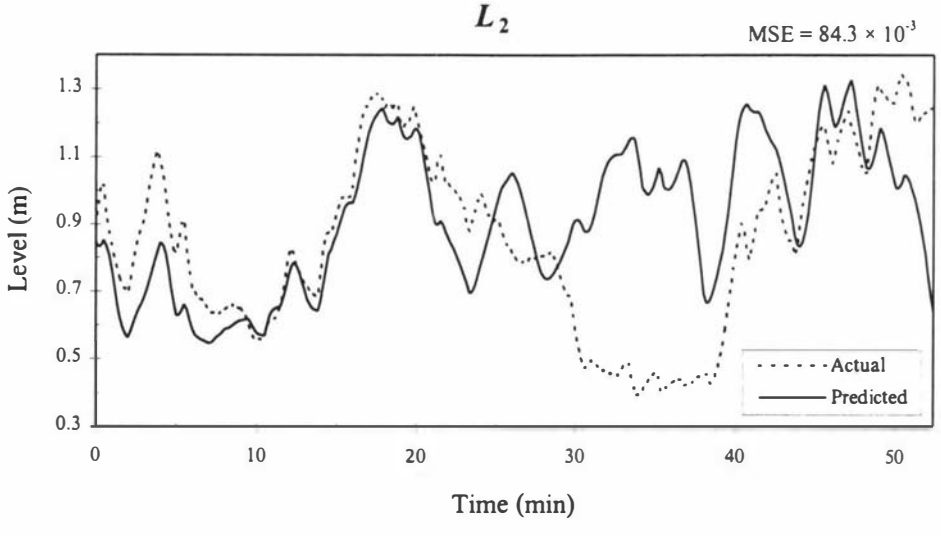


Figure 7-4: Neural model prediction of L_2 for Validation 2 data

Product flowrates

Generally the flowrate outputs tend to produce a smoothed representation of the actual data as can be seen in the following plot.

Table 7-3: Flowrate prediction errors for the modular neural network.

	Mean-Squared Error ($\times 10^{-3}$)		
Output	Training Data	Validation 1 Data	Validation 2 Data
Q_0	3.57	5.26	4.84
Q_1	4.70	9.71	10.00
Q_2	6.07	16.00	26.50
Q_3	7.29	10.40	12.40

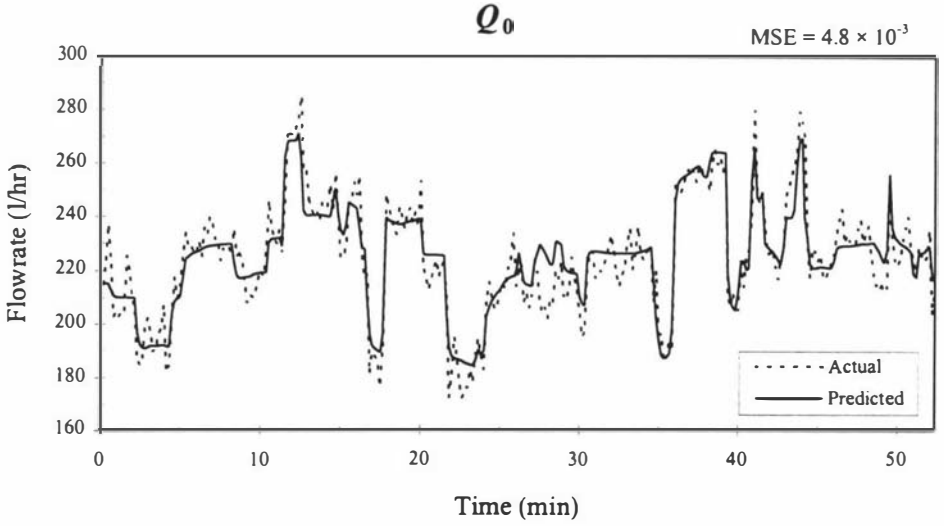


Figure 7-5: Neural model prediction of Q_0 for Validation. 2 data

7.1.7 Global training of the modular network

Most modular systems the sub-networks are trained independently or sequentially [1, 2] as has be the case for the evaporator model. However for these systems it is likely that neither the task decomposition nor the estimation of the network weights is optimal in a global sense. This is obvious from the results shown for the complete evaporator model in that the neural sub-networks perform well when used individually but when combined together the predictions deteriorate. This can be attributed to the imperfect predictions from the sub-networks which combine to cause larger errors to occur. The individual training of the networks only considers the externally-recurrent link for the sub-network but does not account for the recurrent connections between the sub-networks.

The performance of the full model may be improved through training the sub-networks together within the overall model structure and optimising according to a global goal. In this way the network weights may be adapted with all the interactions considered. This type of global training for modular systems has been shown to lead to improved performances for different applications in signal processing [3] and image processing [4]. The approach of training the model as a whole also preserves the advantages offered by a modular model since the sub-nets would still be able to be separated and used independently.

Although the training task is more onerous than individually training each sub-net, it may prove equivalent in terms of total development time. Global training however is likely to be plagued with local minima more so than with the sub-network training. This may be able to be avoided by selecting a reasonable initial starting point for the full

model optimisation. This could be achieved through first training the individual networks separately and then globally training the complete model.

Difficulties with memory and computing power may also arise in the global training of the modular model. A method like LMTT would struggle due to the inefficiency of the algorithm. Both LM optimisation and BPTT are particularly memory intensive and it would require significant computing power and storage to successfully perform the task. For the evaporator model the Jacobian matrix alone would have dimensions of $(13\times N) \times 630$, where N is the number of time steps This is formed at each iteration of the LM algorithm. Additionally, BPTT requires N copies of the network to be stored during training. Other more efficient methods like real-time recurrent learning [5] may be capable of handling the larger problem, although this method is more computationally expensive.

7.1.8 Comparison with ARX model

A further ten linear regression sub-models were developed and combined with the Eff2 ARX model to form a linear, modular evaporator model. The variables used in the sub-models were equivalent to those used in the neural sub-networks. As in the Eff2 ARX model development, the structures which minimised Akaike's FPE criterion were selected for use. A table summarising the structures for all the sub-models is given in Appendix 10. The full ARX model of 13 sub-models has a total of 132 parameters.

The modular ARX model implemented within SIMULINK has a block diagram representation exactly equivalent to that of the modular neural model in Figure 7-2. The only difference being that the sub-network blocks are replaced by the ARX sub-models.

The modular ARX representation was simulated to obtain long-range predictions for all the data sets. Generally the trend of the predictions is similar to that observed in the neural model, for instance the temperature results deteriorate downstream through the plant, the levels have the most difficulty in obtaining accurate predictions and the flowrates appear to be smoothed versions of the actual data. Generally, the physical MSE values are similar to those of the neural model, with only a few exceptions. This is significant when one considers that the number of model parameters is only 132 which is just over a fifth of the 630 weights used in the neural model.

Three plots are given below for a comparison with the neural network model plots. A table of all the ARX model predictions is given in Appendix 10 together with the remainder of the validation 2 plots.

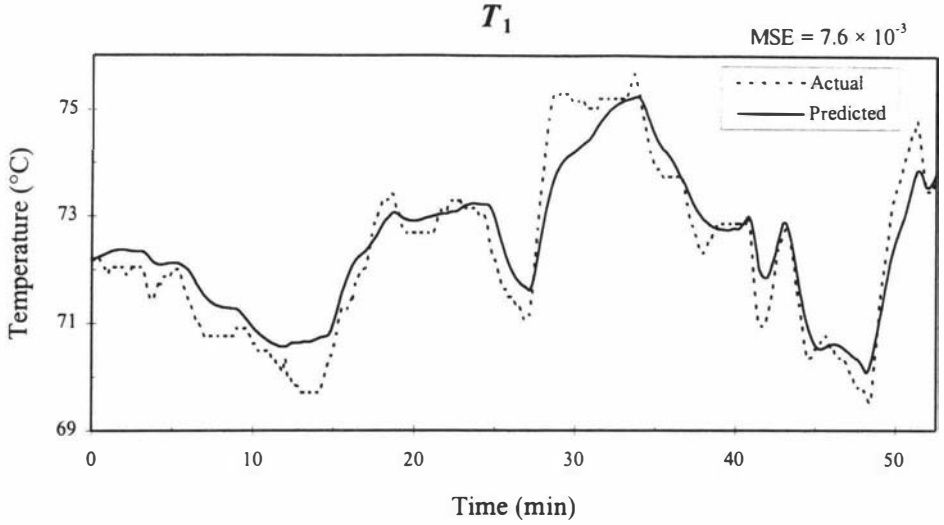


Figure 7-6: ARX model prediction of T_1 for validation 2 data

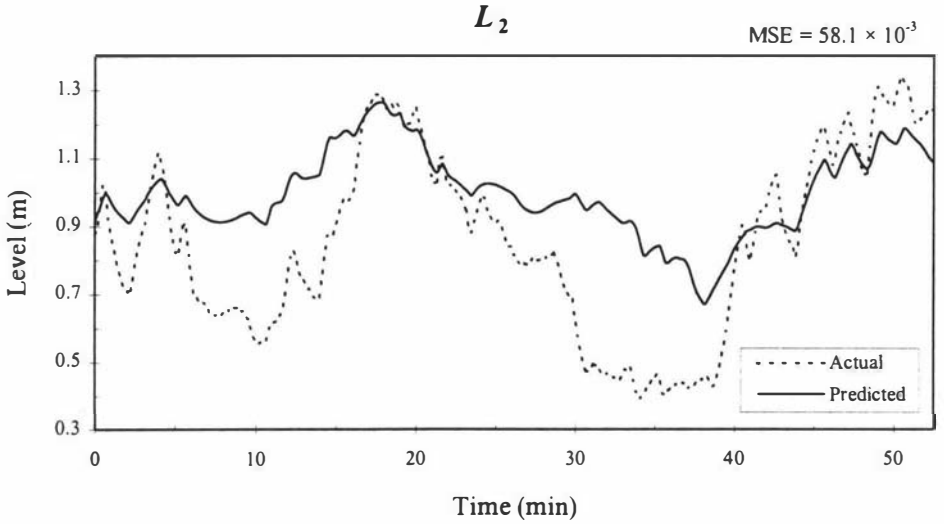


Figure 7-7: ARX model prediction of L_2 for validation 2 data

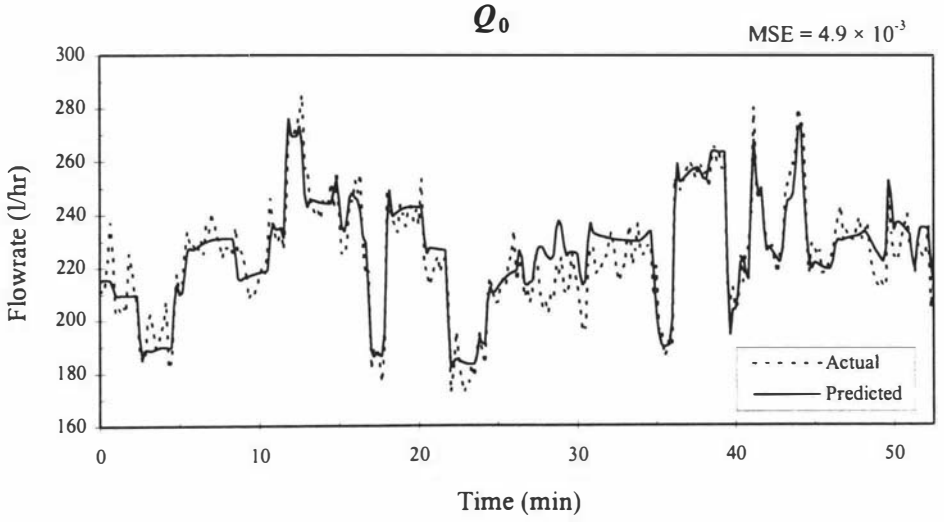


Figure 7-8: ARX model prediction of Q_0 for validation 2 data

7.1.9 A mixed-model system

Some neural sub-networks perform better than the ARX sub-models and some do not. An improved evaporator model could be created by combining only the best modules. It is a simple task to form a mixed neural network and linear regression model by inserting them into the existing SIMULINK block diagram to form a completely new model. This flexibility is one of the main attractions of a modular structured model. The fact that different types of modules are used within the overall structure is not a concern to the operation of the model. By designing the modules to fit into a generic block diagram one can tailor the model for a specific application. An example of this could be if the model was used in a control strategy; the portion of the plant under control could be modelled by modules with a greater accuracy and detail than the modules relating to other sections of the plant. Parts of the model for which data is scarce could be described by a theoretically-derived model while other portions could use empirical modelling techniques.

7.1.10 Comparison with analytical model

It is an interesting exercise to finally compare the performance of the modular neural network and ARX models with the analytical model of Chapter 3.

Using the validation 2 data set as a basis the MSE for the models predictions are tabulated in Table 7-4 on the following page.

An extra column has been included for the MSE of the mean data value. This is obtained by calculating the MSE for the mean value of the actual data and provides an absolute reference against which predictions can be compared. Essentially if we were to model the plant by just assuming the outputs are constant and equal to the data means, these are the MSEs which would result.

Generally for the modular models the predictions are well below this mean-value MSEs apart from the L_2 neural network prediction. As illustrated in Figure 7-4 this result is predominantly due to the poor performance at the low values of the concentrate level.

Inspecting the analytical model prediction errors show them to be worse than the modular neural network and ARX models.

When comparing the modular models to the analytical model one should not only consider the performance of the model predictions but also other factors in the development and implementation of the models. Depending on the final application of the model these factors can be just as important as the prediction results. These

additional factors include; development effort, simulation speed, sampling rate, extrapolating ability and model analysis.

Table 7-4: Comparison of the evaporator models in predicting the Validation 2 data

	Mean-Squared Error ($\times 10^{-3}$)			
Output	Analytical Model	Neural Network	ARX Model	Mean of data
T_s	7.1	6.0	6.8	55.6
T_{ph1}	43.2	30.2	7.6	51.6
T_{ph2}	11.4	6.5	8.2	63.9
T_1	8.1	5.9	24.1	65.8
T_2	26.6	8.2	23.2	50.9
T_3	68.9	21.3	6.9	56.5
L_1	21.3	13.0	11.9	68.5
L_2	105.6	84.3	58.1	78.9
L_3	83.0	13.4	19.1	35.2
Q_0	5.2	4.8	4.9	36.2
Q_1	28.3	10.0	9.7	41.5
Q_2	60.3	26.5	17.2	43.8
Q_3	9.8	12.4	9.6	30.4

Disadvantages of analytical modelling

The disadvantages of analytical modelling techniques include

- development effort and
- simulation speed.

Development effort

Realistically, any well-performed model is going to take time to develop and fine-tune. The main advantage of neural network techniques is that there is less emphasis on the theoretical derivation of equations and more on the processing and manipulation of the plant data.

The effort involved with analytical model development is considered to be greater than that required for black-box modelling methods for the following reasons:

- Lack of adequate theoretical knowledge and mathematical skills

- Identification of plant parameters

For a typical industrial processing company there is often lack of knowledge and expertise available for analytical modelling projects. On the other hand there is often a large quantity of process data available for use.

The other drawback of analytical modelling is the need for estimating the plant parameters. Constants such as heat transfer coefficients, pump and valve constants, friction terms and the like, consume a large proportion of the development effort and often, to accurately estimate these, specific plant trials may be required.

Simulation speed

Analytical models typically contain a number of differential equations and as a result the computational requirements to solve these equations can be large. On the other hand neural networks once trained behave like a nonlinear look-up-table, mapping inputs to outputs. The computational effort is then minimal compared to that of the physical model. For this reason neural networks are much more suitable for on-line implementation within control and optimisation strategies.

Advantages of analytical modelling

An analytical model has the following advantages over empirical models:

- Ability to be give results for a wide range of conditions, not bound by an inability to extrapolate
- Analysis of characteristics and robustness can be performed
- Variable sampling rate

Extrapolation ability

The advantage of an analytical model is that typically the model can perform well outside the normal operating limits of the process. This is particularly useful for examining un-proven operating strategies. For a neural network the model performance typically degrades very rapidly once it attempts to extrapolate beyond the range of the training data.

Ability to analyse model

Analytical models can be analysed with many well established tools and because it is formulated from a theoretical basis many of its properties are well understood. The model could easily be linearised and linear systems theory applied to determine the stability of the model.

Sampling rates

The analytical model requires a variable step-size integration algorithm to solve the differential equation and so this defines the sampling rate. The full dynamics of the system both fast and slow are captured by the model. For neural networks this is dependent on the judicious selection of the data sampling rate and the design of the network.

Rather than stating one technique is better than the other it is more correct is say one is more appropriate to use for certain applications. Neural networks will certainly not make analytical modelling techniques become obsolete.

7.2 Conclusions

Neural network modelling

As expected the prediction errors of the overall model are worse than those of the individual networks but the estimates possess similar characteristics:

- The temperatures are best estimated, however these become less accurate in the downstream sections of the system.
- The concentrate levels are worst predicted with significant offset present.
- The product flowrates are smoothed versions of the noisy measurements.

Combining the sub-nets into a modular model proved to be superior to training a single network to predict a number of the outputs. Modular modelling offers other advantages such as:

- Allowing the combining of different modelling techniques within a single model. The model can therefore be tailored for a specific application.
- Allowing easy extension or improvement to the model by modifying only the necessary modules rather than reforming the whole model.

A promising avenue for investigation is the global training of the whole modular system in order to improve the model performance and reduce the accumulation of errors [6]. Individually training the sub-networks first then applying a global optimisation to the complete model is likely to be the best method of implementation.

Comparison with the ARX model

The fact that the linear model performs satisfactorily for some of the variables indicates that these portions of the process data are not highly nonlinear. Generally the ARX results have a similar characteristic to the neural network model.

The neural network model has not performed as well as one might expect compared with the ARX model, and in absolute terms.

The poorer than expected performance could be for the following reasons:

1. Poor training data - badly ranged data, or poor measurements
2. Training methodology used - difficult to converge to a good solution
3. Difficulty modelling the system solely on measured data - unobservable modes present
4. Poor choice of network structures and variables used.

Of these points the fourth is the least likely to be the cause. This is for two reasons:

i) The superior convergence and generalisation performance of the locally-connected structure over fully-connected structures has been demonstrated.

ii) The choice of variables are believed to be appropriate based on a feedforward training tests of the networks which resulted in good prediction results [7] (also see correlation-based validation tests § 6.4.9).

The problems experienced with the prediction of the levels (particularly L_2) are most likely to be a combination of poor training data–not covering the range of all the levels– and the existence of unobservable modes within the level systems.

The structured approach of training is not guaranteed to find an optimal solution so an alternative combination of sub-networks could possibly be obtained to improve the prediction results. The use of a different recurrent training methodology may result in a better solution but this is also unlikely to be guaranteed.

The use of *teacher forcing* has been shown to be helpful in training recurrent networks [5, 8] and could be used to improve the convergence of the recurrent sub-networks. In this scheme, the target outputs from the plant are used to drive the network dynamics in place of the feedback of its output. One problem with this technique is that the network response may not be stable when the network is implement recurrently after training. Several heuristics have been proposed to enhance the stability of the networks [9]:

Noisy forcing: Add some noise to the forcing input.

Partial forcing: Used a mixed input where the forcing input is a function of the target value and the network output. The forcing signal can be moved closer to value of the network output as training progresses.

Part-time forcing: Initially turn on forcing to synchronise the network to the teacher then turn it off to train using the actual network output.

Using one of these teacher-forcing methods may cause the sub-network training to avoid local minima and result in improved convergence and hence better prediction results.

Analytical model

Generally the prediction errors for the analytical model are worse than those of the modular neural network and linear ARX models. The black-box models may give better prediction results for the validation data and easier development but whether they are more appropriate depends on the application. Other factors like variable sampling rates, extrapolation ability and the ease of model analysis are involved in the choice of the most appropriate model.

7.3 References

[1] **Bennani. Y.**, & **Gallinari, P.**, Task decomposition through a modular connectionist architecture: A Talker Identification System. In: *Third International Conference on Artificial Neural Networks* (I. Aleksander & J. Taylor, Eds.), 1992, Amsterdam: North-Holland, Vol. 1, pp. 783-786.

[2] **Thiria, S., Mejia, C., Badran, F. & Crépon, M.**, Multimodular architecture for remote sensing operations. In: *Advances in Neural Information Processing Systems 4* (J. E. Moody, S. J. Hanson & R. P. Lipmann, Eds.), 1992, San Mateo, CA, USA: Morgan Kaufmann, pp.675-682.

[3] **de Bollivier, M., Gallinari, P. & Thiria, S.**, Neural nets and task decomposition. *Proceedings of the International Joint Conference on Neural Networks*, Seattle, USA, 1991, vol. 2, pp. 573-576.

[4] **Fogelman, F., Lamy, B. & Viennet, E.**, Multimodular neural network architectures for pattern recognition: Applications in optimal character recognition and human faces recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 1993, vol. 7.

[5] **Williams, R.J. & Zipser, D.**, A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989, vol. 1, pp. 270-280.

[6] **Gallinari, P.**, Training of modular neural net systems. In: *The Handbook of Brain Theory and Neural Networks*, (M. A. Arbib, Ed.), 1995, MIT Press, MA, USA, pp. 582-585.

[7] **Russell, N.T. & Bakker, H.H.C.**, Modular modelling of an evaporator for long-range prediction. *Artificial Intelligence in Engineering*, 1997, vol. 11, pp. 347-355.

[8] **Pearlmutter, B.A.**, Dynamic recurrent neural networks. Technical Report CMU-CS-90-196, Carnegie Mellon University, Pittsburgh, PA, 1990.

[9] **Doya, K.**, Recurrent networks: supervised learning. In: *The Handbook of Brain Theory and Neural Networks*, (M. A. Arbib, Ed.), 1995, MIT Press, MA, USA, pp. 796-800.

8

Simulated Model Predictive Control of the Evaporator

OVERVIEW

This chapter draws together the modelling work of the previous chapters to demonstrate the application of Model Predictive Control to the pilot-plant evaporator system. To simulate the MPC strategy the analytical model of Chapter 3 is used to represent the actual plant. The modular neural network and linear regression model developed in Chapter 6 are embedded in separate controllers to perform the predictions. The performance of both the neural network-based and linear model-based MPC controllers are demonstrated for setpoint tracking and disturbance rejection and are compared with a system which employs PI control.

8.1 Control objectives and strategy

As stated in Chapter 2, the main control objectives for an evaporator are to maintain consistent product concentration and flow while minimising energy use and maintaining product quality. For a general evaporation system the main manipulated variables are the steam flow, feed flow and condenser cooling water flow.

These general statements need to be revised for demonstrating control using the simulation model. This is because the product concentration is not modelled as part of the evaporator and the condenser cooling water flowrate for the pilot-plant is manually controlled and is kept at a constant flow during operation.

The two predominant factors influencing the operation of pilot-plant are the energy input and the product flowrate through the plant. Together, these effectively govern the operating point of the plant.

Both the flowrate and steam input influence the rate of evaporation and temperatures in the effects.

With the concentrate levels in each effect constant, the mass of product in the system is constant and the feed flowrate therefore determines the product flow through the plant.

The highest temperature the product experiences is in the first effect. By regulating this temperature one can control the energy input into the system. Additionally this temperature is critical for maintaining product quality. In practice it is important to control the maximum temperature in the system so as to not cause damage to heat sensitive products.

Therefore the MPC strategy applied to the simulation model aims to control the first effect temperature and feed flow.

The inputs used to control these two variables are the setpoint for the steam temperature controller and the feed pump. The position of the steam valve is controlled via a PID controller (discussed in Chapter 3). The MPC controller therefore acts as the master controller in a cascade arrangement with this PID loop. It determines the best value of the steam temperature setpoint in order to obtain the desired first effect temperature.

To regulate the product flow through the plant effectively the concentrate levels in the base of each of the effects should be maintain constant. These concentrate levels are controlled via local PI loops.

The controlled and manipulated variables are summarised in the following table.

Table 8-1: Controlled and manipulated variables

Controlled variables	**Manipulated variables**
MPC controller	
Effect 1 temperature (T_1)	Steam temperature setpoint (T_{sp})
Feed flowrate (Q_0)	Feed Pump (N_0)
PI(D) controllers	
Effect 1 Level (L_1)	Effect 1 Pump (N_1)
Effect 2 Level (L_2)	Effect 2 Pump (N_2)
Effect 3 Level (L_3)	Effect 3 Pump (N_3)
Steam temperature (T_s)	Steam Valve

The control objectives used here are similar to that reported for multivariable model-based control for a milk evaporator [1] and an evaporator used in a pulp mill [2]. Both these examples include the steam flow and feed flow as the primary manipulated variables with the aim to control the product outlet temperature and density and the product flowrate through the evaporator.

Figure 8-1 illustrates the MPC control on the pilot-plant evaporator. For clarity the concentrate level PI loops have been omitted from this diagram.

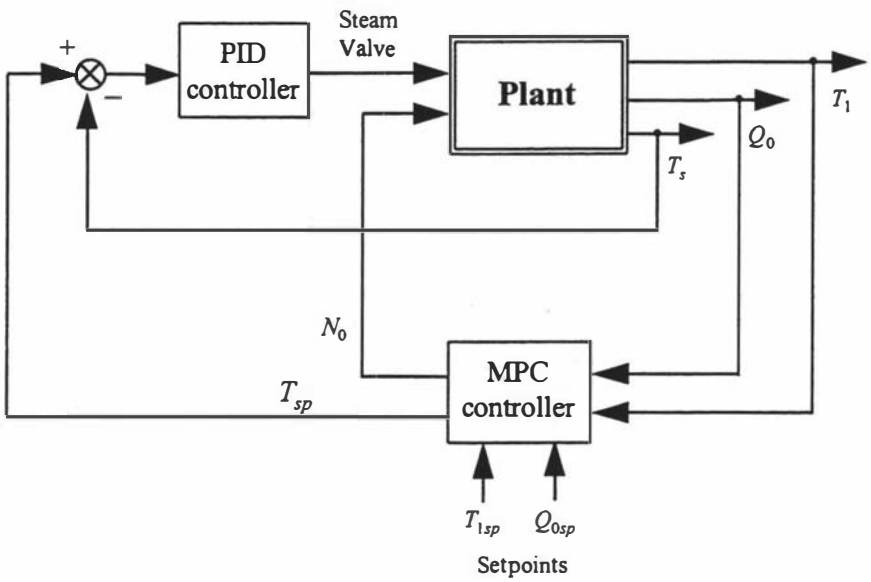


Figure 8-1: Block diagram of the evaporator MPC control strategy.

8.2 Evaporator MPC

The controller simulations were performed with the evaporator analytical model of Chapter 3 assuming the role of the actual plant.

The PI control loops on the concentrate levels were applied to both the 'plant' model and the model embedded in the MPC controller. The controller gains were selected using the Cohen-Coon tuning method [3]. The gains used for the PI(D) controllers (with the independent gains representation, where $k_i = k_p/\tau_i$ and $k_d = k_p \times \tau_d$) are given in Table 8-2.

Table 8-2: PI(D) controller gains

PI Controller	k_p	k_i	k_d
Level 1	-9900	-2400	0
Level 2	-5500	-280	0
Level 3	-2100	-360	0
Steam temperature	4.0	0.1	1

8.2.1 MPC simulation

The MPC strategy used was similar to the DMC method outlined in Chapter 5. The major deviation from the method being the use of the neural network (NN-MPC) and linear ARX (ARX-MPC) models rather than a step response model.

As is the case in DMC separate control and prediction horizons were defined and a disturbance term was used to account for the plant/model mismatch and any unmodelled disturbances.

The MATLAB M-files used to simulate MPC with the evaporator simulation are given in Appendix 11. A brief outline of the MPC algorithm is given below.

Overview of MPC algorithm:

1. *Initialisation*

   At $t = 0$

   Define $\boldsymbol{Q}$, $\boldsymbol{R}$, N_p and N_c

2. *Find optimal set of inputs for* $t = t + 1,\ldots,t + N_c$

   Find $\boldsymbol{u}$

   Which minimises the cost function, Φ

   Subject to constraints on $\boldsymbol{u}$, $\boldsymbol{y}$ and $\Delta\boldsymbol{u}$

3. *Apply first vector of inputs to plant and model at next time step*

   $t = t + 1$

   $\boldsymbol{y}(t) = f_{plant}\{\boldsymbol{u}(t)\}$

   $\hat{\boldsymbol{y}}(t) = f_{model}\{\boldsymbol{u}(t)\}$

4. *Calculate disturbance term*

   $\boldsymbol{d}(t) = \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)$

5. *Return to step 2*

The MPC simulation is run using the script file *MPCrun.m*. This file calls the *MPC.m* function, which in turn calls the *constr.m* optimisation routine. The *constr.m* function requires the performance function and controller model to be defined. The objective function is given in the file *costfun.m* which utilises the required model of the plant.

The optimisation is performed using the constrained optimisation technique Sequential Quadratic Programming (SQP). This is called with the *constr.m* function, which is a standard routine within the MATLAB's *Optimisation Toolbox* [4].

The optimisation attempts to minimise the objective function:

$$\Phi = \sum_{i=1}^{N_p} \boldsymbol{e}(t+i)\, \boldsymbol{Q}\, \boldsymbol{e}(t+i)^{\mathrm{T}} + \sum_{i=1}^{N_c} \Delta\boldsymbol{u}(t+i)\, \boldsymbol{R}\, \Delta\boldsymbol{u}(t+i)^{\mathrm{T}} \tag{8.1}$$

where $$\boldsymbol{e}(t+i) = \boldsymbol{r}(t+i) - \hat{\boldsymbol{y}}(t+i) - \boldsymbol{d}(t), \quad i = 1,\ldots, N_p. \tag{8.2}$$

and $$d(t) = y(t) - \hat{y}(t) \tag{8.3}$$

Terminal weights are used to aid in driving the outputs to the setpoint. This is a measure which approximates to adding a terminal constraint to the optimisation problem. The additional term added to the objective function has the form:

$$\varepsilon_T = \boldsymbol{e}(t+N_p)\, \boldsymbol{Q_T}\, \boldsymbol{e}(t+N_p)^T \tag{8.4}$$

where $\boldsymbol{Q_T}$ is the weighting matrix with larger weights than $\boldsymbol{Q}$.

Constraints were applied to both inputs and outputs. The limits on the velocity of input changes was constrained to 2°C per time step for T_{sp} and 400 rpm per time step for N_0.

The controller model is called at each optimisation step in order to calculate the cost function. Since both the neural network and ARX models have a sampling rate of ten seconds this automatically becomes the sampling rate of the MPC controller.

MPC tuning

Tuning the MPC controller largely consists of trial and error [5].

The prediction horizon is generally taken to be equivalent to the settling time for the controlled outputs. The feed flow has a very short settling time (its time constant is considered negligible) whereas the settling time of the effect temperature is of the order of 2 minutes.

For the MPC strategy developed for the evaporator simulation a prediction horizon (N_p) of 10 time steps and a control horizon (N_c) of 3 time steps were used.

After a great deal of trial and error the following weighting matrices were settled upon:

$$\boldsymbol{Q} = \begin{bmatrix} 10 & 0 \\ 0 & 0.1 \end{bmatrix}, \qquad \boldsymbol{R} = \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix}, \qquad \boldsymbol{Q_T} = \begin{bmatrix} 1000 & 0 \\ 0 & 10 \end{bmatrix}.$$

The apparently heavier weighting on the temperature output and input (first element of matrices) are due to the units used. The temperatures have units of °C where an error of 0.1°C is considered small and 1°C is large whereas the flow and pump speed have units in l/hr and rpm where an error of 1 l/hr and a change of 1 rpm are considered small.

8.2.2 PI Control

For comparison PI control loops were applied to the plant simulation in place of the MPC controller. One controller was used as the master controller for the steam temperature cascade loop and one controlled the feed flowrate through manipulating the feed pump. The PI gains were based on estimates from the Cohen-Coon method. The gains used were:

Table 8-3: PI control gains

	k_p	k_i
Steam temperature cascade controller	0.5	0.01
Feed flowrate controller	5.0	0.02

8.3 Simulation experiments

The control schemes were demonstrated for both setpoint tracking and disturbance rejection problems.

8.3.1 Setpoint tracking

The initial setpoints for the first effect product temperature and the feed flowrate are:

T_{1sp} = 72 °C and Q_{0sp} = 220 l/hr and were altered as shown in Table 8-4.

Table 8-4: Setpoint step changes

Step time	Setpoint change	
(min)	T_{1sp} (°C)	Q_{0sp} (l/hr)
1	72 to 75	220 to 250
8	75 to 72	250 to 200

These setpoint changes represent a reasonably challenging task to control since the two variables are working against each other. With increasing product flowrate the temperature in the effect decreases. Similarly with an increasing effect temperature (and hence pressure) a decrease in the incoming flow occurs. Both these two step changes are simultaneously increasing then decreasing the flowrate and temperature causing a conflict between the two variables.

MPC results

The plots in Figure 8-2 show the responses of the linear ARX (ARX-MPC) and neural network MPC (NN-MPC) schemes to the simultaneous setpoint step changes on T_1 and Q_0. The setpoint changes occur at a time of 1 minute on the plots. Both responses settle down within 4 minutes of the step change.

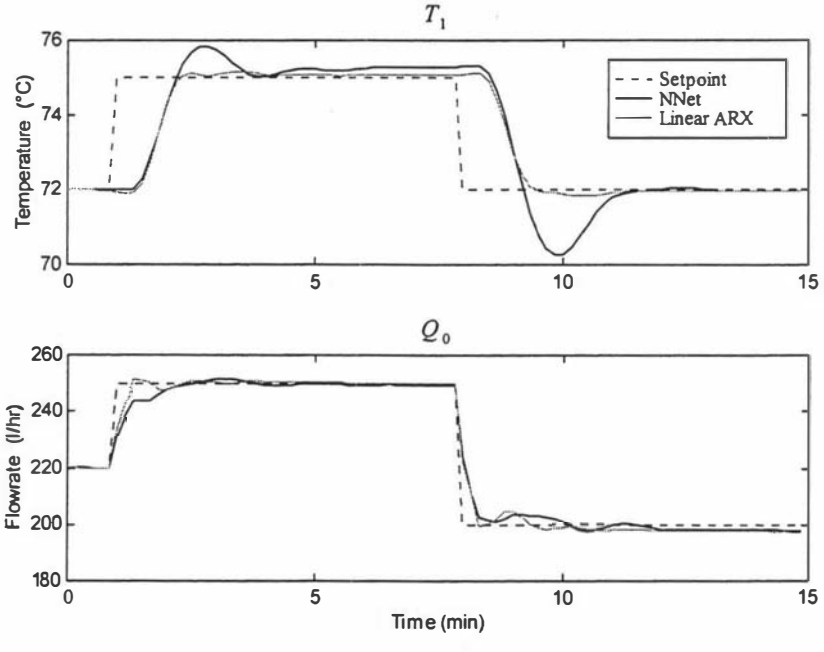


Figure 8-2: MPC setpoint tracking response for step change in T_1 and Q_0 setpoints

Overall the ARX-MPC T_1 response appears better than that of the NN-MPC T_1 response while the Q_0 responses are similar.

The linear ARX model-based controller exhibits much less overshoot and settles down more quickly than the neural network-based controller.

Some offset is apparent in the T_1 steady-state of the NN-MPC simulation after the first step change and both schemes exhibit offset for the Q_0 steady-state after the second step change. Attempts to remove this offset by varying the MPC tuning parameters ($\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{Q_T}$, N_c, N_p) were unsuccessful.

According to [6] offset in the control response is due to the errors between the plant and model and the use of the disturbance term, d (as in equations (8.2) and (8.3)) should remove this effect. However despite the inclusion of the disturbance term the offset remained. Integral-action could be included as an external add-on to the MPC routine. However with the inclusion of integral-action in this way there is no guarantee of the

control acting within the implicit constraints defined for the MPC controller. Problems with integral wind-up and residual integral can also surface.

The plots in Figure 8-3 show the movement of the manipulated variables in order to achieve these responses.

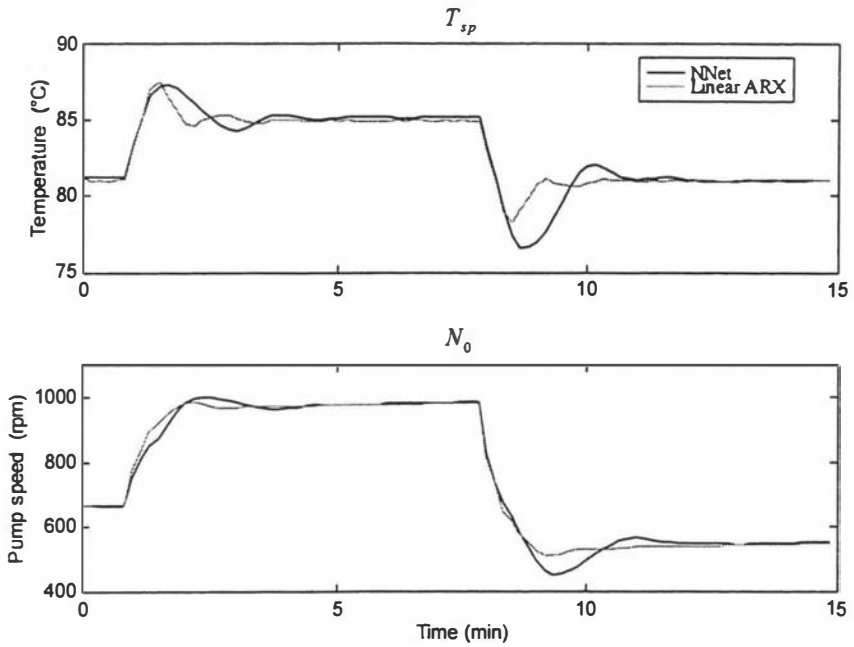


Figure 8-3: MPC control action for setpoint tracking test

The mean-square errors for the responses (Table 8-5) show the improved control attained by the ARX-MPC scheme.

Table 8-5: MSEs for setpoint tracking test

	NN-MPC	ARX-MPC
T_1	0.144	0.115
Q_0	0.006	0.005

PI control results

The dual-PI control simulation underwent a similar setpoint tracking test. This test was required to be longer in duration in order to allow the flowrate to approach the setpoint after the step changes.

The T_1 response (Figure 8-4) shows good control of the temperature however notice that it takes approximately 10 minutes for steady-state to be reached after the step change. The controller response for the feed flowrate (Q_0) is not as good as can be clearly seen; the setpoint has still to be attained 30 minutes after the step change.

The predominant reason for this slow transient response is due to the conflict between the two control loops. It appears that the temperature control loop is dominant over the feed flow controller. The use of higher proportional or integral gains in the feed flow controller leads to erratic or unstable behaviour in the feed pump.

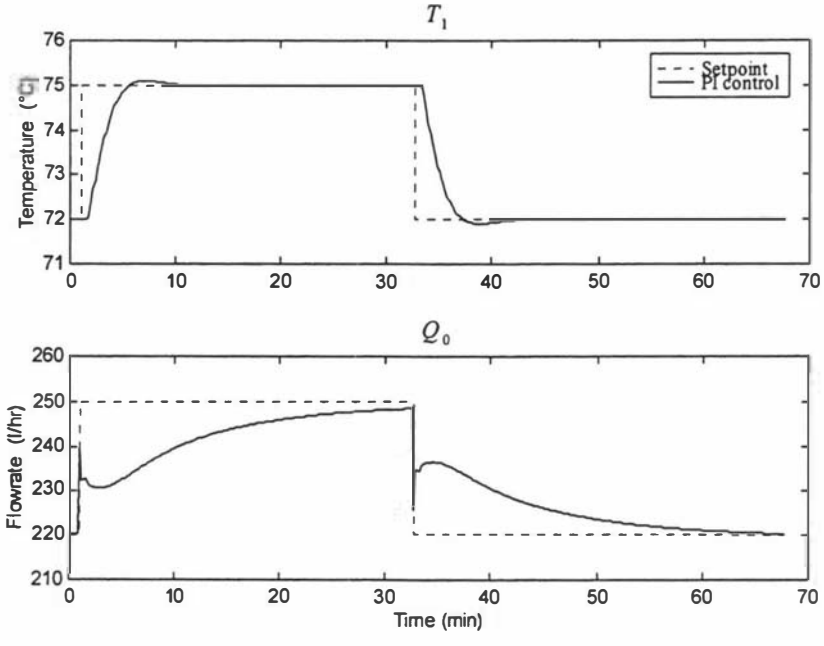


Figure 8-4: PI control setpoint tracking response for step change in T_1 and Q_0 setpoints

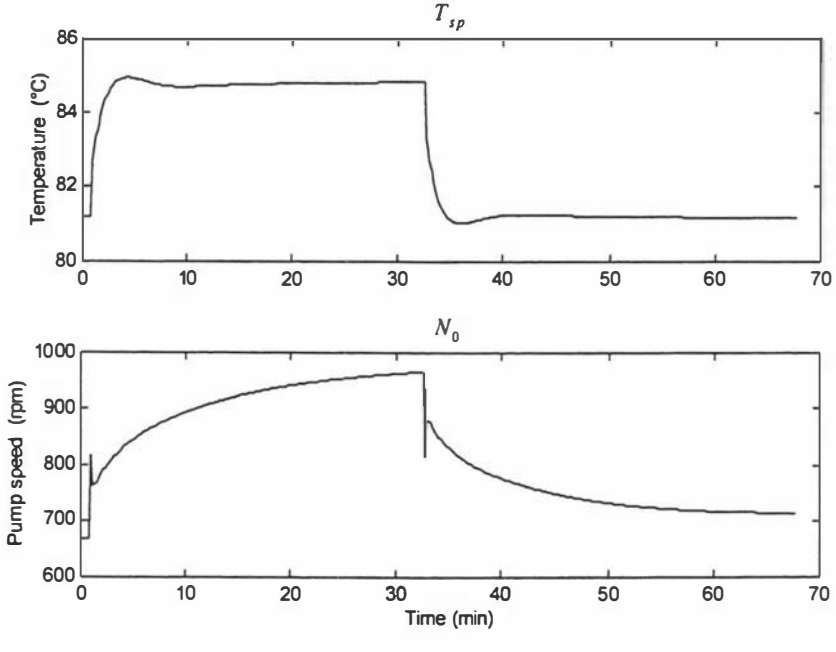


Figure 8-5: PI control action for setpoint tracking problem

8.3.2 Disturbance rejection

To test the disturbance rejection performance of the controllers an impulse disturbance was applied to the steam temperature. This disturbance was used to represent an unexpected increase in the steam supply line pressure which in turn increases the steam flow into the plant. The impulse had an amplitude of +5°C and was applied to the steam temperature at $t = 1$ minute.

The plots in Figure 8-6 show the output responses for the first effect temperature and the feed flowrate for each control scheme with the disturbance applied. It is clear that both the MPC schemes have difficulty in attaining the setpoint and experience problems with steady-state offset. The PI control strategy performs well. The PI control loops do have an advantage in that they have a sampling rate of only 0.5 s which allows a faster response time.

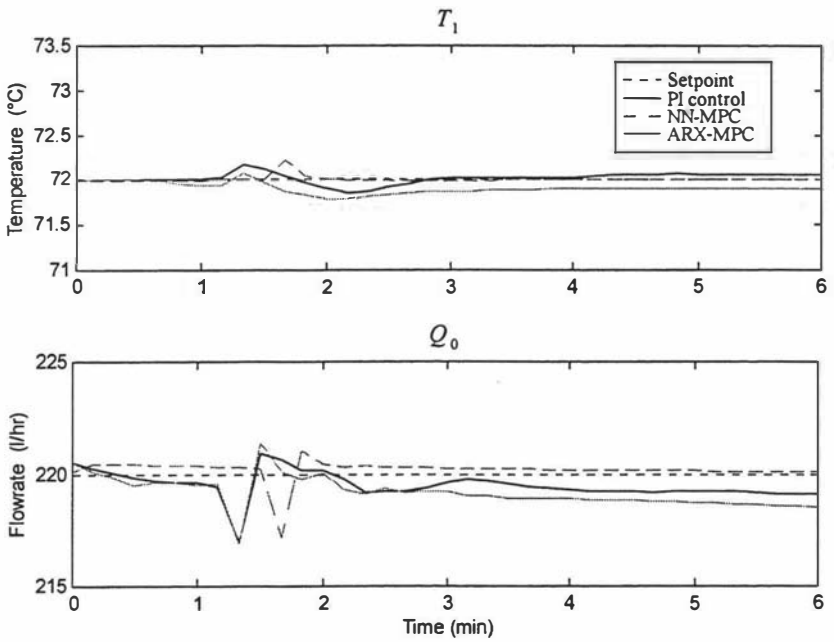


Figure 8-6: Output responses for each control scheme with a steam temperature disturbance

The plots in Figure 8-7 show the control actions required to achieve the above output responses. These plots again illustrate the offset problems encountered by the MPC schemes shown by the increasing feed pump speed. The steady-state offset should be dealt with by the model disturbance term but again appears not to be successful.

