Short–term electric load forecasting using artificial neural networks

by

Eric Lee Daugherty

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Mechanical Engineering
Major: Nuclear Engineering

Iowa State University
Ames, Iowa

1994

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would first like to thank John Lamont and EPRC for providing funding for my research. I would next like to thank Dr. E.B. Bartlett, my major professor, for his support and encouragement. I would also like to thank the members of my committee, Dr. D.B. Bullen and G.B. Sheble, for their time and consideration.

Eric thanks Elizabeth, my one and only and everything; Mom, Dad, Todd, Tracy, Colleen, Chris, and Sydney, my family, for 25–years and counting of support; Rose Pence and Becky Staedtler, more than program secretaries, but friends and great sources of knowledge and help; Dr. Danofsky, Dr. Wechsler and Dr. Rohach, my Iowa State Nuclear Engineering professors; Kory Sylvester the krusty, Thomas Steven the ancient, Dave Baldes the unawareness man and Chris Raymo enuff said, college friends for life; Anujit Basu, Challapathy Dhanwada, Keehoon Kim, Kyung Jin Jang, Scott Wendt, Matt Reid, Tony Hammitt and Amy Jo Patterson, my graduate school pals, ACL buddies, and xtank partners; and Ty Tabor, Doug Pinnick, and Jerry Gaskill, intangibles.

CHAPTER 1. INTRODUCTION

Short–Term Electric Load Forecasting

Short–term electric load forecasting (STELF) plays an important role in managing an electric utility. The forecast can be used to help with operation decisions such as unit commitment, load dispatch, and reserve capacity allocation. The forecast involves predicting the minutes–to–minutes loads for a few minutes ahead to predicting the hourly loads up to a week ahead. With accurate STELF, these operation decisions can be optimized so that economic savings and secure operating schedules can be realized by the utility.

Many techniques have been developed to perform STELF. They can roughly be classified into statistically based techniques and expert system approaches. Statistically based techniques include stochactic time series approaches and multiple linear regression approaches. Time series methods use historical data to extrapolate future values and are well suited for stationary, seasonal time series. Electric load curves typically display this type of behavior so time series methods are widely used and are the most popular methods for STELF [32]. But, electric load curves also possess elements that a time series approaches would have difficulties with. Examples include weekends, holidays, and unseasonably hot or cold days that cause load curves to vary from their normal seasonally. In addition, time series methods do not fully utilize explanatory variables, such as weather, which are important for load forecasting. Multiple linear regression, on the other hand, does use explanatory variables such as weather and societal factors, but creating reasonable

models that contains the dynamics of these external variable relations can be difficult. In addition, the relationships found between the explanatory variables are linear and the assumption of linearity may not be reasonable. An expert system also uses relations between external variables, but in the form of knowledge from an expert. Translating the knowledge of an expert into a set of rules and logic steps is often difficult or impractical.

Artificial Neural Networks

In attempts to gain accuracy and to decrease computational time, artificial neural networks (ANNs) have recently gained popularity for STELF because they demonstrate many favorable characteristics for this type of problem. ANNs can be used for system modeling and prediction [25], can easily incorporate historical and regression data into one model [12], and can infer nonlinear relations between variables [30]. ANNs also save computational time because once trained, they are quite fast [7].

ANNs can do these types of tasks because they generalize [7]. Generalization is the ability to infer a reasonable relation from something never encountered before. ANNs can be used for STELF by utilizing past electric load curves, past weather information, and societal information to infer some relation so that forecasts of electric load curves can be performed. The relationships between electric load curves and weather variables are known to exist but the exact relation between the two is not known. In addition, the relation between the load curve and weather variables is most likely non–linear so linear methods for relating the two will not be completely reasonable. Societal factors are also known to exists is some form. The relation between all of these variables, whatever it may be, can be determined by the ANN. Thus, by utilizing an ANN for STELF, input variables can easily be used in the modeling process by

eliminating having to know the complex relation between the input variables and the forecasted load. In addition to striving for faster and more accurate models, ANNs allow for an easier modeling process.

Problem Statement

This work explores the possibility of implementing an ANN methodology for STELF. Areas that will be examined are accuracy, computational time, and ease of modeling. In addition, the ANN STELF models will be used to perform predictions for an entire year. The forecasting period of interest in STELF is up to a week ahead so the ANN methodology was applied to perform hourly and daily predictions for a week in advance. The methodology involved two different aspects. The first dealt with unit commitment and reserve capacity allocations aspects. These areas were dealt with by designing ANNs that could perform forecast daily so that weekly forecast could be obtained and updated on a daily basis. The second methodology involved designing ANNs that could be updated for the smallest unit of time being considered, which happened to be hours, to address load dispatch concerns. Creating these ANN models involved obtaining hourly load and weather data and involved incorporating societal data. The modeling process also involved a determination of what variables to include in the models. The data used covered three years from 1990 to 1992 where 1992 was the year used to perform predictions.

CHAPTER 2. SHORT–TERM ELECTRIC LOAD FORECASTING

Motivation

SInce the first operations of power systems, electric utilities have had to increase efficiency while maintaining or even increasing system reliability. The deregulation of the electric industry in the late 1990's will push efficiency even more. There are many areas for increasing efficiency within a power system including fuel budgeting, transmission planning studies, system planning studies, system operating policy, and load management studies. The work in this thesis will cover areas that benefit from short–term electric load forecasting (STELF). These areas include unit commitment, reserve capacity allocation and load dispatch.

Unit Commitment

A power system can have vast hourly and daily load variations due to the differences between high mid–day, peak–hour demands and low early morning, valley –hour demands. Regardless of this fact, a utility must meet the electric demand in an economical manner while considering the operating and shutdown constraints of the generating units. The load variation can be meet by operating all generating units at proportional rates to meet the current demand. This strategy will cause plants to operate near maximum capacity during peak hours and near minimum capacity during valley hours. An alternative to operating all generating units simultaneously is to operate the fewest number plants near maximum capacity to meet the electric demand. This later alternative tends to more economical because the fewest possible number of generating units

are on–line. This reduces staffing and operating costs for operating several plants. In addition, the generating units on–line are operating near maximum capacity where power generation is more economical than generating units operating near minimum capacity. Since a power system contains many generating units of different cost characteristics, the generating units can be ranked in order of economics. Thus, when the decision comes to operate the fewest number of generating units, the plants with the highest ranking will operate first.

To operate a power system with the fewest possible number of generating units on–line requires planning the start–up and shut–down schedules of the plants. Start–up times can vary from a few minuets to a few days so obtaining reasonable forecasts are important. Planning the start–up and shut–down schedules also requires accounting for the mandatory up and down times that the generating units may have, which mandates the minimum number of continuos hours that a plant can be operating or dormant. In addition, start–up schedules should be planned in accordance with the associated start–up costs. Start–up costs can increase with longer off–line periods. This can be attributed to the fact that operating temperatures and pressures will continue to decline with longer off–line periods having greater energy requirements to bring the plant back on–line.

Reserve Capacity Allocation

In addition to the economic considerations of unit commitment, system reliability is also an important concern. System reliability ensures that the load demand will be meet during reasonable load fluctuations outside unit commitment forecasts and during possible unplanned events such as forced outages. System reliability is established by having generating reserve capacity. Generating reserve capacity is classified as spinning reserve or quick–start reserve. Spinning reserve is the additional capacity that

a generating unit has by operating at less that full output. A 500–MW plant operating at 400–MW has 100–MW of spinning reserve. Quick–start reserve refers to generating units that can start up and deliver power quickly. Gas–turbines and pumped–storage hydro units are examples of quick–start units.

Planning reserve capacity allocation involves utilizing both spinning reserve and quick–start reserve. Even though it is most economical to operate generating units at maximum capacity, quick–start reserve is less reliable since the plant may fail start or to operate. Generating units operating at 90% provide 10% of spinning reserve still and are operating in an economical manner. Spinning reserve is also distributed among the operating generating units in a power system. This reduces the risk of a single generating unit failure causing the system reliability to fall to an unacceptable level. The distribution of spinning reserve also allows the generating units to operate near maximum capacity.

System reliability also requires that the operation of generating units be distributed to provide area protection. This provides system reliability because the smaller areas in a service region would have a degree of self reliability. There are also the transmission cost of moving electricity over greater distances. A unit commitment/reserve capacity allocation algorithm is presented here [42]. The four–step approach is:

1. Units available for operation are ranked in a priority list according to full–load operating cost in $/MWh.

2. The area protection rule is applied to reorder the list generated in step 1.

3. By using the priority list generated in steps 1 and 2, a minimum number of plants are committed each hour of the week to meet both (1) the load, using the continuous operating ratings of the units, and (2) the load plus spinning reserve requirement, using maximum ratings of the units.

4. Step 3 is reviewed for unit minimum downtime violations. If a violation exist, the unit is required to operate during that period.

Load Dispatch

By performing unit commitment and reserve capacity allocation scheduling, a power system can establish which generating units will be on–line for each hour. Load dispatch planning, on the other hand, determines the portion that each on–line generating unit will deliver to meet the immediate electric load. Load dispatch strives to meet the minute–by–minute system load demands in an economical and reliable manner. Transmission losses factor in determining the amount of power that the on–line generating units will deliver. Longer transmission distances between generating units and demand areas result in greater losses. But, plants with low operating costs can sometimes offset transmission losses. Generating units also have individual characteristics, such as fuel costs and incremental power cost which make some plants more economical to operate during load dispatch operations.

A load dispatch equation as derived by Stoll [42] will be presented in a qualitative manner. He first begins by considering the constraint that system load must be met by power output from the committed units minus the transmission losses. The total system operating cost per hour is the summation over all committed plants of the product of the fuel cost times the fuel input, plus variable incremental operation and maintenance cost times the megawatt output. This cost is to be minimized so a first derivative of the system cost with respect to each unit's power output must be zero. This yields the dispatch equation which implies that the maximum system economy results when the incremental generating cost plus the incremental cost of transmission losses are equal.

Another concern of load dispatch is the environment. These environmental constraints basically deal with the limitations on the amount of pollutants that can be discharged from a generating unit and with the overall air quality in a given area based on the pollution. The first constraint can be dealt with by using appropriate fuel or with

pollution limiting techniques. The second constraint is more of a function of the concentration of plants in a given area. As a plant operates near maximum capacity, the pollution also is produced near maximum capacity. Environmental concerns could limit load dispatch for a concentrated area.

Methods

Accurate STELF can produce optimal unit commitment, reserve capacity allocation, and load dispatch. This, in turn, can benefit an electric utility by the operation of an efficient and reliable system. Many of the methods utilized involve the use of automated systems. For this reason, Bunn and Farmer have outlined four general characteristics that a STELF model should posses [6]. They are;

(1) adaptiveness

(2) recursivness

(3) robustness

(4) economic computer utilization

Adaptiveness allows the parameters in a model to change over time. This may be necessary because the dynamics of some parameters may not be stationary. The greater the number of parameters that change yield a more adaptive model. Recursivness enables a STELF model to process novel data in a reasonable manner. That is, the model should not have to be recomputed. Robustness permits a STELF model to perform regardless of missing or noisy data. Incomplete and noisy data is often common in STELF. WIth any use of computational resources, economic utilization is always important. These four characteristics provide good guidelines for developing STELF models but it is often not possible or difficult to incorporate all of them. Electric utilities differ as do their needs, so their STELF methods will utilize methods with characteristics that are

important to them.

In attempts to create STELF models that incorporate these four characteristics, many different methods have been implemented to accomplish this task. Moghram and Rahman have identified the five most common methods [30]. They are either statistical approaches or an expert system approach. The three most popular are listed below [3,14,34];

(1) Multiple Linear Regression

(2) Stochastic Time Series

(3) Knowledge–Based Approach

All of these methods have their merits but they also have different characteristics. For example, time series approaches are well suited for seasonal data while knowledge–based approaches can deal with irregularities. Any one of these methods can provide reasonable forecasts and with significant development can possibly provide very accurate results.

Multiple Linear Regression

Many processes are functions of other variables, STELF being just one. These variables are usually referred to as explanatory or regression variables. Determining the exact relation of these variables can be often difficult because their relation is often non–linear or just not very well understood. In some instances, an assumption of linearity may be reasonable. Modeling with multiple linear regression involves using explanatory variables to help describe a process. For STELF, the electric loads could be described in terms of weather and societal variables. A typical equation will have the form [33] :

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \varepsilon_i \qquad (2.1)$$

Where $i = 1, 2, \ldots, n$, y_i is the electric load, $x_{i1}, \ldots, x_{ip}$ are explanatory variables correlated

with y_i , ε_i is a random variable, and $\beta_0, \ldots, \beta_p$ are regression coefficients. The explanatory variables are sought by making an educated guess as to what might be important. They are then tested individually with the electric load by using a some sort of linear correlation test. This test is used to determine which of the explanatory variables chosen are significant. Once the statistically significant explanatory variables are found, their coefficients can be determined by using an estimation method such as a least squares estimation technique.

Stochastic Time Series

Often, many processes can be described and modeled as time series. A time series is a collection of observations made sequentially in time. Electric load curves are time series. Time series methods involves representing some series as an output from a linear filter where the input is a random series [37]. The time series can have several classifications depending on the filter characteristics. If the filter uses previous values of the electric load series as well as a random noise, it is said to be an autoregressive (AR) process or filter. If the filter uses the previous values of the random series as well as the electric load, it is said to be a moving–average (MA) process or filter. The filter can combine both processes to yield an autoregressive moving–average (ARMA) process. Another combined filter case involves modeling a non stationary series where a differencing process is involved. This is referred to as a autoregressive integrated moving–average (ARIMA) process or filter.

For STELF, the modeling process involves a data series that contains periodicities. These periodicities cause a slight modification to the ARIMA process . This process involves using a seasonal differencing operator, much like a standard differencing operator, to yield a seasonal ARIMA or an SARIMA. The seasonal differencing operator is the same as a regular differencing operator except that the differencing occurs over

a period determined by the seasonally of the given problem. The general SARIMA equation is [37]:

$$\phi_p(B)\Phi_P(B^s)\nabla^d\nabla^D{}_sX_t = \theta_q(B)\Theta_Q(B^s)Z_t \qquad (2.2)$$

where B is a backshift operator and is denoted by:

$$B^nZ_t = Z_{t-n} \qquad (2.3)$$

Where X_t are previous terms, and Z_t are random shocks. The autoregressive terms (AR) are denoted by

$$\phi_p(B) = (1 - \phi_1B - \phi_2B^2 - \ldots - \phi_pB^{p)}X_t \qquad (2.4)$$

a nonseasonal AR operator, and

$$\Phi_P(B^s) = (1 - \Phi_sB^s - \Phi_{2s}B^{2s} - \ldots - \Phi_{Ps}B^{Ps})X_t \qquad (2.5)$$

a seasonal AR operator. The seasonally is denoted s which represents the seasonal periods. The moving average (MA) terms are denoted by

$$\theta_q(B) = (1 - \theta_1B^1 - \theta_2B^{2-\cdots-}\theta_qB^{q)}Z_t \qquad (2.6)$$

a nonseasonal MA operator, and

$$\Theta_Q(B^s) = (1 - \Theta_sB^s - \Theta_{2s}B^{2s-\cdots-}\Theta_{Qs}B^{Qs)}Z_t \qquad (2.7)$$

a seasonal MA operator. The remaining terms are differencing term and are denoted by

$$\nabla^d = (1 - B)^dX_t \qquad (2.8)$$

a nonseasonal differencing operator, and

$$\nabla^D{}_s = (1 - B^s)^D \qquad (2.9)$$

a seasonal differencing operator. Equation 2.2 is referred to a (p,d,q) x (P,D,Q) SARIMA equation where the coefficients represent the degree of the AR, MA, and differencing performed. The coefficients p,d, and q refer to the non–seasonal components and the coefficients P,D, and Q refer to the seasonal components.

