



Cranfield University

School of Mechanical Engineering

Applied Energy and Optical Diagnostics Group

EngD

1997/98

Philip David Brierley

**Some Practical Applications of  
Neural Networks in the Electricity  
Industry**

Supervisors  
Dr. W.J. Batty  
Professor D. Myddelton

September 1998

This thesis is submitted in partial fulfilment of the requirements for the Degree of  
Doctor of Engineering

ProQuest Number: 10832352

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10832352

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by Cranfield University.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code  
Microform Edition © ProQuest LLC.

ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 – 1346



## ABSTRACT

The development of an optimising model predictive controller for domestic storage radiators was the ultimate goal of this research project. Neural networks are used to create empirical models that are used to predict the likely temperature response of a room to the charging of a storage radiator. The charging strategy can then be optimised based on the real-time price of electricity.

Neural network modelling is investigated by looking at the load forecasting problem. It is shown how accurate neural models can be created and demonstrated exactly how they process the data. Very specific rules are extracted from the neural network that can model the load to a reasonable accuracy.

An efficient optimisation technique is sought by optimising the charging of a domestic hot water tank based on actual consumption data and the pool price of electricity. Initially genetic algorithms were tried but their weaknesses are demonstrated. A stochastic hill climbing method was found to be more suitable. Monetary saving of 40% over the existing E7 tariff was common.

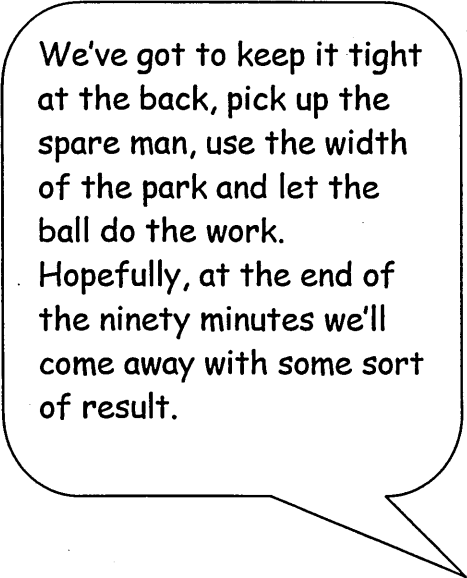
The modelling and optimisation are brought together in a storage radiator simulation. There are improvements in cost and electricity consumption over E7 primarily due to the ability to look ahead and avoid overheating.

A prototype neural controller is developed and tested in a real house. The results are very encouraging.

## Declaration

All work in this thesis and the resulting publications are the sole work of the author  
unless otherwise stated

**To my Grandparents**



We've got to keep it tight at the back, pick up the spare man, use the width of the park and let the ball do the work. Hopefully, at the end of the ninety minutes we'll come away with some sort of result.

Tommy Brown, Fulchester United Supremo

Introducing...



The Author

I was born in 1969 in Oldham where I spent the first seventeen years of my life, mainly playing football and cricket.

I graduated from Hull University in 1991 with a first class honours in Mechanical Engineering. My final year project involved a novel method of extracting oil from rapeseed by using cavitation shock waves instead of screw presses.

I then pursued my interest in energy efficiency by doing an M.Sc. in Energy Studies at Cardiff. The project done there was building and testing a micro-chp system using a Stirling engine from the 1950's, an old tea urn and a heat exchanger from a Ford Cortina.

Wishing to further pursue my main academic interest in helping to preserve the earth's natural resources by improving energy efficiency, I started this work at Cranfield in 1994. I hope it is an enjoyable read!

I can be contacted by email at

[pdbrierley@netscape.net](mailto:pdbrierley@netscape.net)

or

[info@neusolutions.com](mailto:info@neusolutions.com)

and would welcome any correspondence regarding neural networks or life in general.

phil

## Acknowledgements

This research is a direct result of the forward thinking of Mr Andy Robinson at Eastern Electricity. I am sincerely grateful to him, Eastern and my supervisor Dr. Batty for creating the opportunity for me pursue what was an interesting project.

I am grateful to Liz Cutting and Dave Murdin at Eastern for providing me with the electricity consumption data and allowing the results to be published. Thanks again to Andy Robinson for the water data and the provision of the data logging equipment.

The Eastern Region Energy Group, The Royal Academy of Engineers and the Engineering Doctorate Centre have all contributed financially so that I could attend conferences and visit other academic institutions.

Thanks to Professor Pierre-Yves Glorennec for allowing me to visit him at INSA de Rennes and for inspiring some ideas in my fledgling days when I didn't know why, what or how.

Thanks to Malcolm Clapp and Glen Baglin at Satchwell Controls for providing the control equipment hardware. I am very grateful to Andrew Lewis and Jonathan Lawson for the time they spent helping me interface all the hardware and software to create the neural controller.

Thanks to Dimplex for the donation of a storage heater.

Finally, but by no means least, thanks to Sharon for all the commuting you have endured over the last three years.



# CONTENTS

<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Reasons behind this Thesis	1
1.2 Nature of the Work	3
1.3 Chapter Contents	4
1.4 Contribution of this Thesis	6
1.5 Publications from this Thesis	7
1.6 Outcomes from this Thesis	8
<b>2 FEEDFORWARD NEURAL NETWORKS</b>	<b>9</b>
2.1 What are Neural Networks?	9
2.2 Why use Neural Networks?	10
2.3 How do Neural Networks Process Information?	12
2.4 Things to be aware of...	15
2.4.1 Over-fitting and Generalisation	15
2.4.2 Extrapolation	16
2.4.3 The Function being Minimised	17
2.4.4 Local Minima	18
2.4.5 Data Encoding	20
2.5 Chapter Summary	21
<b>3 ELECTRIC LOAD MODELLING</b>	<b>22</b>
3.1 The Data being Modelled	22
3.2 Why Forecast Electricity Demand?	25
3.3 Previous Work	27
3.4 Network Used	28
3.5 Total Daily Load Model	29
3.6 Over-fitting and Generalisation	37
3.7 Rule Extraction	40
3.7.1 Day of the Week	41
3.7.2 Time of Year	43
3.7.3 Growth	44
3.7.4 Weather Factors	45
3.7.5 Holidays	48
3.8 Model Comparisons	51

3.9	Half Hourly Model	53
3.9.1	Initial Input Data	53
3.9.2	Results	54
3.9.3	Past Loads	61
3.9.4	How the Model is Working	62
3.9.5	Extracting the Growth	63
3.10	Populations of Models	64
3.11	Traditional Load Forecasting Methods	65
3.11.1	Multiple Linear Regression	67
3.11.2	Stochastic Time Series	68
3.12	Load Forecasting in Practice	70
3.13	Chapter Summary	71
<b>4</b>	<b>GENETIC INSPIRED OPTIMISATION</b>	<b>73</b>
4.1	What are Genetic Algorithms?	73
4.2	How do GAs Work?	74
4.3	The GA Operators	75
4.4	Implementation	76
4.4.1	Encoding	76
4.4.2	Population Size	77
4.4.3	Selection	78
4.4.4	Crossover	79
4.4.5	Mutation	79
4.5	Experiments with GAs	79
4.5.1	Chinese Hat Optimisation Problem	79
4.5.2	Results	80
4.5.3	Other Iterated Hill-Climbing Methods	85
4.5.4	Royal Road Functions	88
4.6	Chapter Summary	98
<b>5</b>	<b>DOMESTIC HOT WATER OPTIMISATION</b>	<b>101</b>
5.1	Introduction	101
5.2	Model to be Optimised	103
5.3	Simulated Water Heating Model	105
5.4	Data Used	106
5.5	Optimisation Procedure	107
5.6	Profiling Usage Patterns	108
5.7	Results	109
5.8	Discussion of Results	121
5.8.1	Does Water Storage Save Money ?	121
5.8.2	How did the Profiling Perform ?	121
5.8.3	How Much Money could be Saved?	123

5.8.4	Why is the Optimised Schedule Sometimes Worse?	123
5.8.5	How is the Optimisation Working	125
5.8.6	Local Minima	126
5.9	Chapter Summary	128
<b>6 STORAGE RADIATOR CONTROLLER SIMULATION</b>		<b>129</b>
6.1	What are Storage Radiators?	129
6.2	Room Thermal Model	131
6.3	Neural Network Emulator	132
6.4	Optimisation Procedure	134
6.5	Simulation Procedure	136
6.6	Results	138
6.6.1	Did the Controller Work?	138
6.6.2	Why do the Large Initial Errors Occur?	139
6.6.3	Performance of the 1-step-ahead Predictor as a Recursive 48-step-ahead Predictor	141
6.6.4	Comparison with other Heating Strategies	142
6.7	Emulator Improvements	145
6.8	Chapter Summary	145
<b>7 REAL NEURAL STORAGE RADIATOR CONTROL</b>		<b>146</b>
7.1	Background	146
7.2	Data Analysis	150
7.3	Neural Controller	155
7.4	Chapter Summary	159
<b>8 PROJECT OVERVIEW</b>		<b>160</b>
8.1	About this Chapter	160
8.2	Load Forecasting	161
8.3	Water Optimisation	161
8.4	Intelligent Heating Control	162
8.5	Experiences of Pursuing a Neural Network Project	162
8.6	Cost Analysis	164
8.7	The Future	164
<b>REFERENCES</b>		<b>165</b>
Appendix A	Back Propagation Weight Update Rule	174

B	MLP Code	183
C	More on MLPs	187
D	Genetic Algorithm Code	195
E	Water Optimisation Code	198
F	Storage Heater Thermal Model Code	202



# Introduction

---

## 1.1 Reasons behind this Thesis

On the 13<sup>th</sup> September 1998, the UK electricity industry entered its final stage of de-regulation by allowing domestic customers from four selected regions a free choice of supplier. If everything goes to plan then eventually every customer in the country will be able to buy electricity from any supplier, regardless of geographical location. It is envisaged that the increased competition between suppliers will lead to improved products and services being offered and a more efficient electricity industry.

Electricity 'products' are basically tariffs, contractual arrangements that set the price paid by the customer to the supplier. The actual price that suppliers pay generators varies half-hourly as determined by the market forces of supply and demand. This

wholesale trading of electricity is supervised by the Pool, which sets the half-hourly prices for the day ahead.

In conventional free markets, the price consumers pay will reflect the production cost of the commodity involved. In the electricity market this is known as real-time or spot pricing [1,2,3,4], where the amount charged varies half-hourly in line with the 'pool' price. As the cost of sophisticated metering equipment continues to fall, the prospects are ever increasing that real-time pricing will become a viable domestic tariff.

With real-time pricing, customer savings will be made by switching load to the cheaper periods of the day. For certain time dependent usage such as cooking, lighting and television, this is impractical, whereas for domestic chores such as washing and vacuuming it is more of an inconvenience.

Thermal storage devices such as hot water tanks and storage radiators are an un-intrusive method of utilising cheap rate electricity, which can be stored for use at a later time. Such appliances already exist for time-of-use (ToU) tariffs such as Economy 7 (E7), where a reduced unit price is offered for 7 hours during the night.

These off-peak ToU tariffs were introduced as pre-privatisation attempts at trying to level the demand and thus increasing the load factor. They are not satisfactory for several reasons; from the suppliers' point of view they have created a surge in demand at midnight as all the appliances are automatically activated. This causes several problems that the tariff was supposed to alleviate, not least of which is the subsequent Pool price increase resulting from the increased demand. For consumers, off-peak electric storage heating is expensive and does not always provide satisfactory thermal comfort requirements. This is because there is generally at least a 9-hour gap between the end of the charging period at 7 a.m. and the time when the warmth is required in the evening.

As a result of their limitations, storage radiators tend to be found in places where the low maintenance levels are an advantage or heat is required throughout the day, such as rented and sheltered accommodation. The storage radiator market is not expanding with most sales to existing users who require replacement units.

---

Storage radiators are potentially an excellent method of shifting load and utilising cheap rate electricity. The problem that they have is their inflexible control caused by the ToU tariffs within which they operate. Under real-time pricing the control would have to be more flexible, decisions being made on a daily basis when to charge the core. An automatic controller is required that can determine a charging schedule that will provide the required thermal comfort at the cheapest cost. The work in this thesis is an investigation of a potential solution to this control and optimisation problem.

## 1.2 Nature of the Work

If the price and thermal requirements are known for the day ahead, determining a charging strategy over this horizon is an optimisation problem in deciding when to switch the heaters on and off. To evaluate the suitability of each potential solution, a thermal model is required that will mimic the system behaviour to predict the outcome of various strategies.

For hot water tanks, creating a mathematical model of their thermal behaviour is a simple procedure. To produce a controller, all that would be required is an efficient optimisation method and a means of receiving information on future price and demand requirements. It would be relatively easy to produce an off the shelf controller as tanks are mass-produced and will therefore all have the same thermal characteristics.

For storage radiators the optimisation is not so simple, as there is no ready thermal model of the room from which to evaluate the potential charging strategies. What is required is an 'intelligent' controller that can learn the thermal response of the actual room that the heater is in. Once this model is created it can be used for predictive control of the room temperature, with a charging strategy being determined that satisfies the required temperature profile whilst also attempting to minimise costs.

An artificial neural network is a modelling tool that can learn relationships between data without the need for any pre-defined model form. All that is required are previous ex-

amples of the past performance. An evaluation of the potential for using neural networks for model predictive control of storage radiators is the ultimate goal of this research.

### 1.3 Chapter Contents

Chapter 2 introduces a particular type of neural network, the feedforward multi-layer perceptron (MLP). Neural modelling is a relatively new development and there are few (if any) reference sources<sup>1</sup> that clearly illustrate all the issues involved in creating a good model, one reason being that their internal operation is generally poorly understood. For successful employment of neural networks and to avoid the many pitfalls, it is essential that they are used with care. Chapter 2 illustrates with examples how a MLP processes information and highlights some of the issues to be aware of.

In chapter 3 a MLP is used as a tool to help understand the factors that influence electricity consumption patterns for a large region. The purpose of this work was to become familiar with the capabilities of MLPs using real data, as the limitations of simulated data soon became evident. Factors that affect the total daily load over an eight-year period are initially investigated. This is achieved by improving the model by investigating the evidence supplied by the errors. Once a satisfactory model is created the neural 'black box' is opened to determine how the data is being processed. By weight pruning, the model is simplified, meaningful information extracted and a simple set of rules created that describe the load behaviour. The model is then used as a one-day ahead predictor with subsequent analysis to highlight how improvements could be made. Finally, a single model is created and analysed for the half-hourly load over a one-year period.

In the storage heater controller, the neural model will be used as a basis for determining the suitability of potential heating strategies. Methods of searching for an optimum strategy are investigated in chapter 4. A genetic algorithm (GA) is a technique that has

---

<sup>1</sup> See Appendix C for information sources that the author found particularly useful.



---

gained recent popularity as a robust optimisation tool. In chapter 4 an empirical investigation lead to the conclusion that they lack the ability to efficiently search out a global minimum and are not suitable as an optimisation method for the controller. A similar technique, random mutation hill climbing (RMHC), consistently outperformed GAs.

In chapter 5, simulated optimisation of a domestic hot water tank is performed. Data was available from a monitoring project that logged every water outlet in 100 houses every half-hour over one year. From this data, half-hourly hot water demand profiles were created. A mathematical model of a hot water tank and half-hourly 'pool' prices were used as a basis for optimising a daily charging schedule. The results were compared with existing available charging profiles over which significant monetary savings of the order of 20-50% were made.

In chapter 6 a simulation of a heating controller is performed for a room with a storage radiator and a direct acting heater. The results show that a neural network was able to learn the thermal response of the room for half-an-hour ahead and use this for model predictive control 24 hours ahead. Comparisons with other available tariffs show the neural controller is far superior in maintaining thermal comfort levels.

Chapter 7 gives the results of the first prototype controller in a real room where the storage heater set points are determined every 15 minutes for 5 hours ahead. The results are very encouraging and show the concept can work in real life.

Chapter 8 is a short discussion of the experiences gained throughout this research program, offering personal insights and thoughts on how industry can make the most of neural networks.

All the software used in the simulation work was programmed in Fortran 90 with the appendices containing edited versions of some of the code used. The code is included for reference purposes with the hope that it will be of benefit to those wishing to investigate neural networks and GAs.

## 1.4 Contribution of this Thesis

- 1) It is shown how a non-stationary process can be modelled by neural networks without first having to adjust the data. This method is used to extract the growth in base load.
- 2) It is shown how neural network pruning can be used to decompose the load into components relating to specific factors.
- 3) It is shown how a neural network can be used to formulate a simple rule based system for load forecasting.
- 4) A single neural network model for all half-hours of the year is created. No evidence has been found that this has been previously attempted.
- 5) Evidence of the gradual switching off of off-peak electric heating throughout April is identified.
- 6) Extensive empirical tests highlight how the mutation probability is critical for efficient GA performance in global optimisation. It is shown to be more important than crossover.
- 7) A variation of the random mutation hill climbing optimisation algorithm is introduced. This is termed *multiple random mutation hill climbing* and is more likely to escape from local minima during the search.
- 8) Using actual consumption data it was shown how cost savings of typically 40% could be made by optimising water heating based on real time pricing of electricity.
- 9) For the first reported time, a domestic electric storage heater was successfully operated under neural network control.

---

## 1.5 Publications from this Thesis

Work from chapter 3 has resulted in two conference papers,

P.D. Brierley and W.J. Batty, "Electric Load Modelling with Neural Networks: an Insight into the Black Box," *Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems*, Dunedin, November 1997, vol. 2, pp. 1326-1329. ISBN 981-3083-63-8.

P.D. Brierley and W.J. Batty, "Neural Data Mining and Modelling for Electric Load Prediction," in *Engineering Benefits from Neural Networks, Proceedings of the fourth International Conference on Engineering Applications of Neural Networks*, Gibraltar, June 1998, pp. 237-244. ISBN 951-97868-0-5.

Work from chapters 2 and 3 has been accepted for inclusion as a chapter of an edited book,

P.D. Brierley and W.J. Batty, "Data Mining with Neural Networks - an applied example in understanding electricity consumption patterns" - appearing as Chapter 12 in *Knowledge discovery and data mining: theory and practice*, edited by Professor Max Bramer, IEE publication, 1999.

Work based on chapter 5 has been accepted for publication in the journal *Electrical Power Systems Research*: "Genetic optimisation of domestic hot water supply based on real time pricing of electricity".

It is intended to submit further journal papers based on work from chapters 6 and 7.

## 1.6 Outcomes from this Thesis

As a direct result of the knowledge and experienced gained during this Engineering Doctorate research program, the author jointly formed NeuSolutions, which was incorporated as a limited company on 27th April 1999.

# NeuSolutions

NeuSolutions Ltd.

34 Pontamman Road

Ammanford

Carmarthenshire

Wales

SA18 2HX

Tel: 01269 592612

Web: <http://www.neusolutions.com>

Email: [info@neusolutions.com](mailto:info@neusolutions.com)

# 2

## **Feedforward Neural Networks**

---

### **2.1 What are Neural Networks?**

In practical terms, artificial neural networks are essentially very simple computer programs that can automatically find non-linear relationships/patterns in data without any pre-defined model form or domain knowledge. They are definitely not, as many people new to the subject quite often assume, complex mathematical systems requiring a super computer to operate. Several types of neural networks exist, of which the most popular is the feedforward multi-layer perceptron (MLP) and the only type used in this research. Fig 2-1 (on page 11) shows the structure of a typical MLP.

MLPs consist of an input layer, one or more hidden layers and an output layer. Data is fed into the input layer and transformed by weights and neurons as it flows through the

network. The network output is the resultant transformation that forms the relationship between the inputs (independent variables) and the output (dependent variable). In a feedforward network there are only weight connections in the forward direction. Networks with neurons whose outputs can feedback into themselves or other neurons in the same or previous layers are known as recurrent neural networks. Fully connected means all possible weight connections are present so strictly speaking there could be direct connections between the inputs and the output neuron.

MLPs are trained to find a relationship by presenting the network with historical values of inputs and outputs. Training is the search for a set of weights that best match the inputs onto the output for the examples (training patterns) in the historical database. Training the network is thus an optimisation problem where the optimal solution lies somewhere in weight space.

There are numerous methods by which this optimisation can be performed, such as random walks [5] or genetic searches [6,7], but the most popular are based on gradient descent techniques of which there are numerous variations that have evolved from the original back propagation algorithm or *generalised delta rule* (see appendix A).

## 2.2 Why use Neural Networks?

The most common application is to train a neural network on historical data and then use this model to predict the outcome for new combinations of inputs. The hope is that the network has extracted a general relationship that holds for all combinations of inputs. There are generally two main types of problem MLPs are used to model, classification and regression. In classification problems the output will have one of two values, representing 'belongs to the set' and 'does not belong to the set' whereas in regression problems the output is a continuous variable.

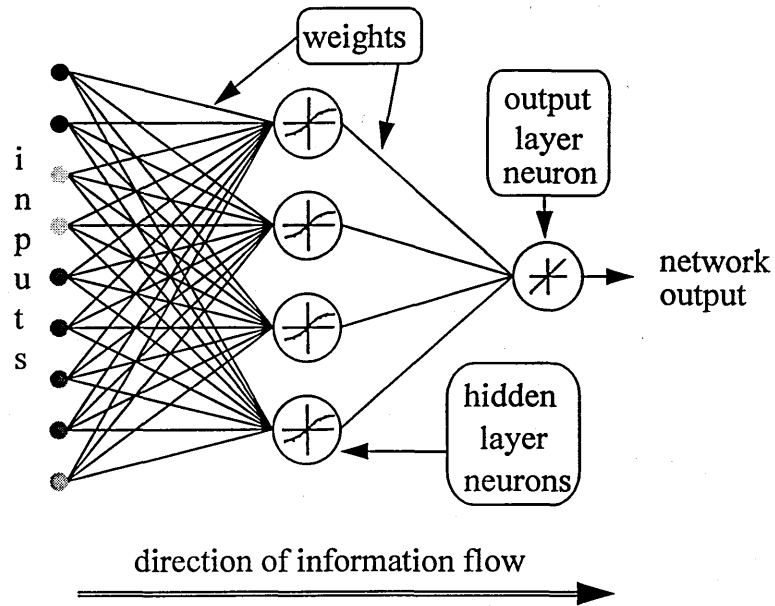


Fig 2-1 A fully connected feedforward multi-layer perceptron

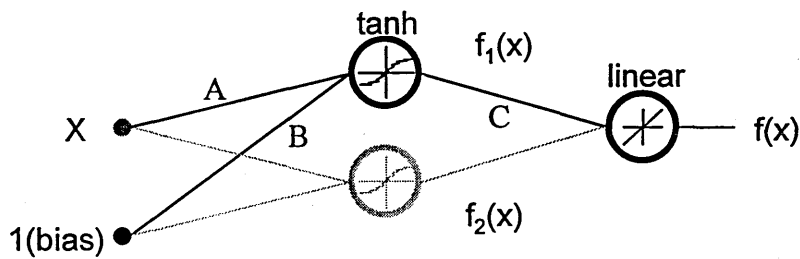


Fig 2-2 Number processing within a network

$$\begin{aligned} \text{Network output, } f(x) &= f_1(x) + f_2(x) \\ f_1(x) &= C \tanh(xA+B) \end{aligned}$$

## 2.3 How do Neural Networks Process Information?

Fig 2-2 (on page 11) shows a neural network with one input ( $x$ ), one hidden layer containing two neurons and one neuron in the output layer. An extra input known as a bias is created and has a constant value, which is 1 in this case. The bias has weight connections to all hidden neurons and there can also be a bias weight connection to the output neuron. Each input is connected to each hidden neuron by an associated weight. Each hidden neuron sums all the weighted inputs (the input multiplied by the connecting weight value, i.e.  $x_A + 1B$ ) that feed into the neuron and passes this value through an 'activation' or 'squashing' function. The result is then multiplied by another associated weight © and the network output is again the summation of all weighted inputs passed through the output neuron activation function.

Any function can act as the activation function but for gradient descent learning it must be continuously differentiable. Two popular sigmoidal shaped activation functions are the logistic and the hyperbolic tangent ( $\tanh$ ) as shown in Fig 2-3.

The non-linearity of these sigmoidal activation functions is what enables neural networks to solve non-linear problems. Fig 2-3 shows how they 'squash' the output between limits (0,1 logistic and  $-1,1$   $\tanh$ ). This is commonly used as an output activation function for classification problems as it acts as a decision boundary where a

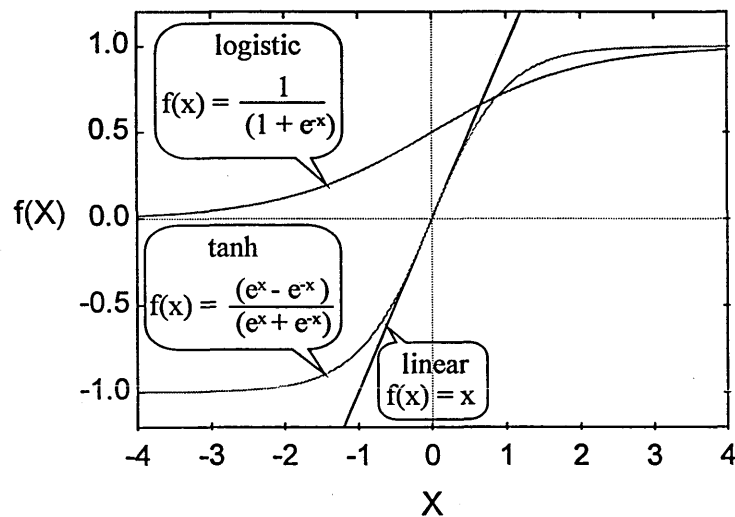


Fig 2-3 Common activation functions



---

continuous valued input is classified as 0 or 1 (in the case of the logistic) with a region of doubt for any value in between. Because of this, the required output of the network must be appropriately scaled to lie within the limits of the associated output activation function.

For regression problems, the output activation is often the 'identity' or 'linear' function (Fig 2-3), as there is no real gain in the output non-linearity because any further transformation could be achieved in previous layers. It can be seen how the sigmoidal functions have near linear regions, which enables them to be used to approximate linear problems. A network with no hidden layer and a linear output activation function becomes a linear regression model.

The output of the network in Fig 2-2 (on page 11), which has tanh and linear activation functions in the hidden and output layers respectively, is,

$$\text{Network output} = f_1(x) + f_2(x)$$

where,

$$f_1(x) = C \tanh(xA+B)$$

and weights A, B and C that provide a good solution must be found for each neuron.

Fig 2-4 (on page 14) shows how a linear input can easily be transformed into a non-linear output by mapping it onto sections of the tanh activation function. The importance of the bias input acting as a shift operator becomes evident.

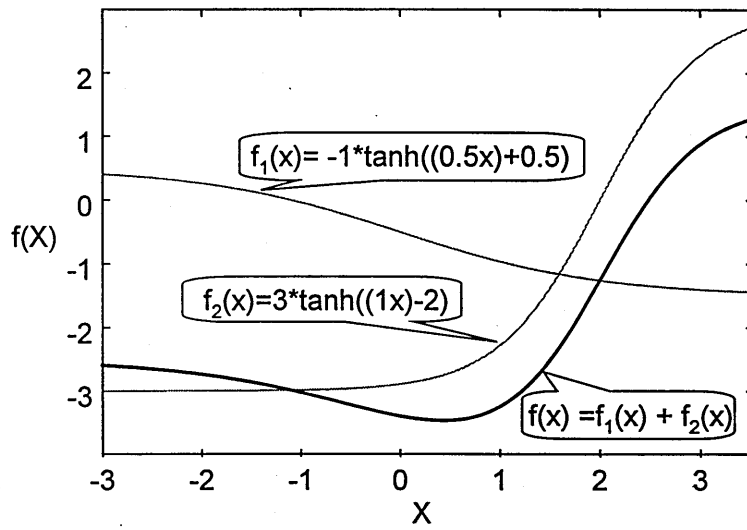


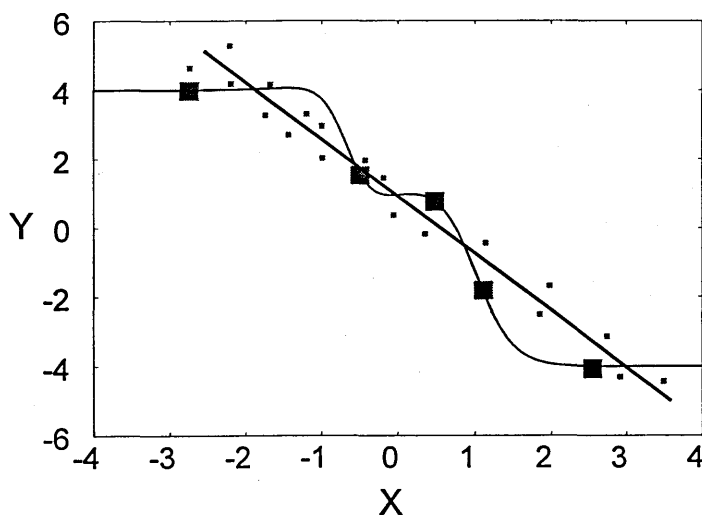
Fig 2-4 How two neurons can transform a linear input into a complex non-linear output

## 2.4 Things to be aware of....

### 2.4.1 Over-fitting and Generalisation

Given enough hidden neurons a neural network can map any function but there is the danger that the network has just learned to 'memorise' the data. Fig 2-5 shows 5 points that can be perfectly mapped by a neural network which it could be assumed has learned the relationship. This is not the case though, as is demonstrated when more data is shown and a noisy linear relationship becomes evident. This is what is known as 'over-fitting' the training data [8] and the generalisation properties of the network become unreliable.

To create a model with good generalisation properties the training data needs to be plentiful and the number of hidden neurons should be restricted. Training with sparse data and adding hidden neurons until there is no error can give a mistaken sense of achievement.



**Fig 2-5** *An apparently perfect model with poor generalisation properties. A neural curve can be fitted through the five larger squares but this relationship does not generalise for the remainder of the data.*

## 2.4.2 Extrapolation

Neural networks do not give an exact physical model but learn to represent the relationship in terms of the activation functions of the neurons. Neural network models cannot extrapolate with any usefulness outside the domain of experience of the training data [9]. This is not a flaw specific to neural models, as any model that is not based on first principles cannot extrapolate with confidence into the unknown.

Fig 2-6 demonstrates this for the function  $y=2x^2-1$ . Within the training range two hidden neurons can accurately mimic this polynomial but outside this range the neural representation does not hold.

The manner with which sigmoidal neurons become saturated outside their training range has potential for stable neuro-control applications. Fig 2-6 also demonstrates the power of neural networks for mapping polynomial functions.

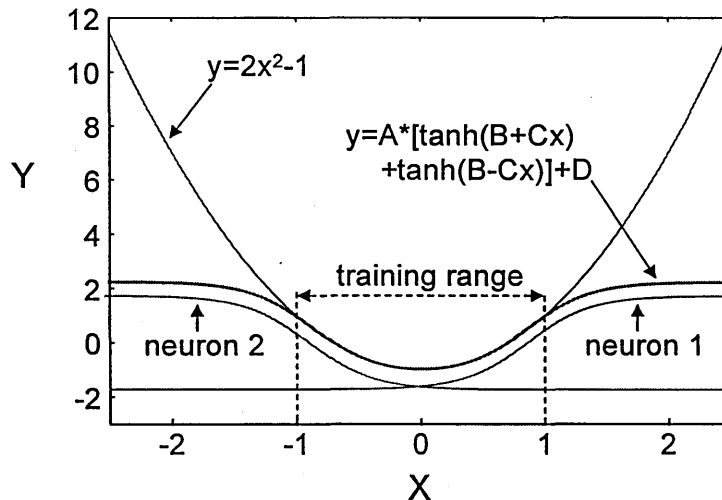


Fig 2-6 Neural networks cannot extrapolate

### 2.4.3 The Function being Minimised

Training a network involves the search for a set of weights that provides a good fit to the data, but what is a good solution? In order to gauge the 'fitness' of each potential solution some quantitative measure of the error must be made. The most common procedure is to search for the weight set that gives the minimum total squared errors [10,11] (or RMS error) over all the training cases, where the error is the difference between the network output and the required output. Other possibilities are the minimum total absolute error [12] (MAE or  $L_1$  norm) or the minimum total square root of the absolute error. Specifically,

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (Act_i - Pred_i)^2}{n}} \qquad MAE = \frac{\sum_{i=1}^n |Act_i - Pred_i|}{n}$$

where  $n$  is the number of training cases,  $Act$  is the actual value and  $Pred$  is the model value.

The choice made depends on the purpose of the model. Fig 2-7 (on page 18) demonstrates this for a neural network with one hidden layer neuron. Minimising the RMS error drives the solution towards what may be three outliers and the final solution is neither here nor there. By minimising the absolute error there is less importance given to these three cases and they become easily identifiable.

A third option would be to define an acceptable error tolerance and the 'fittest' solution would be that which models the most training cases within this tolerance. The line in Fig 2-7 passing close to all but three of the points would be the optimal solution in this case, given a small error tolerance.

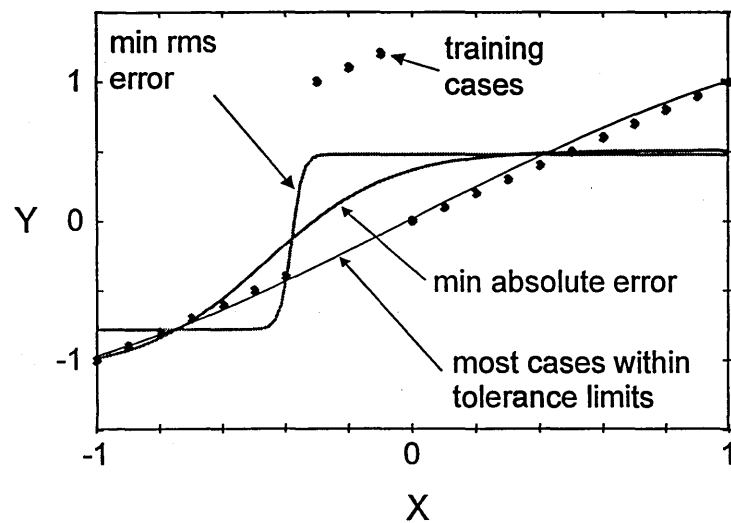


Fig 2-7 Defining the network fitness function

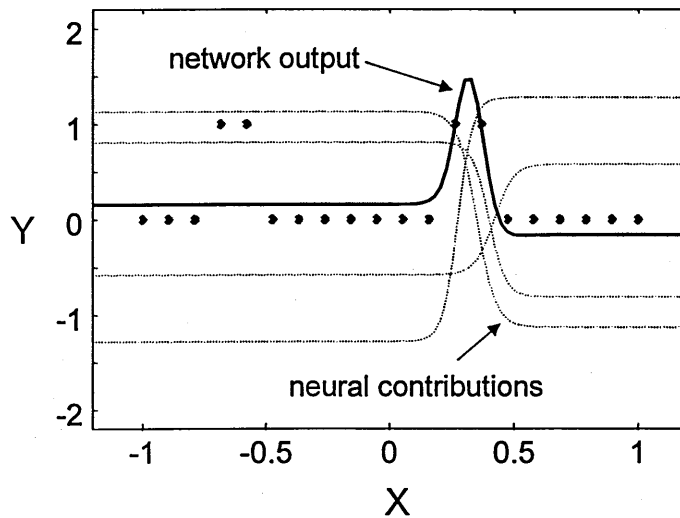
For data mining problems the purpose should be to identify those cases that stand out from the rest with the objective of trying to understand why this is so and hence learning about the data. It can thus be seen how the choice of fitness function can effect the nature of this task.

#### 2.4.4 Local Minima

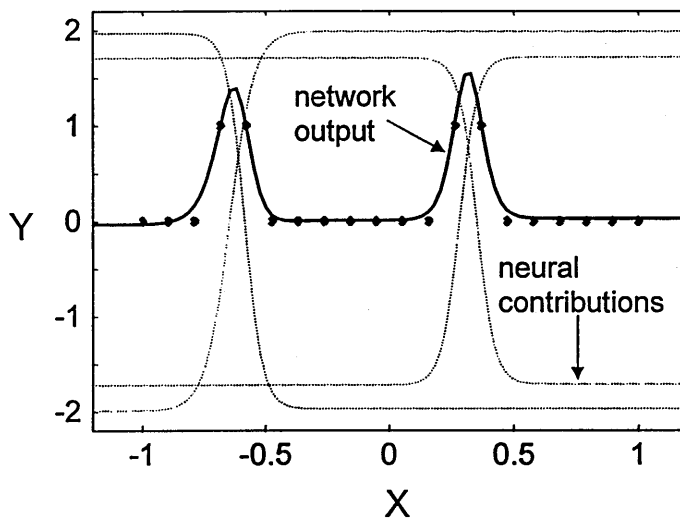
An analogy to training a neural network (searching for the best weight set) is a kangaroo jumping around a mountain range [13]. The kangaroo is searching for the lowest point in the range and does so by jumping around to see if the place he lands in is lower than the place he started from. If he cannot jump very far then he might get stuck in the bottom of a local valley when a lower point actually exists in the next valley. This is what is known as a local minimum solution but in order to find the global minimum he needs the power to jump over the ridge between the two valleys and land at a lower point.

Training a neural network is a similar search procedure and most training algorithms do not guarantee the global solution. Fig 2-8 a) shows a neural network trapped in a local

minimum. The required  $xy$  mapping is possible with four hidden neurons but the weights are such that small weight changes result in the overall 'fitness' function error increasing. Fig 2-8 b) shows the global solution that is possible but unlikely to develop from the situation in Fig 2-8 a).



a) local minimum



b) global minimum

Fig 2-8 Training algorithms can become trapped in sub-optimal solutions

## 2.4.5 Data Encoding

The ability of a neural network to extract relationships in data depends on how the data is encoded to represent the values or features of the inputs. The neural network is only as good as the information it is given and failure to perform satisfactorily is often not that of the neural network but that of the modeller in understanding how networks process the input data. Consider Fig 2-9 where a network with one hidden neuron is used to model the triangular  $xy$  mapping over the range  $x=0,360$ . Trying to map  $x$  directly onto  $y$  (linear encoding) with one hidden neuron is clearly impossible and would require at least two hidden neurons to get close. If the single input is encoded into two inputs that are the sine and cosine of  $x$  (in degrees) then a better result can be obtained but still only using one hidden neuron. These two encoding schemes will also give different extrapolation results outside the 0-360 range. The linear encoding will give constant values as the input will map onto the 'saturated' flat parts of the activation function. The sine, cosine encoding scheme will result in repetitions of the output as  $x$  increases in steps of 360.

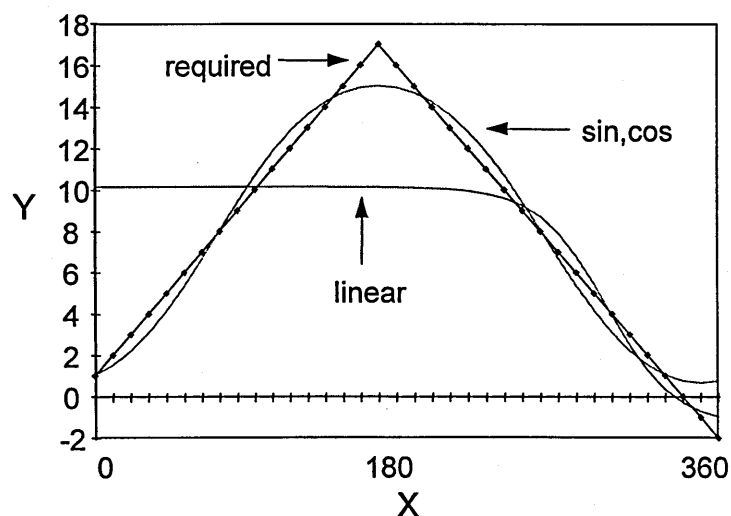


Fig 2-9 *Input encoding schemes giving different results for a single hidden neuron*



## 2.5 Chapter Summary

This brief introduction on MLPs has attempted to bring together and explain some of the most important aspects in neural network modelling. It has been outlined what a MLP is, what it can do, how it does it and what to be aware of.

Most of the points made become very obvious once it is understood how a MLP operates, an understanding which is required if the most is to be made of a neural network project. Successful use of neural networks is as much an art as a science with the art being in ensuring the MLP is given the best conditions for it to succeed.

It is a common misconception that because neural networks are often referred to as 'artificial intelligence' that they are some sort of sophisticated mathematical tool. In reality this is not the case, they are merely another modelling technique that are elegant in their simplicity.

Chapter 3 is a worked example of how a MLP was used to look at a real problem.

# 3

## Electric Load Modelling

---

### 3.1 The Data being Modelled

The purpose of this investigation is to understand what factors influence the electricity consumption of a region of the UK in order to create a robust model. Fig 3-1 to 3-3 (on page 23) show the data in question for the total daily load (throughout this work 'load' refers to the total amount of energy consumed in a given period). Fig 3-4 (on page 24) shows typical half-hourly profiles for a week in summer and winter.

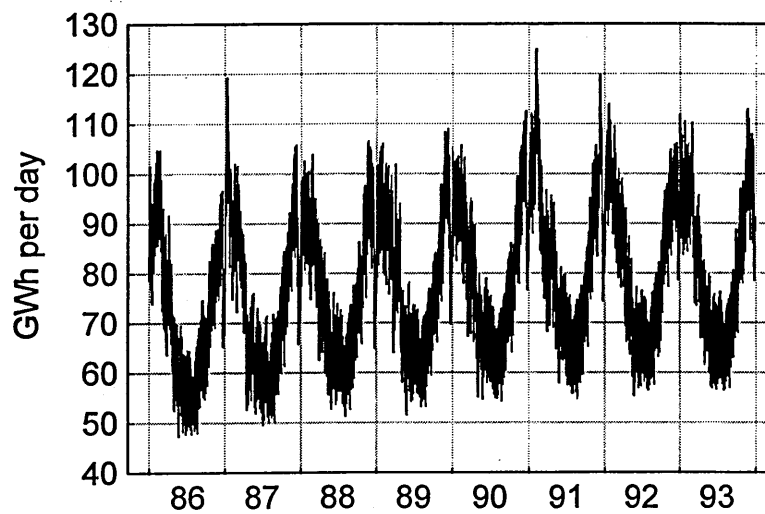


Fig 3-1 Total daily load over an eight year period

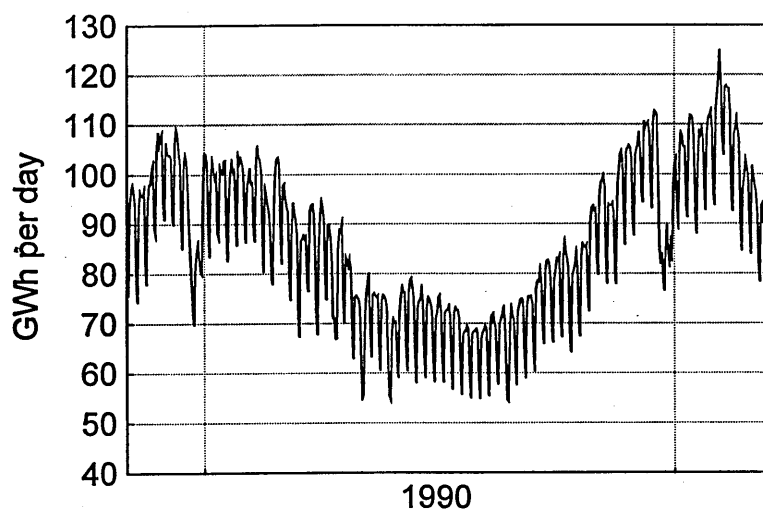


Fig 3-2 Total daily load over a one year period

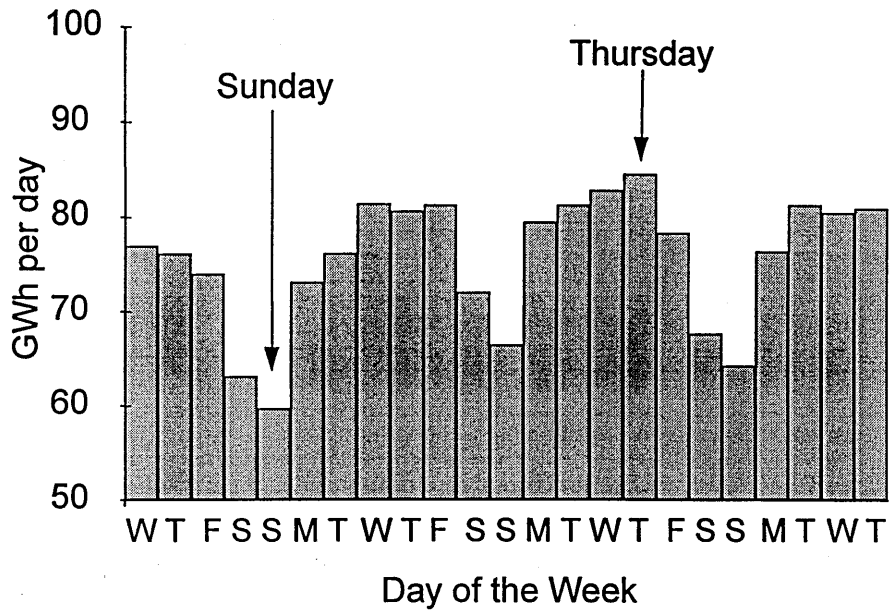


Fig 3-3 Total daily load over a three week period

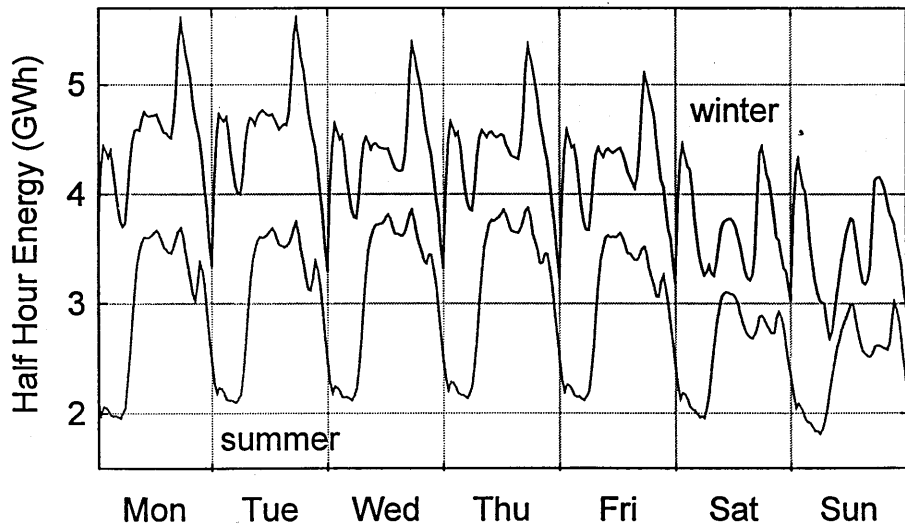


Fig 3-4 Half hourly profiles for summer and winter

Fig 3-1 shows that there is an annual cycle that peaks in the colder winters with the load in the warmer summers being almost half of this peak value. Fig 3-2 shows that there is also a weekly cycle with abnormal activity at Christmas. Fig 3-3 highlights this weekly cycle clearly showing that there is a reduced load demand at weekends. The difference between summer and winter half-hourly profiles is seen in Fig 3-4. In winter there is a large peak after midnight caused by an off-peak tariff that exists for electric storage heating, this peak being absent in summer. There is also a large evening peak in the winter whereas in the summer there are two smaller distinct evening peaks. The two summer peaks coincide with people returning from work (cooking) and dusk (when lights will be switched on). In the winter as the number of daylight hours reduces these two periods coincide to create one peak which is probably amplified through heating requirements.

Electricity consumption is a dynamic process depending on numerous independent and inter-related factors and a challenge for any modelling technique. Consumption patterns vary depending on the location as is shown in [14] where summer and winter weekly profiles are shown for several countries.

### **3.2 Why Forecast Electricity Demand?**

In the long term, future trends in electricity consumption need to be forecast so that an energy policy can be formulated to ensure there will be enough generation plant available in the future to meet the projected requirement. In the medium term demand needs to be known so that required resources such as coal can be stockpiled. Short term load forecasting involves predicting electricity consumption hours to days ahead and is required to ensure an adequate electricity supply that is generated in an economical manner. The nature of load profiles (Fig 3-4, on page 24) means that certain plant may only be required to generate electricity periodically throughout the day so the prediction is required to formulate an operating schedule. The total cost of generation, and hence price, includes start up and shut down costs, which vary depending on the nature of the

plant. For example, it is expensive to shut down nuclear reactors so their most economic operation is continuous generation. Coal powered steam turbines require several hours for preheating the boilers, so once fired they will want to operate for extended periods. Hydro-electric generators on the other hand can be fully operational in about 3 seconds, a feature that allows them to command a high price under certain conditions.

In England and Wales the wholesale trading of electricity is centrally controlled by the electricity Pool (see [15] for a detailed description of the UK electricity supply industry). On a daily basis generators bid in prices at which they are willing to supply electricity for each half-hour of the next day. In order to minimise costs, an optimised unit commitment schedule [16] is calculated based on forecast demand and the bid prices. The bids are stacked starting with the cheapest until the forecast demand is met and the actual traded price for all generators to distributors is based on this marginal bid price. Thus, in theory, nuclear plant are likely to bid a minimal amount to ensure that they are selected to supply for all hours, but the actual amount they receive will be based on the system marginal price. The forecast is generally required at least 24 hours in advance so that generators can be informed of their required schedule.

In the shorter term, forecasts seconds to minutes ahead are required to keep the frequency stable. This is particularly important during television commercial breaks when the switching on of kettles causes surges which have to be balanced by plant with fast reaction times.

With deregulation of the electricity industry, forecasting consumption is becoming more important at the utility level due to the complex nature of electricity trading. Reports on the potential energy savings that could be made by increasing the accuracy of the forecasts are limited [17,18,19,20].

### 3.3 Previous Work

It was noted [21] that the volume of published papers in electric load forecasting is cyclical, with initial great interest gradually declining. Checking the dates from a recent review paper [14] would indicate that the current cycle peaked around 1993. There is no doubt that this current wave of interest is due to load forecasting being developed as another application area for the pattern matching abilities of neural networks. The work appears to have been initiated in 1990 [22,23] and commercial load forecasting software is now available and operational [24,25].

Early papers laid the groundwork and more recent papers tend to report on the results of various modifications. The models usually group input data by individual day types or weekdays and weekends [26,27,28,29,30] and divide data into seasons [28,31] or shorter periods. Holidays are often assumed to resemble weekends [27,32] or the data is adjusted [33,34] or removed [26]. In the literature growth is seldom mentioned [34,35,36,37,38,39] as the time period being investigated is often deemed too short for it to make any difference to the model. The common approach is to adjust the data to eliminate its effect, often by some assumed linear growth rate. The reasons stated for such decisions are frequently based on prior knowledge about the load profiles and probably due to experience of linear modelling techniques.

Such assumptions based on human judgement are underestimating the capabilities of neural networks to indicate themselves how data should be modelled. Dividing up the data set reduces the amount of training information and inhibits a general analysis of long term trends. By including as inputs the relevant information why the data sets are different it is possible to create one model for all the data, this single model being easier to analyse and improve. There has been little reported work on improving and hence learning about systems by visualisation of trends in the model performance [40].

Many published papers show excellent results from systems tailored to individual requirements and generally satisfy the needs of the users in the sense that they are at least as good as previous non-linear models. It is hard to see how future improvements will

emerge if the technique is viewed as a black box that cannot explain how it did or did not arrive at a good solution.

In this work the approach is to use the neural network as a data-mining tool to try and discover exactly what influences the load profile. This is achieved by examining network errors and finding reasons why they are such. In section 3.5 the total daily load is examined over an eight year period to learn about general trends. Section 3.6 demonstrates problems with over-fitting and generalisation that can occur with an example from this real data. In section 3.7 the network is examined to discover just how the data is processed and a small set of rules is extracted that gives a reasonable model of the load over these eight years. Section 3.8 gives the results of a network used as a one day ahead predictor and compares errors to determine where improvements can be made. In section 3.9 the half-hourly load is examined over an eleven month period. Section 3.10 shows how improvements can be made by using populations of models.

### **3.4 Network Used**

A feed forward multi-layer perceptron trained by standard back propagation was used for the analysis. Details are given in the appendices. The fitness measure being evaluated was the minimum RMS error over all the training examples but it would have been a simple task to use any other error measure. Network performance is reported as the mean absolute percentage error (MAPE) as it is frequently used in similar work, although this was not the fitness function being minimised by the network training algorithm. The MAPE is only used as a yardstick to gauge model improvement and should never be used as a means of comparing various techniques on different data sets. For example, if a constant load was added to all the data (for instance if a factory with a constant demand was commissioned) then this would give the same model apart from the required constant term but the MAPE would be reduced. Different methods of reporting errors and comments are given in [14,21,41,42].



### 3.5 Total Daily Load Model

Fig 3-1 (on page 23) shows the data being investigated, the total daily load over eight years (1986-1993) for a region with approximately 10% of the total UK demand. The load for each day will be the output of the MLP (the dependent variable) and the inputs are what are required to be found. Thus the data-mining problem is to establish what the factors are that determine the electric load.

Fig 3-3 (on page 24) shows that there appears to be a weekly pattern so this is the initial clue used. There are two commonly used methods of encoding what day of the week it is. Seven inputs could be created with the day in question having a value of '1' and the remaining six days having a value of '0', a system known as 'flagging'. Alternatively the day could be transformed into an angle (in steps of  $2\pi/7$ ) and the sine and cosine of this angle applied as two inputs [43]. This second scheme was chosen, the consequences of which will become apparent later.

All 2,922 examples of the day/load relationship were used to train the neural network, the errors for which are shown in Fig 3-5 (on page 33). In Fig 3-5 to 3-14 the y-axis is the daily error in GWh and the caption identifies the added input and MAPE achieved.

An annual cycle in the errors is clearly evident from Fig 3-5, with overestimates in the summer and underestimates in the winter. Weather patterns follow an annual cycle so the average daily temperature was included as the second input, the errors for this network being shown in Fig 3-6.

Including the average temperature significantly reduced the error but an annual cycle is still evident. The length of daylight is another factor that varies seasonally which would affect electricity demand due to lighting requirements. In the UK during the summer month of June sunset is around 8.30 p.m. whereas in December it is at 4 p.m. As light levels were not available a continuous term was required as an input that had an annual cycle. A simple number (1-365) is not sufficient for this as it places adjacent days (31st Dec., 1st Jan.) at opposite extremes. Taking the sine and cosine of the day of the year

angle produces the required cyclical seasonal term. Fig 3-7 shows how this has extracted the seasonal variation.

What is evident now are bank holiday Mondays and Good Friday. There are generally six bank holidays each year which are public holidays, as is Good Friday. A flag was created for these days, excluding those in the Christmas period, which is evidently a more complex time and dealt with separately.

Fig 3-8 shows the errors when a flag to indicate the bank holiday is included as an input and a slow trend over the eight year period becomes clearer. This can be explained by growth for which a linearly increasing term (1-2,922) was included to represent this feature.

Fig 3-9 and 3-10 highlight the effect of the number of hidden neurons. In all previous cases and that of Fig 3-9 only 5 hidden neurons were used. Fig 3-10 has the same inputs as Fig 3-9 but with 10 hidden neurons. The overall error is significantly reduced but it is clear that this reduction is due to efforts by the extra neurons to improve the Christmas errors. This is a consequence of minimising the RMS error where outliers are given more significance. From a data mining approach the network with only 5 hidden neurons is more informative.

In order to further examine the Christmas effect the errors were averaged over the eight years on a date basis, as shown in Fig 3-15 (on page 35). With 5 hidden neurons it can be seen how the Christmas error is reduced by lowering the whole period from mid November to early February by manipulating the seasonal inputs to show a reduced load over this period. With 10 hidden neurons there is more resolution available and the Christmas error can be significantly lowered with less interference in the adjoining days. The Christmas period is obviously special but training with this data without any inputs to represent the feature causes distortion of the network. The third case shown in Fig 3-15 used 20 hidden neurons but only trained on patterns where the initial error was below the overall RMS error. Even with more hidden neurons there is little distortion and the residuals are clearly identified because they are never allowed to affect the training of the network. With this technique all patterns should first of all be presented

for training before selective training begins to give them the opportunity to 'stake their claim'.

The errors over the Christmas period were significant from the 22nd December to 4th January with peaks on the 26th December and the 1st January. Inputs were created for these dates where the magnitude of the input was related to the magnitude of the average error. These inputs can be considered to be the 'degree of membership' of Christmas, with the 26th December having a membership value of 1. The errors of the trained network are shown in Fig 3-11 (on page 34).

Fig 3-11 shows that there is residual on the 15th January 1987, highlighted in Fig 3-16 (on page 35). Investigation showed that this error corresponded to a sudden cold front where the average temperature dropped to -10 degrees Celsius on that day. The model error indicates that more electricity is being consumed than would normally be expected. This could be expected, as off-peak heating systems will not anticipate the sudden drop and ancillary heating devices will be used to make up the required balance. Further examination revealed that other periods experiencing sudden changes in temperature resulted in large model errors. A sharp drop in temperature in the middle of summer gave load predictions higher than actually experienced. This is because the uncharacteristically cold weather would normally warrant a higher daily load, but people generally react to weather as opposed to anticipating change, thus there will always be a time delay. Conversely, sudden hot spells resulted in underestimations and there was a distinction between short-term gradual changes and sudden changes. The loads on very windy days were constantly underestimated probably because the wind chill factor can have the effect of reducing the apparent ambient temperature and buildings can become draughty, requiring more heating. Days following windy days were also underestimated, due again to the reactive nature of the system response. People feel the wind chill and adjust heating systems accordingly, requiring several days to restore normality.