Integral action would effectively eliminate any offset problem however as mentioned before this would result in the loss of the inherent constraint handling benefits of MPC.

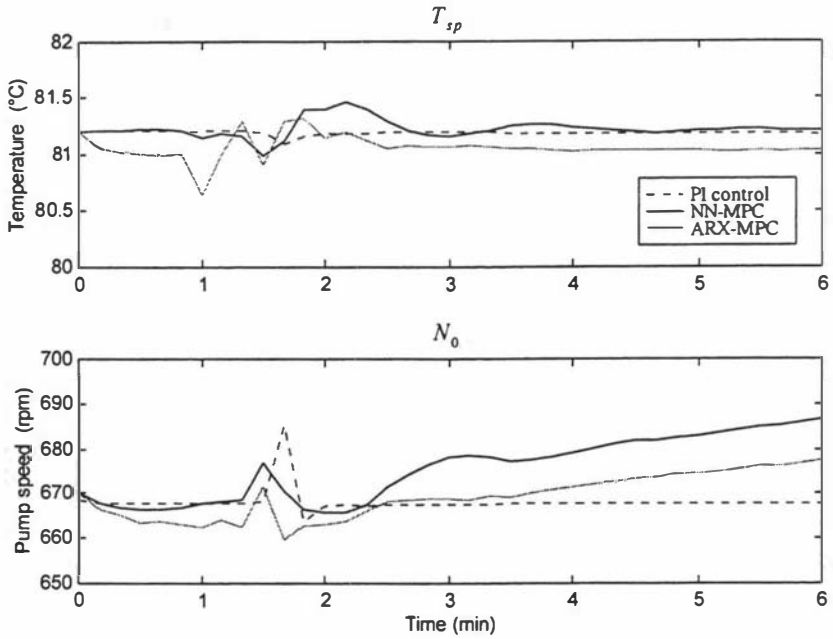


Figure 8-7: Control actions under a steam temperature impulse disturbance

8.4 Conclusions

The application of both linear ARX and neural network MPC strategies has been demonstrated on the evaporator simulation.

The MPC controllers possess a similar control performance although there was a greater overshoot in the NN-MPC setpoint tracking response. The settling time of the MPC schemes were also shown to be significantly quicker than a PI control scheme for setpoint tracking. This is due to the conflict between the two separate PI control loops in achieving the two setpoints whereas the MPC schemes can optimise the actions considering both control objectives.

The performance of the MPC controllers in disturbance rejection was not as good as the PI control scheme. This could be due to the shorter sampling rate used in the PI control loops which allow a quicker response time.

The MPC schemes periods of exhibited steady-state offset for both setpoint tracking and the disturbance rejection tests. Attempts to improve the steady-state performance through the use of a disturbance term in the objective function and varying the tuning parameters were unsuccessful.

8.5 References

[1] **van Wijck, M.P.C.M., Quaak, P. & van Haren, J.J.**, Multivariable control of a four-effect falling-film evaporator. *Food Control*, 1994, vol. 5, pp. 83-89.

[2] **Ricker, N.L., Sim, T. & Cheng, C.-M.**, Predictive control of a multi-effect evaporation system. *Proceedings of the American Control Conference*, 1986, Seattle, USA, pp. 355-359.

[3] **Cohen, G. H. & Coon, G. A.**, Theoretical consideration of retarded control. *Transactions of ASME*, 1953, vol. 75, pp. 827.

[4] *MATLAB Optimisation Toolbox (ver. 1.0c)*, The MathWorks Inc., Natick, MA, USA, 1992.

[5] **Garcia, C.E., Prett, D.M. & Morari, M.**, Model predictive control: theory and practice–a survey. *Automatica*, 1989, vol. 25, pp. 335-348.

[6] **Psichogios, D.C. & Ungar, L.H.**, Direct and indirect model based control using artificial neural networks. *Industrial Engineering Chemistry Research*, 1991, vol. 30, pp. 2564-2573.

9

Conclusion

9.1 Introduction

The primary objective for this thesis was to develop and demonstrate an approach to process system identification using artificial neural networks. The approach combined a number of techniques within the one modelling framework which attempted to simplify network development and provide a method of incorporating an element of transparency into the model. The framework applied was one of employing prior knowledge of the system to create a locally-connected modular model.

The ultimate objective for the model was for use in model predictive control (MPC) and therefore required the capability to predict ahead over a future horizon of arbitrary length. For this reason a dynamic recurrent training methodology was employed for training the modules. A training algorithm combining backpropagation through time and the Levenberg-Marquardt optimisation method was devised.

The modular neural network development method involved:

- the decomposition of the model variables into sub-systems based on prior knowledge of the system;
- structuring of the single-output sub-networks to have locally-connected topology; and
- training of the sub-networks individually using a backpropagation through time technique employing the Levenberg-Marquardt optimisation method (LMTT).

A modular linear regression model with an identical functional form was developed for comparison with the neural network model.

In order to demonstrate the neural network MPC strategy an analytical dynamic model was derived for the evaporator system for use as a simulation tool. This exercise also

proved beneficial in gaining a theoretical appreciation for the process and provided a interesting comparison of differing methods of model development.

A systems approach was employed for the analytical model development and extended the single effect modelling work of Illingworth [1]. The major extensions included re-formulating the model for a feedforward configuration, deriving equations for the preheater and venturi condenser sub-systems and the inclusion of an additional product temperature state variable. The models for three effects and the additional sub-systems were combined to describe the complete evaporator system.

9.2 Conclusions

- The complete analytical model was validated against data from the actual plant. The model was found to perform satisfactorily although it appeared to be not as accurate as the black-box models. However the analytical model possesses other advantages over the data-based models in terms of extrapolating abilities and understanding of the model.
- The modular model approach using locally-connected sub-networks was found to both improve the training and generalisation over that of networks with alternative structures. These alternatives included both fully-connected sub-networks and multi-output networks.
- The use of modular modelling approach was found to improve the transparency of the neural network model somewhat.
- Modular modelling also has benefits in terms of flexibility. Mixed model systems are possible where different techniques are used to model separate sub-sections. It also has positive implications when modifications or extensions are required for the model.
- The neural network model did not perform as well as one might expect compared with the linear model, and in absolute terms. Overall the two modular models were found to exhibit a comparable performance with regard to long-range prediction estimates. This was contrary to what was originally expected since neural networks have the ability to model nonlinear relationships. Theoretically neural networks should perform better for nonlinear data and equally well for linear data. The poorer-than-expected performance could be due to poor training data, training methodology, complex and unobservable process modes or the poor choice of network structures and variables. The most likely reasons are believed to be the sub-optimal solutions from the training method or unobservable process behaviour.

- The application of the modular models was demonstrated within a MPC strategy applied to the analytical model simulation. The control performance was better than that of a PI control scheme for setpoint tracking but was not as good in a disturbance rejection test.

Linear models are commonly used for identification and control of processes such as the evaporator. The results obtained in this thesis reveal that the use of a linear model can indeed produce an adequate process model. The simpler model structure, the ease of analysis and the shorter development effort required for the linear model could outweigh any benefits in terms of accuracy gained through the use of neural networks for such a system. Neural networks may prove to be more superior for highly nonlinear systems such as fermentation or distillation processes.

Despite the lack of evidence of improved performance over linear modelling techniques it is believed that the modular training approach outlined in this thesis has a great deal of potential for the identification of sound, flexible neural network descriptions of process systems.

9.3 Future Work

Due to the broad nature of this thesis future research work could take a number of directions. These relate to further development of both the analytical and neural network models.

9.3.1 Analytical model

Re-coding of the analytical model

The implementation speed of the simulation could be increased through the re-programming of the analytical model into C-code or similar language. The re-coding could also lead to the use of the simulation with software packages other than MATLAB. One possibility could be the linking of the model to the supervisory control software that runs the pilot-plant for on-line comparisons or developing operator simulations.

Inclusion of concentration equations

The inclusion of concentration equations for products with solids content (eg. sugar solutions or milk) would enable the model to be more applicable to industrial processes. In order to perform the model validation, on-line sensors for density would be required on the plant.

Model improvements

The model could be improved by revisiting the equations for the external steam supply temperature so that the nonlinear characteristics of the steam valve are accounted for. An improved model of this system would enhance the performance of the whole model.

9.3.2 Neural network development

Alternative recurrent training methodology

An alternative training method, such as real-time recurrent training [2] or similar, could be applied to training the sub-networks. The LMTT method is not as efficient as other methods however whether an improvement in prediction could be achieved with more efficient training methods is not known.

Training full network model

The development of a training methodology which is capable of training the full evaporator system as a whole would be an interesting extension to this work. It would be expected that the best way to achieve this is by first training the individual sub-networks then combining them before continuing the training of the complete model. This may improve the performance of the network since all the interactions are accounted for.

Again because of its inefficiency, the LMTT is unlikely to be successful for this training approach.

Application of the approach to an alternative system

The application of the modular modelling technique to an alternative process system would be beneficial in terms of formalising the approach. An alternative system with more complex nonlinear behaviour, such as a fermentation or distillation process would provide an interesting study.

9.4 References

[1] **Illingworth, M.I.**, *Dynamic modelling of a three-effect falling-film evaporator.* Final Year Undergraduate Research Project, Dept. Production Technology, Massey University, New Zealand, 1993.

[2] **Williams, R.J. & Zipser, D.**, A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989, vol. 1, pp. 270-280.

Bibliography

Akaike, H., A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 1974, vol. AC-19, pp. 716-723.

Austin, P.C. & Bozin, A.S, Applying dynamic matrix control in the process industries. *Transactions of the Institute of Measurement & Control*. 1996, vol. 18, pp. 32-41.

Baba, N., A new approach for finding the global minimum of error function of neural networks. *Neural Networks*, 1989, vol. 2, pp. 367-373.

Benson, R., Process control-the future. *IEE Computing & Control Engineering Journal*, August 1997, pp. 161-166.

Bequette, B.W., Nonlinear control of chemical processes: a review. *Industrial Engineering Chemistry Research*, 1991, vol. 30, pp. 1391-1413.

Bhat, N. & McAvoy, T.J., Use of neural nets for dynamic modelling and control of chemical process systems. *Computers & Chemical Engineering.*, 1990, vol. 14, pp. 573-583.

Bhat, N., Minderman, P., McAvoy, T. & Wang, N., Modeling chemical process systems via neural computation. *IEEE Control Systems Magazine*, April 1990, vol. 10, pp. 24-30.

Billet, R., *Evaporation Technology*, VCH Publishers, Weinheim, Germany, 1989.

Billings, S.A. & Voon, W.S.F., Correlation based model validity tests for nonlinear models. *International Journal of Control*, 1986, vol. 44, pp. 235-244.

Brennan, J.G., Butters, J.R., Cowell, N.D. & Lilley, A.E.V. *Food Engineering Operations*, 3rd Edition, Elsevier Applied Science, London, 1990.

Chen, Q. & Weigand, W.A., Dynamic optimization of nonlinear processes by combining neural net model with UDMC. *AIChE Journal*, 1994, pp. 1488-1497.

Chen, S., & Billings, S.A., Neural networks for nonlinear dynamic system modeling and identification. *International Journal of Control*, 1992, vol. 56, pp. 319-346.

Chen, S., Billings, S.A. & Grant, P.M., Nonlinear system identification using neural networks. *International Journal of Control*, 1990, vol. 51, pp. 1191-1214.

Chen, S., Cowan, C.F.N. & Grant, P.M., Orthogonal least squares learning algorithm for Radial Basis Function networks. *IEEE Transactions on Neural Networks*, 1991, vol. 2, pp. 302-309.

Clarke, D.W. & Scattolini, R., Constrained receding horizon predictive control. *IEE Proceedings - D, Control Theory Appl.*, 1991, vol. 138, pp. 347-354.

Cohen, G. H. & Coon, G. A., Theoretical consideration of retarded control. *Transactions of ASME*, 1953, vol. 75, pp. 827.

Conlin, J., Peel, C. & Montague, G.A., Modelling pressure drop in water treatment. *Artificial Intelligence in Engineering*, 1997, vol. 11, pp. 393-400.

Cooper, J.R. & Le Fevre, E.J., *Thermophysical Properties of Water Substance*, Edward Arnold Ltd., London, 1986.

Coulson, J.M. & Richardson, J.F., *Chemical Engineering, Vol. 1*, 2nd Edition, Pergamon Press, Oxford, UK, 1970.

Coulson, J.M. & Richardson, J.F. with **Blackhurst, J.R. & Harker, J.H.**, *Chemical Engineering, Vol. 2 - Unit Operations*, 3rd Edition, Pergamon Press, Oxford, UK, 1978.

Cutler, C.R. & Ramaker, B.L., Dynamic matrix control–a computer control algorithm. *Proceedings of AIChE National Meeting.*, 1979, Houston, Texas.

Cybenko, G., Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 1989, vol. 2, pp. 303-314.

Demircioglu, H. & Clarke, D.W., Generalised predictive control with end-point state weighting. *IEE Proceedings - D, Control Theory Appl.*, 1993, vol. 140, pp. 275-282.

Doherty, S.K., Williams, D. & Gomm, J.B., Modelling a highly nonlinear process using an artificial neural network. *Proceedings of the Irish DSP and Control Colloquium*, July 1994, Dublin, Ireland, pp. 45-52.

Draeger, A., Engell, S. & Ranke, H., Model predictive control using neural networks. *IEEE Control Systems Magazine*, October 1995, vol. 15, pp. 61-66.

Fahlman, S. & Lebiere, C., The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, PA, 1991.

Garcia, C.E. & Morari, M., Internal model control – 1. A unifying review and some new results. *Industrial and Engineering Chemistry. Process Design and Development*, 1982, vol. 21, pp. 308-323.

Garcia, C.E., Prett, D.M. & Morari, M., Model predictive control: theory and practice–a survey. *Automatica*, 1989, vol. 25, pp. 335-348.

Gomm, J.B., Lisboa, P.J.G., Williams, D. & Evans, J.T., Accurate multi-step-ahead prediction of non-linear systems using the MLP neural network with spread encoding. *Transactions of the Institute of Measurement & Control*. 1994, vol. 16, pp. 203-213.

Hagan, M.T. & Menhaj, M.B., Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 1994, vol. 5, pp. 989-993.

Hahn, G., Evaporator Design. In: *Concentration and Drying of Foods*, MacCarthy, D. (Ed.), Elsevier Applied Science, London, 1986, pp. 113-131.

Hall, C.W., Farrall, A.W. & Rippen, A.L. (Eds.), *Encyclopedia of Food Engineering*, 2nd Edition, AVI Publishing Co., Westport, Connecticut, USA. 1986.

Hammerstrom, D., Neural networks at work. *IEEE Spectrum*, June 1993, vol. 30, pp. 26-32.

Hammerstrom, D., Working with neural networks. *IEEE Spectrum*, July 1993, vol. 30, pp. 46-53.

Happel, B.L.M. & Murre, J.M.J., Design and evolution of modular neural network architectures. *Neural Networks*, 1994, vol. 7, pp. 985-1004.

Hopfield, J.J., Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 1982, vol. 79, pp. 2554-2558.

Hornik, K., Stinchcombe, M. & White, H., Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989, vol. 2, pp. 359-366.

Hoskins, J.C. & Himmelblau, D.M., Artificial neural network models of knowledge representation in chemical engineering. *Computers & Chemical Engineering*, 1988, vol. 12, pp. 881-890.

Hunt, K.J. & Sbarbaro, D., Neural networks for nonlinear internal model control. *IEE Proceedings - D, Control Theory Appl.*, 1991, vol. 138, pp. 431-438.

Hunt, K.J., Sbarbaro, D., Zbikowski, R. & Gawthrop, P.J., Neural networks for control systems—a survey. *Automatica*, 1992, vol. 28, pp. 1083-1112.

Hush, D.R. & Horne, B.G., Progress in supervised neural networks. *IEEE Signal Processing Magazine*, 1993, vol. 10, pp. 8-39.

Illingworth, M.I., *Dynamic modelling of a three-effect falling-film evaporator.* Final Year Undergraduate Research Project, Dept. Production Technology, Massey University, New Zealand, 1993.

Jacobs, R., Jordan, M., Nowlan, S. & Hinton, G., Adaptive mixture of local experts. *Neural Computation*, 1991, vol. 3, pp. 79-87.

Joerding, W.H. & Meador, J.L., Encoding a priori information in feedforward neural networks. *Neural Networks*, 1991, vol. 4, pp. 847-856.

Kambhampati, C., Delgado, A., Mason, J.D & Warwick, K., Stable receding horizon control based on recurrent networks. *IEE Proceedings - D, Control Theory Appl.*, 1997, vol. 144, pp. 249-254.

Kroll, A.E., The design of jet pumps. *Chemical Engineering Progress*, 1947, vol. 1, pp. 21-24.

Kroschwitz, J.I & Howe-Grant, M. (Eds.), *Kirk-Othmer Encyclopedia of Chemical Technology*, 4th Edition, vol. 9, John Wiley & Sons, New York, USA, 1994.

Lee, P.L., Newell, R.B. & Sullivan, G.R., Generic model control – a case study. *The Canadian Journal of Chemical Engineering*, 1989, vol. 67, pp. 478-484.

Lee, M. & Park, S., A new scheme combining neural feedforward control with model predictive control. *AIChE Journal*, 1992, vol. 38, pp. 193-200.

Leigh, J.R., Neural networks in process control. *IEE Computing & Control Engineering Journal*, May 1992, p. 104.

Leonard, J. & Kramer, M.A., Improvement of the backpropagation algorithm for training neural networks. *Computers and Chemical Engineering*, 1990, vol. 14, pp. 337-341.

Leshno, M., Lin, V.Ya., Pinkus, A. & Schocken, S., Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 1993, vol. 6, pp. 861-867.

Levenberg, K., A method for the solution of certain nonlinear problems in least squares. *Quarterly of Applied Mathematics*, 1944, vol. 2, pp. 164-168.

Lightbody, G. & Irwin, G.W., Direct neural model reference adaptive control. *IEE Proceedings - D, Control Theory Appl.*, 1995, vol. 142, pp. 31-43.

Lin, T., Horne, B.G., Tino, P. & Giles, C.L., Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 1996, vol. 7, pp. 1329-1338.

Ljung, L., *System Identification - Theory for the User*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1987.

McAdams, W.H., *Heat Transmission*, McGraw Hill, New York, 1954

Marquardt, D.W., An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 1963, vol. 11, pp. 431-441.

Mavrovouniotis, M.L. & Chang, S., Hierarchical neural networks. *Computers & Chemical Engineering*, 1992, vol. 16, pp. 347-369.

Mayne, D.Q. & Michalska, H., Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control,* 1990, vol. 35, pp. 814-824.

Michalska, H. & Mayne, D.Q., Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control,* 1993, vol. 38, pp. 1623-1633.

Mills, P.M., Zomaya, A.Y. & Tadé, M.O., Adaptive model-based control using neural networks. *International Journal of Control*, 1994, vol. 60, pp. 1163-1192.

Møller, M.F., A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 1993, vol. 6, pp. 525-533.

Montague, G.A., Morris, A.J. & Willis, M.J., Artificial neural networks: methodologies and applications in process control. In: *Neural Networks for Control and Systems*, Warwick, K., Irwin, G.W. & Hunt, K.J. (Eds.), Peter Peregrinus, Stevenage, UK, 1992, pp. 123-150.

Morris, A.J., Montague, G.A. & Willis, M.J., Artificial neural networks: Studies in process modelling and control. *Chemical Engineering Research & Design (Transactions of the Institution of Chemical Engineers)*, 1994, vol. 72, pp. 3-19.

Nahas, E.P., Henson, M.A. & Seborg, D.E., Nonlinear internal model control strategy for neural network models. *Computers & Chemical Engineering*, 1992, vol. 16, pp. 1039-1057.

Narendra, K.S. & Parthasarathy, K., Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1990, vol. 1, pp. 4-27.

Nguyen, D.H. & Widrow, B., Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, April 1990, vol. 10, pp. 18-23.

Nguyen, D.H. & Widrow, B., Improving the learning speed of 2-layered neural networks by choosing initial values of the of the adaptive weights. *Proceedings of the International Joint Conference of Neural Networks*, July 1990, San Diego, CA., vol. 3, pp. 21-26.

Pearlmutter, B.A., Dynamic recurrent neural networks. Technical Report CMU-CS-90-196, Carnegie Mellon University, Pittsburgh, PA, 1990.

Perry, R.H., Green, D.W. & Maloney, J.O (Eds.). *Perry's Chemical Engineers' Handbook*, 6th Edition, McGraw-Hill Book Co., New York, USA, 1984.

Piché, S.W., Steepest descent algorithms for neural network controllers and filters. *IEEE Transactions on Neural Networks*, 1994, vol. 5, pp. 198-212.

Press, W. H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P., *Numerical recipes in FORTRAN*, 2nd Edition, Cambridge University Press, Cambridge, UK, 1992.

Psaltis, D., Sideris, A. & Yamamura, A.A., A multilayered neural network controller. *IEEE Control Systems Magazine*, April 1988, vol. 8, pp. 17-21.

Psichogios, D.C. & Ungar, L.H., Nonlinear internal model control and model predictive control using neural networks. *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, 1990, pp. 1082-1087.

Psichogios, D.C. & Ungar, L.H., Direct and indirect model based control using artificial neural networks. *Industrial Engineering Chemistry Research*, 1991, vol. 30, pp. 2564-2573.

Psichogios, D.C. & Ungar, L.H., A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 1992, vol. 38, pp. 1499-1511.

Pugmire, R.H., *A neural network based window filter and its training for image processing tasks*. PhD Thesis, Massey University, New Zealand, 1995.

Qin, S.-Z., Su, H.-T. & McAvoy, T.J., Comparison of four neural net learning methods for dynamic system identification. *IEEE Transactions on Neural Networks*, 1992, vol. 3, pp. 122-130.

Quaak, P. & Gerritsen, J.B.M., Modelling dynamic behaviour of multiple-effect falling-film evaporators. In: *Computer Applications in Chemical Engineering*, Bussemaker, H.T. & Iedema, P.D. (Eds.), Elsevier, Amsterdam, 1990, pp. 59-64.

Quaak, P., van Wijck, M.P.C.M. & van Haren, J.J., Comparison of process identification and physical modelling for falling-film evaporators. *Food Control*, 1994, vol. 5, pp. 73-82.

Rawlings, J.B & Muske, K.R, The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 1993, vol. 38, pp. 1512-1516.

Reed, R., Pruning algorithms: A survey. *IEEE Transactions on Neural Networks*, 1994, vol. 4, pp. 740-747.

Richalet, J., Industrial applications of model based predictive control. *Automatica*, 1993, vol. 29, pp. 1251-1274.

Richalet, J., Rault, A., Testud, J.L. & Papon, J., Model predictive heuristic control: applications to industrial processes. *Automatica*, 1978, vol. 14, pp 413-428.

Ricker, N.L., Sim, T. & Cheng, C.-M., Predictive control of a multi-effect evaporation system. *Proceedings of the American Control Conference*, 1986, Seattle, USA, pp. 355-359.

Ronco, E. & Gawthrop, P., Modular neural networks: a state of the art. Technical Report : CSC-95026, University of Glasgow, UK, 1995.

Rosenblatt, F., The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958, vol. 65, pp. 386-408.

Rouhani, R. & Mehra, R.K., Model algorithmic control (MAC); basic theoretical properties. *Automatica*, 1982, vol. 18, pp 401-414.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J., Learning internal representations by error propagation. In: *Parallel Distributed Processing*, vol. 1, Rumelhart, D.E. & McClelland, J.L. (Eds.), MIT Press, Cambridge, MA, USA, 1986, pp. 318-362.

Russell, N.T. & Bakker, H.H.C., Modular modelling of an evaporator for long-range prediction. *Artificial Intelligence in Engineering*, 1997, vol. 11, pp. 347-355.

Saint-Donat, J., Bhat, N. & McAvoy, T.J., Neural net based model predictive control. *International Journal of Control*, 1991, vol. 54, pp. 1453-1468.

Sbarbaro-Hofer, D., Neumerkel, D. & Hunt, K., Neural control of a steel rolling mill. *IEEE Control Systems Magazine*, June 1993, vol. 13, pp. 69-75.

Scokaert, P.O.M. & Clarke, D.W., Stabilising properties of constrained predictive control. *IEE Proceedings - D, Control Theory Appl.*, 1994, vol. 141, pp. 295-304.

Shewchuk, J.R., *An introduction to the conjugate gradient method without the agonizing pain*, Edition 1¼, Carnegie Mellon University, Pittsburgh, PA, USA, 1994.

Sistu, P.B. & Bequette, B.W., Nonlinear Model-Predictive Control: Closed-loop stability analysis. *AIChE Journal*, 1996, vol. 42, pp. 3388-3402.

Smith, R. A., Theory and design of simple ejectors. Chpt. 13 of *Some Aspects of Fluid Flow*, Edward Arnold & Co., 1951, pp. 229-241.

Solis, F.J. & Wets, J.B., Minimization by random search techniques. *Mathematics of Operations Research*, 1981, vol. 6, pp. 19-30.

Special issues on neural networks. *IEEE Control Systems Magazine*, April 1988, vol. 8; April 1989, vol. 9; April 1990, vol. 10.

Stapper, H.L., Control of an evaporator and spray drier using feedback and ratio feedforward controllers. *New Zealand Journal of Dairy Science and Technology*, 1979, vol. 14, pp. 241-257.

Stephan, K., *Heat Transfer in Condensation and Boiling*, Springer-Verlag, Berlin, Germany 1992.

Su, H.-T., McAvoy, J. & Werbos, P., Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial Engineering Chemistry Research*, 1992, vol. 31, pp. 1338-1352.

Thompson, M.L. & Kramer, M.A., Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*, 1994, vol. 40, pp. 1328-1340.

Turner, P., Montague, G. & Morris, J., Nonlinear and direction-dependent dynamic process modelling using neural networks. *IEE Proceedings - D, Control Theory Appl.*, 1996, vol. 143, pp. 44- 48.

van Wijck, M.P.C.M., Quaak, P. & van Haren, J.J., Multivariable control of a four-effect falling-film evaporator. *Food Control*, 1994, vol. 5, pp. 83-89.

Warwick, K., Irwin, G.W. & Hunt, K.J., (Eds.), *Neural Networks for Control and Systems*, Peter Peregrinus, Stevenage, UK, 1992.

Werbos, P.J., Backpropagation through time: what it is and how to do it. *Proceedings of the IEEE*, 1990, vol. 78, pp. 1550-1560.

Widrow, B.& Hoff, M.E., Adaptive switching circuits. *1960 IRE Western Electric Show and Convention Record*, August 1960, part 4, pp. 96-104.

Widrow, B & Lehr, M.A., 30 years of adaptive neural networks: perceptron, madaline and backpropagation. *Proceedings of the IEEE*, 1990, vol. 78, pp. 1415-1442.

Williams, R.J. & Zipser, D., A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989, vol. 1, pp. 270-280.

Willis, M.J., Di Massimo, C., Montague, G.A., Tham, M.T. & Morris, A.J. Artificial neural networks in process engineering. *IEE Proceedings - D, Control Theory Appl.*, 1991, vol. 138, pp. 256-266.

Willis, M.J., Montague, G.A. & Morris, A.J., Modelling of industrial processes using artificial neural networks. *IEE Computing & Control Engineering Journal*, 1992, vol. 3, pp. 113-117.

Winchester, J., Marsh, C. & Bakker, H., Analytical dynamic modelling of falling-film evaporators. *Proceedings of Chemica '97 Conference*, 29 September - 1 October 1997, Rotorua, New Zealand, pp. PC49 - PC56.

Wright, G.T. & Edgar, T.F., Nonlinear model predictive control of a fixed-bed water-gas shift reactor: an experimental study. *Computers & Chemical Engineering*, 1994, vol. 18, pp. 83-102.

You, Y. & Nikolaou, M., Dynamic process modeling with recurrent neural networks. *AIChE Journal*, 1993, vol. 39 pp. 1654-1667.

Appendices

Contents

Appendix 1

Glossary of Analytical Model Symbols

Key Symbols Used			
A	Area (m^2)	P	Pressure (kPa)
c	Specific heat capacity J/(kg.K)	q	Heat flow (W)
d, D	Diameter (m)	Q	Volumetric flow (m^3/s)
h	Height (m)	ρ	Density (kg/m^3)
l	Length (m)	t, τ	Time (s)
L	Liquid height (m)	T	Temperature (°C)
λ	Latent heat (kJ/kg)	U	Overall heat transfer coefficient ($W/m^2.K$)
m	Mass flow (kg/s)	v	Velocity (m/s)
M	Mass (kg)	V	Volume (m^3)
N	Pump speed (rpm)	w	Dry mass fraction (kg/kg)

Symbol	Description	Unit
a_P	Pump flow characteristic constant	Pa/rpm^2
A_c	Area of orifice for the condensate flows	m^2
A_d	Area of liquid surface on the distribution plate	m^2
A_e	Heat transfer area of evaporation tube	m^2
A_h	Area of holes in the distribution plate	m^2
A_l	Area of liquid surface of the level at base of effect	m^2
A_p	Cross-sectional area of the standard pipework	m^2
A_{ph}	Heat transfer area in the preheater	m^2
A_s	Surface area of the evaporation tube sub-system	m^2
A_{sl}	Surface area of the level sub-system	m^2
A_{sp}	Surface area of the pipework between effects	m^2
A_{sph}	Surface area of preheater	m^2
A_{sv}	Surface area of venturi condenser	m^2
A_T	Cross-sectional area of the feed tank	m^2
A_t	Cross-sectional area of diffuser throat in the condenser	m^2
A_v	Cross-sectional area of projection cone throat in the condenser	m^2
α	Ratio of water jet area to projection cone throat area in the condenser	–
β	Ratio of diffuser area and projection cone area in the condenser	–
c_j	Specific heat capacity of condenser water jet	J/(kgK)
c_p	Specific heat capacity of liquid in the effect	J/(kgK)
c_{ph}	Specific heat capacity of liquid in the preheaters	J/(kgK)

Symbol	Description	Unit
C_N	Flow characteristic constant for the distribution nozzle	$m^3/(Pa^{½}s)$
C_V	Flow characteristic constant for the manual outlet valve	$m^3/(Pa^{½}s)$
C_{vc}	Flow characteristic constant for the condensate streams	$kg/(Pa^{½}s)$
d	Internal diameter of standard pipework	m
d_p	Internal diameter of pipework in the preheaters	m
D	External diameter of standard pipework	m
D_p	External diameter of pipework in the preheaters	m
f	Fanning friction factor	–
g	Gravitational acceleration constant	m/s^2
h_c	Height of the condenser discharge point	m
h_N	Height of the distribution nozzle	m
k_d	Derivative constant for steam PID controller	–
k_i	Integral constant for steam PID controller	–
k_p	Proportional constant for steam PID controller	–
K_c	Friction constant for condensate flow from effect shells	–
K_f	Constant for the pressure drop due to friction through the condenser	–
K_h	Friction constant for flow through distribution plate	–
l_e	Equivalent length of pipework between effects (considering fixtures and fittings)	m
l_f	Equivalent length of pipework in feed stream (considering fixtures and fittings)	m
l_p	Equivalent length of pipework in the preheaters (considering fixtures and fittings)	m
L	Height of the liquid level at the base of the effect	m
L_d	Height of liquid above the distribution plate	m
L_T	Height of liquid in feed tank	m
λ	Latent heat of vaporisation of product in evaporation tube	J/kg
λ_{phi}	Latent heat of condensation of vapour in shell of preheater i	J/kg
m_c	Mass flow of total condensate entering the condenser	kg/s
m_{ci}	Mass flow of condensate from the shell of effect i	kg/s
m_{cphi}	Mass flow of condensate from the shell of preheater i	kg/s
m_j	Mass flow of the jet in the venturi condenser	kg/s
m_v	Mass flow of vapour entering the condenser	kg/s
m_{vent}	Mass flow of liquid exiting the venturi condenser	kg/s
m_{vph1}	Mass flow of vapour from preheater 1	kg/s
M	Mass of liquid product in the evaporation tube	kg
M_{cph1}	Mass of condensate produced in the first preheater shell per residence time	kg (per τ_{ph})
M_l	Mass of liquid product in the effect level	kg
M_v	Mass of vapour produced in the evaporation tube per residence time	kg (per τ_e)
N	Effect pump speed	rpm
N_0	Feed pump speed	rpm
P_a	Ambient pressure	Pa
P_e	Pressure in the evaporation tube	Pa
P_f	Pressure in the evaporation tube of the downstream effect	Pa
P_v	Pressure in the shell of the effect	Pa
P_{vent}	Suction pressure produced by the venturi condenser	Pa
Δp_r	Pressure rise in the venturi condenser	Pa

Symbol	Description	Unit
ΔP_c	Pressure drop through condensate orifice	Pa
ΔP_{fric}	Pressure loss due to friction in pipework	Pa
ΔP_N	Pressure loss due to distribution nozzle	Pa
ΔP_P	Pressure loss due to pump	Pa
q_c	Heat supplied by the condensate stream into the condenser	W
q_i	Heat supplied from the product stream into the effect	W
q_{loss}	Heat loss to the surroundings from the evaporation tube system	W
q_{lossl}	Heat loss to the surroundings from the level system	W
q_{lossp}	Heat loss to the surroundings from the pipe between the effects	W
q_{lossv}	Heat loss to the surroundings from the venturi condenser	W
q_{phi}	Heat transfer in preheater i	W
q_{shell}	Heat flow into the evaporation tube from shell	W
q_t	Heat supplied by product flow into effect level	W
q_{tube}	Heat flow into the evaporation tube of the downstream effect	W
q_v	Heat supplied by the vapour stream into the condenser	W
Q_d	Product flowrate through the distribution plate	m^3/s
Q_e	Product flowrate from the evaporation tube	m^3/s
Q_i	Product flowrate into an effect	m^3/s
Q_{in}	Product flowrate into the feed tank	m^3/s
Q_p	Product flowrate out of the effect	m^3/s
Q_0	Feed flowrate	m^3/s
ρ_c	Density of condensate entering the condenser	kg/m^3
ρ_d	Density of product in distribution plate sub-system	kg/m^3
ρ_e	Density of product leaving the evaporation tube	kg/m^3
ρ_i	Density of product entering the effect	kg/m^3
ρ_{in}	Density of product into feed tank	kg/m^3
ρ_j	Density of water jet in the condenser	kg/m^3
ρ_l	Density of product out of the effect level	kg/m^3
ρ_p	Density of product flowing out of the effect	kg/m^3
ρ_{phi}	Density of product in preheater i	kg/m^3
ρ_T	Density of product in feed tank	kg/m^3
ρ_v	Density of vapour entering the condenser	kg/m^3
$\overline{\rho}$	Estimate of product density in the product transport system	kg/m^3
$\overline{\rho}_f$	Estimate of product density in the feed system	kg/m^3
t	Time	s
t_s	Sampling time for the discrete first order equation for the preheaters	s
T_0	Temperature of product leaving the feed tank	°C
T_a	Ambient temperature	°C
T_c	Temperature of the condensate entering the condenser	°C
T_e	Temperature of product in the evaporation tube	°C
T_p	Temperature of product leaving the effect	°C
T_{in}	Temperature of product into feed tank	°C
T_l	Temperature of product in the effect level	°C

Symbol	Description	Unit
T_s	Temperature of the external steam supply	°C
T_{shell}	Temperature of the vapour in the shell of the effect	°C
T_{tube}	Temperature of the product in the downstream evaporation tube	°C
T_{vent}	Temperature of the exit stream from the venturi condenser	°C
$\overline{T}_{ph}$	Steady-state temperature of the product exiting a preheater	°C
ΔT_{phi}	Log-mean temperature difference in preheater *i*	°C
τ_c	Time constant for the steam valve system	s
$\tau_{c(ph)}$	Time constant for the first order preheater systems	s
τ_e	Residence time through the evaporation tube	s
τ_f	Product transport delay between feed tank and first effect	s
τ_p	Product transport delay between effects	s
τ_{ph}	Product transport delay through a single preheater	s
U_e	Heat transfer coefficient for the evaporation tube	W/(m^{2}K)
U_{loss}	Heat transfer coefficient for heat loss to surroundings from tubes	W/(m^{2}K)
U_{lossl}	Heat transfer coefficient for heat loss to surroundings from liquid level	W/(m^{2}K)
U_{lossp}	Heat transfer coefficient for heat loss to surroundings from pipework	W/(m^{2}K)
U_{lossph}	Heat transfer coefficient for heat loss to surroundings from preheater	W/(m^{2}K)
U_{lossv}	Heat transfer coefficient for heat loss to surroundings from the condenser	W/(m^{2}K)
U_{ph}	Heat transfer coefficient for heat transfer in the preheater	W/(m^{2}K)
v_e	Fluid velocity into the liquid level	m/s
v_p	Fluid velocity at the discharge of the effect	m/s
v_T	Fluid velocity at the feed tank	m/s
v_0	Fluid velocity entering effect 1	m/s
V_l	Volume of liquid in the effect level	m^3
V_f	Volume of the pipework in feed system	m^3
V_p	Volume of the pipework between the effects	m^3
w_d	Product concentration through the distribution plate	kg/kg
w_e	Product concentration out of the evaporation tube	kg/kg
w_i	Product concentration in to the effect	kg/kg
w_{in}	Product concentration into the feed tank	kg/kg
w_l	Product concentration out of the effect level	kg/kg
w_p	Product concentration out of the effect	kg/kg
w_T	Product concentration out of the feed tank	kg/kg
w_0	Product concentration into the first effect	kg/kg

Appendix 2

Venturi Condenser Pressure Equation

Formulation of the condenser suction pressure equation

Recall equation (3.69) from Chapter 3, which defines the suction pressure produced in the venturi;

$$p_0 = \frac{A_t}{A_v}(P_a - \rho_3 g h_c - \frac{\rho_2 v_2^2}{2}) + \frac{(\rho_v v_v^2 + \rho_c v_c^2)}{2} \quad \ldots$$

$$\ldots \quad - \frac{(m_j v_j + m_v v_v + m_c v_c - m_{vent} v_2)}{A_v} + \frac{K_f}{2}\rho_2 v_2^2 . \qquad \text{(A2.1)}$$

Recall also the area ratios

$$\alpha = \frac{A_j}{A_v} \qquad \text{(A2.2)}$$

and

$$\beta = \frac{A_t}{A_v} . \qquad \text{(A2.3)}$$

We can simplify equation (A2.1) if we assume that $\rho_2 = \rho_3 = \rho_j$, since the driving fluid flowrate is much greater than the flows of the entrained fluids.

Also if we assume that the velocity pressure for the two entrained flows are equal

$$\frac{1}{2}\rho_c v_c^2 = \frac{1}{2}\rho_v v_v^2 \qquad \text{(A2.4)}$$

so that

$$\frac{1}{2}(\rho_c v_c^2 + \rho_v v_v^2) = \rho_c v_c^2 \qquad \text{(A2.5)}$$

we are able to rewrite equation (A2.1) as

$$p_0 = \beta(P_a - \rho_j g h_c - \frac{\rho_j v_2^2}{2}) + \rho_c v_c^2 - \frac{(m_j v_j + m_v v_v + m_c v_c - m_{vent} v_2)}{A_v} + \frac{K_f}{2}\rho_j v_2^2$$

...(A2.6)

The next step is to replace the velocity terms in equation (A2.6).

The velocity, v, of a stream can be described by

$$v = \frac{m}{\rho A}$$

where m is the mass flow of the stream, ρ the stream density and A the cross-sectional area through which it travels.

Using this equation the following velocities can be defined

$$v_j = \frac{m_j}{\rho_j A_j} = \frac{m_j}{\rho_j \alpha A_v} \tag{A2.7}$$

$$v_2 = \frac{m_j + m_c + m_v}{\rho_2 A_t} = \frac{m_{vent}}{\rho_j \beta A_v} \tag{A2.8}$$

From equation (A2.4) we can define the ratio of the vapour and condensate velocities as

$$\frac{v_v}{v_c} = \sqrt{\frac{\rho_c}{\rho_v}}. \tag{A2.9}$$

We can then write

$$\frac{m_v \rho_c A_{condensate}}{m_c \rho_v A_{vapour}} = \sqrt{\frac{\rho_c}{\rho_{vc}}}. \tag{A2.10}$$

If we define R as the ratio of the cross-sectional areas for the two flows

$$R = \frac{A_{vapour}}{A_{condensate}} = \frac{m_v}{m_c}\sqrt{\frac{\rho_c}{\rho_v}} \tag{A2.11}$$

and noting that the vapour and condensate streams fill the annulus area around the driving fluid jet to give

$$A_{vapour} + A_{condensate} = A_v - A_j = A_v(1-\alpha) \tag{A2.12}$$

we can obtain

$$A_{condensate} = \frac{A_v(1-\alpha)}{R+1} \tag{A2.13}$$

and

$$A_{vapour} = \frac{A_v(1-\alpha)R}{R+1}. \tag{A2.14}$$

We can then formulate expressions for the vapour and condensate velocities.

$$v_c = \frac{m_c}{\rho_c(1-\alpha)A_v}(R+1) \tag{A2.15}$$

$$v_v = \frac{m_v}{\rho_v(1-\alpha)A_v}\frac{(R+1)}{R} \tag{A2.16}$$

If the velocity terms given in equations (A2.7), (A2.8), (A2.15) and (A2.16) are substituted into equation (A2.6) we get

$$p_0 = \beta(P_a - \rho_j g h_c) - \frac{m_{vent}^2}{2\rho_j \beta A_v^2} + \frac{m_c^2 (R+1)^2}{\rho_c (1-\alpha)^2 A_v^2} - \frac{m_j^2}{\rho_j (1-\alpha) A_v^2} \cdots$$

$$\cdots - \frac{m_v^2 (R+1)}{\rho_v (1-\alpha) A_v^2 R} - \frac{m_c^2 (R+1)}{\rho_c (1-\alpha) A_v^2} + \frac{m_{vent}^2}{\rho_j \beta A_v^2} + \frac{K_f m_{vent}^2}{2\rho_j \beta^2 A_v^2} \qquad \text{(A2.17)}$$

By grouping common factors and re-naming the condenser suction pressure as P_{vent}, we arrive at

$$P_{vent} = \beta(P_a - \rho_j g h_c) - \frac{m_j^2}{\rho_j \alpha A_v^2} + \frac{(K_f + \beta) m_{vent}^2}{2\rho_j \beta^2 A_v^2} + \frac{R+1}{(1-\alpha) A_v^2} \left(\frac{m_c^2 (\alpha + R)}{\rho_c (1-\alpha)} - \frac{m_v^2}{\rho_v R} \right). \qquad \text{...(A2.18)}$$

Example condenser pressure calculation

Under standard operating conditions the evaporation rate achieved by the evaporator is approximately 40 kg/hr. We can assume that the mass of vapour condensing in the first preheater is around 5 kg/hr and the vapour flow into the condenser is approximately 10 kg/hr. Typically the driving fluid into the jet condenser has a flowrate in the region of 1080 kg/hr.