Using an SARIMA model involves a long, modeling process of identification, estimation, and diagnostic checking. Identification involves using autocorrelation (ACF)

functions, partial autocorrelation (PACF) functions , and range–mean plots to determine an appropriate starting model from the data set. The results of these test could indicate that the data might need to be transformed or differenced before the modeling process can continue. The ACF and PACF also provide information about what initial model to choose. Estimation involves obtaining values of ϕ , Φ, θ, and Θ for the tentative model chosen in the identification stage. This is usually accomplished by using a linear estimation method such as a least squares technique. These coefficients must meet certain mathematical criteria to ensure that the model is stationary and invertible. These criteria ensure model stability and can be examined further in a time series analysis book [37]. Diagnostic checking is the last step to determine it the chosen model is acceptable. This step involves residual analysis and determines if the identification stage is needed again.

Knowledge–Based Approach

Abnormalities in processes always represented a problem for modeling procedures. Some of these abnormalities are easily identifiable when examined by a human though. Knowledge based systems are efforts to tap into human knowledge in order to help in the modeling of abnormalities. STELF has such abnormalities present with holidays, weekends and days with unusual weather patterns. Knowledge based approaches were used for STELF in attempts to develop models that could deal with abnormal days such as holidays that are normally difficult to predict with other methods. This approach utilizes expert systems which are complex and vast sets of information and knowledge incorporated into a knowledge–base. The knowledge is usually supplied by a human expert. The information is represented by or translated into rules and facts. The rules and facts are arranged in a hierarchic pattern with decision branches. Decisions are made based upon the rules and facts by querying one and deciding what

path to take onto the next rule or fact where the process is repeated again. This process occurs until a solution is reached. Different situations will take different paths and could face separate rules. Rules and facts based upon the relationship between electric load curves, weather information and other variables make up a STELF expert system. Once a decision has been presented by an expert system, a user can trace the path that the decision was made upon to examine the logic and reasoning

CHAPTER 3: ARTIFICIAL NEURAL NETWORKS

Introduction

The human brain is probably regarded as the most advanced, complex, and powerful machine ever. A human can perform actions that no computer has ever been able to recreate such as pattern and image identification. An example of this is that infants can easily identify their parents from strangers. Even though infants can perform these actions, they cannot perform mathematical computations and even adults perform mathematical computations slowly and without exact precision. Computers, on the other hand, can perform numerous mathematical calculations in seconds, with great precision but are not good at pattern and image identification problems. Many pattern recognition problems such as telephone speech recognition and military targeting require speed and accuracy though. Artificial neural networks (ANNs) were developed in attempts to model biological systems for use in these pattern recognition types of problems. In fact, ANNs are good at solving problems that humans solve easily, such as pattern recognition, but are bad at solving problems, such as numerical computations, that computers can solve easily [7].

ANNs consists of simple processing elements, called neurons or nodes, which process signals through some sort of non–linear function to produce output signals. The signals usually come from other neurons so the produced outputs actually become inputs for other neurons. The neurons in an ANN are also usually highly interconnected so that an output signal from one node becomes the input for several other neurons. The connections between the neurons are usually weight values so that a neurons in

an ANN actually receives a weighted sum of input values. The processing elements and interconnections are the fundamental parts of ANNs. The arrangement of the neurons and inter–connections constitutes the ANNs architecture.

ANNs are used by utilizing some sort of data set, that contains a set of exemplars, to determine the internal parameters and values of the ANN so that it can be applicable for a given problem. Thus, ANNs 'learn' by determining some sort of classification or some sort of function mapping from the data sets. The manner in which the ANN utilizes the data sets constitutes a supervised or unsupervised learning scheme. In addition, both learning schemes have several different ways in which they operate. This refers to the manner in which the external data set is utilized in order in determine to internal ANN parameters and values so that learning occurs. The ANN developed for this thesis uses a multi–layer perceptron architecture with a supervised learning scheme that operates with a feed–forward, back propagation method for adjusting the internal parameters and values. The next sections will describe the features of the ANN written and used for this thesis. The ANN code is nnffbp1.f and can be found in Appendix A. More information about ANNs can be found in [7–11,19,21].

Multi–Layered Perceptions

Neurons can be arranged in many different architectures. A single row of neurons is referred to as a single–layer perceptron while a multi–layer perceptron (MLP) ANN consist of several neuron layers. Each layer consists of a number of neurons which are usually fully connected to the neurons in the preceding and proceeding layers but which are not connected to the neurons in it's own layer. For this work, the different layers were fully interconnected. The neuron layers are designated by names where

the first layer, the layer with no preceding layers, is referred to the input layer and receives any external signals that are to be processed by the ANN. Consequently, the last layer, the layer with no proceeding layers, is referred to as the output layer and produces the ANN output. The layers between the input and output layers are called hidden layers. An MLP ANN has one input layer and one output layer but can have several or no hidden layers. Figure 3.1 illustrates an MLP ANN architecture. Figure 2.1 is

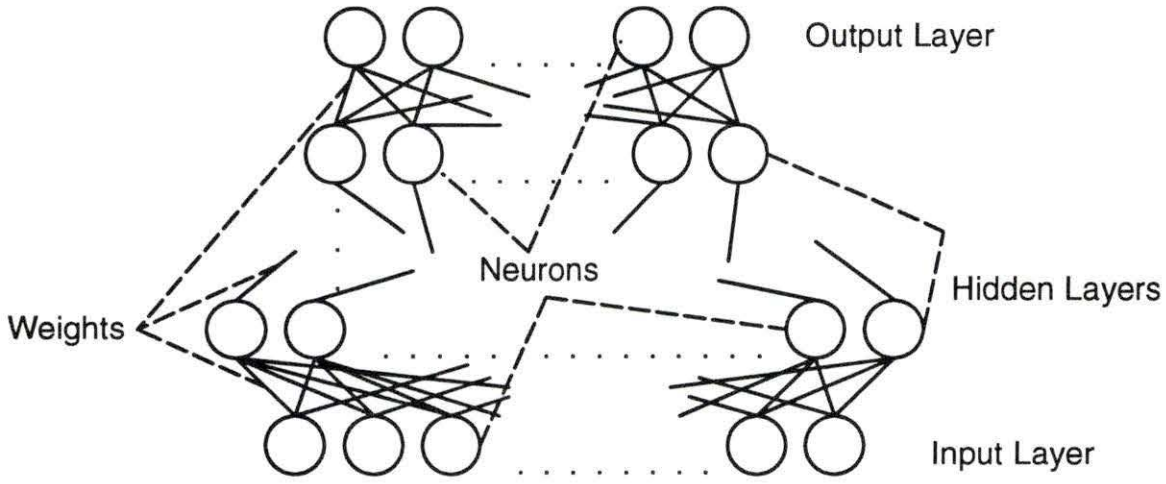


Figure 3.1 An MLP ANN architecture

referred to as a I x H1 x H2 x . . . x Hn x O ANN, where I is the number of neurons in the input layer, H1 is the number of neurons in the first hidden layer, and O in is the number of neurons in the hidden layer and where the ANN has n hidden layers. Sometimes the input layer in not counted when referring to the ANN architecture. For example, an architecture of 5 input neurons, 7 hidden neurons in one hidden layer, 3 hidden neurons in another hidden layer, and 1 output neuron is a 5x7x3x1, three–layer ANN. The reason for not counting the input layer when referring to the number of layers the ANN will be explained in a following section.

In most modeling problems, the number of input and output neurons is defined or determined by the problem. The dilemma lies in determining the number of hidden layers and then the number of neurons in each hidden layer. Kolmogorov's Theorem is used to show that any continuous function can be implemented in a two–layer ANN where the number of hidden nodes equals 2n+1 and where n is the number of input nodes [8]. There are also theorems that state that any function can be mapped using one hidden layer with an infinite number of neurons or two hidden layers and a finite number of neurons [4]. As a general rule, it is desired to have the least number of hidden neurons [5]. Having the least number of hidden neurons yields better generalization. With more hidden neurons, ANNs tend to memorize instead of generalize.

Supervised Learning

Supervised learning refers to using a data set that consists of examples or training patterns where the solutions are known. The examples are presented to the ANN to see if the outputs produced by the ANN matches the solutions of the given examples. These are referred to as the actual and desired outputs. It is quite likely that the actual and desired outputs will not match each other initially so the training patterns are usually presented to ANN in an iterating manner so that the ANN can adjust it's parameters and values so that it 'learns' the correct solutions of the examples. The different supervised learning schemes refers to the the different manners in which an ANN learns. An important note is that the examples used should be representative of the given problem so that the ANN can infer the correct relations between the inputs and the outputs so when novel information is presented, the ANN can formulate a reasonable output.

The manner in which the internal parameters and values were adjusted by the ANN in this work so that the learned occurred, happened in a two–step iterating pro-

cess. The first step is a feed forward process and the second step is a error back propagation process. This learning scheme is referred to as a feed forward back propagation (FFBP) method and quite common. The feed forward process refers to the processing of the examples in the data sets in the ANN by first presenting them to the input layer then through the hidden layers and finally through the output layer to produce outputs. The back propagation process refers to the adjusting of the internal parameters and values so that the actual and desired outputs match.

Feed Forward

The examples in the data sets are first presented to the ANN via the input layer. The input layer then takes those input signals and sends them directly to the neurons in the hidden layer without processing them through non–linear transfer functions. This is the reason that the input layer is often not included when referring the the number of layers in an ANN. Each neuron in the hidden layer then receives a weighted sum of the inputs from the neurons in the input layer. The neurons in the output layer also receive a weighted sum of inputs, but from the hidden layer. This sum is represented by the following equation,

$$sum_j = \sum_{i=1}^{n} w_{ij} o_j \qquad (3.1)$$

where sum_j is the sum for neuron *j* in the current layer, w_{ij} is the weight value from neuron *i* in the previous layer to neuron *j* in the current layer, o_i is the output value from neuron *i* in the previous layer, and *n* is the number of neurons in the previous layer. This equation is used for all the neurons in the current layer. Figure 3.2 illustrates this process for a one–layer MLP with two input neurons and one hidden neuron.

The input of the neuron, the weighted sum, now passes through a non–linear

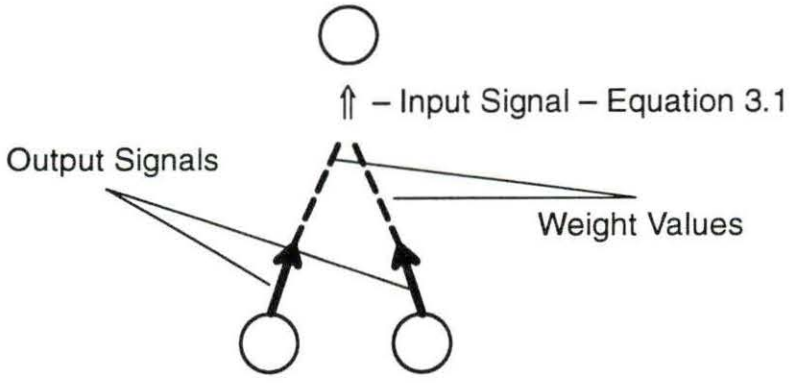


Figure 3.2 Summation process in a 2x1 ANN

transfer function. The function used for this work was an exponential sigmoid function. This function is represented by the following equation.

$$f(sum) = \frac{1}{\left(1 + e^{-[g(sum-t)]}\right)} \qquad (3.2)$$

where *g* is a gain value, *t* is a threshold value, and *sum* is the weighted sum value from equation 3.1. The sigmoid function allows the neurons to produce week, strong, or intermediate activations based on the value of the weighted summed input. The standard exponential sigmoid function has threshold value of zero and a gain value of unity and is illustrated in Figure 3.3. Using different values for the gain and threshold moves the

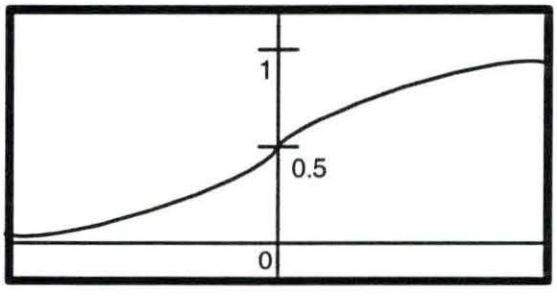


Figure 3.3 A sigmoid function

curve from the 0.5 intersection point and gives the curve a steeper or flatter shape. This activation function was used in all of the hidden and output layer neurons.

Back Propagation

Now that the ANN has produced an output based on the current ANN weights, the difference between the actual and desired output will be used to adjust the internal parameters so that the difference between the two will be minimized. The internal parameters actually refer to the weight values for the ANN in this thesis. It should be noted that the starting values of the weights were set randomly to small values. The learning scheme used in this thesis to adjust the weights was the back propagation (BP) algorithm derived by Hecht–Nielsen [20]. BP is a gradient decent approach that adjust the weights. The weights and the errors between the actual and desired outputs can be plotted against each other [9] to create an error surface. This error surface will be n–dimensional hyperparabola with peaks and valleys. SInce BP is a gradient decent method, the weight values will move in the direction of the negative gradient or towards a valley where the error is small.

The second stage of the FFBP process begins with the adjustment of the weight values between the output layer and the hidden layer preceding it. These weights are adjusted by using the Delta rule [8,20] which is given by

$$W_{new} = W_{old} + \beta E x / |x|^2 \qquad (3.3)$$

where W_{old} is the old weight value, β is a constant defined as the learning rate, E is the error of the neuron output, and x is the neuron input. It is interesting to note that when E = 0, the neuron has produced the correct result so the new weight value will equal the old weight value and thusly, remains unchanged. The error term in Equation 3.2 can be computed easily for the layer between the output layer and the hidden layer

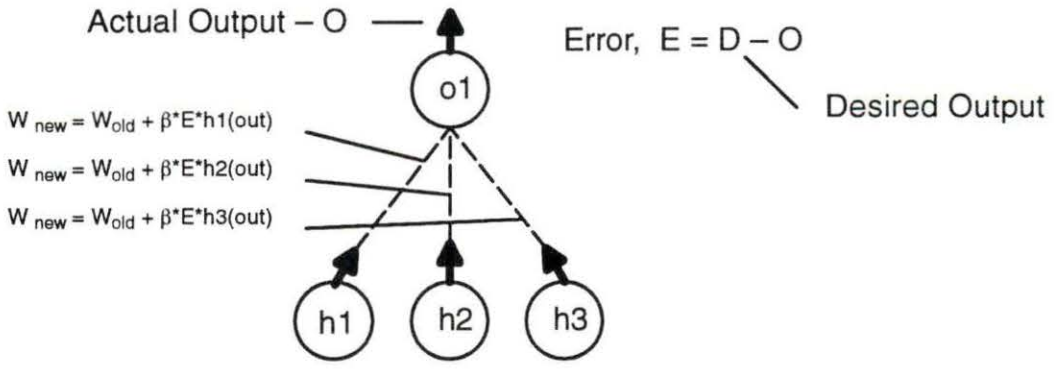


Figure 3.4 Delta Rule for Output Layer Neurons

preceding it because the actual output from the output neurons can be used along with the desired output of the training pattern to determine the error. This weight adjustment is illustrated in Figure 3.4.