To account for these findings maximum, minimum and average temperature and wind speed values were included for the day in question and the three previous days. The

error was significantly reduced by these additional inputs as shown in Fig 3-12. For the obvious residual date now evident (October 16th 1987) it was found that:

*'On the 16th a violent storm with heavy rain brought chaos to southern England. The winds were probably the strongest for 250 years. Millions of trees were uprooted or broken, many crashed into power lines....Some large areas were without electricity for up to a fortnight.....a mean hourly wind of 72 knots was recorded just before a power failure stopped the recorder.'*

*(Whittaker's Almanac 1989)*

Examining the dates giving the largest errors in Fig 3-12 revealed certain clusters. These are shown in Fig 3-17 (on page 36) where the errors are averaged over the eight years. Although the exact dates are not the same every year, four clusters of consistent overestimates are evident. Three of these were identified as Easter bank holiday weekend, August bank holiday weekend and the week of the bank holiday in late May (Whitsuntide, traditionally a school half-term holiday). The remaining anomaly was late July to early August which is when most schools have extended summer breaks.

Flags were created for the bank holiday weekends and Whitsuntide and a rising and falling linear term for the summer period, mimicking the nature of the error in Fig 3-17. Fig 3-13 (on page 34) shows the result of this trained network.

A causal model has been created that describes the load as a function of known events but there is obviously a limit to what information is available for a complete model of this type. Recent loads will contain information about local events that is captured in the load value and cannot be extracted otherwise. The final model (Fig 3-14, on page 34) shows this by also including yesterday's load and the load on the same day the previous week as inputs giving a final MAPE of 1.06 over all the eight years. Caution should be used here as generally the errors are reduced but days or weeks after certain special days can give increased errors.

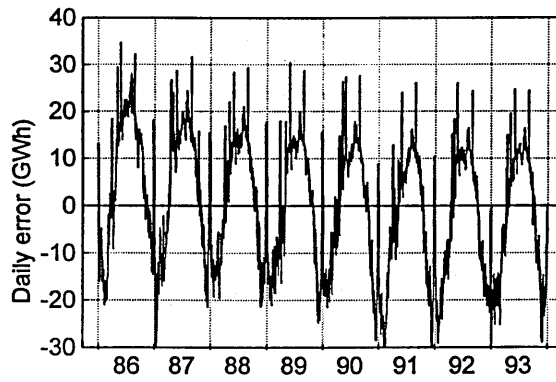


Fig 3-5 *Day of week*  $MAPE=15.00$

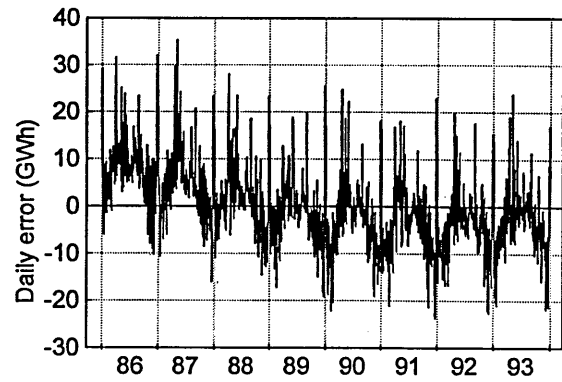


Fig 3-6 *Average temperature*  $MAPE=7.31$

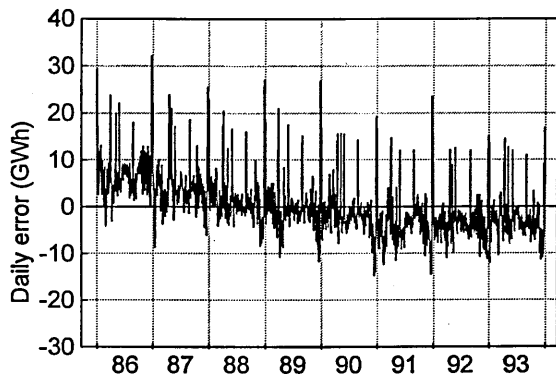


Fig 3-7 *Seasonal term*  $MAPE=5.28$

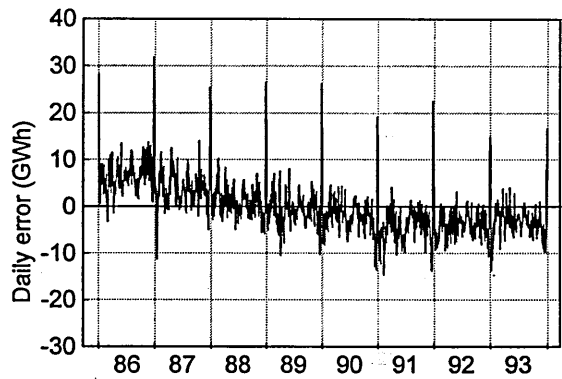


Fig 3-8 *Bank holidays*  $MAPE=5.00$

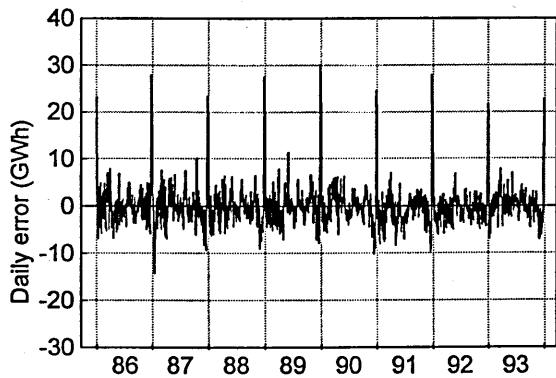


Fig 3-9 *Growth term (5HID)*  $MAPE=3.10$

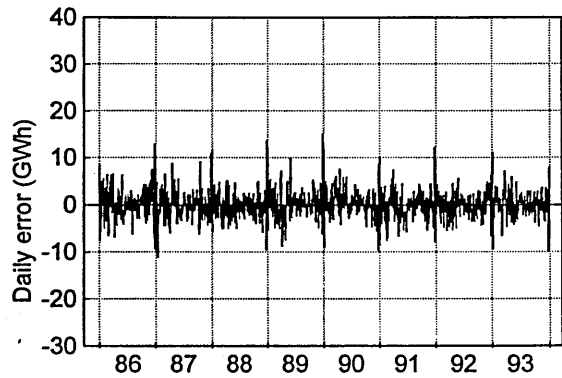


Fig 3-10 *Growth term (10HID)*  $MAPE=2.35$

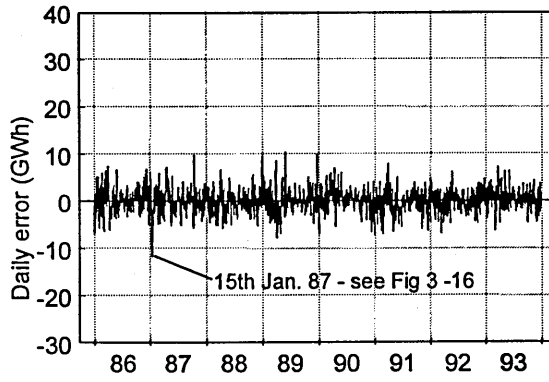


Fig 3-11 *Christmas* MAPE=2.18

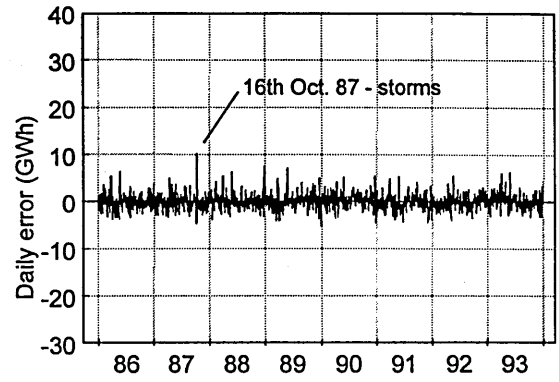


Fig 3-12 *Past weather* MAPE=1.46

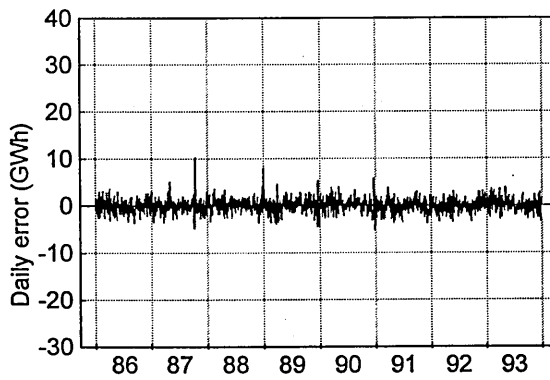


Fig 3-13 *Other holiday effects* MAPE=1.25

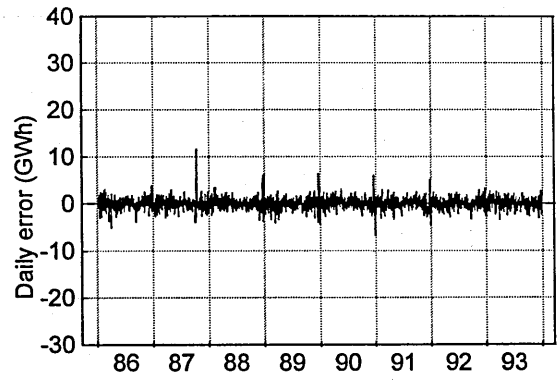


Fig 3-14 *Previous loads* MAPE=1.06

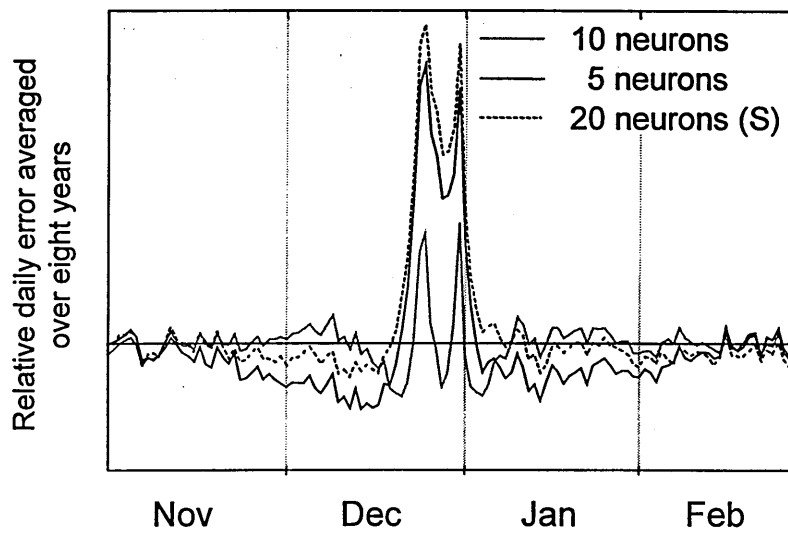


Fig 3-15 *Averaged Christmas errors*

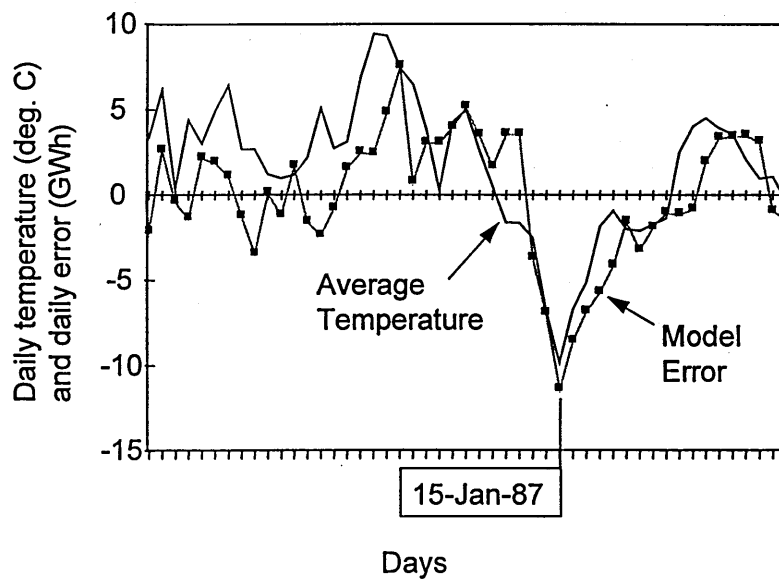


Fig 3-16 *The effect on the error resulting from a sudden cold front*

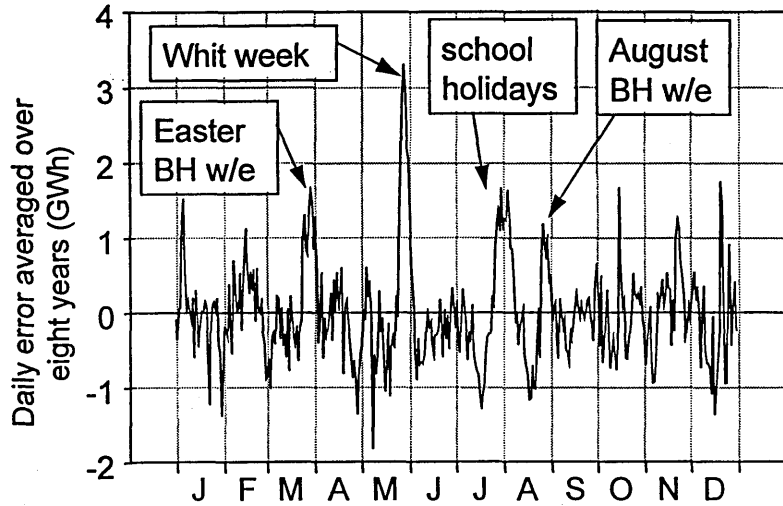


Fig 3-17 Averaged annual errors

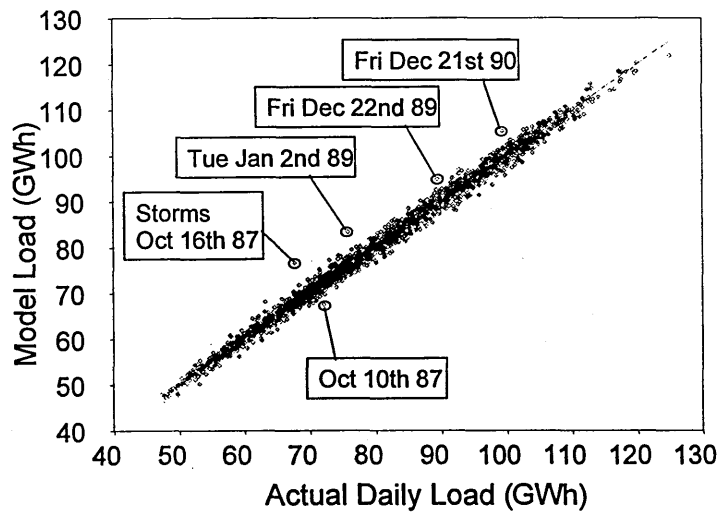


Fig 3-18 Actual v modelled loads

Fig 3-18 was created from the model of Fig 3-13 (on page 34) and shows several residuals that can be easily explained. The underestimate on the 10th October could easily be a result of the network trying to reduce the error for the storms on the 17th October, one week later. The 2nd January '89 was a Tuesday but also a bank holiday. Generally New Year's day is the bank holiday or the Monday if the 1st January falls at the weekend.



Because the 1st January was a Monday an extra day's holiday was given. Friday 22nd December '89 and 21st December '90 are special because they are the last full weekdays before Christmas Eve. The 21st December was not included in the Christmas period but in 1990 it was obviously taken as a holiday due to the fact that Christmas fell the following Tuesday (when else would the office party be held?). This was the only example of Christmas falling on a Tuesday in the eight cases.

### 3.6 Over-fitting and Generalisation

It is important that a trained neural network has the ability to generalise and not learn specific information in a 'photographic memory' type of manner. It must extract general relationships between data rather than specific relationships relating only to explicit data. The ability to learn specific information is known as over-fitting and can become a problem if too many hidden neurons are used, as illustrated by Fig 2-5 (on page 15).

Fig 3-19 (on page 38) illustrates a network that can learn to fit an erroneous datum into a model. The data is the total daily load values for a particular year of the model already created with one value being significantly changed to represent a residual or erroneous reading. For a perfect model all the points will lie on a straight line, the situation where the network correctly repeats all the loads. With 3 hidden neurons the erroneous point is easily identified as standing away from this line. When an extra neuron is added the network can fit the spurious point on this line without much distortion to the remaining points.

Fig 3-20 (on page 38) shows how the neural network has done this. A set of weights has been found that activate the extra neuron for the inputs of the erroneous day. For most other days the combination of weights and inputs are below the operational threshold of the activation function of this extra neuron (in other words it is saturated). This neuron is used to provide the additional load required to account for the error. For days around the erroneous day (day 14) this neuron is also active, inducing errors and thus poor generalisation to days that have similar input patterns.

This is an extreme example as the induced error was rather large but it demonstrates that there is more gain in dedicating a neuron to correcting a single day than reducing the errors for the remaining 364 days. Without careful examination extreme erroneous readings could quite easily go unnoticed and liberally adding more hidden neurons does not necessarily lead to a better model.

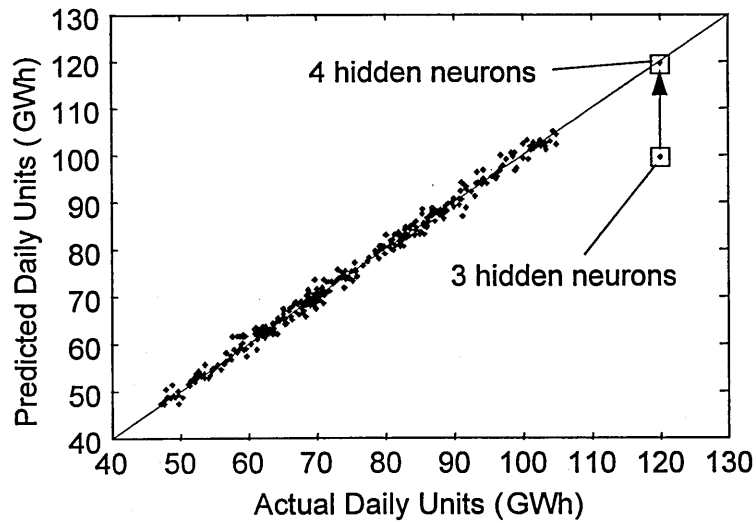


Fig 3-19 *Over-learning by the addition of an extra neuron*

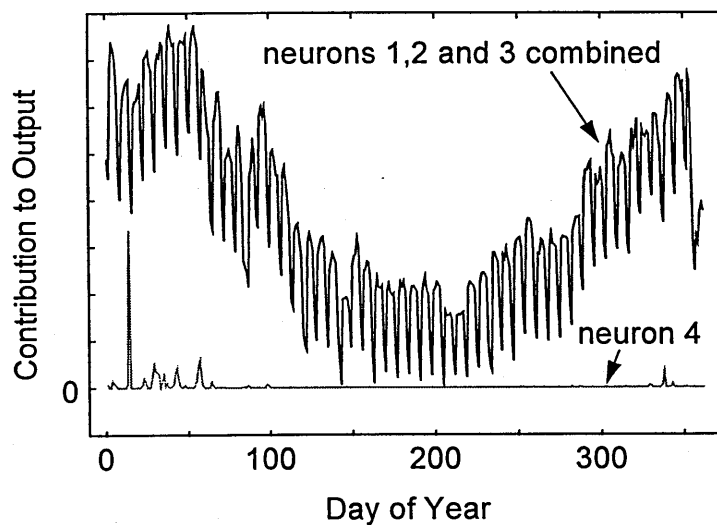


Fig 3-20 *How the extra neuron enables over-learning*

Three networks with 3, 8 and 15 hidden neurons were trained on the same year's data giving MAPEs of 1.09%, 0.46% and 0.59% respectively. On this evidence the network with eight neurons appears superior. All weather inputs in this data were then replaced with the average days weather (the weather for each day of the year being the same) and passed through the trained networks. Fig 3-21 shows that for certain days the two larger networks do not give realistic predictions. The apparently most reliable network for generalisation being the one with three hidden neurons although it gave the largest MAPE.

What has happened is that specific groups of neurons are only active for local features in the data. This can be clearly seen for the network with 15 hidden neurons which during learning has associated the bank holidays with the actual weather patterns in order to give a good model fit. When the actual weather is replaced by the average weather the neurons cannot give a general solution. With only three hidden neurons all data is processed by the same core neurons which experience the full range of values and have the ability to generalise. This is an example of how the quest for a minimum error can have its pitfalls.

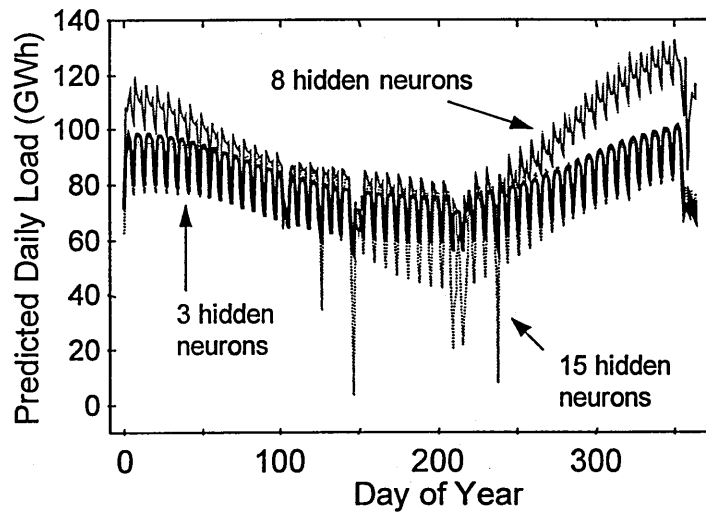


Fig 3-21 Generalisation properties of networks with 3, 8 and 15 neurons in the hidden layers. The network with the fewest hidden neurons gives the most realistic overall generalisation

### 3.7 Rule Extraction

The model created in section 3.5 was arrived at by deducing reasons for the model errors. As these reasons were identified and encoded as network inputs, improvements were seen. The question now becomes 'how is the neural network processing the data?' We know the inputs make sense and that a neural network is a very simple number processing machine, so an explanation should be simple to find. Without explanations there will be continued distrust of neural networks.

Consider the neural network in Fig 2-1 (on page 11). The network output is simply the summation of a number of terms originating from each hidden neuron, which in turn have formed connections of varying strength to the inputs. This basic idea makes neural networks a powerful tool for analysing relationships in data. In its simplest form a neural network with two hidden neurons and a linear output neuron is decomposing the output into two components, the nature of which were investigated for the causal inputs identified (i.e. not including past loads).

It was found that one of the neurons was acting as a filter in that it only appeared to be processing day of the week information. This was obvious when the two components were viewed and confirmed by inspecting the weights feeding into these two neurons, with those weights connecting the day inputs with one of the hidden neurons being much larger than the remaining weights.

A new network was created with three hidden neurons, one neuron with weights removed or 'pruned' so that it only processed 'day' inputs with the remaining two being fully connected. In a similar manner to before, one of the fully connected neurons acted as a filter in that the seasonal components feeding into this neuron were dominant. This process was continued and eventually all the inputs had been isolated and a network similar to that in Fig 3-22 (on page 41) was created.

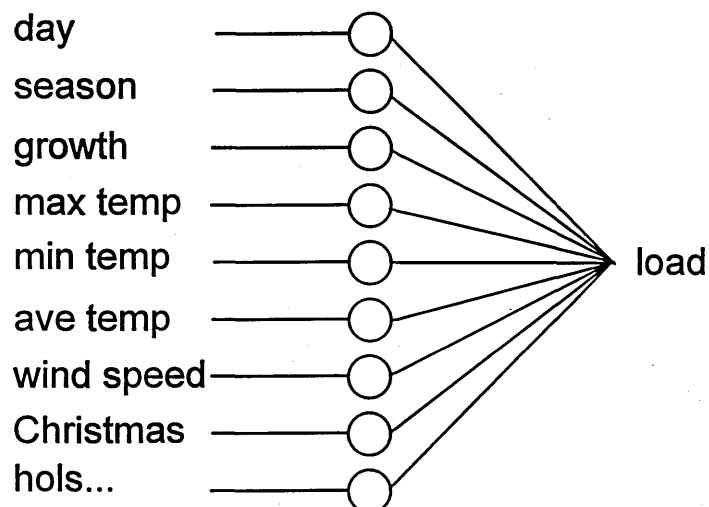


Fig 3-22 Schematic of the 'pruned' network

### 3.7.1 Day of the Week

The inclusion of a single neuron dealing only with day of the week inputs gave disproportionately large errors for Mondays. Examination of the output of this neuron revealed that it was operating close to one extreme (+1) of the tanh activation function for weekdays and close to the other extreme (-1) for Sundays. The addition of a second, third and fourth 'dedicated' neuron improved the overall errors and reduced the errors for Mondays to the same level as for all other days.

Fig 3-23 (on page 42) shows how the final contribution for each day is the summation of the outputs of the four hidden 'day' neurons. One neuron gets close to the solution but the extra neurons are required to increase the resolution. The manner in which the day of the week input was encoded is directly related to the number of hidden neurons required. Fortunately, the daily load requirement follows a cyclical pattern over the week, which is suited to the sine-cosine input encoding. If, for example, Wednesday afternoon was a public holiday, the load would drop on Wednesday and more neurons would be required to model this effect as the continuity of the cycle would be broken. It would appear that a better encoding scheme for general cases is a continuous weekly

cycle is not evident would be seven inputs as described previously which would only require one hidden neuron.

Fig 3-24 shows the first rule, the contribution to the load based solely on the day of the week.

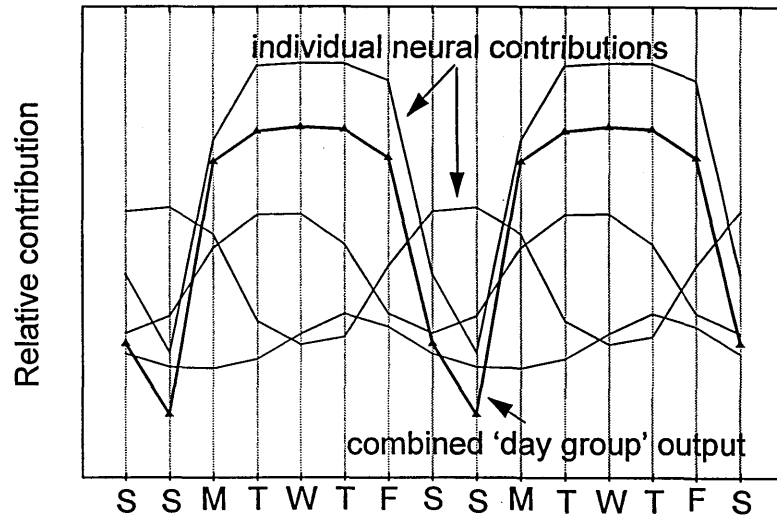


Fig 3-23 Four hidden neurons and their combined contribution to the network output

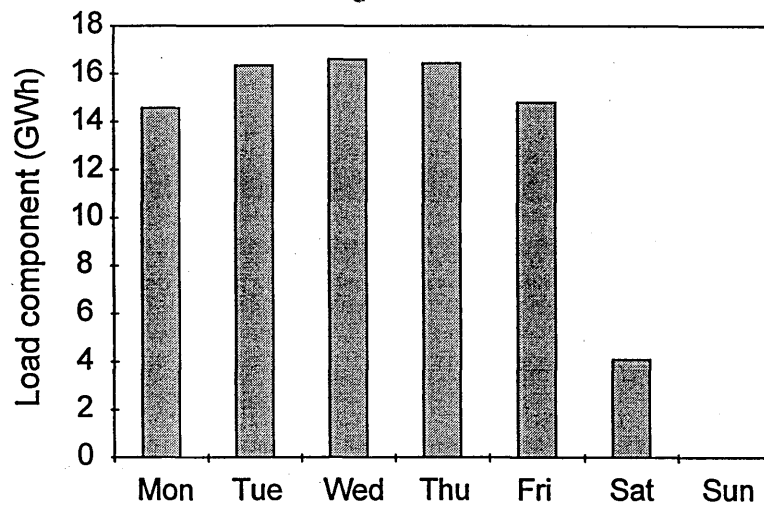


Fig 3-24 Rule 1 - what day of the week is it?

### 3.7.2 Time of Year

The seasonal component of the load with the summer correction included is shown in Fig 3-25. Fig 3-26 (on page 44) shows how this was formed by the summation of four hidden neuron outputs. The sine, cosine encoding is the only practical method of representing this cyclical term as a network input.

The variation between summer and winter is similar in magnitude to the difference between Sunday and Thursday. The significance of the school holiday effect is also clearly evident. The overall nature of the curve closely resembles the patterns of sunset and sunrise times, with the nights drawing in quickly from September to December characterised by the steepness of the slope. From January to May the nights gradually get lighter as the sun re-enters the northern hemisphere. Fig 3-25 is the second rule.

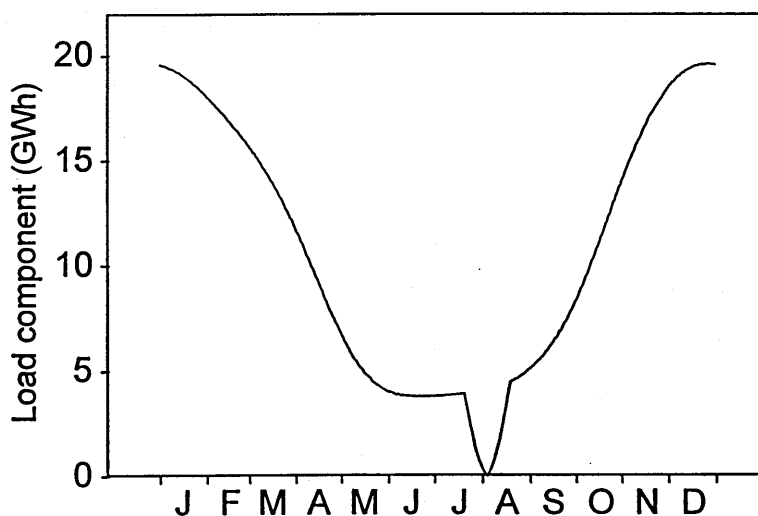


Fig 3-25 Rule 2- what day of the year is it?

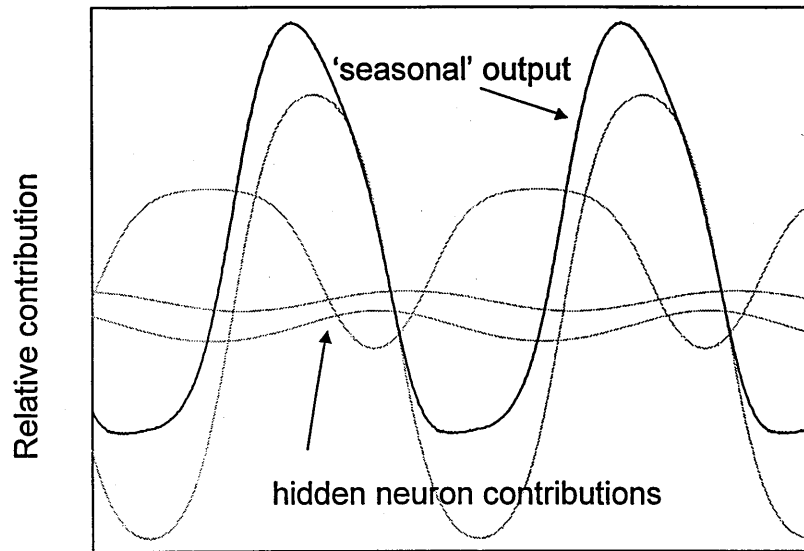


Fig 3-26 How the seasonal term is formed by the summation of four curves

### 3.7.3 Growth

The linear input included to represent growth is transformed by four hidden neurons as shown in Fig 3-27 (on page 45). The nature of the curve follows the economic climate of the time with the UK recession starting in 1991 clearly reflected by a reduction in the demand for electricity.

A second method of calculating the growth was performed using weather correction. A network was trained for each individual year and then the weather inputs were replaced with the averaged eight year weather for each day. If the neural network generalises well this gives a weather corrected load or the load that would have occurred if the weather had been 'average'. The weather corrected average annual daily load was then calculated for each year and assumed to occur at the median day of each year. Eight points resulted which were then curve fitted with a linear input neural network which was used to give a value for each day over the eight years. This weather corrected load resembled that of Fig 3-27.



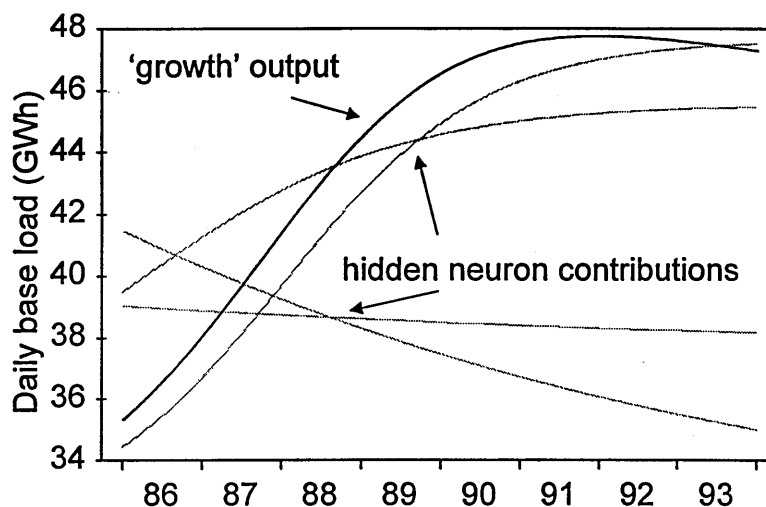


Fig 3-27 Rule 3 - what is the base load?

### 3.7.4 Weather Factors

By introducing a single dedicated neuron to process specific weather inputs as opposed to grouping all weather related inputs together, it was found that the increase in error was negligible. Furthermore, weight examination over several tests revealed consistencies in weight ratios for the time lagged (previous day's) weather inputs (Table 3-1 on page 46). Generally the lag coefficients are as expected, the importance declining with time. Average temperature is different with yesterday's average temperature being slightly more important than today's, which is not unexpected, as today's average is not finalised until the end of the day when most electricity has already been consumed. What is surprising is that the average temperature two days ago is insignificant while that of three days ago is relatively important.

**Table 3-1** *The relative importance of past weather conditions*

	T (today)	T-1 (yesterday)	T-2	T-3
max temp	1	0.59	0.43	0.29
min temp	1	0.19	-	-
ave temp	1	1.08	-	0.63
wind speed	1	0.37	0.13	0.12

Because of the consistencies in weight values the data can be pre-processed to create a single valued input for each weather variable. Fig 3-28 (on page 47) shows these factors and the contribution to the load, where the temperature factors are in degrees and the wind speed in knots and are calculated from the coefficients in Table 3-1. The non-linearity of these curves illustrate the effect of the non-linear regression capabilities of neural networks.

The gradient of the maximum and average temperature curves imply that as the weather gets warmer, electricity consumption decreases. The minimum temperature curve gradient is opposite indicating that there is a load component that increases with increasing temperatures. This could be the effect of the limited air conditioning or refrigeration. The wind speed load component increases as it gets windier. Fig 3-29 (on page 47) shows the weather components of the load over the eight years.

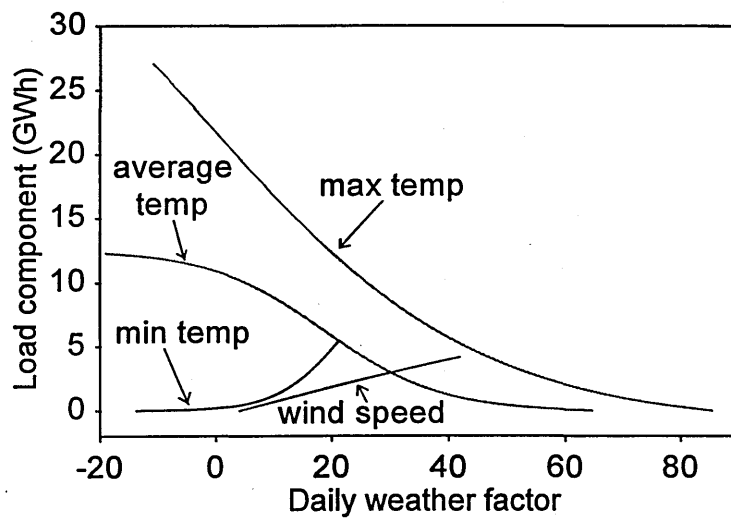


Fig 3-28 Rule 4 - what is the weather like?

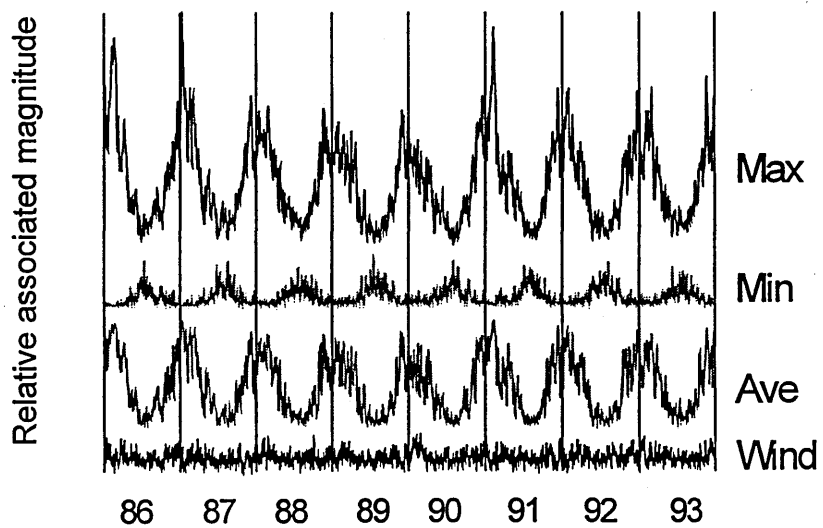


Fig 3-29 The weather components of the load

### 3.7.5 Holidays

Six holiday neurons were created for summer, Christmas, bank holidays and the special days around the bank holidays. No day could be a member of more than one holiday type. Fig 3-30 and Fig 3-31 show the corrections deduced by the neural network.

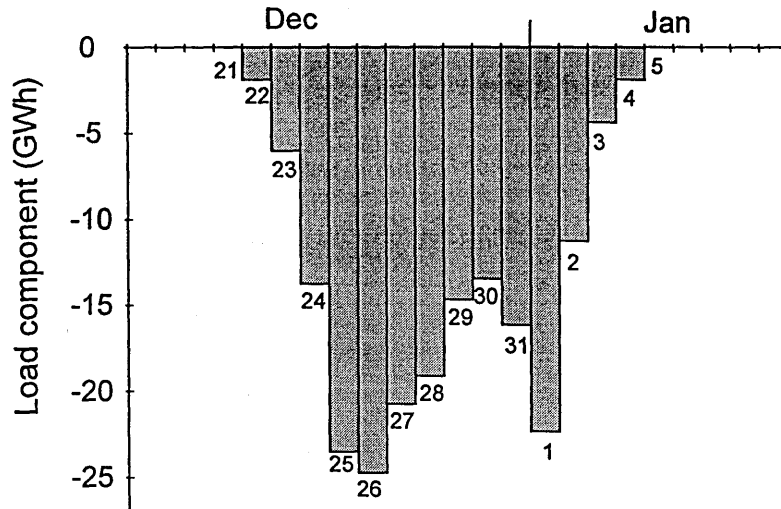


Fig 3-30 Christmas corrections

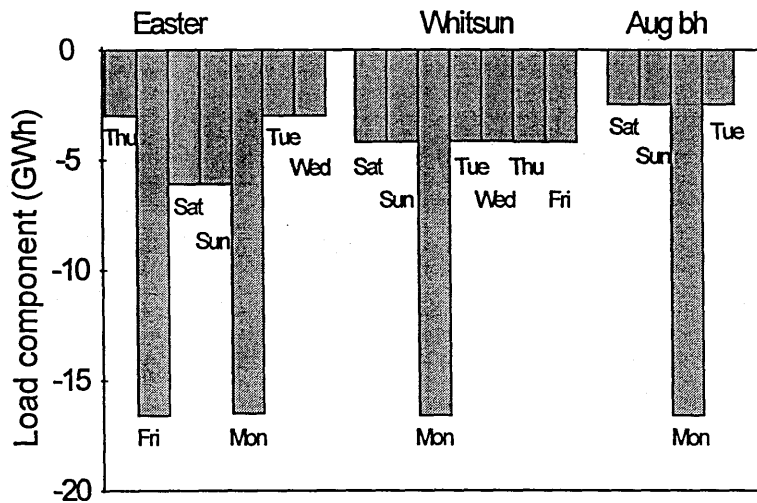


Fig 3-31 Holiday corrections

---

The Christmas corrections were based on observed errors from Fig 3-15 (on page 35) but improvements could be made after the anomalous days around Christmas (Fig 3-18 on page 36) had been identified, as they will distort the average errors. This exemplifies how adding certain information reveals more detailed information that allows refinement of the initial information - a circular process.

For the Easter weekend, 'shoulder' days were identified from the data that indicate an extended holiday period. This fits with observed behaviour as people take extra days as holiday to make the most of the two public holidays. The weekend was observed to have more of a holiday effect than the shoulder days which was reflected in the input encoding. Whitsun week and weekend were all classified as the same holiday group and thus will all have identical corrections.

In a fully connected network relationships with day of the week will also be formed, which will add to the accuracy. This is an example of why this drastically pruned network will not be as good as a fully connected one, as it cannot extract features involving more than one input type. Only independent factors were used in the creation of this model, mainly to keep things simple and graphically observable. Inter-related factors were tested, the most important being relationships between day of the week and day of the year, and day of the year and growth. Fig 3-32 shows the actual load and the errors with the rule based predictions. Fig 3-33 shows the change in the cumulative error distribution caused by pruning the network.

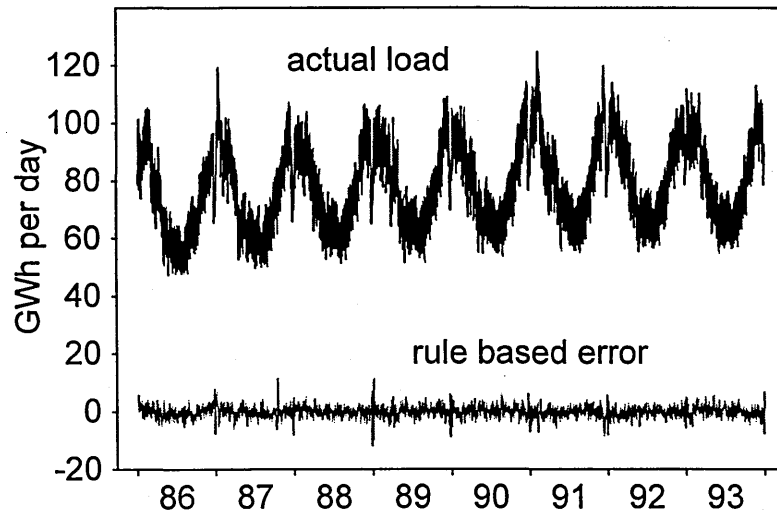


Fig 3-32 Rule based model

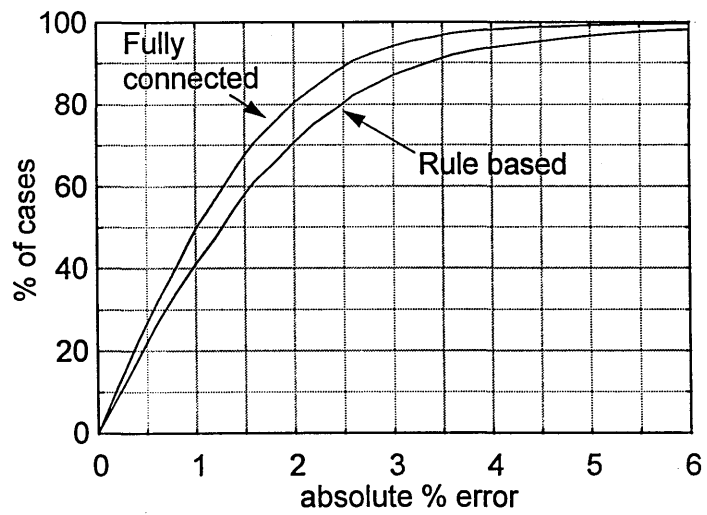


Fig 3-33 Cumulative percentage error distributions

### 3.8 Model Comparisons

The fully connected model was used to give next day predictions using the previous 800 days as a training set, with the weather for the prediction day being used retrospectively. A training set of 800 days was used to capture at least two years of examples for periods such as Christmas. An initial network was trained and used to predict the load for the next day's previously unseen data. This data with the actual observed load was then added to the training set with the distant most day being removed. For each prediction the weights from the previous day's network were used as a starting point and 10 epochs of training were allowed (an epoch is where all the patterns are presented once).

A network with 10 hidden neurons and inputs including yesterday's and last week's load gave a MAPE of 1.3% over the 5.8 years of predictions. Fig 3-34 shows the errors translated into an equivalent continuous error throughout the day. In context 60MW represents approximately 20 Watts per customer (not person) for the region (a light bulb is typically 60-100 Watts). Table 3-2 (on page 52) shows the errors analysed by day type for this prediction model, the fully connected model and the rule based model.

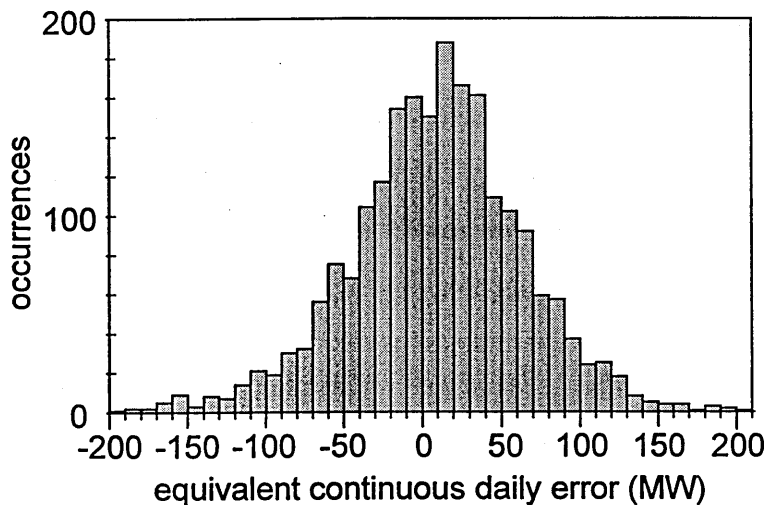


Fig 3-34 *Error distribution for one day ahead predictions*

**Table 3-2 Model errors by day type – mean absolute error (MAE) is given as an equivalent continuous daily error (MW)**

	fully connected	rule based	prediction model	
	MAE	MAE	MAE	MAPE
Christmas	60	144	85	2.38
bank holidays	51	86	54	2.06
summer	28	47	31	1.13
non special weekends	41	51	39	1.26
non special weekdays	42	48	45	1.22

Christmas is the worst period to model but this would be expected as there is a limited number of examples available for training. The rule based and prediction model performances are relatively poor which is not unexpected as there is no information relating the date to the day of the week for the rule based model and there are only two years of examples for the prediction model.

Bank holidays were generally underestimated in the prediction model but the overall MAE was comparable to the fully connected model. The period classified as summer holidays gave the best prediction results and were again close to the fully connected errors.

The errors for all non-special days would indicate that weekdays gave better results than weekends if the MAPE was used as the indicator, as reported in [44]. This is not the case though when the MAE is examined, a result that is due to the reduced weekend load. This highlights the fact that reporting percentages, although common practice, is really meaningless in electric load forecasting.



This error analysis would indicate that for prediction purposes improvements would result for the non-special days if those days identified as holidays were omitted from the data set.

### 3.9 Half Hourly Model

For the half hourly model the data used ranged from 10th January to 6th December 1994, giving 15,888 load values. This range was restricted only by the size of the spread sheet used for data pre-processing but conveniently missed the Christmas period. Hourly valued weather data was available for temperatures and wind speeds. The missing half hours were created by simply repeating the previous value. For descriptive purposes the hours range from 0.5 to 24. The energy consumed between midnight and 00:30 is described as happening in hour 0.5. Similarly hour 13 is 12:30 to 13:00.

#### 3.9.1 Initial Input Data

Based on the previous experience the initial time inputs for this model were created to represent:

- hour of the day (sin, cos)
- day of the week (sin, cos)
- time of year (sin, cos)
- growth (linear)

For temperatures and wind speeds the current value and moving averages [45] of the previous 5, 24 and 48 hours were used as inputs, the time length based on intuition rather than any scientific findings. By filtering in this way, as opposed to using time delays, fewer inputs are required and noise is suppressed. From a common sense point

of view the load in any half hour will not depend on an exact temperature 48 hours previous, more a general underlying temperature.

Another term that was included in the initial model indicated whether it was Greenwich Mean Time (GMT) or British Summer Time (BST). This is an hour change that makes summer evenings lighter. A flag was used to indicate to which set each data point belonged.

### 3.9.2 Results

The errors of the initial network are shown in Fig 3-35 (on page 55). The four bank holiday Mondays along with Good Friday (April 1st) are clearly visible as being over-estimated, meaning the load is lower than usual on these days. A flag was created as an input to indicate bank holidays.

Overestimates occur in the period October - March,

Fig 3-38 (on page 56) showing the hours at which the largest daily overestimates occur. It can be seen quite clearly that the model has problems with hour 24 from November - March and hour 1 in April and May. The change occurs distinctly on 27th March, which corresponds to the start of BST.

Fig 3-39 (on page 56) shows a typical winter day for this model and it can be seen that hour 24 is a cardinal point [46] and thus important in load forecasts. What is happening is that there is a sudden surge in consumption in hour 0.5 caused by a tariff that exists which automatically switches water heating and storage radiator circuits on for seven hours during the night. There are over 1 million customers on this tariff of which 80% are on a fixed clock time switching mechanism explaining why there is a difference between GMT and BST. The remaining 20% are radio tele-switch controlled.

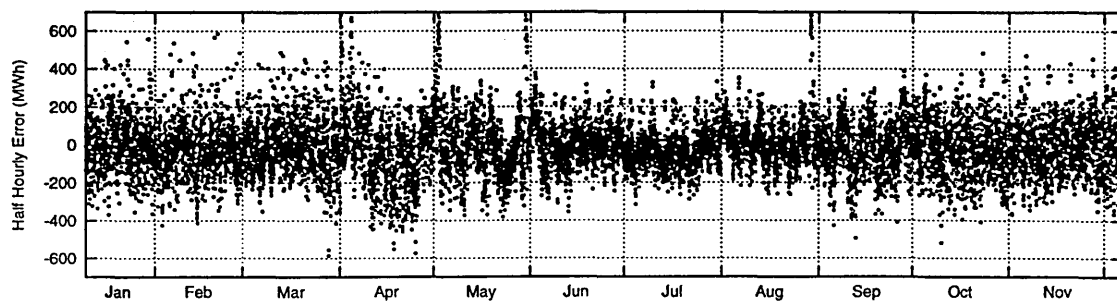


Fig 3-35 Errors with only weather and time as inputs

MAPE=2.9%

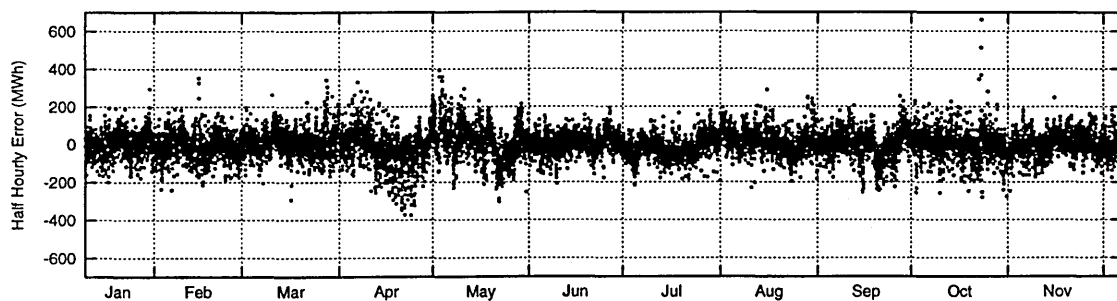


Fig 3-36 Errors with weather, time, holidays, tariffs and daylight as inputs

MAPE=1.8%

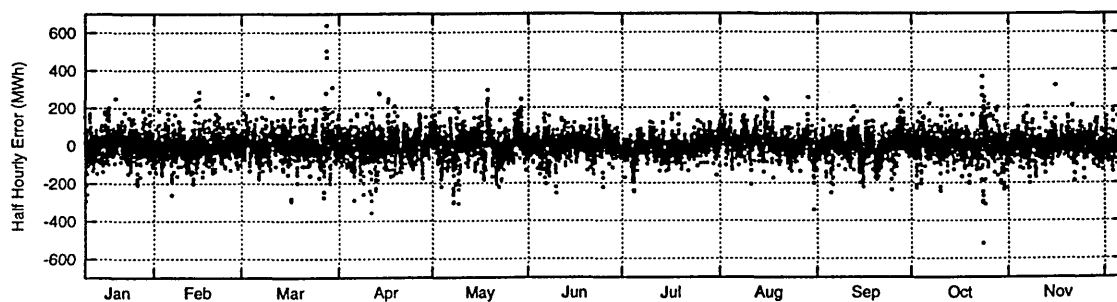


Fig 3-37 Errors with weather, time, holidays, tariffs, daylight and previous loads as inputs

MAPE=1.4%

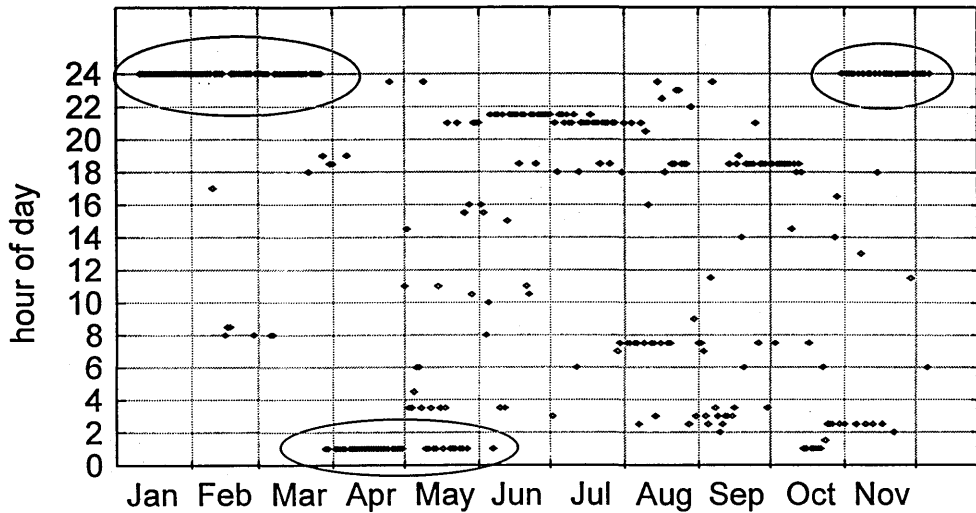


Fig 3-38 The hours at which the largest daily overestimates occur for the initial model

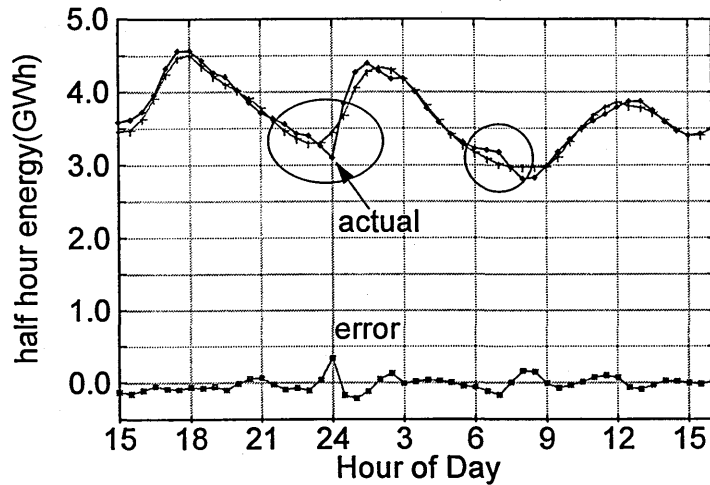


Fig 3-39 The off peak tariff is the reason why the errors occur

From Fig 3-39 it can be seen why the neural model has problems. The model inputs are generally smooth and continuous and so are fitting a smooth continuous curve through the data. There is a discontinuity in the load at midnight with the surge due to the off-peak tariff. The neural model has compensated for this effect by rounding off this discontinuity so that a smooth curve can be fitted, resulting in the overestimates for hour 24. A similar effect is seen in Fig 2-9 (on page 20). The opposite effect can be seen at

hour 7 when the heating and water charging circuits are automatically switched off. A discontinuity is also seen at hour 3 that corresponds to a similar tariff switching on at 2:30 a.m. In order to model the effects on the load due to the tariffs it is necessary to create an input that indicates the hours for which the tariffs are in operation. Flags were created for each tariff.