With the above assumed values and under normal operating temperatures we get flows with the following properties entering the venturi system:

Table A2-1: Venturi flows and temperatures

Flow	m (kg/s)	Temperature (°C)	Density (kg/m^3)
Driving fluid	0.3	18	1000
Condensate	16.7×10^{-3}	70	978
Vapour	2.80×10^{-3}	45	6.54×10^{-2}
Output	0.3195	T_{vent}	~ 1000

With this information the ratio of the cross-sectional areas for the vapour and condensate flows, R, can be calculated (using equation (A2.11)):

$$R = \frac{m_v}{m_c} \sqrt{\frac{\rho_c}{\rho_v}} = \frac{2.8}{16.7} \sqrt{\frac{978}{6.54 \times 10^{-2}}} = 20.503$$

The diameter of the projection cone at point ① in Figure 3-11 is 21.8 mm and the diffuser throat diameter is 5.5 mm, which gives

$$\beta = \left(\frac{5.5}{21.8} \right)^2 = 0.0637 .$$

In practice the jet does not spread out very much and maintains a fairly constant diameter through the venturi. This is due to the high supply pressure of the driving water and the fact that the jet flows vertically downwards. If the jet diameter is assumed to be equal to the nozzle diameter of 3 mm, we get

$$\alpha = \left(\frac{3}{21.8}\right)^2 = 0.0189 .$$

With a diameter of 21.8 mm, A_v is 3.73×10^{-4} m^2. The height of the venturi discharge point above the ground (h_c,) is 1.8 m.

Smith [1] states that generally the friction constant, K_f, has a value between 0.3 and 0.5. Using the above information, and taking K_f = 0.3 and P_a = 101.3 kPa, the suction pressure for the evaporator venturi condenser can be estimated with equation (A2.18)

Calculating each of the four terms in equation (A2.18) separately we can observe that for this example, the fourth term is insignificant and the second and third terms effectively balance each other.

First term: $X_1 = 5.324 \times 10^3\ Pa$

Second term: $X_2 = -34.112 \times 10^3\ Pa$

Third term: $X_3 = 32.883 \times 10^3\ Pa$

Fourth term: $X_4 = 18.622\ Pa$

Adding them together we get

$$P_{vent} = X_1 + X_2 + X_3 + X_4 = 4.113kPa .$$

Additionally, the pressure rise in the condenser can be calculated using equation (3.79) and gives:

$$\Delta P_r = 101.3 \times 10^3 - 1000 \times 9.81 \times 1.8 - 4.113 \times 10^3 = 79.529kPa .$$

Appendix 2 References

[1] **Smith, R. A.**, Theory and design of simple ejectors. Chpt. 13 of *Some Aspects of Fluid Flow*, Edward Arnold & Co., 1951, pp. 229-241.

Appendix 3

Calculation of Model Parameters

Contents

Appendix 3(*a*)

Geometrical parameters

A_d The surface area of the liquid above the distribution plate is a piecewise function based on the height of the liquid, L_d. The function as given in Illingworth [1] is as follows:

$$A_d = \begin{cases} 3.88\times10^{-3} & L_d \le 0.026 \\ 3.88\times10^{-3} + 0.06\sqrt{0.0235^2 - (0.0495 - L_d)^2} & 0.026 \le L_d \le 0.07 \end{cases}$$

A_h The three holes in the distribution plate each have a diameter of 10 mm.

$$A_h = 3\times\frac{\pi}{4}\left(10\times10^{-3}\right)^2 = 2.36\times10^{-4}m^2 .$$

A_e The evaporation tubes are of length 2.9 m, outer diameter 38 mm (effect 1 and 2) and 25 mm (effect 3) and have wall thickness of 1.6 mm. For the heat transfer area the mean of the internal and external diameter is used, 36.4 mm (effect 1 and 2) and 23.3 mm (effect 3). Therefore:

$$A_1 \text{ and } A_2 = 2.9\times\pi\times0.0364 = 0.3313m^2$$

$$A_3 = 2.9\times\pi\times0.0233 = 0.2137m^2$$

A_l As for A_d the surface area of the effect level is a piecewise function based on the value of the level, L. From the plant geometry the following function can be determined:

$$A_l = \begin{cases} 1.792\times10^{-3}m^2 & 0 \le L \le 1.02 \\ 2.168\times10^{-3}m^2 & 1.02 \le L \le 1.06 \\ \frac{\pi}{4}(0.0218^2 + (0.5(L-1.06)+0.0478)^2)m^2 & 1.06 \le L \le 1.11 \\ 4.536\times10^{-3}m^2 & 1.11 \le L \le 1.39 \\ \frac{\pi}{4}(0.0728^2 + (k\times(L-1.39)+0.0218)^2)m^2 & 1.39 \le L \le 1.415 \\ \frac{\pi}{4}(0.0728^2 + (k\times0.025+0.0218)^2)m^2 & 1.415 \le L \le 1.6 \end{cases}$$

Where the constant, k is equal to 3.04 for effects 1 and 2, while for effect 3, k is 5.95, since the dimensions for the third effect are different from the other two.

D, d The external diameter of the standard pipework is 25 mm and with 1.6 mm thick walls, the internal pipe diameter is 21.8 mm.

D_p, d_p The external diameter of the preheater pipework is 12.7 mm and internal diameter is 9.5 mm.

A_{ph} The length of the preheater tube is 5.7 m and the mean of the internal and external diameters is 11.1 mm. The heat transfer area in the preheater is therefore:

$$A_{ph} = \pi \times 5.7 \times 0.0111 = 0.1988 m^2 .$$

A_s The surface area of the evaporation tube sub-system includes the portion of the effect and the separator which are above the liquid level, the pipe through which the vapour flows to the next effect and the steam jacket of the next effect downstream. The heat loss from the current effect's steam jacket is accounted for in the calculation of the heat losses for the upstream effect. If we assume a level of 1.5 m (full-scale) then the tubes which make up the area of interest and their diameters are as follows:

Table A3-1: Areas that make up A_s

Pipe Diameter	***Effect 1 and 2***		***Effect 3***	
(mm)	*Length (m)*	*Area (m^2)*	*Length (m)*	*Area (m^2)*
25	2.5	0.1963	2.5	0.1963
38	0.162	0.0193	0.162	0.0193
76	0.3	0.0716	0.3	0.0716
76	2.9	0.6924	2.9	0.6924
101	0.25	0.0794	–	–
152	–	–	0.25	0.1194
		$A_s = 1.059$		$A_s = 1.099$

A_{sl} If an effect level of 1.5 m is assumed then the surface area of the level sub-system is made up of the following portions:

Table A3-2: Areas that make up A_{sl}

Pipe Diameter	***Effect 1 and 2***		***Effect 3***	
(mm)	*Length (m)*	*Area (m^2)*	*Length (m)*	*Area (m^2)*
25	0.6	0.0471	0.6	0.0471
50	1.2	0.1885	1.2	0.1885
76	0.4	0.0955	0.4	0.0955
101	0.1	0.0317	–	–
152	–	–	0.1	0.0477
		$A_{sl} = 0.363$		$A_{sl} = 0.379$

A_{sp} The length of the pipework between effects is approximately 7 m for effects 1 and 2. For effect 3 the length of pipe to the outlet point is approximately 12 m. With external diameter of 25 mm this gives surface areas of:

Effect 1 and 2: $A_{sp} = 0.550 m^2$ and

Effect 3: $A_{sp} = 0.942 m^2$.

A_{sph} The preheater shell has a height of 2 m and diameter of 101 mm. Considering the length of the shell and the two ends we get a surface area of:

$$A_{sph} = 2 \times \frac{\pi}{4} \times 0.101^2 + 2 \times \pi \times 0.101 = 0.65 m^2$$

A_{sv} The surface area of venturi condenser can be calculated in a similar manner:

$$A_{sv} = 2 \times \frac{\pi}{4} \times 0.101^2 + 0.28 \times \pi \times 0.101 = 0.105 m^2$$

A_T The diameter of the feed tank is 450 mm, giving a cross-sectional area of 0.159 m^2.

A_v The diameter of the throat of a condenser projection cone is 21.8 mm, giving a cross-sectional area of 3.73×10^{-4} m^2.

α The area ratio of the diffuser throat to the projection cone in the condenser is 0.0189 as calculated in Appendix 2.

β The area ratio of the venturi jet to the projection cone in the condenser is 0.0637 as calculated in Appendix 2.

l_e The pipework between effect 1 and 2 and between effect 2 and 3 are both approximately 7 m in length. These paths also include four elbows. The pipework between effect 3 and the discharge point is approximately 12 m with 9 elbows and a tee joint. To allow for these fittings an equivalent line length is calculated. According to Coulson *et al.* [2] elbows are equivalent to 40 pipe diameters and tee joints 90 pipe diameters. Using the internal pipe diameter of 21.8 mm we get:

Table A3-3: Calculation of the effective length (l_e)

Effect	*Pipe (m)*	*Elbows (m)*	*Tee joints (m)*	l_e
1 and 2	7	4 x 40 x d = 3.5	0	10.5m
3	12	9 x 40 x d = 7.8	1 x 90 x d = 2	21.8m

l_f The length of standard pipework in the feed stream is approximately 7.6 m and contains 16 elbows. Therefore the equivalent length of this pipework is:

$$l_f = 7.6 + 16 \times 40 \times 0.0218 = 21.5m.$$

l_p The length of pipework in preheaters is approximately 12.7 m and contains 20 elbows. Its internal diameter is 9.5 mm, therefore the equivalent length is:

$$l_p = 12.7 + 20 \times 40 \times 0.0095 = 20.3m.$$

V_l The volume of liquid in the effect level is a piecewise function of L, similar to that for the cross-sectional area, A_l. From the plant dimensions the following function can be determined:

$$V_l = \begin{cases} (L+0.2)\times A_l \quad m^3 & 0 \le L \le 1.02 \\ 1.792\times10^{-3}(L+0.2)+3.76\times10^{-4}(L-1.02) \quad m^3 & 1.02 \le L \le 1.06 \\ 2.273\times10^{-3}+\frac{\pi}{6}((0.5(L-1.06)+0.0478)^3-0.0478^3)+\frac{\pi}{4}0.0218^2(L-1.06) \quad m^3 & 1.06 \le L \le 1.11 \\ 2.4365\times10^{-3}+(L-1.11)\times A_l \quad m^3 & 1.11 \le L \le 1.39 \\ 3.7066\times10^{-3}+\frac{\pi}{12k}(((L-1.39)k+0.0218)^3-0.0218^3)+\frac{\pi}{4}0.0728^2(L-1.39) \quad m^3 & 1.39 \le L \le 1.415 \\ 3.89\times10^{-3}+(L-1.415)\times A_l \quad m^3 & 1.415 \le L \le 1.6 \end{cases}$$

where: k is 3.04 for effects 1 and 2, and 5.95 for effect 3.

V_f With 7.6 m of standard pipe (21.8 mm dia.) and 12.7 m of preheater pipe (9.5 mm dia.) the volume of the feed pipework is calculated as:

$$V_f = \frac{\pi}{4}\left(7.6\times0.0218^2+12.7\times0.0095^2\right) = 3.737\times10^{-3}m^3.$$

V_p For effect 1 and 2 the product transport pipe is 7 m long and for effect is 12 m long. These give the following volumes:

Effect 1 and 2: $V_p = 2.61\times10^{-3}m^3$ and

Effect 3: $V_p = 4.48\times10^{-3}m^3$.

V_{ph} The preheater tube volume is given by:

$$V_{ph} = \pi d_p^2/4 \times 5.7 = 4.04 \times 10^{-4}\ m^3$$

Appendix 3(*b*)

Heat transfer coefficients

Assume the following steady-state conditions:

$T_s = 80°C$ $T_{ph1} = 37°C$ $T_{ph2} = 70°C$

$T_1 = 72°C$ $T_2 = 64°C$ $T_3 = 42°C$

$T_0 = 20°C$ $Q = 200$ l/hr $= 5.556 \times 10^{-2}$ kg/s

Calculating h_v

The table below summarises the terms needed to calculate h_v for each effect and preheater using equation (3.89). The properties were obtained from steam tables [3].

Table A3-4: Summary of parameters for calculating h_v

		Effect 1	Effect 2	Effect 3	Preheat 1	Preheat 2
T_v	°C	*80.0*	*72.0*	*64.0*	*45.0*	*80.0*
T_l	°C	*72.0*	*64.0*	*42.0*	*28.5**	*53.5**
T_{wv}	°C	*77.0†*	*69.0†*	*54.0†*	*37.0†*	*69.4†*
T_f	°C	*78.5*	*70.5*	*59.0*	*39.5*	*75.0*
ΔT_v	°C	*3.0*	*3.0*	*10.0*	*5.0*	*10.6*
L	m	*2.9*	*2.9*	*2.9*	*5.7***	*5.7***
k_f	W/(m.K)	*0.669*	*0.662*	*0.652*	*0.627*	*0.661*
μ_f	μPa.s	*362*	*402*	*475*	*693*	*408*
ρ_f	kg/m^3	*973*	*977*	*984*	*994*	*978*
λ	kJ/kg	*2313*	*2333*	*2360*	*2414*	*2335*
h_v	W/(m^{2}K)	***6340***	***6155***	***4345***	***3900***	***3775***

* The bulk liquid temperatures for the preheaters are taken as the mean of their input and output product temperatures, therefore:

For $Ph1$: $T_l = \frac{20 + 37}{2} = 28.5$ °C, and

for $Ph2$: $T_l = \frac{37 + 70}{2} = 53.5$ °C.

** The preheater tubes have three passes of 1.9 m in length, therefore the heated length, L, is 5.7 m.

† These wall temperatures (T_{wv}) were obtained thorough an iterative process. The heat transfer coefficients were first calculated using estimates of the wall temperatures. The temperatures were then back calculated using the HTCs and the new wall temperatures were then used to update the HTCs. This process continued until there was little change in estimates the wall temperatures. This iterative process is used for all the wall temperatures estimates for the remainder of the HTC calculations.

Calculating h_l

below summarises the terms needed to calculate h_l for each effect using equations (3.91) and (3.92). Assuming that we have flows through of 220 l/hr, 210 l/hr and 190 l/hr for the first, second and third effects respectively, we get:

Table A3-5: Summary of parameters for calculating h_l for each effect

		Effect 1	Effect 2	Effect 3
T_{wl}	°C	*75.3*	*67.4*	*50.0*
T_f	°C	*73.7*	*65.7*	*46.0*
m	kg/s ($\times 10^{-2}$)	*5.56*	*5.14*	*4.44*
d	mm	*34.8*	*34.8*	*21.8*
Γ	kg/(m.s)	*0.558*	*0.470*	*0.649*
k_f	W/(m.K)	*0.665*	*0.659*	*0.639*
μ_f	μPa.s	*395*	*403*	*586*
ρ_f	kg/m^3	*975*	*980*	*990*
c_p	J/(kg.K)	*4193*	*4188*	*4179*
N_{Re}	–	*5663**	*4963**	*5260**
N_{Pr}	–	*2.49*	*2.73*	*3.83*
h_l	W/(m^{2}K)	***6285***	***5825***	***5280***

* Since the Reynolds numbers (N_{Re}) are greater than 2100, each falling-film is in the turbulent flow regime.

For the preheaters:

d = 9.5 mm, l = 1.9 m and at 220 l/hr the fluid velocity, v = 0.784 m/s.

The table below summarises the terms needed to calculate h_l using equation (3.91) together with either equation (3.96) or (3.97).

Table A3-6: Summary of parameters for calculating h_l for the preheaters

		Preheater 1	Preheater 2
T_{wl}	°C	*33.1*	*61.6*
μ	μPa.s	*825*	*518*
μ_w	μPa.s	*748*	*456*
ρ	kg/m^3	*996*	*987*
c_p	J/(kgK)	*4179*	*4182*
k	W/(mK)	*0.616*	*0.647*
N_{Re}	–	*8998*†	*14173*‡
N_{Pr}	–	*5.50*	*3.28*
N_{Nu}	–	*65.7*†	*72.9*‡
h_l	W/(m^{2}K)	***4260***	***4960***

† Since $2100 < N_{Re} < 10{,}000$, the transition flow equation (equation (3.97)) is used to calculate the Nusselt number, N_{Nu}, for preheater 1.

‡ Since $N_{Re} > 10{,}000$, the turbulent flow equation (equation (3.96)) is used to calculate the Nusselt number, N_{Nu}, for preheater 2.

Calculating h_w

All the tubes have a wall thickness of 1.6 mm. The evaporator is constructed of Type 304 stainless steel which has a thermal conductivity of 16.27 W/(m.K) [4].

Using equation (3.88) we get:

$$h_w = \frac{16.27}{1.6 \times 10^{-3}} = 10{,}170\ W/(m^2K)$$

Overall heat transfer coefficients

Using equation (3.87) we get the following overall heat transfer coefficients:

Table A3-7: Calculation of the overall heat transfer coefficients

W/(m²K)	Effect 1	Effect 2	Effect 3	Preheat 1	Preheat 2
h_v	6340	6155	4345	3900	3775
h_w	10,170	10,170	10,170	10,170	10,170
h_l	6285	5825	5280	4260	4960
U	***2410***	***2310***	***1930***	***1700***	***1770***

Heat loss coefficients

The assumption is that the ambient air temperature, T_a, is 20°C and the temperature difference (ΔT_o) is approximated using the bulk temperatures. Using equation (3.99) and remembering that $U_{loss} = h_o$, we get the following results:

Table A3-8: Summary of heat loss coefficients

	Effect 1	Effect 2	Effect 3	Preheat 1	Preheat 2
ΔT_o (°C)	50	40	25	25	50
U_{loss} (W/(m².K))	***4.8***	***4.5***	***3.8***	***3.8***	***4.8***

Similarly, for the U_{lossl} terms we get:

	Effect 1	Effect 2	Effect 3
U_{lossl} (W/(m².K))	4.8	4.5	3.8

For the venturi, ΔT_o = 50°C, assuming that the temperature of the entrained vapour and condensate mixture is around 70°C. This also gives U_{lossv} = 4.8 W/(m²K).

Appendix 3(*c*)

Residence times and delays

Estimating τ_p, the time delay between effects.

The experiment involved inserting a small amount of dye into the sight glass of the upstream effect when the level is negligible and measuring the time for the dye to appear at the base of the next effect. This was achieved by breaking the connection to the effect pump so the dye could be easily observed flowing out. The route the dye takes includes the following sections:

a – the base of the upstream effect to the effect pump,

b – the pipework from the pump to the top of the next effect,

c – the distribution nozzle and the evaporation tube, and

d – the base of the downstream effect.

Section **b** is the only part that we are interested in for estimating τ_p. It is known from previous tests [1] that the time delay through section **c** is approximately 4 seconds. If we assume that section **a** and **d** take 3 seconds each to travel through then we can estimate the time delay through **c** by subtracting 10 seconds from the measured time. This was done for four different flowrates and compared with estimates calculated using equation (3.18). The aim is to verify whether this equation is appropriate to use in estimating the time delay.

The test was carried out using the first two effects. The volume of the inter-effect pipe is 2.61×10^{-3} m^3. A summary of the test results is given below:

Table A3-9: Comparison of estimated and measured delays for the inter-effect pipes

Flowrate (l/hr)	Ave. time (s) Section **a** to **d**	τ_p (s) *measured*	τ_p (s) *calculated*
178	*60.4*	*50.4*	*52.8*
209	*56.8*	*46.8*	*45.0*
242	*53.0*	*43.0*	*38.8*
282	*46.6*	*36.6*	*33.3*

The results show reasonable agreement between the measured and calculated time delays. Some error could be incurred due to the difficulty in determining the arrival of the dye at the exit point. This shows that the use of equation (3.18) does give realistic time delay estimates.

Estimating τ_f, the feed system time delay.

A similar experiment was carried out for the feed system to estimate τ_f. The experiment involved inserting dye into the feed tank while the tank level was negligible and measuring the time for the dye to appear at the exit of the second preheater.

The total volume of the pipework through the feed system up to the exit of preheater 2 is approximately 3.486×10^{-3} m^3. Note that the pipe to the top of effect 1 was not used so this volume is less than V_f. The following results were obtained:

Table A3-10: Comparison of estimated and measured delays for feed system

Ave. flowrate (l/hr)	Ave. τ_f (s) *measured*	τ_f (s) *calculated*
384	*34.3*	*32.7*
442	*28.8*	*28.3*

Again there is reasonably good agreement between the two estimates of the delay, which gives further indication of the suitability of using the volume and flowrate equation to estimate the delay as in equation (3.38).

Appendix 3(*d*)

Time constants

Three time constants are required to be known for the evaporator model. These are for the two preheater output temperatures (T_{ph1} and T_{ph2}) and the steam pressure (T_s). To estimate these time constants step changes were applied to the setpoint of the steam pressure controller while steam pressure PI controller was active. This was done was because the characteristic of the steam valve is very nonlinear and a step change in the steam valve when applied manually either did not produce a reasonable perturbation in the steam pressure or conversely caused the steam pressure to rise too high without any obvious steady-state. By using the setpoint of the controller to apply the step change we were able to determine an approximation to both the preheater time constants and the steam pressure.

Three step changes were applied to pressure setpoint (P_{sp}):

Step 1: 50 kPa to 60 kPa

Step 2: 60 kPa to 50 kPa

Step 3: 50 kPa to 40 kPa

Figure A3-1 shows the step changes and the responses of the steam preheater 2 temperatures. The steam pressure has been converted into its saturation temperature for the remainder of the analysis. The steam temperature changes with an approximate step change in responses to the setpoint changes which enables estimates for time constants to be calculated. To determine the time constant for the first preheater temperature a step change must be applied to the third effect temperature. Achieving this is difficult due to the absence of direct control on the third effect temperature. For the purposes of the model the first preheater time constant was assumed to be equal to the time constant of the second preheater since physically the preheaters are essentially identical and are utilised in a similar manner.

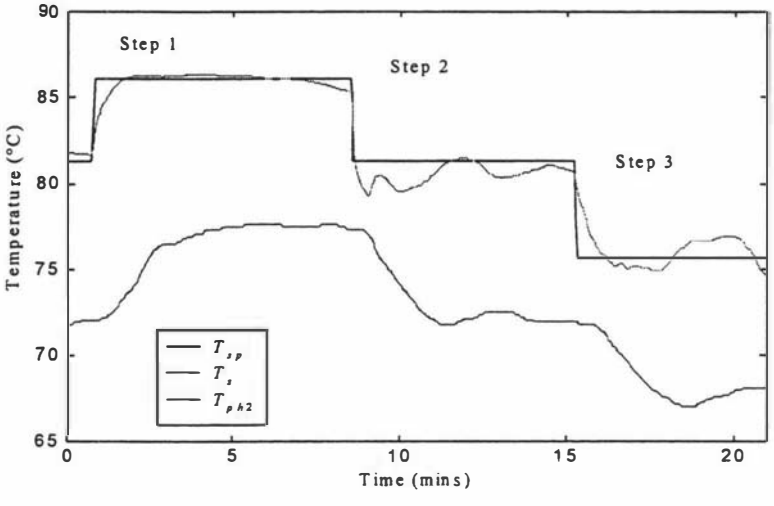


Figure A3-1: Step changes to steam temperature setpoint and the responses.

The time constant, τ_c, of a system is the time taken for the system disturbed with a step input to achieve 63.2% of its response. That is

$$y(\tau_c) = 0.632(y_{ss} - y(t_0)) + y(t_0) \qquad \text{(A3-1)}$$

where y_{ss} is the steady-state or equilibrium value of the system response, $y(t)$ is the system response at time t and the step input is applied at $t = t_0$.

Analysis of the temperature responses for the evaporator step test gives the following results table:

Table A3-11: Step test temperature results for calculating the time constants

	T_{ph2}		T_s	
Step 1	**Value (°C)**	**Time (s)**	**Value (°C)**	**Time (s)**
$y(t_0)$	*72.1*	*48*	*80.5*	*48*
y_{ss}	*77.6*	–	*85.0*	–
$y(\tau_c)$	*75.6*	*150*	*83.3*	*66*
τ_c	$\tau_c = 150 - 48 = 102$ s		$\tau_c = 66 - 48 = 18$ s	
Step 2	**Value (°C)**	**Time (s)**	**Value (°C)**	**Time (s)**
$y(t_0)$	*77.3*	*516*	*	*
y_{ss}	*71.9*	–	*	*
$y(\tau_c)$	*73.9*	*609*	*	*
τ_c	$\tau_c = 609 - 516 = 93$ s		*	
Step 3	**Value (°C)**	**Time (s)**	**Value (°C)**	**Time (s)**
$y(t_0)$	*71.95*	*918*	*79.75*	*918*
y_{ss}	*67.1*	–	*74.3*	–
$y(\tau_c)$	*68.9*	*1029*	*76.3*	*942*
τ_c	$\tau_c = 1029 - 918 = 111$ s		$\tau_c = 918 - 942 = 24$ s	

* Due to the poor response of the steam temperature, in that it failed to settle sufficiently, the *Step 2* calculations were not carried out for the steam temperature time constant.

The averages of these few calculations are taken as estimates of the time constants to give:

System	Time constant, τ_c (s)
Preheater 2 (and Preheater 1)	*102*
Steam temperature	*21*

Appendix 3(*e*)

Pressure-flow characteristic constants

i) *Pump characteristic measurements*

Table A3-12: Experimental results for small impeller

- Effect 1 and 2 and feed pumps

N (rpm)	ΔP (kPa)	N^2 (rpm^2)
1560	*55*	*2433600*
1705	*66*	*2907025*
1840	*78*	*3385600*
1970	*88*	*3880900*
2090	*98*	*4368100*

ΔP was regressed using N^2 as the sole predictor and the intercept was fixed at zero.
Slope = *0.022665* with standard error = *0.0001* and R^2 = *99.8%*
Hence for equation (3.12): $a_P = 0.02267 \pm 0.0002$ with 95% confidence.

Table A3-13: Experimental results for large impeller - *Effect 3*

N (rpm)	ΔP (kPa)	N^2 (rpm^2)
1405	*52*	*1974025*
1550	*65*	*2402500*
1690	*75*	*2856100*
1820	*89*	*3312400*

ΔP was regressed using N^2 as the sole predictor and the intercept was fixed at zero.
Slope = *0.026662* with standard error = *0.00019* and R^2 = *99.6%*
Hence for equation (3.12): $a_P = 0.02666 \pm 0.0004$ with 95% confidence.

ii) *Nozzle characteristic measurements*

Table A3-14: Experimental results for effect 1 and 2 (3 × 1 mm diameter holes)

Q ($\times 10^{-5}$ m^3/s)	ΔP (kPa)	$\sqrt{\Delta P}$ (Pa^{½})
0.351	0.5	22.4
4.81	3	54.8
5.65	7	83.7
6.67	10	100
7.60	13	114
8.66	17	130
9.35	20	141
10.0	23	152

The flowrate, Q, was regressed with $\sqrt{\Delta P}$ as predictor and the intercept was fixed to zero.

Regression results:

Slope = 6.65×10^{-7} with standard error of 1.8×10^{-9}.

Hence for equation (3.14): $C_N = (6.65 \pm 0.04)\times10^{-7}$ with 95% confidence.

Table A3-15: Experimental results for effect 3 (3 × 1.5 mm holes)

Q ($\times 10^{-5}$ m^3/s)	ΔP (kPa)	$\sqrt{\Delta P}$ (Pa^{½})
3.40	10	100
4.17	17	130
4.88	24	155
5.43	31	176
5.95	40	200
6.41	49	221
6.64	57*	239
7.27	65	255
7.75	72	268

* This reading was in poor agreement with the square root law.

The flowrate, Q, was regressed with $\sqrt{\Delta P}$ as predictor and the intercept was fixed to zero.

Regression results:

Slope = 2.97×10^{-7} with standard error of 5.0×10^{-9}.

Hence for equation (3.14): $C_N = (2.97 \pm 0.10)\times10^{-7}$ with 95% confidence.

Note: All results for the pump and nozzles are reproduced from Illingworth [1].

iii) *Condensate flow constants*

The diameters of the condensate orifices vary for each effect and are as follows:

Table A3-16: Dimensions of condensate orifices

	Diameter (mm)	Area, A_c (m^2)
Preheater 2	*1.2*	1.13×10^{-6}
Effect 1	*1.5*	1.77×10^{-6}
Effect 2	*2.0*	3.14×10^{-6}
Effect 3	*3.0*	7.07×10^{-6}

Under standard steady-state operation conditions, assuming that the venturi suction pressure (P_{vent}) is 4.1 kPa, as calculated in Appendix 2, and using equations (3.102) and (3.53) the following estimates for K_c and C_{vc} are obtained:

Table A3-17: Estimates of condensate flow constants

	K_c	C_{vc} (kg/$Pa^{½}$s)
Preheater 2	*0.385*	1.92×10^{-5}
Effect 1	*0.278*	2.16×10^{-5}
Effect 2	*0.196*	2.72×10^{-5}
Effect 3	*0.105*	3.30×10^{-5}

Notice that K_c decreases and the flow constant increases with the size of the orifice.

Appendix 3(*f*)

Thermophysical properties

The following polynomials have been used to approximate thermophysical water properties as a function of the liquid temperature or pressure. The regressions were performed using data from Cooper & Le Fevre [3].

Saturated vapour pressure (Pa)

$$P = 960.68 - 14.735T + 4.5712T^2 - 4.5077 \times 10^{-2} T^3 + 1.0113 \times 10^{-3} T^4$$

for $0 \le T \le 100$ °C.

Saturated temperature (°C)

$$T = 15.221 + 4.272P - 0.15783P^2 + 3.7023 \times 10^{-3} P^3 - 4.8971 \times 10^{-5} P^4$$
$$\ldots 3.3532 \times 10^{-7} P^5 - 9.2297 \times 10^{-10} P^6$$

for $5 \le P \le 100$ kPa.

Liquid density (kg/m^3)

$$\rho = 1001.4 - 0.10424T - 3.2924 \times 10^{-3} T^2$$

for $0 \le T \le 100$ °C.

Latent heat of vaporisation (J/kg)

$$r = 2497.9 \times 10^3 - 2.2063 \times 10^3 T - 2.0131T^2$$

for $34 \le T \le 100$ °C.

Appendix 3 References

[1] **Illingworth, M.I.**, *Dynamic modelling of a three-effect falling-film evaporator*. Final Year Undergraduate Research Project, Dept. Production Technology, Massey University, New Zealand, 1993.

[2] **Coulson, J.M. & Richardson, J.F.**, *Chemical Engineering*, Vol. 1, 2nd Edition, Pergamon Press, Oxford, UK, 1970.

[3] **Cooper, J.R. & Le Fevre, E.J.**, *Thermophysical Properties of Water Substance*, Edward Arnold Ltd., London, 1986.

[4] **Perry, R.H., Green, D.W. & Maloney, J.O** (Eds.). *Perry's Chemical Engineers' Handbook*, 6th Edition, McGraw-Hill Book Co., New York, USA, 1984, pp. 3-262.

Appendix 4

Analytical Model MATLAB Files

Contents

Appendix 4(*a*)

S-function files

The files names are prefixed with an 'f' to identify them as files relating to the feedforward model as opposed to the recirculating model.

Effect 1 S-function: *feff1.m*

```
function[sys,x0] = feff1(t,x,u,flag);
%
%           FEFF1
%
%           An M-file S-Function that defines a system of ord diff eqns (ODEs) that describe the first effect of the Prod Tech
%           Dept's Falling Film Evaporator in feedforward configuration.
%           See the M-file SFUNC for a full description of how an S-Function is structured.
%
%           FLAG  SYS    DESCRIPTION
%           1     DX     state derivatives, dX/dT.
%           3     Y      system outputs.
%
%           This s-fuction has 5 states, 10 inputs and 9 outputs.
%
%           State vector X = [h1 L1 T1 Tl1 Mv1]
%           Input vector U = [Ta Ts Tph2 T2 N1 Qdd1 q1d Tph2d T1d Q0]
%           Output vector Y = [h1 L1 T1 Qd1 Qe1 Qp1 q1 Mv1 Tl1]

%           Written by N.T. Russell
%           Dept. Production Technology
%           Massey University, 1995

global INIT1 INIT2

% return initial values and system description
if (nargin == 0)|(flag == 0),
        sys = [5,0,9,10,0,0]; x0 = INIT1(1:5); return, end
else

% Ensure that input values are non-zero initially (takes 1 step for inputs to activate)
        % This is essential when trying to start the model in a steady-state
        if t == 0,
                u = INIT1(6:15);
        end;

% Assign current state and input values for ease of reading
        h1 = x(1); L1 = x(2); T1 = x(3); Tl1 = x(4); Mv1 = x(5);
        Ta = u(1); Ts = u(2); Tph2 = u(3); T2 = u(4); N1= u(5); Qdd1 = u(6); q1d = u(7); Tph2d = u(8); T1d = u(9); Q0 = u(10);

        % Clip States
        if h1 < 0, h1 = 0;
        elseif h1 > 0.071, h1 = 0.071;
        end;
        if L1 < 0, L1 = 0; end;

        % Define Constants
        Cn2 = 5.45e-7; ap = 2.14e-2;
        Ah = 2.36e-4; te=4; g = 9.81; hN = 4.82; f = 0.009;
        d = 0.0218; Le=10.5; As = 1.059; Asl = 0.363;  xi = 1;
        U1 = INIT1(16); U2 = INIT2(16); Uloss = 4.8; Ulossl = 4.8;
        Ae1 = 0.3313; Ae2 = 0.3313; Cp = 4191;

% Calculate area of liquid on distribution plate and the volume and surface area of liquid level at the base of the effect.
        [Ad,Al,Vl] = CalcA(h1,L1,1);

% Calculate thermophysical properties:

        rhoPh2d = convert(Tph2d,2); rho1 = convert(T1,2); rhol1 = convert(Tl1,2);
```

```
        P1 = convert(T1,1); P2 = convert(T2,1);
        r1 = convert(T1,3); r1d = convert(T1d,3);

% Calculate current flow conditions

        if (h1 >= 0.071) & (Q0 > (Ah*sqrt(2*g*0.071/xi))),
                Qd1=Q0;
        else
                Qd = Ah*sqrt(2*g*h1/1);
        end;
        Qe1 = (rhoPh2d*Qdd1-1/te*Mv1)/rho1;
        if Qe1 <= 0,
                Qe1 = 0;
        end;
        if (L1 <= 0),
                Qp1 = 0;
        else
                dPf= 32*rho1*f*Le/(pi^2*d^5);
                Qp1 = sqrt((rho1*g*(L1-hN)+P1-P2+ap*N1^2)/(1/(Cn2^2)+dPf));
                if (Qp1 < 0) | (imag(Qp1)~=0),
                    Qp1 = 0;
                end;
        end;

        % Estimate mass of liquid in the evaporation tube
        M = (Qd1+Qe1)/2*te*rho1;

        % Heat flow calculations - evaporation tube
        qi = Qdd1*rhoPh2d*Cp*(Tph2-T1);
        q1 = U1*Ae1*(Ts-T1);
        q2 = U2*Ae2*(T1-T2);
        qloss1 = Uloss*As*(T1-Ta);

        % Heat flow calculations - base of effect
        qt = Qe1*rho1*Cp*(T1-Tl1);
        qloss1 = Uloss1*As1*(Tl1-Ta);
        Ml = Vl*rhol1;

% Calculate State Derivatives
        if (flag == 1),
                h1dot = (Q0-Qd1)/Ad;
                if ((h1 <= 0) & (h1dot < 0)),
                    h1dot = 0;
                end;
                L1dot = (Qe1-Qp1)/Al;
                if (L1 == 0 & L1dot < 0),
                   L1dot = 0;
                end;
                if M == 0
                  T1dot = 0;
                else
                  T1dot = (q1+qi-q2-qloss1)/M/Cp;
                end
                Tl1dot = (qt-qloss1)/Ml/Cp;
                Mv1dot = q1/r1-q1d/r1d;
                sys = [h1dot L1dot T1dot Tl1dot Mv1dot];

        % prepare outputs
        elseif(flag == 3),
                sys = [h1 L1 T1 Qd1 Qe1 Qp1 q1 Mv1 Tl1];

        else
                sys = [];
        end;
end
```

Effect 2 S-function: *feff2.m*

```
function[sys,x0] = feff2(t,x,u,flag);
%
%           FEFF2
%
%           An M-file S-Function that defines a system of ord diff eqns (ODEs) that describe the second effect of the Prod Tech
%           Dept's Falling Film Evaporator in feedforward configuration.
%           See the M-file SFUNC for a full description of how an S-Function is structured.
%
%           FLAG  SYS     DESCRIPTION
%           1     DX      state derivatives, dX/dT.
%           3     Y       system outputs .
%
%           This s-function has 5 states, 10 inputs and 9 outputs.
%
%           State vector X = [h2 L2 T2 Tl2 Mv2]
%           Input vector U = [Ta T1 T3 N2 Q2dd q2d T1d T2d Qp1 Tl1]
%           Output vector Y = [h2 L2 T2 Qd2 Qe2 Qp2 q2 Mv2 Tl2]

%           Written by N.T. Russell
%           Dept. Production Technology
%           Massey University, 1995

global INIT2 INIT3

% return initial values and system description
if (nargin == 0)|(flag == 0),
        sys = [5,0,9,10,0,0]; x0 = INIT2(1:5); return, end
else

% Ensure that input values are non-zero initially (takes 1 step for inputs to activate)
        % This is essential when trying to start the model in a steady-state
        if t == 0,
                u = INIT2(6:15);
        end;

% Assign current state and input values for ease of reading
        h2 = x(1); L2 = x(2); T2 = x(3); Tl2 = x(4); Mv2 = x(5);
        Ta = u(1); T1 = u(2); T3 = u(3); N2 = u(4); Qdd2 = u(5); q2d = u(6); T1d = u(7); T2d = u(8); Qp1 = u(9); Tl1 = u(10);

        % Clip States
        if h2 < 0, h2 = 0;
        elseif h2 > 0.071, h2 = 0.071;
        end;
        if L2 < 0, L2 = 0; end;

% Define Constants
        Cn3 = 2.89e-7; ap = 1.92e-2;
        Ah = 2.36e-4; te = 4; g = 9.81; hN = 4.82; f = 0.009;
        d = 0.0218; Le = 10.5; As = 1.059; Asl=0.363; Asp1 = 0.55;
        U2 = INIT2(16); U3 = INIT3(20); Uloss = 4.5; Ulossl = 4.5; Ulosspl = 4.8;
        Ae2 = 0.3313; Ae3 = 0.2137; Cp1 = 4191; Cp2 = 4185; xi = 1;

% Calculate area of liquid on distribution plate and the volume surface and area of liquid level at the base of the effect.
        [Ad,Al,Vl] = CalcA(h2,L2,2);

% Calculate thermophysical properties:
        rho1d = convert(T1d,2); rho2 = convert(T2,2); rho11 = convert(Tl1,2); rho12 = convert(Tl2,2);
        P1 = convert(T1d,1); P2 = convert(T2,1); P3 = convert(T3,1);
        r2 = convert(T2,3); r2d = convert(T2d,3);

% Calculate current flow conditions
        if (h2 >= 0.071) & (Qp1 > (Ah*sqrt(2*g*0.071/xi))),
                Qd2 = Qp1;
        else
                Qd2 = Ah*sqrt(2*g*h2/1);
        end;
        Qe2 = (rho1d*Qdd2-1/te*Mv2)/rho2;
        if Qe2 <= 0,
                Qe2 = 0;
        end;
```

```
        if (L2 <= 0),
                Qp2 = 0;
        else
                dPf= 32*rhol2*f*Le/(pi^2*d^5);
                Qp2 = sqrt((rhol2*g*(L2-hN)+P2-P3+ap*N2^2)/(1/(Cn3^2)+dPf));
                if (Qp2 < 0) | (imag(Qp2)~=0),
                    Qp2 = 0;
                end;
        end;

% Estimate mass of liquid in the evaporation tube
        M=(Qd2+Qe2)/2*te*rho2;

% Apply transport tube heat loss to input flow
        if Qp1 == 0
                Tl1 = 0;
        else
                Tl1 = Tl1 - Ulossp1*Asp1*(Tl1-Ta)/rhol1/Qp1/Cp1;
        end

% Heat flow calculations - evaporation tube
        qi = Qdd2*rho1d*Cp2*(Tl1-T2);
        q2 = U2*Ae2*(T1-T2);
        q3 = U3*Ae3*(T2-T3);
        qloss2 = Uloss*As*(T2-Ta);

% Heat flow calculations - level

        qt = Qe2*rho2*Cp2*(T2-Tl2);
        qlossl = Ulossl*Asl*(Tl2-Ta);
        Ml = Vl*rhol2;

% Calculate State Derivatives
        if (flag == 1),

                h2dot = (Qp1-Qd2)/Ad;
                if ((h2 <= 0) & (h2dot < 0)),
                   h2dot = 0;
                end;
                L2dot = (Qe2-Qp2)/Al;
                if (L2 == 0 & L2dot<0),
                   L2dot = 0;
                end;
                if M == 0
                   T2dot = 0;
                else
                   T2dot = (q2+qi-q3-qloss2)/M/Cp2;
                end
                Tl2dot = (qt-qlossl)/Ml/Cp2;
                Mv2dot = q2/r2-q2d/r2d;
                sys = [h2dot L2dot T2dot Tl2dot Mv2dot];

% prepare outputs
        elseif (flag == 3),
                sys = [h2 L2 T2 Qd2 Qe2 Qp2 q2 Mv2 Tl2];

        else
                sys=[];
        end
end
```

Effect 3 S-function: *feff3.m*

```
function[sys,x0] = feff3(t,x,u,flag);
%
%           FEFF3
%
%           An M-file S-Function that defines a system of ord diff eqns (ODEs) that describe the third effect of the Prod Tech
%           Dept's Falling Film Evaporator in feedforward configuration.
%           See the M-file SFUNC for a full description of how an S-Function is structured.
%
%           FLAG  SYS    DESCRIPTION
%           1     DX     state derivatives, dX/dT.
%           3     Y      system outputs .
%
%           This s-function has 5 states, 14 inputs and 9 outputs.
%
%           State vector X = [h3 L3 T3 Tl3 Mv3]
%           Input vector U = [Ta T0 Tph1 T2 N3 Qdd3 q3d Pa T2d T3d Qp2 mvPh1 Tl2 Cv]
%           Output vector Y = [h3 L3 T3 Mv3 Qd3 Qe3 Qp3 q3 Tl3]

%           Written by N.T. Russell
%           Dept. Production Technology
%           Massey Universtiy, 1995

global INITPHI INIT3

% return initial values and system description
if (nargin == 0)|(flag == 0),
        sys = [5,0,9,14,0,0]; x0 = INIT3(1:5); return, end
else

% Ensure that input values are non-zero initially (takes 1 step for inputs to activate)
        % This is essential when trying to start the model in a steady-state
        if t == 0,
                u = INIT3(6:19);
        end;

% Assign current state and input values for ease of reading
        h3 = x(1); L3 = x(2); T3 = x(3); Tl3 = x(4); Mv3 = x(5);
        Ta = u(1); T0 = u(2); Tph1 = u(3); T2 = u(4); N3 = u(5); Qdd3 = u(6); q3d = u(7); Pa = u(8); T2d = u(9); T3d = u(10);
        Qp2 = u(11); mvPh1 = u(12); Tl2 = u(13); Cv = u(14);

        % Clip States
        if h3 < 0,  h3 = 0;
         elseif h3 > 0.071, h3 = 0.071;
        end;
        if L3 < 0, L3 = 0; end;

% Define Constants
        ap = 3.00e-2;
        Ah = 0.000236; te = 4; g = 9.81; hV = 1.0; f = 0.009; xi=1;
        d = 0.0218; Le = 21.8; As = 1.099; Asl = 0.379; Asp2 = 0.55; Ae3 = 0.2137;
        U3 = INIT3(20); UPh1 = INITPHI(12); Uloss = 3.8; Ulossl = 3.8; Ulossp2 = 4.5;
        APh1 = pi*12.7e-3*5.7; Cp2 = 4185; Cp3 = 4179;

% Calculate area of liquid on distribution plate and the volume and surface area of liquid level at the base of the effect.
        [Ad,Al,Vl]=CalcA(h3,L3,3);

% Calculate thermophysical properties:

        rho2d = convert(T2d,2);rho3 = convert(T3,2); rhol2 = convert(Tl2,2); rhol3 = convert(Tl3,2);
        P3 = convert(T3,1); r3 = convert(T3,3); r3d = convert(T3d,3);

% Calculate current flow conditions - including allowances for boundary conditions
        %(eg no negative flows allowed)

        if (h3 >= 0.071) & (Qp2 > (Ah*sqrt(2*g*0.071/xi))),
                Qd3 = Qp2;
        else
                Qd3 = Ah*sqrt(2*g*h3/xi);
        end;
        Qe3 = (rho2d*Qdd3-1/te*Mv3)/rho3;
```

```
        if Qe3 <= 0,
                Qe3 = 0;
        end;
        if (L3 <= 0),
           Qp3 = 0;
        else

           dPf = (32*rhol3*f*Le)/(pi^2*d^5);
           Qp3 = sqrt((rhol3*g*(L3-hV)+P3-Pa+ap*N3^2)/(dPf+1/Cv^2));

        if (Qp3 < 0) | (imag(Qp3)~=0),
                Qp3 = 0;
           end;
        end;

% Apply transport tube heat loss to input flow
if Qp2 == 0
        Tl2 = 0;
else
        Tl2 = Tl2 - Ulossp2*Asp2*(Tl2-Ta)/rhol2/Qp2/Cp2;
end

% Estimate mass of liquid in the evaporation tube
        M = (Qd3+Qe3)/2*te*rho3;

% Heat flow calculations - evaporation tube
        qi = Qdd3*rho2d*Cp3*(Tl2-T3);
        q3 = U3*Ae3*(T2-T3);
        qloss3 = Uloss*As*(T3-Ta);

        if (Tph1 == T0 | Tph1 >= T3 | T0 >= T3 ),
          Tph1m = T3-(Tph1+T0)/2;
        else
          Tph1m = (Tph1-T0)/log((T3-T0)/(T3-Tph1));
        end;

        qPh1 = UPh1*APh1*Tph1m;
        qv = r3*mvPh1; % heat lost into venturi condenser

% Heat flow calculations - condensate level
        qt = Qe3*rho3*Cp3*(T3-Tl3);
        qlossl = Ulossl*Asl*(Tl3-Ta);
        Ml = Vl*rhol3;

% Calculate State Derivatives
        if (flag == 1),
                h3dot = (Qp2-Qd3)/Ad;
                if ((h3 <= 0) & (h3dot < 0)),
                     h3dot = 0;
                end;
                L3dot = (Qe3-Qp3)/Al;
                if (L3 == 0 & L3dot <0),
                   L3dot = 0;
                end;
                if M == 0
                  T3dot = 0;
                else
                  T3dot = (q3+qi-qPh1-qloss3-qv)/M/Cp3;
                end
                Tl3dot = (qt-qlossl)/Ml/Cp3;
                Mv3dot = q3/r3-q3d/r3d;
                sys = [h3dot L3dot T3dot Tl3dot Mv3dot];

        % prepare outputs
        elseif (flag == 3),
                sys = [h3 L3 T3 Qd3 Qe3 Qp3 q3 Mv3 Tl3];

        else
                sys=[];
        end;
end
```