Changing the weights between the hidden layers or between the hidden and input layers is more difficult because the desired output is not known for the neurons in the hidden layer. For this reason the Generalized Delta Rule must be used to change the weights [9,20]. For this case where the desired neuron output in not known, the BP algorithm determines each hidden neurons contribution to the error of the output neurons. The corresponding contribution is then used in the error determination for the weight change calculation. This rule is basically the same as the Delta rule except for calculation of the error which is given by

$$E_h = f'(x)\left[\sum_{i=1}^{n} w_{ij}E_j\right] \qquad (3.4)$$

where E_o is the error or the output neurons and w_{h-o} are the weights that connect the output neurons to a hidden node. The remaining term is the derivative of the exponential sigmoid activation function which is given by

$$f'(x) = f(x)\,[\,1 - f(x)\,] \qquad (3.5)$$

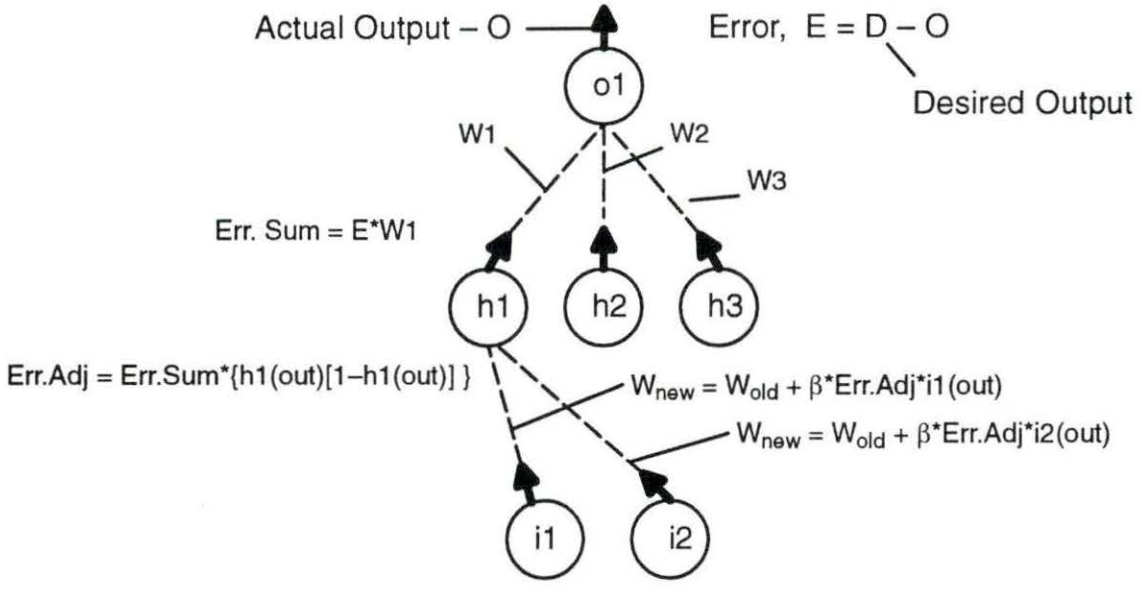


Figure 3.5 Generalized Delta Rule for hidden neurons

It is easy to show that this equation is just the neuron output multiplied by one minus the neuron output. The Generalized Delta rule is in Figure 3.5

Learning Rate

The learning rate, β, is the rate in which the weights change. That is, the learning rate is multiplied by the weight change calculation. This variable can range from zero to one. Large values, around .9, will cause the ANN to converge quickly but can also cause the convergence to oscillate around the minimum due to the large steps taken on the error surface. If the ANN was close to the minimum on the error surface, a large learning rate could cause the next weight jump to miss the minimum and climb the the other side. Small values around .1, on the other hand, will reduce the possibility of oscillation, but will cause the ANN to converge more slowly. Ideally, the learning rate should start out near one and move towards zero as the ANN nears convergence. The ANN developed for this thesis used two learning rates, one for the hidden layer and one

for the output layer. This was incorporated so that the learning between level could be balanced in necessary.

Batch Training

Presenting the training patterns can occur in two fashions. The first method presents one pattern at a time then adjust the weights in the FFBP process. In this method, the ANN adjust it's weights for one pattern then adjusts it's weights for the next pattern. This can cause the ANN to forget some or all of what it has learned by the previous weight adjustment. This, in turn, can cause the ANN to converge slowly due to the fact that it may have to relearn each training pattern. The second method presents all of the training patterns before adjusting the weights. This method is referred to as batch training and was utilized in this work. For batch training, the error value is actually the accumulated error for all the training patterns. This allows the ANN to remember the incorrect response for each training pattern before adjusting the weights. The weights are adjusted only after all of the training patterns have been feed forward through the ANN.

RMS Error and ANN Convergence

After the ANN has performed one iteration of feed forward and backpropigation, or one pass of the entire training set, some criteria must be used to determine if the network has converged. Convergence refers to the correctness in the ANN output and a criteria refers to utilizing some cost function to determine convergence. There are many cost functions that can be used to accomplish this but the one used for this work was the Root Mean Square (RMS) function. The RMS function is common and is most

often used. This equation is given by

$$RMS = \sqrt{\frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij}^2} \qquad (3.6)$$

where m is the number of output neurons, n is the number of training patterns and d is the difference between the actual and desired outputs.

Momentum

As the dimension of an ANN grows, so will the size and complexity of the error surface. With the complexity of an error surface, there is the possibility of the ANN converging into a local minima instead of the global minimum. This ideal is expressed in Figure 3.6 with a two dimensional error surface. This is a common problem with BP and

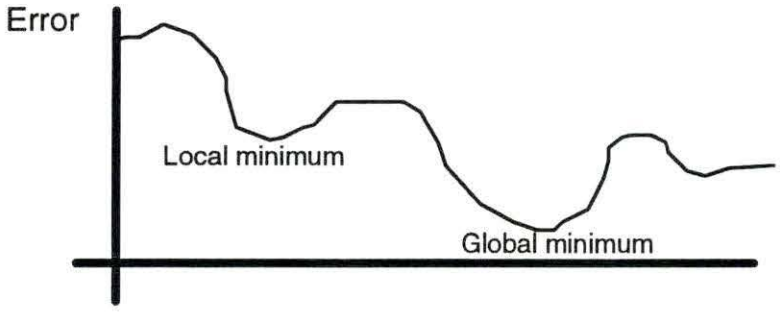


Figure 3.6 Error surface and local minimum illustration

a common solution and the one used in this thesis to help deal with this was to use a momentum term. A momentum term is designed to use the "momentum" built up during downhill descents on the error surface so that enough speed or inertia is present to make it up and over the local minimums. This term is employed by slight modification of the Delta rule

$$W_{new} = W_{old} + \beta E x / |x|^2 + \alpha(W_{new} + W_{old})_{Prev} \qquad (3.7)$$

where α is a momentum rate and *Prev* refers to the previous weight change. As with

the learning rate, the momentum rate is between zero and one. In addition to escaping local minima, the momentum term also helps the ANN performance because it tends to speed up convergence. This is the case because the speed of the downhill descents is increased due to the momentum.

Verification

SInce an ANN was developed for this work, it was first tested with some benchmark problems to determine if was functioning properly. The benchmark problems used were the exclusive–or (XOR) problem and the eight–to one decoder problem. The XOR problem involves two inputs and one output where these values exist in either an on or off pattern. Zero and one are typical values used for this procedure to represent on and off. Zero and one were used for inputs but the outputs used were 0.1 and 0.9 which are close to zero and one, respectively. The results of this are illustrated in Table 3.1. The ANN used for this was a 2x3x1, two layer network. The ANN was trained to an RMS of 0.01

The eight–to one decoder problem also involved inputs and outputs with on or off representations. The input consisted of three neurons where different combinations of on and off patterns would indicate the desired activation for the output. The output

Table 3.1 XOR inputs and outputs.

Input 1	Input 2	Desired Output	Actual Output
0.0	0.0	0.1	0.01023
0.0	1.0	0.9	0.9044
1.0	0.0	0.9	0.8838
1.0	1.0	0.1	0.00995

consisted of eight neurons where one of eight output neurons would be on or have a value of one, and where the remaining input signals would be off or have values of zero. The output signals were represented with 0.9 values to represent one and 0.1 values to represent zero. The desired and actual outputs are illustrated in Table 3.2. This ANN was a 3x9x8 two layer network and was also trained to an RMS of 0.01. The ANNs performed well on the benchmark problems so the assumption that the ANN was valid and functioning properly was made.

Table 3.2 8–to–1 decoder inputs and outputs.

Input			Desired Output								Actual Output							
0.0	0.0	0.0	0.9	0.1	0.1	0.1	0.1	0.1	0.1	0.1	.88	.11	.10	.10	.10	.09	.09	.10
0.0	0.0	1.0	0.1	0.9	0.1	0.1	0.1	0.1	0.1	0.1	.10	.89	.10	.11	.09	.10	.10	.10
0.0	1.0	0.0	0.1	0.1	0.9	0.1	0.1	0.1	0.1	0.1	.10	.11	.90	.10	.09	.10	.11	.10
0.0	1.0	1.0	0.1	0.1	0.1	0.9	0.1	0.1	0.1	0.1	.09	.09	.10	.91	.12	.10	.10	.10
1.0	0.0	0.0	0.1	0.1	0.1	0.1	0.9	0.1	0.1	0.1	.09	.10	.10	.09	.90	.09	.10	.10
1.0	0.0	1.0	0.1	0.1	0.1	0.1	0.1	0.9	0.1	0.1	.10	.09	.10	.10	.10	.90	.11	.09
1.0	1.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.9	0.1	.10	.09	.09	.10	.10	.09	.90	.09
1.0	1.0	1.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.9	.09	.10	.10	.09	.10	.12	.10	.89

CHAPTER 4: PROBLEM AND ARTIFICIAL NEURAL NETWORK METHOD

Data Collecting and Processing

An electric load curve displays highly seasonal behavior. Seasonally refers to the load curve having peaks and valleys with a regular period. This seasonally is due to the fact that society, for the most part, operates on a regular schedule. Society works during the day and sleeps during the night. However, these peaks and valleys do not all posses the same magnitude. One week might have relatively uniform peaks and valleys while the next might show greater variation. One explanation of the variation in magnitude can be attributed to weather factors. For example, when the temperature rises in the summer, the use of air conditioners also rises which requires electrical power. Another source of variation arises through societal factors. For example, most of society works during weekdays while having weekends and holidays off. Consequently, the load demand on these days will be lower due to the non use of industrial equipment. The data required for the ANN modeling of STELF will consist of electric load data to account for the seasonally and weather and societal variables to account for the variability.

The data used in this thesis consisted of hourly electric loads from Omaha Public Power District (OPPD) during the years of 1990, 1991, and 1992 [36]. Hourly weather variables for the same years from the Omaha, Nebraska area were also available [22]. These weather variables include air temperature, soil temperature, wind speed, wind direction, precipitation relative humidity , and solar radiation. Societal variables which indicate the day of the week and holidays were also used the the ANN models but were generated.

The electric load and weather variables were available in individual data files but the societal variables needed to be generated. In addition, the electric load and weather variables needed to be normalized so that the ANNs used could utilize them. Recall that the transfer function was a sigmoid function and that it is asymptotic at zero and one. The electric load and individual weather minimum and maximum values were found and used to normalize the data between 0.1 and 0.9. A computer program was written to read the electric load and weather files, normalize the data, generate societal variables, and to create an input/output set for use in the ANNs. The program is called loadread.f and can be seen in Appendix A. Since the data files were available on an hourly basis for an entire year, the program created input/output sets which had a moving window of an hour.

The generation of the societal variables involved establishing booleans. For the day of the week indicator, seven variables were used so that one could represent each day. Each day would have their variable equal one while the other variables would equal zero. For a Monday, the first variable would have a value of one while the other six variables had a value of zero. The holiday indicator consisted a single variable where a one would indicate a holiday and a zero would indicate a regular day.

Artificial Neural Network Model Development

The first step in developing an ANN model was to examine the data. The electric load curves have been plotted by year and by month and can be found in Appendix B. Many items can be noted from these curves. The first item deals with the shapes of the curves from the yearly plots. The middle portions display greater

values and greater variability. These middle sections represent summer months. The greater magnitude of the load peaks can be attributed to the fact that there is need for cooling which is done with electric air conditioners. The remaining part of the yearly plots, the ends, are the winter months and tend to be fairly constant. This can be explained by the use of natural gas and heating oil, instead of electricity, for heating.

Another item can be noted in the summer and winter months by examining the monthly graphs. There are daily peaks and valleys during both seasons but the shapes of these peaks differ. The peaks for the winter generally display two humps while the peaks for the summer generally display only one hump. The two humps in the winter can be attributed to the industrial and commercial use during the day for the first hump and to the residential lighting and general use during the evening for the second hump. The time between represents commuting time when people are between work and home. The single hump in the summer can be attributed to the longer daylight hours requiring less lighting and the continual use of air conditioning. The differences between the seasons prompted the development of two ANN models, one for the summer and one for the winter. The summer ANN model ranged from the last two weeks in May to the first two weeks in September while the winter ANN model covered the remainder of the year not covered by the summer ANN model. This period was chosen by examining the curves for appropriate cutoffs.

In addition to dividing the ANNs into seasonal models, the ANNs for forecasting the week ahead were also divided into separate models. This kept the ANN model from creating a large ANN with 168 outputs. For both winter and summer models ANNs were developed to perform one hour ahead predictions, 24 hours ahead predictions, and 24 hours ahead predictions for up to seven days ahead. The total num-

ber of hourly networks for each season equaled eight, one for the 24 hour period tomorrow, one for the 24 hour period for the day after tomorrow, and so on for seven days and and eighth for hourly predictions. The hourly ANNs were developed to model and simulate load dispatch while the 24–hours ahead ANNs were developed for unit commitment and reserve capacity allocation scheduling. The ANN models developed are listed in Table 4.1.

Table 4.1 List of ANN STELF models

ANN Number	Description
w1h	Winter model, one hour ahead prediction
w24h	Winter model, next 24 hours ahead prediction
w24h2d	Winter model, 2 days ahead 24 hours prediction
w24h3d	Winter model, 3 days ahead 24 hours prediction
w24h4d	Winter model, 4 days ahead 24 hours prediction
w24h5d	Winter model, 5 days ahead 24 hours prediction
w24h6d	Winter model, 6 days ahead 24 hours prediction
w24h7d	Winter model, 7 days ahead 24 hours prediction
s1h	Summer model, one hour ahead prediction
s24h	Summer model, next 24 hours ahead prediction
s24h2d	Summer model, 2 days ahead 24 hours prediction
s24h3d	Summer model, 3 days ahead 24 hours prediction
s24h4d	Summer model, 4 days ahead 24 hours prediction
s24h5d	Summer model, 5 days ahead 24 hours prediction
s24h6d	Summer model, 6 days ahead 24 hours prediction
s24h7d	Summer model, 7 days ahead 24 hours prediction

In developing 7 ANNs to perform 24 hours ahead predictions for an entire week, several items were considered. The first dealt with developing small networks. By developing 7 ANNs, each with 24 outputs, instead of one ANN with 168 outputs, the dimensionality of the problem was reduced so that the corresponding error surface would be less complex. This would help the training and learning of the ANN models by creating easier problems. This also allowed the data sets to contain training patterns that were similar in nature. Another consideration was that separating the ANNs into 7 models allowed the predictions to be updated daily. This would provide a new weekly–hourly prediction on a daily basis.

Selection of Input Variables

The selection of input variables for the ANNs involved choosing appropriate historical electric load data and pertinent weather data along with the inclusion of the societal variables. The next sections will cover each of these areas and the manner in which the inputs were selected.

Historical Inputs

Seasonallity in any time series indicates that historical data will be useful for any modeling [37]. For this reason, historical electric load data was utilized to indicate the most likely path of the data. The problem was in determining the amount of historical data to utilize. Determining the amount of previous historical data for use in the one–hour ahead ANN models involved examining the load data which showed that the distance from a peak to a valley on a load curve can cover 10 hours. Using the 12 previous hours should provide sufficient periodic information to indi-

cate what the next hour might be. Hourly ANNs were tested with different amounts of historical load data and 12 proved to be a reasonable number. In addition to the information from the immediate previous electric load, previous periodicity also provides information. Daily periodicity can provide information because loads that are 24 hours apart will be somewhat close in value. However, they will not be close for days involving weekends and holidays due to the societal influence of those day types. In addition, loads that are a week apart will also provide information because the days are the same so the societal influence from day types, excluding holidays, will be minimal. Thusly, these days that are a week apart will be similar except for the influence of weather.

For 24 hour ahead predictions, the 24 hours of load data from the previous week was used due to the fact that it basically possesses the same societal data of the predicted day. For example, most industries start up every Monday and shut down on Fridays. Thusly, days that are a week apart should display nearly the same shape or behavior, excluding the influence of weather.