Fig 3-40 shows the two largest daily overestimates on the improved model, showing that the problems with hour 24 have been eliminated (exactly how this is achieved is demonstrated later). What is now evident are problems that correspond to lighting up times in the dusk period [47].

With experience gained from how the neural model reacted to the surges due to tariffs, it is assumed that the new overestimates are caused by lighting surges. Data on effective illumination [48] was unavailable so a day/night indicator was used, the transitions being sunrise and sunset times. An input to represent the dusk period used the half hours before and after sunset as the cut off points.

Fig 3-40 also reveals a cluster around hour 8 in August, which corresponds to school summer holidays and is explained if fewer people are getting up at this time, resulting in reduced demand. This cluster supports the findings in the daily load model that identified a summer anomaly, but in the half hourly case the anomalies extend further into August and early September.

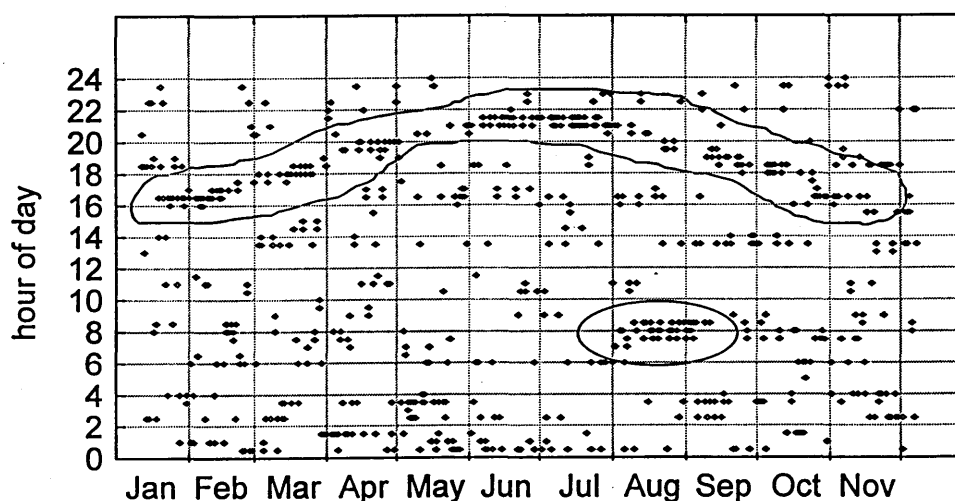


Fig 3-40 The two largest daily overestimates on the improved model now highlight lighting up time

Further inputs were created for the Easter weekend and Whitsun week. It was evident that the effect of bank holidays extended to around 6 a.m. the following morning, which makes sense if people on night shifts take this period as their holiday. This is clearly shown in Fig 3-41 with the model overestimating the early morning load. Errors the day after holidays were also identified in [33,49].

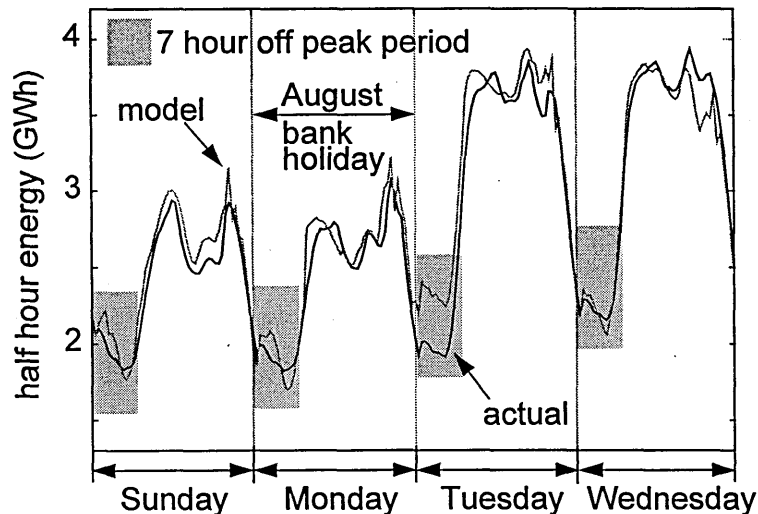


Fig 3-41 August bank holiday period modelled only with one indicator that Monday was a holiday. Reduced load requirement on Sunday and the early hours of Tuesday is very evident.

Fig 3-36 (on page 55) shows the errors of the model modified thus far. Further investigation revealed reasons for some remaining anomalies.

The overestimates in early May occur the morning after the bank holiday in the two half hours following the new cut off of the bank holiday indicator (6 a.m. on Tuesday), suggesting the influence of the bank holiday extends an extra hour in this case. There is also consistent reduced load in the working hours of this day, confirming that many people will take it as a day's holiday to make the most of the long weekend. The overestimates at the end of August occur the evening before and the late morning following the bank holiday. The big dip towards the end of May is actually the week before the bank holiday, indicating that in the run up to the holiday period more electricity than usual could be being consumed, possibly due to increased industrial output. Another

likely reason is that the model is compensating so that the errors during Whitsun are reduced.

At lighting up time the day the clocks changed on Sunday 23rd October the model seems to be an hour early in its predictions (Fig 3-42). This would be the result if the clocks for street lighting that came on with time switches were not adjusted until the Monday. A second suggestion might be a kind of 'jet lag' effect where people have not adjusted to the extra hour change. Another possibility that cannot be discounted is that the data has somehow been adjusted to account for the extra hour so that the data base will still have 24 hours in this day (how the data was dealt with is unknown). The over-estimates that occur when BST starts on Sunday 27th March are in the early hours of the following Monday morning, especially around dawn (Fig 3-43, on page 60).

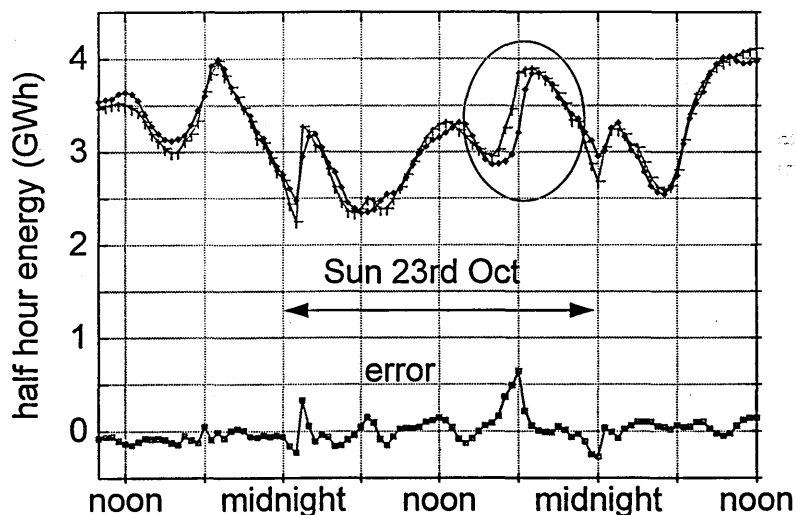


Fig 3-42 An anomaly the day the clocks change

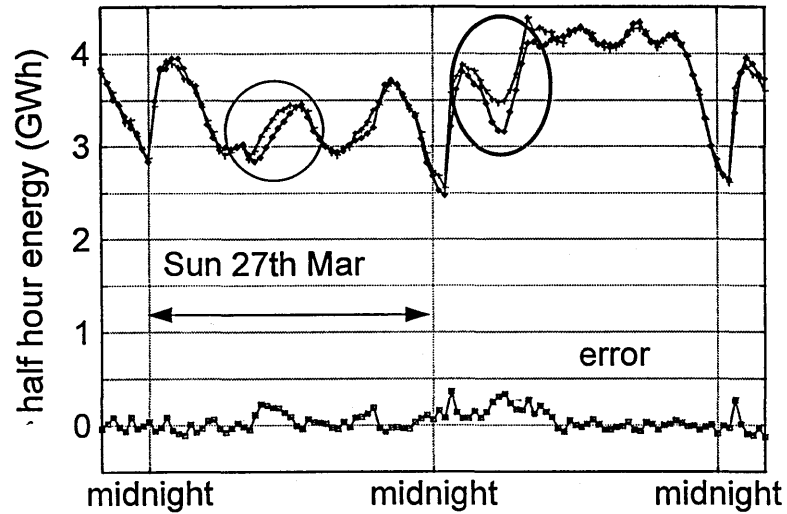


Fig 3-43 *More anomalies the day the clocks change back*

The overestimates in February occur in the late morning of the 15th. On the previous evening blizzards swept across Britain bringing many areas to a stand still, so presumably people were turning up late to work the following day, thus reducing the load (Fig 3-44).

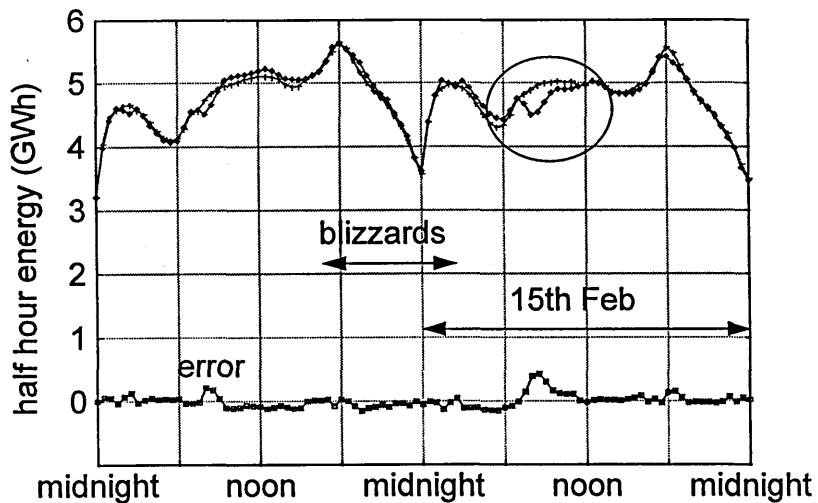


Fig 3-44 *Reduced load because of severe weather conditions*



There is a period at the end of September where there are hardly any load overestimates. Examination of the weather conditions revealed that this was peculiar in that the temperature stayed almost constant at 11 degrees Celsius for three days, both day and night.

The greatest anomaly is the second half of April when there are large underestimates. They all occur in the off-peak tariff hours which gives a clue as to their cause. What is thought to be happening is that people are gradually switching their night-time storage radiators off throughout April, as the weather gets warmer. The model can deal with them being all on or all off but struggles unless it knows exactly how many are on or off.

It would be expected that there should be similar problems in October as heaters are switched on again, which is slightly evident but not as extreme as in April. This can be explained by people's tolerance to put up with overheating but not under heating. In October all heating will probably be switched on at the first cold spell, and then left on knowing that it could always be cold again tomorrow. In April, as it gets warmer, heaters will be turned off gradually over a period of a few weeks as people get around to it. The problem of modelling April was also encountered in [39,47], where the data was for Ireland, a country with similar climate, tariffs and probably even a greater proportion of storage radiators. A possible explanation given by the authors was that the errors were due to the hour change occurring in this bi-monthly model (March - April).

### 3.9.3 Past Loads

Thus far no previous load values were included in the model and the MAPE was around 1.8%. Past loads were deliberately overlooked as it was desired to identify and explain what causes the load to be what it is, not how the model responds to good initial guesses of what the load might be. A causal model as opposed to a time series model was required.

The situation in April is an example of how including past loads might help if causal information is unavailable. The load is determined by variables such as weather, not past loads, which is to say the load depends on the underlying conditions, not what previous loads were. But since previous loads are a consequence of previous underlying conditions, they will contain information about near past conditions that may not be available in raw data form.

In [50] a feature selection algorithm showed that the previous half-hour load and the load at the same time a week before were the most important input variables, with the load at the same time yesterday also being important, a finding that is not unsurprising. Loads for the same half-hour the day and week before were included as inputs to the model, the resulting errors shown in Fig 3-37 (on page 55). The previous half-hour's load was not included as this would have to be based on predictions in a 24 hour ahead forecaster (even though forecast temperatures would have to be used). Fig 3-37 shows how the problems with April disappear and the MAPE is now around 1.4%. Comparing Fig 3-36 and Fig 3-37, the main improvements gained by including past loads are clearly in the transition periods of April and October, with no significant visible improvement in other months. What can also be seen is that although the overall error is reduced some periods are markedly worse. These stand out clearly and there will be reasons for the errors due to the fact that the period 24 hours or 1 week previous was a special event.

### **3.9.4 How the Model is Working**

By including previous loads as inputs improvements were seen in the off-peak load in the transition period of March to April. By re-setting all the binary flags so that no events occur and passing these new patterns through the trained weights, it is possible to visualise what is happening. Fig 3-45 shows this for a winter day for the two cases with and without previous loads included in the input data.

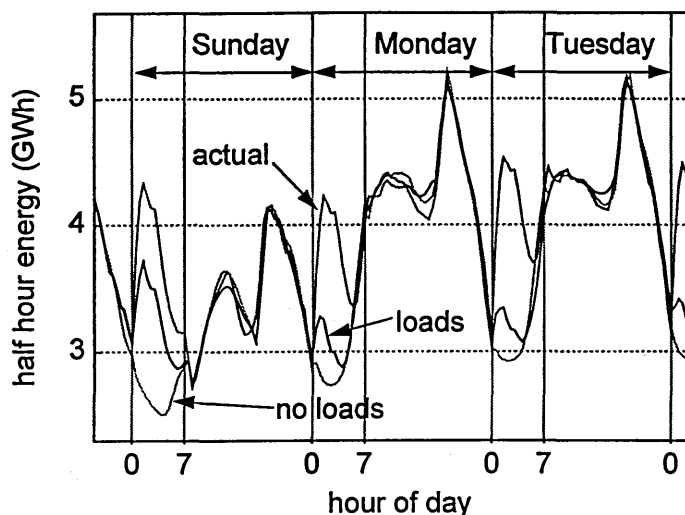


Fig 3-45 How the network models the off peak period in February

It can be seen that the model with no previous loads fits a smooth underlying load whereas when previous loads are included there is an influence of past values in the early hours. The 'no loads' curve can be thought of as the model's estimation of what the load would have been if the off-peak tariff did not exist (although the whole profile would change if these tariffs did not exist). Comparing Fig 3-45 with Fig 3-39 (on page 56) it is evident how the improved models using the binary flag deal with the discontinuity and improve the predictions for the cardinal point.

### 3.9.5 Extracting the Growth

The linear growth term was pruned from all hidden neurons and connected to its own dedicated neurons in an attempt to extract the growth. Fig 3-46 shows the resultant base load for 1994 following the same trend as the base load extracted for the previous eight years in the daily model.

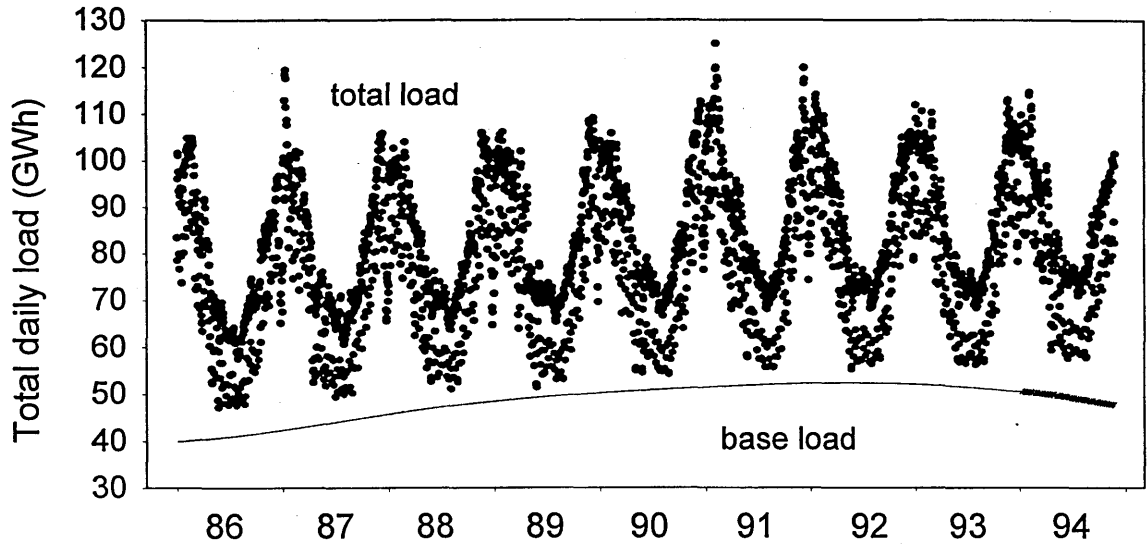


Fig 3-46 The extracted growth for the daily ('86-'93) and half-hourly ('94) models

### 3.10 Populations of Models

When creating the models it was noted that identical models with an identical number of hidden neurons always converged close to a certain value, the only difference being the random starting values given to the weights. It could be assumed that the models were almost identical, but this was not the case if viewed on a half-hourly scale. As there is noise that will exist in the loads due to unpredictable random effects and an incomplete set of variables for the inputs (the model will always be to some degree ill-posed), the neural fit has relaxed constraints and the model will describe a path through the noisy data. There are many paths that can be made through noisy data that give similar errors and always limits on the accuracy that can be achieved, assuming the number of hidden neurons is limited.

The choice then is which model to choose? For any given point the average of two

Table 3-3 Improved performance is seen by averaging several seemingly identical networks

	1	2	3	4	5	Ave
RMSE	63.75	65.80	63.91	64.26	64.36	59.50
MAPE	1.42	1.46	1.43	1.44	1.43	1.30

predictions will always be better than the worst, unless both are the same. Carrying this idea forward then as more models are created the better and more robust should the averaged performance be. Five identically created models were trained, with results shown in Table 3-3.

The overall errors of the averaged model outputs are a significant improvement on any particular individual model. Creating populations of models was a technique used in [51] but in this case each model (inputs and topology) was different. This was done to overcome the uncertainty of what inputs and connections were relevant.

### **3.11 Traditional Load Forecasting Methods**

The main objective of this chapter was to investigate how neural networks operate, using some 'real' load data to achieve this. Although the majority of papers on load forecasting published since 1992 are neural network based, in practice, more traditional techniques still predominate. The progression to neural network models is gradually taking place [24] as the technology becomes more accessible [25] and operators familiar with the issues involved.

In [52] a review of five widely applied (at the time) short term load forecasting techniques is presented, these being:

- 1) Multiple linear regression
- 2) Stochastic time series
- 3) General exponential smoothing
- 4) State space and Kalman filter
- 5) Knowledge based approach

In [53] the model types described are categorised into two basic classes each with subtypes:

- 1) Time of day
  - summation of explicit time function models
  - spectral decomposition models
- 2) Dynamic
  - ARMA models
  - State-space models

Several techniques are presented in [54] and also in [18] which includes descriptions and experiences of operational systems. Comparisons between various techniques are relatively common [52,55]. In [56] Box-Jenkins models are compared with neural networks, with the conclusion being made that;

*'Neural networks appear to be a future alternative to Box & Jenkins forecasting in all circumstances, and if they are not already so it is only due to the lack of a definitive identification procedure'.*

Similarly in [57] the following described the practical experience with using neural network forecasting in Florida;

*"..gives robust and more accurate forecasts and allows greater adaptability to sudden climatic changes compared with statistical methods."*

In [20] it was reported that in a survey of electric utilities, 16 state that the use of neural networks significantly reduced errors in daily electric load demand forecasts, while only 3 found otherwise.

It is clear that neural networks can give better results than other techniques, but why should this be so? In the following sections two of these techniques are described in an attempt to answer this question. For a more detailed descriptions of these and other forecasting techniques see [58].

### 3.11.1 Multiple Linear Regression

In multiple linear regression (MLR or multivariate regression) load forecasting [59,60,61], a simple linear model is formed from causal (or explanatory) factors that influence the load such as temperatures and day of the week. In essence the load is represented as a linear combination of these causal factors. This model has the general form:

$$\text{Load} = b_0 + b_1x_1 + b_2x_2 + \dots + b_kx_k + e,$$

where  $x_1$  to  $x_k$  are the independent causal variables,  $b_1$  to  $b_k$  their respective regression coefficients and  $e$  the error term. The regression coefficients are typically found using the least square estimation technique [52].

This model is the same as that of a neural network in its most simple form i.e. with no hidden neurons (or 1 linear hidden neuron) and a single linear output neuron. The inputs are  $x_1$  to  $x_k$  with  $b_1$  to  $b_k$  the respective weights and  $b_0$  the bias.

The addition of non-linear hidden neurons enables neural networks to perform multiple non-linear regression. As has been shown in this chapter, pruning the network can be performed so that hidden neurons only process specific causal inputs. This results in a model of the form:

$$\text{Load} = b_0 + \phi_1(x_1) + \phi_2(x_2) + \dots + \phi_kx_k + e,$$

where  $x_1$  to  $x_k$  are the independent causal variables and  $\phi_1$  to  $\phi_k$  their respective non-linear functions. These non-linear functions are formed by the linear addition of one or

more activation functions (depending on the number of hidden neurons given to each input), the exact form of which is determined by training the network.

It is therefore no surprise that neural networks can never give worse results than multiple linear regression models as they have the capacity to perform linear regression themselves if a linear hidden neuron is used.

### 3.11.2 Stochastic Time Series

The time series approach [62,63,64] to load forecasting was the most widely discussed method prior to neural networks.

The simplest form is an autoregressive (AR) model, where the load is expressed as a linear addition of terms relating to its previous values. This is similar to multiple linear regression except that the causal variables become time lagged values of the load, hence the name autoregression. The order,  $k$ , of an AR model depends on the oldest previous value that is regressed on;

$$\text{Load}(t) = b_0 + b_1 \text{load}(t-1) + b_2 \text{load}(t-2) + \dots + b_k \text{load}(t-k) + e_t$$

Moving average (MA) models are similar to AR models but the load is expressed in terms of previous error values:

$$\text{Load}(t) = b_0 + b_1 e(t-1) + b_2 e(t-2) + \dots + b_k e(t-k) + e_t$$

By effectively coupling AR and MA models the popular autoregressive/moving average (ARMA) model is created.

ARMA models are only useful for modelling stationary processes. In practical terms<sup>2</sup> a stationary process is one where the mean (stationary in the mean) or variance (stationary in the variance) do not change with time. As was seen in this chapter, a long-term trend existed in the load data due to system growth, resulting in a non-stationary series. In

---

<sup>2</sup> See [64] pg 26 for a formal definition of *stationarity*.



order to deal with a non-stationary series, it must first be transformed into a more stationary series by using a differencing process.

The first difference,  $X'_1$ , of a series  $X$  is:

$$X'_{1t} = X_t - X_{t-1}$$

With the second difference,  $X'_2$ , being:

$$X'_{2t} = X_t - X_{t-2}$$

A second order difference is first differences of first differences.

This differencing can be incorporated in the ARMA process, resulting in an *autoregressive integrated-moving average* (ARIMA) model (the 'integrated' referring to the differencing process). George Box and Gwilym Jenkins [64] have extensively studied ARIMA models and their names (Box & Jenkins) are synonymous with the general ARIMA time series model.

The three processes of ARIMA models can all be applied to the inputs of a neural network.

- 1) Autoregression involves including previous loads as inputs
- 2) Including the errors as inputs during training could incorporate the Moving Average. These error inputs would be the current model error for the specific time lags on which the error is being regressed. The error inputs for any particular pattern would thus be continually evolving from epoch to epoch as errors are reduced due to the learning. In prediction mode the model would be limited as to how far ahead it could forecast because certain time lagged errors would have to be known so that they could be used as inputs to the neural network. For example, in half-hourly modelling, if errors at the same hour the day before and a week before were used as inputs, predictions would be limited to 24 hours ahead.

- 3) A technique was developed in this chapter (see Fig 3-27) that could extract growth in the load which makes the differencing to produce a stationary series unnecessary.

### 3.12 Practical Load Forecasting

In practice, operational forecasting models are as much of an art as a science, with honest accounts of what actually happens seldom being reported due to commercial sensitivities.

In [65] the forecasting system used at British Gas to predict hourly and daily gas requirement in Great Britain is described. The basic Box-Jenkins model is used but only after a large amount of data pre-processing. National average temperatures and wind speeds are used in the forecasts, which are created from a weighted average of values from 8 weather stations around the country. The weightings are related to the relative quantity of gas sold around each station. If no forecast is available (for predictions over three days ahead) then a seasonal normal temperature is used. When only maximum and minimum daily temperature forecasts are available then the average daily temperature used is just the average of this maximum and minimum.

The data is also adjusted before it goes into the model. *'The first involves modifying the temperatures above 14 °C to account for the decrease in response of the demand series to similar changes in temperature below 14 °C....The second modification made to the raw data is to take account of holiday periods....the demand on each of these days being multiplied by the holiday factor before modelling commences'*. Because their model can only have one input (temperature), the demand is adjusted for wind speed effects. There is a 1% increased adjustment of demand for every 2.5 knots increase in wind speed between 8 and 15 knots and a 2.8% increase for wind speeds 15 knots and above.

The result of the model is about a 5% accuracy in June and 1.5% accuracy in February. It was noted that the worst forecasts occur at weekends so *'To avoid this problem the*

---

*logarithm of the demand series is taken before modelling. ....The use of 'log' models in the winter period, however, makes the forecasts worse'.*

Another method used by Company A<sup>3</sup> is to split the load into cooling and heating regimes. This is chosen to be 18.3°C and is supposed to represent the temperature above which air conditioning will be used as opposed to heating. It is known that the actual value varies with time of day and season of year but this value is used because it seems to work. Although it appears as though it is a scientifically derived temperature it is only used because  $18.3^{\circ}\text{C} = 65^{\circ}\text{F}$  – a nice round number!

Company B divides its load into day and night with 7 seasons and 5 temperature bands. An additional temperature variable is created to include '*cooling power of the wind*' for temperatures below this magic 18.3°C. An *effective temperature* is also created based on weighted average temperatures for the day in question and the three previous days. The weights are simply 1, 1/2, 1/4 and 1/8.

At Company C two models are used to give two predictions. The final decision is then made by an experienced operator who will generally choose his own figure. In this instance it would be interesting to see the results if the operator made his decision first.

### 3.13 Chapter Summary

In this chapter a MLP has been employed as a tool to help bring meaning to a large amount of data.

A common approach is to use other statistical tools, such as cross-correlation, to find possible relationships between the inputs and output in order to identify relevant inputs. Many published papers include all possible information and hope that the MLP will sort out the numbers for itself. In this chapter we have used the MLP as a tool itself in order to identify relationships in data. The reasoning is that if MLPs can produce non-linear models then why use other linear based techniques in order to identify inputs.

In academia there is an obsession with statistical measures in order to gauge the performance of a model. As has been shown in the real data used, statistical anomalies often have reasons, and determining these reasons is much more beneficial in real situations than quoting significance measures. The disadvantage is that it requires a little more work.

In this chapter several new ideas have been introduced which it is hoped will help advance the understanding of neural networks and the load forecasting problem. The approach was to challenge the common misconception of neural networks that they are 'black boxes' that have no explanation of what they do. This 'black box' idea does not make sense knowing the simple processing that goes on within the neurons. Understanding what goes on is just a matter of investigation.

---

<sup>3</sup> These are from the authors personal 'in trust' communications with several companies.

# 4

## **Genetic Inspired Optimisation**

---

### **4.1 What are Genetic Algorithms?**

Genetic algorithms (GAs) are directed random search techniques used to look for parameters that provide a good solution to a problem. Essentially they are nothing more than educated guessing. The ‘education’ comes from knowing the suitability of previous candidate solutions and the ‘guessing’ comes from combining the fitter attempts in order to evolve an improved solution.

For example, the back propagation algorithm is a gradient based method for finding a weight set for a MLP that best maps the inputs onto the output, a search that can also be performed by GAs [7]. The optimisation problem of interest in this work is finding a schedule for electrically charging storage devices (hot water tanks and storage radiators)

over a 24 hour period, given that electricity prices vary half-hourly. A solution is sought that minimises electricity costs whilst satisfying the hot water or thermal comfort requirements.

## 4.2 How do GAs Work?

The inspiration for GAs came from nature and survival of the fittest. In a population, each individual has a set of characteristics that determine how well suited it is to the environment. Survival of the fittest implies that the 'fitter' individuals are more likely to survive and have a greater chance of passing their 'good' features to the next generation. In sexual reproduction, if the best features of each parent are inherited by their offspring, a new individual will be created that should have an improved probability of survival. This is the process of evolution.

In nature the 'blueprint' of individuals is contained within their DNA. The DNA can be thought of as a string of genes, with each gene or combination of genes representing a particular feature. Reproduction is the 'crossover' of two DNA strings to produce a new blueprint that has genes from both parents. Mutation can also occur where a particular gene is not an exact copy of either parent.

In genetic algorithm terms, a candidate solution is often referred to as a chromosome or string, which is a sequence of encoded numbers. This is commonly referred to as a bit string if the numbers are binary encoded.

The process involved in GA optimisation problems is based on that of natural evolution and broadly works as follows,

1. Randomly generate an initial population of potential solutions.
2. Evaluate the suitability or 'fitness' of each solution.
3. Select two solutions biased in favour of fitness.

4. Crossover the solutions at a random point on the string to produce two new solutions.
5. Mutate the new solutions based on a mutation probability.
6. Goto 2.

### 4.3 The GA Operators

Selection, crossover and mutation are the basic operators involved in GAs. How these and other factors can affect the operation of GAs will be demonstrated by means of several examples and experimental observations.

Consider the popular board game 'Mastermind' where a player has to determine a hidden sequence of colours starting from an initial random guess. This initial guess is scored with a black marker for each colour in the correct position and a white marker for a correct colour but in the wrong position. Further guesses are made and scored until the correct sequence is determined or a given number of attempts have been made. In this game the correct solution evolves from the more suitable of all previous attempts, with clues from unsuitable candidate solutions also being part of the deduction process. This is a type of 'blind' optimisation problem where no information is available on what makes a good solution, only information on how good solutions are.

Given a few initial guesses the player will *select* high scoring attempts and perform *crossover* to see if this results in an improvement. New colours will almost certainly have to be *mutated* into the 'educated guesses' in the attempt to find the correct sequence.

Fig 4-1 demonstrates how these three operators work considering a scoring scheme where a point is scored only for a number in the correct position.

The GA search procedure is very easy to understand and implement, with nature providing ready examples of exactly how things could be done.

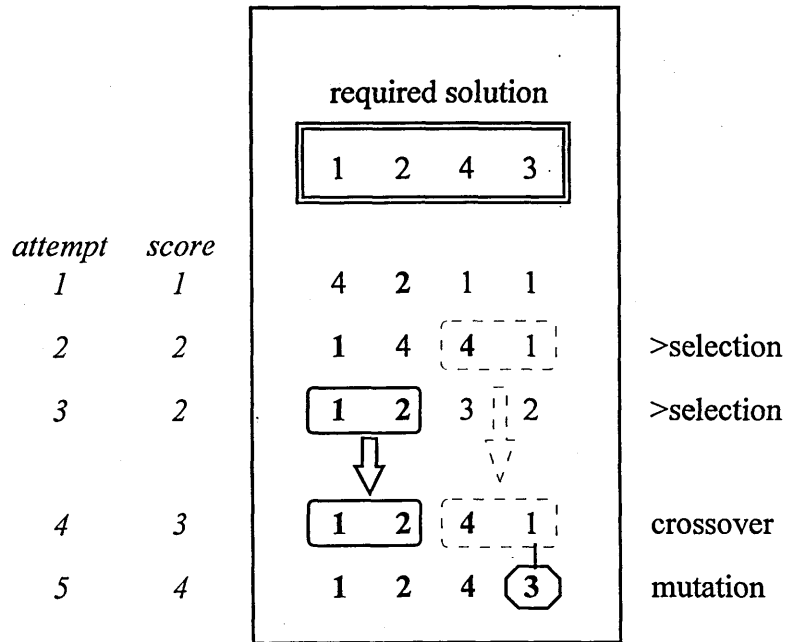


Fig 4-1 An example of how the required solution evolves using the selection, crossover and mutation operators

## 4.4 Implementation

### 4.4.1 Encoding

In optimisation problems a set of parameters is sought that will give the best solution to a particular problem. In order to implement a GA these parameters must be encoded into a string so that crossover and mutation can be applied. Binary encodings are the most common, due to the fact that Holland used them in his early pioneering work [66]. In DNA base 4 encoding is used, as the building blocks of DNA can take on 4 values, translated as A, C, G, or T.

Any base can be used, as it is just a different method of encoding the same information, but the lower the base the longer the string will be. For example, if a number is sought between 0 and 255 then this can be encoded as a binary string of length 8, a base 4



**Table 4-1** Representations of the base 10 numbers 0 and 255 in different bases

base 2 length=8	base 4 length=4	base 10 length=3	base 16 length=2	base 256 length=1
00000000	0000	000	00	0
11111111	3333	255	FF or  15 15	255

string of length 4, a base 16 string of length 2 or a base 256 string of length 1, as shown in Table 4-1.

It is clear that the importance of the operators will change depending on the base used. In base 2, the two given strings contain all the information required to derive any number between 0-255 by crossover alone. In base 16, two strings can at most lead to only 4 different numbers by crossover alone, mutation being required to introduce new information. In base 256 crossover cannot occur, mutation being the only operator that can introduce new numbers and finding a specific number becomes a pure random search.

In choosing an encoding scheme the nature of the problem will play a major role. If many real valued numbers are required in a solution then binary encoding becomes impractical as the string length increases. In [67] a method of selective genome growth is proposed that helps solve the problem of choosing how to represent a genetic algorithm.

#### 4.4.2 Population Size

The population size is the number of candidate solutions in any one generation. In natural evolution the total population size is governed by what is sustainable by the environment and similarly in GAs the larger the population size the more computationally intensive (in terms of memory requirement) is the search.

In nature, the bigger the gene pool the more diverse is the genetic make up of the population with many individuals each with their own set of characteristics that enable them

to survive. One advantage of this diversity is that there will be no dominant gene that, for instance, may be susceptible to a particular disease and result in the elimination of the whole species. In the bird family, sub-species have evolved with dominant characteristics that allow them to survive their local conditions and in effect are sub-optimal solutions in the search for a global 'super-bird'. With large populations it can be seen how the search for the global optimal solution can be a slow (if not never-ending) process.

If the population size is small (e.g. a pride of lions), then a strong individual quickly becomes dominant and the diversity of the gene pool is restricted. The result is that good individuals (local optima) are quickly created but the dominance of particular genes restricts the search space. The chance of evolving the ultimate 'super-lion(ess)' (global optimum) is severely limited and would depend on mutation introducing new genes to diversify the search.

As new solutions are generated it is common to keep the population size constant by eliminating individuals (or letting them die), although this does not have to be the case. Ideas for the selection procedure for elimination are plentiful in nature. For example, each generation could be completely replaced by its offspring, or as a new offspring is created it could be accepted or rejected depending on its fitness. The advantage computers have over nature is that good individuals do not have to die and can be retained for indefinite reproduction. The retention of certain fit individuals is known as 'elitism'.

#### **4.4.3 Selection**

This is the procedure for choosing individuals (parents) on which to perform crossover in order to create new solutions. The idea is that the 'fitter' individuals are more prominent in the selection process, with the hope that the offspring they create will be even fitter still.

---

Two commonly used procedures are ‘roulette wheel’ and ‘tournament’ selection. In roulette wheel, each individual is assigned a slice of a wheel, the size of the slice being proportional to the fitness of the individual. The wheel is then spun and the individual opposite the marker becomes one of the parents. In tournament selection several individuals are chosen at random and the fittest becomes one of the parents.

#### **4.4.4 Crossover**

Along with mutation, crossover is the operator that creates new candidate solutions. A position is randomly chosen on the string and the two parents are ‘crossed over’ at this point to create two new solutions. Multiple point crossover is where this occurs at several points along the string. A crossover probability ( $P_c$ ) is often given which enables a chance that the parents descend into the next generation unchanged.

#### **4.4.5 Mutation**

After crossover, each bit of the string has the potential to mutate, based on a mutation probability ( $P_m$ ). In binary encoding mutation involves the flipping of a bit from 0 to 1 or vice versa.

### **4.5 Experiments with GAs**

#### **4.5.1 Chinese Hat Optimisation Problem**

To empirically evaluate the importance of the various parameters and techniques in GAs, several optimisation tests were performed. The code used is based on that in Appendix D. The experiments used tournament selection and a constant population size with the offspring replacing the parents every generation.

**Table 4-2** An example of how the fitness of the solutions to the Chinese Hat problem are evaluated for a string length of 8. Each bit value in a solution is multiplied by the value in the same position in the scoring template and the total fitness is the square of the sum of all the bit scores. Each bit can have a value of 1 or -1

Scoring Template	4	3	2	1	-1	-2	-3	-4
Candidate Solution 1	1	1	-1	1	-1	-1	1	-1
Bit by bit Score 1	4	3	-2	1	1	2	-3	4
Total Score 1 = $10^2 = 100$								
Candidate Solution 2	-1	-1	-1	-1	1	1	1	1
Bit by bit Score 2	-4	-3	-2	-1	-1	-2	-3	-4
Total Score 2 = $(-20)^2 = 400$								

The fitness evaluation function (fitness landscape, scoring template) of candidate solutions for the first optimisation problem examined is shown in Table 4-2. For reference purposes this problem has been named the Chinese Hat because the scoring template diverges linearly outwards from the centre. There are two possible solutions for maximum fitness, one of which is shown by candidate solution 2, the other is the inverse of this where all the bits flip. The total number of candidate solutions is  $2^{(\text{string length})}$ .

In the experiments, tests for each particular parameter setting were repeated to convergence 200 times to determine the average number of generations required to find the solution. Each subsequent trial differed by randomly generating a new initial population. After each crossover, mutation was only allowed on one randomly selected bit and whether it occurred depended on  $P_m$ .

### 4.5.2 Results

The results of varying the GA parameters for the Chinese Hat optimisation problem are shown in Fig 4-2 to Fig 4-8. All comments and discussion related to each figure are included below that figure.

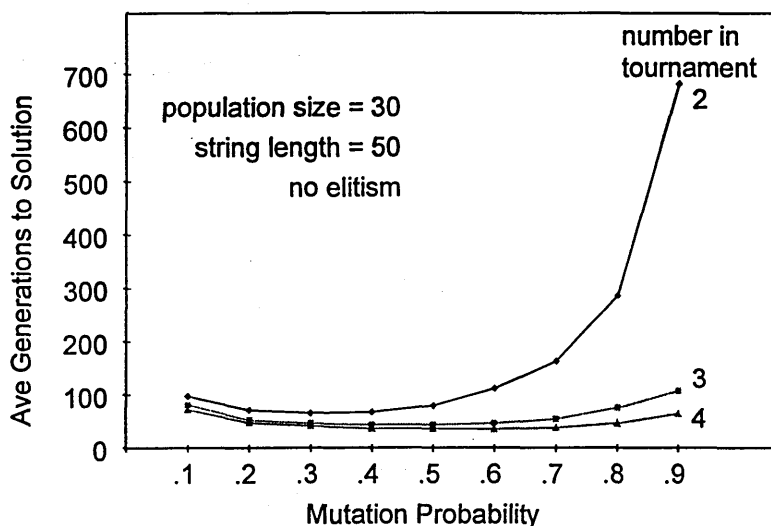


Fig 4-2 The effects of  $P_m$  and the selection procedure

By increasing the number of candidates (competitors) in the tournament for parenthood the number of generations required to convergence reduces. This would indicate that little diversity in the gene pool is required for this particular problem. There is also an optimum  $P_m$  around 0.5. With a higher mutation probability the number of generations starts to increase, although this becomes less significant as the selection procedure is made more competitive. In Fig 4-2,  $P_c = 1$ .

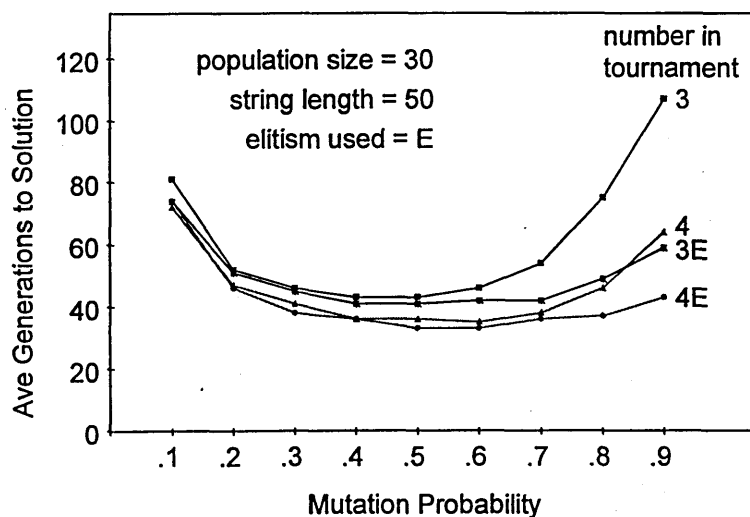


Fig 4-3 Elitism

The introduction of an elitist strategy, where the best individual is always retained, shows significant improvements in performance but only for the higher mutation rates, indicating the solution is evolved from mutation of this 'best' individual.

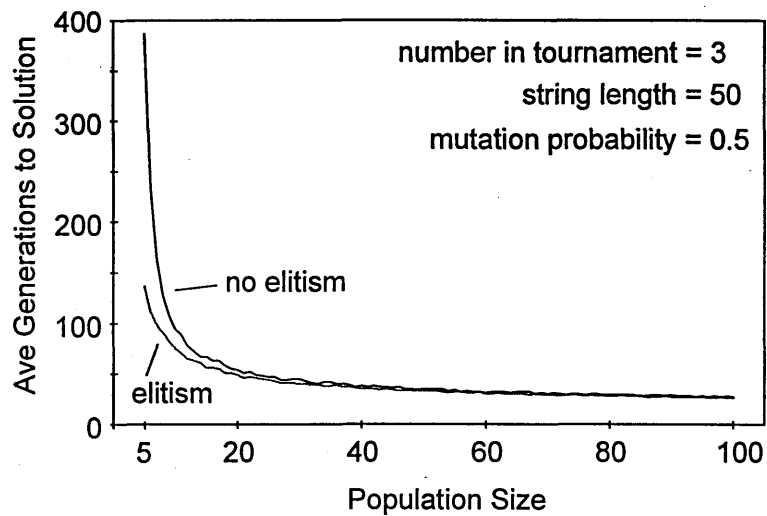


Fig 4-4 Population size

As would be expected, the larger the population size the fewer generations are required as the search space is increased. The highest rates of gain are seen by increasing the population size to 20, but even after this consistent reduction still occurs, as shown in Fig 4-5.

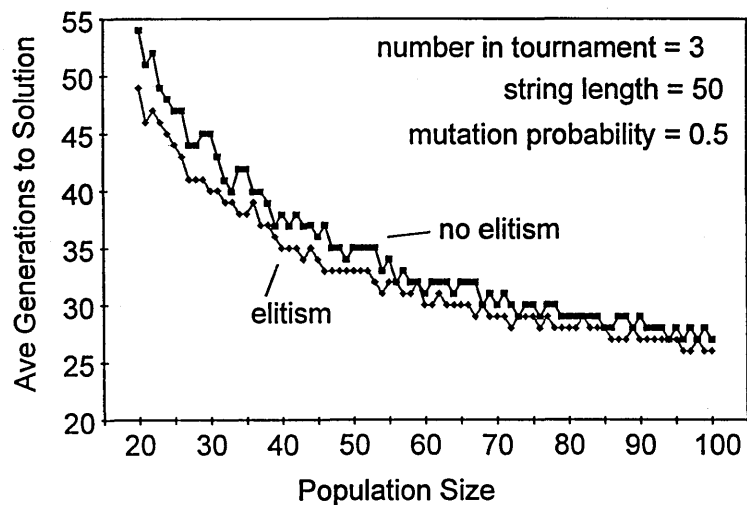


Fig 4-5 Population size

A close up of Fig 4-4 shows consistent improvement in performance with increasing population size.

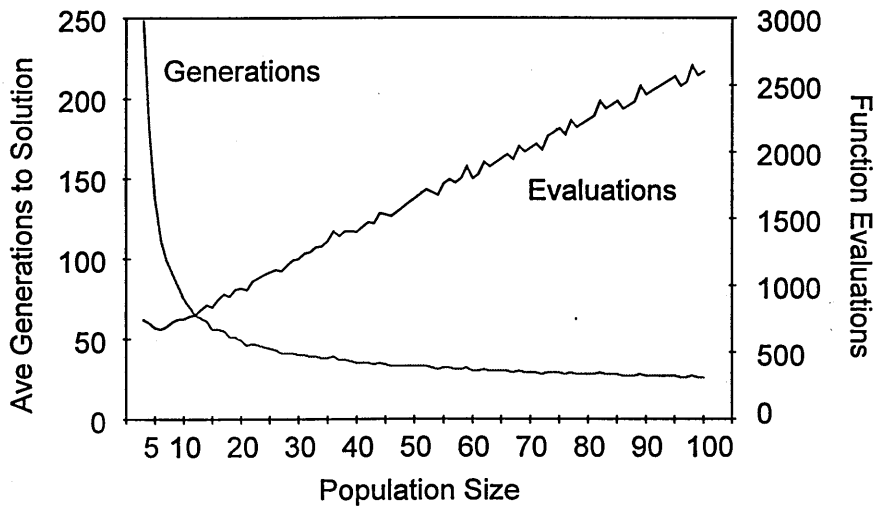


Fig 4-6 *Function evaluations*

In serial computing it is not the number of generations that is important but the number of function evaluations. That is, how many solutions must be evaluated before the optimum is reached, or roughly the number of generations multiplied by the population size. This gives an indication of the computing power (or time) required to solve the problem, assuming that evaluating the cost of each solution is a significant portion of the whole process. Fig 4-6 is the same data as that of the elitist tests in Fig 4-4, but with the number of evaluations also shown. It can be seen that for the given parameters, a population size of 6 is the most economical. After this the number of evaluations increases linearly with population size.

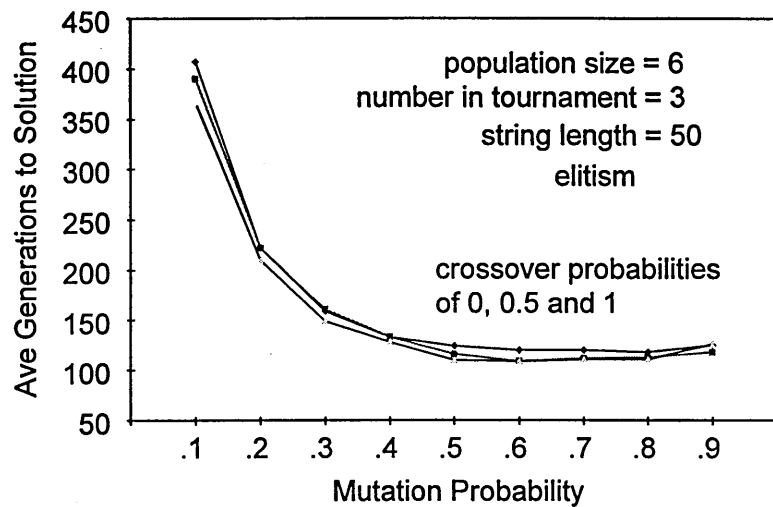


Fig 4-7 The effect of crossover and mutation probabilities for a population size of 6

Thus far crossover has occurred in every reproduction. By introducing a crossover probability, the relative importance of crossover and mutation can be examined. This is shown for the most efficient population size of 6 and crossover probabilities of 0, 0.5 and 1. What is seen is that the optimisation procedure for this small population relies solely on mutation with the crossover probability having a negligible effect.

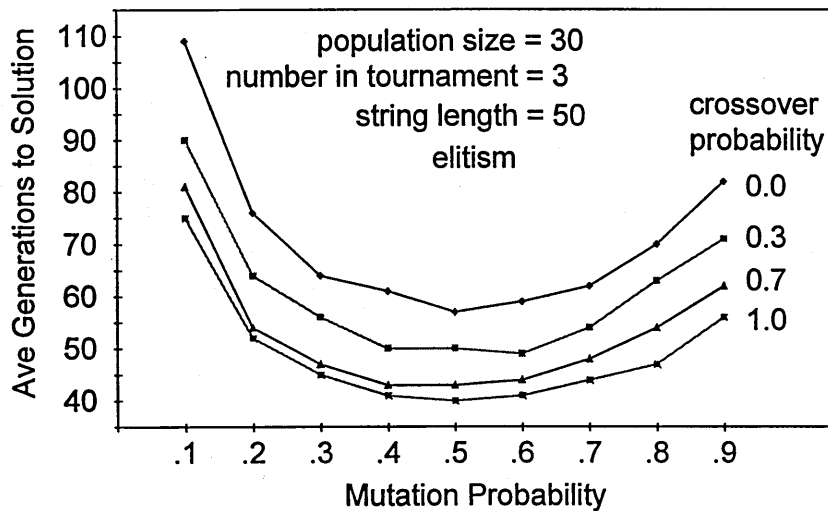


Fig 4-8 The effect of crossover and mutation probabilities for a population size of 30

With a larger population size increasing the crossover probability does improve the performance. Fig 4-8 is generated by the same procedure as Fig 4-7 but with a population size of 30. It can be seen with this larger population size there is an optimal  $P_m$  but improvements are also made by increasing  $P_c$ . What Fig 4-7 and Fig 4-8 show is not unexpected and is reflected in nature in that small populations rely on mutation for diversity whereas in larger populations it is a combination of crossover and mutation.



The experiments on this simple optimisation problem have illustrated that selecting the correct parameters is very important in genetic algorithms. What is also very evident is that there are definite relationships between all the parameters showing that fine-tuning is required to increase the speed to success, or reduce the chance of failure. The technique always managed to solve the problem, but how does it compare with other hill-climbing methods?

### 4.5.3 Other Iterated Hill-Climbing Methods

Other optimisation methods exist of which three were used for comparison with the GA. The following descriptions of these techniques are reproduced from [68].

#### 4.5.3.1 Steepest-Ascent Hill Climbing (SAHC)

1. Choose a string at random. Call this string *current-hilltop*.
2. Going from left to right, systematically flip each bit in the string, recording the fitness of the resulting strings.
3. If any of the resulting strings give a fitness increase, then set *current-hilltop* to the resulting string giving the highest fitness increase (ties are decided at random).
4. If there is no fitness increase, then save *current-hilltop* and goto step 1. Otherwise goto step 2 with the new *current-hilltop*.
5. When a set number of function evaluations have been performed (here, each bit flip in step 2 is followed by a function evaluation), return the highest hilltop that was found.

#### 4.5.3.2 Next-Ascent Hill Climbing (NAHC)

1. Choose a string at random. Call this string *current-hilltop*.
2. For  $i$  from 1 to  $l$  (where  $l$  is the length of the string), flip bit  $i$ ; if this results in a fitness increase, keep the new string, otherwise flip bit  $i$  back. As soon as a fitness increase is found, set *current-hilltop* to that increased fitness string without evaluating any more bit flips of the original string. Go to step 2 with the new *current-hilltop*, but continue mutating the new string starting immediately after the bit position at which the previous fitness increase was found.
3. If no increase in fitness were found, save the *current-hilltop* and goto step 1.
4. When a set number of function evaluations has been performed, return the highest hilltop that was found.

#### 4.5.3.3 Random-Mutation Hill Climbing (RMHC)

1. Choose a string at random. Call this string *current-hilltop*.
2. Choose a bit at random to flip. If the flip leads to an equal or higher fitness, then set *current-hilltop* to the resulting string.
3. Goto step 2 until an optimum string has been found or until a maximum number of evaluations has been performed.
4. Return the current value of *current-hilltop*.

1000 trials of each of these three algorithms were performed on the Chinese Hat problem for a string length of 50. The average number of evaluations, given in Table 4-3, shows that a GA is not the best method of solving this particular problem. In fact NAHC

**Table 4-3** *The average number of function evaluations over 1000 trials for the Chinese Hat problem with a string length of 50*

SAHC	NAHC	RMHC	best GA	MRMHC
1082	48.6	190	650	430

always reaches a global solution by traversing the string just once because the Chinese Hat is a smooth function when traversed from left to right.

The best GA based performance had a population size of 6 with a randomly selected gene being mutated with a probability of 0.5. With these parameters it was shown that crossover had a very limited effect (Fig 4-7, on page 84). As mutation appears to be the dominant process a variation of RMHC we called *multiple random mutation hill climbing* (MRMHC) was tried on the Chinese Hat function. This is basically RMHC but with each gene having a mutation probability as opposed to one randomly selected gene being mutated. Another way of describing MRMHC is a GA with a population size of 1 and the mutated offspring only surviving if it is as good as or better than the parent.

Fig 4-9 (on page 88) shows the average performance of MRMHC over 1000 tests with varying mutation probabilities and string lengths. The circled points represent the optimum mutation probabilities for the various string lengths with a pattern emerging that a  $P_m$  of  $(1/\text{string length})$  appears to be the optimum. On average this is equivalent to 1 bit change per mutation which is an unsurprising result. Any less than this and some evaluations will be wasted as there will be no change, any more and there will be problems in fitting the last bit into position as there is a higher probability that more than one bit will be changed at once. The best performance with MRMHC for a string length of 50 was 430 evaluations, which occurred with  $P_m$  at just over  $1/50$  or 0.02.

The optimum mutation rate for MRMHC is on average 1 bit change per string, which is almost similar to RMHC, in which only one bit change per string is permitted. Similar results would be expected but it is noted that MRMHC requires over twice as many evaluations as RMHC. Why should this be so?

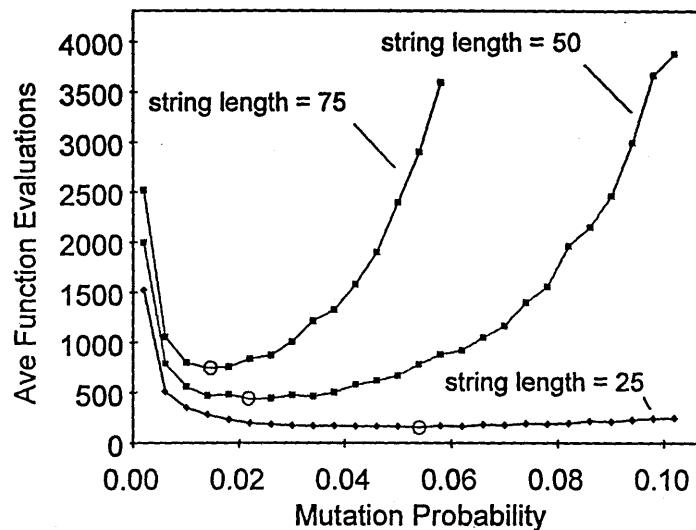


Fig 4-9 The effect of mutation probability and string length for Multiple Random Mutation Hill Climbing optimisation

If there is only one bit flip that is required to reach the optimum then RMHC should find this in an average number of evaluations equivalent to the string length. In MRMHC it is possible that in three consecutive evaluations the number of bit flips is 0, 1 and 2, giving an average of 1. An evaluation with 0 flips is wasted and because only one bit requires correction, two bit flips will never find the optimum. It can thus be hypothesised why MRMHC gives a worse performance than RMHC for this particular problem.

#### 4.5.4 Royal Road Functions

So what kind of problems will GAs be superior at solving than other search techniques?

The Schema Theorem and Building Block Hypothesis [66, 69] play on the idea that solutions are made up of short blocks of fit schema that use crossover to build up these schema into desirable solutions. A set of functions known as the 'Royal Roads' [68, 70, 71, 72] were developed that provide a fitness landscape designed specifically to be easily solvable by GAs if they did work in this building block manner. As described by the developers (Mitchell et al.), *'given the building block hypothesis, one might expect*

**Table 4-4** *The Royal Road ( $R_1$ ) fitness function. A bit string of length 64 contains 8 short schema that are the building blocks of the optimal schema. The wildcard '\*' represents a 0 or 1 (or 'do not care'). The fitness of each candidate solution increases with the number of these building blocks present.*

11111111 *****	Schema 1	= 8
***** 11111111 *****	Schema 2	= 8
***** ***** 11111111 *****	Schema 3	= 8
***** ***** ***** 11111111 *****	Schema 4	= 8
***** ***** ***** ***** 11111111 *****	Schema 5	= 8
***** ***** ***** ***** ***** 11111111 *****	Schema 6	= 8
***** ***** ***** ***** ***** ***** 11111111 *****	Schema 7	= 8
***** ***** ***** ***** ***** ***** ***** 11111111	Schema 7	= 8
11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111	Schema Opt	= 64
11111111 11110011 01110011 11111111 11111111 00000001 11110010 11111111	e.g. Score	= 32

that the building block structure of  $R_1$  will lay out a "royal road" for the GA to follow to the optimal string'. Table 4-4 shows one of these Royal Road functions,  $R_1$ .

In their analysis, Mitchell et al. used a GA with a population size of 128, single point crossover with  $P_c$  fixed at 0.7 and  $P_m$  at 0.005, full details are given in [68]. Over 200 runs the mean number of GA function evaluations was 61,334, an order of 10 times higher than RMHC (6,179). NAHC and SAHC never reached the optimum solution, which is not unexpected given the nature of the fitness landscape.

In section 4.5.2 the importance of the GA parameters was demonstrated, although only on a simple smooth function that proved easier to solve by other methods. The Royal Road problem was investigated in the same manner to determine if the nature of the problem affected the relationship between the parameters.