Preheater 1 S-function: *fpreht1.m*

```
function[sys,x0]=fpreht1(t,x,u,flag);
%
%   FPREHT1
%
%         An M-file S-Function that defines a system of ord diff eqns (ODEs) that describe the first preheater of the Prod Tech
%         Dept's Falling Film Evaporator in feedforward configuration.
%         See the M-file SFUNC for a full description of how an S-Function is structured.
%
%         FLAG  SYS     DESCRIPTION
%           2     DS     discrete states, X(n+1).
%           3     Y      system outputs.
%           4     TNEXT  next time interval for update  (only discrete systems)
%
%         This s-function has 3 states (1 discrete), 8 inputs and 5 outputs.
%
%         State vector X=[LT McPh1 Tph1]
%         Input vector U=[Ta Qin T0 T3 Q0 Qv3d qPh1d Tph1d]
%         Output vector Y=[LT Tph1 qPh1 qlossPh1 mcPh1 mvPh1]

%         Written by N.T. Russell
%         Dept. Production Technology
%         Massey University, 1995

global INITPH1 INIT3

% return initial values and system description,
if (nargin == 0)|(flag == 0),
        sys = [2,1,5,8,0,0]; x0 = INITPH1(1:3); return, end
else

% Define initial inputs ie. when t=0.
        if t == 0,
      u = INITPH1(4:11);
          T = 0;
        end;

% Assign states and inputs for ease of reading.
        LT = x(1); McPh1 = x(2); Tph1 = x(3);
        Ta = u(1); Qin = u(2); T0 = u(3); T3 = u(4); Q0 = u(5); Mv3d = u(6); qPh1d = u(7); Tph1d = u(8);

% Clip states

        if LT > 0.75, LT = 0.75;
        elseif LT < 0, LT = 0; end;

% Define constants

        UPh1 = INITPH1(12); Uloss1 = 3.8; Cp = 4179; Asph = 0.65; L = 5.7;
        te = 4; Atank = pi*0.45^2/4; tph = calctph(Q0); dT = 1; a = 9.8e-3;
        T0m = (T0+Tph1)/2; d = 11.1e-3; %mean diameter d = (di+do)/2
        Aph = pi*d*L;

        if LT < 0.25,      % account for pipe below feed tank
          Atank = pi*0.0478^2/4;
        end;

% Calculate thermophysical properties:
        rho0 = convert(T0m,2); rhoPh1 = convert(Tph1,2); rhoPh1d = convert(Tph1d,2);
        rPh1 = convert(Tph1,3); rPh1d = convert(Tph1d,3);

% Calculate Preheater1 Heat Flows
        qlossPh1 = Uloss1*Asph*(T3-Ta);
        if Q0 == 0
          Tph1s = 0;
        else
          Tph1s = T3-(qlossPh1/rho0/Q0/Cp + T3 - T0)*exp(-UPh1*Aph/rho0/Q0/Cp);
        end;

        if (Tph1 == T0 | Tph1 >= T3 | T0 >= T3 ),
          Tph1m = T3-(Tph1+T0)/2;
```

```
        else
          Tph1m = (Tph1-T0)/log((T3-T0)/(T3-Tph1));
        end;

        qPh1 = UPh1*Aph*Tph1m;

        mcPh1 = McPh1/tph;              % Calculate vapour and condensate flows
        mvPh1 = (Mv3d/te)-mcPh1;        % from Ph1 (in kg/s) for venturi heat balance.
        if (mvPh1 <=0),
                mvPh1 = 0;
                mcPh1 = Mv3d/te;
        end;
        if mcPh1<=0,
                mcPh1 = 0;
                mvPh1 = Mv3d/te;
        end;

% Calculate state derivatives
        if (flag==1),

                LTdot = (Qin-Q0)/Atank;
                if (LT >= 0 & LTdot >= 0),
                    LTdot = 0;
                end;
                McPh1dot = (qPh1-qlossPh1)/rPh1-qPh1d/rPh1d;

                sys = [LTdot McPh1dot];

% Calculate DISCRETE state derivative
        elseif (flag == 2),
                Tph1 = (1-a)*Tph1 + a*Tph1s;

                sys = Tph1;

% prepare outputs
        elseif (flag==3),

                qPh1 = qPh1-qlossPh1;
                sys = [LT Tph1 qPh1 mcPh1 mvPh1];

% Next Time interval for discrete update
        elseif (flag == 4)
                sys = T+dT;

        else
                sys = [];
        end;
end;
```

Preheater 2 S-function: *fpreht2.m*

```
function[sys,x0] = fpreht2(t,x,u,flag);
%
%       FPREHT2
%
%       An M-file S-Function that defines a system of ord diff eqns (ODEs) that describe the second preheater of the Prod
%       Tech Dept's Falling Film Evaporator in feedforward configuration.
%       See the M-file SFUNC for a full description of how an S-Function is structured.
%
%       FLAG  SYS    DESCRIPTION
%         2     DS    discrete states, X(n+1).
%         3     Y     system outputs.
%         4     TNEXT next time interval for update  (only discrete systems)
%
%       This s-function has 1 discrete state, 4 inputs and 1 outputs.
%       State vector X=[TPh2]
%       Input vector U=[Ta Ts TPh1 Q0]
%       Output vector Y=[TPh2]

%       Written by N.T. Russell
%       Dept. Production Technology,
```

```
%           Massey University, 1995.

global INITPH2

% return initial values and system description,
if (nargin == 0)|(flag == 0),
        sys = [0,1,1,4,0,0]; x0 = INITPH2(1); return, end
else

% Define initial inputs ie. when t=0.
        if t == 0,
      u = INITPH2(2:5);
         T = 0;
        end;

% Assign states and inputs for ease of reading.
        Tph2 = x(1);
        Ta = u(1); Ts = u(2); Tph1 = u(3); Q0 = u(4);

% Define constants
        UPh2 = INITPH2(6); UlossPh2 = 4.8; Cp = 4179; Asph = 0.65; L = 5.7; g = 9.81;
        dT = 1; a = 9.8e-3; % discrete time step and time constant = 102 secs.
        T0m = (Tph1+Tph2)/2; d=11.1e-3; %mean diameter d = (di+do)/2
        Aph = pi*d*L;

% Calculate thermophysical properties:
        rho0 = convert(T0m,2);

% Calculate Preheater2 Heat Flows
        qlossPh2 = UlossPh2*Asph*(Ts-Ta);
        if Q0 == 0
          Tph2s = 0;
        else
          Tph2s = Ts-(qlossPh2/rho0/Q0/Cp + Ts - Tph1)*exp(-UPh2*Aph/rho0/Q0/Cp);
        end;

% Calculate DISCRETE state derivative
        if flag == 2
                Tph2 = (1-a)*Tph2 + a*Tph2s;
                sys = Tph2;

% prepare outputs
        elseif flag == 3
                sys = Tph2;

% Next Time interval for discrete update
        elseif flag == 4
                sys = T+dT;
        else
                sys=[];
        end;
end;
```

Appendix 4(*b*)

Analytical model M-files

Venturi condenser equations: *venturi.m*

```
function y = venturi(u);

%           VENTURI
%
%           An M-file that calculate the pressure and temperature in the venturi condenser for the three-effect Falling-Film
%           Evaporator.
%
%           U = VENTURI(U);
%
%           Inputs:
%           U = [Pa Ta Ts Te1 Te2 Te3 mcPh1 mvPh1 mv1 mv2 Pv]
%           Pv is the Pvent output fedback from previous sample time.
%  Outputs:
%           Y = [Tvent Pvent mvent]
%           Tvent = venturi driving water outlet temp.
%           Pvent = venturi suction pressure.
%           mvent = [mcPh2 mc1 mc2 mc3 mcPh1 mv] are the condensate and vapour flows into the venturi in (kg/s).

%           Written by Nigel Russell
%           Dept. Production Technology,
%           Massey University, 1997

% Assign inputs
Pa = u(1); Ta = u(2); Ts = u(3); Te1 = u(4); Te2 = u(5); Te3 = u(6);
mcPh1 = u(7); mv = u(8); Mv1 = u(9); Mv2 = u(10); Pv = u(11);

% Define Constants
g = 9.81; Ulossv = 4.8; Asv = 0.105; cp = 4180; mj = 0.3; Tj = 20;
CvcPh2 = 1.92e-5; Cvc1 = 2.16e-5; Cvc2 = 2.72e-5; Cvc3 = 3.3e-5;

% Calculate thermophysical properties
r3 = convert(Te3,3); Ps = convert(Ts,1);
P1 = convert(Te1,1); P2 = convert(Te2,1); P3 = convert(Te3,1);

% Calculate condensate flows
mcPh2 = CvcPh2*sqrt(Ps-Pv); mc1 = Cvc1*sqrt(Ps-Pv);
mc2 = Cvc2*sqrt(P1-Pv);    mc3 = Cvc3*sqrt(P2-Pv);

% Condensate flows limited by the vapour flowing into the shells
mv1 = Mv1/te; mv2 = Mv2/te;
mc2 = min(mc2,mv1); mc3 = min(mc3,mv2);

% Calculate Venturi Heat Flows
qv = mv*r3;
mc = mcPh2+mc1+mc2+mc3+mcPh1;

% calculate condensate flow temp.
if mc == 0,
        qc = 0; Tc = 0;
else
        Tc = ((mcPh2+mc1)*Ts + mc2*Te1 + mc3*Te2 + mcPh1*Te3)/mc;
        qc = mc*cp*(Tc-Tj);
end;

% Assume most of heat loss is from the Ph1 vapour (therefore use T3)
qlossV = Ulossv*Asv*(Te3-Ta);

% Calculate driving fluid output temperature
Tvent = Tj + (qc+qv-qlossV)/mj/cp;
if Tvent <= Tj,
        Tvent = Tj;
end;
```

```
% Calculate densities
rhoj = convert(Tj,2); rhoc = convert(Tc,2);
% an polynomial equation for vapour density
rhov = 4.1302e-9*Te3^4 - 2.2265e-8*Te3^3 + 1.7847e-5*Te3^2 + 1.8891e-4*Te3 + 5.8964e-3;

% Geometrical constants
hc = 1.8; Av = pi*21.8e-3^2/4; a = pi*5.5e-3^2/4; dj = 3;
alpha = (dj/21.8)^2; beta = a/Av;
kf = 0.3; %friction constant

% Calculate venturi pressure
b1 = beta*(Pa-rhoj*g*hc);
b2 = mj^2/rhoj/alpha/Av^2;
b3 = (kf+beta)*(mj+mv+mc)^2/2/rhoj/Av^2/beta^2;

R = mv/mc*sqrt(rhoc/rhov);
Z = (1+R)/(1-alpha)/Av^2;

if mv==0        %Then R = 0
        b4 = Z*mc^2*alpha/(1-alpha)/rhoc;
else
        b4 = Z*(mc^2*(alpha+R)/(1-alpha)/rhoc - mv^2/R/rhov);
end

Pvent = b1-b2+b3+b4;

% Pressure limited by saturation pressure of venturi water
Psat = convert(Tvent,1);
Pvent = max(Psat,Pvent);

mvent=[mcPh2 mc1 mc2 mc3 mcPh1 mv];
y = [Tvent Pvent mvent];
```

calca.m

Calculates the cross-sectional area and volume of the effect level and distribution plate level.

```
function [Ad,Al,Vl] = CalcA(h,L,eff)

%       CALCA
%       Calculates cross-sectional areas of distribution plate and effect levels and volume of liquid in the effect level.
%       Inputs are the heights of the two levels and the effect number.
%
%       [Ad,Al,Vl] = CALCA(h,L,eff);
%
%       Al = CALCA(L,eff);          If effect level area only is desired, or
%       [Al,Vl] = CALCA(L,eff);     If effect level area and volume is desired.

%       Written by N.T. Russell
%       Dept. Production Technology, Massey University, 1995

if nargin == 2
        eff = L;
        L = h;
        h = [];
end

% Calculate area of liquid on distribution plate
if h <= 0.026,
        Ad = 0.00388;
elseif h < 0.071 & h > 0.026,
        Ad = 0.00388+0.06*sqrt(0.0235^2-(0.0495-h)^2);
else
        Ad = 0.00445;
end;

if eff == 3, k = 5.95;
else, k = 3.04; end

% Calculate the surface area and volume of liquid in the bottom of the effect.
```

```
if L < 1.02,
        Al = 0.001792;
        Vl = (L+0.2)*Al;
elseif L < 1.06,
        Al = 0.002168;
        Vl = (L+0.2)*0.001792+(L-1.02)*0.000376;
elseif L < 1.11,
        Al = pi/4*(0.0218^2+(0.5*(L-1.06)+0.0478)^2);
        Vl = 0.002273+pi/6*((0.5*(L-1.06)+0.0478)^3-0.0478^3)+pi/4*0.0218^2*(L-1.06);
elseif L < 1.39,
        Al = 0.004536;
        Vl = 0.0024365+(L-1.11)*Al;
elseif L < 1.415,
        Al = pi/4*(0.0728^2+(k*(L-1.39)+0.0218)^2);
        Vl = 0.0037066+pi/12/k*((k*(L-1.39)+0.0218)^3-0.0218^3)+pi/4*0.0728^2*(L-1.39);
else
        Al = pi/4*((k*0.025+0.0218)^2+0.0728^2);
        Vl = 0.00389+(L-1.415)*Al;
end;

if nargout <= 2,
        Ad = Al;
        Al = Vl;
end
```

ffcalcq0.m

Feed flow (Q_0) calculation based on current plant conditions.

```
function Q0 = ffCalcQ0(LT,Pa,Te1,N0)

%       FFCALCQ0
%       This function calculates the feed flowrate of the evaporator, Q0 (in m^3/s) for the feedforward flow configuration.
%
%       Q0 = FFCALCQ0(LT,Pa,Te1,N0);
%       where:  LT = Tank level (m)
%               Pa = Atm. pressure (Pa)
%               Te1 = Eff1 Temperature (C) - used to calculate P1
%               N0 = Feed pump speed (rpm)

%       Written by N.T. Russell
%       Dept. Production Technology, Massey University, 1995

%       Equivalent lengths:
%       Std tubes: Length = 7.6m + 16 elbows => Lf1 = 21.5m
%       Preheater tubes: Length = 12.7m + 20 elbows => Lf2 = 20.3m
%       where: elbow = 40*dia.

% Define constants
ap0 = 3.65e-2; hN = 4.82; g = 9.81; f = 0.009; Cvn1 = 5.36e-7; Lf1 = 21.5; d1 = 0.0218; Lf2 = 20.3; d2 = 0.0095;

% Use approximate densities
rho1 = 998;         % Liq. density in feed tank (@ 20°C)
rho2 = 977.5;       % Liq. density at nozzle (@ 70°C)
rho3 = 990;         % Liq. density at average feed temp (ie. @ T = (70+20)/2 = 45°C)

P1 = convert(Te1,1); % Performs calculation of saturation pressure.
Ppump = ap0*N0.^2;
Ptank = g*rho1*LT;
Ptop = g*rho2*hN;
Pfric = 32*rho3*f/(pi^2)*(Lf1/d1^5+Lf2/d2^5);

Q0 = sqrt((Pa-P1+Ppump+Ptank-Ptop)/(1/(Cvn1^2)+Pfric));

if LT <= 0, Q0=0; end
```

calctp.m

Calculates an estimate of the time delay between effects (τ_p).

```
function t = calctp(Qp,e)

%       CALCTP
%       Routine to calculate the approx. transport delay between Eff1 and Eff2, between Eff2 and Eff3 & from Eff3 to output.
%       t = CALCTP(Qp,e)
%               e is the effect number.
%               Qp must be in m^3/s.

%       Written by N.T. Russell
%       Dept. Production Technology, Massey University, 1995

if e == 3
        if Qp <= 0, t = 80;             % Define maximum for t
        else, t = 4.106e-3/Qp;
        end
else
        if Qp <= 0, t = 60;             % Define maximum for t
        else
                t = 2.61e-3/Qp;
        end
end
```

calctph.m

Calculates the preheater residence time, τ_{ph}

```
function tph = calctph(Q0)

%           CALCTPH
%           Routine to calculate the preheater residence time.
%           tf = CALCTPH(Q0)        Q0 must be in m^3/s

%           Written by N.T. Russell
%           Dept. Production Technology, 1995

if (Q0 <= 0),
        tph = 15;     % Define maximum time
else
        tph = 4.04e-4/Q0;
end
```

convert.m

Estimates the thermophysical properties of water, given the temperature (or pressure).

```
function Y = Convert(X,n)

%           CONVERT
%           Converts either temperature or pressure values into values of a related thermophysical property, ie. saturation
%           pressure, temperature, enthalpy or density.
%
%           Y = CONVERT(X,n);
%
%           X is pressure or temperature data, n defines which conversion is made:
%
%           n = 0 : P (in kPa) to sat. Temperature
%           n = 1 : T to sat. Pressure (in Pa)
%           n = 2 : T to density (kg/m^3)
%           n = 3 : T to enthalpy of vaporisation (J/kg)
%           The conversion equations were determined by simple regression using polyfit.m.

%           Written by N.T. Russell
%           Dept. Production Technology, Massey University, 1995

if all(n ~= [0 1 2 3])
        error('n must be an integer between 0 and 3')
end
```

```
[r,c]= size(X);
X=X(:);

if n == 0          % Converting P to Tsat
        coeff=[-9.2297e-10; 3.3532e-7; -4.8971e-5; 3.7023e-3; -0.15783; 4.2720; 15.221];
end

if n == 1          % Converting T to Psat
        coeff = [0; 0; 1.0113e-3; -4.5077e-2; 4.5712; -14.735; 960.68];
end

if n == 2          % Converting T to density
        coeff = [0; 0; 0; 0; -3.2924e-3; -0.10424; 1001.4];
end

if n == 3          % Converting T to enthalpy
        coeff = [0; 0; 0; 0; -2.0131; -2.2063e3; 2497.9e3];
end

Y = [X.^6 X.^5 X.^4 X.^3 X.^2 X ones(size(X))] * coeff;
Y = reshape(Y,r,c);
```

Initcond.m

Script file which sets up the initial conditions for the model simulation.

```
% Initcond
% Definition of initial conditions for the feed-forward evaporator model.  Sets all initial conditions for the model before the start
% of the simulation routine.
% Creates five global variables which contain the initial values.


%  Written by N.T. Russell
%  Dept. Production Technology, Massey University, 1995

Ts0=Ts(1); Tph1=Tph1(1); Tph2=Tph2(1); Tl1=Tl(1); Tl2=Tl2(1); Tl3=Tl3(1);
L1=L1(1); L2=L2(1); L3=L3(1); LT=0.75;
N0o=N0(1); N1o=N1(1); N2o=N2(1); N3o=N3(1);

%Tsp=80;Tph1=37.5; Tph2=70; Tl1=72; Tl2=64; Tl3=42;
%L1=1; L2=1; L3=1; LT=0.75;
%Q1=210; Q2=200; Q3=190; Q0=220;
%Ts0=80; N1=1340; N2=1760; N3=1840; N0=670;
%Q0=ffCalcQ0(LT,Pa,T1,N0o);

Qin=220/36e6; Pa=101300; Ta=18; T0=18;
kp=4; ki=0.1; kd=1; Tc=21; tsam=5; Cv = 5.90e-7;
T1=Tl1+0.5; T2=Tl2+0.3; T3=Tl3+0.2;
UPh1=1900; UPh2=1800; U1=2600; U2=2500; U3=1800;

T = (size(N0,1)-1)*tsam;

% Calculation and conversion of initial conditions.

At1=0.3313; At2=0.3313; At3=0.2137; Aph=0.1988;
q1=U1*At1*(Ts0-T1); q2=U2*At2*(T1-T2);
q3=U3*At3*(T2-T3); qPh1=UPh1*Aph*(Tph1-T0)/log((T3-T0)/(T3-Tph1));

Q0=Q0(1)/3.6e6; Qp1=Q1(1)/3.6e6; Qp2=Q2(1)/3.6e6; Qp3=Q3(1)/3.6e6;
Mv1=10*4/3600; Mv2=10*4/3600; Mv3=20*4/3600;
McPh1 = 8/3600*calctph(Q0); mvPh1=12/3600;
```

```
mv1=Mv1/convert(T1,2)/4; mv2=Mv2/convert(T2,2)/4; mv3=Mv3/convert(T3,2)/4;

Qd1=Q0; %Qp1=Qd1-mv1;
Qd2=Qp1; %Qp2=Qd2-mv2;
Qd3=Qp2; %Qp3=Qd3-mv3;

h1=(Qd1/2.36e-4)^2/19.62; h2=(Qd2/2.36e-4)^2/19.62;
h3=(Qd3/2.36e-4)^2/19.62;

INITPH1=[LT;McPh1;Tph1;Ta;Qin;T0;T3;Q0;Mv3;qPh1;Tph1;UPh1];
INITPH2=[Tph2;Ta;Ts0;Tph1;Q0;UPh2];
INIT1=[h1;L1;T1;Tl1;Mv1;Ta;Ts0;Tph2;T2;N1o;Qd1;q1;Tph2;T1;Q0;U1];
INIT2=[h2;L2;T2;Tl2;Mv2;Ta;T1;T3;N2o;Qd2;q2;T1;T2;Qp1;Tl1;U2];
INIT3=[h3;L3;T3;Tl3;Mv3;Ta;T0;Tph1;T2;N3o;Qd3;q3;Pa;T2;T3;Qp2;mvPh1;Tl2;Cv;U3];

clear Aph At1 At2 At3 UPh1 UPh2 U1 U2 U3 N1o N2o N3o N0o
clear Ta T0 Ts0 Pa Qtop Q0 Cv4 LT L1 L2 L3
clear Qp1 Qp2 Qp3 Qd1 Qd2 Qd3 q1 q2 q3 qPh1 qPh2
clear T1 T2 T3 Tph1 Tph2 Tl1 Tl2 Tl3 h1 h2 h3 Mv1 Mv2 Mv3 MvPh1 MvPh2
```

Fevaprun.m

Run file for the model simulation.

```
% Run file for simulating the feedforward evaporator model.

% Written by N.T. Russell
% Dept. Production Technology
% 1995

% Definition of Initial Conditions
Initcond

global INITPH1 INITPH2 INIT1 INIT2 INIT3

t0=clock;
rk45('ffevap',T,[],[1e-6,0.5,1]);  % Simulate the evaporator
runtime(clock,t0);

P1 = convert(T1,1); P2 = convert(T2,1); P3 = convert(T3,1);

% Plotting commands
subplot(121),plot(Time,[Tl1 Tl2 Tl3 Tph1 Tph2 ts]),ylabel('Temps (C)'),title('Temperatures')
subplot(222),plot(Time,[L1 L2 L3]),ylabel('Levels (m)'),title('Levels')
subplot(224),plot(Time,[Qp1 Qp2 Qp3 Q0]),ylabel('Flows (l/hr)'),title('Flowrates')
%subplot(121),plot(Time,[N1 N2 N3 N0]),ylabel('Pumps. (rpm)')
%subplot(222),plot(Time,[P1 P2 P3]),ylabel('Press. (kPa)')

model=[T0 ts Tph1 Tph2 Tl1 Tl2 Tl3 L1 L2 L3 Q0 Qp1 Qp2 Qp3 N0 N1 N2 N3 P1 P2 P3];
i=find(rem(Time,tsam)==0);
model=model(i,:); Pvent=Pvent(i); mvent=mvent(i,:); Tvent=Tvent(i); Mv1=Mv1(i); Mv2=Mv2(i); Mv3=Mv3(i);
```

Appendix 4(*c*)

SIMULINK block diagrams

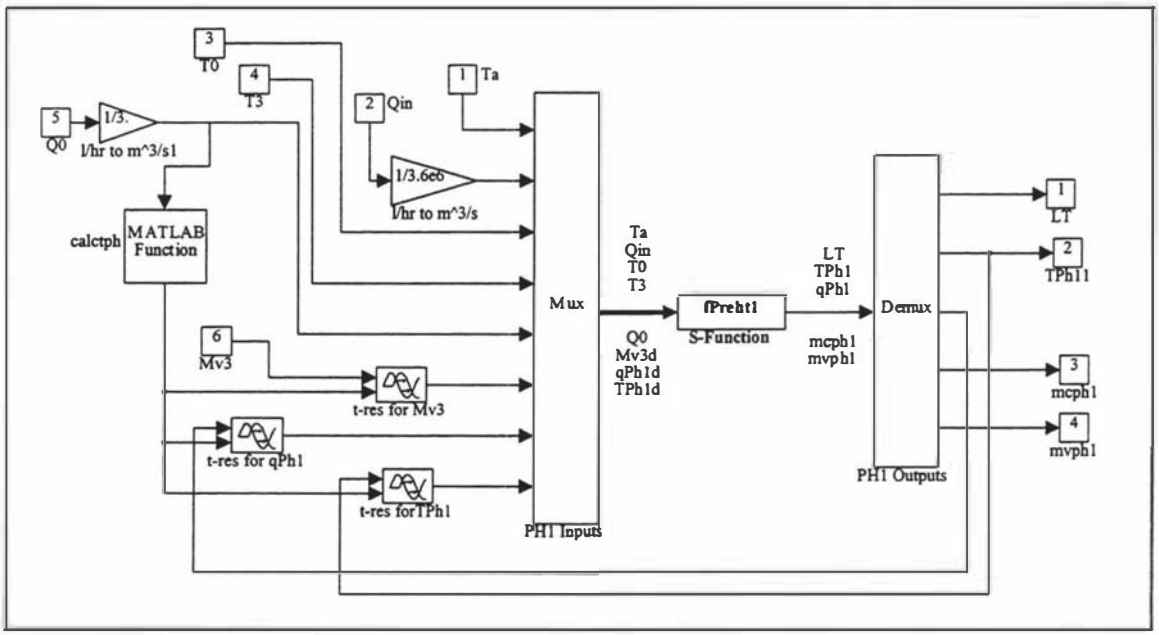


Figure A4-1: ***Preheater 1*** **block -** ***ffevap.m***

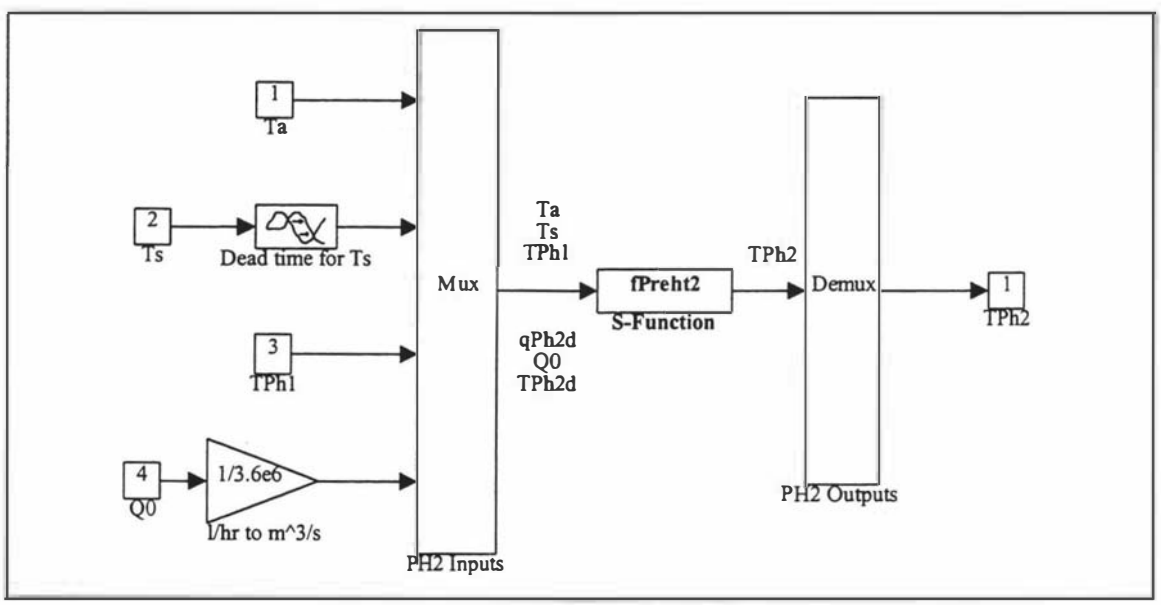


Figure A4-2: ***Preheater 2*** **block -** ***ffevap.m***

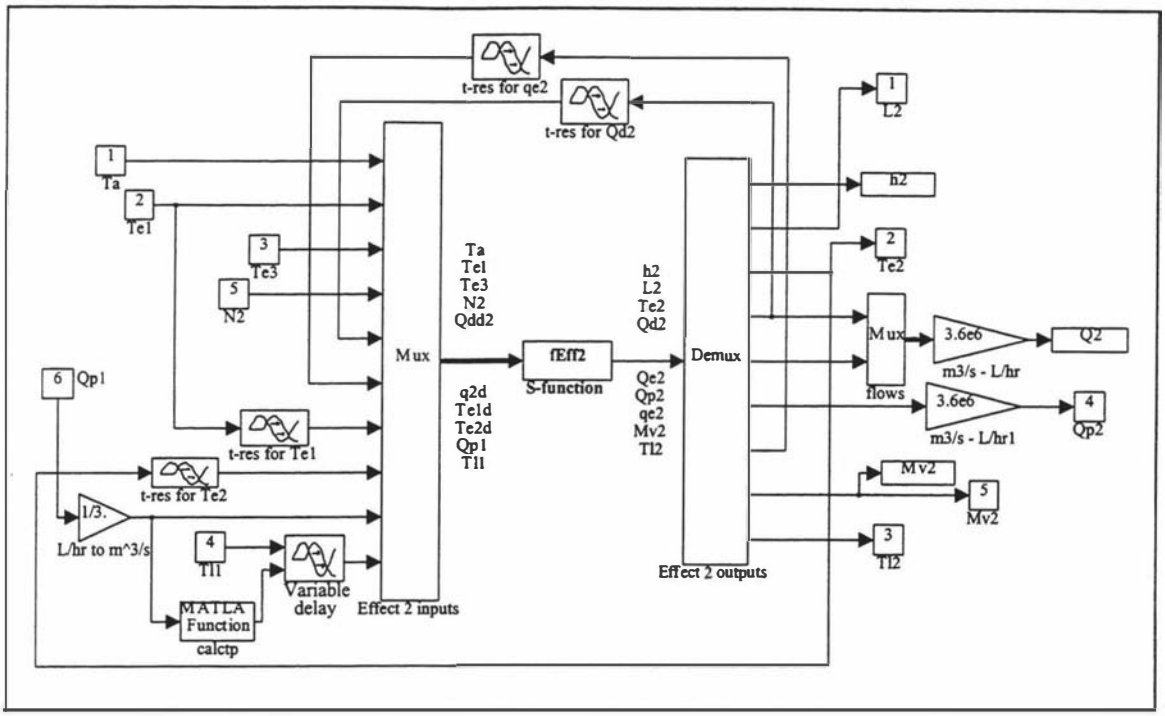


Figure A4-3: *Effect 2* block - *ffevap.m*

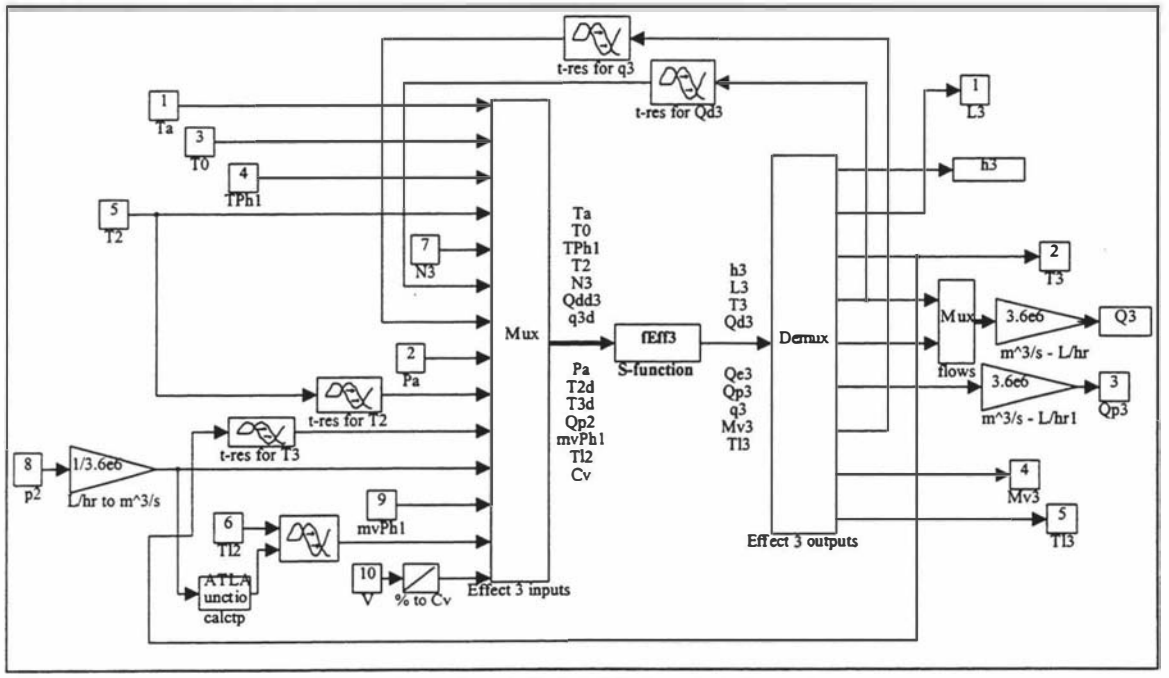


Figure A4-4: *Effect 3* block - *ffevap.m*

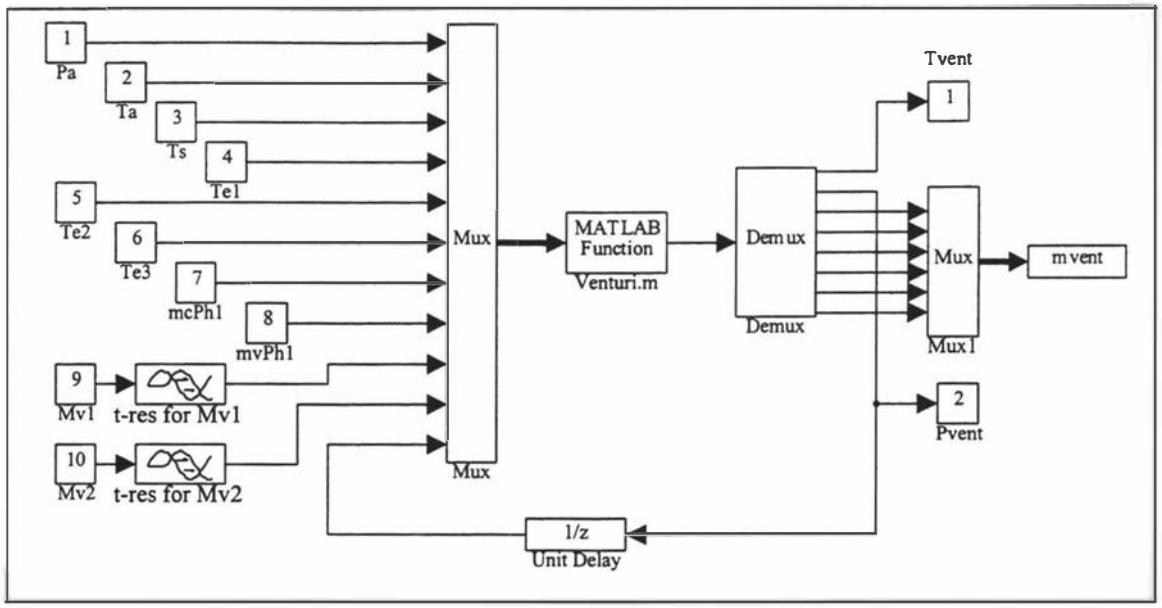


Figure A4-5: *Venturi* block - *ffevap.m*

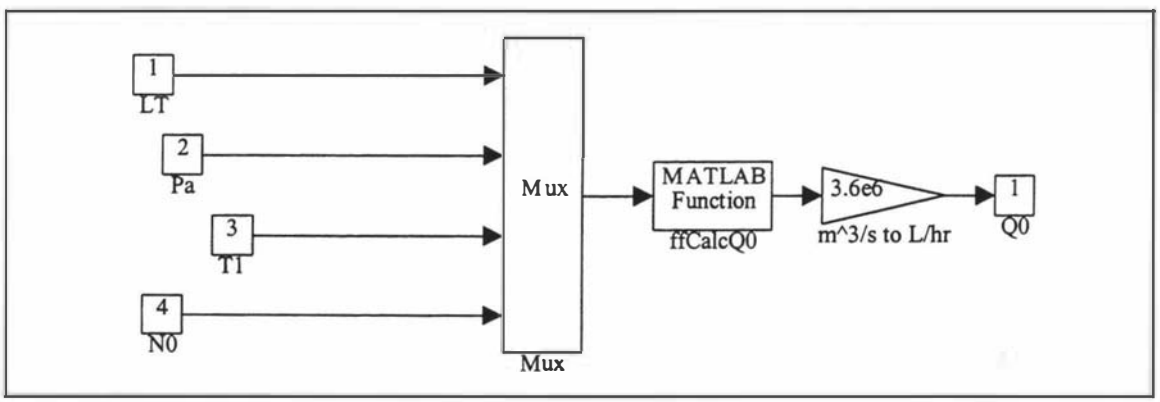


Figure A4-6: *ffcalcQ0* block - *ffevap.m*

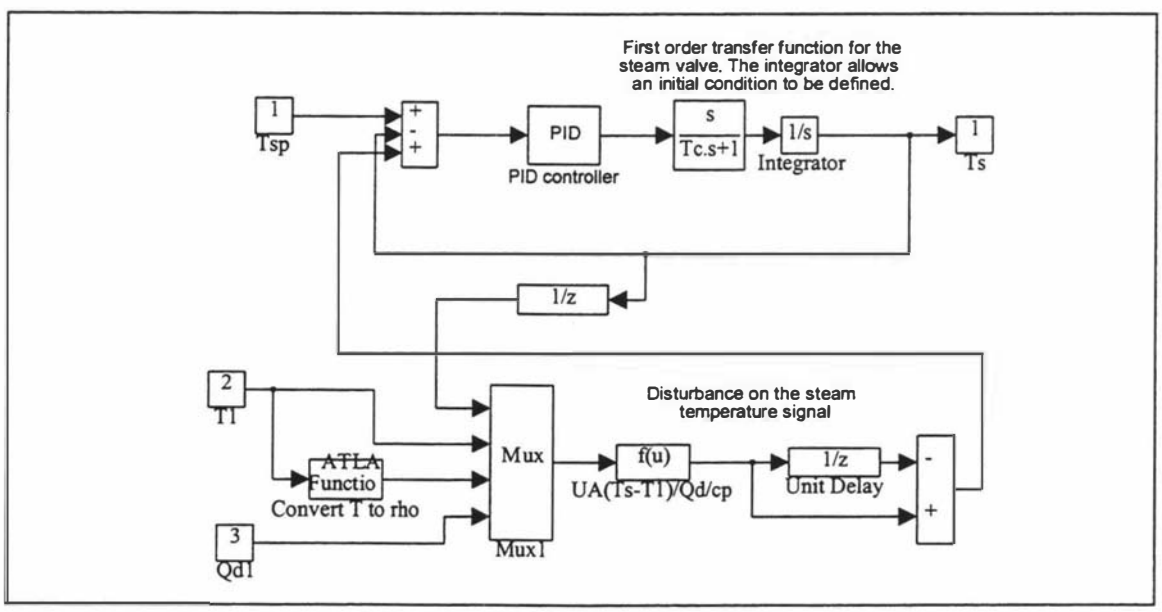


Figure A4-7: *calcTs* block - *ffevap.m*

Appendix 5

Analytical Model Validation

Contents

Appendix 5(*a*)

Steady-state data

The steady-state trial involved varying the feed pump speed and the steam pressure setpoint and holding the plant at specific points. Nine combinations of feed pump and steam pressure settings were used (Table A5-1).

The steady-state trial resulted in the nine operating points given in Table A5-2.

Table A5-1: Steady-state trial feed pump and steam pressure setpoint settings

	Feed pump (rpm)	Steam pressure (setpt.) (kPa)
1	1092	40
2	365	40
3	742	40
4	1092	50
5	365	50
6	742	50
7	557	60
8	920	60
9	1255	60

Table A5-2: Steady-state data from the evaporator – Trial 0

	Operating points								
Variable	1	2	3	4	5	6	7	8	9
T_s	75.52	75.61	76.02	81.12	81.24	81.73	86.08	86.67	86.97
T_{Ph1}	35.41	36.42	36.09	37.28	38.28	38.02	39.46	39.05	38.35
T_{Ph2}	62.39	66.04	64.79	67.56	71.66	70.41	76.95	74.46	71.71
T_1	69.04	68.85	69.35	72.94	72.28	73.02	76.13	76.99	77.19
T_2	63.75	62.13	63.18	66.58	64.42	65.74	6739	69.01	70.06
T_3	42.89	41.59	42.24	44.53	42.95	43.70	44.37	45.59	46.51
Q_0	307.4	208.0	244.0	286.2	179.9	225.6	175.2	230.6	289.6
Q_1	286.1	191.8	224.8	264.1	173.5	210.7	154.6	206.0	259.7
Q_2	274.9	188.3	217.9	251.4	164.7	199.5	154.9	199.2	250.2
Q_3	255.5	170.5	204.0	231.2	146.2	180.9	139.2	175.6	229.4
N_1	1495	1301	1343	1406	1246	1301	1148	1278	1365
N_2	2199	1676	1849	2036	1522	1715	1446	1699	2013
N_3	1889	1825	1840	1862	1817	1828	1795	1817	1852
Levels	All the levels were constant at 1.05 m								

Note: Temperatures are in °C, flowrates in l/hr and pump speeds in rpm.