Weather Inputs

Choosing weather variables involved determining which ones caused variation in the load curves. Many weather variables are known or suspected to affect the behavior of the electric load curves. Air temperature is one and Stoll has shown how higher temperatures correlate with higher peak loads [42]. In determining what additional weather variables to include, besides air temperature, into the ANN models, a series of experiments were performed. Several ANNs were set up to establish an importance ranking of weather variables. This was done by creating ANNs that contained the same historical input but only one of the seven weather variables. These ANNs were trained with the same training sets, except for the weather vari-

ables, to the same RMS value. These ANNs were then tested on the same test set, again the only difference being the weather variable, to examine the test RMS. The ratio of test to training RMS was taken to determine the most valuable weather variables. By using this measure, a generalization method can be used to determine a ranking. For both winter and summer periods, air temperature was among the highest ranked but that is where the similarity ended. For the winter months, wind speed also ranked high. Thus, air temperature and wind speed were used in the winter models. For the summer periods, three weather variables were used due to the fact that this period displayed greater variances in load peaks and valleys. The variables that ranked high for this period were air temperature, relative humidity, and solar radiation. It is interesting to note that the variables that ranked high in these experiments are also commonly used by other methods and are considered important for STELF in general [6,13].

Now that the weather variables were determined, the manner of their use as inputs still needed to be determined. For the one hour ahead ANNs, historical weather information was used to indicate the general trend of the weather variable in question. By obtaining information on the trend, the most likely proceeding weather value could be used to determine the influence it might have on the electric load. Instead of using the 12 previous weather values as with the historical electric load data, the 6 previous values were used instead. Only 6 previous values were used because weather data typically displays more of a random behavior instead of seasonal behavior. With typical random behavior, less information is gained from previous values that are further in distance. Although there is little correlation in previous weather information, there was still information available. Since the electric load has correlation due to the seasonallity, the weather data that corresponds with the previous 24th hour or days load and previous 168th hour or weeks load will

be of value to provide correctional information for the differences for the deviation from seasonally.

The 24 hour ahead models used the weather information in a different manner. Since the goal was to predict the entire 24–hour period, more general weather terms were sought. Daily averages were computed and used in predicting the overall size of the daily curve. For the summer time, higher average air temperatures mean hotter days and increased electrical consumption due to air conditioner use. In addition, partial daily averages were also used to help estimate the shapes of the curves. They were established by averaging the hourly weather information in 4–hour blocks. Consequently, a day would have 6 different daily averages. The separation began at 12am to 4am and continued with 4am to 8am , 8am to 12pm, 12pm to 4pm, 4pm to 8pm, and finished with 8pm to 12am. These were used to indicate the weather information at various points along the electric load curve.

Since the weather is known to effect the magnitude of the load curve it would be beneficial if the weather was known for the future period corresponding with the future period of the desired forecasted load. Fortunately, forecasted weather is performed and can be used to help this process. Unfortunately, the data set used in this work did not contain forecasted weather. To overcome this dilemma, pseudo–forecasted weather was generated from the actual weather data. The was performed by adding Gaussian noise to the actual weather data to simulate forecasted weather information. The noise was based on the errors of forecasting periods that ranged from hours in advance to days in advance [17,23,35]. The forecasting errors were generalized for all the weather variables and are as follows: 1 day – 3%, 2 days – 4%, 3 days – 5%, 4 days – 7%, 5 days – 10%, 6 days – 13%, and 7 days – 17%. Since the pseudo–forecasted weather was generated using Gaussian noise, generalizing the error for all of the weather variables seemed reasonable.

Societal Inputs

As mentioned earlier, the societal inputs used were day indicators and holiday indicators. The day indicators were used to indicate to the ANN the day being predicted. While weekends are typically different than weekdays, weekdays are usually similar, baring weather, to one another except for societal differences from sources such as early week industrial start up activities or late week industrial shut down procedures. For example, if the the last twelve hours of load data for a weekday were being utilized in determining the first hour of the next day, how would the ANN know if the next day was a weekend or a weekday? The boolean was used to indicate the day that was being predicted. For the hourly ANN models, w1h and s1h, the day indicator represented the day of the week that the predicted hour fell on even if the historical load data was from the same day, the previous day, or from both the previous and current day. SInce the 24 hour ahead models were separated into training patterns for every 24 hours, the historical data utilized was entirely from the previous day or days.

The holiday variable indicated if there was a holiday or a regular day. As with the daily indicator, the holiday indicator was used to indicate the status of the predicted hour or day. There are many holidays in a calendar year and the number for a certain area is difficult to determine without exact data of the policies of the area employers. This information was not available so the graphs were examined to determine which holidays displayed decreased loads. Out of the standard government holidays, seven holidays were found to affect the expected pattern by producing electric load curves that display peaks and overall shapes that are smaller than expected. The holidays used for the ANN models are New Years Day, Memorial Day, Independence Day, Labor Day, Thanksgiving, the day after, and Christmas.

Figure 4.1 illustrates the architecture of one of the ANNs developed for this work, w1h. Table 4.2 shows the architectures for the one–hour ahead ANN models, Table 4.3 shows the architectures for the winter ANN models and Table 4.4 shows the architectures for the summer ANN models.

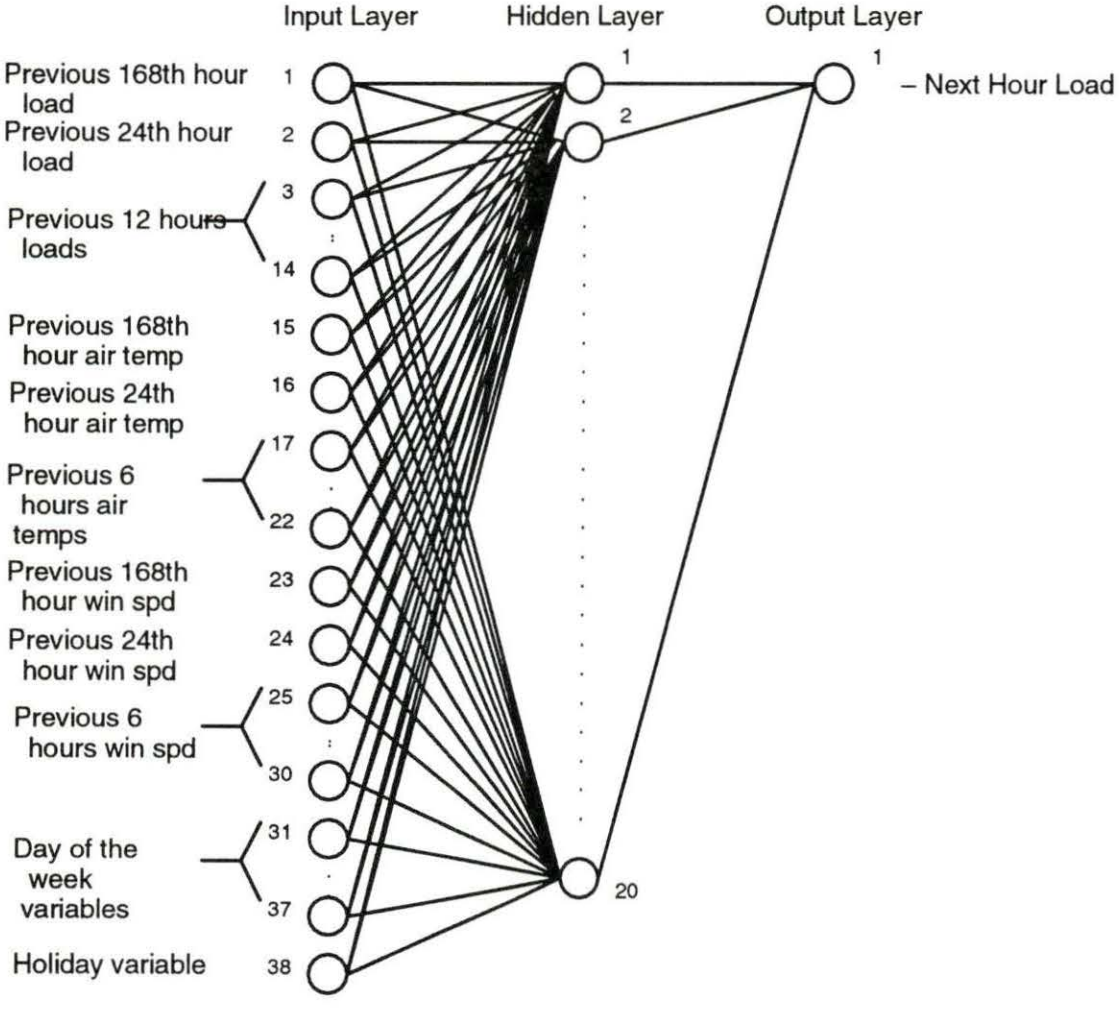


Figure 4.1 Input node designation for winter one hour ahead ANN

Table 4.2 ANN architectures for one–hour ahead ANN models.

Summary of architectures designations for one–hour ahead ANN models					
w1h – 20 hidden neurons, 1 output neuron			s1h – 30 hidden neurons, 1 output neuron		
Input #	Description	Values	Input #	Description	Values
1	Previous Hourly Electric Loads	–168th	1	Previous Hourly Electric Loads	–168th
2		–24th	2		–24th
3...14		–12th ... –1st	3...14		–12th ... –1st
15	Previous Hourly Air Temperatures	–168th	15	Previous Hourly Air Temperatures	–168th
16		–24th	16		–24th
17...22		–6th ... –1st	17...22		–6th ... –1st
23	Previous Hourly Wind speeds	–168th	23	Previous Hourly Relative Humidities Values	–168th
24		–24th	24		–24th
25...30		–6th ... –1st	25...30		–6th ... –1st
31...37	Day Indicator	boolean	31	Previous Hourly Solar Radiation Values	–168th
38	Holiday indicator	0 or 1	32		–24th
Output	Forecasted Load	Value	33...38		–6th ... –1st
1	Next Hour	1st	39...45	Day Indicator	boolean
Note: "–1st" is the value before the value to be predicted "1st".			46	Holiday indicator	0 or 1
			Output	Forecasted Electric Load	Value
			1	Next Hour	1st

Table 4.3 ANN architectures for 24–hour ahead ANN winter models

Summary of architecture designations for 24–hour ahead winter ANNs								
Models – 50 hidden, 24 output neurons		w24h	w24h2d	w24h3d	w24h4d	w24h5d	w24h6d	w24h7d
Input	Description	Values						
1...24	Previous Hourly Electric Loads	–168th ... –145th	–144th ... –121st	–120th ... –97th	–96th ... –73rd	–72nd ... –49th	–48th ... –25th	–24th ... –1st
25...32	Air Temps. Total Daily Averages	–7d ... –1d & 1d	–6d ... –1d & 1d .. 2d	–5d ... –1d & 1d .. 3d	–4d ... –1d & 1d .. 4d	–3d ... –1d & 1d .. 5d	–2d ... –1d & 1d .. 6d	–1d & 1d .. 7d
33...38	6 Daily Hourly Averages	–7d	–6d	–5d	–4d	–3d	–2d	–1d
39...44		1d	2d	3d	4d	5d	6d	7d
45...52	Wind Speed Total Daily Averages	–7d ... –1d & 1d	–6d ... –1d & 1d .. 2d	–5d ... –1d & 1d .. 3d	–4d ... –1d & 1d .. 4d	–3d ... –1d & 1d .. 5d	–2d ... –1d & 1d .. 6d	–1d & 1d .. 7d
53...58	6 Daily Hourly Averages	–7d	–6d	–5d	–4d	–3d	–2d	–1d
59...64		1d	2d	3d	4d	5d	6d	7d
65...71	Day Indicator	Boolean						
72	Holiday	0 or 1						
Output	Description	Value						
1...24	Forecasted Electric Load	1st ... 24th	25th ... 48th	49th... 72nd	73rd... 96th	97th... 120th	121st... 144th	145th... 168th

Notes: For weather variables, a"–" sign indicates previous values while no sign indicates to a pseudo–forecasted value. A "–1d" is the last value before "1d" where "–1d" is the last known day and "1d" is the next day.

Table 4.4 ANN architectures for 24–hour ahead ANN summer models

Summary of architecture designations for 24–hour ahead summer ANNs								
Models – 60 hidden, 24 output neurons		w24h	w24h2d	w24h3d	w24h4d	w24h5d	w24h6d	w24h7d
Input	Description	Values						
1...24	Previous Hourly Electric Loads	–168th ... –145th	–144th ... –121st	–120th ... –97th	–96th ... –73rd	–72nd ... –49th	–48th ... –25th	–24th ... –1st
25...32	Air Temps. Total Daily Averages	–7d ... –1d & 1d	–6d ... –1d & 1d .. 2d	–5d ... –1d & 1d .. 3d	–4d ... –1d & 1d .. 4d	–3d ... –1d & 1d .. 5d	–2d ... –1d & 1d .. 6d	–1d & 1d .. 7d
33...38	6 Daily Hourly Averages	–7d	–6d	–5d	–4d	–3d	–2d	–1d
39...44		1d	2d	3d	4d	5d	6d	7d
45...52	Rel Hum Total Daily Averages	–7d ... –1d & 1d	–6d ... –1d & 1d .. 2d	–5d ... –1d & 1d .. 3d	–4d ... –1d & 1d .. 4d	–3d ... –1d & 1d .. 5d	–2d ... –1d & 1d .. 6d	–1d & 1d .. 7d
53...58	6 Daily Hourly Averages	–7d	–6d	–5d	–4d	–3d	–2d	–1d
59...64		1d	2d	3d	4d	5d	6d	7d
65...72	Solar Rad. Total Daily Averages	–7d ... –1d & 1d	–6d ... –1d & 1d .. 2d	–5d ... –1d & 1d .. 3d	–4d ... –1d & 1d .. 4d	–3d ... –1d & 1d .. 5d	–2d ... –1d & 1d .. 6d	–1d & 1d .. 7d
73...78	6 Daily Hourly Averages	–7d	–6d	–5d	–4d	–3d	–2d	–1d
79...84		1d	2d	3d	4d	5d	6d	7d
85...91	Day Indicator	Boolean						
92	Holiday	0 or 1						
Output	Description	Value						
1...24	Forecasted Electric Load	1st ... 24th	25th ... 48th	49th... 72nd	73rd... 96th	97th... 120th	121st... 144th	145th... 168th

Notes: For weather variables, a"–" sign indicates previous values while no sign indicates to a pseudo–forecasted value. A "–1d" is the last value before "1d" where "–1d" is the last known day and "1d" is the next day.

ANN Testing and Training

Testing and training the ANN models involved partitioning the electric load and weather data into two distinct sets. The training set consisted of the 1990 and 1991 data while the test set consisted of the 1992 data. An original training set consisted of only 1990 data, but it was soon realized that set alone was not enough information for the 24–hour ahead ANN models. The single year did not provide enough patterns with all the possible input/output combinations. The training set that contained the 1990 and 1991 sets performed much better.

For the one hour ahead ANNs, the training sets were generated with loadread.f which used a moving window of an hour. The corresponding files contained 5808 training patterns for the summer ANN model and 11543 training patterns for the winter ANN model. The ANN models were trained with these large sets and tested with their corresponding test sets. The winter test set contained 5880 patterns and the summer test set contained 2904 test sets. These large training sets were also stratified with the program loadstat.f to reduce their size in efforts to save computational time. The winter set was sampled at ever 9th pattern to produce a new training set with 1432 patterns. The summer set was sampled at every 7th pattern to produce a set with 953 patterns. The reduced sets contained the entire hours for holidays so that the ANN models would still get a good representation of the holiday dynamics. The performance of these reduced training sets on the full test sets was nearly the same as the performance of the full training sets.