Initial tests were performed to see if the results of Mitchell et al. could be replicated and also to examine the effect of varying the GA parameters. Mutation probabilities between 0.002 (0.13 in 64) and 0.05 (3.2 in 64) were tested for crossover probabilities between 0 and 1 inclusive. Each set of parameters was repeated to convergence 20 times and the mean value recorded. Tournament selection was used where each parent was the best of 5 randomly chosen candidates. The results are shown in Fig 4-10.

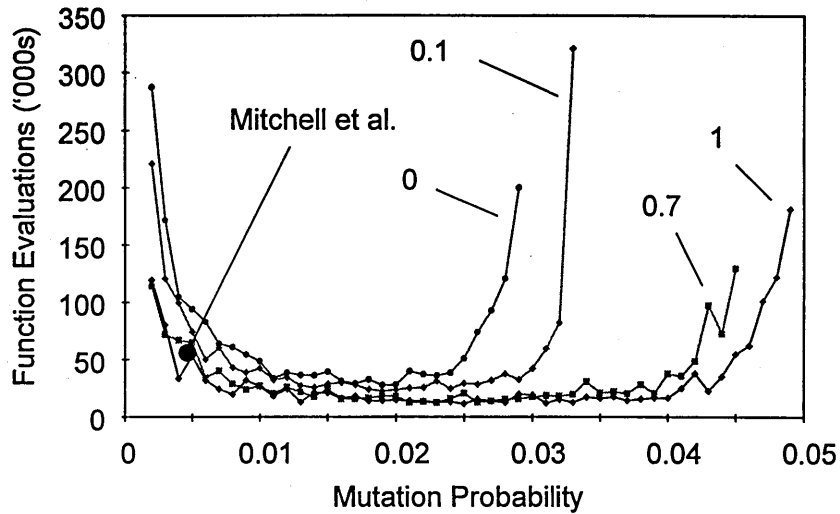


Fig 4-10 The effect of the mutation probability for four crossover probabilities (0,0.1,0.7,1) on the Royal Roads ( $R_1$ ) landscape. Each point is the average over 20 tests with a population size of 128. Mitchell et al. used a mutation probability of 0.005 (0.33 in 64) and crossover probability of 0.7 that gave a mean of 61,334 function evaluations to convergence over 200 tests.

The results of Mitchell et al. were easily replicated even though a different selection procedure was used. By increasing  $P_m$  to 0.02 (1.3 in 64) the number of evaluations was reduced to around 14,000, a factor of 4 improvement and only twice as many as RMHC. With this mutation probability the function could be optimised in 28,000 evaluations without using crossover at all, half as many as the evidently poorly tuned GA of Mitchell et al. With no crossover, each offspring is a mutation of a parent chosen due to its fitness.

It has been demonstrated that a GA with no crossover can outperform a poorly tuned GA on a fitness landscape purposely designed to suit the crossover operator. If it can be discovered what determines a good mutation probability with no crossover then this should be generally applicable when crossover is applied.

With no crossover, the relationship between the mutation probability and selection procedure was examined. In previous tests on the  $R_1$  landscape tournament selection was used where each parent was the fittest of 5 randomly selected candidates. The number of candidates was varied along with  $P_m$ , as shown in Fig 4-11 (on page 91). What is clear is that the less stringent the selection, the tighter the band is for an acceptable value of  $P_m$ .

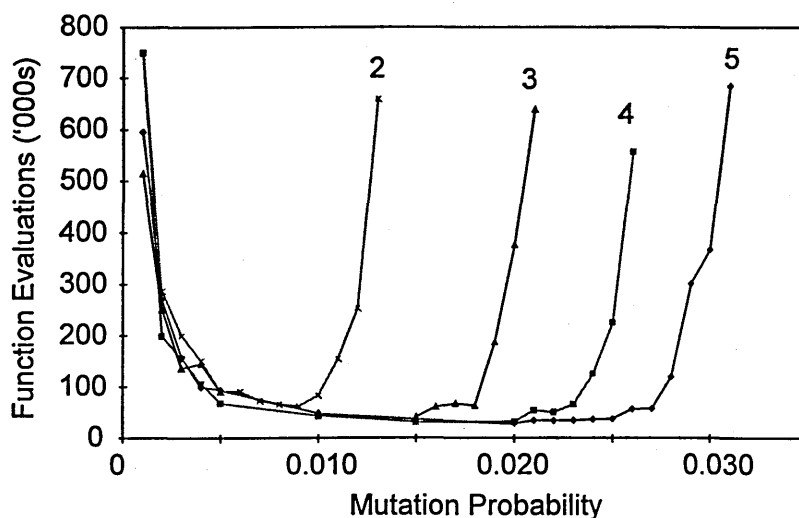


Fig 4-11 The relationship between the mutation probability and the number of contestants in tournament selection (2,3,4, and 5). The crossover probability is 0 and the population size is 128. The mean of 20 trials was recorded.

For each selection policy there is also an upper mutation probability past which the required evaluations increase exponentially; the more stringent the selection then the higher is this upper limit.

The objective of this exercise was to discover what determines a good mutation probability, which has been shown for  $R_1$  to also depend on the selection procedure used, becoming more important the weaker the selection procedure. There is a definite lower limit around 0.005 or 0.33 in 64.

In order to determine a desirable mutation rate the effect of the population size must also be investigated. Fig 4-12 (on page 92) shows that given a near optimal  $P_m$  (0.01) there is little sensitivity to population size, but as  $P_m$  increases so does the sensitivity to population size.

The conclusion reached thus far is that the mutation probability is the most important GA parameter in solving the  $R_1$  landscape. There is also much evidence (and common sense) to suggest that the optimum mutation probability is related in some way to the string length. In order to test this theory MRMHC (a GA with no crossover and population size 1 with the new solution being retained if it is better than or equal to the parent)

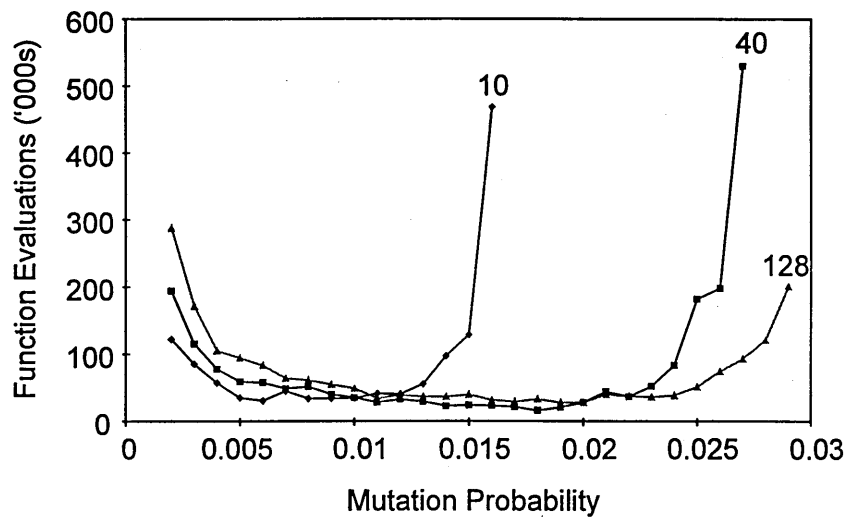


Fig 4-12 The effect of the mutation probability and the population size (10,40,128). In these cases the  $P_c = 0$  and the number of candidate parents is 5.

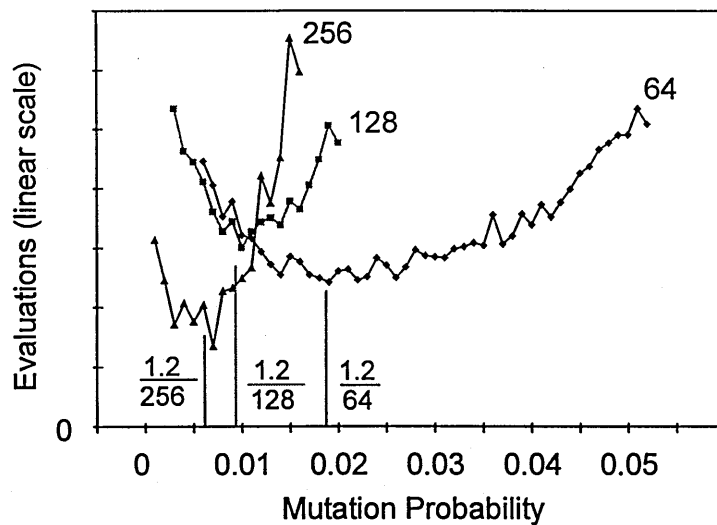


Fig 4-13 The effect of the mutation probability on MRMHC, averaged over 200 tests. The optimum is  $(1.2/64)$ .

was used to solve three versions of  $R_1$ , with string lengths of 64, 128 and 256. The results in Fig 4-13 show that the longer the string the more sensitive is the search to  $P_m$  (the y-axis scale in Fig 4-13 is different for each population size). The optimum value of  $P_m$  was observed to be about  $(1.2/\text{string length})$ . The question to be investigated now is what is special about this mutation rate?



In their work, Mitchell et al. analysed the RMHC algorithm with a simple derivation based on probability that gave the expected number of function evaluations to solve  $R_1$ .

Consider  $R_1$  as in Table 4-4. In each schema of length 8 the number of possible combinations is  $2^8$ . If one and only one bit is changed in each evaluation then the chance of this bit being in a specific schema is  $1/8$ , since there are 8 schemas in total. Thus the chance of randomly creating a particular schema is once every  $2^8 \times 8$  evaluations. Initially there are eight schemas to choose from so the chance of creating any schema is once in every  $2^8 \times 8/8$  evaluations. Once one schema is found the chance of finding a further schema decreases to  $7/8$  of that of finding the first since 1 in 8 bit changes are likely to be wasted changing the already discovered schema. The number of evaluations required to find this second schema thus increases to  $8/7$  that required to find the first. The expected number of evaluations to find a single schema is in fact slightly more than  $2^8$  and as determined by a Markov-chain analysis it is 301.2 [68]. The expected number of evaluations to solve the problem is thus,

$$301.2 \times 8 \times \left( \frac{1}{8} + \frac{1}{7} + \frac{1}{6} + \frac{1}{5} + \frac{1}{4} + \frac{1}{3} + \frac{1}{2} + \frac{1}{1} \right)$$

↑  
1st

↑  
discovered  
schema

↑  
8th

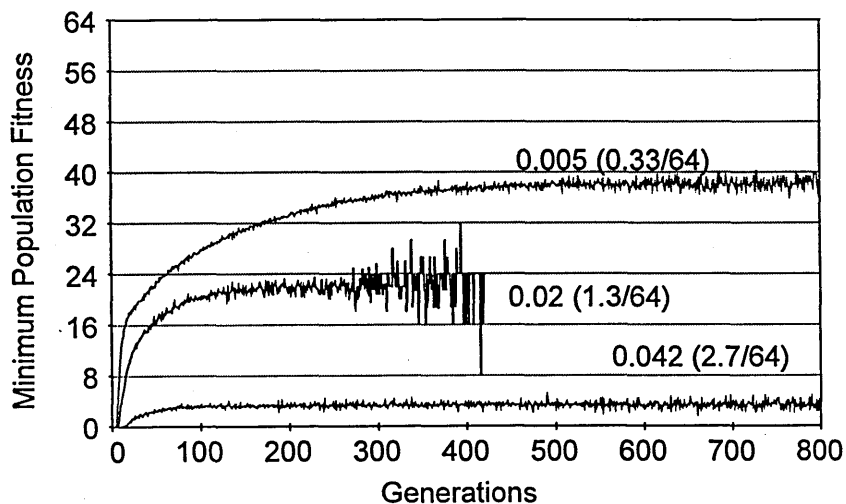
Tests were performed for RMHC that tracked the creation of the schema in the solution in order to confirm the theoretical performance. Table 4-5 shows the results averaged over 1000 trials which almost mirror the theoretical expectations.

**Table 4-5** *The theoretical and experimental (averaged over 1000 trials) number of evaluations to discover each subsequent schema for  $R_1$  using RMHC. The total theoretical evaluations = 6,549, experimental = 6,542 and Mitchell et al. = 6,179.*

schema	1	2	3	4	5	6	7	8
theoretical evaluations	301.2	344.6	401.6	481.9	602.4	803.2	1204.8	2409.6
experimental evaluations	284	355	384	508	622	797	1182	2410

RMHC has been shown to behave as the probability theory predicted. In GAs the theory of how they behave remains a theory, with little experimental evidence to try to observe their actual behaviour.

Tests were performed using GAs on the  $R_1$  landscape that tracked the formation of the schema. The trials were performed 500 times with the maximum number of generations set at 800. The maximum, minimum and average fitness of the population were recorded at each generation and averaged for the trials that had not converged. Initially the crossover rate was set at 0.7, population size 128 and the number of competitors in the tournament was 5. Three mutation rates were used, 0.33/64, 1.3/64 and 2.7/64. The results are shown in Fig 4-14 to Fig 4-17.



**Fig 4-14** *The effect of the mutation rate on the minimum population fitness*

The lower the mutation rate the fitter is the worst individual in the population. Note that in all cases the minimum fitness reaches a plateau and only for the middle mutation rate of 1.3/64 do all the trials converge.

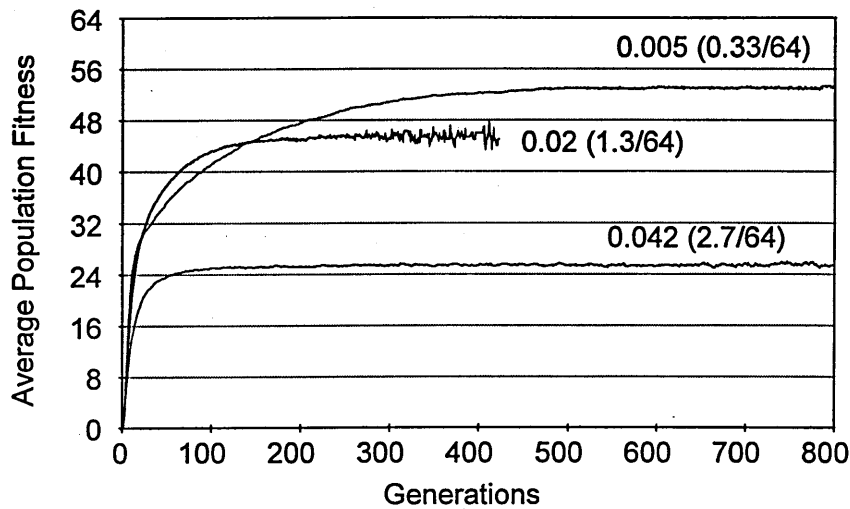


Fig 4-15 *The effect of the mutation rate on the average population fitness*

It can be seen how the rise in average population fitness is initially high for all cases. The best average population is with the lowest mutation rate, but this does not find the global solution in all cases.

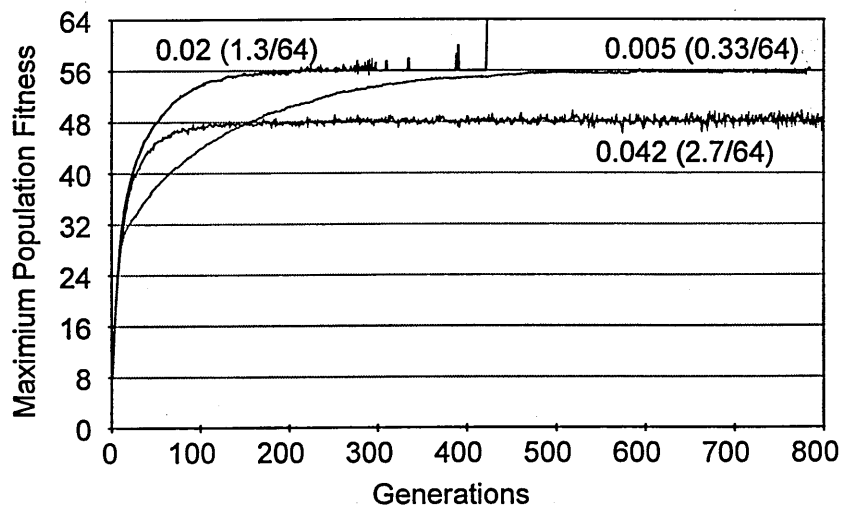


Fig 4-16 *The effect of the mutation rate on the maximum population fitness*

Note that the high mutation rate generally limits the maximum population fitness to 6 schema (a fitness of 48). This is because schema are destroyed as new ones are created.

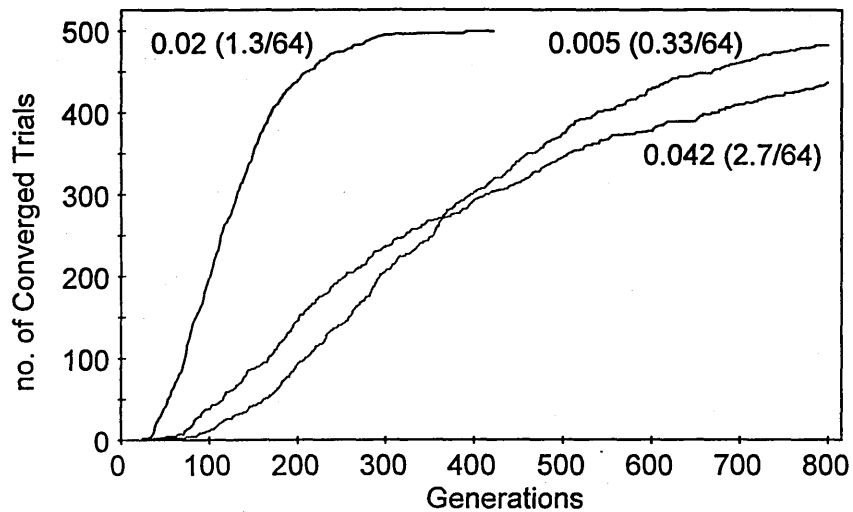


Fig 4-17 The number of converged trials at each generation for the three mutation rates

Quite clearly from a pure optimisation perspective, where the goal is to find the global solution, the mutation rate of 1.3/64 is superior in all respects, as shown by Fig 4-17.

Fig 4-18 to Fig 4-21 show the effect of the crossover probability for the near optimum mutation rate of 1/64. It can be seen that increasing  $P_c$  only improves the speed to convergence, with no other effect on the behaviour of the GA, as identified by all the lines converging to the same fitness value. In all cases every trial converged, even with  $P_c=0$ . The conclusion drawn is that mutation is the most important operator for this particular problem.

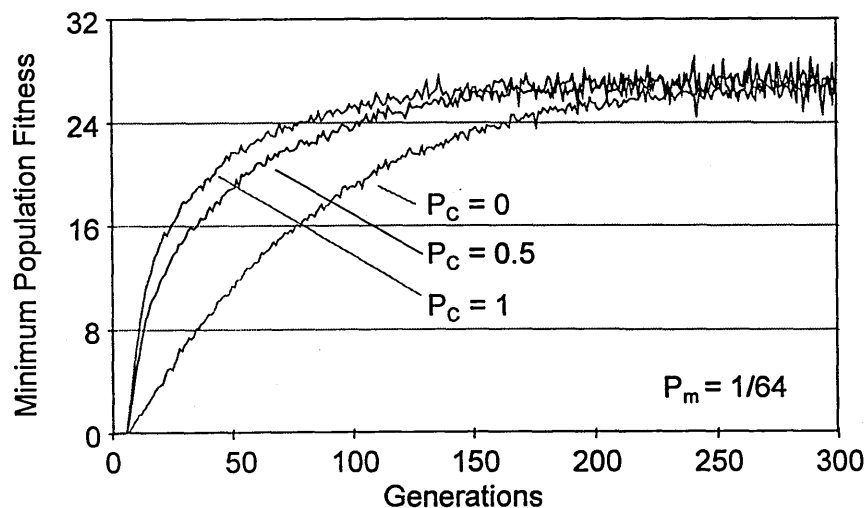


Fig 4-18 The effect of  $P_c$  on the minimum fitness

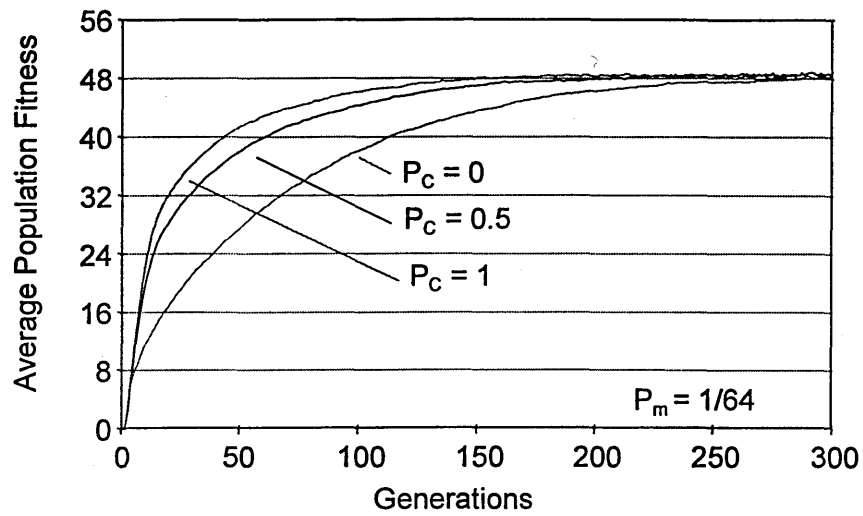


Fig 4-19 The effect of  $P_c$  on the average fitness

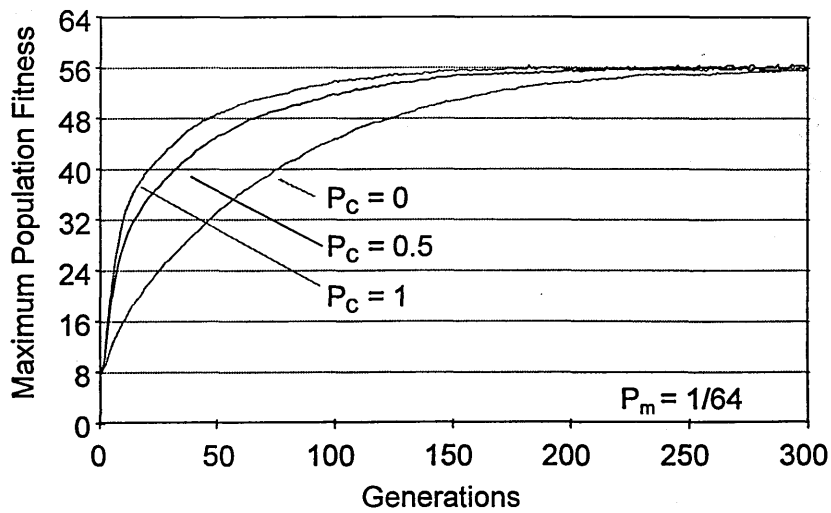


Fig 4-20 The effect of  $P_c$  on the maximum fitness

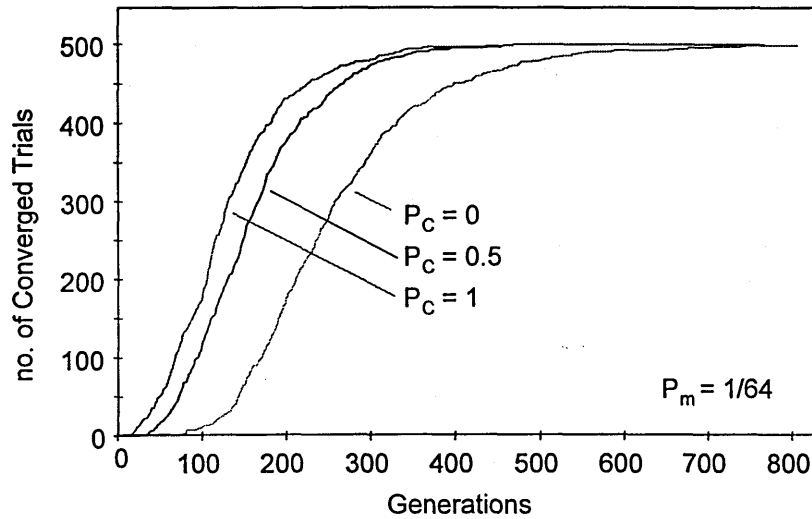


Fig 4-21 The number of converged trials at each generation for the three crossover probabilities

## 4.6 Chapter Summary

The work in this chapter has been an empirical investigation of parameters that affect GA performance. As commented in [73], *'there is a growing realisation that results obtained empirically are no less valuable than theoretical results'*.

What has been concluded is summed up in [74], *'From a function optimization point of view, GAs frequently don't exhibit a killer "instinct" in the sense that, although they rapidly locate the region in which a global optimum exists, they don't locate the optimum with similar speed'*.

This 'killer instinct' has been shown to be dependent on the mutation rate, which is critical for efficient GA performance in global optimisation. In humans, characteristics of individuals that enable them to stand out from the norm are often a result of mutation. This is exemplified by Veikko Hakulinen, a Finnish cross-country skier who won medals in the 50k, 30k, 15k and 4x10k relay at the 1956 winter Olympics. On medical examination it was found that he had an excessive red blood cell count that enabled him to take in more oxygen and not become out of breath. This was caused by a genetic defect with a probability of occurring equal to that of picking a specific light bulb with all the light bulbs on earth to choose from.

There has been much theoretical academic work in trying to improve the efficiency of GAs by optimising parameter settings. In other work, previously claimed ‘good’ settings are taken and used on totally unrelated problems. If GAs are to be used for function optimisation then a thorough investigation of the parameters is required.

It must be remembered that ‘*Genetic algorithms are NOT function optimizers*’ [74] and that other techniques do exist, that, although they do not sound as interesting, may be more appropriate for solving a particular class of problem. Optimising a system where there is no information on the dynamics (‘black box optimisation’) is essentially a directed random search, with the direction being guided by the strategy used. The purpose of these strategies is to guide the search to increase the probability that in time, a solution will be found. As was demonstrated (see Table 4-5), on average over many trials, random mutation hill climbing behaves exactly as a Markov chain analysis predicts. Nix and Vose [75] performed a similar Markov chain analysis for a simple genetic algorithm and claim that ‘*if the finite population is sufficiently large, we can accurately predict the convergence behaviour of a real GA*’.

Along with GA’s, simulated annealing [76] is another popular strategy for ‘black box’ optimisation that is inspired by nature. This is based around the fact that close temperature control must be maintained when cooling liquids into solids in order to attain a specific lattice structure. The most energy efficient lattice structure is obtained by very slow cooling and sometimes slight heating. This is reflected in the optimisation by only applying slight random perturbations and limiting the ‘temperature gradient’ (the amount of improvement allowed in new solutions). Successive solutions are also allowed to be ‘hotter’ (or worse) than previous attempts.

Many other optimisation strategies exist [77], including and tabu search [78] and branch and bound [79] (branch and bound methods are not strictly black box since they rely explicitly on the cost structure of partial solutions [80]).

In conclusion ‘*for any algorithm, any elevated performance over one class of problems is offset by performance over another class*’ [80].

In chapter 5 a variation of RMHC is used for the optimisation of a domestic hot water tank based on real-time pricing of electricity.



# 5

## **Domestic Hot Water Optimisation**

---

### **5.1 Introduction**

The objective of this thesis is to develop control strategies for electric thermal storage (ETS) systems under real-time pricing tariffs. The ETS devices under consideration are domestic hot water tanks and storage radiators. In the previous chapters the tools that are to be employed were investigated and chapters 5,6 and 7 evaluate the effectiveness of these tools in both simulation and actuality.

In this chapter the charging schedule for a hot water tank is optimised. Computer simulations using actual consumption data compare the real costs of an optimised schedule and existing charging schedules. Eleven houses are simulated for one month.

In chapter 6 the controller for a storage radiator is simulated. This uses a similar optimisation method to that used for the hot water tank, but introduces neural networks as a means of creating a thermal model from which to evaluate the candidate charging schedules. A ten week simulation compares the performance of the learning-optimised strategies to that of existing control options.

Finally in chapter 7, a storage radiator in a real room is controlled using a neural model predictive controller. Data was recorded for five months and an empirical neural thermal model of the room created. This model was then used to determine control set points five hours in advance to track a given room temperature profile, but with no optimisation. The controller was in continuous operation for 2 weeks.

Optimising ETS devices has been widely studied from various perspectives. In [81] the approach taken is to centrally control the water heating of blocks of houses, the main objective being to reduce peak load, a utility benefit. In [82] storage radiators are optimised for cost and comfort but using time-of-use (ToU) tariffs, genetic algorithms and a resistance-capacitance (RC) building thermal model.

Neural networks have also been used to model building energy consumption [83,84]. In [85] a recurrent neural network was used to model a crèche with a heated floor. The objective here was to optimise the start-up time so as to minimise energy consumption. A particularly ambitious project for using neural networks for domestic control is outlined in [86], where a house has been 'computerised'. Optimising the heating control is being attempted in simulation [87] but the initial work only used neural networks to predict occupancy with a RC model used to predict the building response. The planning horizon is 120 minutes.

Model predictive control using neural network empirical models rather than first principle models has been attempted in simulation mainly for the chemical process industries [88,89].

Any controller that is developed will ultimately rely on communication so that it can receive price and weather information. There will also be the need for half-hourly metering if real-time tariffs are to be introduced. Such technology is already available and under trial in domestic houses [90]. Actual experiments in the logistics and hardware requirements of real-time control for thermal storage have been performed as far back as 1989 [91,92].

This thesis is concerned with the development of control technology that is required to make real-time pricing feasible. An analysis of such tariffs is not given but sources for reference are [1,2,3,4,93,94,95,96,97]. What has to be considered is that using a prediction of the next day's demand sets the daily pool price. If this demand has the potential to adapt to the set price then the initial forecast is wrong. Will this have the desired effect of flattening the demand profile?

## 5.2 Model to be Optimised

Fig 5-1 shows the water heating system to be optimised. For each half-hour period there is a demand (litres) and price (pence/kWh), profiles of which are given at midnight for the following 24 hours. The criterion to be satisfied is that the demanded water (in the 24 hours following midnight) must be supplied at a set temperature in the cheapest manner. Two heating elements exist, one in the storage tank (element 1) and one at the outlet (element 2). The latter is to ensure adequate supply temperature ( $T_{\text{required}}$ ) and can be supplied with warm water from the storage tank via tap 1 or ambient water from the mains via tap 2. For a 24-hour period of known demand and price, the challenge is to determine for each half-hour the water source (tap1/tap2) and the state of heating element 1 (on/off) that will give the cheapest cost. Heating element 2 is not controlled but delivers the required amount of energy to maintain the delivered water at  $T_{\text{required}}$ .

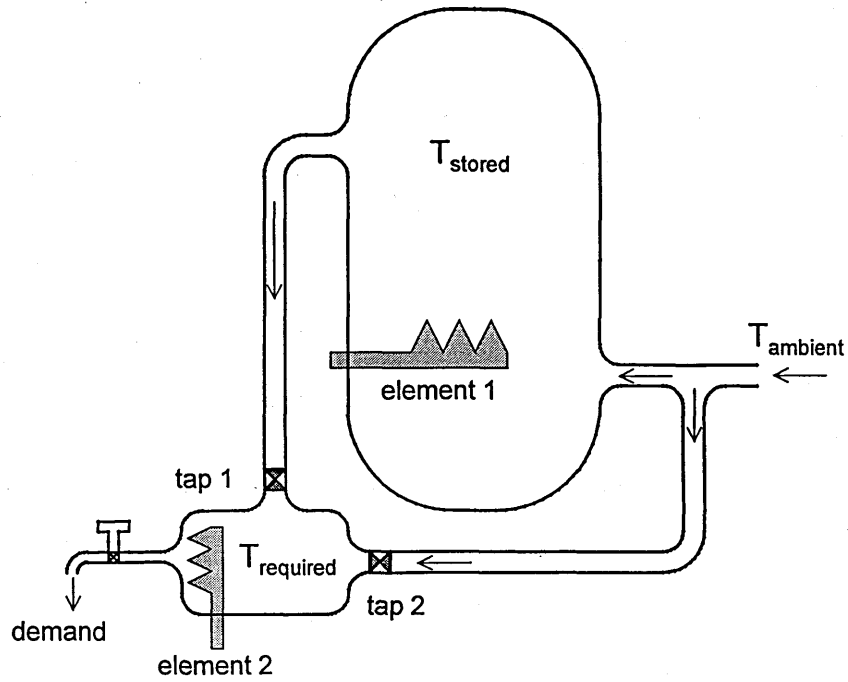


Fig 5-1 Schematic of the hot water system

For each half-hour two decisions have to be made,

1. If there is demand then shall the source be tap 1 or tap 2?
2. Shall the tank be charged by activating element 1?

There are thus two options for each decision. If water is consumed in all of the 48 periods during a day then there are  $2^{(48 \times 2)}$  potential solutions, of which the cheapest is sought, as depicted by Table 5-1.

Table 5-1 The control sequence to be optimised for the water-heating model

time slot 1		time slot 2		::	time slot 48	
decision 1	decision 2	decision 1	decision 2	::	decision 1	decision 2
Source?	Charge?	Source?	Charge?		Source?	Charge?
tap 1 ?	yes ?	tap 1 ?	yes?	::	tap 1 ?	yes ?
or	or	or	or		or	or
tap 2 ?	no ?	tap 2 ?	no ?		tap 2 ?	no ?

### 5.3 Simulated Water Heating Model

A simplified computer model of Fig 5-1 was created to determine the fitness of each candidate solution, which is the total cost over the 24 hour period (see appendix E for an example of the code used). The process was continuous in that the final tank temperature (time slot 48) was used as the starting temperature for the following day. The absolute accuracy of the model compared with a real hot water tank is not vital since the comparative costs of the existing schedules are being simulated from the same model. Several assumptions and rules were made to simplify the model,

- 1) No heat loss from the tank.
- 2) Complete mixing of water in the tank so it is always at a uniform temperature.
- 3) All demand is given instantaneously at the start of each half-hour, charging commencing on the recalculated tank temperature.
- 4) Charging stops when the tank water reaches the set point (demanded) temperature (70° C).
- 5) All water being delivered is topped up to the set point temperature by the direct acting electrical element (element 2) costing whatever the price is in that specific half-hour.
- 6) In the simulations for the existing charging profiles (E7 and E10) all the water was delivered from the tank via tap 1 and extra heat was added from heating element 2 if it was below the required temperature.

The E7 charging profile used is 00:00-07:00. The E10 profile is 02:30-07:00, 12:30-15:00 and 19:00-21:30. Element 1 was set at 2kW. Although heating element 2 is generally not present in domestic tanks it is required so that a fair comparison can be made between the charging schedules, as it ensures all schedules deliver water at the required temperature so that comfort is guaranteed.

## 5.4 Data Used

The data (purchased commercially) used in these simulations originated from 100 houses monitored over the course of a year. Each water outlet was logged every half-hour and from this all hot water outlets were grouped to find the total hot water demand in each half-hour period. There was no indication in the data of how the water was actually heated. Eleven houses were randomly chosen for simulations, which were performed for November 1994, with the corresponding actual pool selling price (PSP) used to calculate the cost.

Fig 5-2 shows actual hot water consumption and PSP over four days for a particular house. It can be seen that there are small price peaks just after midnight caused by the surges due to the existing E7 and E10 tariffs. This is even more pronounced on Saturday when the early morning price is almost as high as the maximum price for that day. The difference between weekdays and weekends can also be seen, with the weekend price generally lower because of reduced overall demand. The high peak on Thursday occurs at evening meal time and is a result of increased domestic heating, lighting and cooking electricity consumption. The water consumption tends to be concentrated between 8am and 10am that can be a period of high price. The consumption pattern on Sunday is spread throughout the day, highlighting how usage is related to lifestyle.

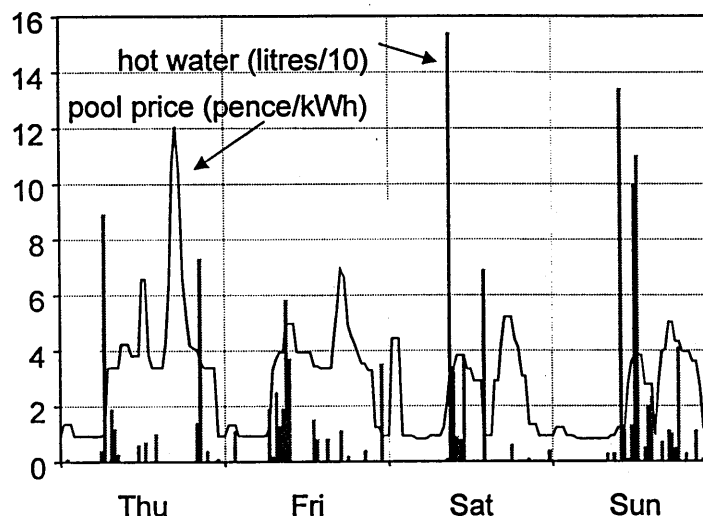


Fig 5-2 Actual pool price and hot water consumption from a random house for four days in November 1994

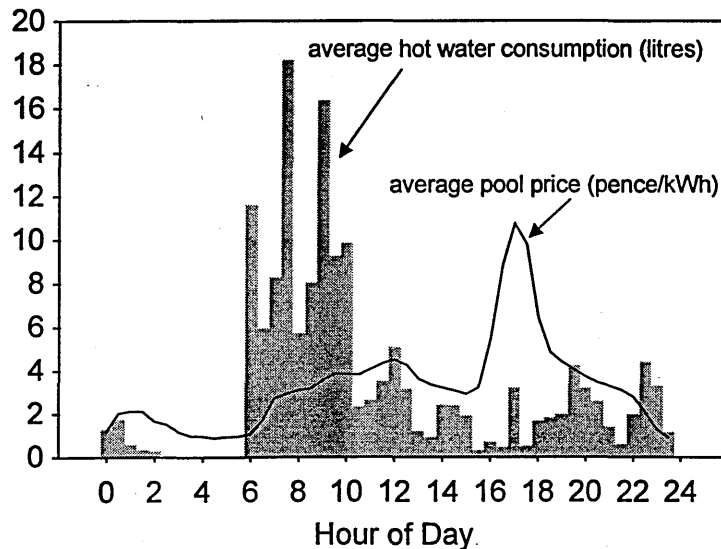


Fig 5-3 The November daily averaged price and consumption for the same house

Fig 5-3 shows the average daily consumption profile for the same house throughout November and the corresponding average price. The morning water consumption is emphasised (hours 6-10), as are the early evening and midnight peaks in pool price.

These two figures show that although on the average things look predictable, on a day-to-day level there is much variation and potential for customised control strategies.

## 5.5 Optimisation Procedure

The optimisation technique used was based on random mutation hill climbing, as described in section 4.5.3.3 on page 86. As well as proving superior in performance to GAs it is also more desirable from a controller memory standpoint as only two solutions have to be stored, as opposed to many if GAs were used.

In the original version of RMHC only one bit change is allowed between successive potential solutions, which means that it is unlikely to escape from any local minima. Three bit changes were introduced to overcome this potential problem. Introducing more than one bit change also has the effect of speeding up the process. This is because if there is no demand in any particular half-hour then the choice of tap 1 or tap 2 is

irrelevant and a bit flip will make no change to the solution. Pre-processing the string to eliminate redundant bits would reduce the search space but require more processing power.

In any stochastic (i.e. having an element of chance) search procedure, there is no guarantee that the global optimum solution will be found. Once a solution had been given adequate time to reach a steady value, it was found more beneficial to restart the search as opposed to continue searching from the current position. In the optimisation the search was repeated three times with the best overall solution used. Each search consisted of 2,000 evaluations, with the whole process taking about 7 seconds on a P133 to optimise all 30 days.

The hot water tank is an example of a system where one change can have a profound effect on the outcome. If the tank is at its maximum temperature then the thermostat in the model will ensure that no more heating is allowed, regardless of the control signal to the element. For instance, if there is no demand all control signals for heating the tank would be ignored once it was at its maximum temperature. A change early in the day could result in a previously ignored signal becoming active. This makes the search more random rather than gradient based. To overcome this, all signals indicating that the tank should be charged were reversed if the tank was already at its maximum temperature.

The initial starting point can affect the search procedure, especially if there is low demand throughout the day. To capture this possibility the initial guess is 'do not charge the tank at all', for which the associated cost will be that of using direct acting heating to satisfy the requirement. This is often the cheapest solution if there is low demand as it saves heating the tank and having excess hot water at the end of the day.

## **5.6 Profiling Usage Patterns**

In the optimisation process the actual half-hourly consumption data was used. In reality, the controller will have to use estimated values on which to base the optimisation. This



could be done via a keypad, with the occupants entering times at which they are likely to take showers, baths or use washing machines. An alternative method is to use past consumption patterns to make educated guesses as to a likely profile for the following day.

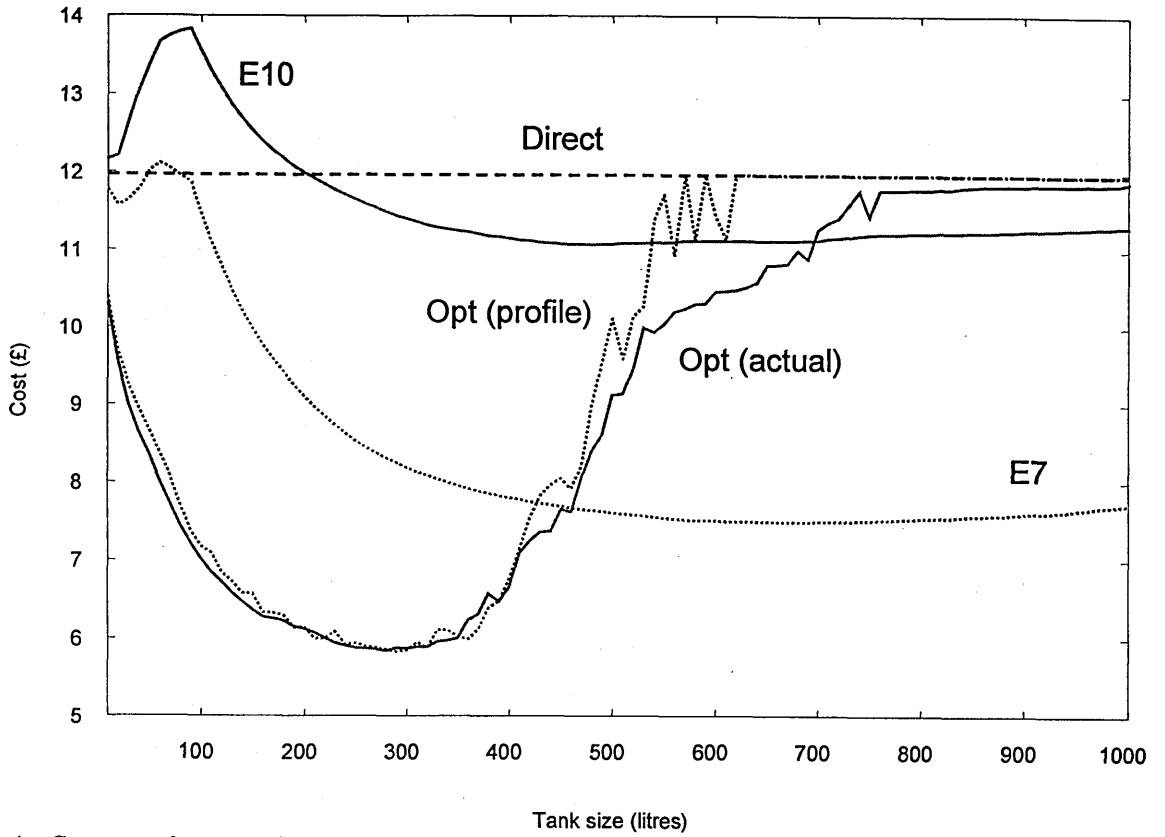
If consumption is to be predicted based on previous occurrences, it has to be assumed that there is some cyclical pattern involved. This is likely to be a predominant daily cycle with an underlying weekly cycle, which life generally revolves around.

A simple method of predicting consumption is to take an average value of volumes that occurred in the same half-hour of previous weeks. The method actually employed was to use a neural network to create a curve fit with daily and weekly components. This was achieved by having inputs representing hour-of-the-day and day-of-the-week, appropriately coded as sines and cosines in order to achieve the cyclic pattern. Each unique combination of inputs thus had four output values for the four weeks of data available, an ill-posed problem. This has the effect of basically averaging the consumption but fitting a generally smooth curve through the data, achieved by limiting the number of hidden neurons. 15 hidden neurons were used in this case.

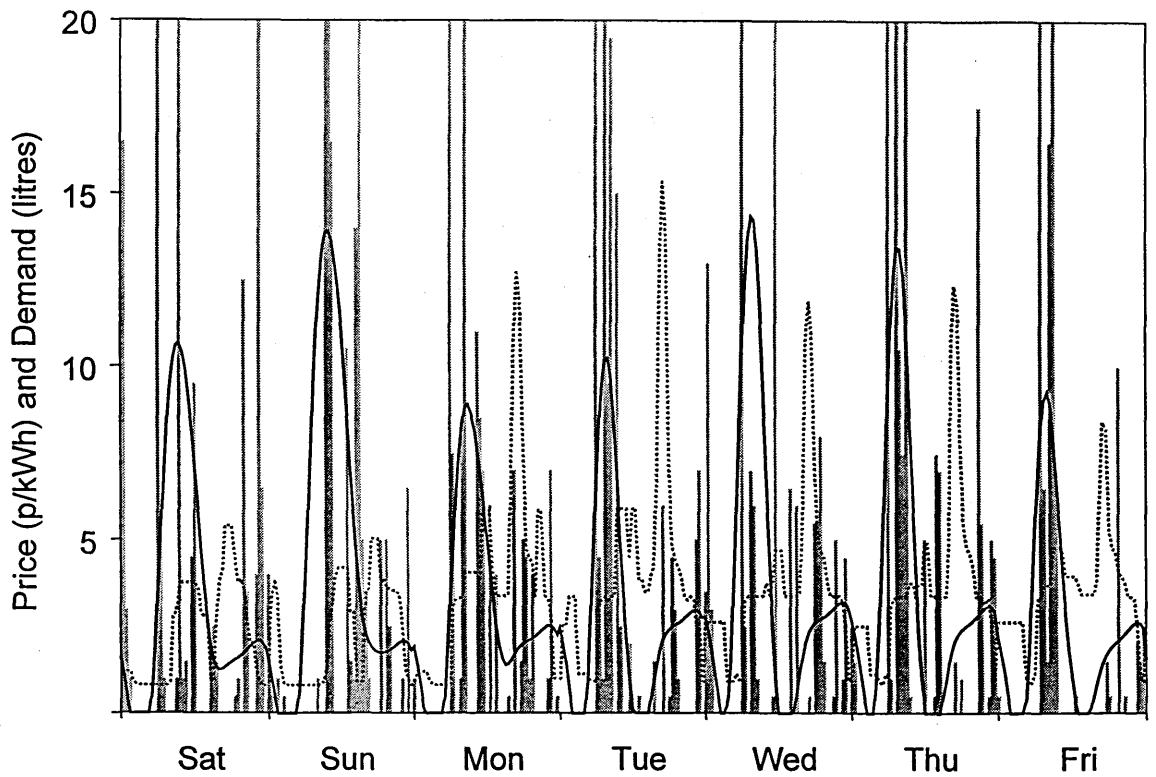
The resultant profiles were used to optimise the heating system and the costs calculated by then using the actual consumption patterns. Because only four weeks of data were used the actual consumption figures for any half-hour contribute to the predicted profile. A more realistic test would be to use a running profile and use it for the week ahead, with the prediction day's data not being involved in creating the profile.

## 5.7 Results

Consumption data from 11 houses for November 1994 was simulated for boiler sizes of 10 to 1000 litres. Costs for E7, E10 and direct acting only heating strategies were also recorded. The results are shown Fig 5-4 to Fig 5-14. The actual demands in b) are truncated at 20 litres and the x-axis in a) starts at 10 litres.

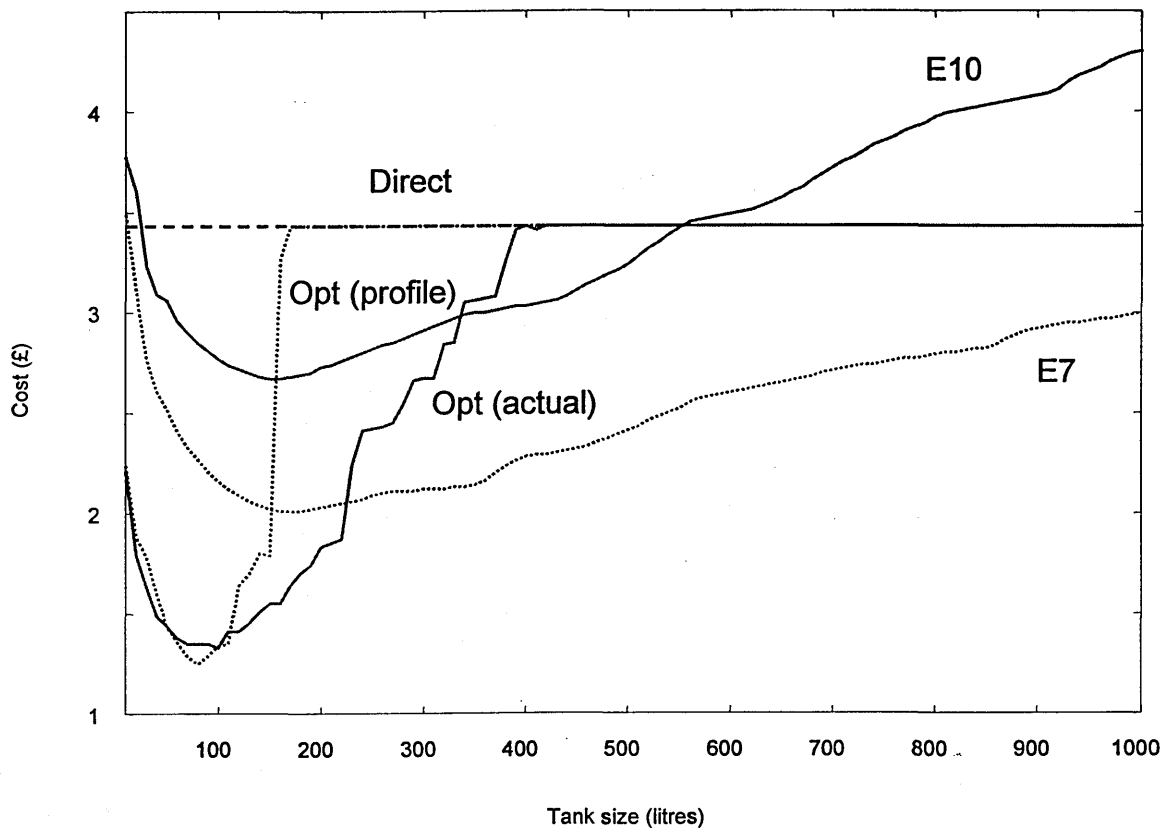


a) Costs as a function of tank size

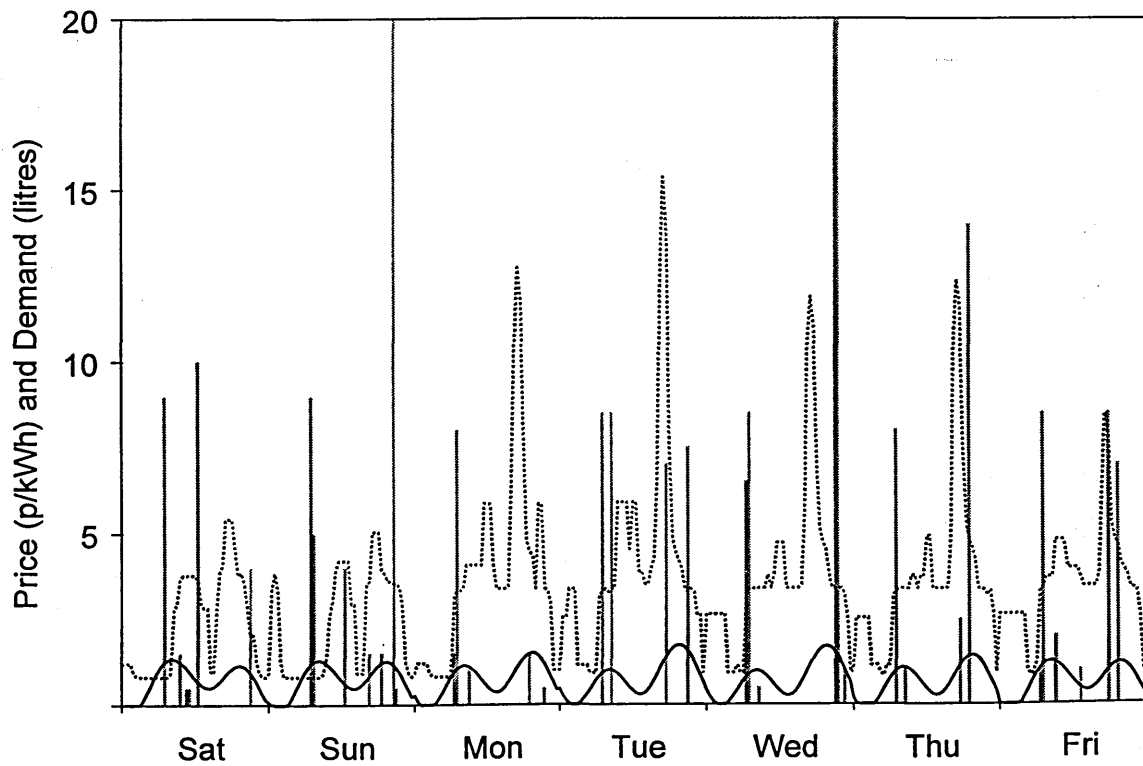


b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-4 HOUSE 1 mean daily demand = 157 litres

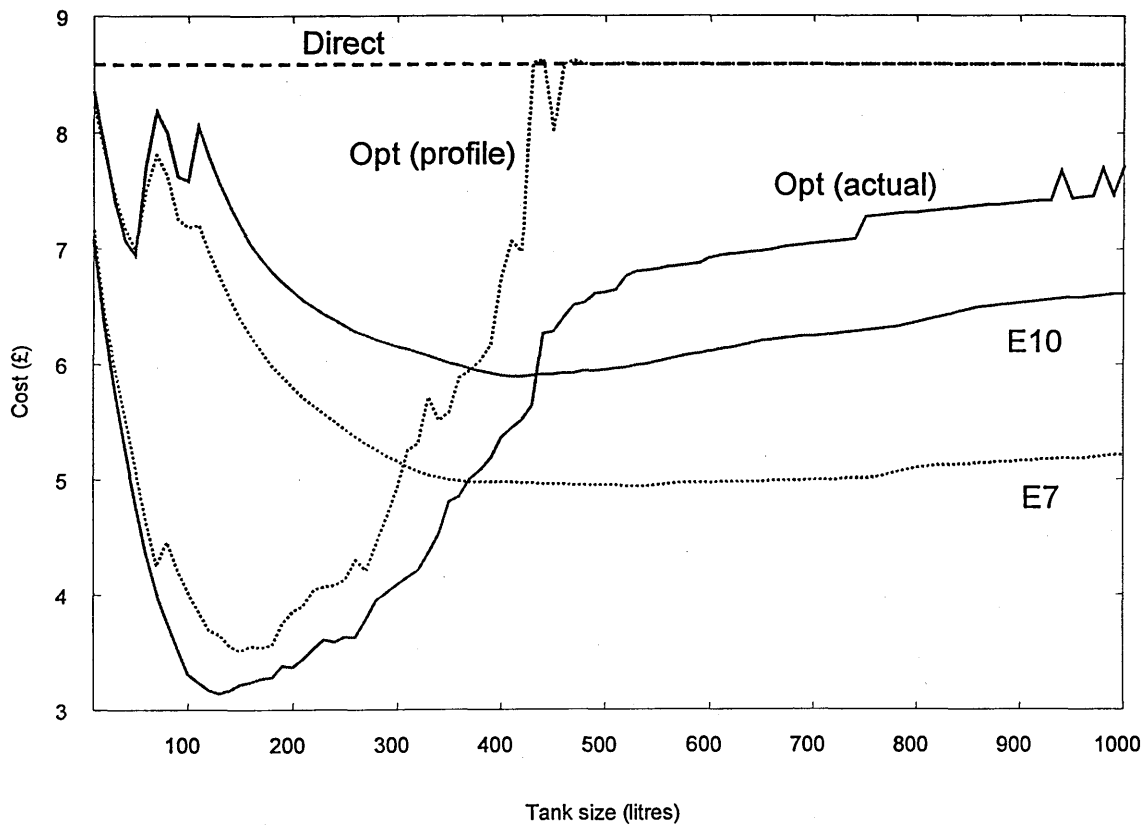


a) Costs as a function of tank size

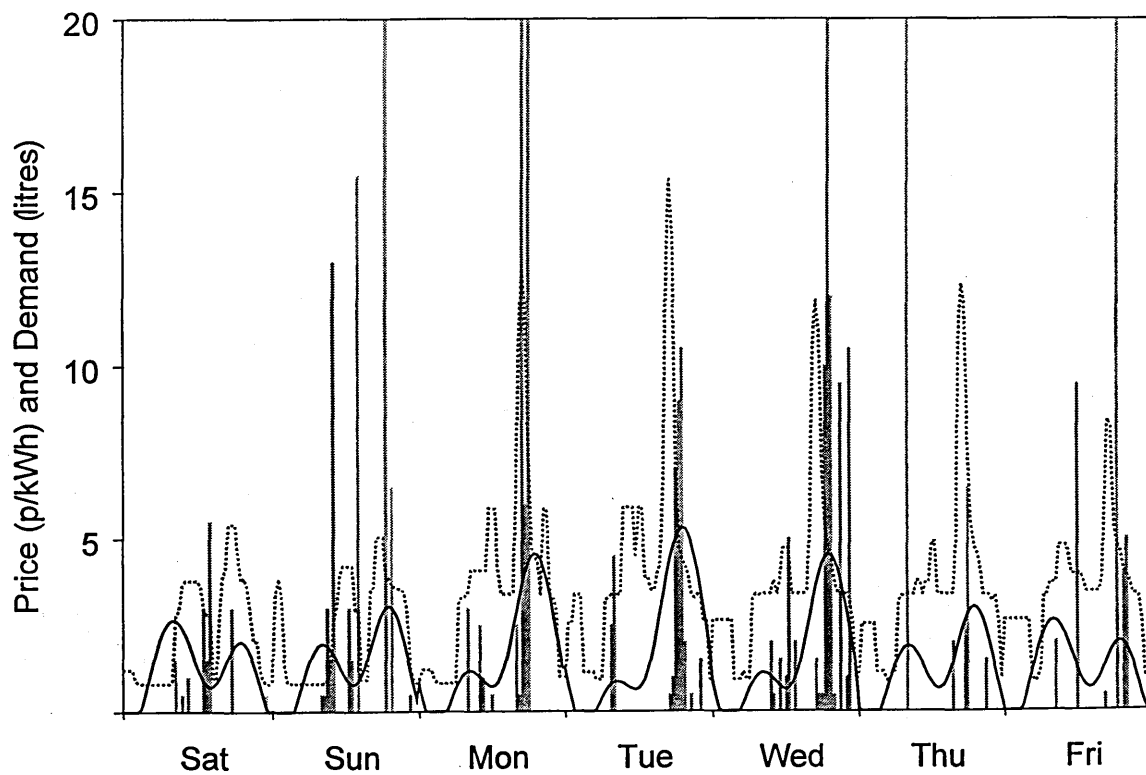


b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-5 HOUSE 2 mean daily demand = 36 litres