Appendix 5(*b*)

Fitting of model parameters

From the steady-state trial data heat transfer coefficients were estimated by setting the model derivatives to zero. For the nine operating points the heat transfer coefficients in Table A5-3 were calculated assuming that the theoretical heat loss coefficients were correct.

Table A5-3: HTCs calculated from steady-state data

Operating point	U_{Ph1} (W/(m^{2}K))	U_{Ph2} (W/(m^{2}K))	U_1 (W/(m^{2}K))	U_2 (W/(m^{2}K))	U_3 (W/(m^{2}K))
1	1807	1647	2787	2794	1841
2	1965	1783	3029	2989	1803
3	2088	1904	3528	3325	1781
4	2006	1881	4617	3899	1797
5	1883	1761	4193	3465	1830
6	1761	1662	3979	3272	1889
7	1824	1799	2451	2534	1738
8	1964	1876	2719	2712	1719
9	2116	1971	2922	2885	1676
Mean with 95% CI	1935 ± 288	1809 ± 252	3358 ± 1743	3097 ± 992	1786 ± 153

Using the plant data from the three trials (Trial 0, 2 and 3) the flow pump, nozzle and outlet valve flow constants were optimised in order to predict the flowrates. The following tables summarise the results of the optimisation calculations.

Table A5-4: Optimal pump constants based on plant trial data

Constant	Units	Data set			
		Trial 0 ***Steady-state*** ($\times 10^{-2}$)	**Trial 2** ***Random*** ($\times 10^{-2}$)	**Trial 3** ***Random*** ($\times 10^{-2}$)	**Mean** with 95% CI (10^{-2})
a_{P0}	Pa/rpm^2	4.0593	3.6662	3.4032	3.7096 ± 1.42
a_{P1}	Pa/rpm^2	2.1919	2.0822	2.1600	2.1447 ± 0.24
a_{P2}	Pa/rpm^2	2.3149	1.8027	1.6317	1.9164 ± 1.53
a_{P3}	Pa/rpm^2	2.9533	3.0253	3.0102	2.9963 ± 0.16

Table A5-5: Optimal nozzle and valve constants based on plant trial data

Constant	Units	Data set			Mean with 95% CI ($\times 10^{-7}$)
		Trial 0 *Steady-state* ($\times 10^{-7}$)	Trial 2 *Random* ($\times 10^{-7}$)	Trial 3 *Random* ($\times 10^{-7}$)	
C_{N1}	$m^3(Pa^{½}s)$	4.6345	4.9885	5.0940	4.9057 ± 1.04
C_{N2}	$m^3(Pa^{½}s)$	5.7684	5.4556	5.1323	5.4521 ± 1.37
C_{N3}	$m^3(Pa^{½}s)$	2.5118	2.9517	3.2059	2.8898 ± 1.51
C_V	$m^3(Pa^{½}s)$	6.3070	5.4459	5.9433	5.8987 ± 1.86

Summary of Model Parameters ~ after validation

The following tables summarise the final parameters used in the analytical model.

Table A5-6: Model parameters for evaporation effect systems

Symbol	Equation	Units	Effect 1	Effect 2	Effect 3
a_P	3.12	Pa/rpm^2	2.14×10^{-2}	1.92×10^{-2}	3.00×10^{-2}
A_h	3.1	m^2	2.36×10^{-4}		
A_d	3.2	m^2	Function of L_d, see Appendix 3(a)		
A_l	3.8	m^2	Function of L, see Appendix 3(a)		
A_e	3.28	m^2	0.3313	0.3313	0.2137
A_s	3.31	m^2	1.059	1.059	1.099
A_{sl}	3.324	m^2	0.363	0.363	0.379
A_{sp}	3.25	m^2	0.550	0.550	0.942
c_p	3.23/26/30	J/(kgK)	4191	4185	4179
$C_{N2}/C_{N3}/C_V$	3.14	$m^3(Pa^{½}s)$	5.45×10^{-7}	2.89×10^{-7}	5.90×10^{-7}
d	3.13	m	0.0218		
l_e	3.13	m	10.5	10.5	21.8
f	3.13	–	0.009		
h_N	3.15	m	4.82		
τ_e	3.5	s	4		
U_e	3.28	$W/(m^2K)$	2600	2500	1800
U_{loss}	3.31	$W/(m^2K)$	4.8	4.5	3.8
U_{lossl}	3.24	$W/(m^2K)$	4.8	4.5	3.8
V_l	3.22	m^3	Function of L, see Appendix 3(a)		
V_p	3.18	m^3	2.61×10^{-3}	2.61×10^{-3}	4.48×10^{-3}
K_h	3.1	–	1		

Table A5-7: Model parameters for feed system

Symbol	Equation	Units	Value
a_{P0}	3.36	Pa/rpm^2	3.71×10^{-2}
A_T	3.36	m^2	0.159
C_{N1}	3.36	$m^3(Pa^{1/2}s)$	4.91×10^{-7}
d, d_p	3.35	mm	21.8, 9.5
f	3.35	–	0.009
l_f, l_p	3.35	m	21.5 , 20.3
τ_c	3.81	s	21
V_f	3.38	m^3	3.737×10^{-3}

Table A5-8: Model parameters for the preheater system

Symbol	Equation	Units	Preheater 1	Preheater 2
A_{ph}	3.47/3.48	m^2	0.1988	
A_{sph}	3.47/3.48	m^2	0.65	
c_{ph}	3.47/3.48	J/(kgK)	4179	
d_p	3.34	m	9.5×10^{-3}	
U_{ph}	3.47/3.48	$W/(m^2K)$	1800	1780
U_{lossph}	3.47/3.48	$W/(m^2K)$	15	19
V_{ph}	3.58	m^3	4.04×10^{-4}	
$\tau_{c(ph)}$	3.40	s	102	

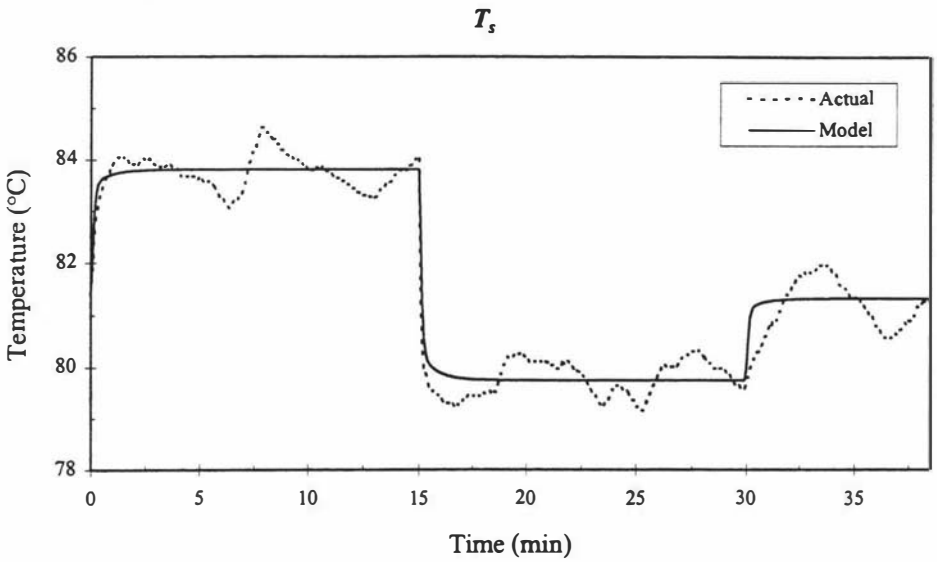
Table A5-9: Model parameters for the condenser system

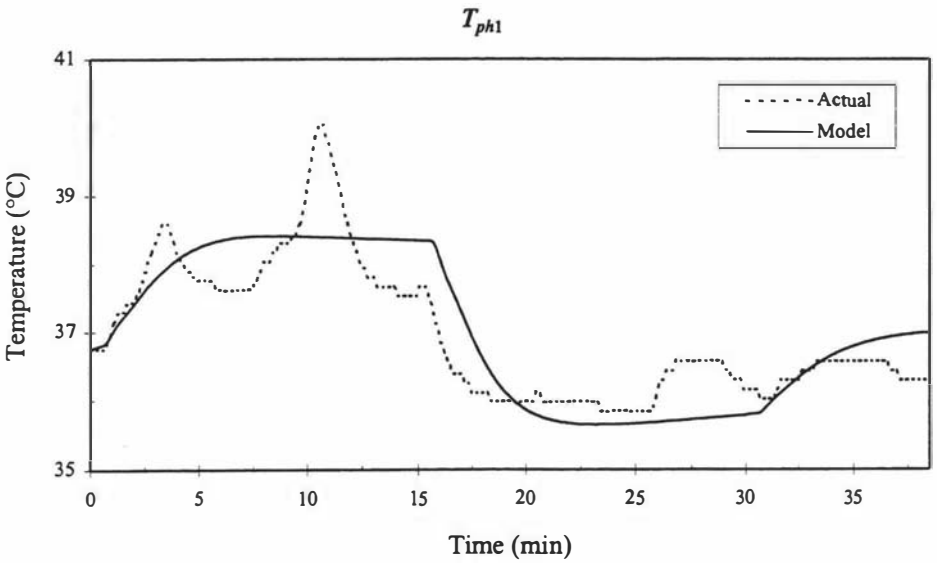
Symbol	Equation	Units	Value
α	3.70	–	0.0189
A_{sv}	3.63	m^2	0.105
A_v	3.65	m^2	3.73×10^{-4}
β	3.71	–	0.0637
c_j	3.62	J/(kgK)	4182
C_{vc1}	3.51	$kg/(Pa^{1/2}s)$	2.16×10^{-5}
C_{vc2}	3.51	$kg/(Pa^{1/2}s)$	2.72×10^{-5}
C_{vc3}	3.51	$kg/(Pa^{1/2}s)$	3.30×10^{-5}
C_{vcph2}	3.51	$kg/(Pa^{1/2}s)$	1.92×10^{-5}
h_c	3.68	m	1.8
K_f	3.67	–	0.3
U_{lossv}	3.63	$W/(m^2K)$	4.8

Appendix 5(*c*)

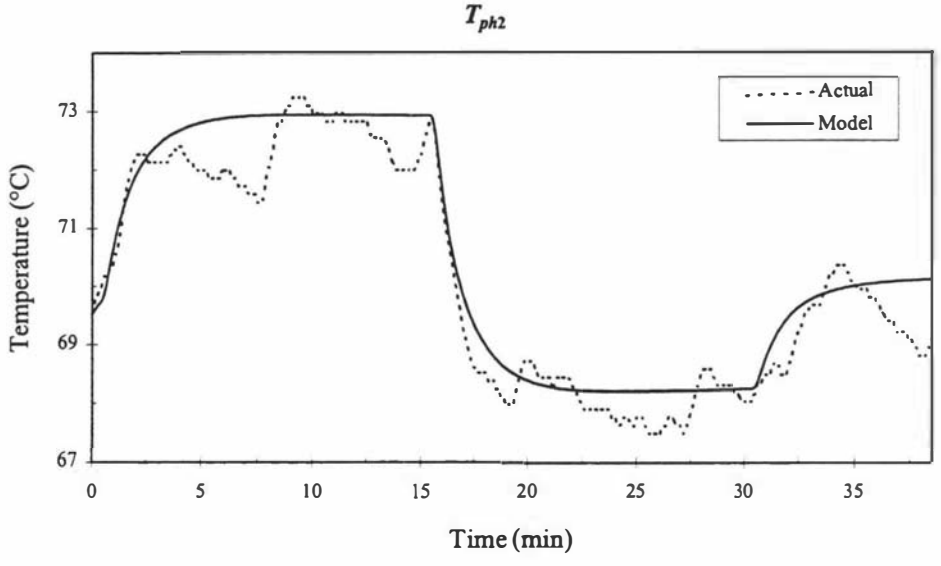
Model Simulations

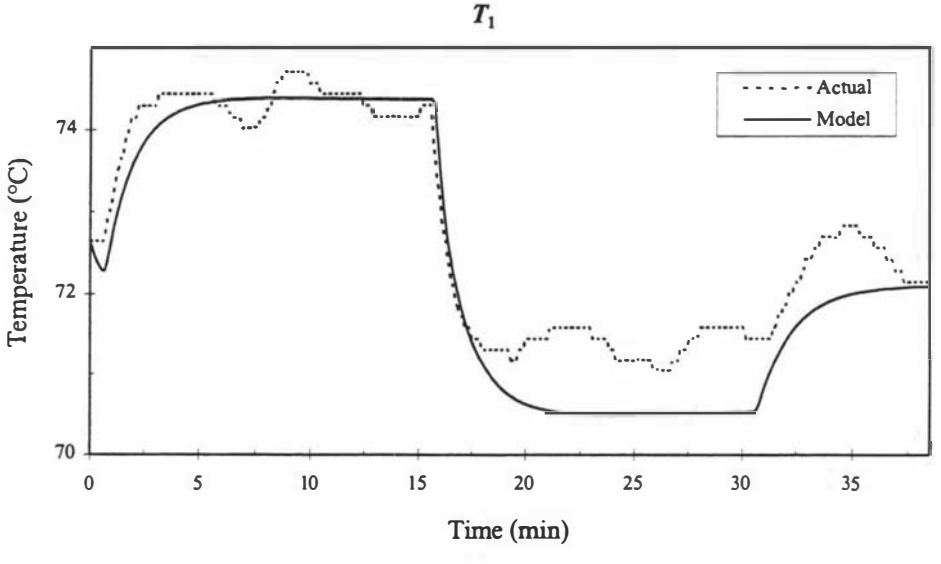
Trial 1 - Step-responses

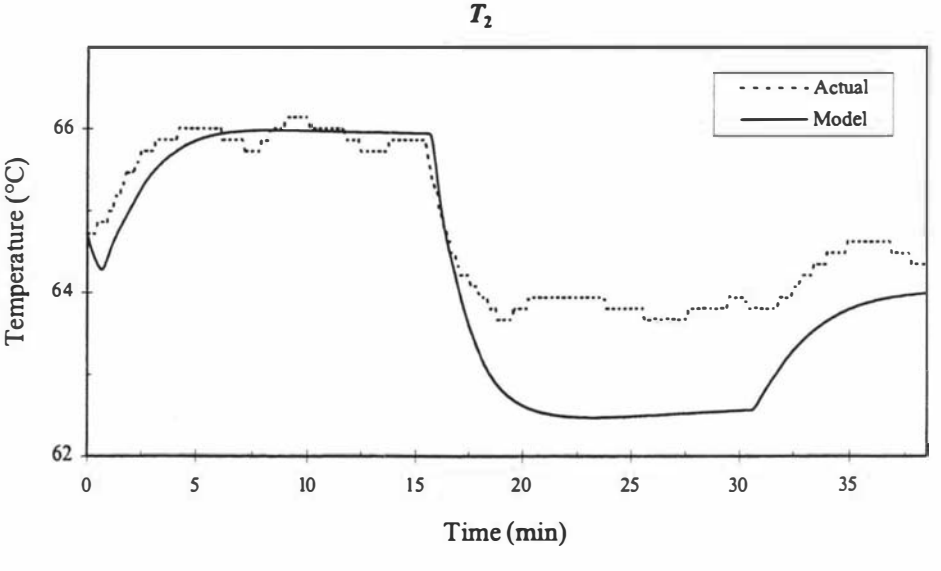




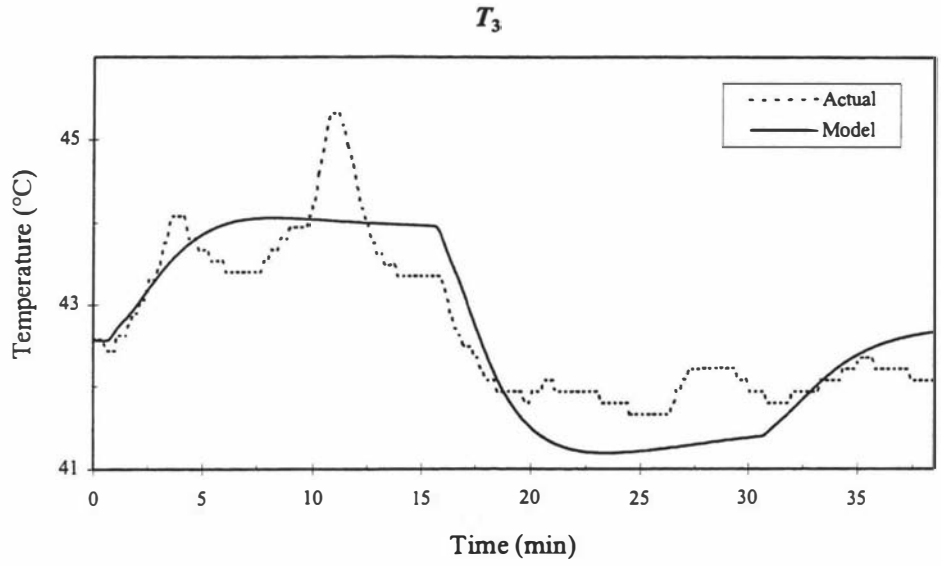
Model simulation: Trial 1 - Step-responses

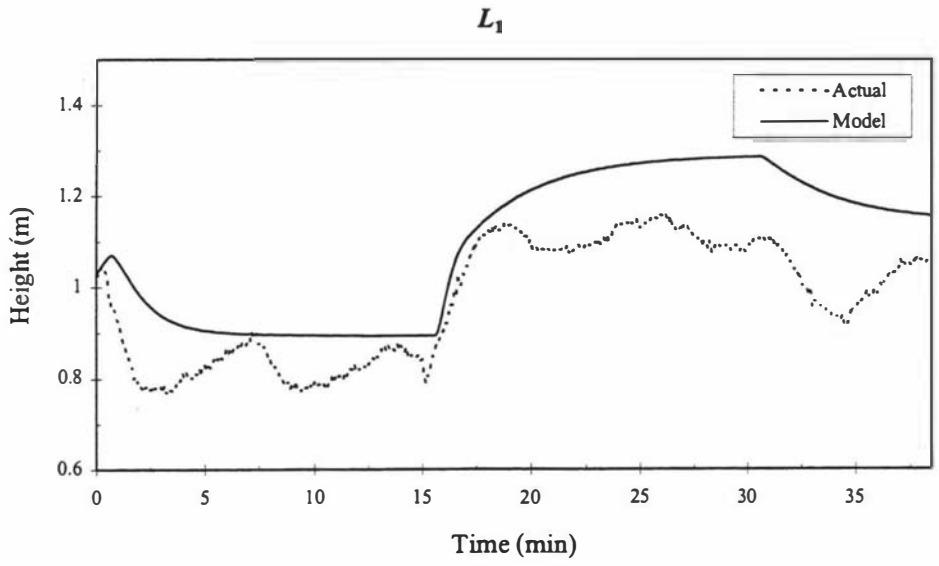


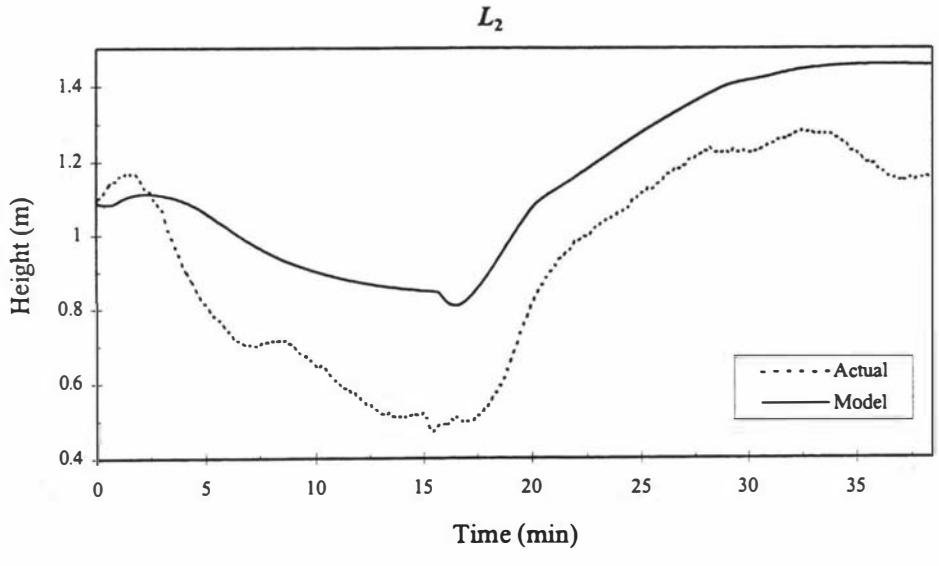




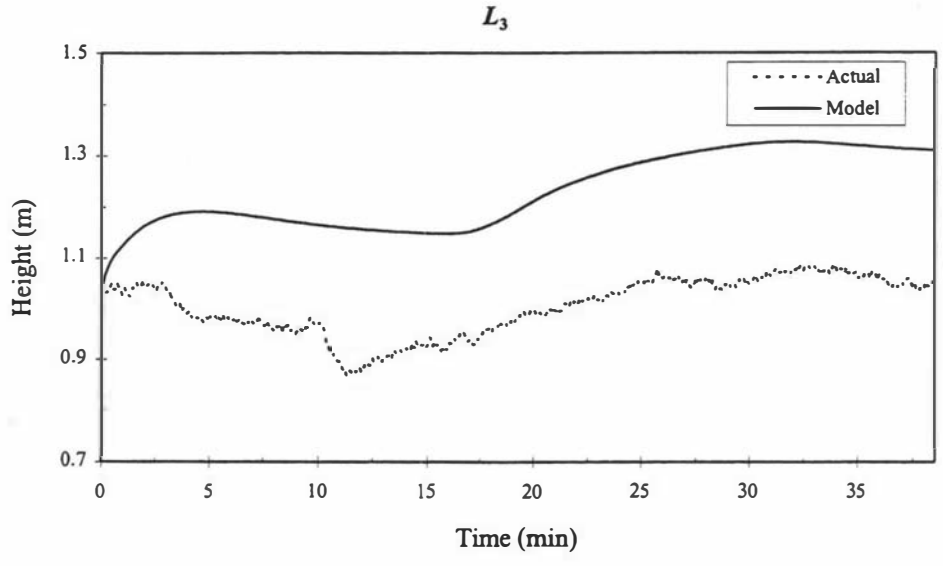
Model simulation: Trial 1 - Step-responses

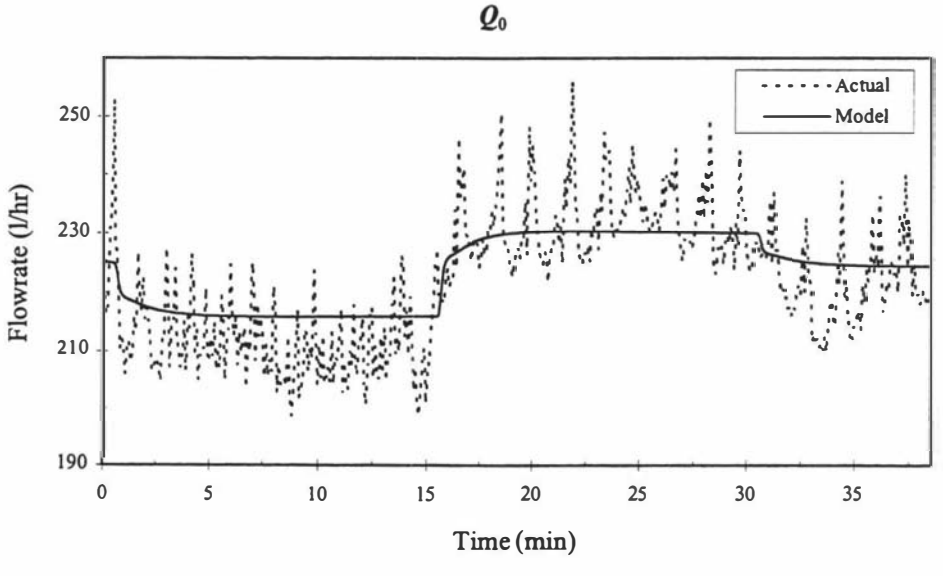


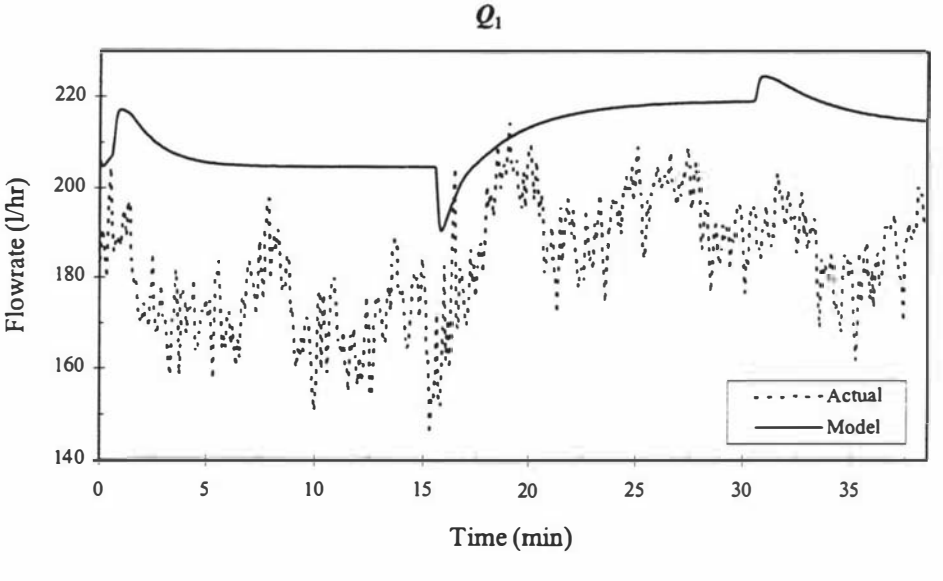




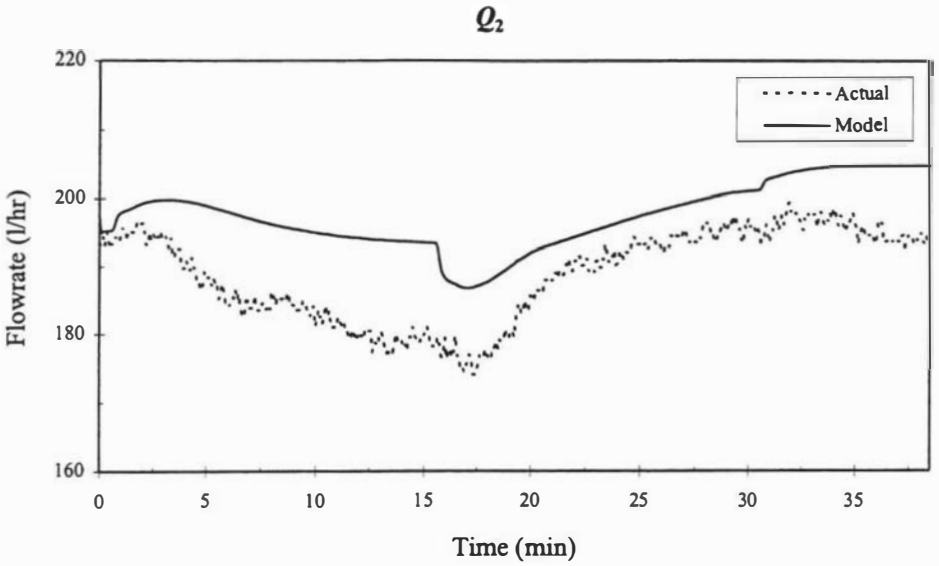
Model simulation: *Trial 1 - Step-responses*

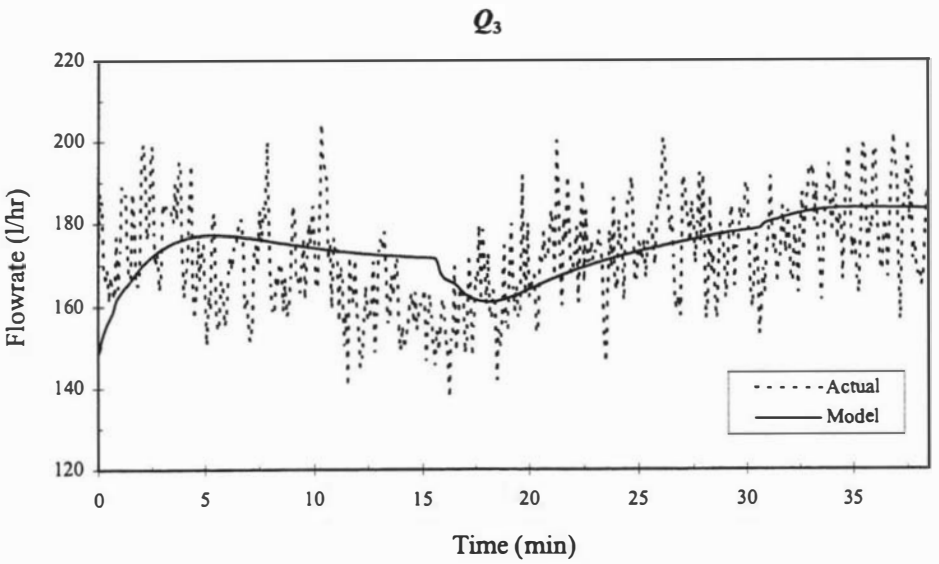






Model simulation: *Trial 1 - Step-responses*





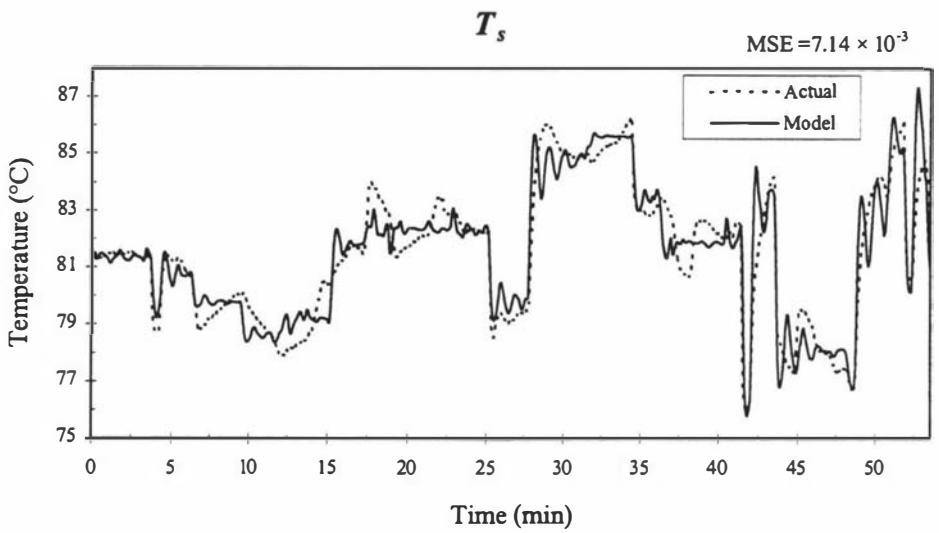
Random input trial

Using the model parameters listed in Tables A5-6 to A5-9 the analytical simulation was run with the same random inputs as used in Trial 4. When compared with the actual measured data the simulation estimates gave the following mean-square-error (MSE) results. The data used to calculate these MSE values have been standardised (ie. scaled to lie between 0 and 1) so that a direct comparison can be made between the results for different variables.

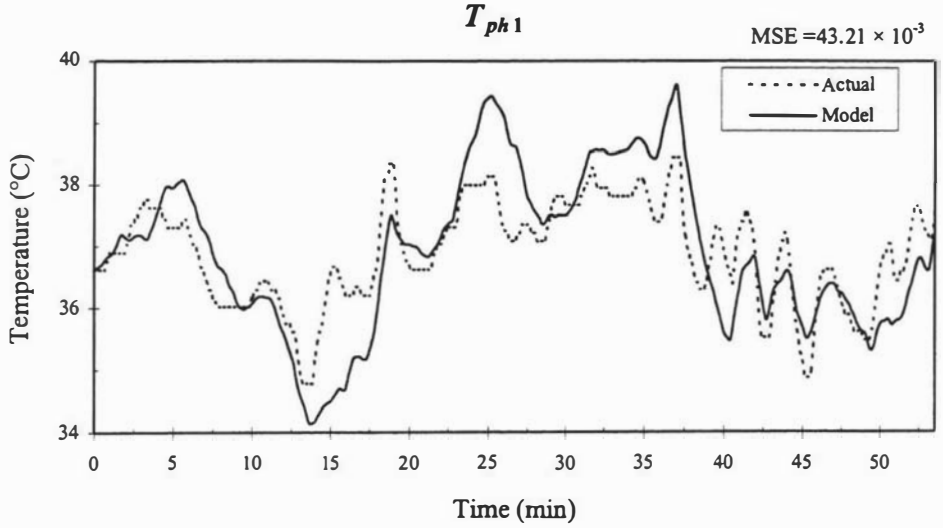
Table A5-10: MSE results for the Trial 4 model estimates

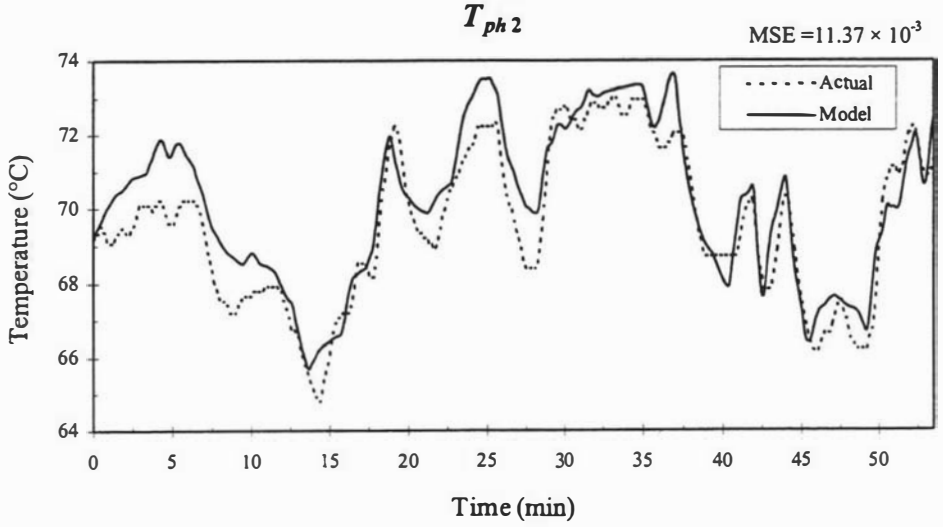
Variable	Trial 4 data ($\times 10^{-3}$)
T_s	7.14
T_{Ph1}	43.21
T_{Ph2}	11.37
T_1	8.07
T_2	26.55
T_3	68.88
L_1	21.25
L_2	105.57
L_3	82.99
Q_0	5.19
Q_1	28.30
Q_2	60.31
Q_3	9.82

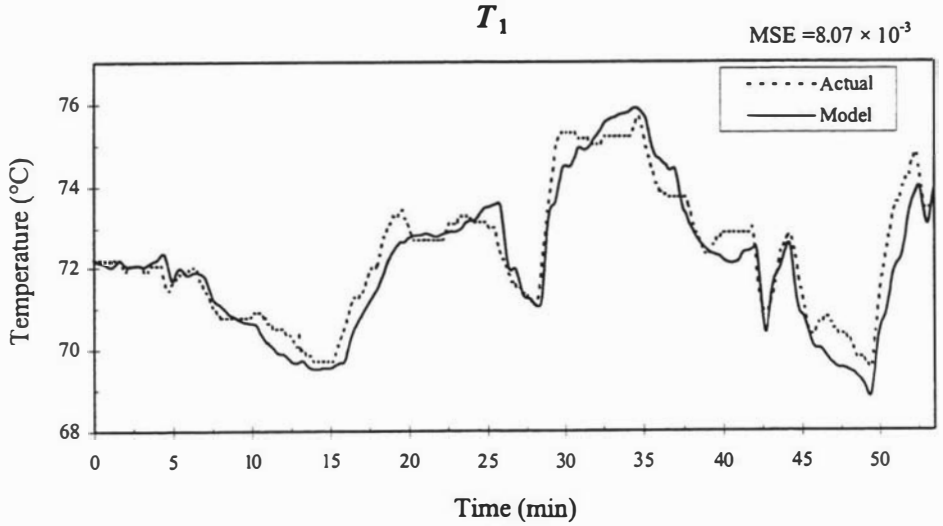
Model simulation : Trial 4 - Validation data



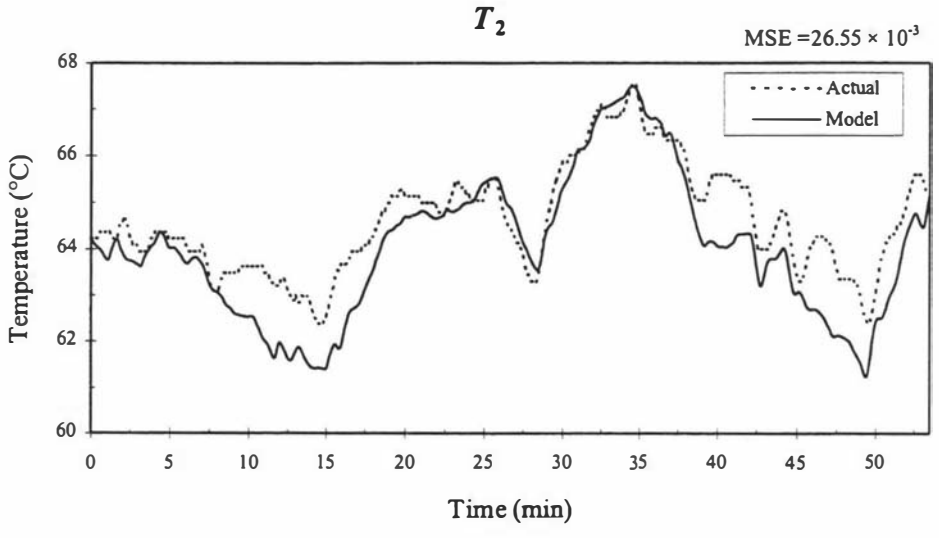
Model simulation : Trial 4 - Validation data

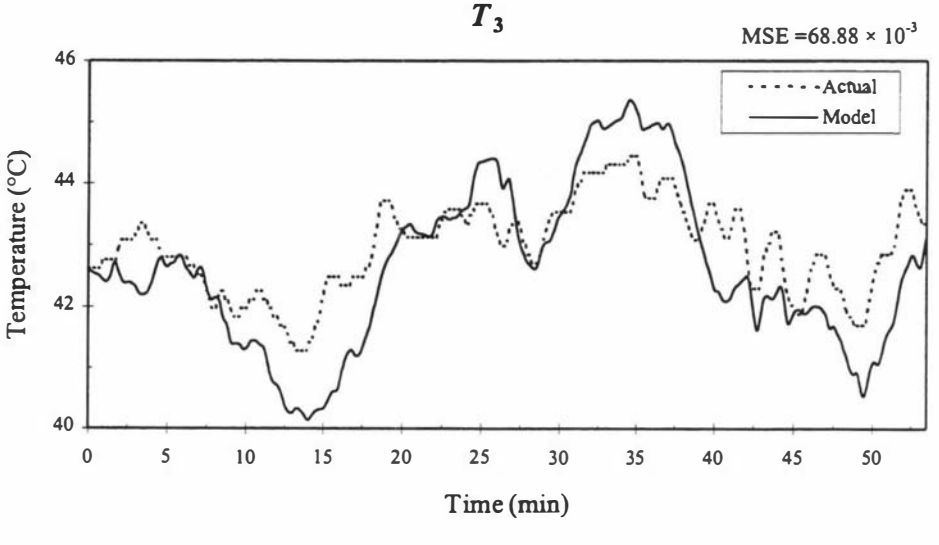


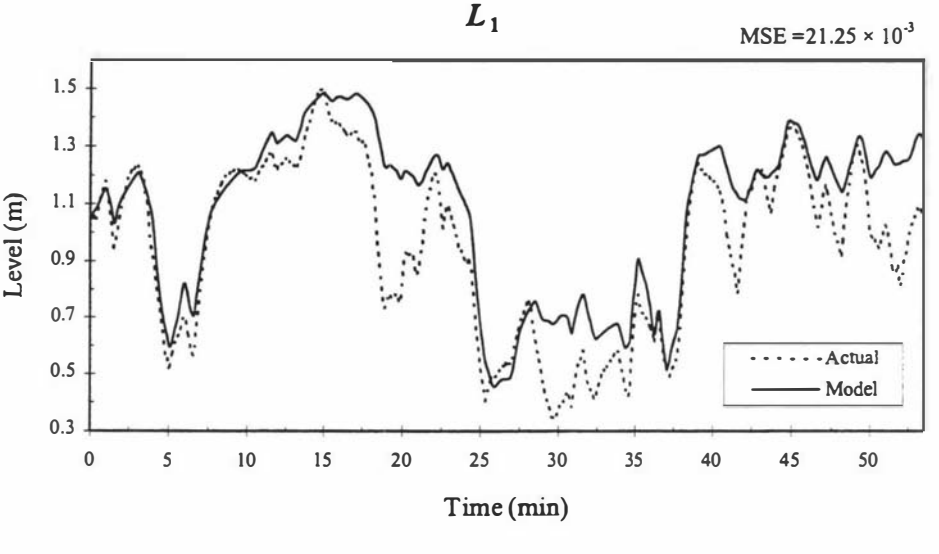




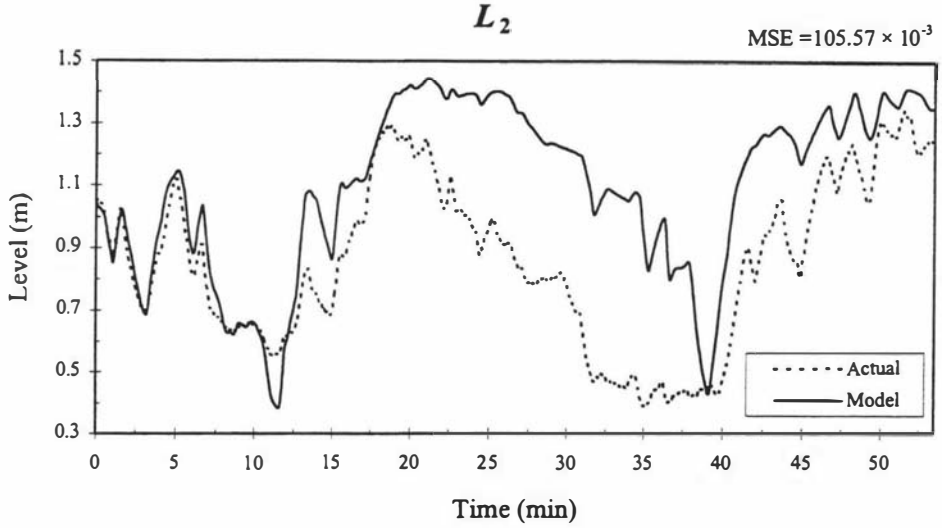
Model simulation : Trial 4 - Validation data