The 24–hour ahead models were developed to make predictions on a daily basis so their training sets were sampled at every 24 pattern to yield daily patterns. The summer models all contained 242 patterns in their training sets and 121 patterns in their test sets. The number of patterns in the winter ANN models are illus-

trated in Table 4.5. The reason that the number of training and test patterns decreases for the ANN models with increasing prediction days is due to the fact that the increasing distance of the predicted day from the beginning of the data sets causes one less day due to the finite size of the data available.

Table 4.5 Size of training and test sets for winter ANN models.

ANN Model	Number of Patterns in Training Set	Number of Patterns in Test Set
w24h	481	241
w24h2d	480	240
w24h3d	479	239
w24h4d	478	238
w24h5d	477	237
w24h6d	476	236
w24h7d	475	235

CHAPTER 5. RESULTS

Other Results

This section will present STELF results from other papers. Both traditional methods and ANN methods will be presented. Traditional methods will be covered by presenting a paper that directly compares the five most common traditional methods. ANN methods will be covered by reviewing several papers that implement ANN methodologies.

Traditional Methods

As stated previously, there are many methods for performing STELF, and Moghram and Rahman [32] have published a paper with a comparison of the five most common traditional methods. These methods include a multiple linear regression approach, a stochastic time–series approach including transfer function modeling, an expert–system approach, a general exponential smoothing approach, and a state–space and Khalman filter approach. The results presented in this paper are illustrated in Table 5.1. The paper was done to illustrate these different methods and to make a comparison with one another by making predictions for the same day. The comparisons also involved using the same data set to develop the individual models. The paper also briefly illustrates the modeling procedure involved in developing the individual models. These modelling illustrations show the processes involved in using these methods and shows that some of the models development can be quite involved and can require a high degree of skill.

Other ANN Models

A survey was also performed on other ANN models. The other ANN models developed performed a wide variety of predictions which ranged from hourly predictions to the prediction of the next days peak load. The other models also used different sized data sets for testing and training which ranged from a few weeks to a few months. Some of the ANNs even used different approaches in their design and execution which included the use of self–organizing ANNs for data segregation to non–fully connected architectures for input enhancement. A summary of these ANNs can be found in Table 5.2. Some of the results produced by the ANNs in Table 5.2 compare well with some of the results from Table 5.1.

Table 5.1 Illustration of traditional STELF results

Analysis and Evaluation of Five Short–Term Load Forecasting Techniques Moghram and Rahman [32] Forecasting Period – Hourly forecasts for 24 hours ahead			
Model	Data Set	% Summer Error	% Winter Error
Multiple Linear Regression	4 weeks of hourly data	2.78	3.76
Time Series – SARIMA	4 weeks of hourly data	0.79	2.17
Time Series – Transfer Function		0.51	2.70
General Exponential Smoothing	5 previous weekdays	2.12	1.79
State Space and Kalman Filter	4 weeks of hourly data	1.57	1.71
Expert System	Selection of reference day	1.22	1.29

Table 5.2 Comparison of other ANN models

Comparison of Other ANN Models					
#	Ref.	Prediction Period(s)	Data Set	% error	Notes
1	[39]	a. 24hr, one week ahead b. 24hr ahead (T–F)	Tr. – Winter peak Utility Te. – a. 5–mo. period from Feb.1 b. 1000 hr. period (T–F) from Jan. 1	a. 3.38 b. 2.09	a. Performed decomposition of load and used 5 Adaline to model. Each day had own set of ANNs.
2	[13]	Next days peak	a. Tr. 1/12–1/25 Te. 1/26–2/1 b. Tr. 1/19–2/1 Te. 2/2–2/8	a. 1.15 b. 1.22	Nonfully connected ANN, Pseudo forecasted weather.
3	[38]	a. peak load b. total load c. hourly load	Tr. 11/1/88 – 1/30/89 , except test Te. 1. 1/23–1/30 2. 11/9–11/17 3. 11/18–11/29 4. 12/8–12/15 5. 12/27–1/4	Average a. 2.04 b. 1.68 c. 1.40	Focus – normal weekdays, i.e. no holidays or weekends.
4	[16]	One hour ahead	Tr. 20 days Te. 25 days	4.1	Used adaptive NN, no weekends.
5	[2]	One half hour ahead	16 months of data from 4/90–7/91 forecasted for one year	Average Win.–3.16 Spr.–3.67 Sum–2.69 Aut.–4.24	Modular ANN design partitioned into season and day types, total of 48 ANNs. No holidays.
6	[39]	Total daily load	Tr. 1 yr. Te. 1 yr. x is total of both years	2.95	ANN retrained each day because if forecasting period =x, then training cases = x–1. Data divided into 5 subsets for day types.
7	[1]	Peak load on Thursday	6/83–4/94 16 patterns for training	3.11	
8	[29]	a. 24hr ahead b. 1hr ahead	Forecasted on 6 mo. Feb. to Jul.	a. 1.89 b. 1.84	a. Weekend and weekday models weekends grouped into 5 types.
9	[31]	a. 1hr ahead b. 24hr ahead c. peak load	One year or data divided into 12 train/test groups. Tr. 1 to 2 mo. Te. 7 days	a. 2.28 b. 3.08 c. 2.44	Errors averaged for same experiments for different companies. 12 ANN models, 1 for each mo.
10	[27, 28]	peak and valley	Tr. 10 previous similar days Te. the next similar day	p. 1.57 v. 0.20	Day identification w/self–organizing ANN. Retraining for each day.
11	[26]	peak and valley	Tr. 30 previous similar patterns Te. next 7 similar days	P. 0.10 v. 0.26	Day type identification. Only 4 examples presented.

Artificial Neural Network Solution

The results produced by the ANN models from this thesis for the entire test set, which consisted of data for an entire year, are illustrated in Table 5.3. It is now desired to make some comparison between the ANN models in this thesis and the the ANN models from Table 5.2. The comparison between the results produced in this thesis and the results from the other papers is difficult to make though due to the fact that the ANNs from Table 5.2 used different data sets with regards to their composition and to their sizes. Regardless, the ANNs in this thesis produced results that compared well with the other ANN models. This is the case because the results produced in this thesis contained a years worth of data for ANN testing which was larger than most test sets

Table 5.3 Summary of ANN performance over entire training set

Results of ANN models on entire test sets		
ANN Model	Winter Models (Test RMS) / Avg. % Err	Summer Models (Test RMS) / Avg. % Err
1 hour ahead	(0.01613) / 2.003	(0.01719) / 2.134
1 day ahead, 24 hours	(0.02464) / 3.212	(0.02502) / 3.303
2 days ahead, 24 hours	(0.02918) / 3.454	(0.03013) / 3.528
3 days ahead, 24 hours	(0.03167) / 3.521	(0.03000) / 3.472
4 days ahead, 24 hours	(0.03088) / 3.502	(0.03254) / 3.605
5 days ahead, 24 hours	(0.03392) / 3.832	(0.03468) / 3.793
6 days ahead, 24 hours	(0.03764) / 4.325	(0.03723) / 4.217
7 days ahead, 24 hours	(0.04247) / 4.674	(0.04085) / 4.5525

in Table 5.2. Generally, one expects deteriorated performance with larger data sets but the performance of the ANNs in this work performed well. Also, many of the other methods excluded weekends and holidays in training and testing their models. This was done to eliminate these unusual or abnormal days from their training and testing procedure so that their accuracy could be enhanced. To help illustrate this point, test sets were created without holidays and weekend days. Table 5.4 is a summary of the ANN models performance, in terms of % error, without weekends or holidays in the test sets. There is some improvement by excluding weekends and holidays from the test sets but the overall performance was not significantly better. The ANNs models in this work were able to model these days. Modeling weekends and holidays involves choosing the appropriate inputs.

Table 5.4. Summary of ANN performance over entire data set containing no holidays or weekend days

Results of ANN models on entire data set containing no holidays or weekend days		
ANN Model	Winter – % Error	Summer – % Error
1 hour ahead	1.923	2.042
1 day ahead, 24 hours	3.103	3.263
2 days ahead, 24 hours	3.318	3.375
3 days ahead, 24 hours	3.485	3.407
4 days ahead, 24 hours	3.493	3.549
5 days ahead, 24 hours	3.738	3.632
6 days ahead, 24 hours	4.113	4.152
7 days ahead, 24 hours	4.523	4.492

Figures 5.1 and 5.2 illustrate some predictions with the one–hour ahead winter ANN model, w1h, for a holiday period and a non–holiday period, respectively. Figures 5.3 and 5.4 illustrate some predictions with the one–hour ahead summer ANN model, s1h, for a holiday period and a non–holiday period, respectively. All of these figures show that the one–hour ahead ANN models performed well, regardless of a holiday, weekend, or normal day.

Figures 5.5 through 5.9 illustrates the 24–hours ahead winter ANN models for the holiday period from Figure 5.1. Figures 5.10 through 5.14 illustrates the 24–hours ahead summer ANN models for the holiday period from Figure 5.3. These figures illustrate the predictions made by the seven seasonal ANN models. The figures are separated by a single day to illustrate the daily update. For example, the seventh day in Figure 5.5 is predicted with ANN model, w24h7d, and with ANN model, w24h3d in Figure 5.9. These predictions show that the ANN models performed fairly well but still lacked top accuracy. The accuracy was reasonable with some of the the other ANN models from Table 5.2, considering the sizes of the data sets involved.