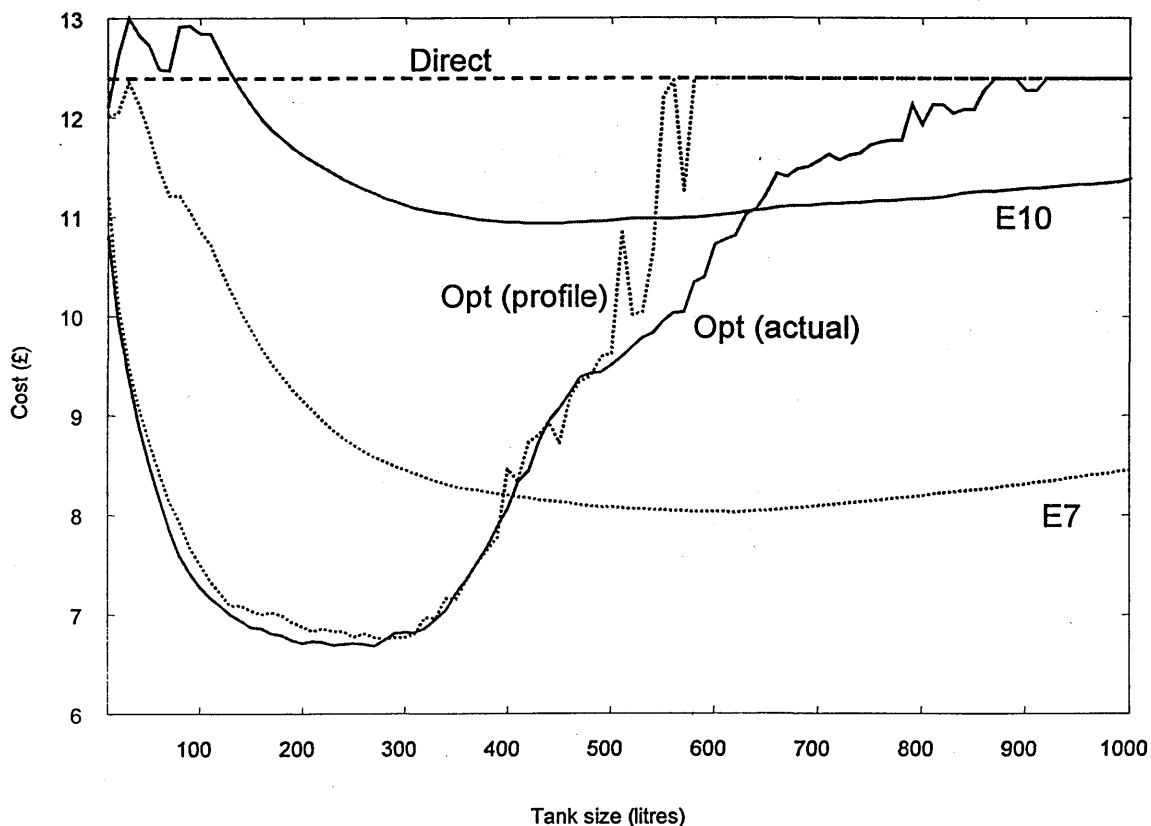


a) Costs as a function of tank size

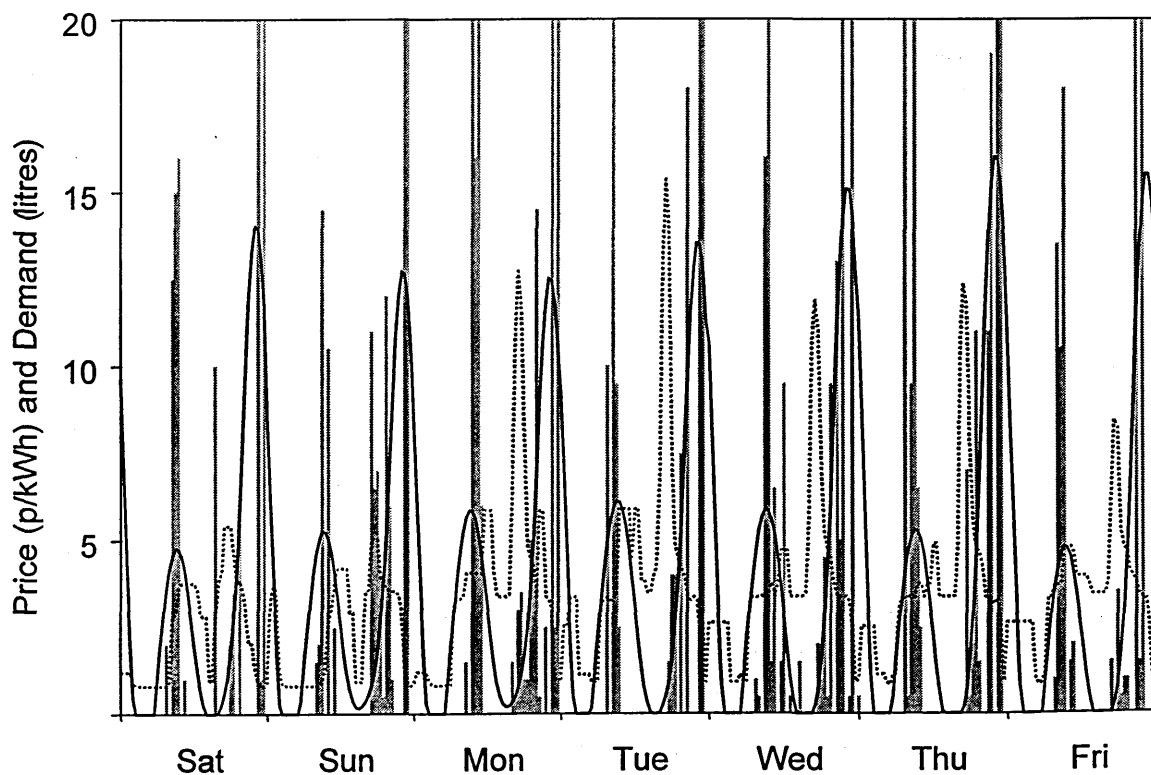


b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-6 HOUSE 3 mean daily demand = 74 litres

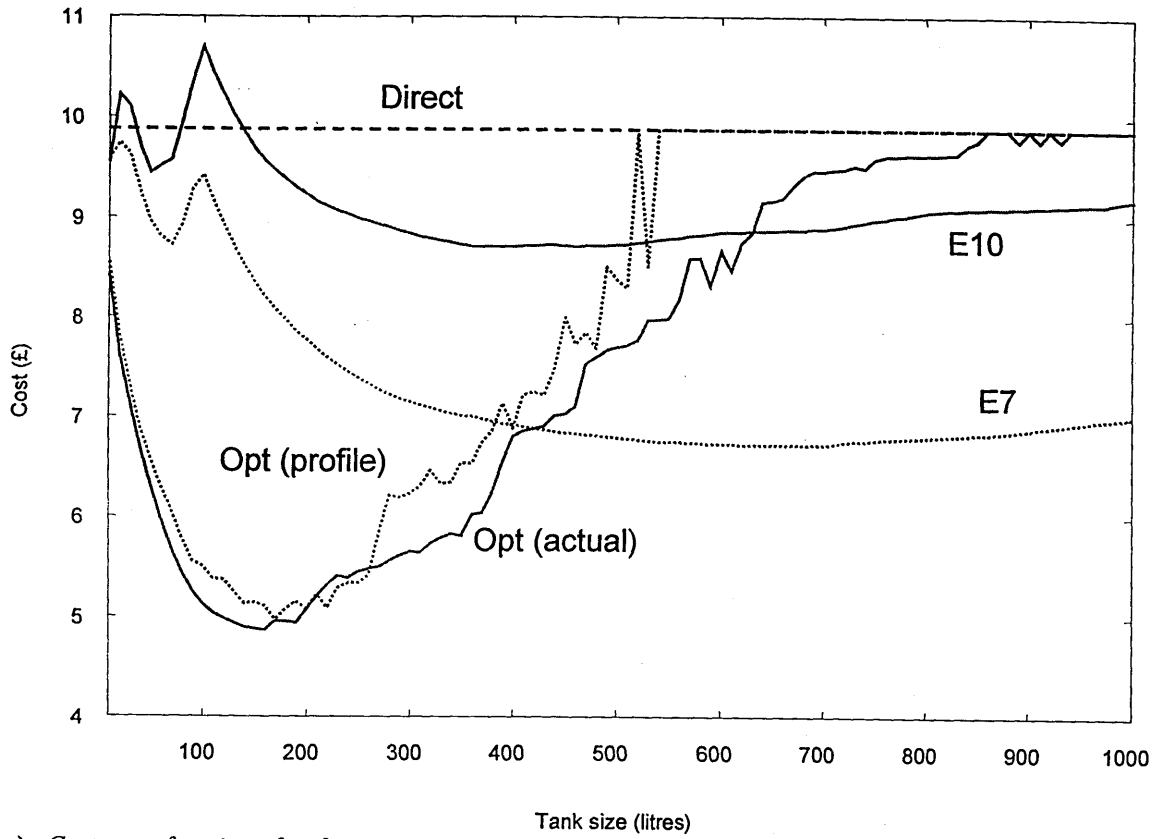


a) Costs as a function of tank size

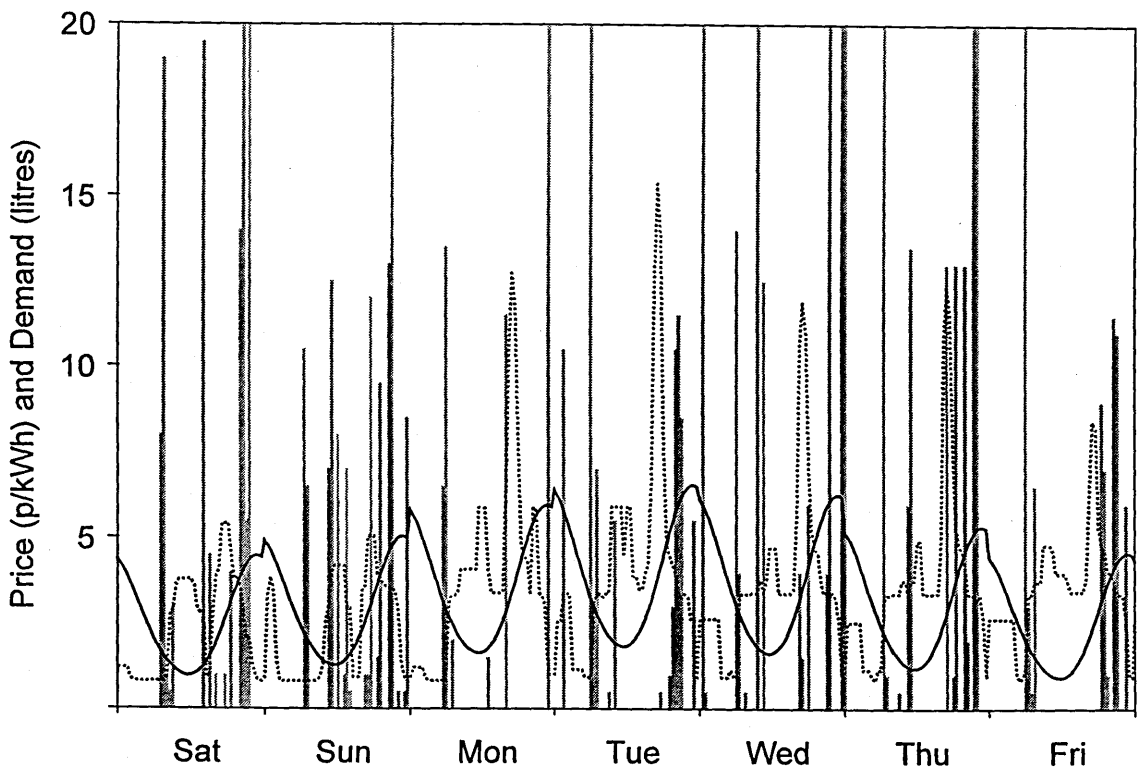


b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-7 HOUSE 4 mean daily demand = 186 litres

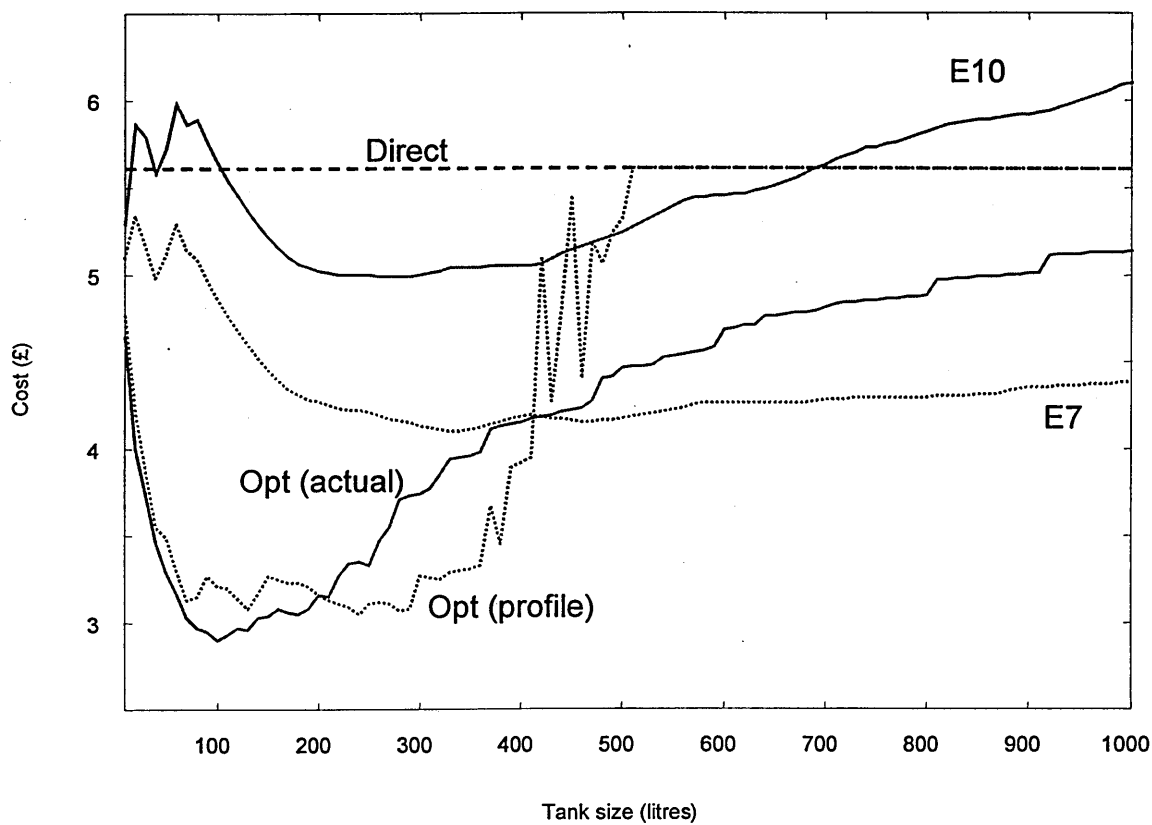


a) Costs as a function of tank size

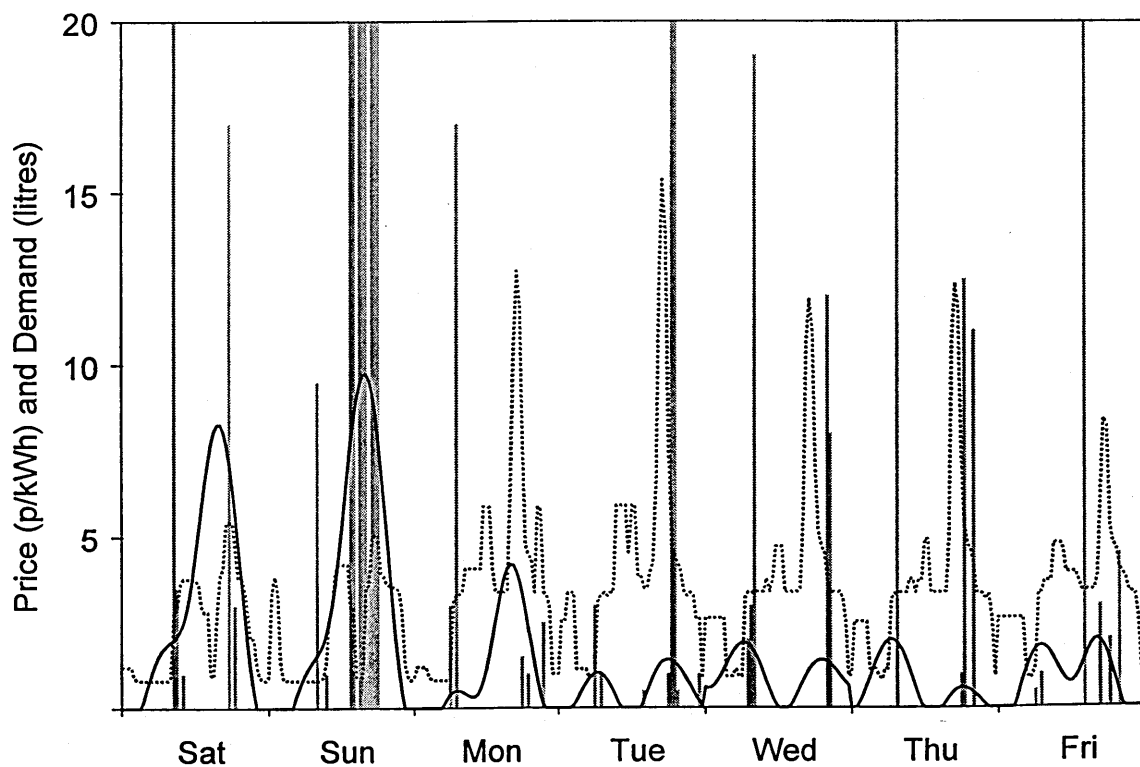


b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-8 HOUSE 5 mean daily demand = 150 litres



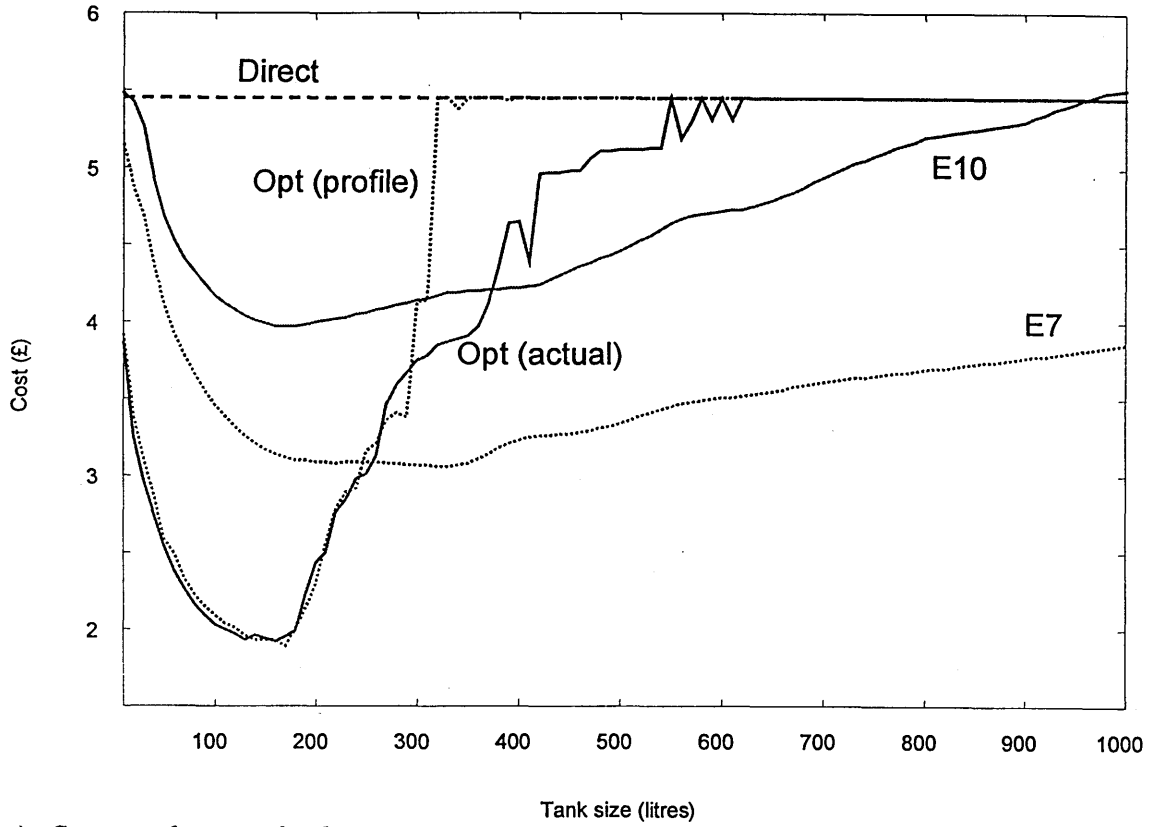
a) Costs as a function of tank size



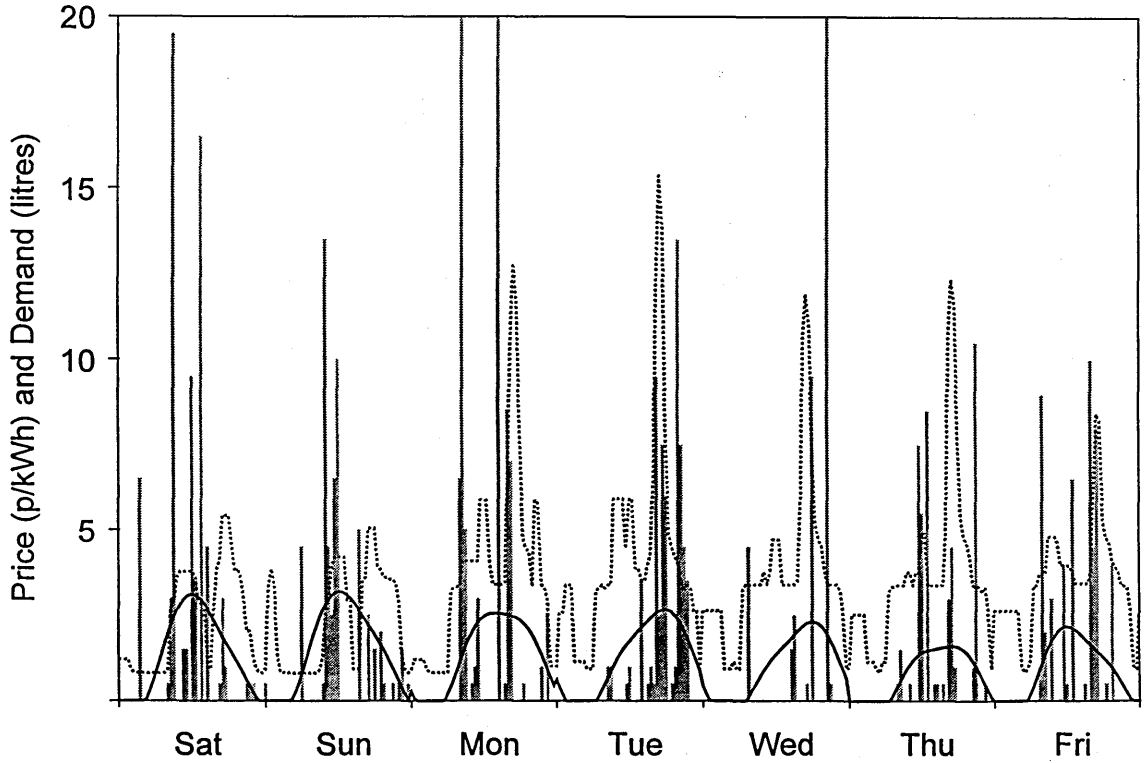
b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-9 HOUSE 6

mean daily demand = 71 litres



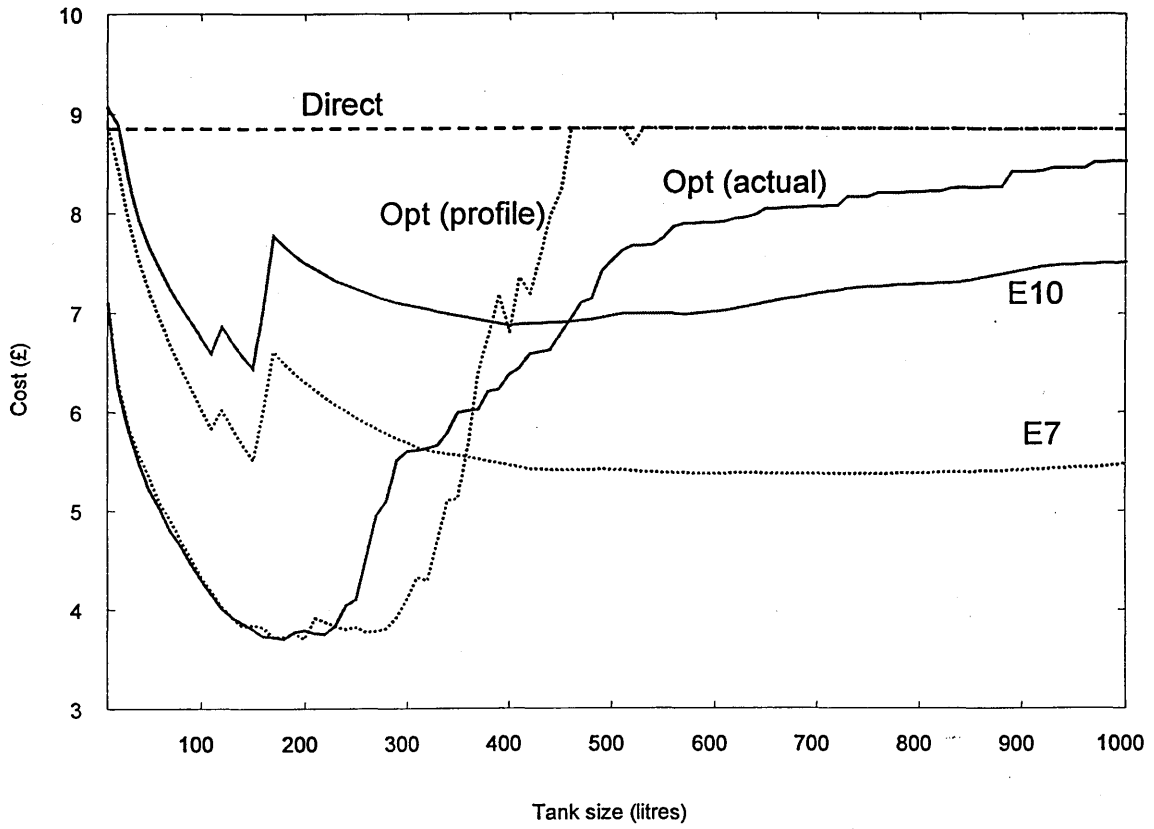
a) Costs as a function of tank size



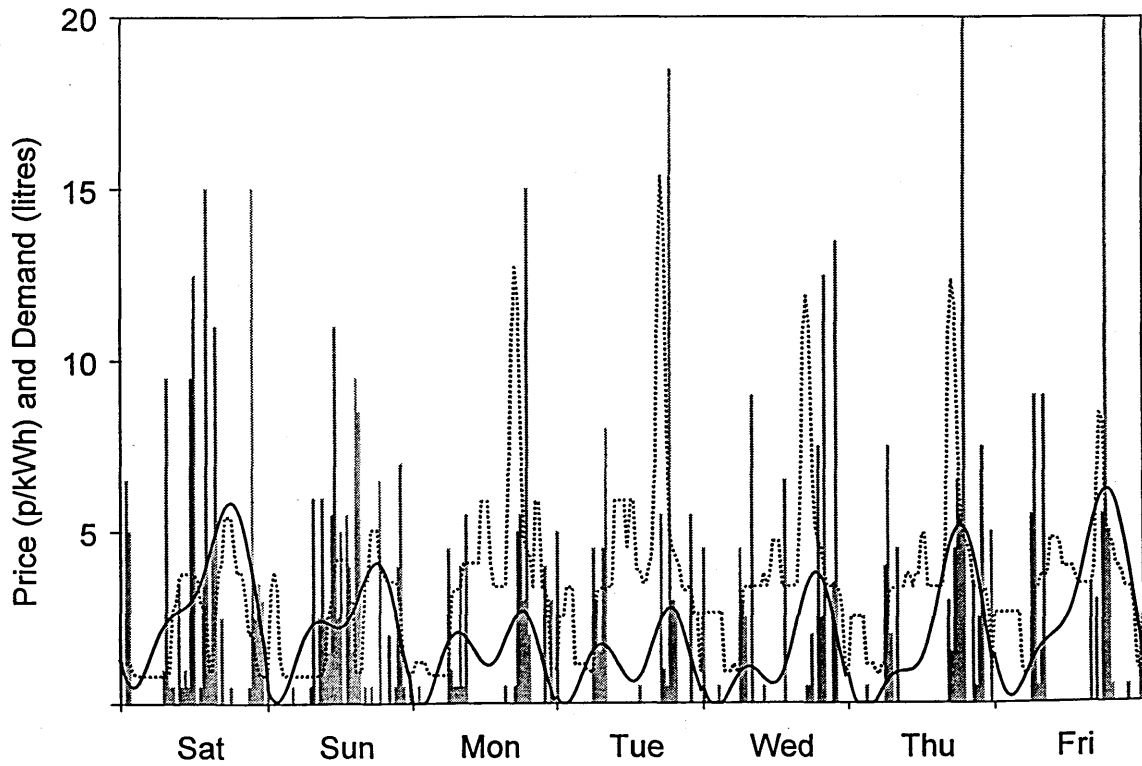
b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-10 HOUSE 7 mean daily demand = 55 litres



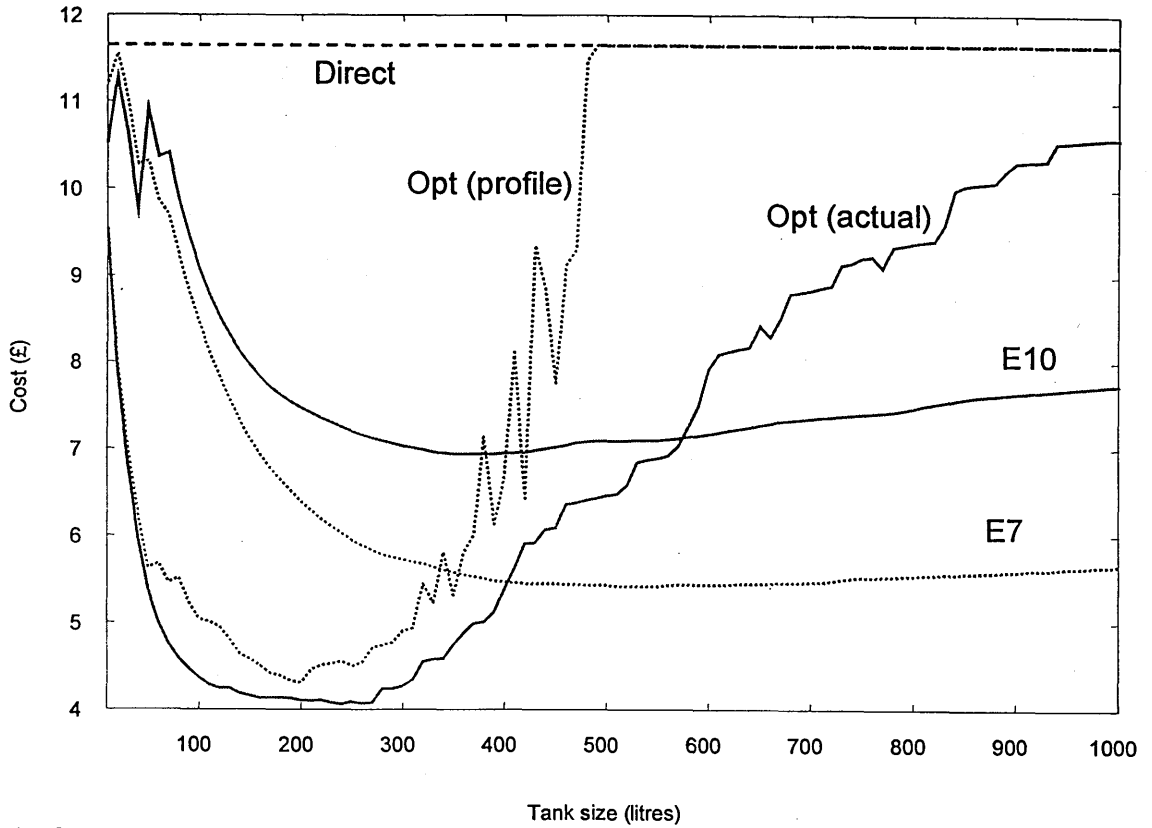


a) Costs as a function of tank size

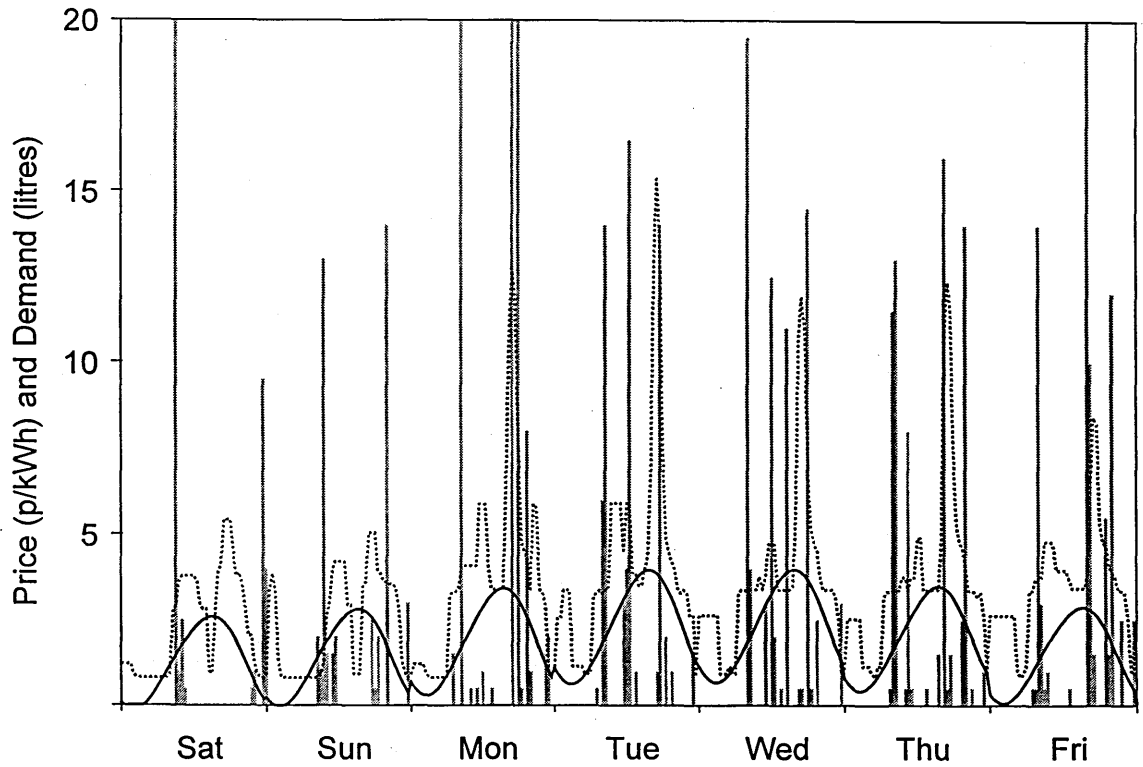


b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-11 HOUSE 8 mean daily demand = 96 litres

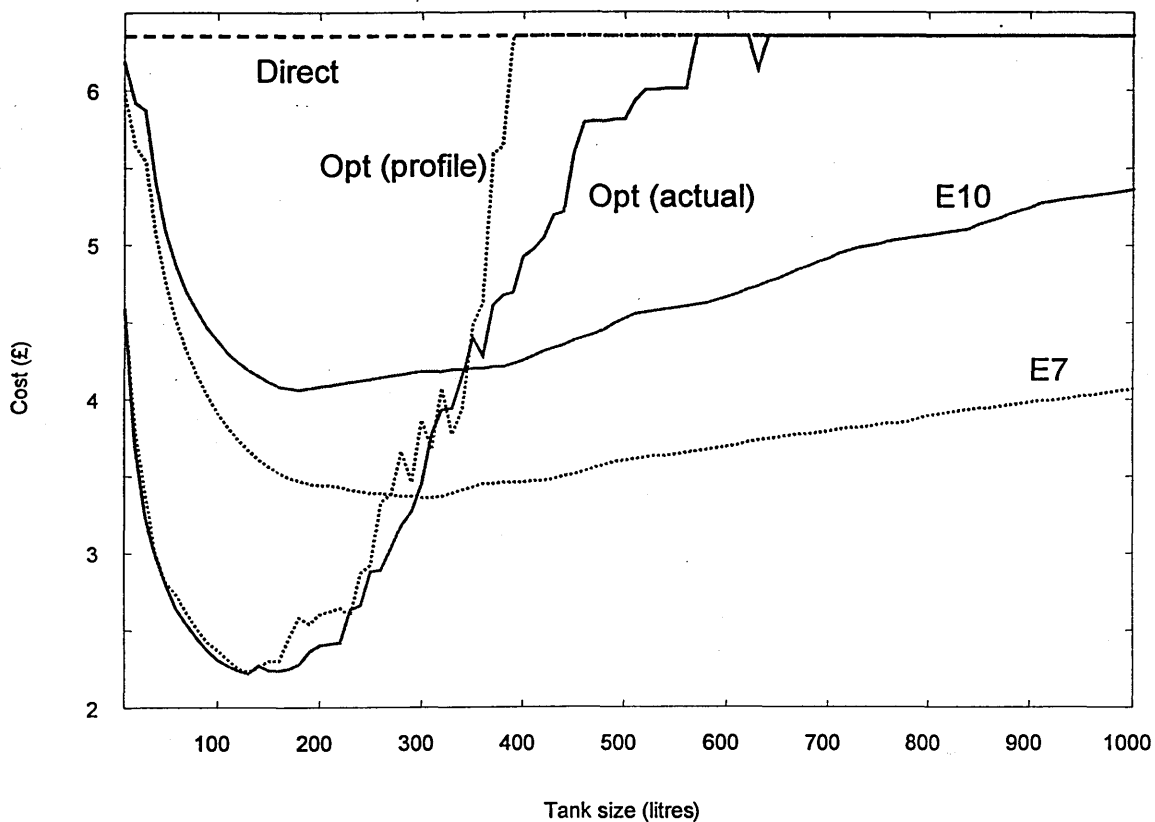


a) Costs as a function of tank size

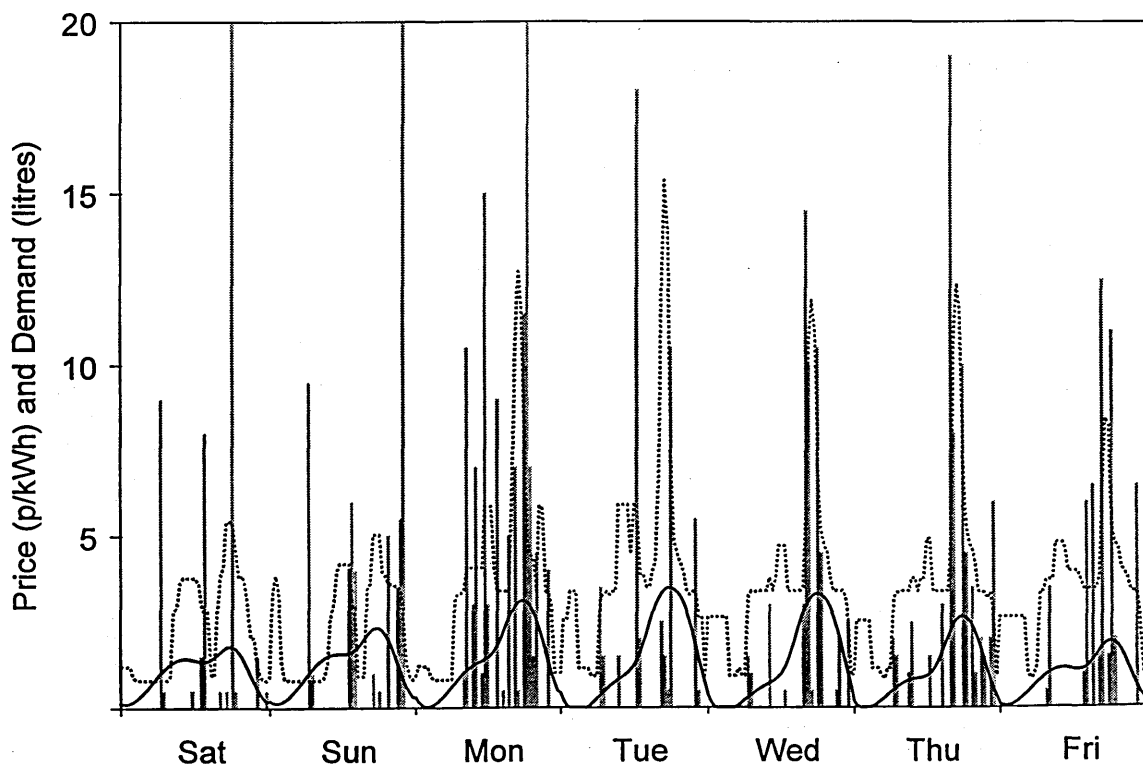


b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-12 HOUSE 9 mean daily demand = 86 litres

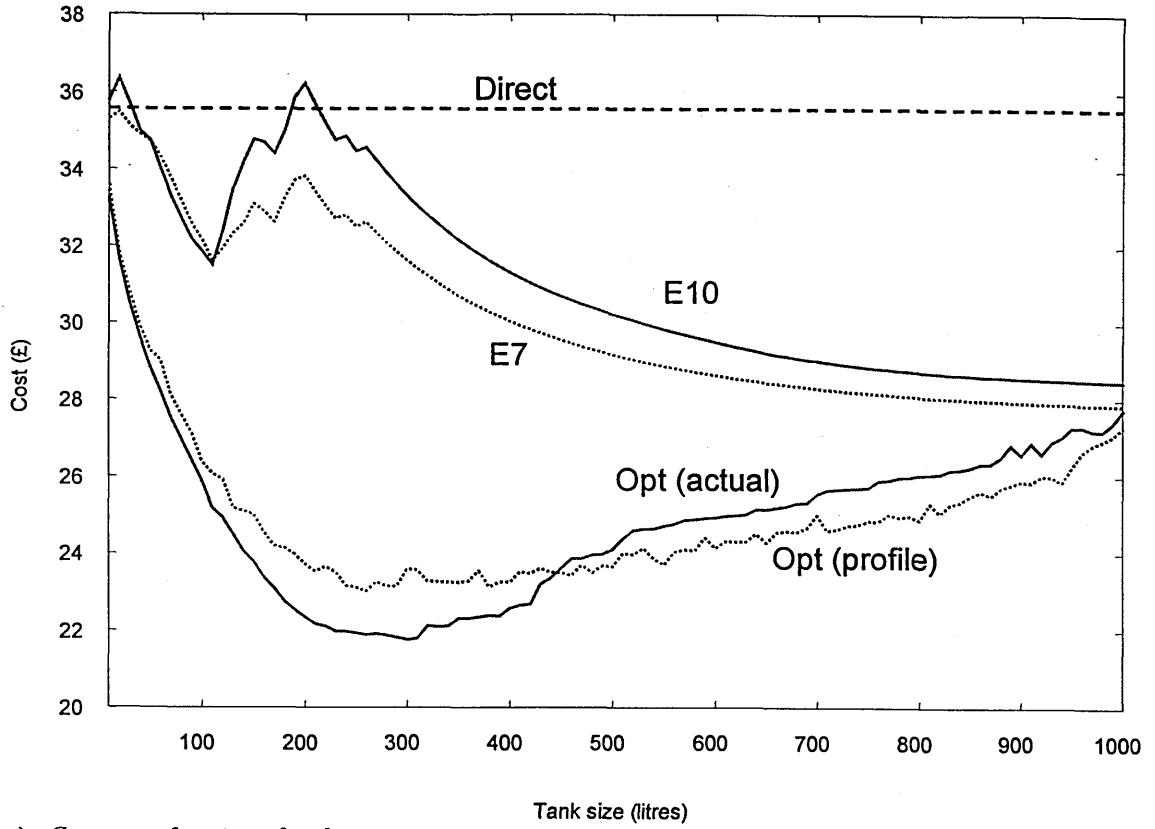


a) Costs as a function of tank size

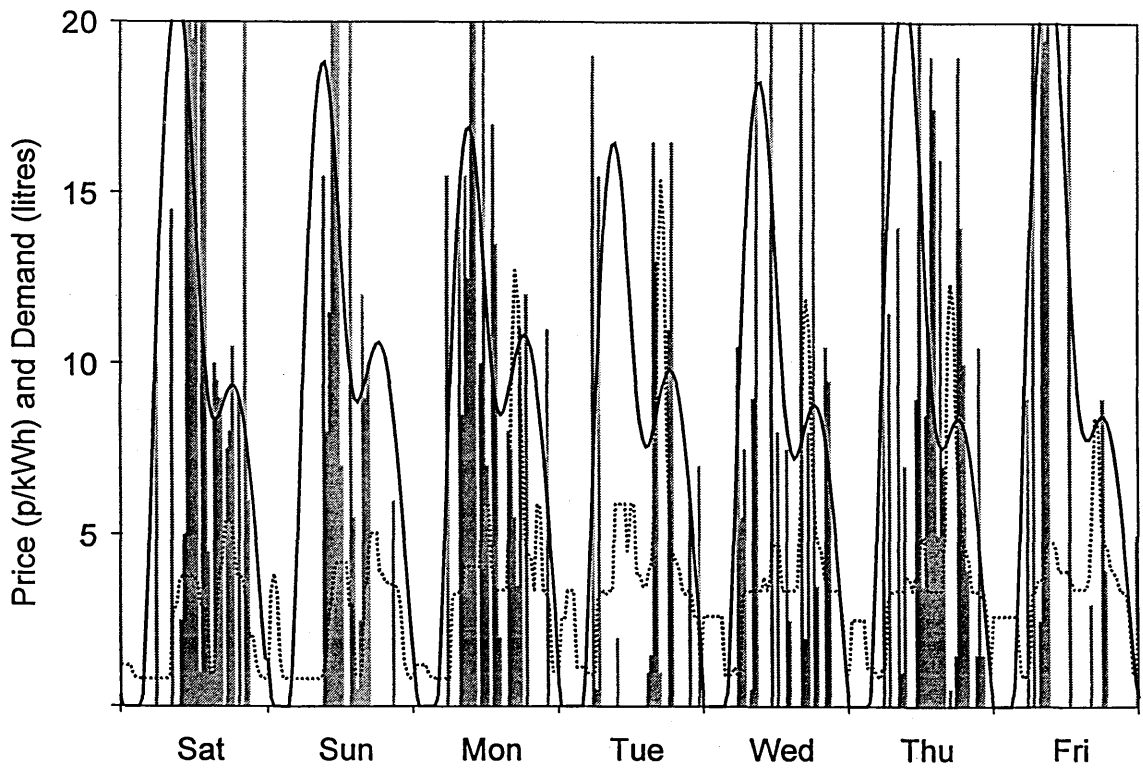


b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-13 HOUSE 10 mean daily demand = 57 litres



a) Costs as a function of tank size



b) The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)

Fig 5-14 HOUSE 11 mean daily demand = 395 litres

## 5.8 Discussion of Results

### 5.8.1 Does Water Storage Save Money ?

If storage tanks did not exist then the water must be heated on demand at the cost of the current pool price. The cost of this option is shown in Fig 5-4 a) to Fig 5-14 a) by the straight line labelled 'direct'.

In all cases the E7 and the optimised charging schedules are cheaper or as cheap as direct acting heating. E10 is generally cheaper but depends on the tank size and demand levels. House 2 shows that for a very low demand excessive charging with large tanks is wasteful.

### 5.8.2 How did the Profiling Perform ?

Two methods of optimisation were performed. The first was to use a predicted '*Opt(profile)*' daily demand and the second was to use the actual '*Opt(actual)*' demand. After the optimised schedules were derived the costs were then calculated based on the actual demand.

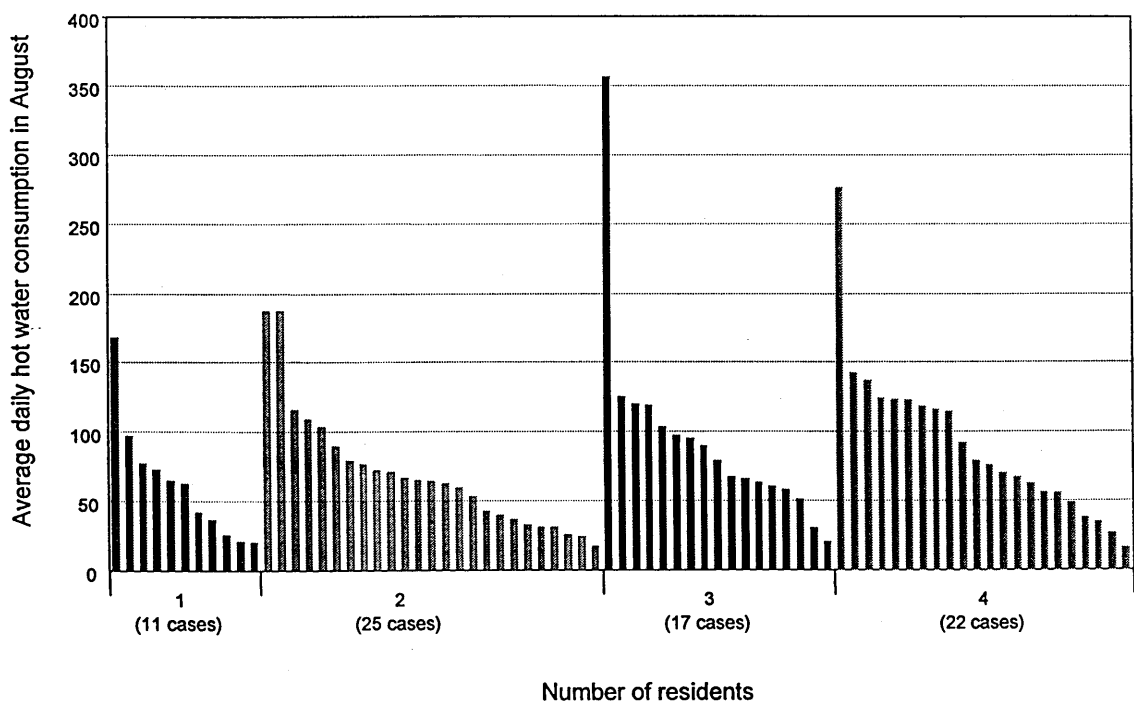
Fig 5-4 a) to Fig 5-14 a) show that using the profiled demand compares very favourably to using the actual demand. Generally as the tanks get larger the actual demand is required to give a cheaper solution. This is because the more continuous nature of the profiled patterns will result in the tanks being over charged at times of low demand. This cannot be avoided as there is more storage capacity and hence increased energy consumption.

In the case of house 6 the profiled optimisation costs were cheaper than the actual optimisation costs for tank sizes of 200 to 400 litres. This means that the optimisation procedure did not perform satisfactorily when using the real data. By looking at the demand profile for week 1 (Fig 5-9 b) it can be seen that water is only consumed in about 5 half hours of the day, immediately making 44% of the search space redundant.

By using the profiled consumption there is predicted demand in most half-hours, which assists the search procedure. For instances similar to this the search space could be severely reduced by eliminating the redundant bits.

The individual profiles in Fig 5-4 b) to Fig 5-14 b) show a wide variation from house to house and there is no 'typical' profile. The profiles are similar to a smoothed time-averaged demand, and in all cases the total profiled demand was within 5% of the actual total demand.

It might be suggested that 'group' profiles could be created for specific users, which would alleviate the need to measure actual consumption. The groups might be related to the number of residents, but as Fig 5-15 shows there is no obvious relationship between the number of residents and average daily consumption.



**Fig 5-15** Mean daily hot water consumption related to the number of residents

### 5.8.3 How Much Money could be Saved?

Fig 5-16 shows the percentage cost savings over E7 for the 11 houses. The data is from the optimisation results using the profiled consumption patterns, which is close to what could be achieved in reality. For most houses savings of between 20-40% are possible for tank sizes between 50 and 250 litres. If it is assumed hot water accounts for 40% [81,90] of the domestic electricity bill then this relates to savings in the range 8-16%. Existing domestic tank capacities are within the range 100-250 litres.

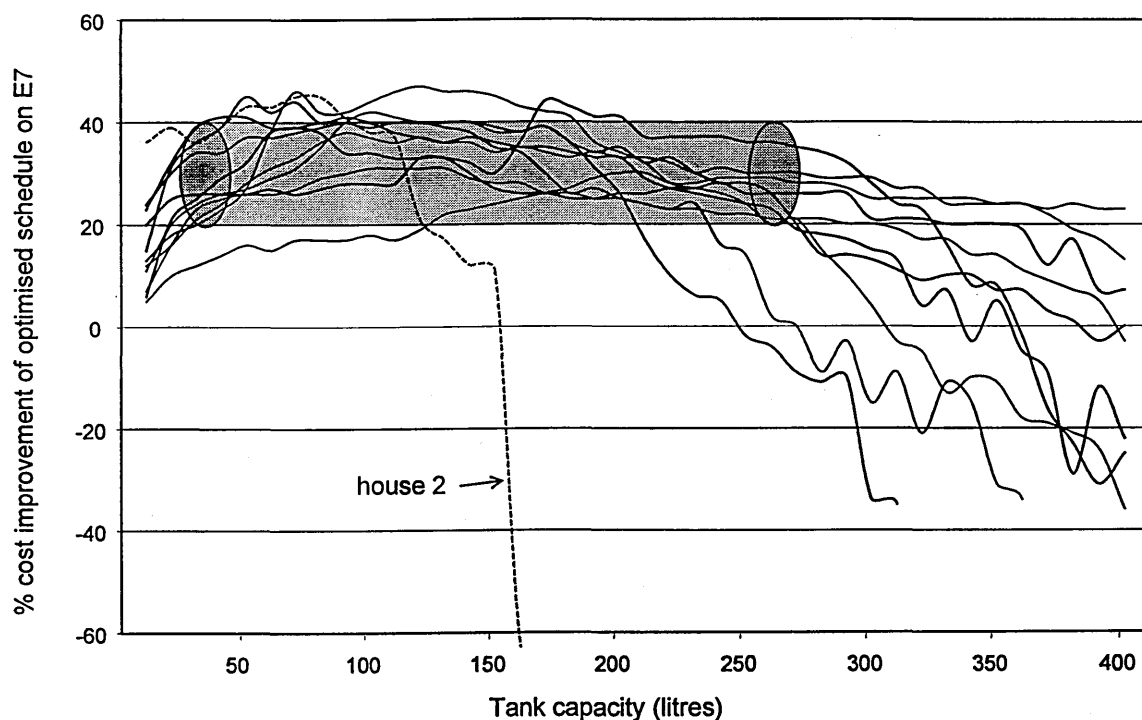


Fig 5-16 For tank capacities between 50-250 litres the optimised schedules show consistent savings between 20-40% compared with E7.