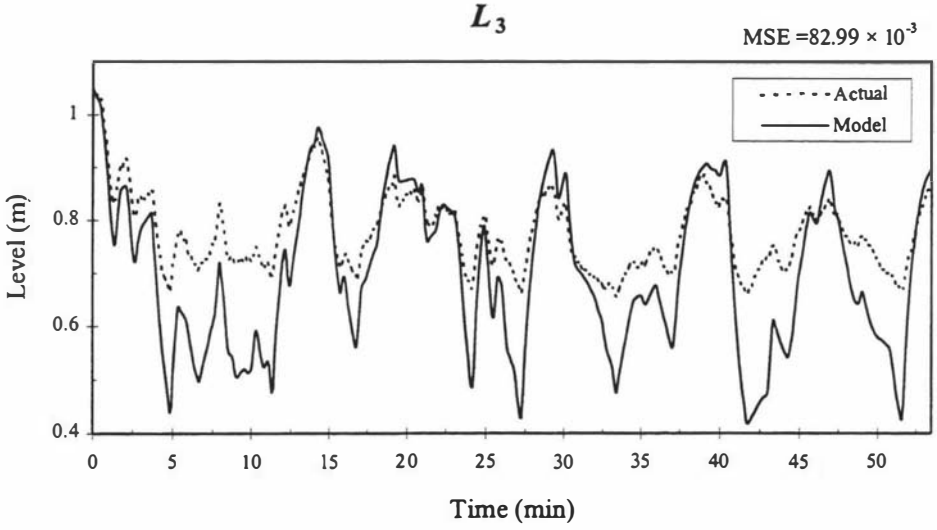


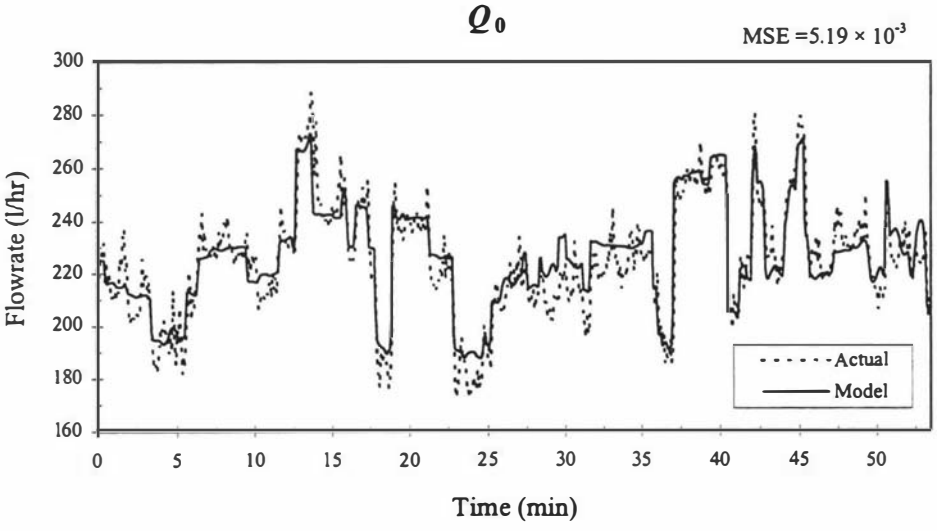




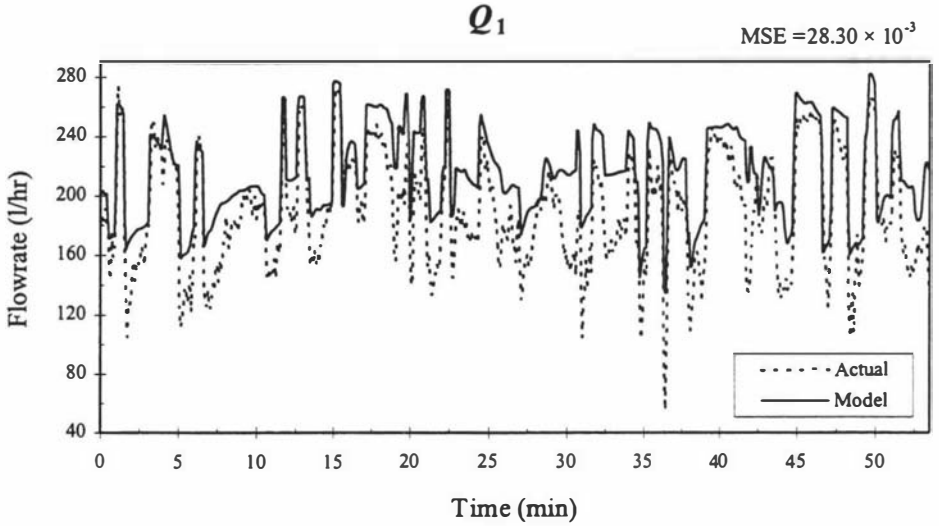
Model simulation : Trial 4 - Validation data

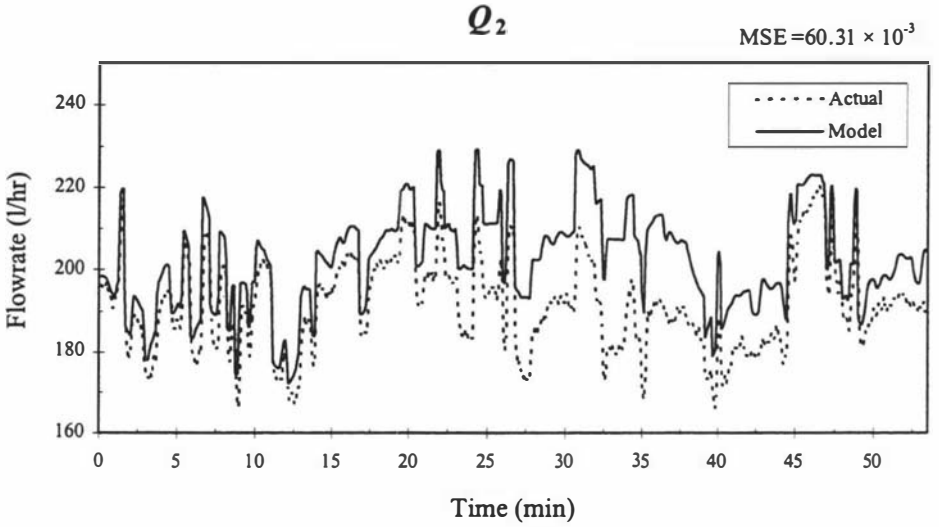


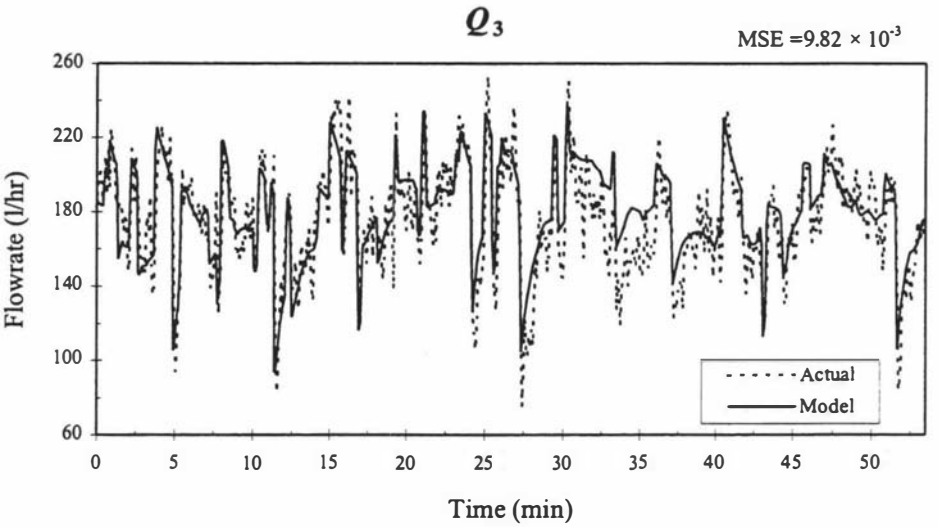




Model simulation : Trial 4 - Validation data







Appendix 6

Derivation of the Backpropagation Algorithm

The backpropagation algorithm is used to calculate the change in weights for any given neuron in a network according to

$$\Delta w_{ij}^{l} = -\mu \frac{\partial E}{\partial w_{ij}^{l}} \tag{A6.1}$$

where Δw_{ij}^{l} is the change in weight of the connection between node i (layer l), and node j (layer $l-1$) and

E is the sum of squares error for the network outputs.

The backpropagation method is a generalisation of the Delta Rule (§4.1.2) for neurons connected in feedforward networks. The Delta Rule is given by:

$$\Delta w_{ij}^{l} = \mu \delta_i^{l} a_j^{l-1} \tag{A6.2}$$

where: $\delta_i^{l} = \left(d_i^{l} - a_i^{l}\right)$,

d_i^{l} is the target output for node i in layer l and

a_j^{l-1} is the output of node j in the previous layer (layer l–1).

If we have an L-layered feedforward neural network, then the output error for the network with n output neurons each having activation function, $f_L(.)$, is given by

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(d_i^{L} - a_i^{L}\right)^2 = \frac{1}{2}\sum_{i=1}^{n}\left(d_i^{L} - f_L\left(s_i^{L}\right)\right)^2 \tag{A6.3}$$

where

$$s_i^{L} = \sum_{j=1}^{m} w_{ij}^{L} a_j^{L-1} + b_i^{L} \tag{A6.4}$$

is the weighted sum of m inputs into node i, plus the bias term.

For backpropagation the activation function, $f(.)$, is required to be continuous and differentiable for all its range.

Using the chain rule: $\dfrac{\partial E}{\partial w_{ij}^{L}} = \dfrac{\partial E}{\partial s_i^{L}} \times \dfrac{\partial s_i^{L}}{\partial w_{ij}^{L}}$,

where $\frac{\partial s_i^L}{\partial w_{ij}^L} = a_j^{L-1}$ from equation (A6.4) above,

and $$\frac{\partial E}{\partial s_i^L} = \frac{\partial E}{\partial a_i^L} \times \frac{\partial a_i^L}{\partial s_i^L} = -\left(d_i^L - a_i^L\right) f_L{}'\left(s_i^L\right), \tag{A6.5}$$

then $$\frac{\partial E}{\partial w_{ij}^L} = -\left(d_i^L - a_i^L\right) f_L{}'\left(s_i^L\right) a_j^{L-1}. \tag{A6.6}$$

Since node i is in the output layer (layer L) d_i^L is known. Using equations (A6.1), (A6.2) and (A6.6), we get a nonlinear version of the Delta Rule:

$$\Delta w_{ij}^L = \mu \delta_i^L a_j^{L-1}$$

with $$\delta_i^L = \left(d_i^L - a_i^L\right) f_L{}'\left(s_i^L\right). \tag{A6.7}$$

However if node i is not in the output layer ($l < L$) then d_i^l is not known prior to training and δ_i^l must be calculated by an alternative method.

By inspection of equations (A6.5) and (A6.7), it is evident that:

$$\delta_i^l = -\frac{\partial E}{\partial s_i^l} \tag{A6.8}$$

δ_i^l is defined as the square-error derivative associated with the i^{th} element in layer l.

Using equation (A6.8) and the chain rule:

$$\delta_i^l = -\frac{\partial E}{\partial a_i^l} \times \frac{\partial a_i^l}{\partial s_i^l} = -\frac{\partial E}{\partial a_i^l} f_l{}'\left(s_i^l\right). \tag{A6.9}$$

The term, $\partial E/\partial a_i$, represents the sensitivity of the network's output error to the output of node i in layer l. This node exhibits its influence on the error through all the nodes in the following layers. Thus $\partial E/\partial a_i$ can be expressed as a function of the sensitivities to nodes in the next layer as follows:

$$\frac{\partial E}{\partial a_i^l} = \sum_{h=1}^{m} \left(\frac{\partial E}{\partial s_h^{l+1}} \times \frac{\partial s_h^{l+1}}{\partial a_i^l} \right). \tag{A6.10}$$

Where s_h^{l+1} is the summed input to the h^{th} node in the following layer (l+1) which is made up of m nodes. Equation (A6.10) uses the chain rule summed over all the nodes in the following layer since node i influences the error through all the these nodes.

From equation (A6.10):

$$\frac{\partial E}{\partial a_i^l} = \sum_{h=1}^{m} \frac{\partial E}{\partial s_h^{l+1}} w_{hi}^{l+1} = -\sum_{h=1}^{m} \delta_h^{l+1} w_{hi}^{l+1}. \tag{A6.11}$$

From equation (A6.9) and equation (A6.11):

$$\delta_i^l = \sum_{h=1}^{m} \left(\delta_h^{l+1} w_{hi}^{l+1} \right) f_l'\left(s_i^l\right). \tag{A6.12}$$

Therefore the procedure for calculating the square-error derivative for a given hidden node, i, involves multiplying each δ associated with each node in the layer downstream (layer l+1), by their respective weights on the connections from node, i. These weighted square-error derivatives are then summed together and multiplied by the derivative of node i's activation function at its current operating point.

To summarise, using the backpropagation method the network weights are updated according to

$$\Delta w_{ij}^l = \mu \delta_i^l a_j^{l-1},$$

where, for an output neuron ($l = L$): $\delta_i^L = \left(d_i^L - a_i^L\right) f_L'(s_i)$, and

for a hidden-layer neuron ($l < L$): $\delta_i^l = \sum_{h=1}^{m} \left(\delta_h^{l+1} w_{hi} \right) f_l'(s_i)$.

Similarly the update equation for the bias weights is given by

$$\Delta b_i^l = -\mu \frac{\partial E}{\partial b_i^l},$$

and since an input to a bias is always unity it can be easily shown that

$$\Delta b_i^l = \mu \delta_i^l.$$

Appendix 7

Sub-Network Training Algorithm

Backpropagation through time with Levenberg-Marquardt optimisation

The training methodology used for the evaporator neural network modelling combined the backpropagation through time (BPTT) and Levenberg-Marquardt (LM) algorithms.

LM optimisation method is discussed in Chapter 4, while the BPTT training algorithm is introduced in Chapter 5. A brief outline of the sub-network training algorithm is given in Chapter 6, but this is extended here to give the full equations and details for the training methodology used.

After the initial network weights have been set to small random values the training algorithm consists of the following steps:

1. Presentation of the input data to the network to generate the network outputs and calculation of the output errors.

The input data matrix, $\boldsymbol{X}$, is passed through the recurrent network to produce the estimated outputs, $\hat{\boldsymbol{Y}}$.

$$\hat{\boldsymbol{Y}} = \Gamma_{\boldsymbol{W}_k}[\boldsymbol{X}] \tag{A7.1}$$

where $\Gamma_{\boldsymbol{W}_k}[.]$ is the nonlinear function representation of the network with weight vector $\boldsymbol{W}_k$.

The weight vector is made up of all the network weights and is of length N_w.

$$W_k = \left[w_{11}^1 \quad \cdots \quad w_{ij}^L \right] \tag{A7.2}$$

The output prediction error for each time step is calculated (equation A7.3) and the overall performance of the network is measured using the sum of squares of error (equation A7.4).

$$e(t) = y(t) - \hat{y}(t) \tag{A7.3}$$

$$E = \frac{1}{2}\sum_{t=1}^{N} e(t)^2 \tag{A7.4}$$

2. Backpropagation of the error derivatives (deltas) for time, $t = N$ to the current time through each time step.

The following calculations are carried for each time step from time, $t = N$ to $t = 0$.

i) The deltas for the output layer neurons include the deltas from the input nodes at future time steps. Therefore, the delta for the output neuron i (layer L) at time t is

$$\delta_i^L(t) = f_L'(s_i(t))\left(\sum_{\tau=1}^{K_y} \delta^0(t+\tau)\right) \tag{A7.5}$$

where $f_L'(.)$ is the derivative of the output neuron activation function,

$s_i(t)$ is the weighted sum of inputs entering neuron i from the previous layer,

K_y is the number of delayed outputs used, and

$\delta^0(t+\tau)$ is the corresponding delta from the input layer at time t + τ

Note that $\delta^0 = 0$ for all $t > N$.

ii) The deltas for hidden layers at time t are then calculated. The delta for neuron j in layer l is given by

$$\delta_j^l(t) = \sum_i \left(\delta_i^{l+1}(t) w_{ij}^{l+1}\right) f_l'(s_j(t)) \tag{A7.6}$$

iii) The deltas at time t for the input layer are calculated. The delta for neuron j in the input layer ($l = 1$) is given by

$$\delta_j^0(t) = \sum_i \left(\delta_i^1(t) w_{ij}^1\right) \tag{A7.7}$$

The input layer deltas relay the required information from time $t = n + 1$ to the output layer of the previous time step ($t = n$).

3. Formation of the Jacobian matrix of the network.

Using the deltas obtained in step 2, each row of the Jacobian matrix is formed using

$$\frac{\partial e(t)}{\partial w_{ij}^l} = \delta_i^l(t) a_j^{l-1}(t) \tag{A7.8}$$

to give

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial e(1)}{\partial w_1} & \frac{\partial e(1)}{\partial w_2} & \cdots & \frac{\partial e(1)}{\partial w_{N_w}} \\ \frac{\partial e(2)}{\partial w_1} & \frac{\partial e(2)}{\partial w_2} & \cdots & \frac{\partial e(2)}{\partial w_{N_w}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e(N)}{\partial w_1} & \frac{\partial e(N)}{\partial w_2} & \cdots & \frac{\partial e(N)}{\partial w_{N_w}} \end{bmatrix} \tag{A7.9}$$

$\boldsymbol{J}$ has dimensions $(S_o \times N) \times N_w$, where S_o is the number of output neurons.

4. Perform gradient check

The optimisation is assumed to have converged if the norm of the gradient vector is less than a predefined value, ∇_{min}.

$$\nabla = \left\| \boldsymbol{J}^T \boldsymbol{e} \right\|_2 \tag{A7.10}$$

if $\nabla < \nabla_{min}$, terminate algorithm.

5. Update the network weights

The change in weights is calculated using the LM modification to the Gauss-Newton method, where λ is set to a small value for the initial iteration.

$$\Delta \boldsymbol{W}_k = -\frac{\boldsymbol{J}^T \boldsymbol{e}}{[\boldsymbol{J}^T \boldsymbol{J} + \lambda \boldsymbol{I}]} \tag{A7.11}$$

$$\boldsymbol{W}_{new} = \boldsymbol{W}_k + \Delta \boldsymbol{W}_k \tag{A7.12}$$

6. Evaluate the new network.

The inputs are presented to the new network and the output errors calculated.

$$\hat{\boldsymbol{Y}} = \Gamma_{\boldsymbol{W}_{new}}[\boldsymbol{X}]$$

$$e(t) = y(t) - \hat{y}(t)$$

$$E = \frac{1}{2}\sum_{t=1}^{N} e(t)^2$$

If the solution is improved the new weights are accepted and λ decreased, else λ is increased and the algorithm returns to step 5.

If $E < E_{min}$

$\lambda = \lambda \times dec$

$\boldsymbol{W}_{k+1} = \boldsymbol{W}_{new}$

$E_{min} = E$

else

$\lambda = \lambda \times inc$

if $\lambda > \lambda_{max}$, terminate algorithm.

return to step 5

inc and *dec* are the constants used to adapt λ. If λ rises above a predefined maximum value the algorithm terminates.

7. Terminate if solution has converged or the maximum number of epochs has been reached, otherwise continue training.

If $E \leq E_{goal}$ or $k > k_{max}$

terminate algorithm

else

$k = k + 1$

return to step 2

The user defined constants required for the algorithm are listed below (with typical values in brackets).

inc – Increment factor for λ (10)	λ_{max} – Maximum limit for λ (10^{12})
dec – Decrement factor for λ (0.1)	∇_{min} – Minimum limit for gradient (10^{-10})
E_{goal} – Error convergence goal	k_{max} – Maximum number of epochs

Appendix 8

Neural Network MATLAB Files

npattern.m

Creates a pattern data matrix for use in training a time-delayed neural network.

```
function [V,U] = npattern(data,var,Du,Dy,opt);

%NPATTERN   Creates training data matrix for an n-step-ahead
%           network for the falling-film evaporator.
%
%           [V,U] = NPATTERN(data,'var',Du,Dy,opt);
%
%           Given the raw data matrix (data), the target variable name ('var'), the number of delayed observations of the
%           inputs required according to vectors Du and Dy and the scaling option, opt (see SCALE.m), this function
%           generates the appropriate training data matrix.
%           Both scaled data (V) and unscaled data (U) matrices are generated.
%           Du and Dy = [z1 z2 z3] are the delays applied to the inputs (Du) and past outputs (Dy) where delays are in
%           ascending order.
%
%           The file makes use of VARGROUP.m and MOVWIND.m files.  See these for further details.

%           Written by N.T. Russell
%           Dept. Production Technology
%           Massey University
%           1995

if nargin~=5
 error('Incorrect number of input arguments');
end

% Prepare data matrix according to variable to be modelled
d = vargroup(var,data);
[r,c] = size(d);

my=max(Dy); mu=max(Du);
n = max(mu,my);

% Split into input matrix and past output and target vectors
u = d(:,1:c-1);
y = d(:,c);
t = y;

% Scale matrix
us = scale(u,opt);
ys = scale(y,opt);
ts = ys;

% Create pattern matrix of inputs according to moving window, K
Us = movwind(us,Du); % scaled data
U = movwind(u,Du);   % unscaled data
Ys = movwind(ys,Dy); % scaled data
Y = movwind(y,Dy);   % unscaled data
if Dy == 0; % no output feeback
```

```
 Y=[]; Ys=[];
end

d = mu-my;

if d > 0
 Ys = Ys(1+d:r-my,:);
 Y = Y(1+d:r-my,:);
end
if d < 0
 Us = Us(1-d:r-mu,:);
 U = U(1-d:r-mu,:);
end

% Create target vector of correct length
T = t(n+1:r);
Ts = ts(n+1:r);

% Combine inputs and corresponding targets
V = [Us Ys Ts];
U = [U Y T];

if nargout == 1
  U = [];
end

[rows,cols] = size(V);

fprintf('\nThe pattern matrix has %1.0f rows and %1.0f columns\n',rows,cols);
fprintf('The neural net needs %1.0f inputs and 1 output\n\n',cols-1);
```

vargroup.m

Groups training data according to target variable, for use within *npattern.m*.

```
function d = vargroup(var,D);

%VARGROUP   Groups training data according to target variable.
%
%          D = VARGROUP('var',data);
%
%          Given the target variable name 'var', the data matrix and the scaling option this function produces D data
%          matrix. The data matrix has one column for each input variable the last column being the target variable.
%          The input data matrix is the full simulation data (23 columns), ie. data = ...
%          [T0 Ts Tph1 Tph2 T11 T12 T13 L1 L2 L3 Q0 Qp1 Qp2 Qp3 N0 N1 N2 N3 P1 P2 P3 Vs Tsp];
%          To change the training data for a particular network the code can be altered in the appropriate place.

%          Written by N.T. Russell
%          Dept. Production Technology
%          Massey University
%          1995

if nargin~=2
 error('Incorrect number of input arguments');
end

var = abs(var);   % Sets variable to its decimal ASCII representation

if length(var) == 4
 if var == [84 112 104 49] % for Tph1
   var = [104 49];
 elseif var == [84 112 104 50] % for Tph2
   var = [104 50];
 end
end

if length(var) ~= 2
 error('Incorrect variable label')
end
```

```
[R,C]=size(D);

% Filter flow inputs for level networks
q0 = expfilt(D(:,11),0.05); q1 = expfilt(D(:,12),0.05);
q2 = expfilt(D(:,13),0.05); q3 = expfilt(D(:,14),0.05);

% *** T1 ***
if var==[84 49],          % ASCII for 'T1'
  d = [D(:,2) D(:,6) D(:,11) D(:,5)]; % d = [Ts T2 Q0 T1]
end

% *** T2 ***
if var==[84 50],          % ASCII for 'T2'
  d = [D(:,5) D(:,7) D(:,12) D(:,6)]; % d = [T1 T3 Q1 T2]
end

% *** T3 ***
if var==[84 51],          % ASCII for 'T3'
  d = [D(:,6) D(:,3) D(:,13) D(:,7)]; % d = [T2 Tph1 Q2 T3]
end

% *** TPh1 ***
if var==[104 49],         % ASCII for 'h1' (Tph1)
  d = [D(:,7) D(:,11) D(:,3)];      % d = [T3 Q0 Tph1]
end

% *** TPh2 ***
if var==[104 50],         % ASCII for 'h2' (Tph2)
  d = [D(:,2) D(:,3) D(:,11) D(:,4)];       % d = [Ts Tph1 Q0 Tph2]
end

% *** Ts ***
if var==[84 115],         % ASCII for 'Ts'
  d = [D(:,23) D(:,5) D(:,11) D(:,2)];% d = [Tsp T1 Q0 Ts]
end

% *** L1 ***
if var==[76 49]           % ASCII for 'L1'
  d=[q0 q1 D(:,5) D(:,8)];          % d = [Q0 Q1 T1 L1]
end

% *** L2 ***
if var==[76 50]           % ASCII for 'L2'
  d=[q1 q2 D(:,6) D(:,9)]; % d = [Q0 Q1 T2 L1]
end

% *** L3 ***
if var==[76 51]           % ASCII for 'L3'
  d = [q2 q3 D(:,7) D(:,10)];       % d = [Q2 Q3 T3 L3]
end

% *** Q0 ***
if var==[81 48],          % ASCII for 'Q0'
  d = [D(:,5) D(:,15) D(:,11)];     % d = [T1 N0 Q0]
end

% *** Q1 ***
if var==[81 49],          % ASCII for 'Q1'
  d = [D(:,5)-D(:,6) D(:,8) D(:,16) D(:,12)]; % d = [T1-T2 L1 N1 Q1]
end

% *** Q2 ***
if var==[81 50],          % ASCII for 'Q2'
  d = [D(:,6)-D(:,7) D(:,9) D(:,17) D(:,13)]; % d = [T2-T3 L2 N2 Q2]
end

% *** Q3 ***
if var==[81 51],          % ASCII for 'Q3'
  d = [D(:,7) D(:,10) D(:,18) D(:,14)];     % d = [T3 L3 N3 Q3]
end
```

movwind.m

Applies a moving window to a data matrix, for use within *npattern.m*.

```
function v = movwind(m,K)

%MOVWIND  Moving window pattern file
%
%               V = MOVWIND(M,K)
%
%               Given a data matrix M this function generates a patterned matrix (V) based on a moving window defined by
%               vector K, of length n  Each element of K defines the shift of the moving window and must be in ascending order.
%               The output consists of each column of M repeated n times but shifted down by the number of rows defined by K
%               for each repeated column.
%               If M is of dimension r x c then the output matrix, V has dimensions: r-n+1 x c*n.  If using for time series data,
%               then n is the number of delayed observations of each variable at delay K(1), K(2) etc., where the variables are
%               stored in columns.
%
%               The format of the matrix V is as below:
%               V = [u1(t-K(1)) u1(t-K(2))...u1(t-K(n)) u2(t-K(1)) u2(t-K(2))...u2(t-K(n))...]
%               This can be easily reversed by altering the file.

%               Written by N.T. Russell
%               Dept. Production Technology
%               Massey University
%               1996

if nargin~=2
 error('Incorrect number of input arguments');
 end

[r,c] = size(m);
n = length(K);

if max(K) > r
  error('Not enough rows in data matrix')
end

p = r-K(n);  % p = row size of output matrix.

% stack columns of m, n times
b = zeros(r*n,c);
ind = 1:r;
for i=[0:n-1]*r
  b(ind+i,:) = m;
end

% Create pattern matrix using window of length K(1) to K(n)
v = zeros(p*n,c);
ind = 1:p;
for j = [0:n-1]
                v(ind+j*p,:) = b(ind+j*r+K(n)-K(j+1),:);
% this gives: v = [t-K(1), t-K(2), ..., t-K(n-1), t-K(n)]

% NB: if you want to have:-
%       v = [t-K(n), t-K(n-1), ..., t-K(2), t-K(1)]
% then use:
%               v(ind+j*p,:) = b(ind+j*r+K(n)-K(n-j),:);

end

v = reshape(v,p,n*c);
```

scale.m

Scales data matrix.

```
function Y = scale(X,a,b);

%SCALE        Scales input.
%
%             Y = SCALE(X,option);
%
%             If X is a matrix then column-wise scaling is performed
%             option defines the range within which to scale X
%             option = 0 : no scaling occurs (maybe necessary when used as a subroutine)
%             option = 1 : range of +/-0.9
%             option = 2 : range of 0.1-0.9
%             option = 3 : data is normalised to have mean of 0 and std. deviation of 1.
%
%             Alternatively the range can be defined by using:
%
%             Y = SCALE(X,LOW,HIGH);
%
%             Where the LOW and HIGH limits for the scaling are specified.

%             Written by N.T. Russell
%             Dept. Production Technology, Massey University
%             1995

if all(nargin ~= [2 3])
 error('Wrong number of input arguments.');
end

n=size(X,1);
mn=ones(n,1)*min(X);
mx=ones(n,1)*max(X);

if nargin==2
  option = a;

 if option==0, Y = X; break; end

 if option==1
                k1=1.8; k2=-0.9;  % Scales X into the range -0.9 to 0.9
 elseif option==2
                k1=0.8; k2=0.1;   % Scales X into the range 0.1 to 0.9
 end

 if option==3
  m = ones(n,1)*mean(X); s = ones(n,1)*std(X);
  Y = (X-m)./s;
  break
 end
end

if nargin==3
 if a < b,  k1 = b-a;   k2 = a;
 else
  k1 = a-b;  k2 = b;
 end
end

if mx==mn
  Y = (k1 + 2*k2)/2;
else
  Y = (k1*(X-mn)./(mx-mn))+k2;
end
```

wscale.m

Scales the weights of a trained neural network so that unscaled data can be presented to it.

```
function [w1,b1,w2,b2] = wscale(v,option,W1,B1,W2,B2,t);

%WSCALE  Scales weight matrices of a trained net so that it can be used with unscaled data.
%
%                [w1,b1] = wscale(v,option,W1,B1); - if just input layer
%
%                [w1,b1,w2,b2] = wscale(v,option,W1,B1,W2,B2,t);  - if both input and output layers
%
%               v is the unscaled input data set (no targets) and must  have dimensions: observations × input variables.
%               t is the unscaled set of targets, with dimensions: observations x target variables.
%               W and B are the trained weight matrices.
%               option defines the method of scaling the training data:
%                  option = 1: data was scaled to within the range +/-0.9
%                  option = 2: data was scaled to within the range 0.1-0.9
%                  option = 3: data was normalised (x-mean/std.dev.)

%               Written by N. T. Russell
%               Dept. Production Technology, Massey University
%               1995

if all([4 7] ~= nargin)
 error('Incorrect number of input arguments')
end

[q,n] = size(v);
h = size(W1,1);

if n ~= size(W1,2)
 error('Weight matrix dimensions does not match input matrix')
end

if h ~= size(B1,1)
 error('Weight and bias matrices are not compatible')
end

b1=zeros(size(B1));

if option==1
 k1=1.8; k2=-0.9;
end
if option==2
 k1=0.8; k2=0.1;
end

if option == 3

 m = mean(v);
 s = std(v);
 for i = 1:h
  for j = 1:n
   w1(i,j) = W1(i,j)/s(j);
   X(j) = w1(i,j)*m(j);
  end
  b1(i) = B1(i) - sum(X);
 end

else

 m = min(v);
 s = max(v)-m;
 for i = 1:h
  for j = 1:n
   w1(i,j) = k1*W1(i,j)/s(j);
   X(j) = w1(i,j)*m(j) - k2*W1(i,j);
  end
```

```
   b1(i) = B1(i) - sum(X);
  end

end

if nargin==7
  [q,n] = size(t);

  if n ~= size(W2,1)
    error('Weight matrix dimensions does not match target matrix')
  end

  if h ~= size(W2,2)
    error('Weight and bias matrices are not compatible')
  end

  if size(W2,1) ~= size(B2,1)
    error('Weight and bias matrices are not compatible')
  end

  b2=zeros(size(B2));

  if option==3
    m = mean(t);
    s = std(t);

    for i = 1:n
      for j = 1:h
        w2(i,j) = W2(i,j)*s(i);
      end
      b2(i) = B2(i)*s(i) + m(i);
    end

  else

    m = min(t);
    s = max(t)-min(t);

    for i = 1:n
      for j = 1:h
        w2(i,j) = W2(i,j)*s(i)/k1;
      end
      b2(i) = (B2(i)-k2)*s(i)/k1 + m(i);
    end
  end

end
```

trnlmtt.m

Trains a neural network with backpropagation through time using the Levenberg-Marquardt weight update algorithm. Developed for modelling the falling-film evaporator.

```
function [W1,B1,W2,B2,y,TR]=trnlmtt(Dy,V,S1,W1,B1,W2,B2,U,Utst,L);

%TRNLMTT  Train L-M neural network model for the falling-film evaporator.
%
%              2-layers = one hidden layer and one output layer.
%
%              [W1,B1,W2,B2,Y,TR] = TRNLMTT(Dy,V,S1,U,Utst,L);
%
%              Dy = [z1 z2 z3] where zi = delays on the past outputs in ascending order. The matrix V is the scaled pattern
%              matrix obtained from running the M-file NPATTERN (or similar routine) with the raw plant data. S1 is the
%              number of neurons in the hidden layer of the net.
%              If on-line testing is required U and Utst need to be given: Utst - unscaled testing data, U - unscaled training
%              data.  The routine outputs the weight, bias and output matrices.
%              L is an optional input which controls the number of times the training is repeated when wanting to train a
```

```
%               network a number of times. It also is used to label the weights that are saved during training
%
%               User-defined initial weights and biases (w1,b1,w2,b2) can also be included.
%
%               [W1,B1,W2,B2,Y] = TRNLMTT(Dy,V,S1,w1,b1,w2,b2,L);

%               Written by N.T. Russell
%               Dept. Production Technology
%               Massey University
%               1996

if all([3 5 6 7 9 10] ~= nargin),
  error('Wrong number of input arguments');
end

% Define scaling option
 option = 1;

if nargin==5,  U = W1;  Utst=B1; end
if nargin==6, U = W1;  Utst=B1;  L = W2; end
if all([6 10] ~= nargin), L=1; end

tst_sse = 10;
[n,m] = size(V);

% Training Parameters
 disp_freq = 20;
 max_epoch = 200;
 err_goal = 0.1;
 Jmin = 1e-20;
 mu = 0.001;
 mu_inc = 10;
 mu_dec = 0.1;
 mumax = 1e12;

% Define the input and associated target vectors
s2 = 1; %number of output neurons
P = V(:,1:m-s2)';
T = V(:,m-s2+1:m)';

% Define thresholding functions
f1 = 'logsig';
f2 = 'purelin';

% Set input vector size R, output layer size S2 and batch size N.
 [R,N] = size(P);
 [S2,N] = size(T);

%**** Start Loop *****

for count = L
fprintf('Training run %1.0f...\n',count);

% Initialise weights and biases

if sum([3 5 6] == nargin)
 Pmin = min(P')';
 Pmax = max(P')';
 [W1o,B1o] = nwlog(S1,[Pmin Pmax]);
 [W2o,B2o] = rands(S2,S1);

% add zeros to W1 (for creating locally-connected weight layer)
  if S1>= 7
                    if S1==7,x = 2;
                    elseif S1==9, x = 3;
                    elseif S1==11, x = 4; end;
                    W1o = zerow(W1o,x);
  end
end;

if nargin==7 | nargin>=9
 W1o = W1; W2o = W2;
```

```
B1o = B1; B2o = B2;
end;

% Network training

TP = [disp_freq max_epoch err_goal Jmin mu mu_inc mu_dec mumax count];

if nargin==3 | nargin==7
 [W1,B1,W2,B2,epoch,TR] = tlmtt(Dy,W1o,B1o,f1,W2o,B2o,f2,P,T,TP);
else
 [W1,B1,W2,B2,epoch,TR] = tlmtt(Dy,W1o,B1o,f1,W2o,B2o,f2,P,T,TP,U,Utst);
end
clg

figure(1)
semilogy(0:epoch,TR),xlabel('epochs'),
ylabel('sum squared error'),title('epochs vs sum squared error');
hold on; plot([0 epoch],[err_goal err_goal],'r:')

if exist('U')
 [w1,b1,w2,b2] = wscale(U(:,1:m-s2),option,W1,B1,W2,B2,U(:,m-s2+1:m));
 y = simurec(U(:,1:m-s2)',w1,b1,f1,w2,b2,f2,Dy);
 sse = sumsqr(U(:,m-s2+1:m)'-y);
 F = 1/(max(U(:,m-s2+1:m))-min(U(:,m-s2+1:m)))^2;
 mse = sse*F/(n-1);

% Save final weights for each training run
 ww=sprintf('ww%1.0f',count);
 feval('save',ww,'W1','W2','B1','B2');

 if length(L)==1 % suppress plotting when using a loop
  figure(2)
  clg
  x=[1:N];
  plot(x,U(:,m-s2+1:m),'r',x,y,'y')
  title(['SSE = ',num2str(sse),' and MSE = ',num2str(mse)])
  xlabel('Actual = red Predicted = yellow')
 end

else

 y = simurec(V(:,1:m-s2)',W1,B1,f1,W2,B2,f2,Dy);
 sse=sumsqr(V(:,m-s2+1:m)'-y); mse = sse/(n-1);
 figure(2)
 clg
 x=[1:N];
 plot(x,V(:,m-s2+1:m),'r',x,y,'y')
 title(['SSE = ',num2str(sse),' and MSE = ',num2str(mse)])
 xlabel('Actual = red Predicted = yellow')

 end
end %**** End Loop *****
```

tlmtt.m

LMTT training routine called by *trnlmtt.m.* Based on a training file from the *Neural Network Toolbox,* Mathworks Inc.

learmlm.m and *nncpyi.m* are functions from the *Neural Network Toolbox.*

```
function [w1,b1,w2,b2,i,tr] = tlmtt(Dy,w1,b1,f1,w2,b2,f2,p,t,tp,U,Utst)

%TLMTT Recurrent (n-step-ahead) training of a network.
%
%[W1,B1,W2,B2,TE,TR] = tlmtt(Dy,W1,B1,'F1',W2,B2,'F2',P,T,TP,U,Utst)
%             Dy  - [z1 z2 z3] zi = sample delays on outputs in ascending order
%             Wi - Weight matrix of ith layer.
%             Bi - Bias vector of ith layer.
```

```
% F - Transfer function (string) of ith layer.
% P - RxQ matrix of input vectors.
% T - S2 xQ matrix of target vectors.
% TP - Training parameters (optional).
% Returns:
% Wi - new weights.
% Bi - new biases.
% TE - the actual number of epochs trained.
% TR - training record: [row of errors]

% Training parameters are:
% TP(1) - Epochs between updating display, default = 25.
% TP(2) - Maximum number of epochs to train, default = 1000.
% TP(3) - Sum-squared error goal, default = 0.02.
% TP(4) - Minimum gradient, default = 1e-6.
% TP(5) - Initial value for MU, default = 0.001.
% TP(6) - Multiplier for increasing MU, default = 10.
% TP(7) - Multiplier for decreasing MU, default = 0.1.
% TP(8) - Maximum value for MU, default = 1e10.
% TP(9) - Loop counter for repeated training, default = 1.
% Missing parameters and NaN's are replaced with defaults.

% Based on Mathworks file TLM2.m,
% Altered to include:
% - LMTT training,
% - on-line testing, and
% - handling locally connected weight layers.
% Altered by N.T Russell
% Dept. Production Technology, Massey University
% 1995

if nargin < 9,error('Not enough input arguments.'),end

% TRAINING PARAMETERS
if nargin == 9, tp = []; end
tp = nndef(tp,[25 1000 0.02 1e-6 0.001 10 0.1 1e10 1]);
df= tp(1);
me = tp(2);
eg = tp(3);
grad_min = tp(4);
mu_init = tp(5);
mu_inc = tp(6);
mu_dec = tp(7);
mu_max = tp(8);
kk = tp(9); % counter
df1 = feval(f1,'delta');
df2 = feval(f2,'delta');

% DEFINE SIZES
q = size(p,2);
[s1,r] = size(w1);
[s2,s1] = size(w2);

%find number of weights in w1 (necessary if layer is locally connected)
v = find(w1);
Nw1 = length(v);

w1_ind = [1:Nw1];
b1_ind = [1:s1] + w1_ind(length(w1_ind));
w2_ind = [1:(s1*s2)] + b1_ind(length(b1_ind));
b2_ind = [1:s2] + w2_ind(length(w2_ind));
ii = eye(b2_ind(length(b2_ind)));
dw1 = w1; db1 = b1;
dw2 = w2; db2 = b2;

% INITIAL PRESENTATION PHASE

[a1,a2,px] = simurec(p,w1,b1,f1,w2,b2,f2,Dy);
px=p;

[R,N]=size(p);
Ky = length(Dy);
```

```
s2 = size(w2,1);

opt_sse=1e6*ones(s2,1); count = 0; tst_sse=0;

e=t-a2;
SSE=sumsqr(e);

if nargin < 12
 mxu=1; mnu=0; mxt=1; mnt=0;
end

% Perform on-line test
if nargin == 12
 m = R+s2;
 mnu=min(U(:,m-s2+1:m)); mxu=max(U(:,m-s2+1:m));
 mnt=min(t'); mxt=max(t');
 SSE = SSE*(mxu-mnu).^2/(mxt-mnt).^2;
 option = 1;
 [W1,B1,W2,B2] = wscale(U(:,1:m-s2),option,w1,b1,w2,b2,U(:,m-s2+1:m));
 [Ytst,tst_sse] = oltest(Utst,W1,B1,f1,W2,B2,f2,Dy);
 s = [SSE tst_sse];
end

% TRAINING RECORD
tr = zeros(1,me+1);
tr(1) = SSE;

% PLOTTING FLAG
plottype = (r==1) & (s2==1);

% PLOTTING
clg
message = sprintf('TRNLMTT: %%g/%g, mu = %%g, SSE = %%g, Test_SSE = %%g\n',me);
fprintf(message,0,mu_init,SSE,tst_sse)
if plottype
  h = plotfa(p,t,p,a2);
else
  h = ploterr(tr(1),eg);
end

mu = mu_init;

% ---------- Start training loop ----------
for i=1:me

  % CHECK PHASE
  if SSE < eg, i=i-1; break, end

  % FIND JACOBIAN

  ext_p = nncpyi(px,s2);
  ext_a1 = nncpyi(a1,s2);

% Call backpropagation-through-time routine
  [ext_d1,ext_d2] = lmtt(df1,df2,a1,a2,w1,w2,Dy);

  j1 = learnlm(ext_p,ext_d1);

  % If w1 is locally connected the
  % zero links (columns) are removed from j1
  j1 = j1(:,v);

  j2 = learnlm(ext_a1,ext_d2);
  j = [j1, ext_d1', j2, ext_d2'];

  % CHECK MAGNITUDE OF GRADIENT
  je = j' * e(:);
    grad = norm(je);
    if grad < grad_min, i=i-1; break, end

  % INNER LOOP, INCREASE MU UNTIL THE ERRORS ARE REDUCED
  jj = j'*j;
```

```
  while (mu <= mu_max)

    dx = -(jj+ii*mu) \ je;

    % reconstruct dw1 matrix
    dw1 = zeros(s1,r);

    dw1(v) = dx(w1_ind); db1 = dx(b1_ind);
    dw2(:) = dx(w2_ind); db2 = dx(b2_ind);
    new_w1 = w1 + dw1; new_b1 = b1 + db1;
    new_w2 = w2 + dw2; new_b2 = b2 + db2;

  % EVALUATE NEW NETWORK

    [a1,a2,new_p] = simurec(px,new_w1,new_b1,f1,new_w2,new_b2,f2,Dy);
    new_e=t-a2;
    new_SSE = sumsqr(t-a2);
    new_SSE = new_SSE*(mxu-mnu).^2/(mxt-mnt).^2;

    if (new_SSE < SSE), break, end
    mu = mu * mu_inc;
  end
  if (mu > mu_max), i = i-1; break, end

  mu = mu * mu_dec;

  % UPDATE NETWORK
  w1 = new_w1; b1 = new_b1;
  w2 = new_w2; b2 = new_b2;
  e = new_e; SSE = new_SSE;

% Online testing - long range test
if nargin == 12
  [W1,B1,W2,B2] = wscale(U(:,1:m-s2),option,w1,b1,w2,b2,U(:,m-s2+1:m));
  [Ytst,tst_sse] = oltest(Utst,W1,B1,f1,W2,B2,f2,Dy);
  count = count+1;
  s(count+1,1:2) = [SSE tst_sse];
  sse = sprintf('sse%1.0f',kk); % save training and testing errors
  feval('save',sse,'s');

  if tst_sse < opt_sse
     W1=w1; W2=w2; B1=b1; B2=b2;
     minw = sprintf('minw%1.0f',kk);
     feval('save',minw,'W1','W2','B1','B2','SSE','tst_sse');  % save optimal weights
     opt_sse = tst_sse;
  end
end

  % TRAINING RECORD
  tr(i+1) = SSE;

  % PLOTTING
  if rem(i,df) == 0
    fprintf(message,i,mu,SSE,tst_sse)
    if plottype
      delete(h); h = plot(p,a2,'m'); drawnow;
    else
      h = ploterr(tr(1:(i+1)),eg,h);
    end
  end
end

% TRAINING RECORD
tr = tr(1:(i+1));

% PLOTTING
if rem(i,df) ~= 0
  fprintf(message,i,mu,SSE,tst_sse)
  if plottype
    delete(h);
    plot(p,a2,'m');
```

```
  drawnow;
 else
  ploterr(tr,eg,h);
 end
end

% WARNINGS
if SSE > eg
 disp(' ')
 if (mu > mu_max)
  disp('TRAINLMTT: Error gradient is too small to continue learning.')
 else
  disp('TRAINLMTT: Network error did not reach the error goal.')
 end
 disp(' ')
end

ep = find(s(:,2)==opt_sse)-1;
fprintf('Best test SSE = %6.4f at epoch %1g\n\n',opt_sse,ep);
```