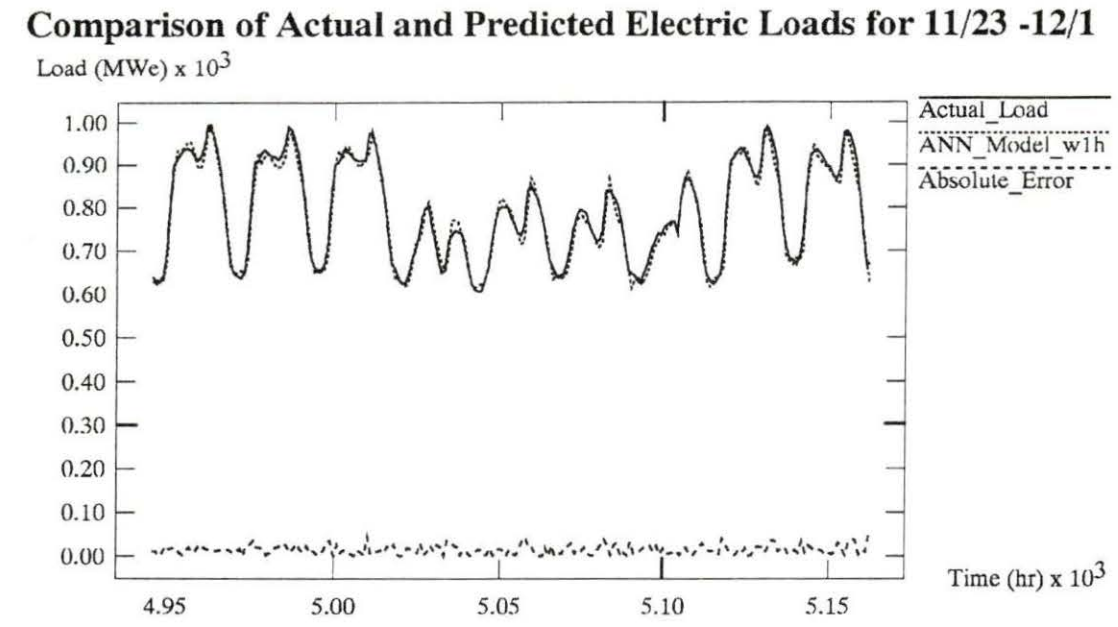


Figure 5.1 ANN model w1h on a holiday period

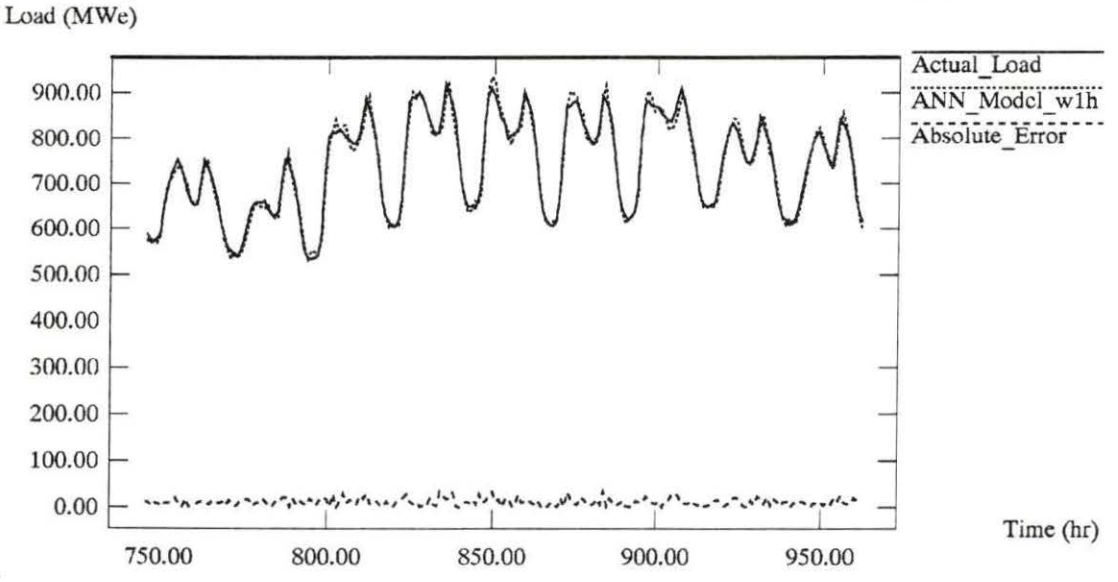


Figure 5.2 ANN model w1h on a non–holiday period

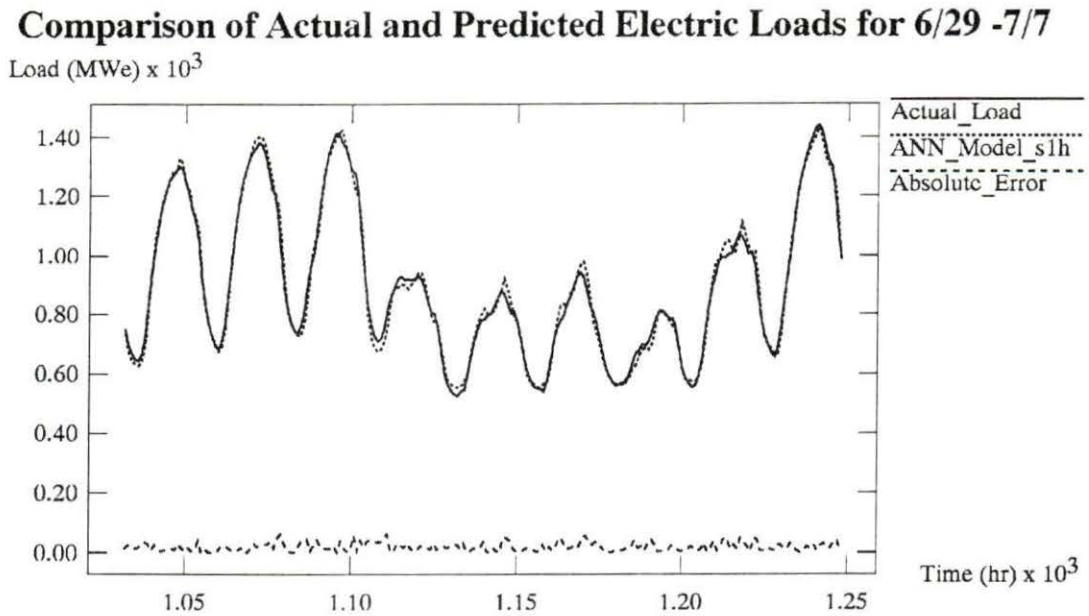


Figure 5.3 ANN model s1h on a holiday period

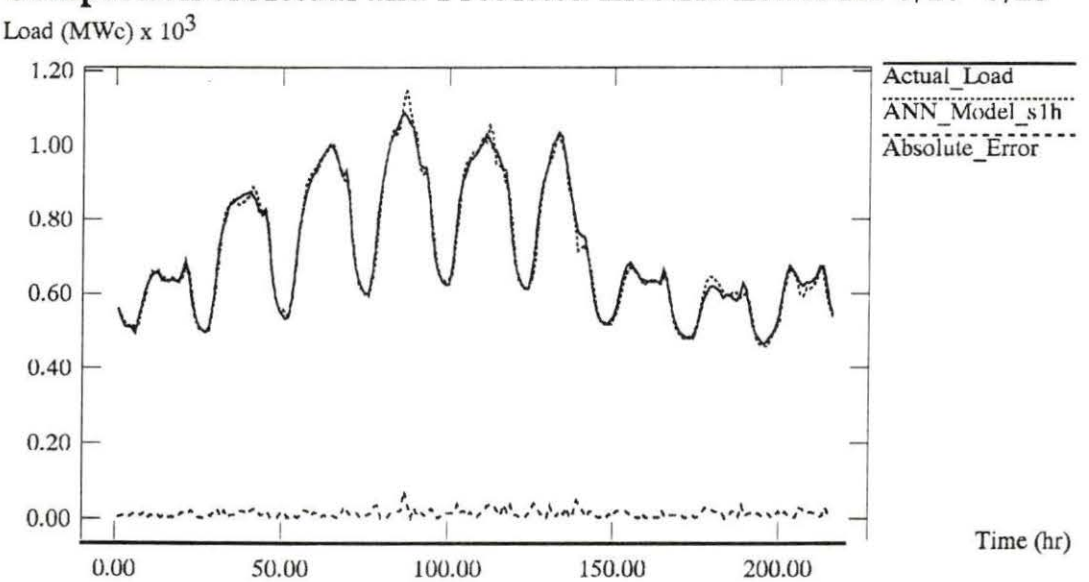


Figure 5.4 ANN model s1h on a non–holiday period

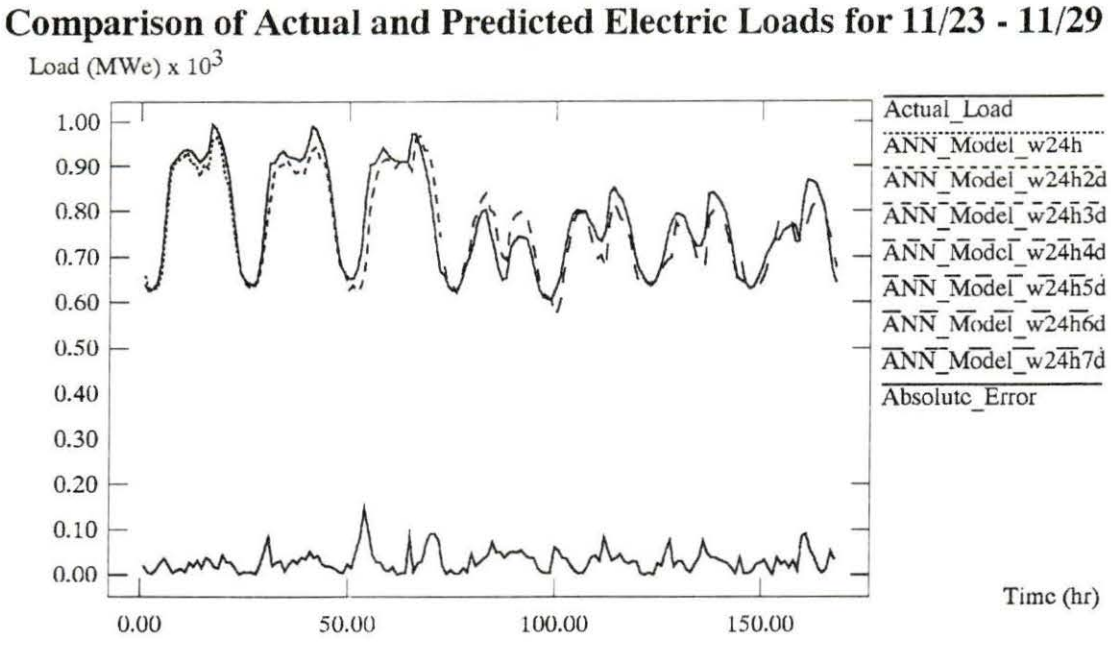


Figure 5.5 24–hour ahead winter ANN models for a holiday period

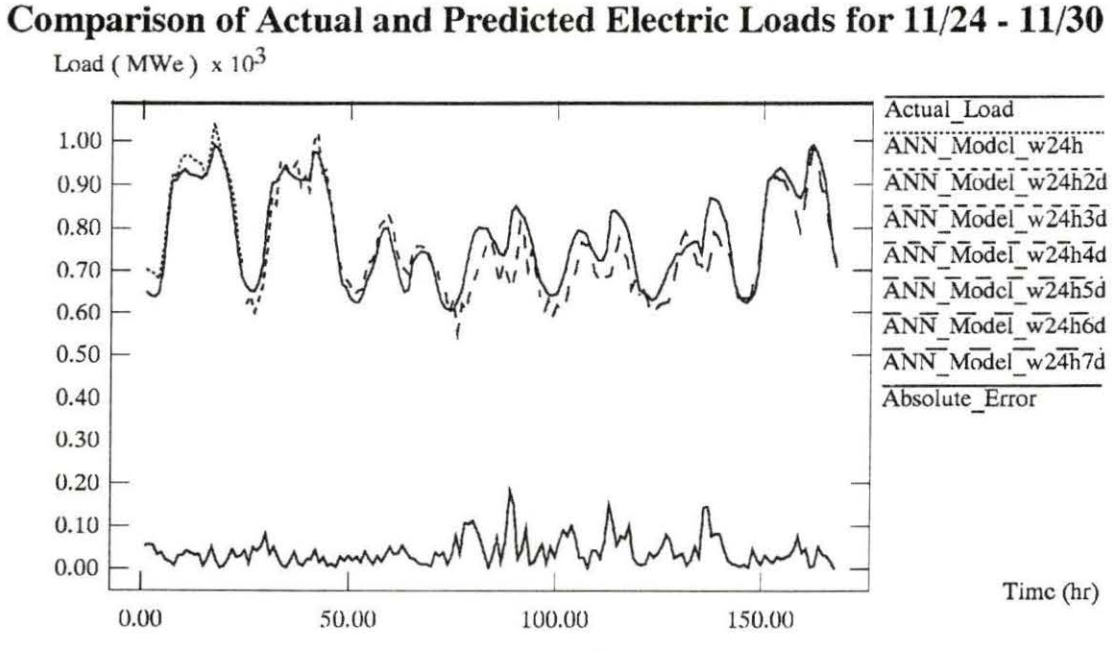


Figure 5.6 24–hour ahead winter ANN models for a holiday period, updated one day

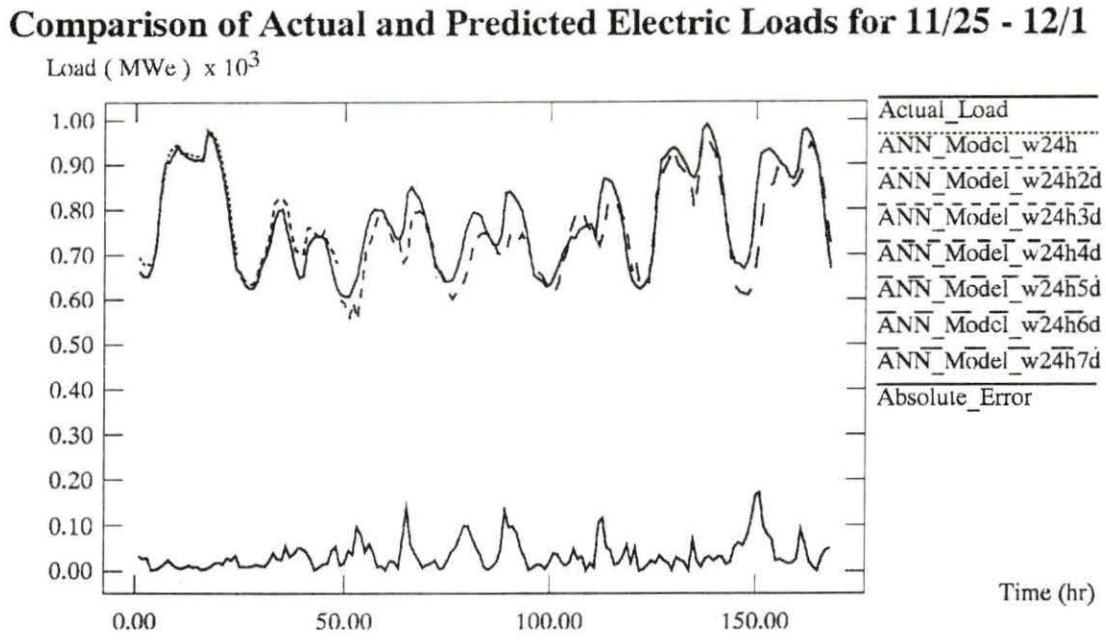


Figure 5.7 24–hour ahead winter ANN models for a holiday period, updated two days

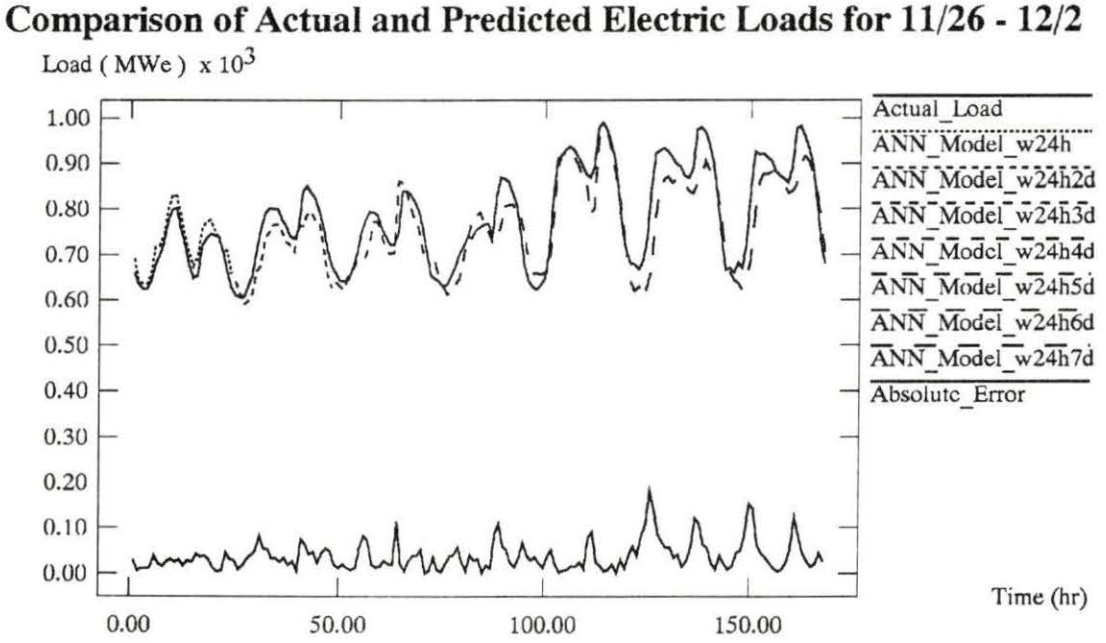


Figure 5.8 24–hour ahead winter ANN models for a holiday period, updated three days

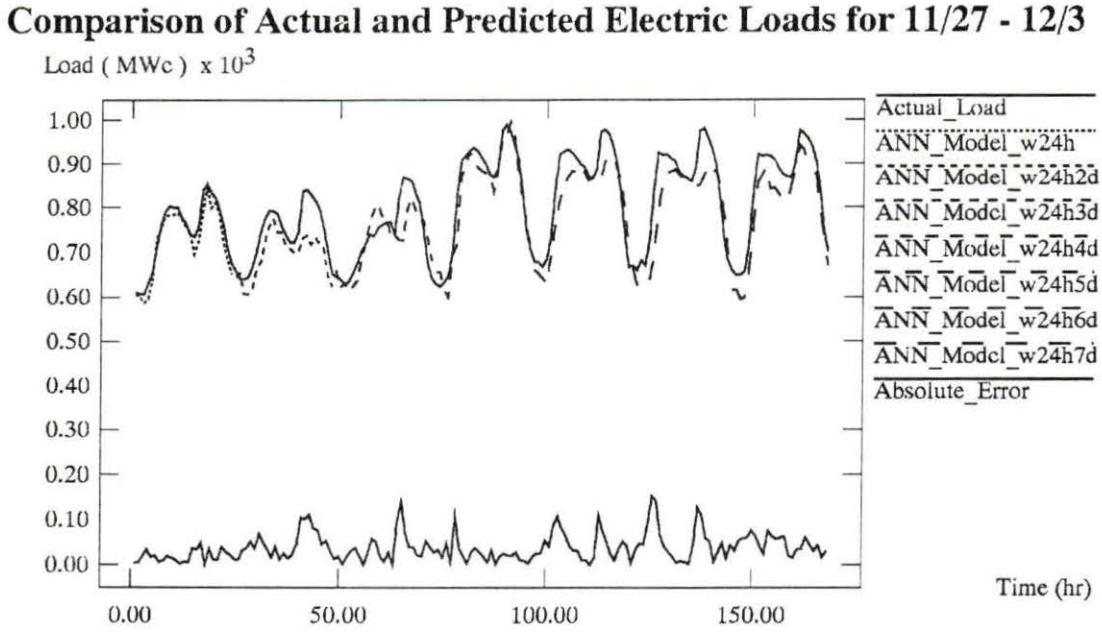


Figure 5.9 24–hour ahead winter ANN models for a holiday period, updated four days

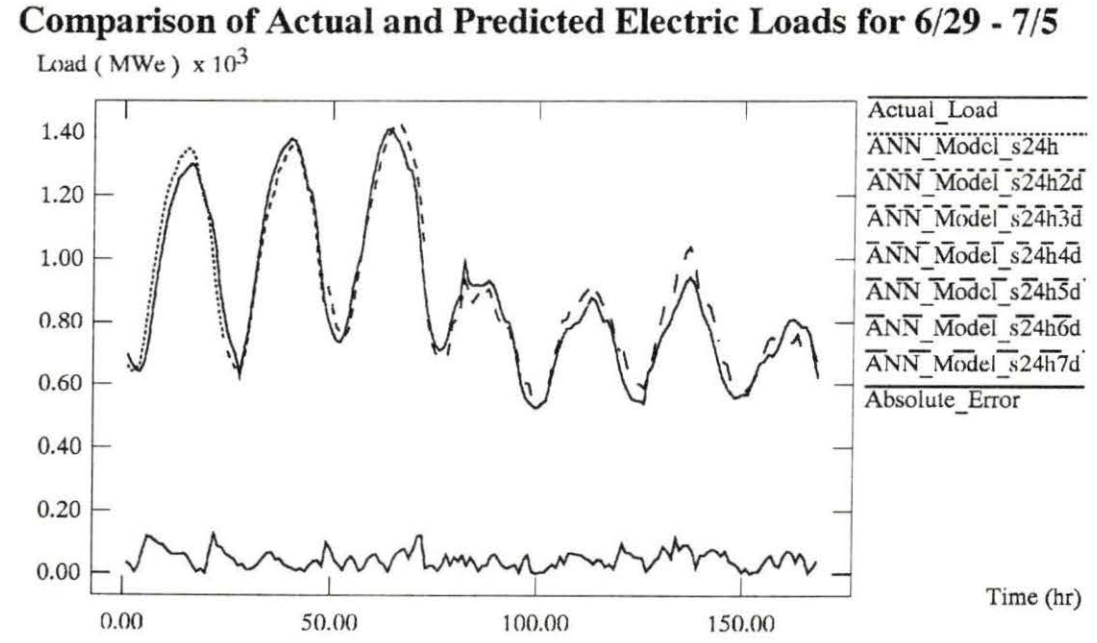


Figure 5.10 24–hour ahead summer ANN models for a holiday period

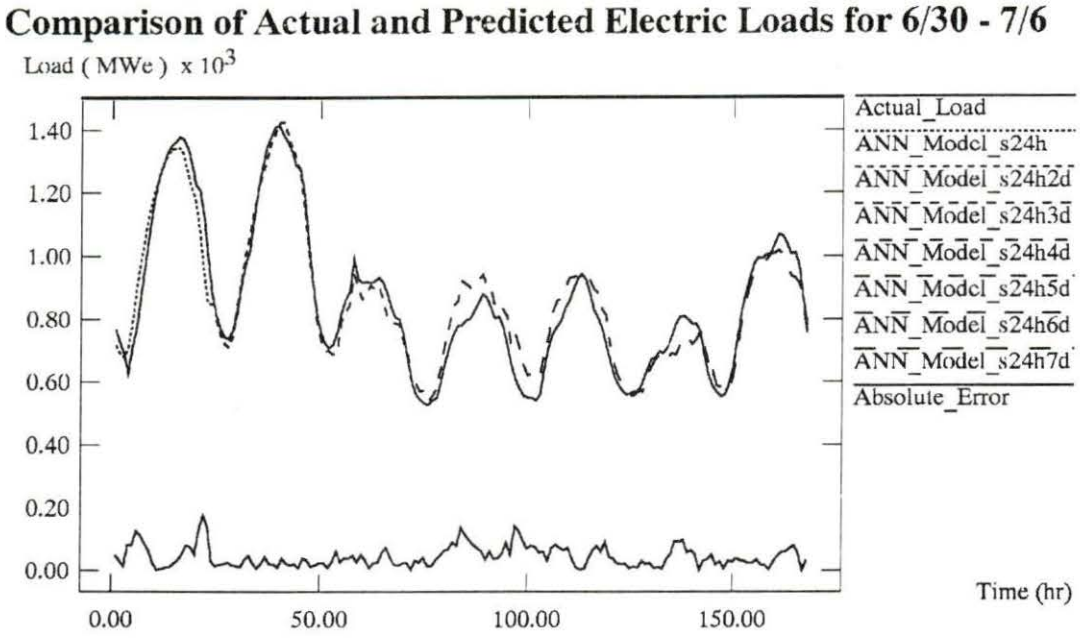


Figure 5.11 24–hour ahead summer ANN models for a holiday period, updated one day