### 5.8.4 Why is the Optimised Schedule Sometimes Worse?

For tank capacities over 300 litres the relative performance of the optimised schedules degrade compared with E7. For house 2, which has very low consumption, this decline

starts at 150 litres (Fig 5-16). E7 is outperforming the optimised schedule, so why did the optimiser not arrive at a schedule similar to E7?

The reason for this is the balance between tank size, consumption and the optimisation process. In the system used the optimisation window was 24 hours and the schedule was calculated once per day. With larger tanks better performance will be achieved by increasing the optimisation window. A tank of 600 litres can typically hold enough hot water for three days consumption. An optimal 24 hour schedule will probably not involve charging the tank as it can be wasteful because of excess heating that is not required within that 24 hour period. Similarly for low demand a shorter window or continuous optimisation would result in improved performance.

Fig 5-17 shows the relationship between the mean daily water consumption and the tank size resulting in the cheapest cost for the E7 and optimised schedules. The optimised schedules are a significant improvement and suggest a range within currently available tank sizes.

It is interesting to note that the optimum E7 tank size has 5 times the capacity of the average daily requirement. By keeping a large volume of hot water the daily tempera-

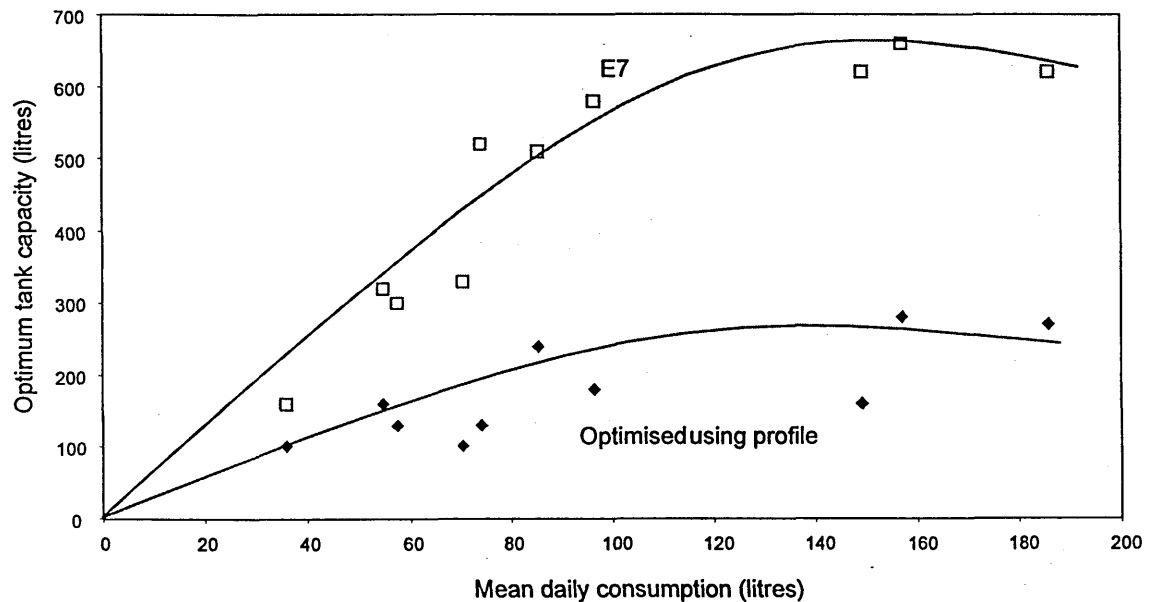


Fig 5-17 Optimum tank sizes



ture reduction is small, so less input will be required by element 2. Introducing time dependent heat losses into the model would give a more realistic situation. In reality the system behaviour is not like that of the model, as water is not always delivered at the required temperature.

### 5.8.5 How is the Optimisation Working

Fig 5-18 compares an optimised solution with the E7 situation, showing how the tank temperatures and cumulative daily costs vary for a case with a 200-litre tank.

It can be seen that for the E7 schedule the tank is charged to full capacity starting at midnight. The tank does not require a full 7 hours charge, typically only 2 or 3 hours are required before it reaches the set point temperature. Examination reveals that the difference in cost occurs in the way the tank is charged over this night time period. The optimised solution delays charging in order to miss the peak prices that occur after midnight. Ironically these prices are a result of the night time tariffs being introduced, as there is a surge in demand at midnight when appliances are switched on. This is seen consistently for all three days. Days 1 and 2 show that the relatively low demand enables a reduced temperature in the water tank. Rather than heat a full tank of water it is

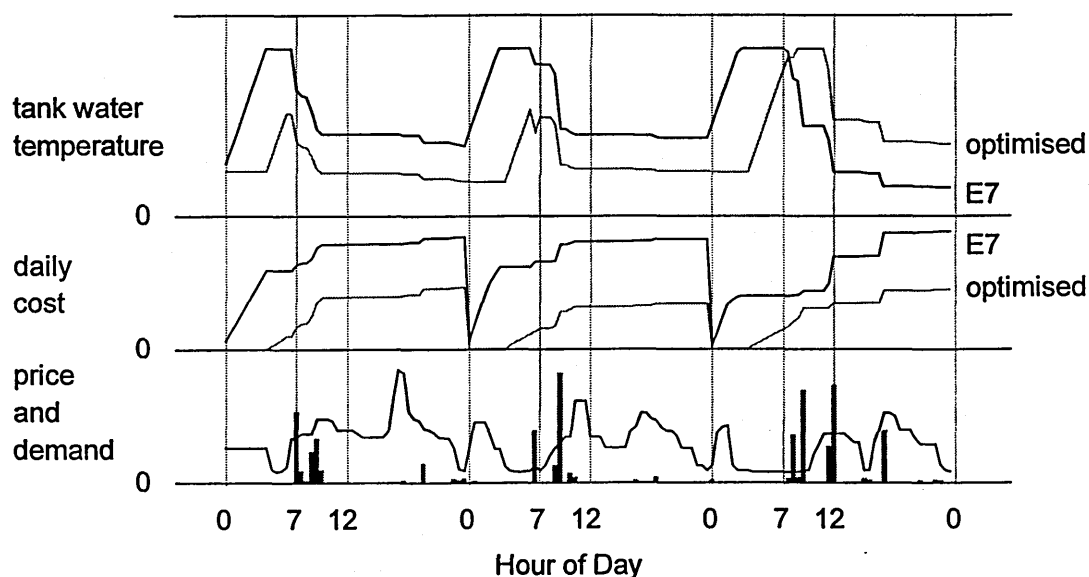


Fig 5-18 Comparison of an E7 and optimised solution over 3 days

more economical to part heat it and then just top up the demand to the required temperature with direct heating. On day 3 the optimised schedule is roughly half the cost of the E7 schedule but has warmer water in the tank at the end of the day even though it started colder.

### 5.8.6 Local Minima

An example of how a search can become trapped in a sub-optimal solution is demonstrated by Fig 5-19. Two charging schedules are shown (schedule 1 and 2) with their resultant cost and tank temperature profiles. The only difference in the two solutions is the hours at which the tank is charged, schedule 1 being charged in time slots 8,9,10 and 11 while schedule 2 is charged in slots 1,7,8 and 9. The tap source was identical for both solutions. When this particular day was optimised in isolation these two solutions were constantly found, but schedule 2 would never be reduced to schedule 1. In order to achieve schedule 1 the search would have to be restarted.

What can be seen is that for any improvement more than one bit flip in the charging schedule is required. Four charging periods appear to be required but one bit flip will result in 3 or 5 charging periods. In order to keep four periods but redistribute the times, two bit flips are required, one to destroy and one to create. To jump directly from schedule 2 to 1 requires 4 bit flips to simultaneously flip periods 1,7,10 and 11, which is why it never occurred as only 3 flips were allowed. With four bit flips allowed the chance of jumping from schedule 2 to 1 is less than 1 in over 80 million, assuming the string is not reduced in length from 96 bits.

No intermediate solution is possible because of the demand in time slot 4. This illustrates the benefit in frequently restarting the stochastic search as opposed to continuing the search from a local minimum and also having the possibility of more than one bit flip between solutions.

What is interesting in these optimised schedules is that although both have a full tank of hot water available for the demand in time slot 15, cold water from tap 2 is selected and

the hot water saved for the higher level of demand in slot 16. Once the tank is almost emptied (due to the demand at time slot 16) it is not recharged because there are no further cheap periods of which to take advantage.

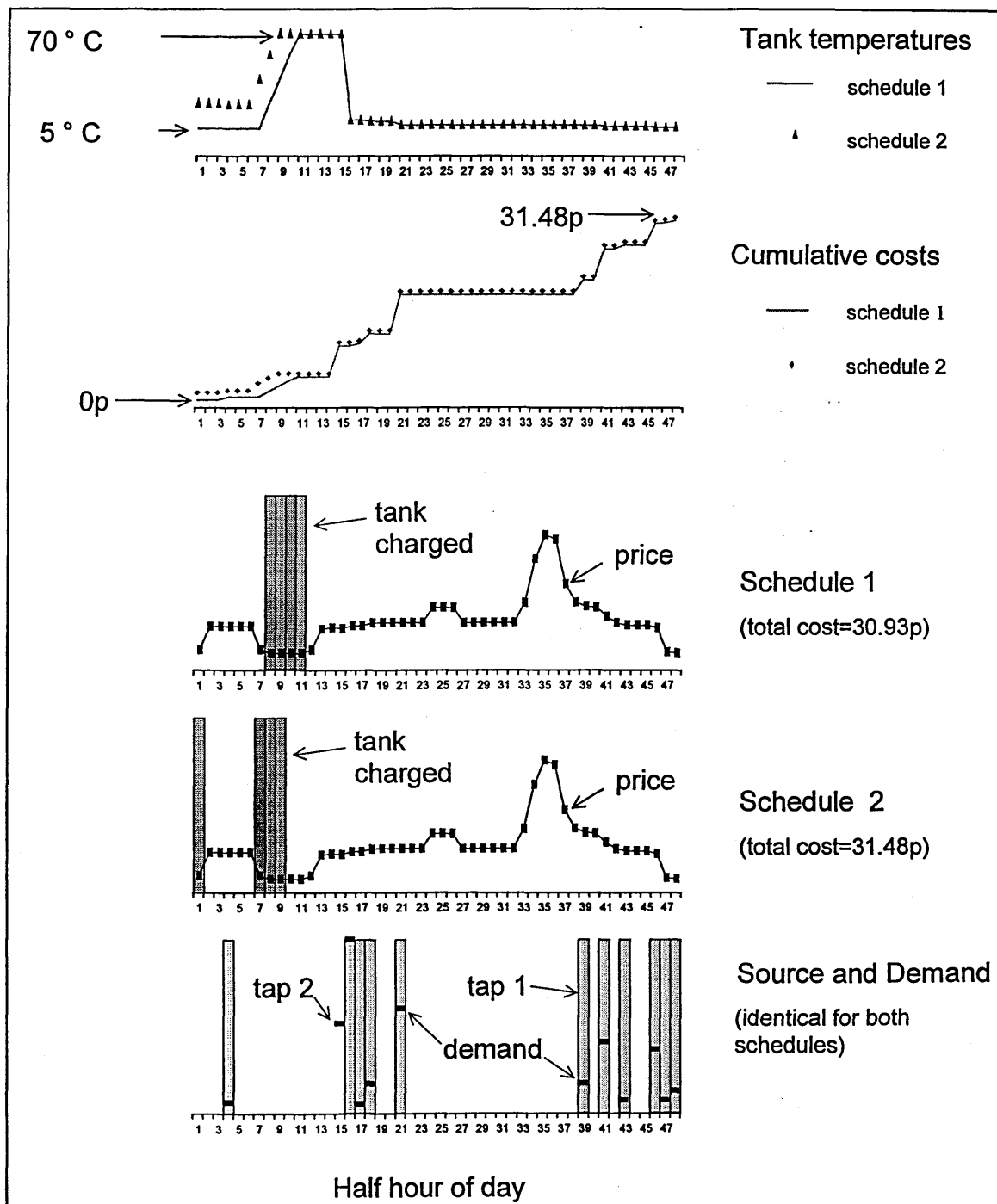


Fig 5-19 Global and local minimum solutions. The x-axes are relative linear values for visual comparison only.

## 5.9 Chapter Summary

The simulations have demonstrated the potential for large improvement in water heating strategies based on real consumption data and prices. Saving of the order of 40% were not untypical, which would reduce the cost of electricity supply to domestic customers by 16%, based on water heating being 40% of the total consumption [81,90]. There are no technological barriers to implementing such a control scheme.

# 6

## **Storage Radiator Controller Simulation**

### **6.1 What are Storage Radiators ?**

A storage radiator (also commonly known as a storage heater) is essentially a brick that is electrically heated during the night and dissipates the stored heat gradually throughout the day. The idea is that the thermal storage capacity of the bricks is utilised to shift electrical heating load in order to help increase the load factor.

Fig 6-1 shows a cross section through a typical radiator. An electrical heating element is encased within the bricks and is used to heat up this 'core'. Surrounding the core is thermal insulation that helps retain the heat. The core contains channels through which air circulates by natural convection, heating up the room.

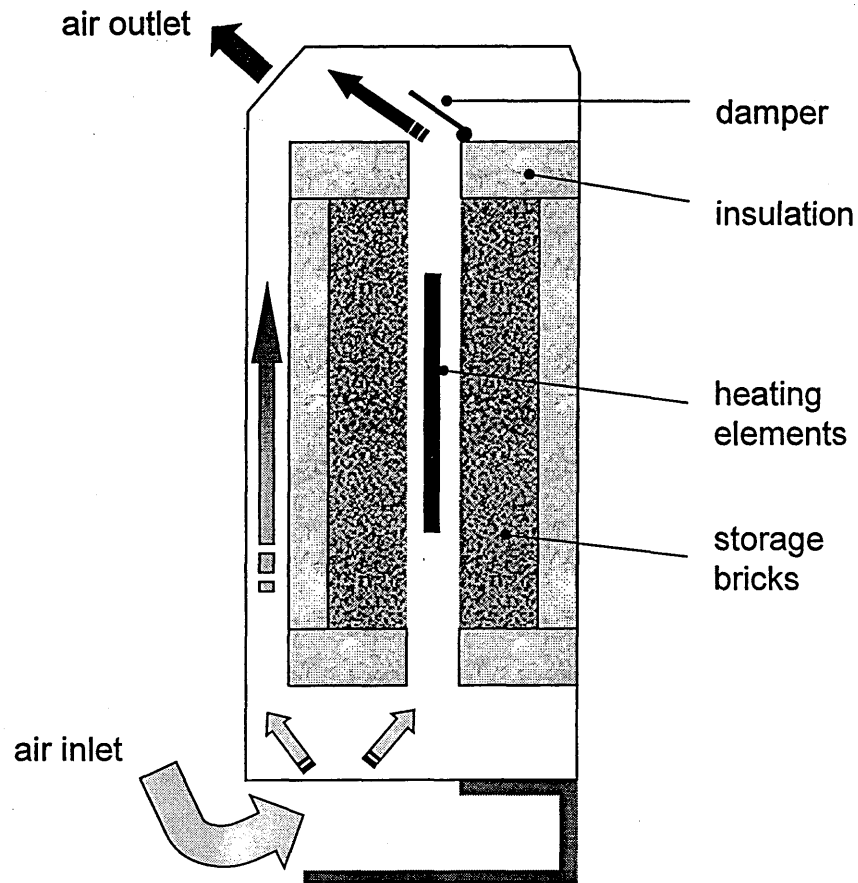


Fig 6-1 A cross section through a basic storage radiator

There are two manually operated controllers on the basic radiator. The first controls the energy input and adjusts the thermostat regulating the core maximum temperature. The second controls the heat output and adjusts the damper position, regulating the amount of air that is allowed to circulate through the core. The radiators (and water heaters) are hard-wired to a separate electrical circuit, which is activated from the electricity meter by a time clock or radio tele-switch.

In order to improve their controllability, fan storage radiators were developed. These have a high level of insulation to minimise heat loss, and an electric fan that can be activated to force air through the core when heat is required. The air gap in the core is designed so that this forced convection is required to extract the heat. This is done by having an inverted 'u' shaped air passage.

## 6.2 Room Thermal Model

In order to simulate the heating controller, a thermal model of a room is required which the neural network has to attempt to emulate. The neural model is then used to evaluate heating strategies, of which an optimum is sought. The thermal model is derived from first principles and attempts to emulate the response of a real system.

An explicit finite difference method was used to create the thermal model. This is a nodal approach that calculates temperatures at nodes within building elements (walls, core etc.) at discrete time intervals, which was every 5 seconds in this particular simulation. Heat transfer within the elements is by conduction, with convection taking place at the element surfaces. Each wall can be given different thermal properties and exterior air temperatures. Wind speed and outside air temperature were the only weather variables required, as solar gains were ignored. Every 5 seconds the net energy input into the room is calculated and the room air temperature updated. Appendix F gives more details of how the storage radiator was simulated and code for a room with a storage fan heater.

The simulated room had a 2kW storage radiator and a 1kW direct acting heater. The direct acting heater was only allowed to operate at times when there was a required internal temperature. The storage heater had a thermostat that stopped charging if the core temperature was above 700 °C. The room dimensions were 4 × 5 × 2.5 metres high, with 50% of the wall area exterior, 10% of this glazed. There were 0.1 air changes per hour with the outside air. Data on the room conditions was recorded at the end of every half-hour.

The exact details of the room configuration are not important and do not have to reflect any 'typical' type of room that the heater will be placed in. What is being investigated is if the neural network can learn the behaviour of the given room, whatever its properties.

### 6.3 Neural Network Emulator

The purpose of the neural network is to generate an empirical model that will emulate the behaviour of the theoretical model. The objective is to be able to predict what the room temperature will be in half-an-hour, given the current conditions and the heat inputs in that half-hour. The heat inputs can then be optimised so that the required temperature is satisfied in the cheapest manner.

The neural model was created by reducing the heat transfer process into three distinct parts. Fig 6-2 is a schematic of the process and Fig 6-3 the neural emulator created.

Network 1 predicts the next inner core temperature ( $TC_{in+1}$ ) given the current core condition ( $TC_{in}$ ,  $TC_{out}$ ) and the storage charge occurring in that time slot ( $SH_{charge+1}$ ).

Network 2 predicts the next outer core temperature given current core conditions, the room temperature and the previously predicted next inner core temperature.

Network 3 predicts the next room temperature given the current and predicted core states, the direct acting input ( $DA_{charge+1}$ ) and historical weather temperatures for the previous 12 hours.

Networks 1 and 2 had two hidden neurons and network 3 had three. Hyperbolic tangent ( $\tanh$ ) and linear neurons were used in the hidden and output layers respectively, and all data scaled to lie in the range  $[-1,1]$ . Any network output outside this range was rounded to  $-1$  or  $1$ , effectively using an activation function known as softmax.

The time step of the neural model is half-an-hour. Starting at midnight, two charge values for the storage and the direct acting heaters are fed into the emulator and the thermal changes predicted. The time step is advanced and the predicted state is fed-back to become the current state. This process is repeated for all 48 time steps so the emulator can predict the room temperature response to the given 24 hour charging schedule, but based on a single model predicting half-an-hour ahead.



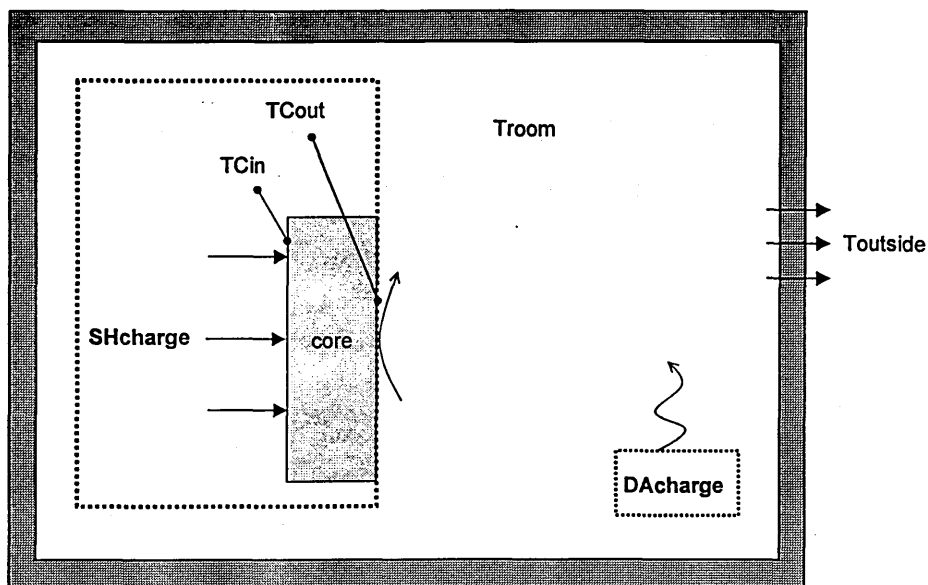


Fig 6-2 Schematic of the heat transfer boundaries in the model

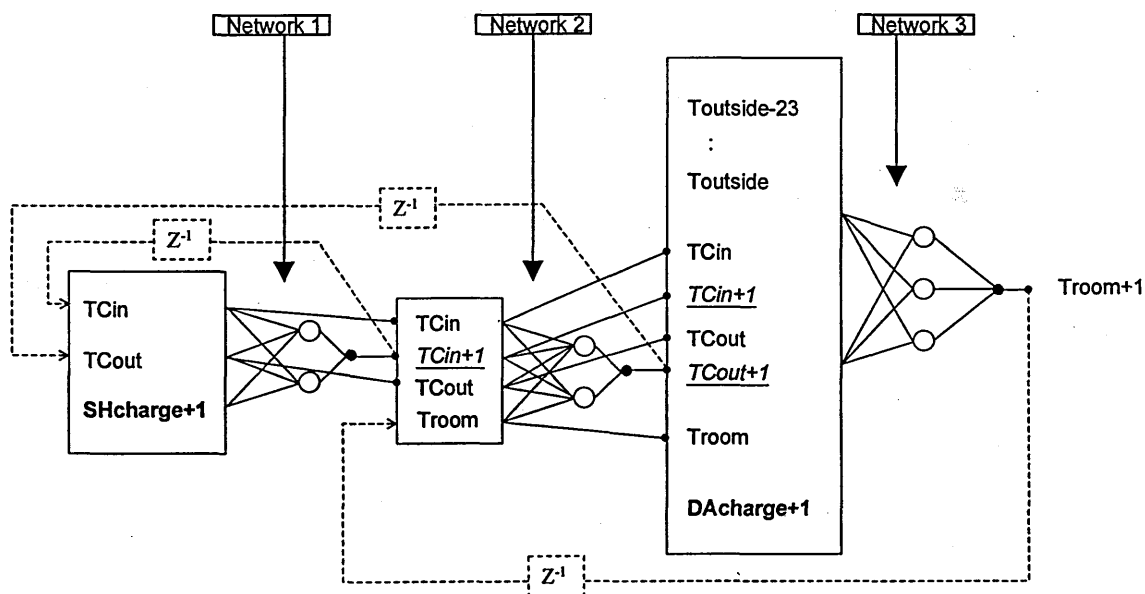


Fig 6-3 The neural emulation of the theoretical model for predicting the room temperature in half-an-hour, given the storage and direct acting energy inputs

TCout = core outer surface temperature  
 Troom = room temperature  
 SHcharge = energy to storage heater

TCin = core inner surface temperature  
 Toutside = outside temperature  
 DCharge = energy to room by direct acting heater

In the neural emulator, outside weather temperatures are required for the 12 hours previous to the current time slot under consideration. In the simulations retrospective actual temperatures are used, although in reality these would have to also be predicted.

## 6.4 Optimisation Procedure

The neural emulator was used to evaluate proposed daily schedules of half-hourly charges for the storage and direct acting heaters. A schedule consists of a string of 48 or 96 numbers, depending on the optimisation used. The evaluation was equated in terms of cost, calculated by using the pool price.

Occupancy profiles were created giving the hours at which set point temperatures are required. Both the occupancy times and set point temperature values were varied so as to create a diverse range of conditions for the neural network to learn. If no set point was given (room unoccupied) then the temperature could behave in any manner, but a condition was given that the direct acting heater could not be on in these periods.

To compare the neural controller performance with existing heating strategies on a like-for-like basis, it was a requirement that the schedules must attempt to satisfy the room temperature set points given by the daily profile. This was the reason for including the direct acting heater, which operates only when there is occupancy in order to make up any shortfall in temperature.

In each half-hour there is a storage heater electrical charge and a direct acting electrical charge. In the theoretical model these are either 2 kW (storage) or 1 kW (direct). Because the thermostats can operate every 5 seconds, the recorded value at the end of each half-hour was equivalent to a continuous charging at a fixed percentage of full power. The load inputs to the neural emulator are thus real numbers, equivalent to the percentage of full power that should be utilised.

---

In order to implement this in the optimisation procedure, the string representation has to be changed. The control actions, previously represented by a bit at a particular location on a string, are no longer binary choices. Each action can now be any real number between 0 and 100, so the string could be real valued with a mutation being the random generation of a new real number as opposed to a bit flip. From the experience gained in chapter 5, a random number of mutations (between 3 and 10) were allowed between each evaluation.

The primary optimisation task is to track the given temperature profile. The ability to give the heaters variable charge levels (although constant in each half-hour period) means that there is an infinite number of solutions to this problem. Because heat can be stored and the price varies every half-hour, each solution will have a different associated cost, the one giving the lowest being sought. This is a difficult optimisation problem and attempting to find the global solution would be a futile task. What is required is a method that can give a reasonable solution.

To reduce the search space the storage heater loads were limited to 5 discrete values equivalent to 0, 0.5, 1, 1.5 and 2 kW. This is more realistic to how a real controller would operate by activating a set of four 0.5kW elements. It also has the advantage over using continuous real numbers in that a random real mutation is unlikely to set the charge to zero, or 'off', whereas there is a 1 in 5 chance with the discrete coding.

As there is no control of the storage radiator output, the optimisation is simpler than that of the hot water system in chapter 5 as no decision has to be made as to the source of the heat. If a storage fan heater were simulated then the decision to switch the fan on would be equivalent to taking hot water from the tank.

Because the set point temperature has to be satisfied then only the storage heater charge needs to be optimised. If the storage schedule does not meet the required set point in any particular time slot, the direct acting heater is activated by incrementing the emulator load from 0 to 1kW until it is predicted that the set point will be achieved. A storage

schedule will thus automatically have an associated direct acting cost, giving the total cost for that solution.

By optimising for minimum cost, a likely solution is never to charge the core. To prevent this scenario, a high penalty cost was added to the total cost if there was under heating at times when set points were required. Because of this it was necessary for the initial starting schedule to include a high degree of charging. If the initial solution was not to charge then there would be a high penalty cost due to the under heating. If random mutations did not instantly eliminate under heating in at least one time slot, then the same penalty cost would still be incurred as well as an added cost due to the charging, thus increasing the overall cost. The search procedure would thus never advance.

## 6.5 Simulation Procedure

Two simulations were performed in the optimisation. One was for a situation where the thermostat switches on the radiator core and direct acting heater were operational when the optimised storage schedule was passed through the building model (NNopt). The model then automatically activates the direct acting heater when required, giving a similar control scheme to existing strategies.

The second simulation (NN) was performed where both direct acting and storage charges were simultaneously optimised (using a real valued string of length 96) to maximise the comfort satisfaction. The cost function being minimised was thus the total absolute error, where the error is the difference between the required set point and the model prediction. The optimised solution was fine tuned by adjusting the direct acting charge so that the set points were satisfied, and the schedule then passed through the building model with no thermostat switches. This was performed to provide a yardstick from which the effectiveness of the cost optimisation could be assessed and to test the accuracy of the emulator predictions.

E7, E10 and direct acting only performances were also evaluated. In these cases the control rule was given that the direct acting heater would switch on if at any time during the occupancy periods the room temperature was below the set point, and switch off again once the set point was exceeded. During non-occupancy periods the direct acting heater was switched off. For the simulation with only a direct acting heater, the element size was increased to 2 kW to prevent under heating.

The controller was simulated for 72 days starting from 1st January using weather data from Kew. The neural networks need some initial data on which to train, so 3 days on an E7 charging profile were simulated and used to create this initial database.

Fig 6-4 is a representation of how the simulation proceeded. From the database of actual past behaviour, training patterns were created and the three networks trained. The networks were then used as an emulator to evaluate candidate charging schedules, which had been suggested by the optimisation process. Once a solution had been obtained, the charge levels were then applied to the theoretical model and this 'actual' behaviour recorded and added to the database. The process was then repeated for the next day.

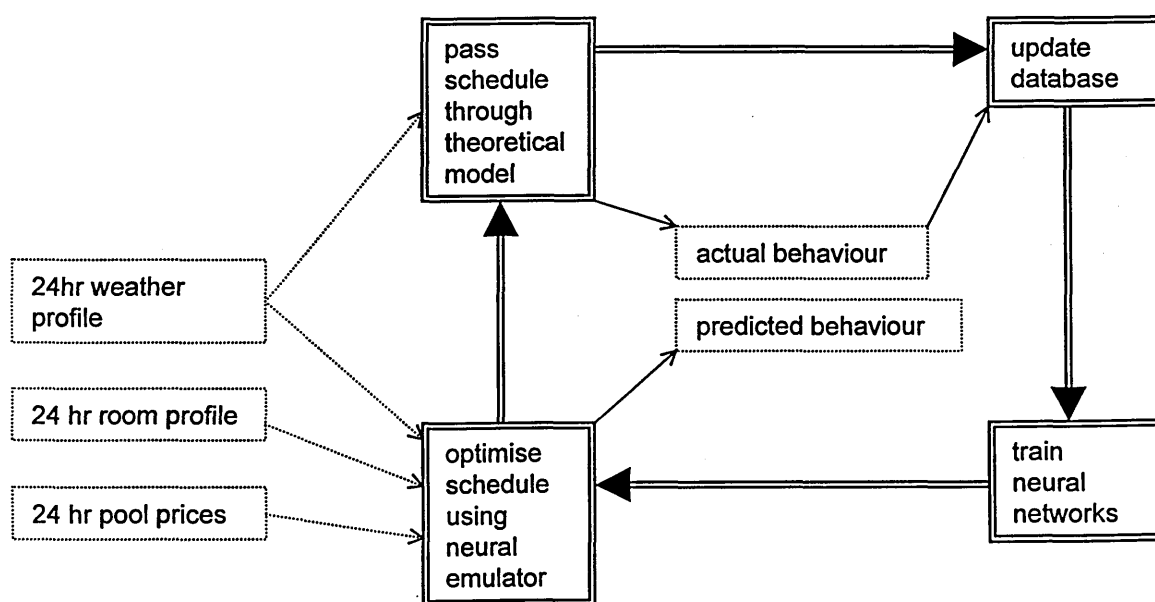


Fig 6-4 A schematic of how the simulations were performed

In re-training the networks, the previous weights were used and given 25 epochs training on the patterns in the updated database. The optimisation was allowed to proceed for 800 function evaluations, which took considerably longer than the network training, although still only in the order of 10's of seconds.

## 6.6 Results

### 6.6.1 Did the Controller Work ?

Without analysing the results too deeply, did the controller generally achieve a satisfactory performance in terms of temperature control – or basically, ‘did it work? The results of simulation NN, where there is no thermostatic control, is shown in Fig 6-5. The differences between the predicted and achieved temperatures are shown chronologically over the 72 days.

It can be seen that eventually, after some initially large errors, the achieved temperatures consistently fall well within 1 °C of the predictions. For building thermal control this is within acceptable limits and it can thus be stated that in simulation the model predictive controller does work.

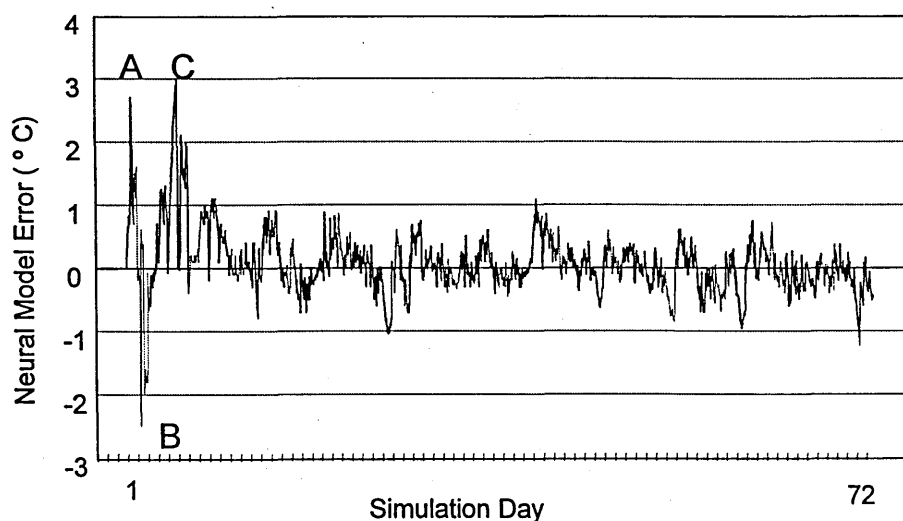


Fig 6-5 Half-hourly errors over the 72 day simulation. The error is the difference between the emulated and achieved temperatures.

## 6.6.2 Why do the Large Initial Errors Occur ?

From Fig 6-5 it is observed that early in the simulation there are days when the errors are relatively large, indicated by A, B and C. A close examination of the data, shown in Fig 6-6, reveals why these errors occur.

Large errors occur on day A, the first day that the controller is used. The three previous day's data were used for training, which originated from an E7 charging schedule. During these three days, the core temperature never falls below a certain value and the data would be scaled between these current limits of experience. The emulator core temperature will never fall below its previous minimum whatever the charge, due to the fact that all predictions fed back have to lie within the scaled range  $[-1,1]$ . There can thus be a solution with no storage charge but a heat input equivalent to the core being at the previous minimum. This is what has happened on day A, with the actual response of the core temperature to no charging falling below what the network predicted. As a result the room is under heated.

The same effect is repeated for day C. There is no occupancy during this day so there will be no storage charge. The core is already at the minimum temperature previously experienced at the start of this day, which the emulator guarantees will not be surpassed. Also at the start of this day the room temperature is close to its lowest ever value and the emulator again has restricted the predicted temperatures to lie within current experience. This is why the predicted temperature for day C is almost constant at its previous minimum. Once the new data has been added to the database the limits change, as can be seen for the following day.

For day B the direct acting charge is at a higher level than had previously occurred. Any charge level above that previously experienced will not have the corresponding increase in effect. This is due to the saturation of the tanh activation function that limits extrapolation. When the optimised charge is then applied, overheating occurs. This shows an inefficiency of the optimisation process, as the direct acting input could have been

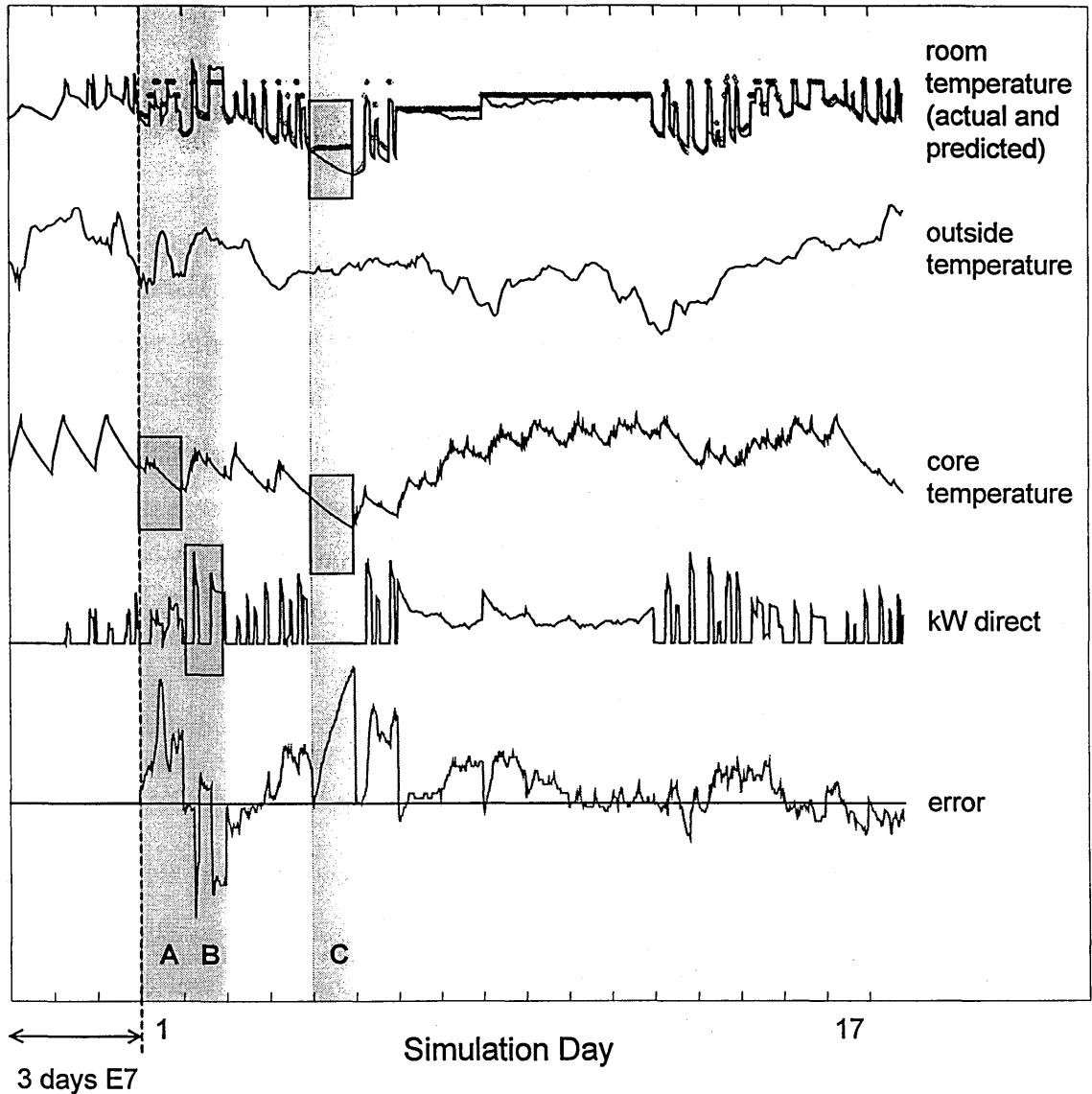


Fig 6-6 Investigating why the early errors occur

reduced giving the same comfort for less cost. The fine tuning did not correct this as it only adjusted the direct acting heat input upwards.

The initially large errors occur due to the constraints placed on the emulator to prevent it from extrapolating (by using the softmax output function). If these constraints had not been set (by having a linear output function) then similar results would have been expected if the model had used the full range of the tanh activation function. This would ensure that any fed-back input outside current experience would be in the saturated regions and thus automatically set at either  $-1$  or  $1$ .



The neural network learns the room characteristics within its current limits of experience almost immediately. Errors occur when these limits are exceeded but once the full range of data has been experienced the performance is satisfactory.

### 6.6.3 Performance of the 1-step-ahead Predictor as a Recursive 48-step-ahead Predictor

The neural emulator was created in such a way so that there were predictions used within the current time step to estimate the room temperature – or a prediction based on predictions. These predictions were then fed-back to be used as inputs for a further 47 time steps through the day. This deliberately created ‘worst case scenario’ has the potential for multiple error accumulation and therefore should rigorously test the accuracy of the emulator models.

Fig 6-7 shows how the prediction errors accumulate throughout the day. The initial one step ahead error for hour 0.5, where the exact previous conditions are known, is around 0.07 °C. This immediately doubles for the next time step but gradually levels off at 0.4 °C around time step 20 (hour 10). The sharp jump at hour 7 is due to the occupancy

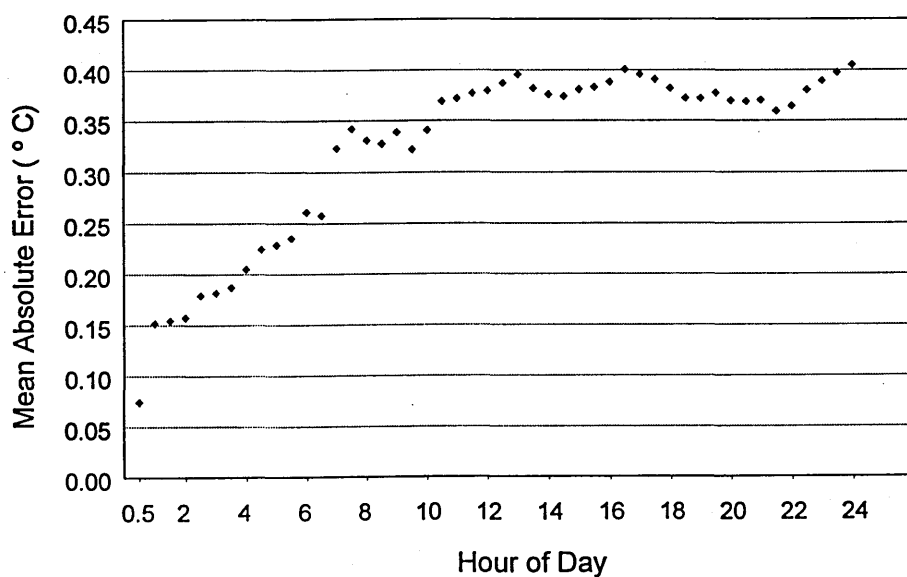


Fig 6-7 The absolute error of the emulator for each time step averaged over the 72 days

patterns consistently requiring a set point temperature for the start of the day. This will result in direct acting heaters being switched on, giving a large change in room temperature and the potential for larger errors.

Averaged over time it can be seen that there is a gradual deterioration in performance up to hour 10 and then a relatively constant error for the remainder of the day. It could easily be assumed from these results that the errors of each individual day will follow a similar pattern, with errors gradually accumulating as the fed back predictions become gradually worse.

A closer inspection of the actual daily errors shown in Fig 6-6 and in more detail for a different period in Fig 6-8 reveals this is not the case. For day 50 (Fig 6-8) when there is no occupancy and thus no heating, there is a general drift in the error up to a point, but even here it starts to improve in the later stages of the day.

#### **6.6.4 Comparison with other Heating Strategies**

To assess the relative performance of the model predictive controller the simulations were repeated for E7, E10 and a direct acting (DA) only charging schedule. The results are shown in Table 6-1.

The comfort optimised (NN) neuro-controller is almost twice as expensive as all the other heating strategies, although its actual energy consumption is comparable to E7 and E10. This is unsurprising, as its only objective is to satisfy the demand without any cost considerations. This demonstrates how electric heating can be very expensive if not efficiently regulated.

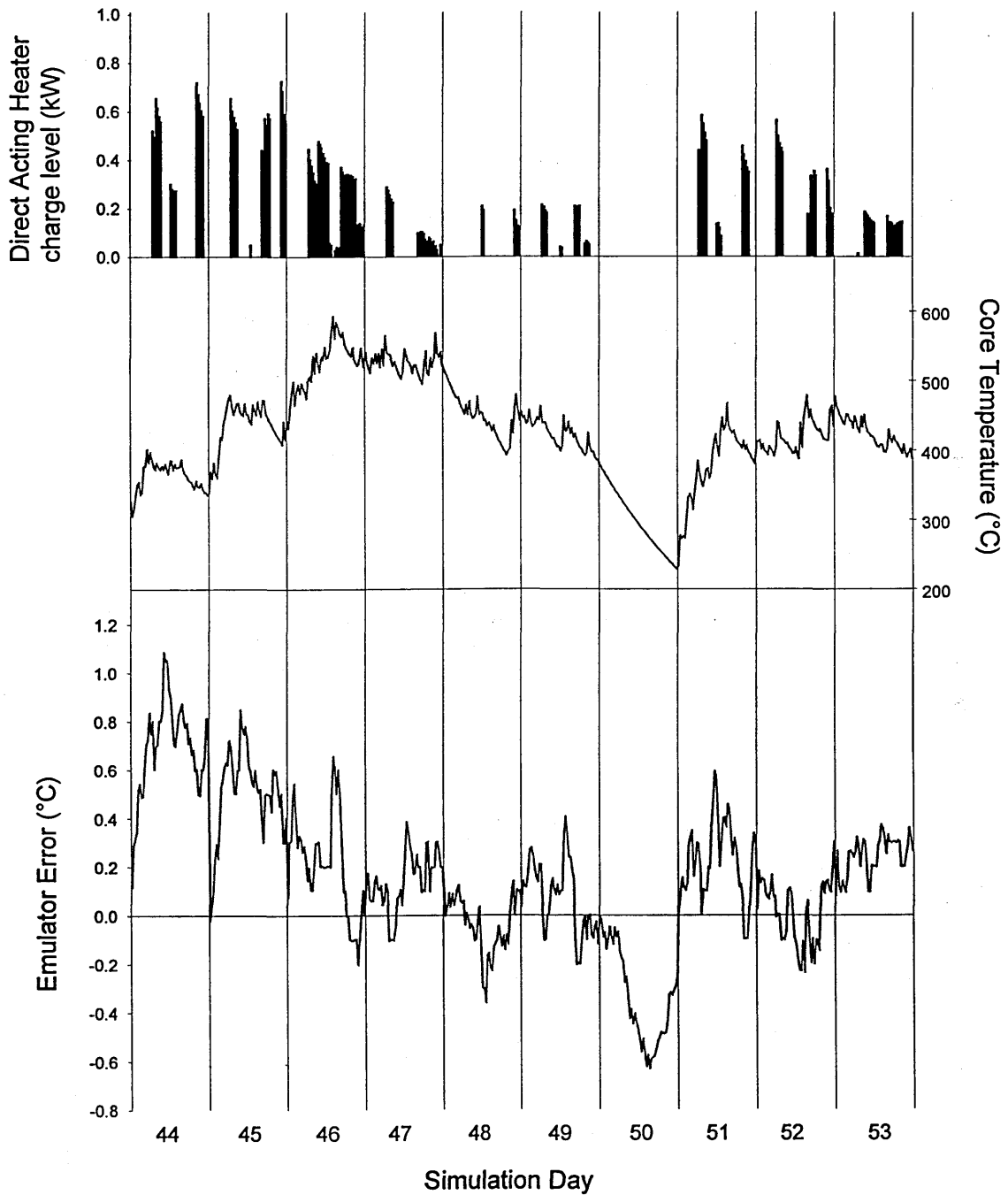


Fig 6-8 The actual daily errors do not follow the time averaged pattern of Fig 6-7

The cost optimised solution (NNopt) is 5.5% cheaper than E7 and uses 27% less electricity. This imbalance in the cost savings is in part due to the fact that there were some high pool prices. The peak pool price was 70p/kWh whereas the mean price of the cheapest 90% of half-hours was only 1.5p/kWh. For NNopt, 34% of the direct acting cost was accumulated in only 4% of the time that the direct acting heater was operational.

The improvement brought by the controller is in its ability to accurately regulate the temperature. In E7 there is excessive overheating of effectively 2.29 °C for a continuous period of 10 days, with E10 being worse. The neuro-controller has weather information for the day ahead so it can set the loads at a level so that overheating will not occur. Overheating does occur for 21 slots out of a possible 1,383, due to set points of 16 °C being given, the cooling rate being too low for the temperature to fall enough in the time specified.

Table 6-1 The relative performances of 5 heating strategies

	NN	NNopt	E7	E10	DA only
<b><u>COST (£)</u></b>					
Storage	30.99	6.70	12.17	17.16	---
Direct	4.55	9.37	4.83	3.01	21.86
<b>TOTAL</b>	<b>35.54</b>	<b>16.07</b>	<b>17.00</b>	<b>20.17</b>	<b>21.86</b>
<b><u>CONSUMPTION (kWh)</u></b>					
Storage	986	429	959	1097	---
Direct	104	341	104	62	622
<b>TOTAL</b>	<b>1090</b>	<b>770</b>	<b>1063</b>	<b>1159</b>	<b>622</b>
<b><u>COMFORT</u></b>					
<i>Overheat (&gt;0.5 °C)</i>					
Occurrences	404	21	476	696	---
Average (°C)	1.44	2.44	2.29	2.62	---
<i>Underheat (&gt;0.5 °C)</i>					
Occurrences	47	---	---	---	---
Average (°C)	-0.73	---	---	---	---

---

The most energy efficient solution is direct acting only, as energy is not wasted heating periods that do not require a set point. The neuro-optimised solution (NNopt) is 23% less efficient than direct acting but 26% cheaper.

## 6.7 Emulator Improvements

The neural emulator created was a 1-step-ahead predictor that used its own predictions to extrapolate to 48 time steps ahead. At each time step the only information it receives that is not predicted directly from the starting conditions is the outside temperature, for which actual values are used. By feeding back the predictions to advance a time step there is thus no new information being introduced apart from potential errors.

A better approach would be to have 48 networks, each trained to predict the temperature for a given time step ahead. The inputs would be the initial starting conditions and the weather and loads that had occurred up to that time step. Intermediate temperatures are thus not introduced as they do not need to be known for the current prediction.

By taking the average temperature from a population of models for each time step (see section 3.10 on page 61), the accuracy would be further improved.

The potential effect on the error of 'wrong' temperature predictions needs to be quantified. It is hypothesised that this will only be important when there are sudden unpredicted cold fronts and the model underestimates the heating requirement.

## 6.8 Chapter Summary

The simulations performed in this chapter have demonstrated that theoretically neural networks could work as model predictive controllers of domestic storage heating. The main benefit over existing systems is anticipating when overheating will occur and reducing the charge appropriately.



## **Real Neural Storage Radiator Control**

---

### **7.1 Background**

The simulation results from the previous chapter indicate that neural networks could be used to improve storage radiator control. Computer simulations can significantly reduce the development time of new products, but more practical problems may only come to light once working prototypes are developed. The development and results from a prototype of what is believed to be the world's first neural network controlled domestic storage radiator are reported in this chapter.

The room being controlled was on the first floor of an occupied two-bedroom property located close to Bedford town centre. A plan of the first floor and photograph of the exterior are shown in Fig 7-1 and Fig 7-2.



The house is a converted nursery that was constructed around the turn of the century. The walls are solid 9-inch thick brick and there is a large glass skylight above the stairs. The room being monitored was the spare bedroom which was used as a study and had a 1.7kW storage radiator initially on E7, as did the bedroom. Below the study is a living/kitchen area with a storage-fan-convactor heater on E7. Below the bedroom is the bathroom. On the landing of the stairs was a clothes drier that was used from time to time with and without an extraction hose venting through the skylight.

Temperatures logged were ambient, room ( $\times 2$ ), storage heater core, storage heater air outlet and a desk lamp temperature. This lamp temperature was included as a potential means of identifying occupancy and thus heat gains from computer equipment, lighting and body heat. Logging commenced on 25th September 1997 and continued in 15-minute intervals until 14th May 1998. The neural controller was in operation from the 25th February with the period 17th – 30th March being the final ‘de-bugged’ version.

Fig 7-3 shows the ambient, core and room temperatures for November, when the heater was being charged on an E7 schedule. It can be seen how adjusting the input charge setting changes the maximum allowable core temperature. The reliance of the room temperature on ambient is evident, with there being little time delay between the underlying room temperature and ambient. This would suggest that the house is relatively lightweight. The responsiveness to the core temperature can also be seen, with the maximum room temperature occurring at 7:00 am when the core is hottest, confirming how the existing heater does not give the heat when it is required.

Fig 7-4 shows the whole heating season. The charge was initially set on E7 and then varied by disconnecting it from the off-peak circuit and controlling it by means of a timer plug. This change was introduced so that a diverse range of data could be collected. Several periods were included when the heater was switched off and the core allowed to drop to room temperature. This gave data at the lower extreme and an indication of the rate of heat loss of the core at different temperatures. A second reason for switching off the heater can be seen by the relatively high room temperature that was being maintained.



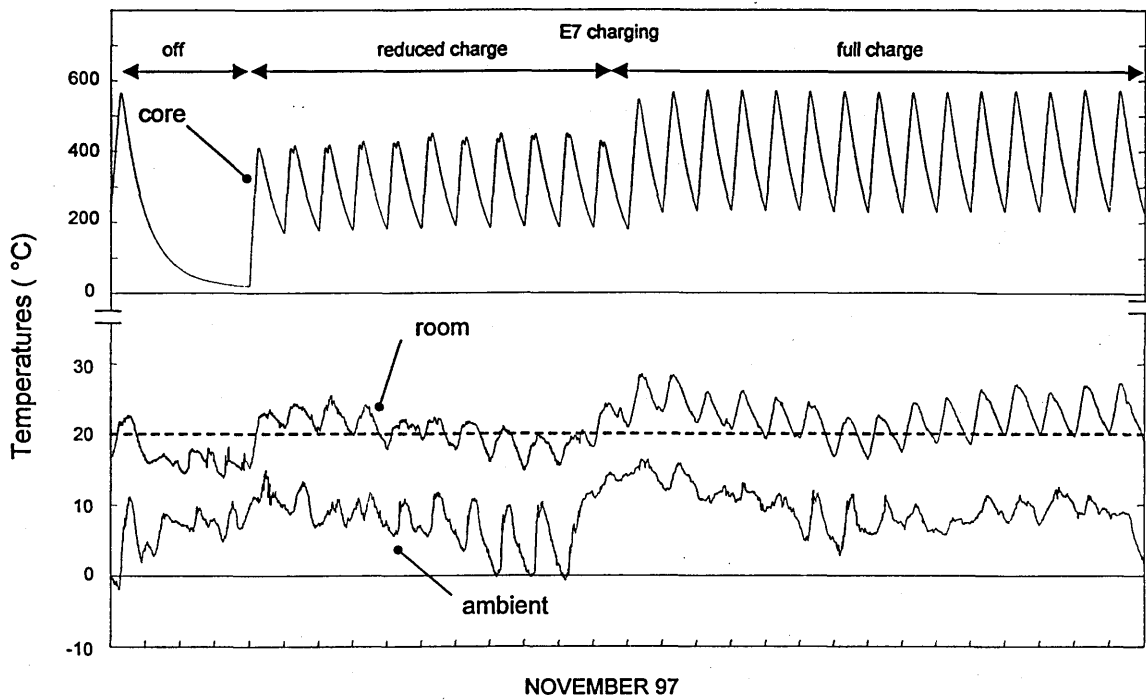


Fig 7-3 Room, core and ambient temperatures for November

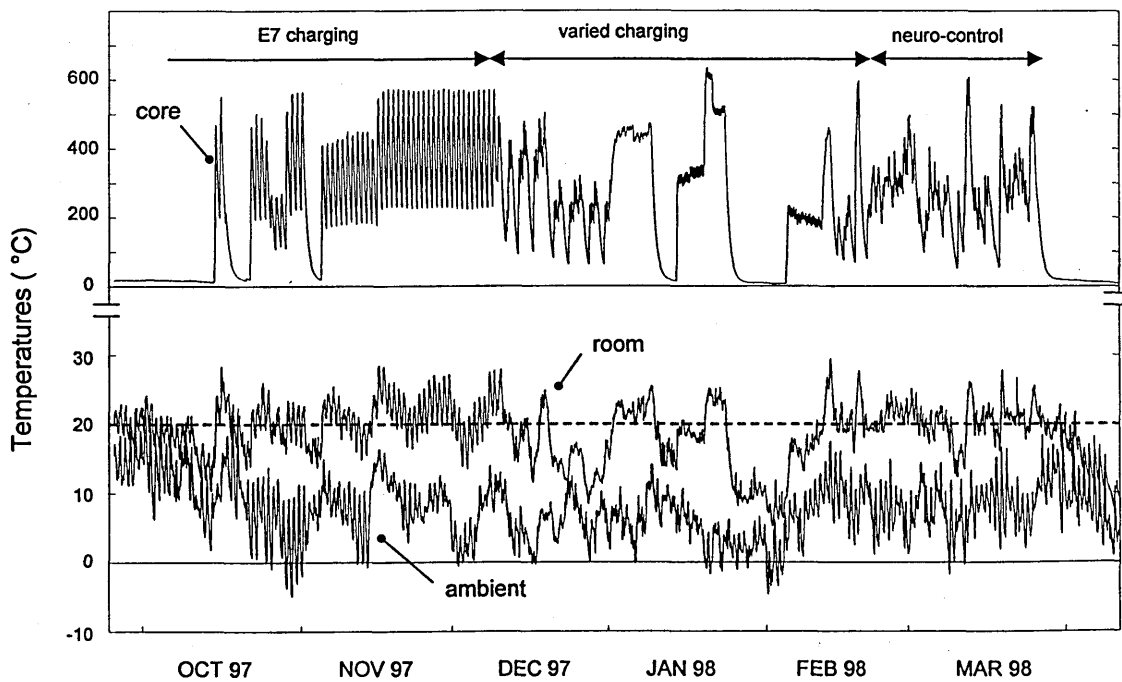


Fig 7-4 Room, core and ambient temperatures for the whole heating season

The heating season in which the work was undertaken was unusually warm. Because it was the last winter of the project, in order to gather data, the heating had to be on when normally it would not have been. The following are press releases from the Met. Office describing the weather at the time,

*ENGLAND 1997: TURNED OUT WARM AGAIN - This year has been the third warmest in England since records began over 300 years ago. August in 1997 was the second warmest on record, and February and March were both particularly warm months over England.*

*SCORCHING START TO 1998 - The first six months of 1998 have been easily the warmest first half of a year globally since reliable records began in 1860. Provisional observations analysed jointly by The Met. Office and the University of East Anglia show that the temperature averaged over January - June 1998 has been some  $0.6^{\circ}\text{C}$  greater than the average climate (calculated from the period 1961-1990). Each individual month in 1998 so far has been the warmest such month on record. Furthermore, the twelve month period July 1997 to June 1998 (with an anomaly of  $0.56^{\circ}\text{C}$ ) has been very much warmer than any other 12 month period not influenced by the current El Niño.*

*Not surprisingly, by tomorrow (Friday) morning April 1998 will be the wettest April this century.*

## 7.2 Data Analysis

Data from September to February was analysed in order to create a fixed model that would be placed in the controller.