lmtt.m

Performs the backpropagation through time calculation for two-layer , externally recurrent neural network using the Levenberg-Marquardt optimisation method.

```
function [ext_d1,ext_d2] = lmtt(df1,df2,a1,a2,w1,w2,Dy);

%LMTT        Performs Backpropagation-Through-Time calculation for 2-layer, externally recurrent neural network using the
%            Levenberg-Marquardt optimisation method.
%
%            [ext_d1,ext_d2] = lmtt(df1,df2,a1,a2,w1,w2,Dy);
%
%            The prediction errors are not only propagated back through each layer but also through each timestep from t=T
%            to t=1.
%
%            Inputs:  a1          - Outputs from hidden layer
%                     a2          - Output from output layer
%                     w1, w2      - current network weight matrices
%                     Dy          - vector of 3 delays for outputs fedback to the input layer
%                                   ie. Dy = [z1 z2 z3];
%                     df1,df2     - define the derivative of activation functions used for each layer

%            Outputs: ext_d1, ext_d2
%                                 - extended delta matrices for the hidden and output
%                                 layers which are used for updating the weights.
%
%            NOTE: Written for n output neurons (n = S2)

%            Written by N.T. Russell
%            Dept. Production Technology
%            Massey University
%            1997

% Time horizon = columns of input data
T = size(a2,2);
% r = number of input neurons
r = size(w1,2);

n = size(w2,1); % n is the number of output neurons
Ky = length(Dy);
ext_a1 = nncpyi(a1,n);

% Create d0 matrix (deltas for input layer)
d0 = zeros(Ky*n,T+max(Dy));

% Note that for t > T, d0 = 0;

for t = T:-1:1  % repeat for time, t=T to t=1
```

```
 sd0 = zeros(n,1);

% Calc. sum of input layer deltas, according to delays
 for i = 1:n
  for z = 1:Ky
   s = d0(z+(i-1)*Ky,t+Dy(z));
   sd0(i) = s + sd0(i);
  end
 end

% Calculate output and hidden layer deltas at time t
 d2 = feval(df2,a2(:,t)) + sd0;
 d1 = feval(df1,a1(:,t),d2,w2);

% Form the extended matrices for the deltas
 x = n*(t-1)+1:n*t;
 ext_d2(:,x) = -nncpyd(d2);
 ext_d1(:,x) = feval(df1,ext_a1(:,x),ext_d2(:,x),w2);

% Calculate input layer deltas for past outputs only
 D0 = w1'*d1;
 d0(:,t) = D0(r-n*Ky+1:r);        % deltas for the last n*Ky input neurons which are linked to the
                                  % previous output values

end              % end of for loop
```

simurec.m

Simulation of a two-layered externally-recurrent network. Performs long-range prediction.

```
function [a1,a2,P_new] = simurec(P,W1,B1,f1,W2,B2,f2,Dy)

% SIMUREC    Simulation of a two-layered externally-recurrent network.
%            Performs long-range prediction.
%
%            [a1,a2,P_new] = simurec(P,W1,B1,f1,W2,B2,f2,Dy);
%
% or         a2 = simurec(P,W1,B1,f1,W2,B2,f2,Dy);
%
%            P - input data (column vectors)
%            W, B & f are the network weights and activation functions
%            Dy = [z1 z2 z3] where zi = sample delays on the outputs in ascending order
% Returns:
%            a1, a2 - network predictions for first and second layer
%            P_new = updated input matrix

%            Written by N.T. Russell
%            Dept. Production Technology
%            Massey University
%            1997

if nargin ~= 8
  error('Wrong number of input arguments')
end

[R,C] = size(P);
n = size(W2,1);
nd = length(Dy);

z1 = Dy(1);

if nd > 1, z2 = Dy(2); end
if nd == 3,  z3 = Dy(3); end

for c=1:C  % Present patterns to network one by one
```

```
  a1(:,c) = feval(f1,W1*P(:,c),B1);
  a2(:,c) = feval(f2,W2*a1(:,c),B2);
  A = a2;

% Update input matrix with network's output
  for i = 1:n
    r = nd*(n-i);
    if c < C-z1+1
      P(R-r-nd+1,c+z1) = A(i,c);
      if c < C-z2+1 & nd > 1
        P(R-r-nd+2,c+z2) = A(i,c);
        if c < C-z3+1 & nd > 2
          P(R-r,c+z3) = A(i,c);
        end
      end
    end
  end % end of for loop

end % end of for loop


if nargout <= 1
  a1 = A;
end
```

testrec.m

Simulation of a two-layered externally-recurrent network. Performs long-range prediction, plots the results and calculates the prediction error.

```
function [Y,MSE] = testrec(Data,W1,B1,W2,B2,Dy);

%TESTREC     Tests a recurrent Neural Network model for the falling-film evaporator with testing data.
%
%            Y = TESTREC(Data,w1,b1,w2,b2,Dy);
%
%            "Data" is the testing data to be presented to the net.
%            This data has first to be scaled and structured using the m-file NPATTERN or similar routine.  Wi and Bi are the
%            weights and biases of the net.
%            The function gives the net's output estimates (Y), calculates the error and plots the predicted output.
%            Dy = [z1 z2 z3] where zi = sample delays on the outputs in ascending order
%            Can also output the MSE for the prediction which suppresses the plotting command using -
%
%            [Y,MSE] = TESTREC(Data,w1,b1,w2,b2,Dy);

%            Written by N.T. Russell
%            Dept. Production technology
%            Massey University
%            1996

if all([6] ~= nargin),
  error('Wrong number of input arguments');
end

[n,m]=size(Data);

f=size(W2,1);    % f is the number of target variables

% Define the input and associated target vectors
P=Data(:,1:m-f)';
T=Data(:,m-f+1:m);

% Define thresholding functions
f1='logsig';
f2='purelin';

a2=simurec(P,W1,B1,f1,W2,B2,f2,Dy);
Y=a2';
```

```
% Error analysis
[x,y] = size(Y);

err = T-Y;
SSE = sum(err.^2);
F=1./(max(T)-min(T)).^2;
MSE = SSE.*F/(x-1);
ME=sum(abs(err))/(x*y); PE=ME/abs(mean(T))*100;

if nargout == 1

 if f == 1
  fprintf('SSE = %6.5g\nnorm-MSE = %6.2fe-3\n',SSE,MSE*1e3);
  fprintf('Mean Error = %6.5g and Percent Error = %6.5g\n',ME,PE);

% Plot commands - plotting test estimates vs actual
  disp('Hit any key to show test prediction plot.')
  pause
  clg
  plot([1:n],T,'r')
  hold on; plot([1:n],Y,'y')
  title(['Testing Network: SSE = ',num2str(SSE),' and MSE = ',num2str(MSE)],'FontSize',10)
  xlabel('Actual = red   Predicted = yellow','FontSize',10)
  figure(gcf)
 end

 if f > 1
  for i = 1:f
  fprintf('SSE = %6.5g\nnorm-MSE = %6.2fe-3\n',SSE(i),MSE(i)*1e3);

% Plot commands - plotting test estimates vs actual
  fprintf('Hit any key to show test prediction plot #%1.0g\n',i)
  pause
  clg
  plot([1:n],T(:,i),'r')
  hold on; plot([1:n],Y(:,i),'y')
  title(['Output ',num2str(i),': SSE = ',num2str(SSE(i)),' and MSE = ',num2str(MSE(i))],'FontSize',10)
  xlabel('Actual = red   Predicted = yellow','FontSize',10)
  figure(gcf)

  end % for loop
 end % if f > 1
end % if nargout==1
```

oltest.m

On-line testing of network during training. Performs long-range prediction of testing data.

```
function [Y,sse] = oltest(V,W1,B1,f1,W2,B2,f2,Dy)

% OLTEST      On-line testing of network during training routine.  Performs long-range prediction using testing data.
%
%             [Y,sse] = oltest(V,W1,B1,f1,W2,B2,f2,Dy);
%
%             V - testing data
%             W, B & f are the network weights and activation functions
%             Dy - [z1 z2 z3] where zi = sample delays in ascending order
% Returns:
%             Y - network predictions
%             sse - SSE of the predictions

%             Written by N.T. Russell
%             Dept. Production Technology
%             Massey University, 1996

[R,C] = size(V);
```

```
f = size(W2,1); % f = number of output neurons

P = V(:,1:C-f);    % Input matrix
T = V(:,C-f+1:C);          % Target matrix

Y = simurec(P',W1,B1,f1,W2,B2,f2,Dy);

Y=Y';
e = T-Y;
sse = sumsqr(e);
```

zerow.m

Initialises an input-layer weight matrix to have a locally-connected structure according to rules outlined in Chapter 6.

```
function W1 = zerow(w1,n);

% Initialises a locally-connected structure for input-layer weight matrix.
% w1 is the fully connected weight matrix,
% n is the is the number of input variables for network (2, 3 or 4)
% W1 is the resulting locally-connected weight matrix

% Written by N.T. Russell
% Dept. Production Technology, Massey University
% 1996

W1=w1;

% For 2 inputs
if n == 2
  W1(1:2,4:6) = zeros(2,3); W1(3:4,1:3) = zeros(2,3);
  W1(5:7,1:3) = diag(diag(W1(5:7,1:3)));
  W1(5:7,4:6) = diag(diag(W1(5:7,1:3)));
end

% For 3 inputs:
if n == 3

% Each variable type has two hidden nodes; Times have 1 hn. Tot=9.
  W1(3:6,1:3) = zeros(4,3);
  W1(7:9,1:3) = diag(diag(W1(7:9,1:3)));

  W1(1:2,4:6) = zeros(2,3); W1(5:6,4:6) = zeros(2,3);
  W1(7:9,4:6) = diag(diag(W1(7:9,4:6)));

  W1(1:4,7:9) = zeros(4,3);
  W1(7:9,7:9) = diag(diag(W1(7:9,7:9)));
end

% For 4 inputs:
if n == 4
% Each variable type has two hidden nodes; Times have 1 hn. Tot=11.

  W1(3:8,1:3) = zeros(6,3);
  W1(9:11,1:3) = diag(diag(W1(9:11,1:3)));

  W1(1:2,4:6) = zeros(2,3); W1(5:8,4:6) = zeros(4,3);
  W1(9:11,4:6) = diag(diag(W1(9:11,4:6)));

  W1(1:4,7:9) = zeros(4,3); W1(7:8,7:9) = zeros(2,3);
  W1(9:11,7:9 ) = diag(diag(W1(9:11,7:9)));

  W1(1:6,10:12) = zeros(6,3);
  W1(9:11,10:12) = diag(diag(W1(9:11,10:12)));
end
```

expfilt.m

Exponential filtering function.

```
function y=expfilt(x,a)

% EXPFILT  y=expfilt(x,a)
%
%               Performs exponential filtering on vector x.
%               a is the weighting on the current input value,
%               ie. (1-a) is the weighting on past output value, where: 0 <= a <= 1

%               Written by N.T. Russell
%               Dept. Production Technology
%               1995

[r,c]=size(x);

if c~=1
 error('Input should be a column vector')
end

y(1)=x(1);

for R=2:r,
 y(R)=(1-a)*y(R-1) + a*x(R);
end

y=y';
```

Appendix 9

Neural Network Model Results

Sub-network structures

Table A9-1 below summarises the sub-network structure that were finally settled upon for the modular neural model of the evaporator. D_u is the delay vector for the input variables, D_y is the delay vector for the recurrent outputs, S_i is the number of inputs nodes, S_h is the number of hidden layer neurons, S_o is the number of output neurons and N_w is the total number of parameters (weights and biases) of the network. Note that Q_i^f represents a filtered flowrate.

Table A9-1: Summary of sub-network structures for the evaporator neural model

Output	Inputs	D_u	D_y	S_i-S_h-S_o	N_w
T_s	$[T_{sp}\ T_1\ Q_0]$	[1 2 3]	[1 2 3]	12-11*-1	59
T_{ph1}	$[T_{ph1}\ Q_0]$	[1 2 4]	[1 2 4]	9-9*-1	46
T_{ph2}	$[T_s\ T_2\ Q_0]$	[1 5 7]	[1 5 7]	12-11*-1	59
T_1	$[T_s\ T_2\ Q_0]$	[1 5 7]	[1 5 7]	12-11*-1	59
T_2	$[T_1\ T_3\ Q_1]$	[1 2 4]	[1 2 4]	12-11*-1	59
T_3	$[T_2\ T_{ph1}\ Q_3]$	[1 2 4]	[1 2 4]	12-11*-1	59
L_1	$[Q_0^f\ Q_1^f\ T_1]$	[1 2 3]	[1 2 3]	12-11*-1	59
L_2	$[Q_1^f\ Q_2^f\ T_2]$	[1 2 3]	[1 2 3]	12-11*-1	59
L_3	$[Q_2^f\ Q_3^f\ T_3]$	[1 2 3]	[1 2 3]	12-11*-1	59
Q_0	$[T_1\ N_0]$	0	1	3-6-1	31
Q_1	$[T_1 - T_2\ L_1\ N_1]$	0	1	4-5-1	31
Q_2	$[T_2 - T_3\ L_2\ N_2]$	0	1	4-3-1	19
Q_3	$[T_3\ L_3\ N_3]$	0	1	4-5-1	31
* Locally connected layer.			*Totals:*	120-116-13	630

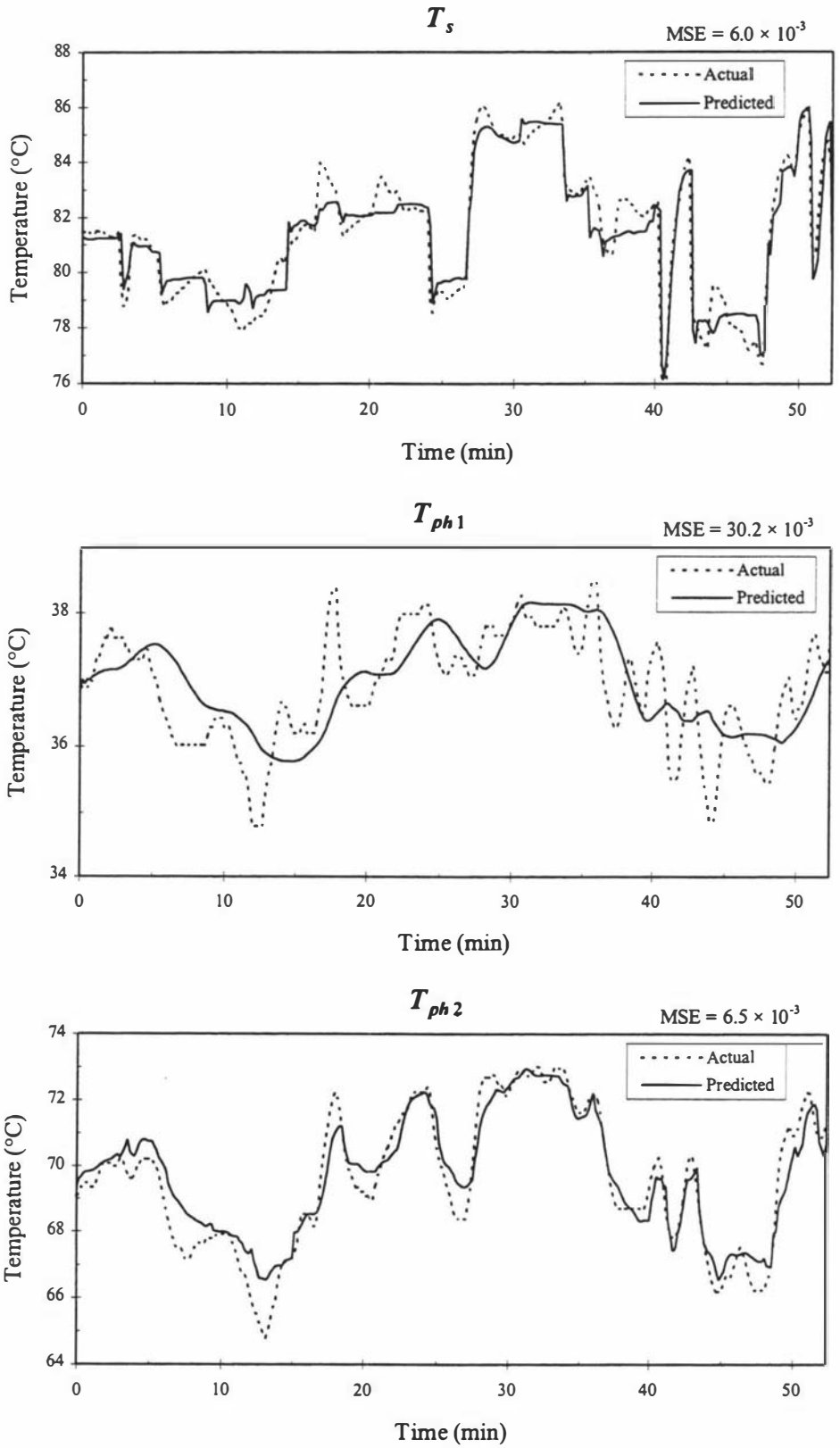
Sub-network training and testing

The sub-network with the best prediction errors for the testing data were selected for use within the full evaporator model. The results listed in Table A9-2 are for the networks which produced the best testing errors during training.

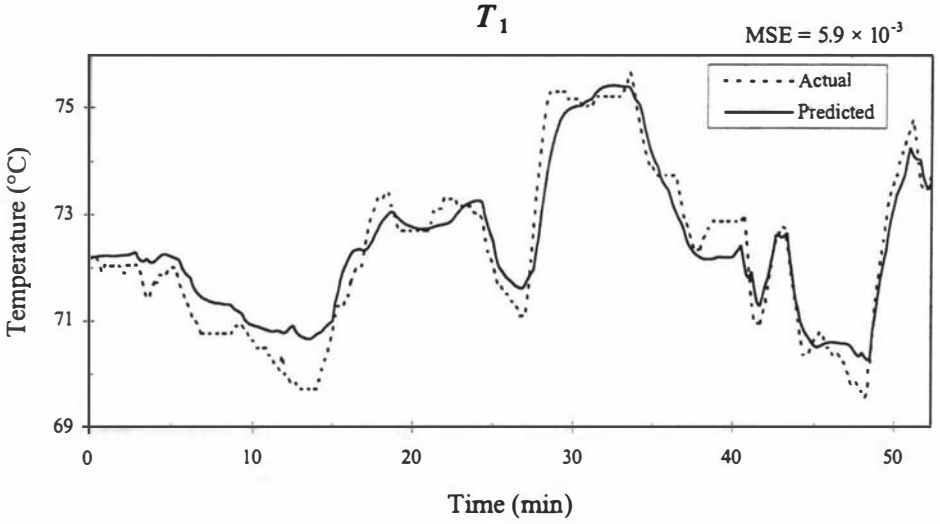
Table A9-2: Long-range prediction errors for the individual sub-nets

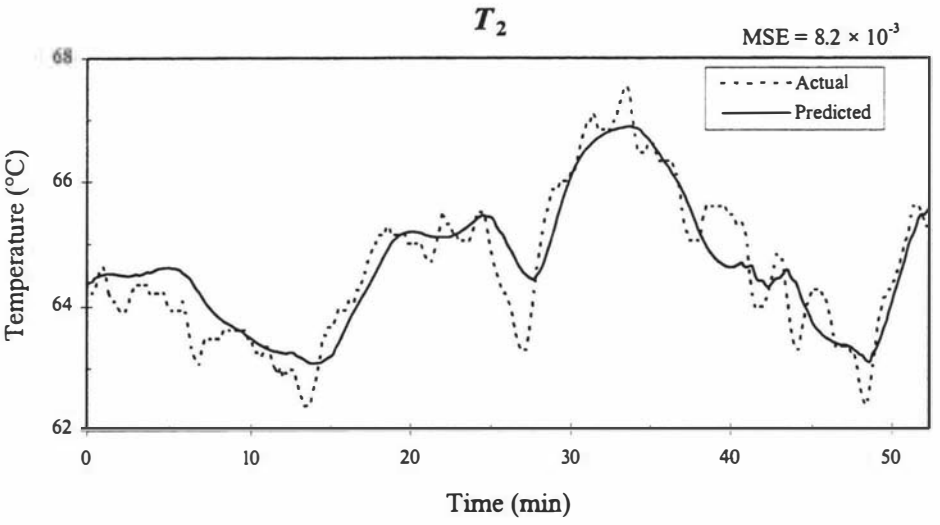
	Mean-Squared Error ($\times 10^{-3}$)	
Sub-network	Training Data	Testing Data
T_s	2.6	3.7
T_{ph1}	2.0	1.8
T_{ph2}	0.5	1.2
T_1	0.6	1.6
T_2	1.5	3.8
T_3	1.4	3.9
L_1	9.9	14.1
L_2	9.7	28.4
L_3	9.3	12.7
Q_0	3.5	5.8
Q_1	4.9	11.7
Q_2	1.8	4.7
Q_3	7.3	7.7

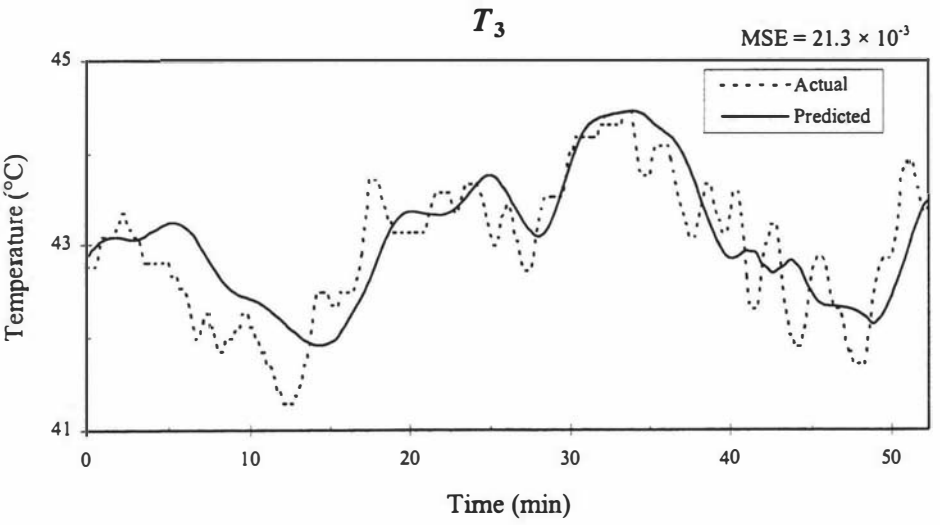
Full neural network model plots – Long-range predictions for validation 2 data



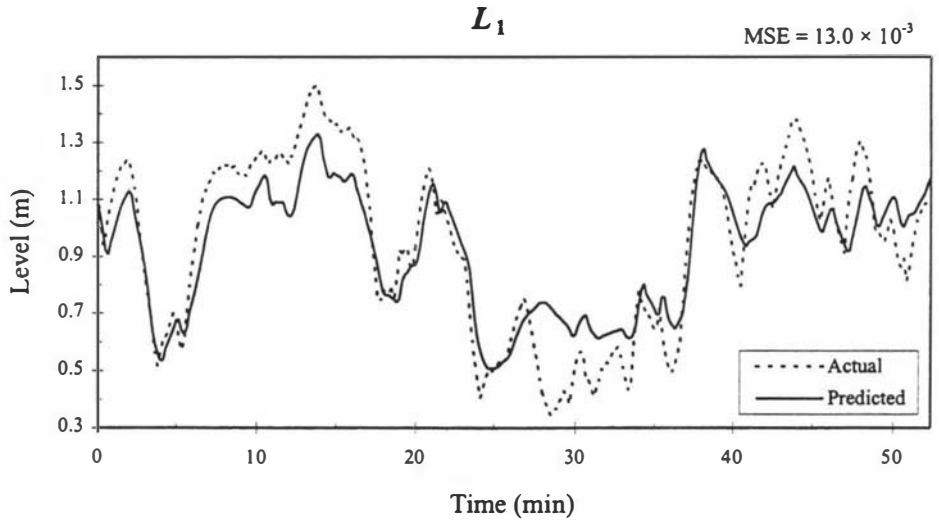
Full neural network model plots – Long-range predictions for validation 2 data

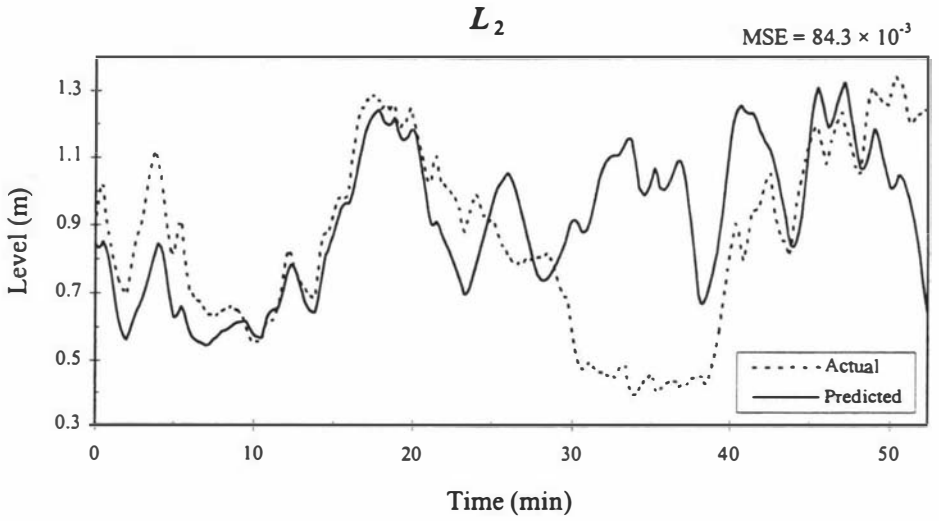


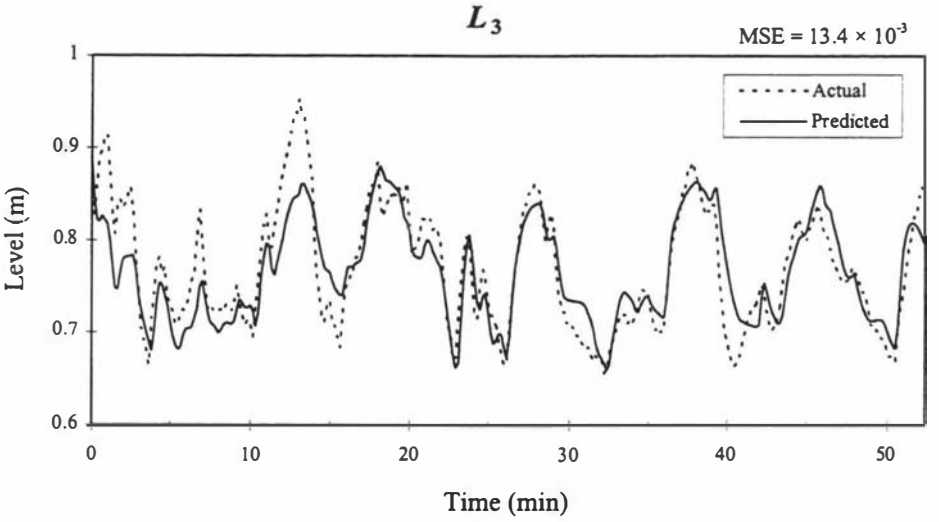




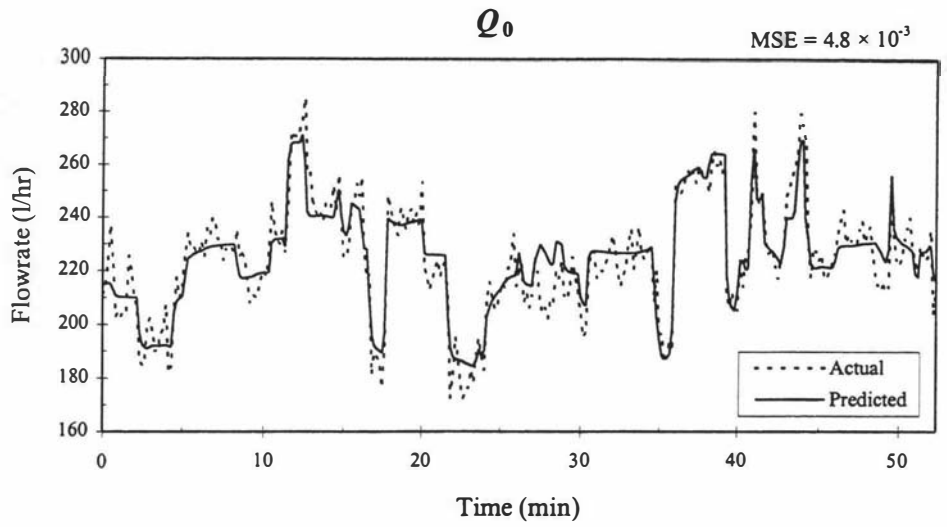
Full neural network model plots – Long-range predictions for validation 2 data

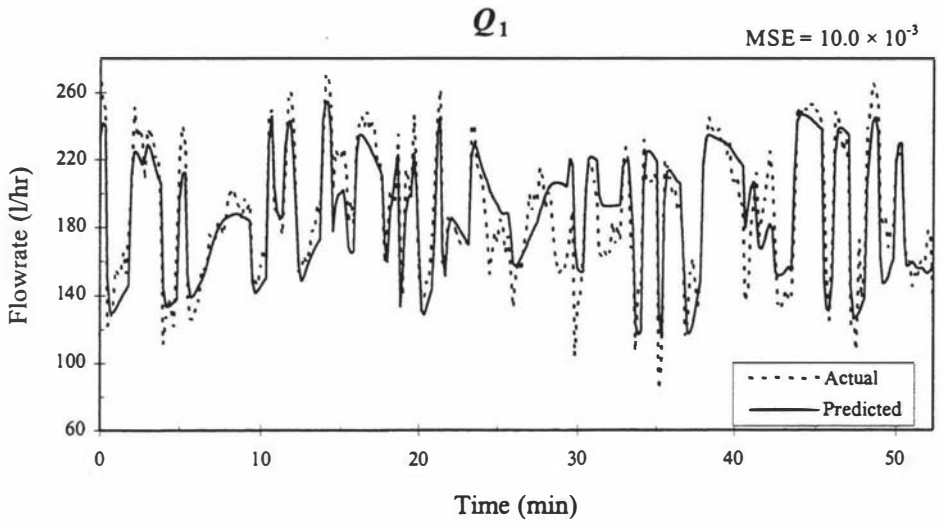


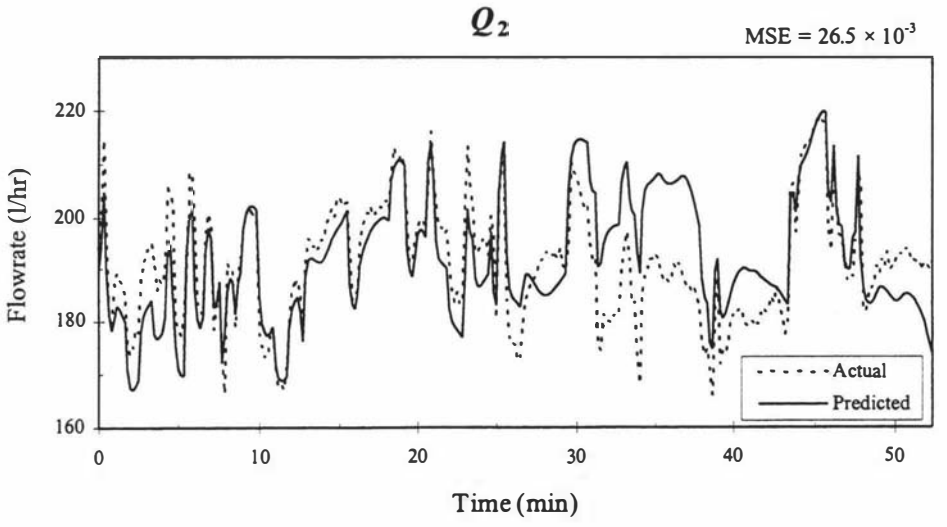




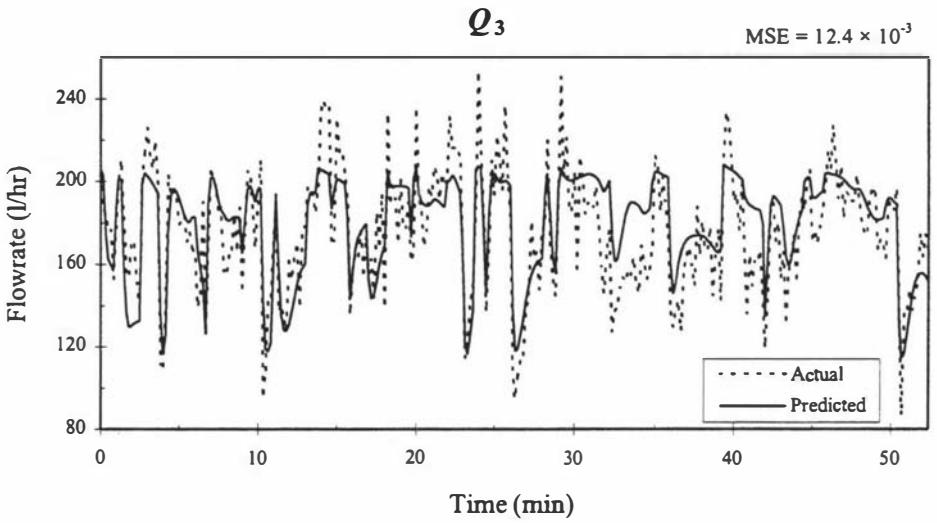
Full neural network model plots – Long-range predictions for validation 2 data







Full neural network model plots – Long-range predictions for validation 2 data



Appendix 10

Linear ARX Model Results

Sub-model structures

Table A10-1: Summary of sub-model structures for the evaporator ARX model

Model Output	Model inputs	Order of A n_a	Order of B n_b	Input - output delay, n_k	N_w
T_s	$[T_{sp}\ T_1\ Q_0]$	4	[2 5 3]	1	14
T_{ph1}	$[T_{ph1}\ Q_0]$	3	[4 1]	1	8
T_{ph2}	$[T_s\ T_2\ Q_0]$	5	[3 1 2]	1	11
T_1	$[T_s\ T_2\ Q_0]$	3	[2 1 5]	1	11
T_2	$[T_1\ T_3\ Q_1]$	2	[4 2 6]	1	14
T_3	$[T_2\ T_{ph1}\ Q_3]$	2	[6 2 1]	1	11
L_1	$[Q_0^f\ Q_1^f\ T_1]$	1	[2 3 5]	1	11
L_2	$[Q_1^f\ Q_2^f\ T_2]$	2	[3 1 3]	1	9
L_3	$[Q_2^f\ Q_3^f\ T_3]$	2	[2 2 1]	1	7
Q_0	$[T_1\ N_0]$	2	[2 4]	0	8
Q_1	$[T_1 - T_2\ L_1\ N_1]$	1	[2 3 2]	0	8
Q_2	$[T_2 - T_3\ L_2\ N_2]$	1	[1 3 3]	0	8
Q_3	$[T_3\ L_3\ N_3]$	4	[2 1 5]	0	12
				Total:	132

Individual Sub-model results - These results are given here so a directly comparison can be drawn with the individual sub-network results in listed Appendix 9.

Table A10-2: Long-range prediction errors for the individual ARX sub-models

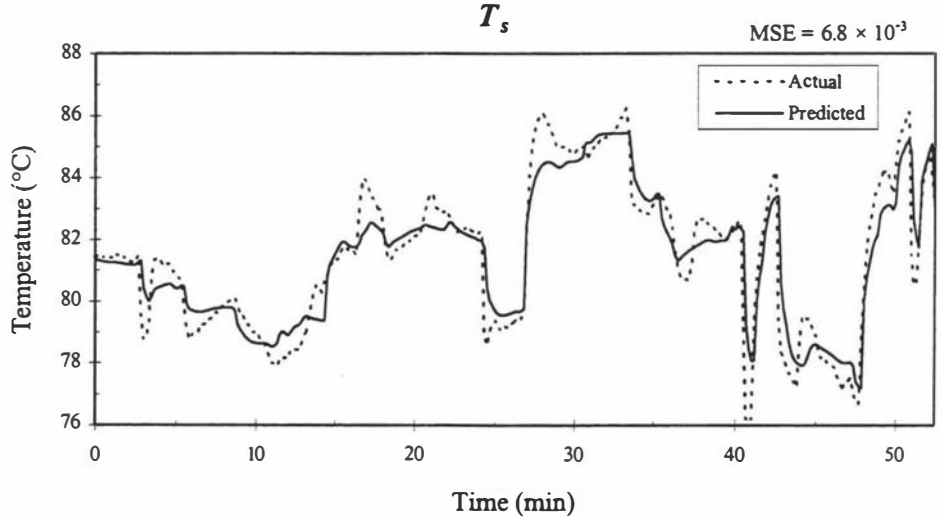
	Mean-Squared Error ($\times 10^{-3}$)	
Sub-model	Training Data	Testing Data
T_s	4.0	5.3
T_{ph1}	2.0	2.9
T_{ph2}	0.6	1.2
T_1	0.7	1.5
T_2	1.3	5.5
T_3	1.7	3.2
L_1	9.4	33.8
L_2	15.4	62.8
L_3	12.5	20.4
Q_0	3.0	5.2
Q_1	3.1	8.2
Q_2	1.7	5.6
Q_3	5.9	7.8

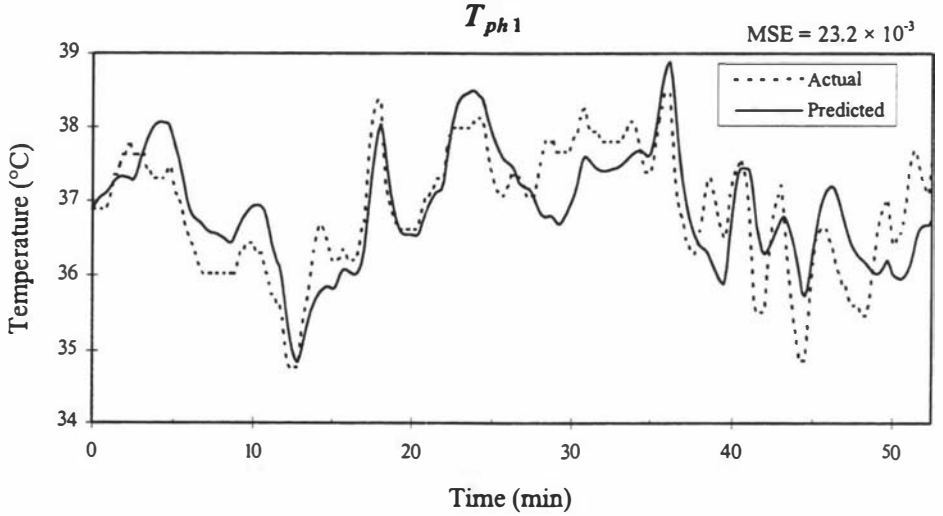
Full evaporator model results

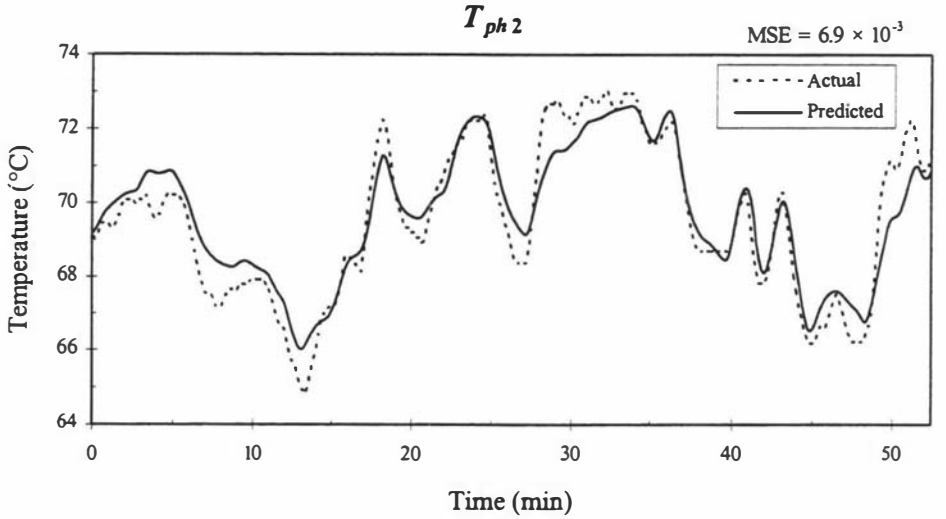
Table A10-3: Long-range prediction errors for the evaporator ARX model

	Mean-Squared Error ($\times 10^{-3}$)		
Output	Training Data	Validation 1 Data	Validation 2 Data
T_s	4.2	6.9	6.8
T_{ph1}	10.6	21.4	23.2
T_{ph2}	2.3	5.3	6.9
T_1	2.5	6.3	7.6
T_2	3.2	12.4	8.2
T_3	10.0	26.8	24.1
L_1	7.5	15.3	11.9
L_2	16.0	88.9	58.1
L_3	12.1	32.2	19.1
Q_0	3.2	4.6	4.9
Q_1	5.2	10.1	9.7
Q_2	5.3	20.7	17.2
Q_3	5.7	9.5	9.6

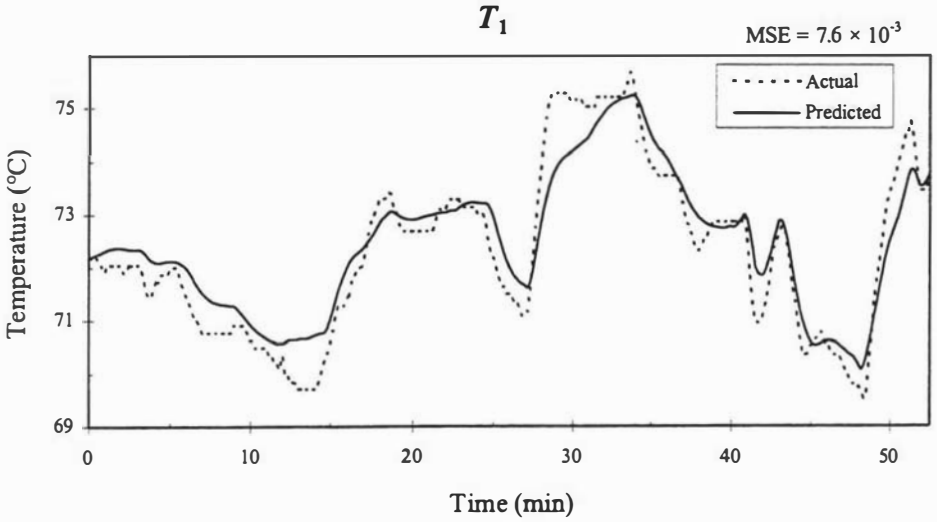
Full ARX model plots – Long-range predictions for validation 2 data