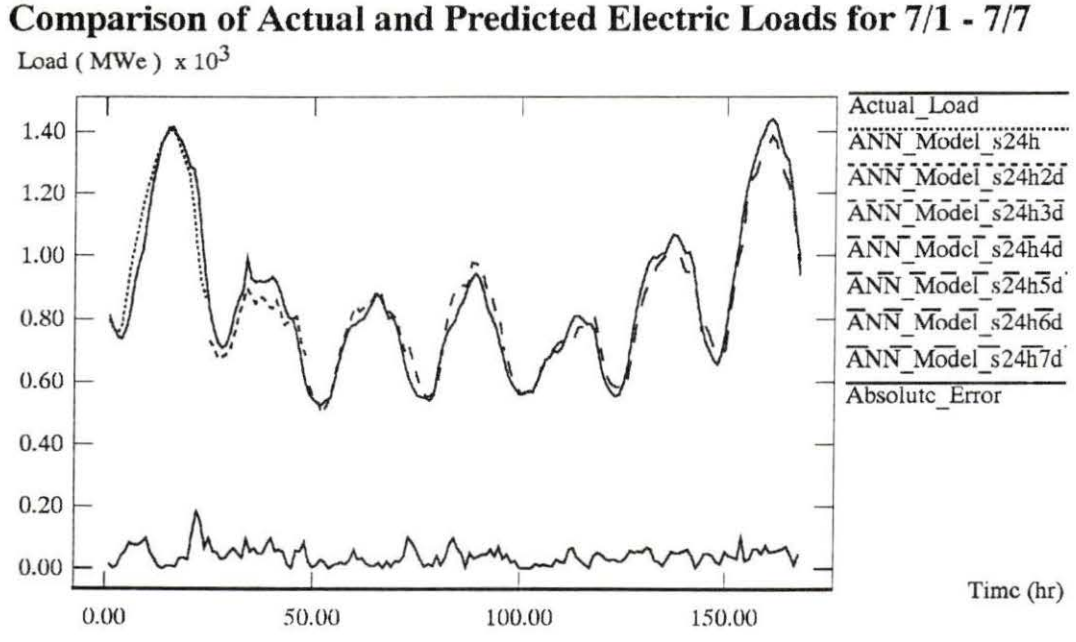


Figure 5.12 24–hour ahead summer ANN models for a holiday period, updated two days

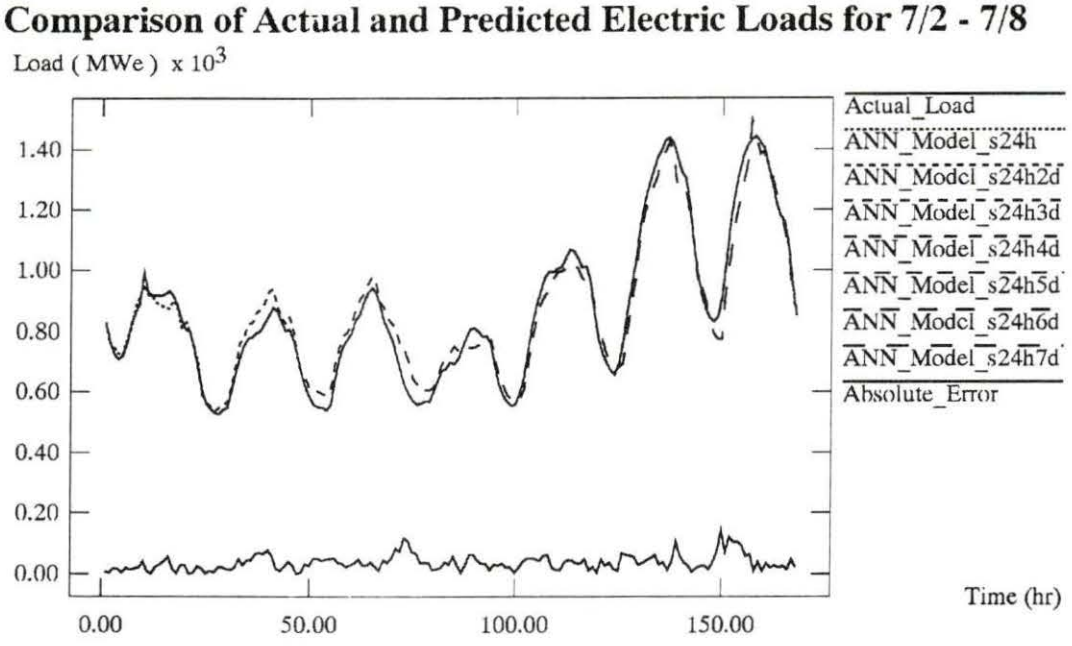


Figure 5.13 24–hour ahead summer ANN models for a holiday period, updated three days

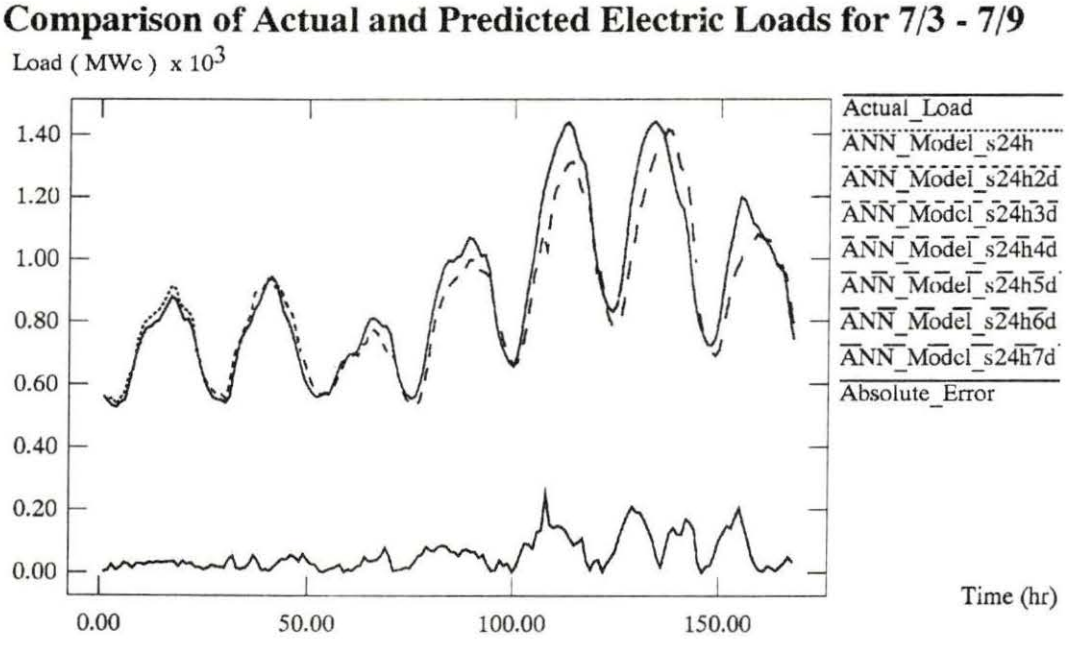


Figure 5.14 24–hour ahead summer ANN models for a holiday period, updated four days

Discussion

From the results produced by this work and from the survey of other ANNs given in Table 5.2, the accuracy of an ANN methodology for STELF can be quite good but generally does not match current methods for now. However, ANNs do benefit in the area of computational time. It is true that the initial time investment is great due to the training of the ANN models, but once this stage is over, the predictions can be performed almost instantaneously. In addition, as the work in this thesis shows, these predictions can be made for an entire year after the initial training process is completed. Statistical approaches such as time–series and multi–linear regression modeling have to be continuously performed to give reasonable results.

Another area where an ANN method is beneficial is in the ease of modeling. Modeling any problem requires two types of knowledge. The first type of knowledge is a good understanding of the problem. A good understanding of the problem is important so that correct components of the problem are used in obtaining a solution. The second type of knowledge involves a good understanding of a tool or a method for solving the problem. To use statistical methods, complex assumptions regarding important input variables often have to be made. Also, as mentioned earlier, an expert–system may be very difficult to program. ANNs, on the other hand, do not require any complex assumptions for the inclusion of input variables. Thus, ANNs can help with the second type of knowledge with an easier modeling process.

CHAPTER 6. CONCLUSIONS

Summary

The work done in this thesis shows the feasibility of ANNs for STELF. The results produced in this thesis compare favorably with other ANN models and some of the other ANN results compare favorably with traditional methods. The accuracy achieved by ANNs can be good but overall are still not quite as good as traditional methods. However, ANNs do provide significant time savings in the fact that, once trained, can perform predictions instantaneously. ANNs can also save time by being utilized to make predictions for an entire year with a single set of models. There is no need to create separate models for holidays or weekends. The ease of modeling is also a benefit. This allows the modeling process to be performed without having to make assumptions on the modeling process itself. Overall, ANNs are close to traditional methods for STELF and it is just a matter of time before the accuracy catches up with the other benefits of computational time and ease of modeling.

Future Work

There are many areas to explore for possible future work. One such area involves utilizing ANN techniques to improve performance to produce more accurate STELF. The selection of hidden nodes can be performed by utilizing a dynamic node architecture scheme as demonstrated by Basu [5]. This would improve the generalization of the ANN by selecting the appropriate number of hidden nodes. Another tech-

nique that could be used is an importance determination method as used by Lanc [24]. Lanc utilizes this method to determine the importance of input variables. This could be used for STELF by creating an input layer with all of the suspected variables and letting the ANN determine the important ones. An extension of this would be to use the importance idea to rank the training patterns so that a reduction could be performed to reduce the size of the data set while keeping the important ones while eliminating the redundant ones. Another idea would be to Use a learning scheme such as the one Bartlett [4] introduces. He uses a stochastic code which generally trains better than Back–Propagation.

Another area that could be investigated is to use more data in the training process. The performance or the 24–hour ahead ANN models improved when the training set increased from one year to two. There is no reason not to believe that the performance can benefit even more with more data. Since load and weather data are recorded and kept by most utilities, this would be little problem. Also, other societal variables could be examined. Variables such as television ratings, computer sales, or the patterns of recreational athletic leagues could be important. These variables, on the other hand, could be more difficult to obtain.

BIBLIOGRAPHY

[1] Azzam–ul–Asar, and J.R. McDonald. "A Modularized Artificial Neural Network Approach to Short Term Load Forecasting." Proceedings of the 1994 America Power Conference (April, 1994): 360–364.

[2] Azzam–ul–Asar, and J.R. McDonald. "A Specification of Neural Network Application the Load Forecasting Problem." In Press, IEEE Transactions on Control Systems Technology (1994).

[3] Adela Maria Bolet, Editor. Forecasting U.S. Electricity Demand Trends and Methodologies. Boulder, Colorado: Westview Press, 1985.

[4] E.B. Bartlett. "Nuclear Power Plant Status Diagnostics Using Simulated Condensation: An Auto–Adaptive Learning Technique." Ph.D. Dissertation, University of Tennessee at Knoxville. (1990).

[5] A. Basu. "Nuclear power plant status diagnostics using a neural network with dynamic node architecture." M.S. Thesis, Iowa State University. (1992).

[6] D.W. Bunn and E.D. Farmer, Editors. Comparative Models for Electrical Load Forecasting. Dublin, Northern Ireland: John Wiley & Sons Ltd. 1985.

[7] M. Caudill. "Neural Networks Primer, Part 1." AI Expert 6 (Dec., 1987): 46–52.

[8] M. Caudill. "Neural Networks Primer, Part 2." AI Expert 7 (Feb., 1988): 55–61.

[9] M. Caudill. "Neural Networks Primer, Part 3." AI Expert 7 (June, 1988): 53–59.

[10] M. Caudill. "Neural Networks Primer, Part 4." AI Expert 7 (Aug., 1988): 61–67.

[11] M. Caudill. "Neural Networks Primer, Part 5." AI Expert 7 (Nov., 1988): 57–65.

[12] Kanad Chakraborty, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. "Forecasting the Behavior Of Multivariate Time Series Using Neural Networks." Neural Networks 5 (April, 1992): 961–970.

[13] Shin–Tzo Chen, David C. Yu, and A.R. Moghaddamjo. "Weather Sensitive Short–Term Load Forecasting Using Nonfully Connected Artificial Neural Network." IEEE Transactions on Power Systems 7(3) (Aug., 1993): 1098–1105.

[14] Robert T. Crow, Michael Robinson, and Raymond L. Squitieri. Forecasting Electricity Sales and Loads: A Handbook for Small Utilities. Los Altos, California: American Public Power Association, 1981.

[15] M.J. Damborg, M.A. El–Sharkawi, M.E. Aggoune, and R.J. Marks II. "Potential of Artificial Neural Networks in Power System Operation." Proceedings of the 1990 IEEE International Symposium on Circuits and Systems (May, 1990): 2933–2937.

[16] T.S. Dillon, S. Sestito, and S. Leung. "Short Term Load Forecasting Using an Adaptive Neural Network."IEEE Transactions on Power Systems 13(4) (Aug., 1991): 186–192.

[17] Neil Duncan. "Who says they can forecast the weather?" New Scientist (June, 26 1993): 44–45.

[18] W.J. Freeman. "The Physiology of the Perceptron." Scientific American (Feb., 1991): 78–85.

[19] R. Hecht–Nielsen. Neurocomputing. New York, New York: Addison–Wesley Publishing Company, 1990.

[20] R. Hecht–Nielsen. "Theory of the Backpropigation Neural Network." Proceedings of the International Joint Conference on Neural Networks, 1, 1989. 593–605.

[21] J. Hertz, A. Krogh, and R.G. Palmer. Introduction to the Theory of Neural Computation. Redwood City, California: Addison–Wesley Publishing Company, 1991.