The simulations in the previous chapter used electrical charge as the control variable. In the practical tests only temperatures were measured, so the control decision is to pre-determine a future core temperature that will track a future set point temperature. The control variable is thus the core temperature.

The control horizon in the simulations was up to 24 hours and depended on knowing future ambient temperatures, which were used retrospectively. In reality a weather forecast is required, the only option available for the prototype being an 'educated guess' (using a neural forecast), which limited the prediction horizon that could be used.

The importance of ambient temperature to the system determines how far ahead it would be reasonable to model, knowing that ambient predictions are likely to be poor. Several neural networks were trained to model a future room temperature given the current core, room and ambient temperatures. Both the intervening core and ambient temperatures were included and then only the intervening core temperatures. This was done in order to quantify the importance of the ambient temperature. In the neural networks it was found that one neuron was sufficient to model the system. A naïve prediction, which simply says that the future temperature will be the same as the current temperature, was also made. This is the worst case model and should always be used for evaluating neural models to ensure that they are doing more than simply repeating the last known value. The results are shown in Fig 7-5.

The intervening core and ambient temperatures determine the major heat transfer processes into the room and will thus influence the future room temperature. By including both of these in the model the rms error for a 15 hour horizon was 1.07 °C, compared

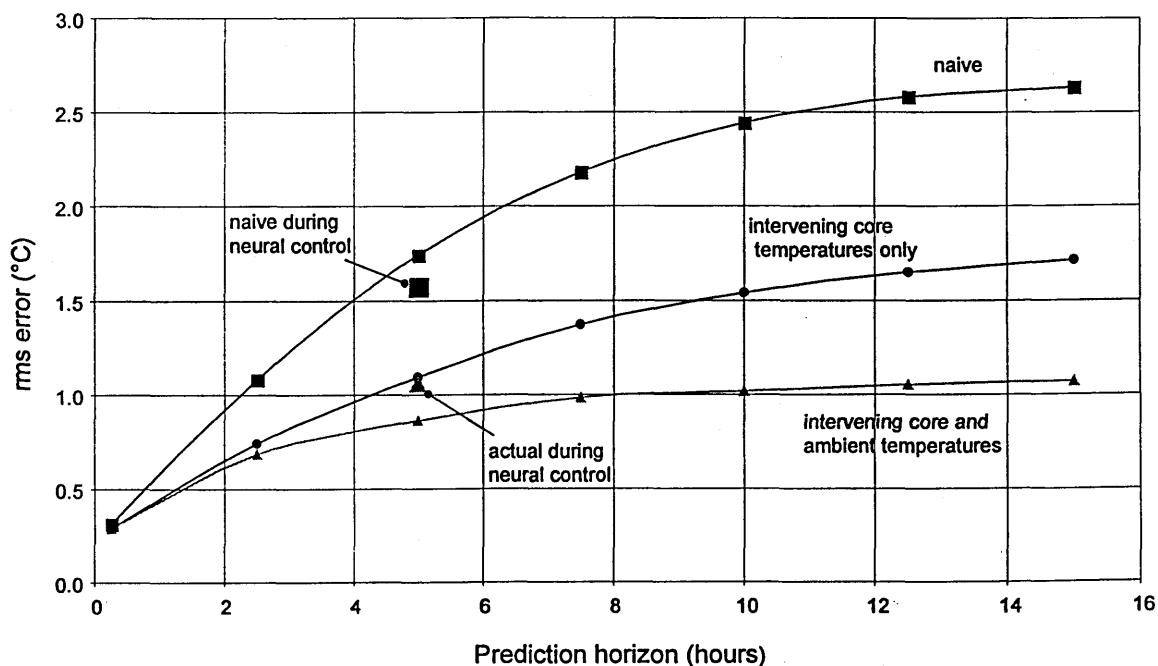


Fig 7-5 Neural model errors for predicting future room temperatures. Inputs are the current core, room and ambient temperatures and intervening ambient and/or core temperatures. A naïve prediction assumes that the future room temperature will be the same as the current. The data used is for the period September-February.

with  $1.71^{\circ}\text{C}$  by only including the core temperatures and  $2.63^{\circ}\text{C}$  for a naïve prediction. The longer the prediction horizon the more important it becomes to know the ambient temperatures, although just knowing the core temperatures is still a significant improvement over a naïve prediction. These results show that a network with a single hidden neuron gives a reasonable model and suggest that linear regression might work just as well. A prediction horizon of 5 hours was chosen, for which the difference in rms errors between knowing and not knowing the intervening ambient temperatures was  $0.23^{\circ}\text{C}$  for the data set analysed.

If a building is left for long enough then the room temperature will reach thermal equilibrium with its surroundings. It is thus not necessary to know the current room temperature if a long enough prediction horizon is being used. Two models were created, one included the previous 20 hours core and ambient temperatures but not the initial temperature 20 hours ago, while the other included the past 5 hours core and ambient temperatures but also the initial room temperature, as was used in the controller. The errors of these models are shown in Fig 7-6.

The rms error for model 1 is at a level that is only marginally better than a naïve prediction. This might have been discarded as poor but a close analysis of the errors gives some meaningful information.

The model indicates that something different is occurring at the end of October and December, shown by the constant bias from zero error. The last few days in October saw the first cold front, as can be seen from the ambient temperature of Fig 7-4. The downstairs heater had a controller that would keep the room at a set temperature by activating either the fan to release stored heat or the convector, which was available 24-hours. During this period this control was set so that there was a constant heat output throughout the day in order to keep off the chill. Normally this was only activated during occupancy. The effect of this on the room above is that there is an increased level of heat input through the floor, which changes the model and is why the errors are seen to be constantly underestimating the room temperature. This extra heat input to the room can be seen to equate to  $3\text{-}4^{\circ}\text{C}$ .

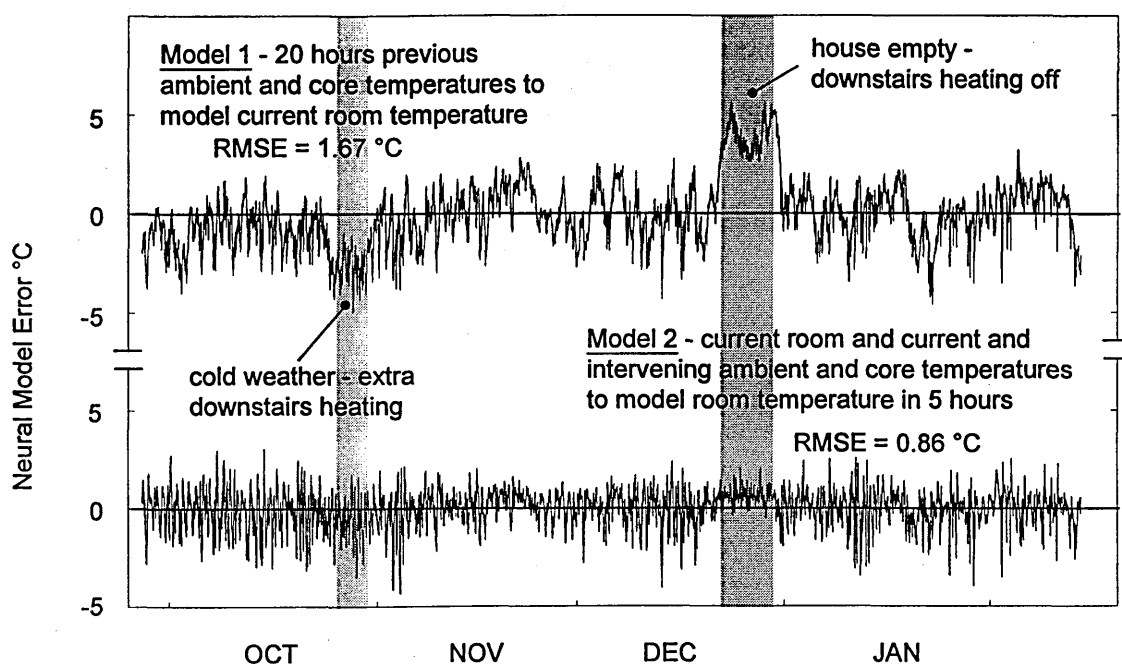


Fig 7-6 *Not including the current temperature in the model gives a worse error but anomalies are easier to spot*

At the end of December the house was unoccupied and the downstairs heating switched off. Model 1 clearly identifies the change by showing that the room is about 4 °C cooler than is expected, the missing heat input from the room below. Model 2 does pick up some change during this period with the errors being constantly positive, but what is more noticeable is the reduced noise level in the errors. This is due to the house being empty and identifies that there will be noise in the data caused by the occupant's behaviour.

The information extracted parallels with the causal model created for electrical load modelling in chapter 2. Including the existing temperature has the same effect as including previous loads, giving a better model by introducing a good initial guess. This can give a good model, but as has been demonstrated, causal models can reveal a lot of information even though the errors are worse. What they do is give clues as to how the model could be improved, which in this case would be to include as inputs the temperature of the room below and all other events that cause heat fluctuations, or 'noise'.

Examples of temperature fluctuations in the test room that are caused by events other than heater or ambient temperatures are shown in Fig 7-7. The 11 day period shown includes 3 days when the house was unoccupied, seen to be a time when the room temperature is well behaved. Occupancy can be seen to introduce a certain amount of noise, with the periods when the drier was on being clearly identified as spikes.

These 'spikes' are short-term energy inputs that do not affect the background temperature in the long run, and as such are considered to be noise. If the current temperature happened to be during one of these brief anomalies then errors would be introduced in the model as this would not represent the 'real' background temperature. In an attempt to filter out this noise and give a more stable longer term temperature, the three preceding as well as the current room temperature were included in the model. From this an effective time averaged current temperature will be calculated, with the resulting model showing a reduced level of noise in the errors.

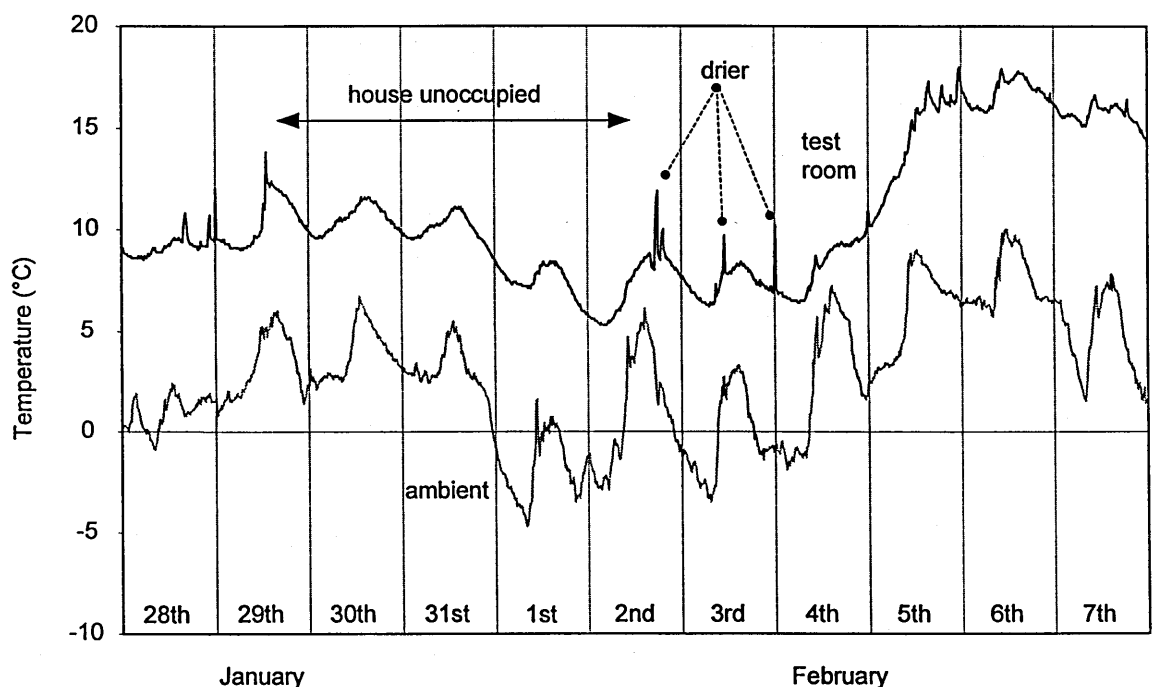


Fig 7-7 An example of occupancy creating 'noise' that the network does not have causal information to model. The unoccupied period can be seen to be 'noise' free.

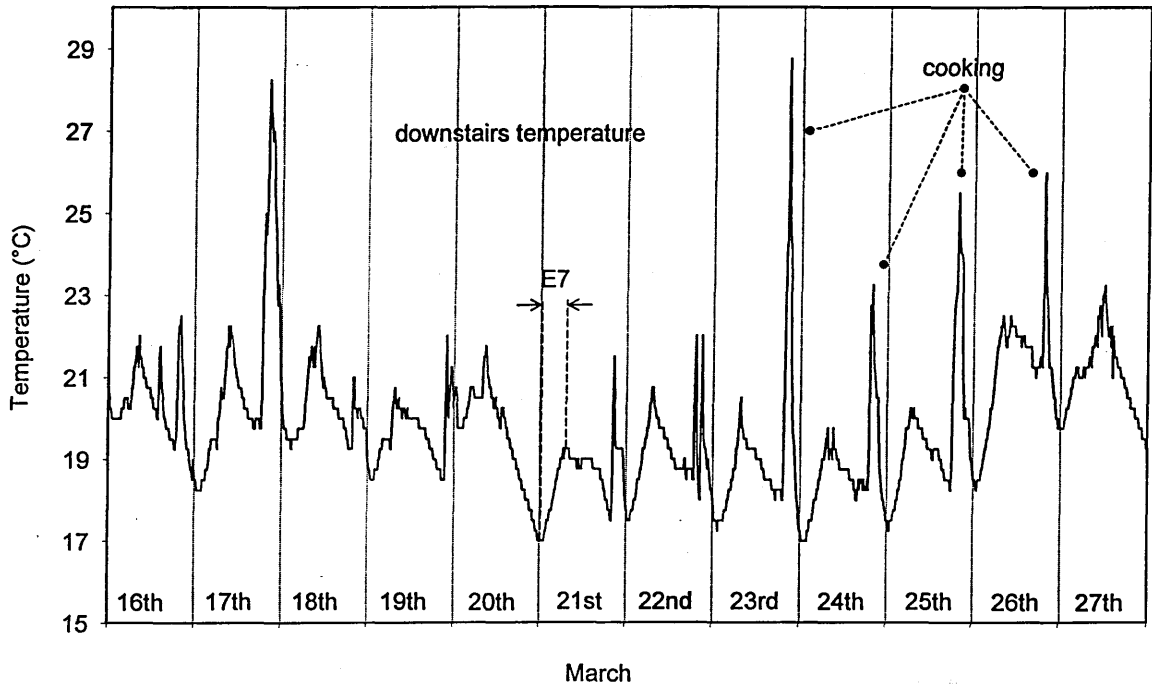


Fig 7-8 An example of temperature fluctuations in the downstairs room

Fig 7-8 shows an example of the temperature fluctuations that occur in the room below that will be transmitted into the test room through the floor and by natural air circulation. Short-term temperature increases in the late evening due to cooking and heater fan activation are very evident as is the underlying E7 room heating. These fluctuations will affect the test room temperature and are reasons why the noise is present in the model errors.

### 7.3 Neural Controller

In the prototype controller it was considered ambitious to attempt both predictive control and optimisation in the first instance. The only objective required was to track a given temperature by implementing control set points of the core temperature determined 5 hours previous. This was implemented by using the neural emulator as shown in Fig 7-9 and Fig 7-1.

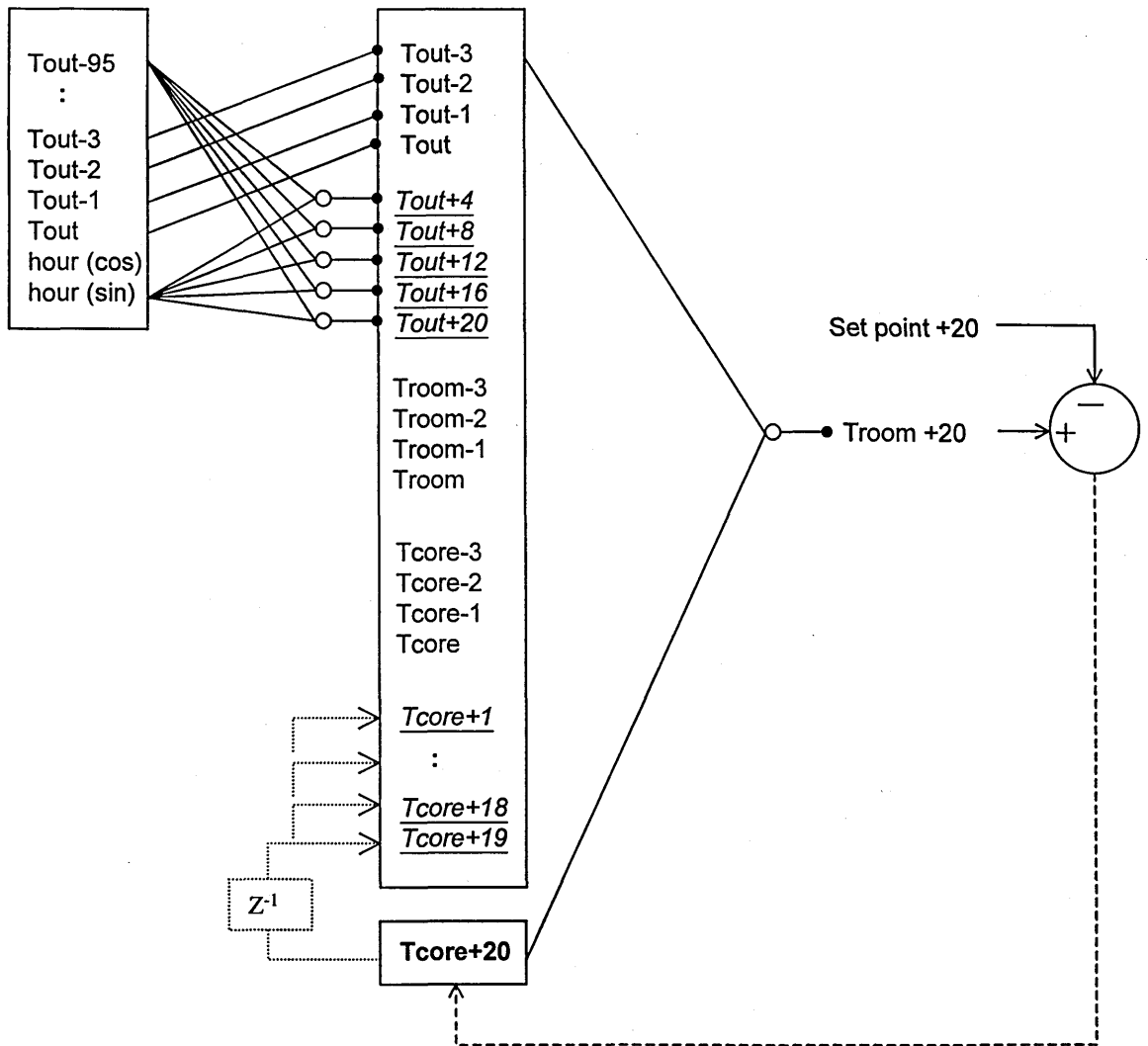


Fig 7-9 The neural emulator created for the heater control. Every 15 minutes a new core temperature for 5 hours time is calculated ( $T_{core+20}$ ) by finding such a temperature that minimises the error between the set point and the model prediction. There are upper and lower bounds on this core temperature determined by what it is physically possible to achieve from its previous state. Once a core temperature is determined it is added to a stack and the controller acts as a thermostat switch that tries to achieve the core temperature at the top of the stack ( $T_{core+1}$ ). Once a core temperature is determined and enters the stack it cannot be adjusted.



The emulator consisted of 6 neural networks. Five of these were to predict the ambient temperature 1,2,3,4 and 5 hours ahead. These predictions were then used by the main network to determine the required core temperature.

The network inputs to predict the ambient temperature were the previous 24 hours temperatures along with the hour of the day. The output was the temperature either 1,2,3,4 or 5 hours ahead. Including the hour of the day, coded as a sine and cosine, improved the model performance by giving some reference as to when the turning points would occur. Each network weights were trained individually, not as a single network with 5 outputs.

The network weights were calculated off line and then implemented in the controller with no further training. Fig 7-10 shows the errors of the created emulator on the training data, showing the created model predicting a temperature within  $1^{\circ}\text{C}$  in 81% of cases.

The controller hardware was an IAC600 donated by Satchwell Controls with the software custom written Visual Basic. As different temperature sensors to the data logger were used some calibration was required which is a possible source of error.

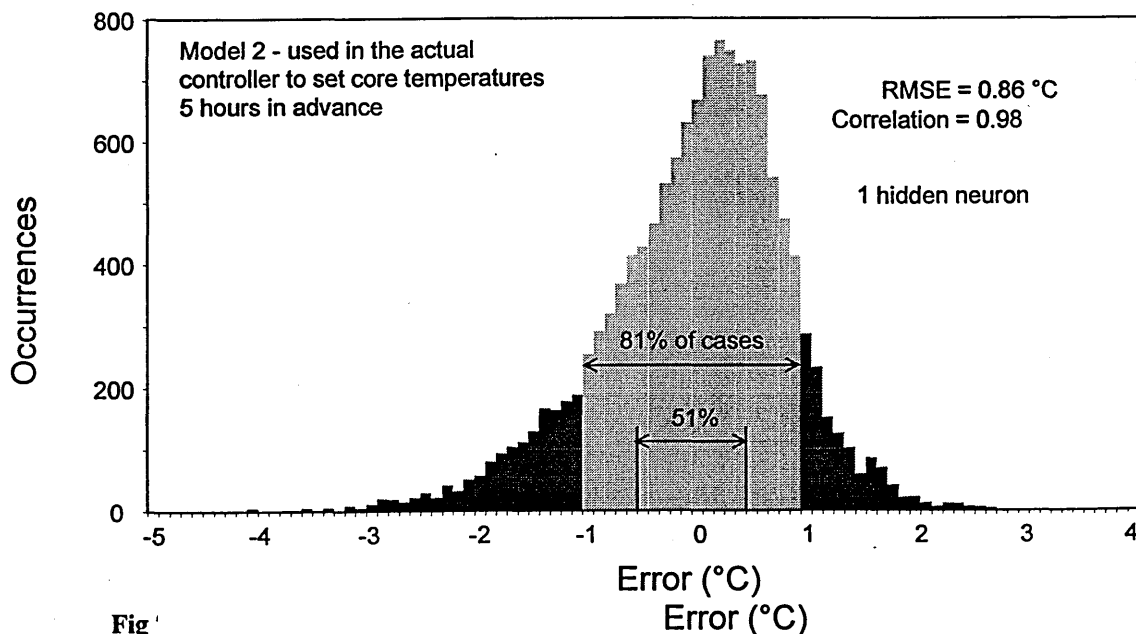


Fig 7-10

The errors on the training data for the model implemented in the controller

Fig 7-11 shows the results over two periods in March, indicating how the set point was continuously varied. Because storage radiators have a slow response the set point cannot always be achieved. The reported error is thus the difference between the model prediction and the achieved temperatures.

The rms error over the final two week period 17th-30th March was  $1.06^{\circ}\text{C}$  with a naïve prediction error being  $1.57^{\circ}\text{C}$ . How these relate to the training data is shown in Fig 7-5 (on page 151). Because there is set point temperature control the naïve prediction is better than on the training data when there was no temperature control. This is because temperatures are more constant and thus likely to be the same 5 hours ahead. The actual rms error is slightly less than would have been expected if no intervening ambient temperatures were used. This would suggest that the temperature predictions were poor, as was later confirmed.

The errors can be seen to be constantly underestimated, which is probably due to a slight calibration error. If this was not the case and they had a mean error of zero then the rms

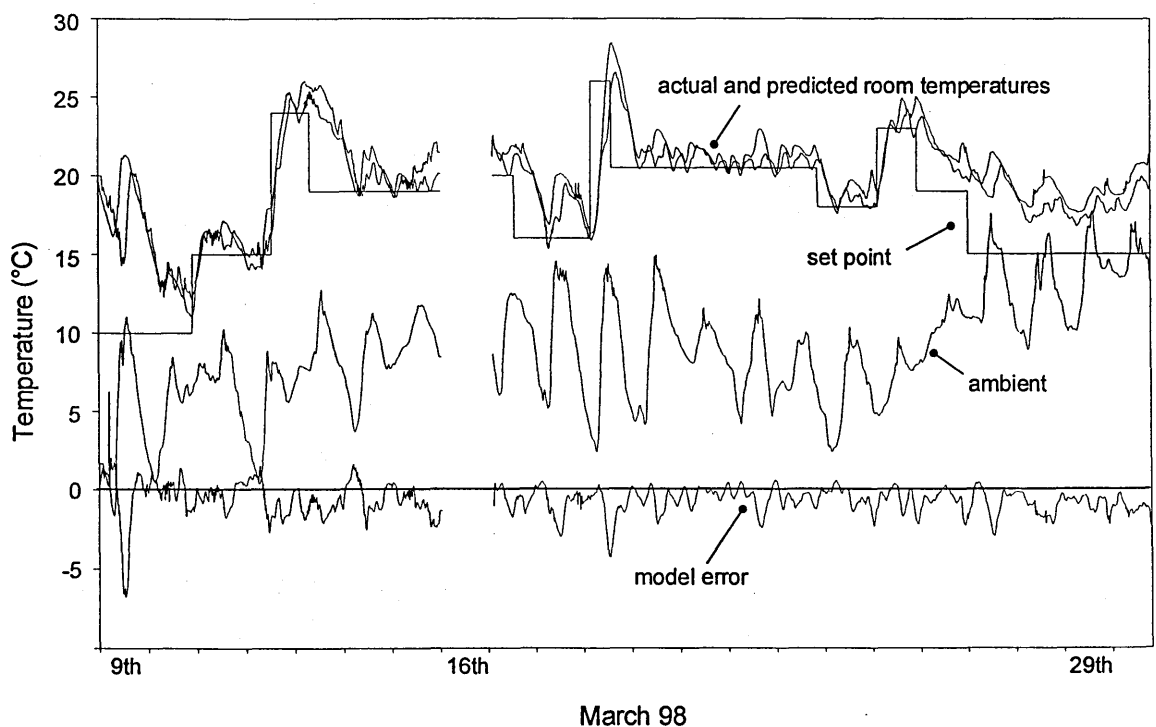


Fig 7-11 The results of the neuro-controlled storage radiator. The 'de-bugged' controller operated constantly for two weeks from the 17th March.

error would be significantly reduced.

Fig 7-12 shows the bedroom temperature over the same period as for the neuro-control. The room is smaller but has an identical heater operating on E7 and is thus useful to gauge what the uncontrolled test room response might have been. Daily temperature swings of up to  $4^{\circ}\text{C}$  are common, which would have been even greater in a larger room with more exterior wall area. It is also noticeable that the room is at its coldest in the late evening when warmth will be desired. The control achieved by the neuro-controller can thus be considered a success.

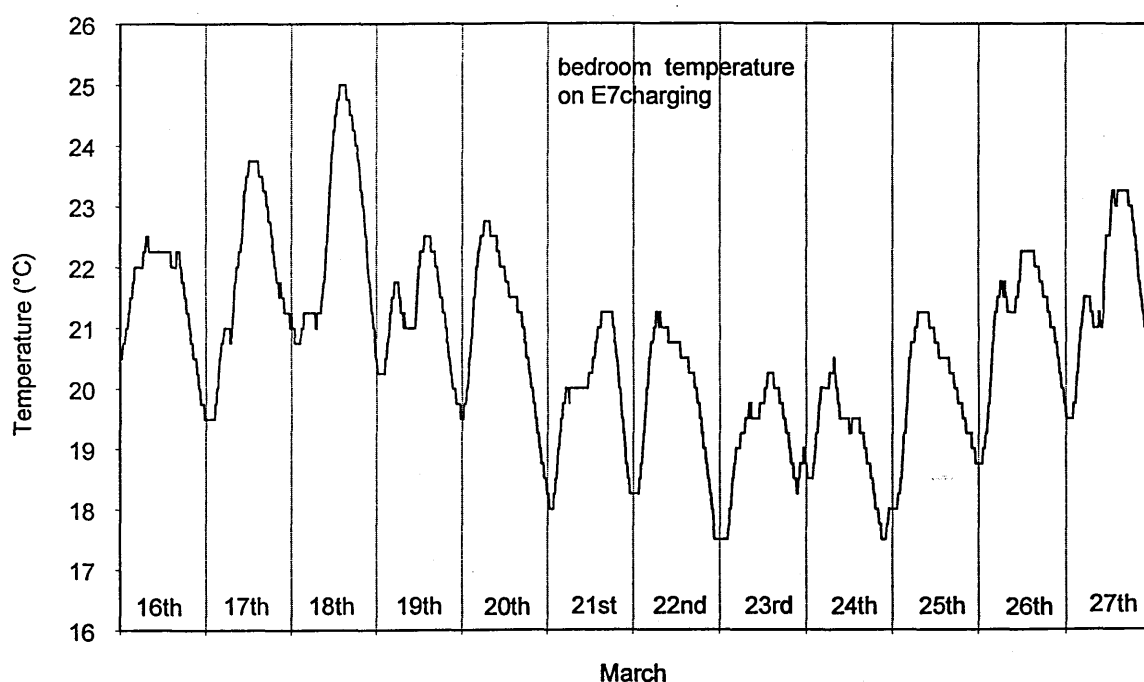


Fig 7-12 The bedroom temperatures over the same period as for the neuro-control with an identical heater operating on an E7 charging schedule

## 7.4 Chapter Summary

This chapter has reported the results of a prototype neuro-controller for a storage radiator. Important lessons have been learnt that would not be evident in simulation, namely the introduction of noise caused by occupants and the importance of heat gains through floors.

# 8

## Project Overview

---

### 8.1 About this Chapter

The New Product Development Department of Eastern Electricity sponsored this research project to develop an intelligent controller for storage radiators. Soon after commencement the decision was taken to close the department, ending any interest they had in the research and in any future development of a product.

This chapter summarises the key stages of the work, giving recommendations of how the results might be of use to Eastern. The most important offering that can probably be given is a description of how this neural network project evolved over the four years, highlighting important lessons that were learnt. This will hopefully be of use to others embarking on similar work.

## 8.2 Load Forecasting

The load forecasting techniques developed should be of interest to Eastern as electricity trading in the free market will require accurate predictions to cut costs and stay ahead of the competition. Initially, all Eastern wanted to know from their data was a growth trend over the years, which was achieved, with much more besides.

Compared with existing commercial products the techniques developed deal with growth and holiday periods in much more detail. The reported results, although not directly comparable, appear superior.

American utilities are investing heavily in neural forecasting techniques, with investments in commercial software of up to \$100,000 [98] resulting in improved forecasting accuracy [19,20] and financial savings. With such large amounts of money being paid for a technology in its infancy there is potential for developing commercial software, which would be very easy to accomplish.

## 8.3 Water Optimisation

The results from the simulations showed that savings were made by avoiding the mid-night price peaks, caused by the E7 surge. This can be achieved by utilising the radio tele-switch control of off-peak heating systems. This option has not been utilised by Eastern until very recently, with switching occurring at the same times every day. Although it is in the tariff contract that times can be varied, the reasons for not exercising this control option appears to be a fear of upsetting customers. If this is the case then it was an expensive investment installing the tele-switches in the first place. Complaints from customers since the switching times were varied have been very few.

## 8.4 Intelligent Heating Control

Development of a commercial product for domestic storage radiator control is not an immediate necessity, as other technologies have to fall in place to provide the infrastructure required. These include half-hourly meters, real time tariffs and communication channels for weather and pool price data. The last four years has seen exponential growth in telecommunication and microchip technologies including the internet and the use of home PC's. What was probably considered a fantasy product four years ago is now very much achievable at ever decreasing costs.

It would be advisable for anyone wanting to develop the idea to start in commercial buildings, where the potential of large savings might encourage the initial investment costs. Any heating system could be optimised by learning the building characteristics, it does not have to be limited to storage radiators.

It is only a matter of time before the 'smart house' becomes a reality and there is the opportunity for money to be made.

## 8.5 Experiences of Pursuing a Neural Network Project

At the start of this research it has to be said that neither I, or anyone involved with the project really new what a neural network was, apart from it was something that could learn things. As a result, there was a lot of time wasted and things were probably done the hard way. The following is a list of key points that directly influenced the research.

- 1) A lot of early time was wasted trying to read 'introductory' textbooks and journal papers. The best and quickest way to find out what neural networks are is to have it explained.

- 2) During a visit to the DTI Neural Awareness Campaign I was informed that there was an abundance of 'free' neural simulators on the internet. One was downloaded (SNNS) and basically played with for a while.
- 3) A chance meeting with a fellow student who was having problems coding her own neural network (and seeking my advice!) introduced me to the idea that neural networks were not very difficult to create.
- 4) The development of my own code in a suitable language. Fortran 90 was chosen because it is simple to follow and has many intrinsic matrix multiplication functions, which is all neural networks are.
- 5) The decision to switch from a UNIX system to PC's made the coding much more portable and it could be done at home.
- 6) Realising that to simulate all the various components of a controller, the whole system would have to be coded, unless a tool like MATLAB was used. It was decided that using custom code is much better for research purposes as you can control exactly what is happening and are not reliant on someone else's algorithms. Initially a building thermal analysis package was used but it was soon realised that this could not be interfaced with a controller.
- 7) It was soon evident that no real understanding of how neural networks really work could be achieved by using simulated data, as they contained no noise. The event that redirected the whole research focus was the analysis of the 'real' electrical load data.
- 8) A neural model was created for the load data and presented to Eastern. It was then that a quite reasonable question was asked, 'how do your network weights relate to my linear regression weights'. This was not the type of question people are supposed to ask as a neural network is a black box, isn't it? It was then realised that there must be some meaning in the weights, only no one had really bothered investigating this before.

9) It was then realised that neural networks are non-linear regression and definitely NOT artificial intelligence.

## **8.6 Cost Analysis**

Because this research has been mainly computer based the equipment costs have been minimal. The 'extra unavoidable' costs required were £300 for a Fortran90 compiler for PC's and roughly £200 for wiring and thermocouples for the controller. This works out at around £2.50 per week. Neural network projects can be very cheap, the major investment required is in human resources.

## **8.7 The Future**

Once the hard work of the first application is overcome, further applications can be achieved almost instantly. The possibilities are then just a matter of a creative imagination and inquisitive mind.



## References

---

- [1] S.C. Littlechild, "Spot pricing of electricity – Arguments and prospects," *Energy Policy*, August, 1998, pp. 398-403.
- [2] F.P. Shiohansi, "The pros and cons of spot pricing – electric utility perspective," *Energy Policy*, August, 1998, pp. 353-358.
- [3] J.R. McDonald, P.A. Whiting, and K.L. Lo, "Spot-pricing: evaluation, simulation and modelling of dynamic tariff structures," *Electrical Power and Energy Systems*, vol. 16, no. 1, 1994, pp. 23-34.
- [4] J.R. McDonald, P.A. Whiting, and K.L. Lo, "Optimized reaction of large electrical consumers in response to spot-price tariffs," *Electrical Power and Energy Systems*, vol. 16, no. 1, 1994, pp. 35-48.
- [5] F. Solis and J. Wets, "Minimisation by random search techniques," *Mathematics of Operation Research*, vol. 6, 1981.
- [6] K. KrishnaKumar, "Genetic algorithms - a robust optimization tool", AIAA report 93-0315.
- [7] B. Yoon, D.J. Holmes, G. Langholz, and A. Kandel, "Efficient genetic algorithms for training layered feedforward neural networks", *Information Sciences*, vol. 76, 1994, pp. 67-85.
- [8] S. Haykin, *Neural Networks, A Comprehensive Foundation*, Macmillan College Publishing, 1994, pg 177.
- [9] L. Tarassenko, *A Guide to Neural Computing Applications*, Arnold, 1998, pg. 69,94 and 97.
- [10] L. Tarassenko, *A Guide to Neural Computing Applications*, Arnold, 1998, pg. 13 and 123.
- [11] See the *comp.ai.neural-nets FAQ's* at: <ftp://ftp.sas.com/pub/neural/FAQ2.html>  
Subject: Should I nonlinearly transform the data?
- [12] *Best Practice Guidelines for Developing Neural Computing Applications* DTI, 1994, pg. 152.

- 
- [13] See *Kangaroo's and Training Neural Networks* by Warren S. Sarle at <ftp://ftp.sas.com/pub/neural/kangaroos>.
- [14] T. Czernichow, A. Piras, K. Imhof, P. Caire, Y. Jaccard, B. Dorizzi, and A. Germond, "Short-term electrical load forecasting with artificial neural networks," *Engineering Intelligent Systems for Electrical Engineering and Communications*, vol. 4, no. 2, June 1996, pp. 85-99. (review).
- [15] G. Doyle and D. Maclaine, *Power as a Commodity*, Financial Times Energy Publishing, 1996.
- [16] X. Ma, A.A. El-Keib, R.E. Smith, and H. Ma, "A genetic algorithm based approach to thermal unit commitment of electric power systems," *Electric Power Systems Research*, vol. 34, no. 1, July 1995, pp. 29-36.
- [17] D.K. Ranaweera, G.G. Karady, and R.G. Farmer, "Economic impact analysis of load forecasting," *IEEE Trans. Power Systems*, vol. 12, no. 3, August 1997, pp. 1388-1392.
- [18] D.W. Bunn and E.D. Farmer, *Comparative Models for Electrical Load Forecasting*, John Wiley & Sons Ltd, 1985, pp. 3-9.
- [19] B.F.Hobbs, S.Jitprapaikularn, S. Konda, V. Chankong, K.A. Loparo, and D.J. Maratukulam, "Analysis of the Value for Unit Commitment of Improved Load Forecasts," appearing in *IEEE Trans. Power Systems*.
- [20] B.F.Hobbs, U. Helman, S.Jitprapaikularn, S. Konda, and D.J. Maratukulam, "Artificial Neural Networks Short-Term Energy Forecasting: Accuracy and Economic Value," appearing in *International Journal of Neurocomputing*.
- [21] S. Rahman and I. Drezga, "Identification of a standard for comparing short-term load forecasting techniques," *Electric Power Systems Research*, vol. 25, no. 3, December 1992, pp. 149-158.
- [22] D. Park, M. El-Sharkawi, R. Marks, A. Atlas, and M. Damborg, "Electric load forecasting using an artificial neural network," Presented at the IEEE Power Engineering Society Winter Meeting, 1990.
- [23] T.M. Peng, N.F. Hubele, and G.G. Karady, "Conceptual approach to the application of neural network for short-term load forecasting," *Proceedings of the 1990 IEEE International Symposium on Circuits and Systems*, pp. 2942-2945.
- [24] A. Khotanzad, R. Afkhami-Rohani, T.L. Lu, A. Abaye, M. Davies, and D.J. Maratukulam, "ANNSTLF - a neural-network-based electric load forecasting system," *IEEE Trans. Neural Networks*, vol. 8, no. 4, July 1997, pp. 835-845.

- 
- [25] see <http://www.neusolutions.com>
- [26] D.K. Ranaweera, N.F. Hubele, and A.D. Papalexopoulos, "Application of radial basis function neural-network model for short-term load forecasting," *IEEE Proceedings- Generation Transmission and Distribution*, vol. 142, no. 1, January 1995, pp. 45-50.
- [27] B. Xiao and P.G. McLaren, "An artificial neural network for short term load forecasting," *Proceedings of the Communications Power and Computing Conference, WESCANEX '95*, 95CH3581-6/0-7803-2741-1/95, pp. 129-132.
- [28] P.K. Dash, H.P. Satpathy, A.C. Liew, and S. Rahman, "A real-time short-term load forecasting system using functional link network," *IEEE Trans. Power Systems*, vol. 12, no. 2, May 1997, pp. 675-680.
- [29] J.K. Mandal, A.K. Sinha, and G. Parthasarathy, "Application of recurrent neural network for short term load forecasting in electric power system," *Proceedings of the 1995 IEEE International Conference on Neural Networks*, 0-7803-2768-3/95, pp. 2694-2698.
- [30] M.A. El-Sharkawi, R.J. Marks, S. Oh, and C.M. Brace, "Data partitioning for training a layered perceptron to forecast electric load," *Proceedings of the IEEE 2nd International Forum on Applications of Neural Networks to Power Systems*, April 1993, Yokohama, Japan, 0-7803-1217-1/93, pp. 66-68.
- [31] Y. Al-Rashid and L.D. Paarmann, "Short-term electric load forecasting using neural network models," *Proceedings of the 39th Midwest Symposium on Circuits and Systems*, 1997, 0-7803-3636-4/97, pp. 1436-1439.
- [32] S.P. Singh and O.P. Mailk, "Single ANN architecture for short-term load forecasting for all seasons," *Engineering Intelligent Systems for Electrical Engineering and Communications*, vol. 3, no. 4, December 1995, pp. 249-254.
- [33] A.G. Bakirtzis, V. Petridis, S.J. Klartzis, M.C. Alexiadis, and A.H. Maissis, "A neural-network short-term load forecasting model for the Greek power-system," *IEEE Trans. Power Systems*, vol. 11, no. 2, May 1996, pp. 858-863.
- [34] A.D. Papalexopoulos, S.Y. Hao, and T.M. Peng, "An implementation of a neural network based load forecasting model for the EMS," *IEEE Trans. Power Systems*, vol. 9, no. 4, November 1994, pp. 1956-1962.
- [35] O. Mohammed, D. Park, R. Merchant, T. Dinh, C. Tong, A. Azeem, J. Farah, and C. Drake, "Practical experiences with an adaptive neural-network short-term load forecasting system," *IEEE Trans. Power Systems*, vol. 10, no. 1, February 1995, pp. 254-265.
- [36] J.T. Connor, "A robust neural network filter for electricity demand prediction," *Journal of Forecasting*, vol. 15, no. 6, November 1996, pp. 437-458.

- [37] Y. Shimakura, Y. Fujisawa, Y. Maeda, R. Makino, Y. Kishi, M. Ono, J.Y. Fann, and N. Fukusima, "Short-term load forecasting using an artificial neural network," *Proceedings of the IEEE 2nd International Forum on Applications of Neural Networks to Power Systems*, April 1993, Yokohama, Japan, 0-7803-1217-1/93, pp. 233-238.
- [38] T. Matsumoto, S. Kitamura, Y Ueki, and T. Matsui, "Short-term load forecasting by artificial neural networks using individual and collective data of preceding years," *Proceedings of the IEEE 2nd International Forum on Applications of Neural Networks to Power Systems*, April 1993, Yokohama, Japan, 0-7803-1217-1/93, pp. 245-250.
- [39] O. Hyde and P.F. Hodnett, "Modelling the effect of weather in short-term electricity load forecasting," *Mathematical Engineering in Industry*, vol. 6, no. 2, 1997, pp. 155-169.
- [40] M. Rasanen, R.P. Hamalainen, and J. Ruusunen, "Visual interactive modeling in electric load analysis," *Proceedings of th 13th IASTED International Conference on Modelling, Identification and Control*, February 1994, 0-88986-138-8, pp. 339-342.
- [41] J.M. Agosta, N.R. Nielsen, and G. Andeen, "Fast training of neural networks for load forecasting," *Proceedings of the American Power Conference*, 1996, vol. 58, no. 1, pp. 219-224.
- [42] R.S. Zebulum, M. Vellasco, M.A. Pacheco, and K. Guedes, "A multi-step hourly load forecasting system using neural nets," *Proceedings of the 38th Midwest Symposium on Circuits and Systems*, 1996, 0-7803-2972-4/96, pp. 461-464.
- [43] L. Tarassenko, *A Guide to Neural Computing Applications*, Arnold, 1998, pg. 85.
- [44] T.M. Peng, N.F. Hubele, and G.G. Karaday, "Advancement in the Application of Neural Networks for Short-Term Load Forecasting", *IEEE Trans. on Power Systems*, vol. 7, no.1 ,1992, pp. 250-257.
- [45] D.J.C. Mackay, "Bayesian nonlinear modeling for the prediction competition," *ASHRAE Transactions*, 1994, vol. 100, pt. 2, pp. 1053-1062.
- [46] J.M. Hannan, S. Majithia, C. Rogers, and R.J. Mitchell, "Implementation of neural networks for forecasting cardinal points on the electricity demand curve," *4th International Conference on Power System Control and Management*, April 1996, pp. 160-164.
- [47] O. Hyde and P.F. Hodnett, "An adaptable automated procedure for short-term electricity load forecasting," *IEEE Trans. Power Systems*, vol. 12, no. 1, February 1997, pp. 84-93.

- 
- [48] L.A. Kiernan, J.M. Hannan, M.Bishop, R.J. Mitchell, and C. Kambhampati, "Neural networks for load profile reshaping," *Proceedings of the 29th Universities Power Engineering Conference*, 1994, pp. 358-361.
- [49] Y.Y. Hsu and C.C. Yang, "Design of artificial neural networks for short-term load forecasting 1. self-organizing feature maps for day type identification," *IEEE Proceedings-C Generation Transmission and Distribution*, vol. 138, no. 5, September 1991, pp. 407-413.
- [50] L. Kiernan, C. Kambhampati, R.J. Mitchell, and K. Warwick, "Automatic integrated system load forecasting using mutual information and neural networks," *Control of Power plants and Power Systems, SIPOWER '95*, Cancun, Mexico, 1995, pp. 503-508.
- [51] B.P. Feuston and J.H. Thurtell, "Generalized nonlinear regression with ensemble of neural nets : the great energy predictor shootout," *ASHRAE Trans.*, vol. 100, pt. 2, 1994, pp. 1075-1080.
- [52] I. Moghram and S. Rahman, "Analysis and evaluation of five short-term load forecasting techniques," *IEEE Trans. Power Systems*, vol. 4, no. 4, October 1989, pp. 1484-1491.
- [53] G. Gross and F.D. Galiana, "Short-term load forecasting," *Proceedings of the IEEE*, vol. 75, no. 12, December 1987, pp. 1558-1573.
- [54] IEEE Committee Report, "Load Forecasting Bibliography Phase 1," *IEEE Trans. Power App. Sys.*, vol. 99, pp. 53-58.
- [55] K. Liu, S. Subbarayan, R.R. Shoults, M.T. Mantry, C. Kwan, F.L. Lewis and J. Naccarino, "Comparison of very short term load forecasting techniques," *IEEE Trans. Power Systems*, vol. 11, no. 2, May 1996, pp. 877-882.
- [56] J. Blake, P. Francino, J.M. Catot and I. Sole, "A comparative study for forecasting using neural networks vs genetically identified Box&Jenkins models," *Neural Comput and Applic*, vol. 3, 1995, pp. 139-148.
- [57] B.E. Bitzer, "European consortium studies intelligent forecasting," *IEEE Computer Applications in Power*, April 1997, pp. 36-38.
- [58] Mailridakis, Wheelwright and McGee, *Forecasting methods and applications*, Wiley 1983.
- [59] A.D. Papalexopoulos and T.C. Hesterberg, "A regression-based approach to short-term load forecasting," *IEEE Trans. Power Systems*, vol. 5, no. 4, November 1990, pp. 1535-1547.

- 
- [60] T. Haida and S. Muto, "Regression based peak load forecasting using a transformation technique," *IEEE Trans. Power Systems*, vol. 9, no. 4, November 1994, pp. 1535-1547.
- [61] G.T. Heinemann, D.A. Nordman and E.C. Plant, "The relationship between summer weather and summer loads – a regression analysis," *IEEE Trans. Power App. Sys.*, vol. 85, no. 12, November 1966, pp.1144-1154.
- [62] M.T. Hagan and S.M. Behr, "The time series approach to short term load forecasting," *IEEE Trans. Power Systems*, vol. 2, no. 3, August 1987, pp. 785-791.
- [63] W.M. Grady, L.A. Groce, T.M. Huebner, Q.C. Lu and M.M. Crawford, "Enhancement, implementation and performance of an adaptive short term load forecasting algorithm," *IEEE Trans. Power Systems*, vol. 6, no. 4, November 1991, pp. 1404-1410.
- [64] G.E.P. Box and G.M. Jenkins, *Time Series Analysis: Forecasting and Control* Holden-Day, 1976.
- [65] J.L. Piggot, "Short-term forecasting at British gas," Chapter 11 in D.W. Bunn and E.D. Farmer, *Comparative Models for Electrical Load Forecasting*, John Wiley & Sons Ltd, 1985.
- [66] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.
- [67] L. Altenberg, "Evolving better representations through selective genome growth," *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, vol.1, 1994, pp. 182-187.
- [68] M. Mitchell, *An introduction to Genetic Algorithms*, The MIT press, 1996.
- [69] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
- [70] M. Mitchell, S. Forrest, and J.H. Holland, "The royal road for genetic algorithms: Fitness landscapes and GA performance," *Proceedings of the First European Conference on Artificial Life*, 1992, pp. 245-254.
- [71] S. Forrest and M. Mitchell, "Relative building-block fitness and the building-block hypothesis," In *Foundations of Genetic Algorithms 2*, Morgan Kaufmann, 1993, pp.109-126.
- [72] M. Mitchell, J.H. Holland, and S. Forrest, "When will a genetic algorithm outperform hill climbing?," In *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, 1994.

- 
- [73] T. Walsh, "Empirical methods in AI," *AI Magazine*, summer 1998, pp.121-124.
- [74] K.A. De Jong, "Genetic algorithms are NOT function optimizers," In *Foundations of Genetic Algorithms 2*, Morgan Kaufmann, 1993, pp.5-17.
- [75] A.E. Nix and M.D. Vose, "Modeling genetic algorithms with Markov chains," *Annals of Mathematics and Artificial Intelligence*, vol. 5, 1992, pp.79-88.
- [76] S. Kirkpatrick, D.C. Gellat and M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, 1983, pp. 671-680.
- [77] A.H.G. Rinnooy Kan and G.T. Timmer, "Global optimisation: a survey," *International Series of Numerical Mathematics*, vol. 87, 1989, pp. 133-155. Birkhauser Verlag Basel.
- [78] F. Glover, "Tabu search I," *ORSA J. Comp.*, vol. 1, 1989, pp. 190-296.
- [79] E.L. Lawler and D.E. Wood, "Branch and bound methods : a survey," *Journal of Oper. Res.*, vol. 14, 1966, pp. 699-719.
- [80] D.H. Wolpert and W.G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, April 1997, pp. 67-82.
- [81] B. Rautenbach and I.E. Lane, "The multi-objective controller: a novel approach to domestic hot water load control," *IEEE Trans. Power Systems*, vol. 11, no. 4, November 1996, pp. 1832-1837.
- [82] S.J. Dickinson, "A building heating system simulation and optimisation tool incorporating bond graphs and genetic algorithms," PhD Thesis, Lancaster University, 1996.
- [83] Symposium papers on "Predicting hourly building energy usage: the great energy predictor shootout – the aftermath," *ASHRAE Transactions*, 1994, vol. 100, pt. 2, pp. 1053-1118. (also see refs 37 and 43)
- [84] A. Dhar, "Development of Fourier series and artificial neural network approaches to model hourly energy use in commercial buildings," PhD Thesis, Texas A&M University, May 1995.
- [85] P.Y. Glorennec, "Modélisation d'un bâtiment par un réseau neuronal récurrent," Présenté aux Septièmes Journées Internationales Les Réseaux Neuromimétiques et leurs Applications, Paris, October 1994. (in French)
- [86] M.C. Mozer, R.H. Dodier, M. Anderson, L. Vidmar, R.F. Cruickshank III, and D. Miller, "The Neural Network House: An overview," In L. Niklasson & M. Boden (Eds.) *Current trends in connectionism* , 1995, pp. 371-380. Also online <http://www.cs.colorado.edu/~mozer/nnh/index.html>

- 
- [87] M.C. Mozer, L. Vidmar, and R.H. Dodier, "The neurothermostat: predictive optimal control of residential heating systems," In M.C. Mozer, M.I Jordan & T. Petsche (Eds.), *Advances in neural information processing systems 9*. 1997.
- [88] K.O. Temeng, P.D. Schnelle, and T.J. McAvoy, "Model predictive control of an industrial packed bed reactor using neural networks," *J. Proc. Cont.*, vol. 5, no. 1, 1995, pp.19-27.
- [89] G. Gvazdaitis, S. Beil, U. Kreibaum, R. Simutis, I. Havlik, M. Dors, F. Schneider, and A. Lubbert, "Temperature control in fermenters: application of neural nets and feedback control in breweries," *J. Inst. Brew.*, Vol. 100, 1994, pp. 99-104.
- [90] L. Clark, "Smart meter gives you the power," Sunday Star-Times, November 30 1997, pp. D5. (New Zealand newspaper) relating to Exicom Technologies (1996).
- [91] B. Daryanian, R.E. Bohn, and R.D. Tabors, "An experiment in real time pricing for control of electric thermal storage systems," *IEEE Trans. Power Systems*, vol. 6, no. 4, November 1991, pp.1356-1365.
- [92] B. Daryanian, R.E. Bohn, "Sizing of electric thermal storage under real time pricing," *IEEE Trans. Power Systems*, vol. 8, no. 1, February 1993, pp. 35-43.
- [93] K.L. Lo, J.R. McDonald, and T.Q. Le, "Time-of-day electricity pricing incorporating elasticity for load management purposes," *Electrical Power and Energy Systems*, vol. 13, no. 1, 1991, pp.230-239.
- [94] M. Räsänen, J.Ruusunen, and R.P. Hämäläinen, "Customer level analysis of dynamic pricing experiments using consumption-pattern models," *Energy*, vol. 20, no. 9, 1995, pp.897-906.
- [95] M. Räsänen, J.Ruusunen, and R.P. Hämäläinen, "Optimal tariff design under consumer self-selection," *Energy Economics*, vol. 19, 1997, pp. 151-167.
- [96] J. Zarnikau, "Customer responsiveness to real-time pricing of electricity," *Energy*, vol. 11, 1990, pp. 99-116.
- [97] A.K. David and Y.Z. Li, "Effect of inter-temporal factors on the real time pricing of electricity," *IEEE Trans. Power Systems*, vol. 8, no. 1, February 1993, pp. 44-52.
- [98] Ben Hobbs – International One Day Symposium on Electricity Load Forecasting – London Business School, 6th July 1998.
- [99] P.J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioural sciences," Ph.D. Thesis, 1974, Harvard University, Cambridge, MA.

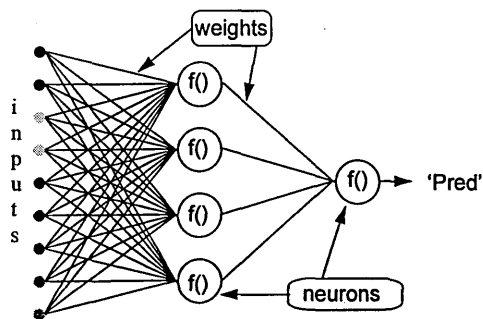


- 
- [100] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning internal representation by error propagation," In *Parallel Distributed Processing: Exploration in the Microstructure of Cognition* (D.E Rumelhart and J.L. McClelland, eds.) 1986 , vol. 1, chapter 8, Cambridge, MA, MIT Press.
- [101] 'Back Propagation family album' – Technical report C/TR96-05, Department of Computing, Macquarie University, NSW, Australia.  
[www.comp.mq.edu.au/research.html](http://www.comp.mq.edu.au/research.html) (Sept 1998)
- [102] S. Haykin , *Neural Networks – a Comprehensive Foundation*, Macmillan College Publishing Company 1994, Chapter 6, pg. 160.
- [103] P. Incropera and D.P. De Witt, *Fundamentals of heat and mass transfer*, John Wiley and Sons.

# Appendix A

## Back Propagation Weight Update Rule

---



This idea was first described in [99] and popularised by [100].

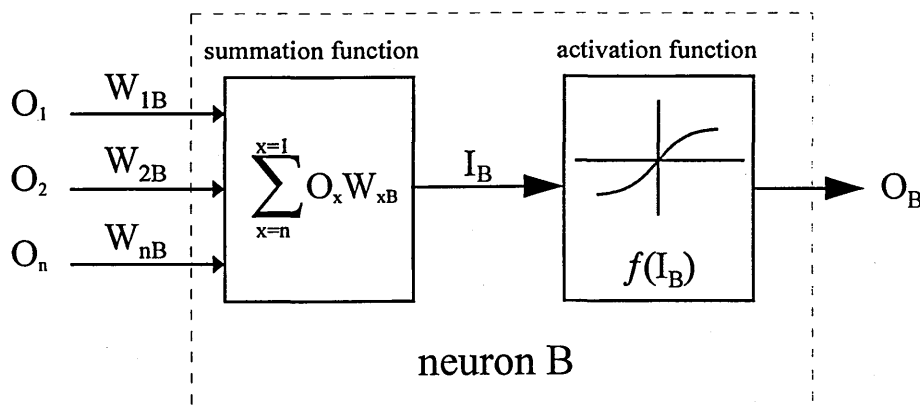
Consider the network above, with one layer of hidden neurons and one output neuron. When an input vector is propagated through the network, for the current set of weights there is an output 'Pred'. The objective of supervised training is to adjust the weights so that the difference between the network output 'Pred' and the required output 'Req' is reduced. This requires an algorithm that reduces the absolute error, which is the same as reducing the squared error, where,

$$\begin{aligned} \text{Network Error} &= \text{Pred} - \text{Req} \\ &= E \end{aligned} \tag{1}$$

The algorithm should adjust the weights such that  $E^2$  is minimised. Back-propagation is such an algorithm that performs a gradient descent minimisation of  $E^2$ .