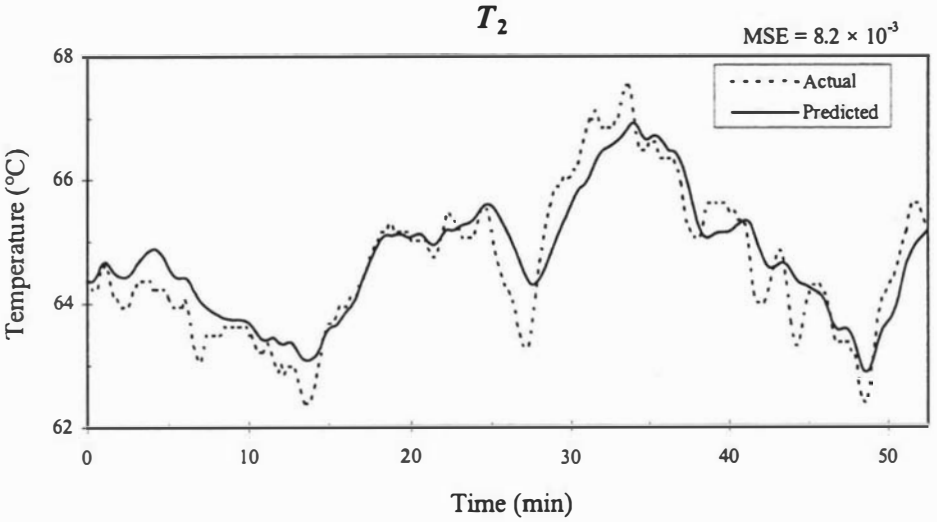


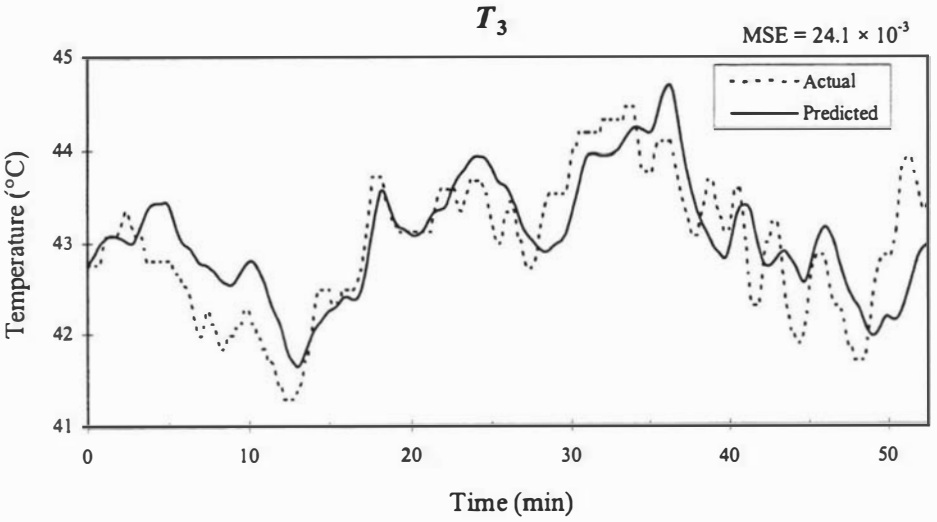




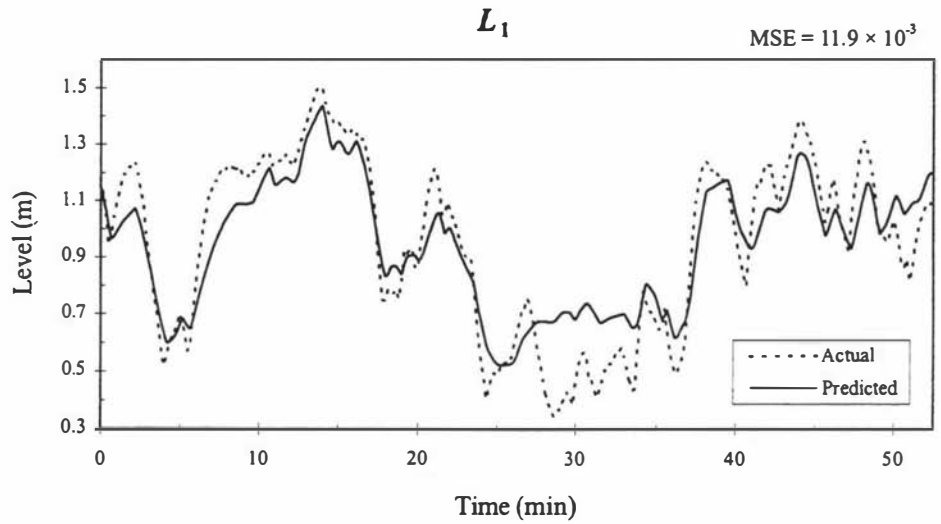
Full ARX model plots – Long-range predictions for validation 2 data

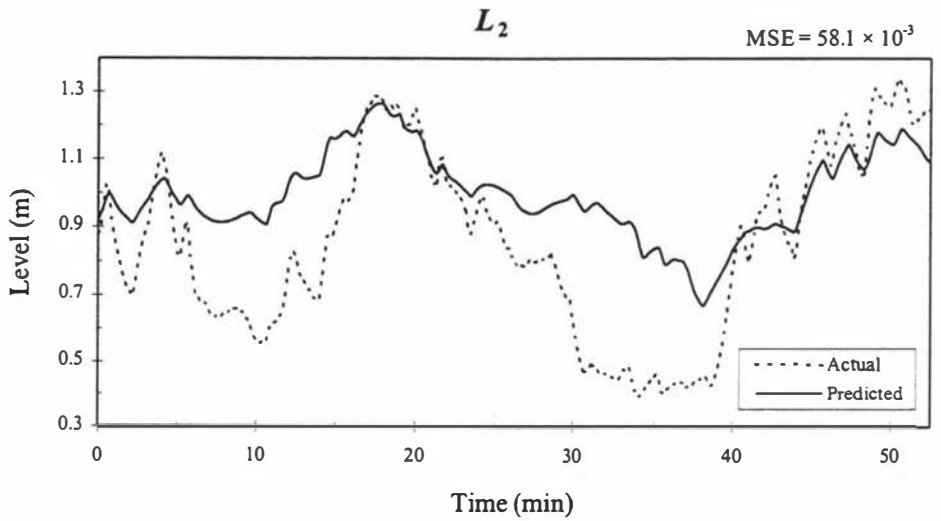


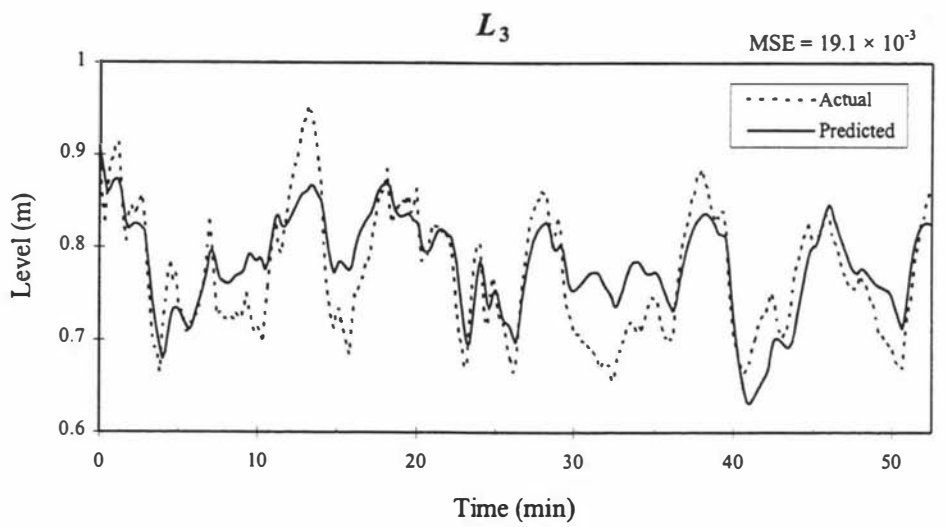




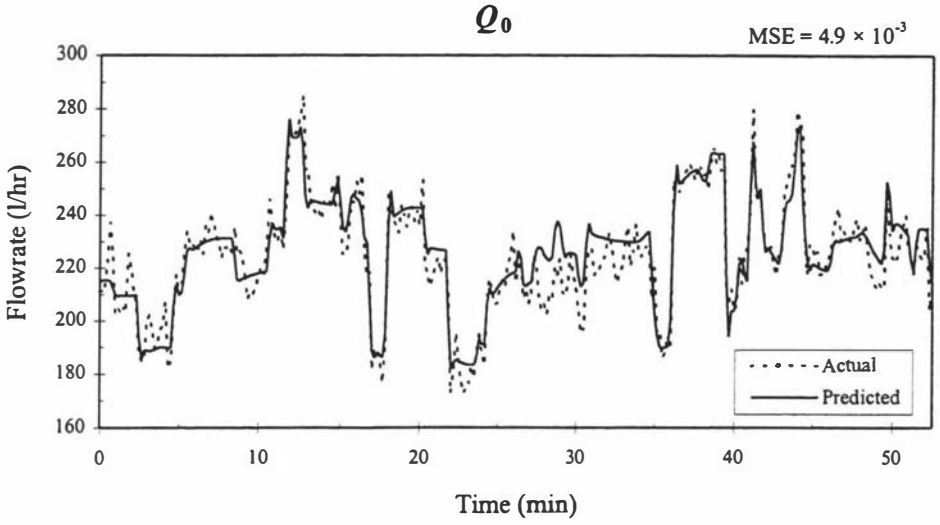
Full ARX model plots – Long-range predictions for validation 2 data

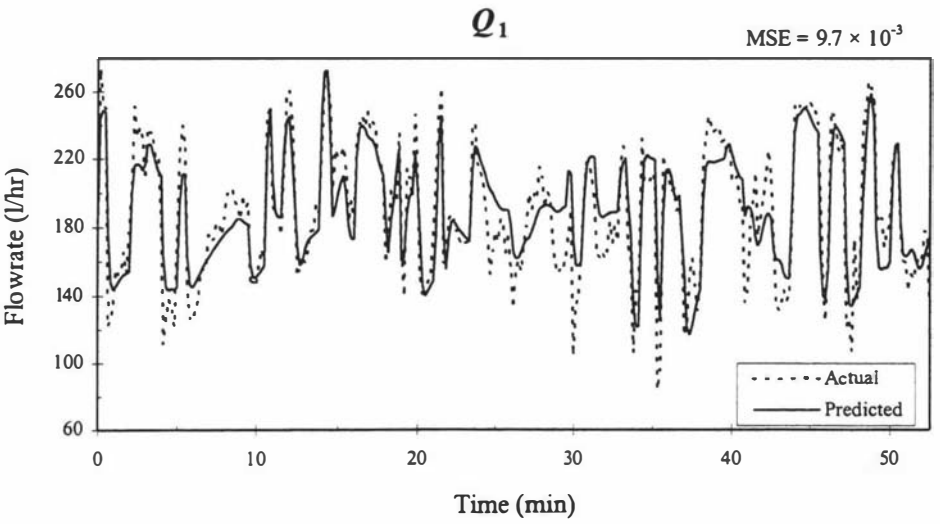


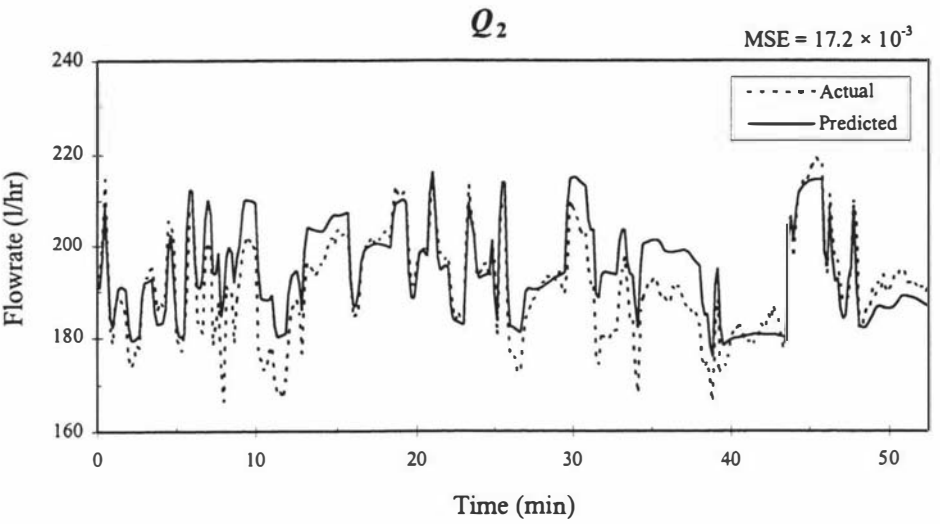




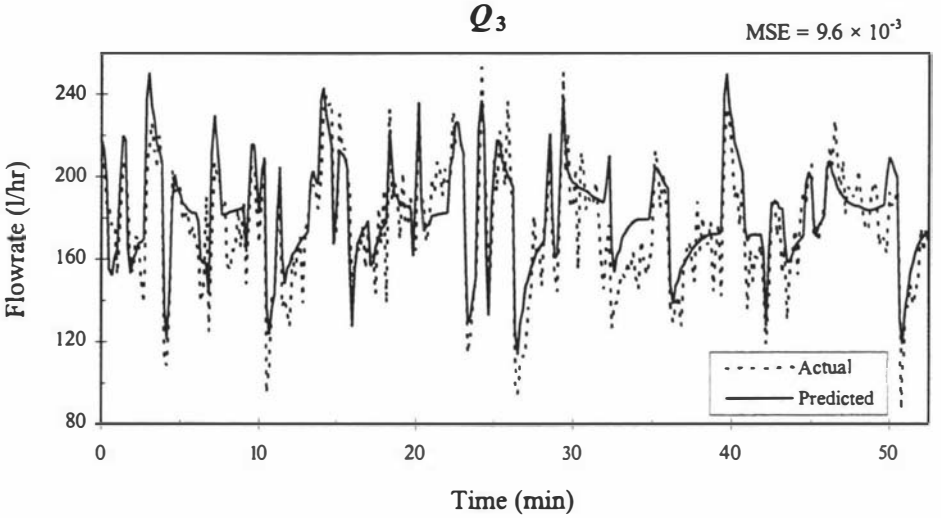
Full ARX model plots – Long-range predictions for validation 2 data







Full ARX model plots – Long-range predictions for validation 2 data



Appendix 11

Model Predictive Control MATLAB Files

MPCrun.m

Run file for the evaporator MPC simulation

```
% MPCRUN
% A script file to run the Model Predictive Control simulation

% Written by N.T. Russell
% Dept. Production Technology
% Massey University
% 1997

disp('Initialisation...')

% ---------- Define global variables ----------

global T1w1 T1w2 T1b1 T1b2 L1w1 L1w2 L1b1 L1b2 Q1w1 Q1w2 Q1b1 Q1b2;
global T2w1 T2w2 T2b1 T2b2 L2w1 L2w2 L2b1 L2b2 Q2w1 Q2w2 Q2b1 Q2b2;
global T3w1 T3w2 T3b1 T3b2 L3w1 L3w2 L3b1 L3b2 Q3w1 Q3w2 Q3b1 Q3b2;
global Tph1w1 Tph1w2 Tph1b1 Tph1b2 Tph2w1 Tph2w2 Tph2b1 Tph2b2;
global Q0w1 Q0w2 Q0b1 Q0b2
global Aq0 Bq0 Cq0 Aq1 Bq1 Cq1 Aq2 Bq2 Cq2 Aq3 Bq3 Cq3
global Ats Bts Cts Atp1 Btp1 Ctp1 Atp2 Btp2 Ctp2 At1 Bt1 Ct1
global At2 Bt2 Ct2 At3 Bt3 Ct3 A11 B11 C11 A12 B12 C12 A13 B13 C13
global INIT1 INIT2 INIT3 INITPH1 INITPH2 TS P M c k

% ---------- Define simulation settings ----------

% Define controller model
%Model = 'nnmodel'; getX0 = 'nnetx0';
Model = 'arxmodel'; getX0 = 'arxx0';

% Define run time (seconds)
time = 600;
% and sampling time (seconds)
dT = 10;

% Define prediction and control horizons
P = 10;
M = 10;


% ---------- Initialise plant and model ----------

% Initialise the inputs matrix: u=[Tsp N0];
u0 = [80.5 670];
Uopt=ones(P,1)*u0;
Ninputs = length(u0);

% Constant inputs after control horizon
Uopt(M+1:P,:) = ones(P-M,1)*Uopt(M,:);
```

```
% Define steps where inputs are applied
% (This enables reducing the number of input changes)
 c = [1 2 4 6 8];
 Uopt = Uopt(c,:);
end

% Initialisation for the 'plant' model and MPC model
init   % Call model initialisation file
X0 = feval(getX0,D); % Initial state vector for model

% Get initial outputs from the plant
 ul = [u0(1);u0(1)]; u2 = [u0(2);u0(2)];
tu = [0;dT];
[ti,X,y] = rk45('plant',dT,[],[1e-6,0.5,1]);
row = size(y,1);
y = y(row,:);

% Define controlled outputs
k = [4 10];
yp = y(k);

% Initialise model output
ym = feval(Model,u0,X0);
PI_int = ym([14:16])';
ym = ym(k);
d = yp-ym;

% Update initial conditions for plant and model
initmod

[rD,cD] = size(D);
D(2:rD,:) = D(1:rD-1,:);
D(1,:) = [y(1:13) u0(2) y(14:16) u0(1)];  % top row is most recent info.
X0 = feval(getX0,D,PI_int);

% ---------- Initialise controller settings ----------

% Initialise setpoints vector  [T1 Q0]
r = [72 220];
yp
Noutputs = length(r);
Ysp = ones(P,1)*r;

% Define bounds on inputs
Low = [65 200];
Up = [90 3000];

% Define weight matrices
Q = diag([100 0.1]);  % for output errors
R = diag([10 1e-2]); % for change in inputs


% ---------- Initialise the optimisation options ----------
FOpt = foptions;
FOpt(2) = 0.05;      % Tolerance for Uopt
FOpt(3) = 0.01;   % Tolerance for f (cost function)
FOpt(14) = 25*P*Ninputs;  % Max no. of optimiser iterations
FOpt(16) = 0.01;    % Minimum change for finite diffs in gradient calculation


% ---------- Initialise simulation outputs ----------

T = [0:dT:time]';
ys = ones(length(T),Noutputs)*NaN;
us = ones(length(T),Ninputs)*NaN;
ysp = ones(length(T),Noutputs)*NaN;
ymod = ones(length(T),Noutputs)*NaN;
df = ones(length(T),Noutputs)*NaN;
ys(1,:)=yp; ysp(1,:)=r;
us(1,:)=u0; ymod(1,:)=ym;
df(1,:)=d;  Y(1,:)=y;
```

```
% ----------- Initialise the plots ----------

subplot(221),
ltsp = line(T,us(:,1),'color',[1 0 0],'erasemode','xor');
lt1 = line(T,ys(:,1),'color',[0 1 1],'erasemode','xor');
lt1set = line(T,ysp(:,1),'color',[1 1 1],'erasemode','xor');
title('Temperatures');
set(gca,'ylim',[60,90]);
set(gca,'xlim',[0,max(T)]);

subplot(222),
ln0 = line(T,us(:,2),'color',[1 0 0],'erasemode','xor');
title('Feed Pump');
set(gca,'ylim',[500,1000]);
set(gca,'xlim',[0,max(T)]);

subplot(212)
lq0set = line(T,ysp(:,2),'color',[1 1 1],'erasemode','xor');
lq0 = line(T,ys(:,2),'color',[1 0 0],'erasemode','xor');
title('Feed Flow');
set(gca,'ylim',[150,300]);
set(gca,'xlim',[0,max(T)]);


% ************* This is the main loop *************

for t = 1:1:time/dT,
 fprintf('Timestep = %g\n',t)


% ------------ Find optimal control inputs ----------
disp('Optimise inputs...')
 Uopt = mpc(Model,Uopt,Ysp,Q,R,u0,Low,Up,FOpt,d,X0);
 u = Uopt(1,:);

% ---------- Addition of an input disturbance ----------

DIST = 1;  % Input disturbance input on/off
% Disturbance applied to steam temperature
 if t == 9 & DIST == 1,
   ud = 4;
   INIT1(7) = INIT1(7) + ud;
   INITPH2(3) = INITPH2(3) + ud;
 end

% ---------- Apply first input to the plant ----------
disp('Apply inputs to plant...')
runplant

% output vector:
% y = [Ts Tph1 Tph2 T1 T2 T3 L1 L2 L3 Q0 Q1 Q2 Q3 N1 N2 N3]

% Get the controlled outputs
yp = y(k);

% ---------- Generate output predictions from MPC model ----------
ym = feval(Model,u,X0);
PI_int = ym([14:16])';
ym = ym(k);

% display outputs while running:
pl = yp
mod = ym

% ---------- Calculate plant/model mismatch ----------
 d = yp-ym;

%filter disturbance
 f = 0.3;                                  % f is the proportion of the current disturbance
 d = (1-f)*df(t,:) + f*d;
```

```
 df(t+1,:) = d; % store disturbance


% ------------ Set up vectors of past information for MPC model ----------
[rD,cD] = size(D);
D(2:rD,:) = D(1:rD-1,:);
D(1,:)=[y(1:13) u(2) y(14:16) u(1)];  % top row is most recent info.
X0 = feval(getX0,D,PI_int);

% ----------- Set up initial inputs for the next timestep ----------
 u0 = u;
 [ru,cu]=size(Uopt);
 %Uopt = [Uopt(2:ru,:); Uopt(ru,:)];


% ---------- Store variables ----------
 Y(t+1,:) = y;                          % store all outputs (not just the controlled ones)
 ys(t+1,:) = yp(1,:);                   % store controlled outputs
 us(t+1,:) = u;                         % store control inputs
 ysp(t+1,:) = Ysp(1,:);                 % store setpoints
 ymod(t+1,:) = ym;                      % store model outputs


% ---------- Step change to setpoint ----------

SP = 0; % setpoint change on/off

if t >= 5 & SP==1
  r=[74 250];
  Ysp = ones(P,1)*r;
end

if t >= 65 & SP==1
  r=[70 220];
  Ysp = ones(P,1)*r;
end


% ---------- Update plots ----------
 set(ltsp,'ydata',us(:,1))
 set(ln0,'ydata',us(:,2))
 set(lt1,'ydata',ys(:,1))
 set(lq0,'ydata',ys(:,2))
 set(lt1set,'ydata',ysp(:,1))
 set(lq0set,'ydata',ysp(:,2))
 drawnow;
 set(0,'currentfigure',gcf);

end;

% ******* End of loop *******
```

MPC.m

MPC simulation M-file

```
function Uopt = mpc(Model,u,Ysp,Q,R,u0,Low,Up,FOpt,d,P1,P2,P3,P4)

% MPC  Model Predictive Controller
%
% Uopt = MPC('model', u, Ysp, Q, R, u0, Low, Up, FOpt,d, P1, P2, P3, P4)
%
% Returns optimal inputs, uopt, for rows(u) time steps ahead, as estimated from the predictive model. % 'model', using the
performance index:
%
%   J = SUM(E*Q*E' + deltaU*R*deltaU')
%
% where E is Ysp-Y-d and deltaU is u(k)-u(k-1) and Y is the predictive output of the model. u0 % contains the inputs at u(0).
```

```
%
% u is the starting point for the optimisation and u0 contains the current inputs, i.e. at u(0). Ysp is a row vector/matrix of output
% setpoints. Q and R are weighting matrices for the outputs and controls respectively. If not given these are defined as identity
% matrices with the size given by the number of outputs and inputs respectively. 'model' is the name of an m-file containing the %
% model. The model should return two arguments; a scalar of the function to be minimised, F, and a matrix of constraints, G (see
% the CONSTR function), i.e. [F,G] = model(u,P1,P2,P3,P4). The arguments P1, P2, P3, and P4 are passed directly to the
% function. If more than four are required use the GLOBAL command.
% Low and Up are row vectors of lower and upper bounds, respectively, on possible control inputs. They are set to zeros and
% ones, respectively, if not given. FOpt, if given, is an options vector which is passed to the CONSTR function.. d is the
% disturbance used in the performance index to account for plant-model mismatch.

% Written by Huub Bakker 1993
% Dept. Production Technology
% Massey University

% Adapted by N.T. Russell
% 1996

% Define global variables

global T1w1 T1w2 T1b1 T1b2 L1w1 L1w2 L1b1 L1b2 Q1w1 Q1w2 Q1b1 Q1b2;
global T2w1 T2w2 T2b1 T2b2 L2w1 L2w2 L2b1 L2b2 Q2w1 Q2w2 Q2b1 Q2b2;
global T3w1 T3w2 T3b1 T3b2 L3w1 L3w2 L3b1 L3b2 Q3w1 Q3w2 Q3b1 Q3b2;
global Tph1w1 Tph1w2 Tph1b1 Tph1b2 Tph2w1 Tph2w2 Tph2b1 Tph2b2;
global Q0w1 Q0w2 Q0b1 Q0b2;
global Aq0 Bq0 Cq0 Aq1 Bq1 Cq1 Aq2 Bq2 Cq2 Aq3 Bq3 Cq3
global Ats Bts Cts Atp1 Btp1 Ctp1 Atp2 Btp2 Ctp2 At1 Bt1 Ct1
global At2 Bt2 Ct2 At3 Bt3 Ct3 A11 B11 C11 A12 B12 C12 A13 B13 C13
global P M c count fmin

% Check for minimum number of arguments
if nargin < 10,
 error('Too few arguments')
 return;
end
count = 0; fmin = 1e10;

% Define the name of the performance index function
PerformFun = 'costfun';

% Find the number of inputs
NInputs = size(u,2);

% Define options for optimisation if not already specified
if length(FOpt) == 0,
 FOpt = foptions;      % Get the default options for the CONSTR function
 FOpt(16) = 0.01;      % Minimum change for finite diffs in gradient calculation
 FOpt(2) = 0.1;        % Tolerance for Uopt
 FOpt(3) = 0.1;        % Tolerance for f (cost function)
 FOpt(14) = 25*NInputs; % Max No of optimiser iterations (Keep it short!)
end

% Produce matrices of lower and upper bounds for CONSTR
if size(u,1)<P
 Low = ones(size(u,1),1)*Low;
 Up = ones(size(u,1),1)*Up;
else
 Low = ones(P,1)*Low;
 Up = ones(P,1)*Up;
end

% Construct a string to evaluate the CONSTR function
evalstr = ['constr(PerformFun,u,FOpt,Low,Up,[],Model,Ysp,Q,R,u0,d'];
 for i = 1:nargin - 10
  evalstr = [evalstr,',P',int2str(i)];
 end;
evalstr = [evalstr, ')'];

% Call the CONSTR function with the required arguments
Uopt = eval(evalstr);
Uopt
```

costfun.m

Calculates performance function for MPC optimisation

```
function [f,g] = costfun(u,Model,Ysp,Q,R,u0,d,P1,P2,P3,P4);

% COSTFUN Evaluate cost function
%       [F,G] = COSTFUN(U,'model',Ysp,Q,R,U0,d,P1,P2,P3,P4) returns the performance index for the model 'model'
%       under the given control inputs. 'model' is a string giving the name of the m-file containing the model. U0 is the
%       input at U(0) and d is the disturbance which accounts for plant-model mismatch.
%       The performance index is given by
%
%               J = SUM(E*Q*E' + deltaU*R*deltaU')
%
%               where E is Ysp-Y-d and deltaU is U(k)-U(k-1)
%
%       G is the constraint vector with constraints on the change in u and for inputs beyond the control horizon, M.
%       The 'model' should return the predicted outputs and any constraints if necessary, Ym = model(U,P1,P2,P3,P4).
%
%       [F,G] = COSTFUN(U,'model',YSP,Q,R,U0,d,P1,P2,P3,P4) allows the arguments,
%       P1, P2, P3 etc, to be passed directly to the function.

%       Written by Huub Bakker 1993.
%       Dept. Production Technology
%       Massey University

%       Re-written by N.T. Russell, 1996

% Define global variables

global T1w1 T1w2 T1b1 T1b2 L1w1 L1w2 L1b1 L1b2 Q1w1 Q1w2 Q1b1 Q1b2;
global T2w1 T2w2 T2b1 T2b2 L2w1 L2w2 L2b1 L2b2 Q2w1 Q2w2 Q2b1 Q2b2;
global T3w1 T3w2 T3b1 T3b2 L3w1 L3w2 L3b1 L3b2 Q3w1 Q3w2 Q3b1 Q3b2;
global Tph1w1 Tph1w2 Tph1b1 Tph1b2 Tph2w1 Tph2w2 Tph2b1 Tph2b2;
global Q0w1 Q0w2 Q0b1 Q0b2
global Aq0 Bq0 Cq0 Aq1 Bq1 Cq1 Aq2 Bq2 Cq2 Aq3 Bq3 Cq3
global Ats Bts Cts Atp1 Btp1 Ctp1 Atp2 Btp2 Ctp2 At1 Bt1 Ct1
global At2 Bt2 Ct2 At3 Bt3 Ct3 A11 B11 C11 A12 B12 C12 A13 B13 C13
global P M c k count fmin

Ninputs = size(u,2);
% Weighting on error at Pth step
Px = diag([1000 10]);

% Evaluate the model
if nargin == 7, Ym = feval(Model, u); end;
if nargin == 8, Ym = feval(Model, u, P1); end;
if nargin == 9, Ym = feval(Model, u, P1, P2); end;
if nargin == 10, Ym = feval(Model, u, P1, P2, P3); end;
if nargin == 11, Ym = feval(Model, u, P1, P2, P3, P4); end;

% Error signal with compensation for plant/model disturbance
if size(Ysp,1) ~= P
  Ysp = ones(P,1)*Ysp;
end

Ym = Ym(:,k);    % controlled outputs

e = Ysp-Ym-ones(P,1)*d;

% Extend input matrix to include all P timesteps
u = extendu(u,c,P);

% Input constraints
ui = [u0; u];
du = ui(2:P+1,:) - ui(1:P,:);
dulim = [2 400]; % limits on du
for i=1:Ninputs
   g1(:,i) = abs(du(:,i)) - dulim(i);
end
g1=g1(:);
```

```
% Constraints on Ym
ymin=0.5*Ysp; ymax=1.25*Ysp;
Noutputs = size(Ysp,2);
for i=1:Noutputs
  g2(:,i) = ymin(:,i) - Ym(:,i);
  g3(:,i) = Ym(:,i) - ymax(:,i);
end
g2=g2(:); g3=g3(:);

% Form constraint vector
g=[g1;g2;g3];

% Cost function calculation
f = sum(diag(e*Q*e')) + sum(diag(du*R*du')) + e(P,:)*Px*e(P,:)';

% To periodically display function evaluations during optimisation

count = count+1;
if f < fmin
 fmin = f;
end
if rem(count,100) == 0
 fprintf('%g f = %g\n',count,fmin)
 fmin = 1e10;
elseif rem(count,20) == 0
 fprintf('.')
end
```

arxmodel.m

Calls the evaporator ARX SIMULINK model (*arxevapm.m*) during MPC simulation. The *arxevapm.m model* is exactly the same as *arxevap.m* but configured so that it can be embedded within an MPC control strategy.

```
function y = arxmodel(u,X0);

% Calls the SIMULINK evaporator ARX model
% The model predictions are of the form:
% y = [Ts Tph1 Tph2 T1 T2 T3 L1 L2 L3 Q0 Q1 Q2 Q3]
% u is the input matrix
% X0 is the current state of the model (see arxX0.m)

% Written by N.T. Russell
% Dept. Production Technology, Massey University
% 1997

% Define global variables

global Aq0 Bq0 Cq0 Aq1 Bq1 Cq1 Aq2 Bq2 Cq2 Aq3 Bq3 Cq3
global Ats Bts Cts Atp1 Btp1 Ctp1 Atp2 Btp2 Ctp2 At1 Bt1 Ct1
global At2 Bt2 Ct2 At3 Bt3 Ct3 A11 B11 C11 A12 B12 C12 A13 B13 C13
global P c

if size(u,1)>1 & size(u,1)<P
  u = extendu(u,c,P);
end

% Define parameters:
p = size(u,1);
time=[0:p-1]';

% Define Inputs
U = [time u];

% Simulate the ARX evaporator model
[t,x,y] = rk23('arxevapm',p-1,X0,[1e-6,1,1],U);
```

nnmodel.m

Calls the evaporator neural network SIMULINK model (*nnevapm.m*) during MPC simulation. The *nnevapm.m model* is exactly the same as *nnevap.m* but configured so that it can be embedded within an MPC control strategy.

```
function y = nnmodel(u,X0);

% Calls the SIMULINK evaporator neural network model
% The model predictions are of the form:
% y = [Ts Tph1 Tph2 T1 T2 T3 L1 L2 L3 Q0 Q1 Q2 Q3]
% u is the input matrix
% X0 is the current state of the NN model (see NnetX0.m)

% Written by N.T. Russell
% Dept. Production Technology, Massey University
% 1997

% Define global variables

global T1w1 T1w2 T1b1 T1b2 L1w1 L1w2 L1b1 L1b2 Q1w1 Q1w2 Q1b1 Q1b2;
global T2w1 T2w2 T2b1 T2b2 L2w1 L2w2 L2b1 L2b2 Q2w1 Q2w2 Q2b1 Q2b2;
global T3w1 T3w2 T3b1 T3b2 L3w1 L3w2 L3b1 L3b2 Q3w1 Q3w2 Q3b1 Q3b2;
global Tph1w1 Tph1w2 Tph1b1 Tph1b2 Tph2w1 Tph2w2 Tph2b1 Tph2b2;
global Q0w1 Q0w2 Q0b1 Q0b2
global INIT1 INIT2 INIT3 INITPH1 INITPH2 P c

if size(u,1)>1 & size(u,1)<P
  u = extendu(u,c,P);
end

% Define parameters:
p = size(u,1);
time=[0:p-1]';

% Define Inputs
U = [time u];

% Simulate the neural network evaporator
[t,x,y] = rk23('nnevapm',p-1,X0,[1e-6,1,1],U);
```

arxX0.m

Updates the current state of the ARX evaporator model *arxevapm.m*.

```
function X0 = arxx0(D,PI_int)

% D inputs and outputs at t-1 to t-6 dimensions:=[6 x variables]
% D = [Ts Tph1 Tph2 T1 T2 T3 L1 L2 L3 Q0 Q1 Q2 Q3 N0 N1 N2 N3 Tsp];
% PI_int are the current values for the PI controllers' integrals

% Define past conditions:
Ts=D(:,1); Tp1=D(:,2); Tp2=D(:,3); T1=D(:,4); T2=D(:,5); T3=D(:,6);
L1=D(:,7); L2=D(:,8); L3=D(:,9); Q0=D(:,10); Q1=D(:,11); Q2=D(:,12);
Q3=D(:,13); N0=D(:,14); N1=D(:,15); N2=D(:,16); N3=D(:,17); Tsp=D(:,18);

if nargin==1
  PI_int = [0;0;0];
```

```
end

% X0 is a 130 element vector defining the current state of the ARX model

X0 = [PI_int;Ts(1:2);T2(1);Q0(1:5);T1(1:3);T3(1:4);Q0(1);Tp1(1:3);Ts(1:3);Tp1(1);...
Q0(1:2);Tp2(1:5);T1(1);N0(1:3);Q0(1:2);Q0(1:2);Q1(1:3);T1(1:5);L1(1);T1(1:4);T3(1:2);...
Q1(1:6);T2(1:2);T1(1)-T2(1);L1(1:2);N1(1);Q1(1);Q1(1:3);Q2(1);T2(1:3);L2(1:2);...
T2(1:6);Tp1(1:2);Q2(1);T3(1:2);L2(1:2);N2(1:2);Q2(1);T3(1);Q2(1:2);...
Q3(1:2);T3(1);L3(1:2);N3(1:4);Q3(1:4);Tsp(1:2);T1(1:5);Q0(1:3);Ts(1:4);...
Q3(1);Q2(1);Q2(1);Q1(1);Q1(1);Q0(1)];
```

nnetX0.m

Updates the current state of the neural network evaporator model *nnevapm.m*.

```
function X0 = nnetx0(D,PI_int)

% D inputs and outputs at t-1 to t-7 dimensions:=[7 x variables]
% D=[Ts Tph1 Tph2 T1 T2 T3 L1 L2 L3 Q0 Q1 Q2 Q3 N0 N1 N2 N3 Tsp]:
% PI_int are the current values for the PI controllers' integrals

% Define past conditions:
Ts=D(:,1); Tp1=D(:,2); Tp2=D(:,3); T1=D(:,4); T2=D(:,5); T3=D(:,6);
L1=D(:,7); L2=D(:,8); L3=D(:,9); Q0=D(:,10); Q1=D(:,11); Q2=D(:,12);
Q3=D(:,13); Tsp=D(:,18);

if nargin==1
  PI_int = [0;0;0];
end

% X0 is a 161 element vector defining the current state of the NN model

% define different delay vectors
a=[1 5 7];
b=[1 2 4];
c=[1 2 3];
d=[6 4 3 2];

X0 = [PI_int;Ts(a);T2(a);Q0(a);T1(a);T3(b);Q1(b);Tp1(b);Ts(a);Tp1(a);...
Q0(a);Tp2(a);Q0(1);Q0(c);Q1(c);T1(c);L1(c);T1(b);T3(b);Q1(b);T2(b);...
Q1(1);Q1(c);Q2(c);T2(c);L2(c);T2(b);Tp1(b);Q2(b);T3(b);Q2(1);Q2(c);...
Q3(c);T3(c);L3(c);Q3(1);Tsp(c);T1(c);Q0(c);Ts(c);Q2(3);T2(3);Tp1(3);...
T3(3);Q3(1);Q2(1);Q1(3);T2(3);T3(3);T1(3);Q2(1);Q1(1);Q0(d);T1(d);...
T2(d);Ts(d);Q1(1);Q0(1);Tp2(d);Q0(d);Tp1(d);Ts(d);Tp1(3);Q1(3);T3(3)];
```

runplant.m

Calls the analytical model of the evaporator (*plant.m*) during MPC simulation. *plant.m* is exactly the same as *ffevap.m* but configured for use with the MPC simulation.

```
% runplant
% Run file for simulating the feedforward evaporator model with MPC.

% Written by N.T. Russell
% Dept. Production Technology
% 1996

u1 = [u(1);u(1)]; u2 = [u(2);u(2)];
```

```
min_step = 0.5;

% Reduce step size if a have a large change in inputs
if t > 1
 if abs(u(1)-us(t-1,1))>2 | abs(u(2)-us(t-1,2))>50
  min_step = 0.25
 end
end

tu=[0;dT];

[ti,X,Yout] = rk45('plant',dT,[],[1e-6,min_step,1]);   % Simulate the evaporator

row = size(Yout,1);
y = Yout(row,:);

% update model initial conditions for next run
initmod
clear ti X Yout row u1 u2 tu
```

extendu.m

Extends the input matrix to span all time steps of the prediction horizon.

```
function u_ext = extendu(u,c,P)

% Extends input matrix to include all P timesteps - for use in optimisation phase during MPC simulation
% Inputs can be applied at specific steps (defined by c) to reduce the dimensions of the MPC optimisation.
% u is the input matrix
% c is the vector defining at which steps the inputs are applied
% P is the length of prediction horizon

% Written by N.T. Russell
% Dept. Production Technology, Massey University
% 1997

L = length(c);
for n = 1:L-1
  for i = c(n):c(n+1)-1
    u_ext(i,:) = u(n,:);
  end
end

for i = c(L):P
  u_ext(i,:) = u(L,:);
end
```

init.m

Definition of initial conditions for the evaporator models for use at the start of the MPC simulation.

```
% init
% Definition of initial conditions for the evaporator models for use at the start of the MPC simulation.
% Creates six global variables which contain the initial values.

% Written by N.T. Russell
% Dept. Production Technology
% 1996

Tsp=u0(1); N0=u0(2);
N1=1340; N2=1760; N3=1840;

L1=1; L2=1; L3=1; LT=0.75;
Qin=220/3.6e6; Pa=101300; Ta=18; T0=18;

T11=72; T12=63.5; T13=42;
T1=T11; T2=T12; T3=T13;
Tph1=37.5; Tph2=70; Ts0=Tsp; T0=18;
kp=2; ki=0.2; kd=0; Tc=21;

UPh1=1900; UPh2=1900; U1=2850; U2=2500; U3=1740;
V=100; Cv = 5.90e-7;

% Calculation and conversion of initial conditions.

Ae1=0.3313; Ae2=0.3313; Ae3=0.2137; Aph=pi*5.7*12.7e-3;
q1=U1*Ae1*(Ts0-T1); q2=U2*Ae2*(T1-T2);
q3=U3*Ae3*(T2-T3); qPh1=UPh1*Aph*(Tph1-T0)/log((T3-T0)/(T3-Tph1));

Q0=ffCalcQ0(LT,Pa,T1,N0)*3.6e6;
mv1=10; mv2=8; mv3=21;
Qp1=Q0-mv1; Qp2=Qp1-mv2; Qp13=Qp2-mv3;
Qp1=Qp1/3.6e6; Qp2=Qp2/3.6e6; Qp3=Qp3/3.6e6; Q0=Q0/3.6e6;
Mv1=mv1*4/3600; Mv2=mv2*4/3600; Mv3=mv3*4/3600;
MvPh1 = 10/3600*calctph(Q0); mvPh1=10/3600;

Qd1=Q0; Qd2=Qp1; Qd3=Qp2;

h1=(Qd1/2.36e-4)^2/19.62; h2=(Qd2/2.36e-4)^2/19.62;
h3=(Qd3/2.36e-4)^2/19.62;

% Define global variables for analytical model
INITPH1=[LT;MvPh1;Tph1;Ta;Qin;T0;T3;Q0;Mv3;qPh1;Tph1;UPh1];
INITPH2=[Tph2;Ta;Ts0;Tph1;Q0;UPh2;0];
INIT1=[h1;L1;T1;T11;Mv1;Ta;Ts0;Tph2;T2;N1;Qd1;q1;Tph2;T1;Q0;U1;0];
INIT2=[h2;L2;T2;T12;Mv2;Ta;T1;T3;N2;Qd2;q2;T1;T2;Qp1;T11;U2;0];
INIT3=[h3;L3;T3;T13;Mv3;Ta;T0;Tph1;T2;N3;Qd3;q3;Pa;T2;T3;Qp2;mvPh1;T12;Cv;U3;0];
TS=ones(5,1)*Ts0;

Q0 = Q0*3.6e6; Q1 = Q0-mv1; Q2 = Q1-mv2; Q3 = Q2-mv3;

% Initial conditions for controller model
v=[Ts0 Tph1 Tph2 T11 T12 T13 L1 L2 L3 Q0 Q1 Q2 Q3 N0 N1 N2 N3 Tsp];

% if using ARX model
```

```
D = ones(6,1)*v;

% if using NNet model
% D = ones(7,1)*v;
```

initmod.m

Updates the current state of the analytical model during MPC simulation.

```
% initmod
% Updates the current state of the analytical model during MPC simulation.
% Used within the runplant.m routine

% Written by N.T. Russell
% Dept. Production Technology, Massey University
% 1996

h1=h1(row); h2=h2(row); h3=h3(row); LT=LT(row); Mv1=Mv1(row); Mv2=Mv2(row); Mv3=Mv3(row); mvPh1=mvPh1(row);
Qd1=Q1(row,1)/3.6e6; Qd2=Q2(row,1)/3.6e6; Qd3=Q3(row,1)/3.6e6; q1=q1(row); q2=q2(row); q3=q3(row); qPh1=qPh1(row);
T1=T1(row); T2=T2(row); T3=T3(row); int_Ts=int_Ts(row); int_L1=int_L1(row); int_L2=int_L2(row); int_L3=int_L3(row);

Q0 = y(10)/3.6e6; Q1 = y(11)/3.6e6; Q2 = y(12)/3.6e6;

MvPh1 = mcPh1(row)*calctph(Q0);

INITPH1 = [LT;MvPh1;y(2);Ta;Qin;T0;T3;Q0;Mv3;qPh1;y(2);INITPH1(12)];
INITPH2 = [y(3);Ta;y(1);y(2);Q0;INITPH2(6);int_Ts];
INIT1 = [h1;y(7);T1;y(4);Mv1;Ta;y(1);y(3);T2;y(14);Qd1;q1;y(3);T1;Q0;INIT1(16);int_L1];
INIT2 = [h2;y(8);T2;y(5);Mv2;Ta;T1;T3;y(15);Qd2;q2;T1;T2;Q1;y(4);INIT2(16);int_L2];
INIT3 = [h3;y(9);T3;y(6);Mv3;Ta;T0;y(2);T2;y(16);Qd3;q3;Pa;T2;T3;Q2;mvPh1;y(5);INIT3(19);.INIT3(20);int_L3];
TS = [y(1);TS(1:4)];
```