In order to minimise  $E^2$ , its sensitivity to each of the weights must be calculated. In other words, we need to know what effect changing each of the weights will have on  $E^2$ . If this is known then the weights can be adjusted in the direction that reduces the absolute error.

The notation for the following description of the back-propagation rule is based on the diagram below,



The dashed line represents a neuron B, which can be either a hidden or the output neuron. The outputs of 'n' neurons ( $O_1 \dots O_n$ ) in the preceding layer provide the inputs to neuron B. If neuron B is in the hidden layer then this is simply the input vector.

These outputs are multiplied by the respective weights ( $W_{1B} \dots W_{nB}$ ), where  $W_{nB}$  is the weight connecting neuron n to neuron B. The summation function adds together all these products to provide the input,  $I_B$ , that is processed by the activation function  $f()$  of neuron B.  $f(I_B)$  is the output,  $O_B$ , of neuron B.

For the purpose of this illustration, let neuron 1 be called neuron A and then consider the weight  $W_{AB}$  connecting the two neurons.

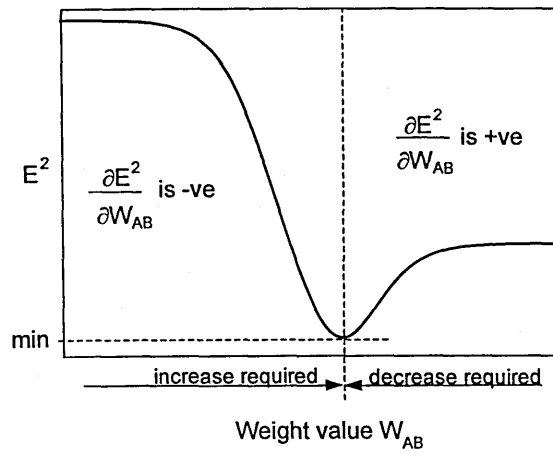
The approximation used for the weight change is given by the delta rule,

$$W_{AB(\text{new})} = W_{AB(\text{old})} - \eta \frac{\partial E^2}{\partial W_{AB}} \quad (2)$$

where  $\eta$  is the learning rate parameter, which determines the rate of learning and

$$\frac{\partial E^2}{\partial W_{AB}}$$

is the sensitivity of the error,  $E^2$ , to the weight  $W_{AB}$  and determines the direction of search in weight space for the new weight  $W_{AB}$  as illustrated in the figure below.



From the chain rule,

$$\frac{\partial E^2}{\partial W_{AB}} = \frac{\partial E^2}{\partial I_B} \frac{\partial I_B}{\partial W_{AB}} \quad (3)$$

and

$$\begin{aligned} \frac{\partial I_B}{\partial W_{AB}} &= \frac{\partial \sum_{x=n}^{x=1} O_x W_{xB}}{\partial W_{AB}} \\ &= \frac{\partial (O_A W_{AB})}{\partial W_{AB}} + \frac{\partial \sum_{x=n}^{x=2} O_x W_{xB}}{\partial W_{AB}} \\ &= O_A \end{aligned} \quad (4)$$

since the rest of the inputs to neuron B have no dependency on the weight  $W_{AB}$ .

Thus from (3) and (4), (2) becomes,

$$W_{AB(\text{new})} = W_{AB(\text{old})} - \eta \frac{\partial E^2}{\partial I_B} O_A \quad (5)$$

and the weight change of  $W_{AB}$  depends on the sensitivity of the squared error,  $E^2$ , to the input,  $I_B$ , of unit B and on the input signal  $O_A$ .

There are two possible situations,

1. B is the output neuron.
2. B is a hidden neuron.

**Considering the first case:**

Since B is the output neuron, the change in the squared error due to an adjustment of  $W_{AB}$  is simply the change in the squared error of the output of B.

$$\begin{aligned} \partial E^2 &= \partial (\text{Pred-Req})^2 \\ \frac{\partial E^2}{\partial I_B} &= 2(\text{Pred-Req}) \frac{\partial \text{Pred}}{\partial I_B} \\ &= 2E \frac{\partial f(I_B)}{\partial I_B} \\ &= 2E f'(I_B) \end{aligned} \quad (6)$$

combining (5) with (6) we get,

$$W_{AB(\text{new})} = W_{AB(\text{old})} - \eta O_A 2E f'(I_B) \quad (7)$$

the rule for modifying the weights when neuron B is an output neuron.

If the output activation function,  $f(\bullet)$ , is the logistic function then,

$$f(x) = \frac{1}{1+e^{-x}} = (1+e^{-x})^{-1} \quad (8)$$

differentiating (8) by its argument  $x$

$$f'(x) = -1(1+e^{-x})^{-2} \cdot -1(e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2} \quad (9)$$

But,

$$f(x) = \frac{1}{1+e^{-x}} \quad (10)$$

$$\Rightarrow e^{-x} = \frac{(1-f(x))}{f(x)} \quad (11)$$

inserting (11) into (9) gives,

$$\begin{aligned} f'(x) &= \frac{(1-f(x))}{f(x)} \bigg/ \frac{1}{(f(x))^2} \\ &= f(x) \times (1-f(x)) \end{aligned} \quad (12)$$

similarly for the tanh function,

$$f'(x) = (1-f(x)^2)$$

or for the linear (identity) function,

$$f'(x) = 1$$

This gives,

$$W_{AB(\text{new})} = W_{AB(\text{old})} - \eta O_A 2E O_B (1-O_B) \quad (\text{logistic})$$

$$W_{AB(\text{new})} = W_{AB(\text{old})} - \eta O_A 2E (1-O_B^2) \quad (\text{tanh})$$

$$W_{AB(\text{new})} = W_{AB(\text{old})} - \eta O_A 2E \quad (\text{linear})$$

Considering the second case:

B is a hidden unit.

$$\frac{\partial E^2}{\partial I_B} = \frac{\partial E^2}{\partial I_O} \frac{\partial I_O}{\partial O_B} \frac{\partial O_B}{\partial I_B} \quad (13)$$

where the subscript, o, represents the output neuron.

$$\frac{\partial O_B}{\partial I_B} = \frac{\partial f(I_B)}{\partial I_B} = f'(I_B) \quad (14)$$

$$\frac{\partial I_O}{\partial O_B} = \frac{\partial \sum_p O_p W_{pO}}{\partial O_B} \quad (15)$$

where p is an index that ranges over all the neurons including neuron B that provide input signals to the output neuron. Expanding the right hand side of equation (15),

$$\frac{\partial \sum_p O_p W_{pO}}{\partial O_B} = \frac{\partial O_B W_{BO}}{\partial O_B} + \frac{\partial \sum_{p \neq B} O_p W_{pO}}{\partial O_B} = W_{BO} \quad (16)$$

since the weights of the other neurons,  $W_{pO}$  ( $p \neq B$ ) have no dependency on  $O_B$ .

Inserting (14) and (16) into (13),

$$\frac{\partial E^2}{\partial I_B} = \frac{\partial E^2}{\partial I_O} W_{BO} f'(I_B) \quad (17)$$

Thus  $\frac{\partial E^2}{\partial I_B}$  is now expressed as a function of  $\frac{\partial E^2}{\partial I_O}$ , calculated as in (6).

The complete rule for modifying the weight  $W_{AB}$  between a neuron A sending a signal to a neuron B is,

$$W_{AB(\text{new})} = W_{AB(\text{old})} - \eta \frac{\partial E^2}{\partial I_B} O_A \quad (18)$$

where,

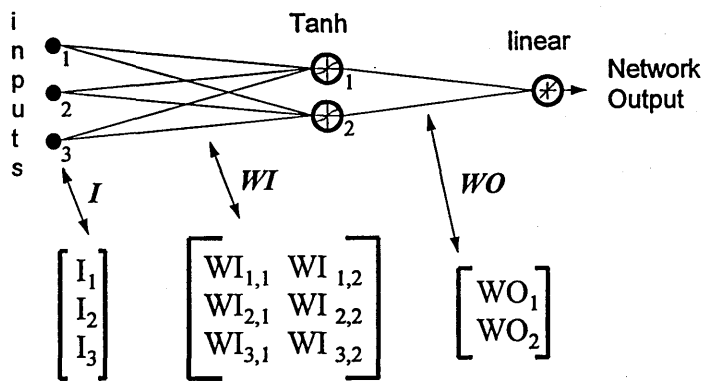
$$\frac{\partial E^2}{\partial I_B} = 2E f'_o(I_B) \quad \text{-}I_B \text{ is the output neuron} \quad (19)$$

$$\frac{\partial E^2}{\partial I_B} = \frac{\partial E^2}{\partial I_O} W_{BO} f'_h(I_B) \quad \text{-}I_B \text{ is a hidden neuron} \quad (20)$$

where  $f_o(\bullet)$  and  $f_h(\bullet)$  are the output and hidden activation functions respectively.



### Example



$$\text{Network Output} = [\text{Tanh}(I^T \cdot WI)] \cdot WO$$

let  $\text{HID} = [\text{Tanh}(I^T \cdot WI)]^T$  - the outputs of the hidden neurons

$$\text{ERROR} = (\text{Network Output} - \text{Required Output})$$

LR = learning rate

The weight updates become,

#### linear output neuron

$$WO = WO - (\text{LR} \times \text{ERROR} \times \text{HID}) \quad (21)$$

local gradient
input signal

#### tanh hidden neuron

$$WI = WI - \{ \text{LR} \times [\text{ERROR} \times WO \times (1 - \text{HID}^2)] \cdot I^T \}^T \quad (22)$$

local gradient
input signal

Equations 21 and 22 show that the weight change is an input signal multiplied by a local gradient. This gives a direction that also has magnitude dependent on the magnitude of the error. If the direction is taken with no magnitude then all changes will be of equal size which will depend on the learning rate.

There are many algorithms that have evolved from the original algorithm with the aim to increase up the learning speed. These are summarised in [101].

The algorithm above is a simplified version in that there is only one output neuron. In the original algorithm more than one output is allowed and the gradient descent minimises the total squared error of all the outputs. With only one output this reduces to minimising the error.

# Appendix B

## MLP Code

---

The following source code is for a multilayer perceptron trained with the original back propagation algorithm. The weights are updated after the presentation of each pattern. The hidden neurons have tanh activation functions and there is one output neuron that has a linear activation function. Hidden and output layer bias inputs of 1 are created. There are two individual learning rates for each layer of weights. The patterns are presented in a random order and an epoch said to have occurred when as many patterns have been presented as there are in the training set. On completion of each epoch it is decided whether to accept the new weights depending if the cost function has been lowered. This procedure could be done after each pattern presentation rather than after each epoch.

The code is written in Fortran 90 as it has convenient matrix multiplication routines. All undeclared variables are real unless the variable name begins with the letters I-N in which case they are integers. The routine for setting the random seed is specific to the `ftn90` compiler for PC's by NAG/Salford software, otherwise it should work on any Fortran 90 compiler. In the code anything following a `'!'` is a comment.

## The code starts here

```

PROGRAM Neural_Simulator
!!coded in Fortran 90 by Philip Brierley

!! declare arrays !!
REAL,ALLOCATABLE :: DATA(:,:),TRAININP(:,:),TRAINOUT(:)
REAL,ALLOCATABLE :: WIH(:,:),WHO(:),HVAL(:)
REAL,ALLOCATABLE :: WIHBEST(:,:),WHOBEST(:)
REAL,ALLOCATABLE :: INPUTS_THIS_PAT(:),DUMMY1(:,:),DUMMY2(:,:)

!! define some constants which can be varied!!
NO_OF_EPOCHS=10000      !number of epochs
NHIDDEN=5              !number of hidden neurons
ALR=0.1                !learning rate for input-hidden weights
BLR=0.01               !learning rate for hidden-output weights

!! open pattern file !!
OPEN(UNIT=10,FILE='AVE.PAT',STATUS='OLD')
READ(10,*)NPATS,INPUTS      !read no. of patterns, no. of inputs
NOUTPUTS=1                 ! number of outputs (fixed)

!! number the neurons !!
NHIDDEN=NHIDDEN+1         !accounts for bias to output
NDU=INPUTS+NOUTPUTS       !Number Data Units
INPPB=INPUTS+1            !INPut Plus Bias
NHS=INPPB+1               !Number Hidden Start
NHF=INPPB+NHIDDEN        !Number Hidden Finish
NOS=NHF+1                 !Number Output Start

!! set the array dimensions !!
ALLOCATE(DATA(NPATS,NDU))  !raw data read from file
ALLOCATE(TRAININP(NPATS,INPPB)) !input patterns
ALLOCATE(TRAINOUT(NPATS))  !output
ALLOCATE(WIH(INPPB,NHS:NHF)) !input-hidden weights
ALLOCATE(WIHBEST(INPPB,NHS:NHF)) !best weights
ALLOCATE(WHO(NHS:NHF))    !hidden-output weights
ALLOCATE(WHOBEST(NHS:NHF)) !best weights
ALLOCATE(HVAL(NHS:NHF))   !hidden neuron outputs
ALLOCATE(INPUTS_THIS_PAT(INPPB)) !pattern being presented
ALLOCATE(DUMMY1(1,NHS:NHF)) !dummy matrix
ALLOCATE(DUMMY2(INPPB,1))  !dummy matrix

!! read patterns from file !!
DO I=1,NPATS
READ(10,*)(DATA(I,K),K=1,NDU)
ENDDO
CLOSE(10)

!! create training inputs and outputs !!
TRAININP(:,1:INPUTS)=DATA(:,1:INPUTS)
TRAININP(:,INPPB)=1      !create input bias
TRAINOUT(:)=DATA(:,NDU)
DEALLOCATE(DATA)         !the array 'data' is no longer required

```

```

!! generate initial random weights !!
CALL DATE_TIME_SEED@                               !set seed for random number generator

DO K=NHS,NHF
  WHO(K)=((RANDOM()-0.5)*2)/10                       !generate initial weights
  DO J=1,INPPB
    WIH(J,K)=((RANDOM()-0.5)*2)/10                 !generate initial weights
  ENDDO
ENDDO

WIHBEST=WIH                                         !record of best weights so far
WHOBEST=WHO                                         !record of best weights so far
RMSEBEST=100000000                                  !the best rms error, initially set high

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! THE TRAINING STARTS HERE !!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

PRINT *,'epochs rms_error'                          !print to screen

DO M=1,NO_OF_EPOCHS
  DO I=1,NPATS

    !! select a pattern at random !!
    IPAT_NUM=NINT(RANDOM()*(NPATS-1))+1
    INPUTS_THIS_PAT(:)=TRAININP(ipat_num,:)
    OUTPUT_THIS_PAT=TRAINOUT(ipat_num)

    !! calculate the network output !!
    HVAL=MATMUL(TRANPOSE(WIH),INPUTS_THIS_PAT)      ! inputs × weights
    HVAL=TANH(HVAL)                                  ! tanh activation function1
    HVAL(NHF)=1                                       ! acts as output bias
    OUTPRED=SUM(WHO*HVAL)                             ! model output
    ER_THIS_PAT=(OUTPRED-OUTPUT_THIS_PAT)           ! model error

    !! change weight hidden - output !!
    WHO=WHO-(BLR*HVAL*ER_THIS_PAT)

    !! change weight input - hidden !!
    DUMMY2(:,1)=TRAININP(IPAT_NUM,:)
    DUMMY1(1,:)=ER_THIS_PAT*WHO*(1-(HVAL**2.00))
    WIH=WIH-(MATMUL(DUMMY2,DUMMY1)*ALR)

  ENDDO      !! (I) one more epoch done

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! evaluate 'fitness' of of the network after each epoch !!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

SQERROR=0                                           !! in this case the fitness function is the squared errors
DO J=1,NPATS
  INPUTS_THIS_PAT(:)=TRAININP(J,:)

```

<sup>1</sup> If the argument of the intrinsic Tanh function gets too large then floating point overflow will occur. An alternative method is to code using DO Loops and only use the tanh function if the argument,  $x < 20$  and  $x > -20$ . If  $x < -20$ ,  $\tanh(x) = -1$ , If  $x > 20$ ,  $\tanh(x) = 1$  else  $\tanh(x) = \text{TANH}(x)$ .

```

OUTPUT_THIS_PAT=TRAINOUT(J)
HVAL=TANH(MATMUL(TRANPOSE(WIH),INPUTS_THIS_PAT))
HVAL(NHF)=1
OUTPRED=SUM(WHO*HVAL)
ER_THIS_PAT=(OUTPRED-OUTPUT_THIS_PAT)
SQERROR=SQERROR+(ER_THIS_PAT**2)
ENDDO !(J)

RMSE=SQRT(SQERROR/NPATS)      !! root of the mean squared error

IF (RMSE<RMSEBEST) THEN      ! if the rms error has improved then
    WIHBEST=WIH              ! keep the new weights
    WHOBEST=WHO
    RMSEBEST=RMSE
ELSE
    ! else if it has increased then
    ! go back to the old weights
    WIH=WIHBEST
    WHO=WHOBEST
ENDIF

!! print errors to screen !!
PRINT *,M,RMSEBEST

ENDDO      !! (M) after all the epochs have been done !!

```

END PROGRAM Neural\_Simulator

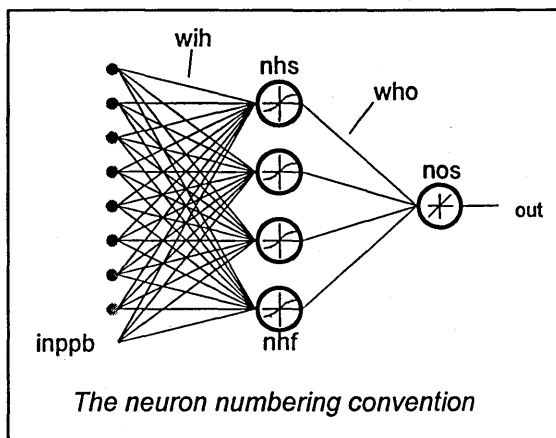
**The code finishes here**

The following is the contents of the demonstration pattern file called 'ave.pat'. The output is the average of the 3 inputs. The first line is the number of patterns and the number of inputs.

```

patterns, inputs
5,3
0.1,0.3,0,0.133 ← pattern 1
0.2,0.01,0.2,0.137
0.6,0.3,0.3,0.400
0.2,0.05,0.2,0.150
0.3,0.2,0.1,0.200 ← pattern 5
    ↑ input 1
    ↑ input 3
    ↑ output

```



# Appendix C

## More on MLPs

---

This Appendix is included to provide more information for those embarking on a project involving multilayer perceptrons. It attempts to answer, in no particular order, some of the questions that are likely to be asked and other advice that it would have been helpful to know at the onset of this project.

### Reference Sources

#### Books

Good reference books for those wanting to use MLPs to solve practical problems are few and far between. I found the following some of the better ones if you are an engineer and not a computer scientist.

*Simon Haykin, Neural Networks – A Comprehensive Foundation, 1994, Macmillan College Publishing Company.*

This book is described by the title. The first impression is that it looks quite daunting but chapter 6 on MLPs is quite readable.

*Kevin Swingler, Applying Neural Networks – A Practical Guide, 1996, Academic Press Limited.*

This is a valiant attempt to fill the gap of neural network books that can be read by non-rocket scientists who only want to understand and apply the techniques. Comes with a disc of software, which I cannot comment on because I did not use it.

*DTI, Neural Computing, Learning Solutions – Best Practice Guidelines for Developing Neural Computing Applications*

These guidelines were part of a 3 year neural network awareness campaign run by the DTI. They are specifically aimed at industry and contain many examples of practical applications. The campaign has finished but further information contact the Electronics and Engineering Division, Department of Trade and Industry, 151 Buckingham Palace Road, London SW1W 9SS. Contact names were Ray Browne and Bob Wiggins.

## The Web

There is a vast amount of information on the World Wide Web concerning neural networks. The following site at Pacific Northwest Laboratories has links to research groups and companies around the world involved in neural networks,

<http://www.emsl.pnl.gov:2080/proj/neuron/neural/gateway/> (Oct 98)



---

Another useful source of information is at the IEEE,

<http://www.ewh.ieee.org/tc/nnc/> (Oct 98)

There is a frequently asked questions list (FAQ's) at,

<ftp://ftp.sas.com/pub/neural/FAQ.html> (Oct 98)

and a newsgroup at,

news:comp.ai.neural-nets (Oct 98)

always check the FAQ list for the answer to your question before posting it to the newsgroup. A word of caution, do not always take as gospel replies you get from the newsgroup.

## Papers

There are many journal and conference papers on neural networks – many of them unreadable to mere mortals. If they are unintelligible then the chances are they are of no practical use. Trying to understand theory from papers is not a good place to start if you are new to the subject, and just because work is published does not mean that it is good.

Many practical applications of neural networks exist in non-neural network based journals and conferences. The chances are that no application is completely new and papers describing experiences with similar applications can provide some good ideas.

An example for the electricity industry,

'Neural networks, fuzzy logic and genetic algorithms in the electricity supply industry – Models and Applications,' Report of the UNIPEDE CORECH Working Group, June 1996, Eng. Int. Syst., September 1997, vol. 5, no. 3, pp. 127-155.

## Experts

It was not until several months into this project that I found out what a neural network really was. Introductory texts like to talk about the brain, artificial intelligence and consciousness, which is useless and often confusing information if all you want to know is how they work in practice.

If there had been access to an expert who could clearly explain everything and give good advice, then the time taken to complete this research would have probably been reduced by about 2 years. Saying this though, because there was no subjective suggestions of what could and could not be achieved, new ideas, explanations, view - points and solutions were explored with little regard for past paradigms.

I would recommend to anyone embarking on a neural network project to seek 'impartial' advice from an expert, although good ones are few and far between. Do not think buying a piece of neural software is the end of the story.

## How Many Hidden Neurons should be used?

MLPs can have many hidden layers with many neurons in each layer. Bigger is not necessarily better. Advice claiming to recommend the optimal number of hidden neurons based on network parameters such as the number of weights or training patterns is basically rubbish. As has been shown in Chapters 2 and 3 it depends on the complexity of the problem and how the data is encoded.

The technique developed in this thesis is to restrict the network by using a small number of hidden neurons and take clues from the errors as to how it can be improved. As the number of hidden neurons is increased the cost function (usually rms error) will decrease and then probably plateau or start to rise as more hidden neurons are added. It rises due to inefficiencies in the training algorithms that have problems setting weights to zero if they are not required.

A common approach to improving a model is to add more hidden neurons. Looking at the input data and how it is encoded will often be more worthwhile.

### What about Train and Test Sets?

A common technique used to create a model that can generalise is to divide the data into a training and test set. The test set is used as some unseen examples on which the performance of the network can be judged. It is usual practice to train until the error on the test set starts to rise (known as early stopping), at which point the model is losing its generalisation properties. Such techniques were not used in this thesis for several reasons,

- 1) Data is a scarce resource so as much as possible should be used. If the task is data mining then why throw some away?
- 2) Networks cannot extrapolate, so all the test set data must be carefully chosen to lie within the training ranges. A common mistake in load forecasting is to train on one year's data and use the next year as a test set. This is likely to give a poor model due to growth. Similarly one month might be used for training with the next for testing. Seasonal drift is likely to mean that new extremes of temperature are experienced.
- 3) A network trained on all the data can be retrospectively divided into a training and test set showing almost perfect generalisation properties. This set of weights does exist but would not be found by early stopping. Which set of weights is best?
- 4) If a network is poor at generalisation then there is some underlying reason such as too many hidden neurons, bad data or an incomplete or poor specification of the problem. The model will automatically be better at generalisation if such reasons are sought and rectified.

## Why Scale the Training Data?

Theoretically, input data does not need to be scaled as it is fed directly into a weight that will change the value anyway. Output data needs to be scaled to within the limits of the output activation function, which are [0,1] for the logistic and [-1,1] for the hyperbolic tangent.

In practice data is scaled to speed up the training process. For efficient gradient descent it is important that the initial input to each neuron falls on the 'steep' part of the activation function, an input value around zero being required for the tanh and logistic functions. This means that the sum of all the input signals multiplied by their respective weights should be around zero. By scaling inputs to lie within the same range (maybe 0,1 or -1,1) and having small random weights (say +/- 0.1) then the chances are that this will be the case.

If the input to a neuron fall on its 'flat' portion then there is little gradient information and learning will be slow or 'stuck'. This is likely to occur if, for example, a particular raw input has a value 10,000 whereas the others lie in the range (1,10).

## Which Activation Functions are Best?

The two common sigmoidal functions, the logistic and hyperbolic tangent, are identical apart from being biased and re-scaled. As these are the transformations that the neural network performs anyway there is no difference in obtainable accuracy by using one function or the other. What has been found is that the tanh function generally requires less iterations than the logistic to attain this given accuracy, as was also found in this work. It is also found [102] that other asymmetric functions generally perform better than non-symmetric functions, an asymmetric function being one where,

$$f(-x) = -f(x)$$

which the tanh function is.

---

I suspect this is to do with the outputs of asymmetric functions potentially being in a (+/-) range (see previous section for why this will speed up training).

### What Learning Rates should be used?

This is a matter of trial and error. Generally there is no 'best' learning rate, but the rate should generally decrease as learning proceeds. There are algorithms where each weight also has its own adaptive learning rate (delta-bar-delta) but these are often more computationally expensive for little if any speed increase. Neurons in the output layer tend to have larger local gradients than those in the hidden layer, so a lower learning is often used for weights closer to the output of the network.

### Why Shuffle the Pattern Presentation Order?

If the last pattern in an epoch is a residual then the 'good work' in adjusting the weights will always be undone by this last pattern before the error is reported at the end of every epoch. By randomising the pattern presentation order this will not always happen and the learning speed will be increased. In my code the patterns are randomly selected and each pattern is not guaranteed to appear every epoch. This significantly improved the speed of the program, as the time taken to shuffle the pattern order was longer than the actual calculation of the weight changes.

### How Many Outputs should there be?

Just one.

There are many examples of neural networks with lots of outputs. Why?

The only reason appears to be because it seems to work quite well, which is not unsurprising if enough hidden neurons are used. It is not surprising that such complicated networks are treated as black boxes.

The standard algorithm minimises the rms error over all outputs, so a particular output which has a certain amount of noise or erroneous readings can affect the performance of other outputs. Why use one network with 10 outputs when 10 networks with one output each will give a much better performance.

This is a common mistake in classification tasks, where often there are several outputs each relating to a certain classification. For example, a set of inputs might be medical symptoms and four outputs could be dead, dying, very ill or not ill. It would be better to have four networks with one output each, representing dead/(not)dead, dying/(not)dying, very ill/(not)very ill and not ill/(not)not ill.

# Appendix D

## Genetic Algorithm Code

---

The following code is included as a demonstration of a genetic algorithm optimisation procedure. The subroutine CALCSUITABILITY determines the 'performance' of each candidate solution on which the tournament selection procedure is based in the subroutine BREED. In this particular example the more identical adjacent bits in the string the better the suitability. Each bit can have values of 0,1,2 or 3.

### ***The code starts here***

```
PROGRAM Genetic
```

```
!!coded in Fortran 90 by Philip Brierley
```

```
!! declare arrays !!
```

```
INTEGER,ALLOCATABLE :: PARENT(:,:),SUITABILITY(:)
```

```
INTEGER :: STRINGSIZE,POPSIZE,GENERATIONS,CONTESTANTS
```

```
COMMON STRINGSIZE,POPSIZE
```

```
!! set up some user defined parameters !!
```

```
GENERATIONS=1000
```

```
!number of generations before stopping
```

```
POPSIZE=20
```

```
!number of strings in each generation
```

```
STRINGSIZE=30
```

```
!number of bits in each string
```

```
CONTESTANTS=3
```

```
!no of contestants in tournament selection
```

```
PROB_MUT=0.01
```

```
!mutation probability
```

```
PROB_CROSS=0.8
```

```
!crossover probability
```







# Appendix E

## Water Optimisation Code

---

This code was used in chapter 5 to optimise the charging of a storage heater. The optimisation process used was based on random mutation hill climbing.

### **Code starts here**

PROGRAM WaterOptimisation

```
INTEGER      :: current_position(96),hilltop(96),descision(2)
REAL         :: pool_price(48),demand(48)
CHARACTER (LEN=20) dfname
```

```
COMMON      initial_tanktemp,tanktemp_max,coldwater_temp,rating,tanksize,hcap
COMMON      descision,current_position,demand,pool_price,hilltop,cheapest
```

```
!!! set the random number generator seed to computer internal clock !!!
CALL DATE_TIME_SEED@
```

```
!!! define some constants !!!
```

```
tanksize=50      ! hot water storage tank size (litres)
rating=2         ! electric heating element size (kW)
iflips=3        ! number of bits flipped each iteration
initial_tanktemp=5 ! the starting temperature of the water in the tank
tanktemp_max=70 ! the maximum temperature allowed by the tank thermostat
coldwater_temp=5 ! the ambient temperature of cold tap water
```

```

hcap=4.2                ! specific heat capacity of water
iterations=3000         ! number of function evaluations

!!! get the file with consumption and pool price data !!!
OPEN(UNIT=10,FILE='data.dat',STATUS='old')

!!! read in the pool price and consumption data !!!
DO I = 1,48
READ(10,*) pool_price(i),demand(i)
ENDDO
CLOSE(10)

!!! set the initial cheapest solution to a high number !!!
cheapest=10000

!!! generate an initial solution !!!
current_position(:)=0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! the process starts here !!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

DO I =1,iterations

    DO J = 1,iflips                ! repeat until the set number of bits have been flipped
    iflip_bit = (RANDOM()*96-1)+1 ! randomly select a bit to flip
    current_position(iflip_bit) = (current_position(iflip_bit)-1)**2 ! 0's to 1's, 1's to 0's
    ENDDO

    CALL CALCCOST()                ! calculate the cost of the new solution

    current_position(:)=hilltop(:) ! set the current solution to the current hilltop

ENDDO

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! this is the end of the process!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

PRINT *,cheapest

STOP
END

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE CALCCOST()

INTEGER :: descision(2),current_position(96),hilltop(96)
REAL :: demand(48),pool_price(48),needed

COMMON initial_tanktemp,tanktemp_max,coldwater_temp,rating,tanksize,hcap
COMMON descision,current_position,demand,pool_price,hilltop,cheapest

treq=tanktemp_max                ! required water temperature set to tank thermostat temperature
tanktemp=initial_tanktemp        ! initial tank condition
cost=0                            ! a running cost

```

L=1

*! marks position along the solution string***!!! repeat for 48 half hours !!!**

DO I = 1,48

**!!! descision(1) =1 then use tank water      descision(1) =0 then use cold water**  
**!!! descision(2) =1 then charge tank      descision(2) =0 then don't charge tank**

DO K =1,2      *! read in two descisions for this half hour*  
 descision(K)=current\_position(L)      *! read in descision from solution string*  
 L=L+1      *! advance the marker one step along the solution string*  
 ENDDO

needed = demand(I)      *! required hot water this half hour*  
 price = pool\_price(I)      *! pool price this half hour*

**!!! decision(1) which water source?**

IF (needed > 0) THEN      *! if there is demand then:*  
     IF (descision(1)==0) THEN      *! use direct*  
         cost=cost+(((treq-coldwater\_temp)\*hcap\*needed)/3600)\*price  
     ELSE      *! use tank water with direct to ensure treq maintained*  
         IF (needed<tanksize) THEN  
             cost=cost+(((treq-tanktemp)\*hcap\*needed)/3600)\*price  
             tanktemp=(coldwater\_temp\*(needed/tanksize))+(tanktemp\*(tanksize-needed)/tanksize)  
         ELSE  
             extra1=(((treq-tanktemp)\*hcap\*tanksize)/3600)\*price  
             extra2=((treq-coldwater\_temp)\*hcap\*(needed-tanksize)/3600)\*price  
             cost=cost+extra1+extra2  
             tanktemp=coldwater\_temp  
         ENDIF  
     ENDIF  
 ENDIF

ENDIF

**!!! this bit is the constraint which adjusts the solution so that a descision to charge**  
**!!! the tank cannot occur if the tank temperature is already at a maximum**

IF (descision(2)==1 .and. tanktemp==treq) THEN  
 current\_position(L-1)=0  
 descision(2)=0  
 ENDIF

**!!! descision(2) charge the tank**

IF (descision(2)==1) then  
 amaxinput=(treq-tanktemp)\*tanksize\*hcap      *! maximum possible energy input (kJ)*  
 pmax=(amaxinput/3600)\*price      *! maximum possible cost*  
 actinput=rating\*60\*30      *! possible energy input*  
 pact=(actinput/3600)\*price      *! possible cost*

IF (pact>pmax) then      *! heat up water to limiting temperature*  
 pact=pmax  
 tanktemp=treq  
 ELSE      *! or calculate the new tank temperature*  
 tanktemp=tanktemp+(actinput/(hcap\*tanksize))  
 ENDIF

cost=cost+pact      *! add the charging cost to the total*  
 ENDIF

ENDDO **!!! 48 half hours completed**

```

IF ( cost<=cheapest) THEN
cheapest=cost
hilltop(:)=current_position(:)
ENDIF

```

END SUBROUTINE CALCCOST

!!

**The code finishes here**

**!!!!!! contents of data file 'data.dat' !!!!**

<i>pool price (pence/kWhr)</i>			<i>demand (litres)</i>			<i>hour (reference only)</i>		
↓	↓	↓						
0.96	0	0.5	4.26	0	8.5	2.85	0	16.5
0.96	0	1	4.28	10	9	3.19	8	17
1	0	1.5	4.1	10	9.5	3.4	7	17.5
1.89	0	2	4.11	7	10	3.43	7	18
1.89	0	2.5	4.1	0	10.5	4.62	7	18.5
1.89	0	3	4.09	0	11	5.07	6	19
1.89	0	3.5	4.09	0	11.5	5.04	24	19.5
1	0	4	3.24	8	12	4.52	18	20
0.96	0	4.5	3.23	7	12.5	3.38	7	20.5
0.96	0	5	3.21	35	13	2.36	6	21
0.99	0	5.5	3.19	5	13.5	2.17	5	21.5
1	80	6	2.85	16	14	2.1	6	22
1	3	6.5	2.84	6	14.5	1.85	5	22.5
1.85	0	7	3.18	0	15	1.85	5	23
3.05	24	7.5	2.84	7	15.5	1.85	6	23.5
4.07	4	8	2.84	0	16	1	9	24

# Appendix F

## Storage Heater Thermal Model Code

---

The thermal model of a room and heaters was created by using an explicit finite difference method [103]. This is a nodal approach that calculates temperatures at nodes within walls at discrete time intervals (every 5 seconds in this case), with convective heat transfer taking place at the wall surfaces. Each wall can be given different thermal properties and exterior air temperatures, which are read in from a data file. Wind speed as well as temperature effects the external wall heat transfer and is read in from a weather file. In this model solar gains are ignored.

There are two heaters in this model, a direct acting heater and a storage fan heater. The direct acting heater inputs are simply an energy input into the room air. The representation of the storage fan heater is as shown in the diagram.



```

Shrating=3000      !element rating (Watts)
Rlength=4         !!room length
Rwidth=5          !!room width
Rheight=2.5       !! room height
XternalwallPC=50  !!percent of wall area that is external (including windows)
WindowPC=10       !! percent of external wall area that is glazed
Wnodes=5          !!wall nodes
Walls=3           !!wall types (external, internal, ceiling)
Cnodes=10         !! nodes in storage heater core
Inodes=10         !!nodes in storage heater insulation
Sharea=0.6        !!storage heater surface area
Shgap=0.1         !!air gap thickness between storage heater core and insulation
ACPMfanon=10     !!air changes per minute in storage heater air gap with fan on
ACPMfanoff=1      !!air changes per minute in storage heater air gap with fan off

```

```
!! allocate arrays
```

```
!!! vectors for node temperatures in Walls
```

```
ALLOCATE(Wtnow(Walls,Wnodes))    !! current node temperatures
```

```
ALLOCATE(Wtnext(Walls,Wnodes))   !! next node temperatures
```

```
!!! dummy vectors for offset Wnodes for matrix multiplication
```

```
ALLOCATE(Wp1(1:Wnodes-1))       !! node + 1
```

```
ALLOCATE(Wm1(2:Wnodes))         !! node - 1
```

```
!!! array containing wall exterior temperatures
```

```
ALLOCATE(Tout(Walls))
```

```
!!! other properties
```

```
ALLOCATE(Wthick(Walls))          !wall thicknesses
```

```
ALLOCATE(Wdx(Walls))             !distance between wall nodes
```

```
ALLOCATE(Wk(Walls))              !wall thermal conductivities
```

```
ALLOCATE(Wdens(Walls))           !wall densities
```

```
ALLOCATE(Warea(Walls))           !wall areas
```

```
ALLOCATE(Wcp(Walls))             !wall heat capacities
```

```
ALLOCATE(Walpha(Walls))          !wall thermal diffusivities
```

```
ALLOCATE(Wfo(Walls))             !wall Fourier number
```

```
ALLOCATE(Hin(Walls))             !convection coefficient
```

```
ALLOCATE(Hout(Walls))            !convection coefficient
```

```
ALLOCATE(BIin(Walls))            !Biot number of inner wall surface
```

```
ALLOCATE(BIout(Walls))           !Biot number of outer wall surface
```

```
ALLOCATE(Q(Walls,2))             ! heat contribution from each wall
```

```
!!! storage heater arrays
```

```
ALLOCATE(Ctnow(Cnodes))          !current core node temperatures
```

```
ALLOCATE(Ctnext(Cnodes))         !next core node temperatures
```

```
ALLOCATE(Itnow(Inodes))          !current insulation node temperatures
```

```
ALLOCATE(Itnext(Inodes))         !next insulation node temperatures
```

```
!!! storage heater dummy vectors
```

```
ALLOCATE(Cp1(1:Cnodes-1))        !core next node
```

```
ALLOCATE(Cm1(2:Cnodes))          !core previous node
```

```
ALLOCATE(Ip1(1:Inodes-1))        !insolation next node
```

```
ALLOCATE(Im1(2:Inodes))          !insolation previous node
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!! properties of each wall !!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
Warea_tot=2*Rheight*(Rlength+Rwidth)  !! total wall area
```

```
!!! wall 1 , external wall !!!
```



Wthick(1)=0.23                    *!wall thickness (m)*  
 Wk(1)=0.7                         *!thermal conductivity ( W/m deg C.)*  
 Wdens(1)=1700                   *!density (kg/m^3)*  
 Wcp(1)=800                       *!specific heat capacity (J/kg deg C.)*  
 Warea(1)=Warea\_tot\*(XternalwallPC/100)\*((100-WindowPC)/100)   *!surface area*

**!!! wall 1, internal walls !!!**

Wthick(2)=0.12                   *!wall thickness (m)*  
 Wk(2)=0.7                         *!thermal conductivity ( W/m deg C.)*  
 Wdens(2)=1700                   *!density (kg/m^3)*  
 Wcp(2)=800                       *!specific heat capacity (J/kg deg C.)*  
 Warea(2)=Warea\_tot\*((100-XternalwallPC)/100)   *!surface area*

**!!! wall 3, ceiling !!!**

Wthick(3)=0.2                    *!wall thickness (m)*  
 Wk(3)=0.04                       *!thermal conductivity ( W/m deg C.)*  
 Wdens(3)=12                      *!density (kg/m^3)*  
 Wcp(3)=840                       *!specific heat capacity (J/kg deg C.)*  
 Warea(3)=Rlength\*Rwidth       *!surface area*

**!!! window properties !!!**

WINuval=5.4                       *!W/m^2 Deg.C (4mm thick window)*  
 WINarea=Warea\_tot\*(XternalwallPC/100)\*((WindowPC)/100)   *!window area*

**!!! general properties !!!!**

Wdx=Wthick/(Wnodes-1)         *!distance between nodes*  
 Walpha=Wk/(Wdens\*WCp)         *!thermal diffusivity*  
 Wfo=Walpha\*dTIME/(Wdx\*\*2)     *!Fourier number*

**!!! AIR PROPERTIES !!!**

Rvol=Rlength\*Rwidth\*Rheight   *!volume of room*  
 Rmassair=1.2\*Rvol               *!air mass*  
 AirCp=1004                       *!air specific heat capacity*

!!

**!!! STORAGE HEATER PROPERTIES !!!**

!!

**!!! CORE !!!**

Ccp=942                            *!specific heat capacity*  
 Cthick=0.07                       *!thickness (m)*  
 Cdx=Cthick/(Cnodes-1)         *!distance between nodes*  
 Ck=2                                *!thermal conductivity*  
 Cdens=4000                       *!density*  
 Calpha=Ck/(Cdens\*CCp)         *!thermal diffusivity*  
 Cfo=Calpha\*dTIME/(Cdx\*\*2)     *!Fourier number*

**!!! INSULATION !!!**

Icp=1000                           *!specific heat capacity*  
 Ithick=0.15                       *!thickness*  
 Idx=Ithick/(Inodes-1)         *!distance between nodes*  
 Ik=0.1                              *!thermal conductivity*  
 Idens=200                         *!density*  
 Ialpha=Ik/(Idens\*ICp)         *!thermal diffusivity*  
 Ifo=Ialpha\*dTIME/(Idx\*\*2)     *!Fourier number*

**!!! AIRGAP !!!**

SHvol=SHarea\*SHgap           *!volume of air in air gap*  
 SHmassair=1.2\*SHvol         *!mass of air in air gap*

!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
 !!!! set initial conditions !!!!  
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!! Walls !!!

Wtnow(:,:)=290           !all Walls initially 17 deg C.  
 Troom=293               !initial room air temperature 20 deg C.

!!!storage heater !!!

Ctnow(:,:)=290           !core initially 17 deg C.  
 Itnow(:,:)=290           !insulation initially 17 deg C.  
 Tairgap=293              !initial storage heater air gap temperature

!!! set outside temps and windspeeds !!!

OPEN(UNIT=10,FILE='WEATH85.pat',STATUS='OLD')  
 READ(10,\*) CHKFILE                   !column headings  
 READ(10,\*) Tout(1),Windspeed,hr\_of\_day   !read current outside temp and windspeed  
 READ(10,\*) Tout1HR,Windspeed1HR       !read next hours outside temp and windspeed

dTout1=(Tout1HR-Tout(1))\*dTIME/3600           !temperature increment  
 Tout(1)=Tout(1)-dTout1                   !rewind temperature 1 step  
 dWindspeed=(Windspeed1HR-Windspeed)\*dTIME/3600   !windspeed increment  
 Windspeed=Windspeed-dWindspeed           !rewind windspeed 1 step

Tout(2)=293    !wall 2 exterior temp set constant  
 TOUT(3)=273   !0 deg in space above ceiling

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!! check for stability !!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

PRINT\*, 'Stability assessment'

DO I=1,WALLS

IF (Wfo(I)>0.25) THEN   !actually 0.5 for 1 dimensional case but safety factor introduced

PRINT \*, 'Unstable, reduce time step'

STOP

ENDIF

ENDDO

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!! write result file !!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

PRINT \*, 'write results file? (1=yes 0=no)'

READ \*, WTF

IF (WTF==1) THEN

2 PRINT \*, 'filename?'

READ \*, Fname

CALL UPCASE@(Fname)

CALL APPEND\_STRING@(FNAME, '.RES')

!file called 'fname.res'

OPEN(UNIT=20,FILE=FNAME,STATUS='NEW',ERR=2)

ENDIF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!! Graphics mode, plots profiles to screen !!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

PRINT \*, 'plot results to screen? (1=yes 0=no)'

READ \*, Graphics

IF (Graphics==1) THEN

head1='21 deg C'

```

head2='0 deg C'
call vga@
ICOL=2           !room temperature line colour (changes daily)
ICOLOUT1=15     !outside temperature white
ICOLOUT2=14     !room temperature yellow
ELSE
  call text_mode@
ENDIF

Steps=Days*24*60*60/dTIME
HrsDone=0
DaysDone=0
M=0             !hour counter
N=0             !day counter
TermHeight=1/ (((670.656*(Rheight**6.0)) + (120.43*(Rheight**8.7))) ** (1.0/6.0))
!wall characteristic height

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! THIS IS THE START!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

DO I=1,Steps           !in steps given by dTIME

  Tout(1)=Tout(1)+dTout1           !increment outside temperature
  Windspeed=Windspeed+dWindspeed   !increment windspeed

  !!!!!!!!!!!calculate surface heat transfer coefficients !!!!
  !! convection from ceiling internal surface (wall 3)
  Hin(3)=0.6* ((abs(Wtnow(3,Wnodes)-Troom))/(Rheight**2))**0.2

  !! convection from ceiling external surface
  Hout(3)=1.63*(abs(Wtnow(3,Wnodes)-Troom))**(0.333)

  !! convection from internal surface of room walls (wall 1 and wall 2) !!
  Hin(1:2)=TermHeight+(1.23*((abs(Wtnow(1:2,Wnodes)-Troom))**(0.333)))

  !! convection from external surface of interior walls
  Hout(2)=TermHeight+(1.23*((abs(Wtnow(2,1)-Tout(2))) **(0.333)))

  !! convection from external surface of exterior wall
  Hout(1)=5.8+(4.1*Windspeed)

  BIout=hout*Wdx/Wk
  BLin=hin*Wdx/Wk

  !!! assumed values for core and insulation !!!
  Hcore=2           !core heat transfer coefficient (only 1 surface exposed)
  Hinsin=2          !insulation inner
  Hinsout=2         !insulation outer
  BIcone=Hcore*Cdx/Ck
  BIoutins=Hinsout*Idx/Ik
  BLinins=Hinsin*Idx/Ik

  !!!!!!!!!!!!!!!
  !!! Walls !!
  !!!!!!!!!!!!!!!

DO j=1,Walls        !perform for each wall

```

```

Wp1=Wtnow(j,2:Wnodes)      !dummy matrix
Wm1=Wtnow(j,1:Wnodes-1)    !dummy matrix

!!! internal nodes !!!
Wtnext(j,2:Wnodes-1) = Wfo(j)*(Wp1(2:Wnodes-1)+Wm1(2:Wnodes-1)) + ((1-(2*Wfo(j))) * Wtnow(j,2:Wnodes-1))

!!! node on outside of wall !!!
Wtnext(j,1)=(2*Wfo(j)*( Wp1(1)+(Blout(j)*Tout(j)))) + ((1-(2*Wfo(j))-(2*Blout(j)*Wfo(j)))*Wtnow(j,1))

!!! node on inside of wall !!!
Wtnext(j,Wnodes)=(2*Wfo(j)*( Wm1(Wnodes)+(Blin(j)*Troom))) + ((1-(2*Wfo(j))-(2*Blin(j)*Wfo(j)))*Wtnow(j,Wnodes) )

ENDDO !(j=1,walls)

!!! work out heat input to room from each wall !!!

!!! calculate heat inputs over time period for present and next      !!!
!!! inner wall temperatures and average                               !!!
Q(:,1)=Hin(:)*Warea(:)*(Wtnow(:,Wnodes)-Troom)*dTIME
Q(:,2)=Hin(:)*Warea(:)*(Wtnext(:,Wnodes)-Troom)*dTIME
Qtot_Walls=SUM( (Q(:,1)+Q(:,2))/2 )

Wtnow=Wtnext      !wall node temperatures advance

!!! window !!!
Qwin=WINuval*WINarea*(Tout(1)-Troom)*dTIME

!!! energy changes due to ventilation with outside air !!!
MassXchange=Racph*Rmassair*dTIME/3600
Qvent=AIRcp*MassXchange*(Tout(1)-Troom)

!!! direct acting heater input on timer and thermostat switch !!
!!! can use this to switch direct acting heater on at various times !!!
IF (Troom<(20+273) .AND. hr_of_day >8 .AND. hr_of_day<12) THEN
Qdah=dah_rating*dTIME
ELSE
Qdah=0
ENDIF

!!! storage heater !!!
IF (Daysdone<90 .or. Daysdone >270) then !use in winter
IF (hr_of_day < 7 .and. Ctnow(1)<(700+273)) then !E7 charging and 700 deg C. max temp
SHpower=SHrating
ELSE
SHpower=0
ENDIF
ELSE
SHpower=0
ENDIF

FLUX=SHpower/SHarea

!!!core!!!
Cp1=Ctnow(2:Cnodes)      !dummy matrix
Cm1=Ctnow(1:Cnodes-1)    !dummy matrix

!!! internal core nodes !!!
Ctnext(2:Cnodes-1)=Cfo*( Cp1(2:Cnodes-1)+Cm1(2:Cnodes-1)) + ((1-(2*Cfo))*Ctnow(2:Cnodes-1))

!!!core node next to heating element !!!

```

Ctnext(1)=2\*Cfo\*(FLUX\*(Cdx/Ck)+ Ctnow(2)) + (Ctnow(1)\*(1-(2\*Cfo)))

**!!!core node next to air gap !!!**

Ctnext(Cnodes)=(2\*Cfo\*( Ctnow(Cnodes-1) + (BICORE\*Tairgap))) + ((1-(2\*Cfo) -(2\*BICORE\*Cfo)) \* Ctnow(Cnodes) )

**!!! insulation !!!**

Ip1=Itnow(2:Inodes)     !*dummy matrix*

Im1=Itnow(1:Inodes-1)   !*dummy matrix*

**!!!internal insulation nodes!!!**

Itnext(2:Inodes-1)=Ifo\*( Ip1(2:Inodes-1)+Im1(2:Inodes-1) )+( (1-(2\*Ifo))\*Itnow(2:Inodes-1) )

**!!!insulation node on air gap side!!!**

Itnext(1)=(2\*Ifo\*( Ip1(1)+(BIoutINS\*Tairgap))) + ( (1-(2\*Ifo)-(2\*BIoutINS\*Ifo)) \* Itnow(1) )

**!!! insulation node on room side !!!**

Itnext(Inodes)=(2\*Ifo\*( Im1(Inodes)+(BIinINS\*Troom))) + ( (1-(2\*Ifo)-&& &&(2\*BIinINS\*Ifo))\*Itnow(Inodes) )

**!!!work out heat input to air gap from core and insulation. calculate heat inputs over !!!**

**!!!time period for present and next temperatures and take an average !!!**

Qcore1= HCORE\*SHarea\*(Ctnow(Cnodes)-Tairgap)\*dTIME

Qcore2= HCORE\*SHarea\*(Ctnext(Cnodes)-Tairgap)\*dTIME

Qcore=(Qcore1+Qcore2)/2

Qins1=HinSin\*SHarea\*(Itnow(1)-Tairgap)\*dTIME

Qins2=HinSin\*SHarea\*(Itnext(1)-Tairgap)\*dTIME

Qins=(Qins1+Qins2)/2

Qshgap=Qins+Qcore

**!!!!new air gap temperature!!!**

Tairgap=Tairgap+((Qshgap/AIRcp)/SHmassair)

**!!! sh contribution to room via convection from casing**

Qshouter1=HinSout\*SHarea\*(Itnow(Cnodes)-Troom)\*dTIME

Qshouter2=HinSout\*SHarea\*(Itnext(Cnodes)-Troom)\*dTIME

Qshouter=((Qshouter1+Qshouter2)/2)\*2

**!!only half heater modelled**

Itnow=Itnext

*!advance insolation node temperatures*

Ctnow=Ctnext

*!advance core node temperatures*

**!!! energy change due to exchange with storage heater air !!!**

IF (hr\_of\_day > 17 .and. Troom < (273+20) ) then

ACPM=ACPMfanon

ELSE

ACPM=ACPMfanoff

ENDIF

shMassXchange=ACPM\*SHmassair\*dTIME/60

Qventsh=AirCp\*shMassXchange\*(Tairgap-Troom)\*2

Tairgap=Tairgap-((Qventsh/AIRcp)/SHmassair)

**!!! calculate new room temperature !!!**

Qtot=Qwin+Qtot\_Walls+Qvent+Qdah+Qshouter+Qventsh

**!total heat input**

Troom=Troom+((qtot/AIRcp)/Rmassair)

**!new room temperature**

M=M+1

*!counts time until an hour is completed*

IF (M\*dTIME == 60\*60) THEN   !*do the following every hour*

```

IF (Graphics==1) THEN
  IF (HrsDone<200) THEN
    X=HrsDone
  ELSE
    CALL GET_SCREEN_BLOCK@(0,0,200,479,buffer)
    CALL CLEAR_SCREEN_AREA@(200,0,200,450,0)
    CALL RESTORE_SCREEN_BLOCK@(-1,0,BUFFER,0,ERROR_CODE)
    CALL RETURN_STORAGE@(BUFFER)
    X=200
  ENDIF

  CALL PLOTDOT(x,Troom,ICOL)
  CALL PLOTDOT(x,Tout(1),ICOLOUT1)
  twenty1=294
  zero=273
  CALL PLOTDOT(x,twenty1,ICOLOUT1)
  CALL PLOTDOT(x,zero,ICOLOUT2)
  CALL DRAW_TEXT@(head1,240,79,3)
  CALL DRAW_TEXT@(head2,240,290,3)
ENDIF

21 format(i4,1x,f4.1,f6.1,1x,f6.1)
IF (WTF==1) write(20,21) DaysDone,hr_of_day,Tout(1)-273,Troom-273

READ(10,*) Tout1HR,Windspeed1HR,hr1HR
dTout1=(Tout1HR-Tout(1))*dTIME/3600
dWindspeed=(Windspeed1HR-Windspeed)*dTIME/3600

IF (hr1hr==0) then
  hr_of_day=23
ELSE
  hr_of_day=hr1HR-1
ENDIF

HrsDone=HrsDone+1
M=0 !reset hour counter after every hour

ENDIF

N=N+1 !day counter
IF (N*dTIME == 60*60*24) THEN
  IF (ICOL == 8) THEN
    ICOL=2
  ELSE
    ICOL=ICOL+1
  ENDIF

  DaysDone=DaysDone+1
  N=0

  IF (Graphics ==0) print *,DaysDone
ENDIF

ENDDO !(I=1,steps)

!!!this is the end

CONTAINS

```

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE PLOTDOT(X,Y,ICOL)
INTEGER X,ICOL,yyy
REAL Y,YY

YY=(((y-273)*10)*(-1))+300)
yyy=NINT(YY)
CALL SET_PIXEL@(X,yyy,ICOL)
END SUBROUTINE PLOTDOT
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE PLOTLINE(X)
INTEGER X
CALL DRAW_LINE@(X,0,X,400,3)
END SUBROUTINE PLOTLINE
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

```

END PROGRAM Building\_Model

**The code finishes here**

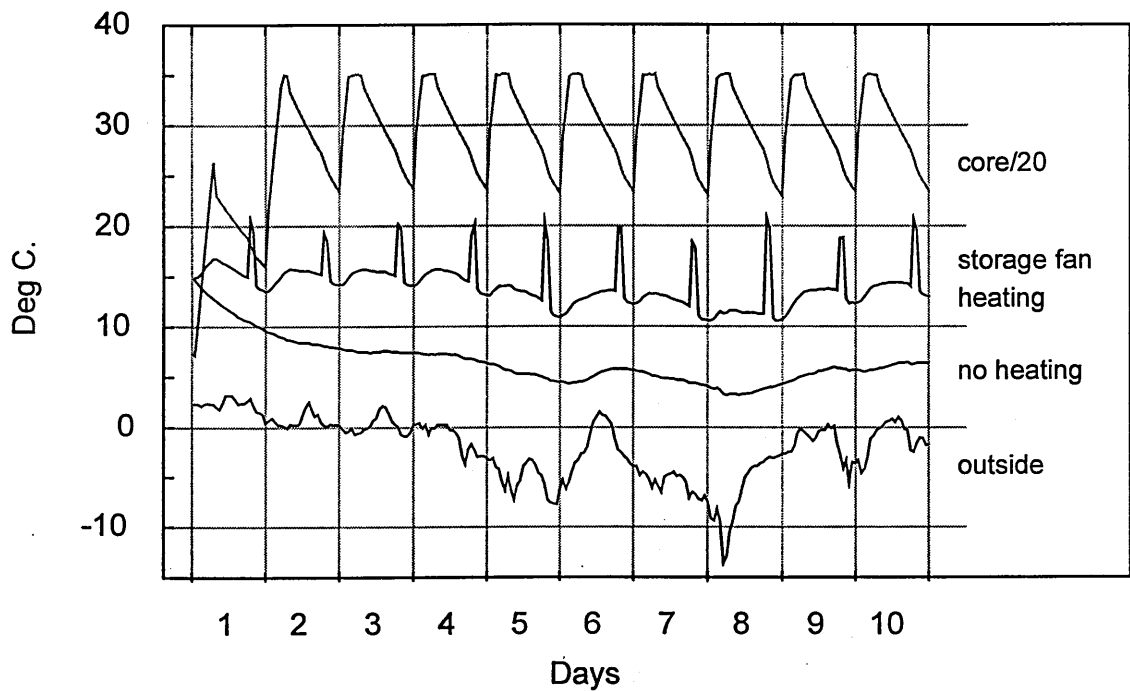


Fig F-1 An example of the output from the code showing room temperature with and without a storage-fan heater on an E7 charging schedule