[22] Hourly Weather data for Omaha, Nebrasks, High–Planes Climatic Center, Lincoln, Nebraska. , 1990–1992.

[23] Richard A. Kerr. "Who Can Forecast the Worst Weather?" Science 250 (Oct. 5, 1990) 29–31.

[24] T.L. Lanc. "The importance of input variables to a neural network fault–diagnostic system for nuclear power plants." M.S. Thesis, Iowa State University, (1992).

[25] Alan Lapedes and Robert Farber. " Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling." Technical Report LA–UR–87–2662, Los Alamos National Laboratory, 1987.

[26] Kun–Long Ho, Yuan–Yih Hsu, and Chien–Chuen Yang. "Short Term Load Forecasting Using a Multilayered Neural Network with an Adaptive Learning Algorithm." IEEE Transactions on Power Systems 7(1) (Feb., 1992): 141–149.

[27] Yuan–Yih Hsu, and Chien–Chuen Yang. "Design of Artificial Neural Networks for Short–Term Load Forecasting. Part I: Self–Organizing Feature Maps for Day Type Identification." IEE Proceedings–C 138(5) (Sept., 1991): 407–413.

[28] Yuan–Yih Hsu, and Chien–Chuen Yang. "Design of Artificial Neural Networks for Short–Term Load Forecasting. Part II: Multilayer Feedfoward Networks for Peak Load and Valley Load Forecasting." IEE Proceedings–C 138(5) (Sept., 1991): 413–418.

[29] K.Y. Lee, Y.T. Cha, and J.H. Park. "Short–Term Load Forecasting Using an Artificial Neural Network." IEEE Transactions on Power Systems 7(2) (Feb., 1992): 124–132.

[30] R.P. Lippmann. "An Introduction to Computing with Neural Nets." IEEE Acoustics Speech and Signal Processing Magazine 4 (April, 1987): 4–22.

[31] C.N. Lu, H.T. Wu, and S. Vemuri. "Neural Network Based Short Term Load Forecasting." IEEE Transactions on Power Systems 8(1) (Feb., 1993): 336–342.

[32] Ibrahim Moghram and Saifur Rahman. "Analysis and Evaluation of Five Short–Term Load Forecasting Techniques." IEEE Transactions on Power Systems 4(4) (Oct., 1989): 1484–1491.

[33] David S. Moore, and George P. McCabe. Introduction to the Practice of Statistics. New York, New York: W.H. Freeman and Company, 1989.

[34] Terry H. Morlan, Editior. Energy Forecasting. New York, New York: American Society of Civil Engineers, 1985.

[35] NOAA Technical Memorandum. NWS FCST 31 A 20–Year Summary of National Weather Verification Results for Temperature and Precipitation . National Weather Service, Silver Springs, Md., 1986.

[36] Hourly Electric Load values, Omaha, Nebraska, Omaha Public Power District. 1990 – 1992.

[37] Alan Pankratz. Forecasting With Univariate Box–Jenkins Models Concepts And Cases. New York, New York:John Wiley & Sons, 1983.

[38] D.C. Park, M.A. El–Sharkawi, R.J. Marks II, L.E. Atlas and M.J. Damborg. "Electric Load Forecasting Using an Artificial Neural Network." IEEE Transactions on Power Systems 6(2) (May, 1991): 442–449.

[39] T.M. Peng, N.F. Hubele, and G.G. Karady. "Advancement in the Application of Neural Networks for Short–Term Load Forecasting." IEEE Transactions on Power Systems 7(1) (Feb., 1991): 250–257.

[40] T.M. Peng, N.F. Hubele, and G.G. Karady. "An Adaptive Neural Network Approach to One–Week Ahead Load Forecasting." IEEE Transactions on Power Systems 8(3) (Aug., 1993): 1195–1202.

[41] S. Rahman and O. Hazim. "A Generalized Knowledge–Based Short–Term Load–Forecasting Technique." IEEE Transactions on Power Systems 8(2) (May, 1993): 508–514.

[42] Harry G. Stoll. Least–Cost Electric Utility Planning. New York, New York: John Wiley & Sons, 1989.

[43] D.E. Welsh, and G.B. Sheble. "A Review of Artificial Neural Network Applications to Short–Term Power System Load Forecasting." Proceedings of the 1992 North American Power Symposium, Reno, Nevada, 1992.

[44] B. Widrow and M.A. Lehr. "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation." Proceedings of the IEEE 78(9) (Sept., 1990): 1415–1441.

Appendix A. COMPUTER CODES

This chapter contains the computer codes used in this thesis. The ANN code is *nnffbp_1.f.* The data processing codes are *loadread.f* and *loadstrat.f.*

nnffbp_1.f

```
**********************************************************************
*                                                                    *
*                    BPNET2                                          *
*                  Eric Daugherty                                    *
*        Feed Forward, Back Propagation with momentum                *
*                                                                    *
* ic = iteration count for saving best weights                       *
* tt = train(1,2) [1 = new, 2 = saved] or test(3) [once through]     *
* ninpn = # of input nodes                                           *
* noutn = # of output nodes                                          *
* nhidn = # of hidden nodes                                          *
* ntp = # of training patterns                                       *
* lrnrt2 = learning rate in layer 2                                  *
* lrnrt1 = learning rate in layer 1                                  *
* mom = momentum                                                     *
* thold = threashhold                                                *
* gain = gain                                                        *
* jran = random number seed (integer)                                *
* ni = # of iterations (0 for infinite)                              *
* ertng = training criteria                                          *
* x,y,z =  loop counters                                             *
* a,b = dummy variables                                              *
* tp(10000,200) = up to 10000 training patterns w/200 in+out node    *
* wt(2,200,200) = weights                                            *
* ran = random number between 0 and 1                                *
* sum = sum of node outputs*weights                                  *
* dlt(2,200,200) = change in weights for entire tp set for 1 itt.    *
* l = tp counter for batch training                                  *
* av(2,200) = activation or output of a node                         *
* err = the absolute error                                           *
* rms = rms                                                          *
* d(200) = difference in output node from tp                         *
* er = backprop error                                                *
* i = iteration counter                                              *
* rmss = rms check to determine when to save weights                 *
* wts(2,200,200) = the best weights saved                            *
* m = momentum flag                                                  *
* dummwt = dummy weight for use in momentum                          *
* wtm(2,200,200) = weights used for momentum                         *
```

```
* h = save weight counter                                                *
*                                                                        *
*****c***$*********$*********$*********$*********$*********$*********$

      real lrnrt1,mom,ertng,tp(10000,200),wt(2,200,200),ran,sum,
     c     dlt(2,200,200),av(2,200),err,rms,d(200),er,rmss,gain,
     c     wts(2,200,200),dumwt,wtm(2,200,200),lrnrt2,thold

      integer  ic,tt,ninpn,noutn,nhidn,ntp,jran,ni,x,y,z,l,a,b,i,m,h

************************************************************************
* Open up files                                                         *
************************************************************************

      open (12,file='config_1.net',status='old')
      open (13,file='io_1.dat',status='old')
      open (14,file='rwts_1.dat',status='unknown')
      open (16,file='out_1.dat',status='unknown')

************************************************************************
* Read in input file (config_1.net)                                     *
************************************************************************

      read (12,*) ic,tt
      read (12,*) ninpn,noutn,nhidn,ntp
      read (12,*) lrnrt2,lrnrt1,mom,thold,gain,jran
      read (12,*) ni,ertng

      close (12)

************************************************************************
* read in data file (io_1.dat)                                          *
************************************************************************

      do 25 x=1,ntp
          read (13,*) (tp(x,y),y=1,ninpn+noutn)
25    continue

      close (13)

************************************************************************
* initalize weights or read in saved weights (rwts_1.dat)               *
************************************************************************

      a=nhidn
      b=ninpn
      do 50 z=1,2
          do 40 x=1,a
```

```
                do 40 y=1,b
                    if (tt.eq.1) wt(z,x,y)=2.0*ran(jran)–1.0
                    if (tt.eq.2.or.tt.eq.3) read (14,*) wt(z,x,y)
40          continue
            a=noutn
            b=nhidn
50      continue

        close (14)

*************************************************************************
* set the inital parameters to zero                                     *
*************************************************************************

        i=0
        rmss=1000.0
        m=0
        h=0
55       l=0
        err=0.0
        rms=0.0

*************************************************************************
* set the inital change to zero                                         *
*************************************************************************
        a=nhidn
        b=ninpn
        do 70 z=1,2
            do 60 x=1,a
                do 60 y=1,b
                    dlt(z,x,y)=0.0
60              continue
                a=noutn
                b=nhidn
70      continue

************************
* count the # of tp's      *
************************

80      l=l+1

*************************************************************************
*                                                                       *
* Foward activation loop                                                  *
*                                                                       *
*************************************************************************
```

```
        a=nhidn
        b=ninpn
        do 350 z=1,2
            do 300 x=1,a
                sum=0.0
                do 250 y=1,b
                    if (a.eq.nhidn) sum=sum+wt(z,x,y)*tp(l,y)
                    if (a.eq.noutn) sum=sum+wt(z,x,y)*av(z-1,y)
250             continue
                av(z,x)=1/(1+exp(-gain*(sum+thold)))
300         continue
            a=noutn
            b=nhidn
350     continue

*****************************************************************
*                                                               *
* Test loop (out_1.dat)                                           *
*                                                                 *
*****************************************************************

        if (tt.eq.3) then
        do 375 x=1,noutn
            write (16,*) tp(l,ninpn+x), av(2,x),abs(tp(l,ninpn+x)-av(2,x))
375         continue
        endif
        if (l.eq.ntp.and.tt.eq.3) goto 9030
        if (tt.eq.3) goto 80

*****************************************************************
*                                                               *
* Back propigation loop with batch learning                       *
*                                                               *
*****************************************************************

*********************************
* calculating error and rms for tp      *
*********************************

        do 400 z=1,noutn
            d(z)=tp(l,ninpn+z)-av(2,z)
            err=err+abs(d(z))
            rms=rms+d(z)**2
400     continue

*****************
* The Delta Rule    *
*****************
```

```
      do 425 x=1,noutn
          do 425 y=1,nhidn
          dlt(2,x,y)=d(x)*av(1,y)+dlt(2,x,y)
425   continue

*****************************
* The Generalized Delta Rule      *
*****************************

      do 475 y=1,nhidn
          er=0.0
          do 440 z=1,noutn
              er=er+d(z)*wt(2,z,y)
440       continue
          do 455 x=1,ninpn
              dlt(1,y,x)=dlt(1,y,x)+(er*(av(1,y)*(1-av(1,y)))*tp(l,x))
455       continue
475   continue

*************************************************************************
* moving on to the next training pattern                                 *
*************************************************************************

      if (l.lt.ntp) goto 80

*************************************************************************
*                                                                       *
* Check to see if rms is below error criteria and save best weights     *
*                                                                       *
*************************************************************************

      i=i+1
      h=h+1
      rms=(rms/ntp/noutn)**0.5
      if (rms.lt.rmss) then
          a=nhidn
          b=ninpn
          do 550 z=1,2
              do 525 x=1,a
                  do 525 y=1,b
                      wts(z,x,y)=wt(z,x,y)
525               continue
                  a=noutn
                  b=nhidn
550       continue
          rmss=rms
      endif
```

```
      if (rms.le.ertng) goto 610

**********************************************************************
*                                                                    *
* Set the new weights                                                *
*                                                                    *
**********************************************************************

      a=noutn
      b=nhidn
      do 600 z=2,1,-1
          do 575 x=1,a
              do 575 y=1,b
                  dumwt=wt(z,x,y)
                  if (z.eq.2) wt(z,x,y)=wt(z,x,y)+dlt(z,x,y)*lrnrt2/ntp
                  if (z.eq.1) wt(z,x,y)=wt(z,x,y)+dlt(z,x,y)*lrnrt1/ntp
                  if(m.eq.1) wt(z,x,y)=wt(z,x,y)+mom*(dumwt-wtm(z,x,y))
                  wtm(z,x,y)=dumwt
575       continue
          a=nhidn
      b=ninpn
600   continue

      m=1

**********************************************************************
* Write best weights to a file (swts_11.dat, swts_12.dat)            *
**********************************************************************

610   if (h.eq.ic.or.i.eq.ni.or.rms.le.ertng) then
      open (15,file='swts_11.dat',status='unknown')
      open (17,file='swts_12.dat',status='unknown')
      open (18,file='netstat_1.dat',status='unknown')
      a=nhidn
      b=ninpn
      do 650 z=1,2
          do 625 x=1,a
              do 625 y=1,b
                  write (15,*) wts(z,x,y)
                  write (17,*) wts(z,x,y)
625       continue
          a=noutn
          b=nhidn
650   continue
      write (18,*) 'Best RMS = ', rmss
      write (15,*) 'Best RMS = ', rmss
      write (17,*) 'Best RMS = ', rmss
```

```
      write (18,*) 'Curent RMS = ', rms
      write (15,*) 'Curent RMS = ', rms
      write (17,*) 'Curent RMS = ', rms
      write (18,*) '# of itterations = ', i
      write (15,*) '# of itterations = ', i
      write (17,*) '# of itterations = ', i
      close (15)
      close (17)
      close (18)
      h=0
      endif

      if (rms.le.ertng) goto 9000

*************************************************************************
* Move on to another itteration                                         *
*************************************************************************

      if (i.eq.ni) goto 9010
      goto 55

*************************************************************************
*                                                                       *
* The end                                                               *
*                                                                       *
*************************************************************************

9000  print *, ' The network has converged '
9010  print *, 'RMS = ', rms,' Error = ', err
      print *, 'The number of itterations is ', i
9030  print *, ' Hasta La Vista.....Baby'

      close (16)

      end
```

loadread.f

```
*************************************************************************
*                                                                       *
*            loadread.f                                                 *
*       read data and put it into form                                  *
*                                                                       *
*************************************************************************
```

```
      real el(30000),rsum,noise,value,
c           w1(30000),w2(30000),w3(30000),w4(30000),
c           w5(30000),w6(30000),w7(30000),
c           sum1,sum2,sum3,sum4,wa(6,14),
c           sum(4,14),w(6,6,14)

      integer i,j,k,h,y,d,acf,x,m,wi,su,lc1,lc2,a1,a2,b1,b2,
c           hol,d1,d2,d3,d4,d5,d6,d7,jran,z,di

      common jran

      open (12,status='unknown',file='eload90-92_space.dat')
      open (13,status='unknown',file='weather90-92_comma.dat')
      open (14,status='unknown',file='winter_train.dat')
      open (15,status='unknown',file='summer_train.dat')
      open (16,status='unknown',file='winter_test.dat')
      open (17,status='unknown',file='summer_test.dat')

**********************************************************************
* Read in electric load files                                        *
**********************************************************************

      do 100 i=0,26303,24
          read (12,*) (el(i+j),j=1,24)
100   continue

      print *, 'Finished reading in electric data'

**********************************************************************
* Normalize electric load data                                       *
**********************************************************************

      do 150 i=1,26304
          el(i)=0.8*((el(i)-441.0)/(1652.0-441.0))+0.1
150   continue

      print *, 'Finished normalizing electric data'

**********************************************************************
* Read weather files                                                 *
**********************************************************************

      do 200 i=1,26304
          read (13,*) w1(i),w2(i),w3(i),w4(i),w5(i),w6(i),w7(i)
200   continue
```

```
**********************************************************************
* Normalize weather data                                             *
**********************************************************************

      do 250 i=1,26304
*********************************
* Air Temperature L90/S91       *
*********************************
      w1(i)=0.8*((w1(i)-(-15.16))/(107.56-(-15.16)))+0.1
*********************************
* Relatitive Humidity L91/S92   *
*********************************
      w2(i)=0.8*((w2(i)-12.95)/(118.07-12.95))+0.1
*********************************
* Soil Temperature L90/S91      *
*********************************
      w3(i)=0.8*((w3(i)-(-3.89))/(102.38-(-3.89)))+0.1
*********************************
* Wind Speed L91/S90            *
*********************************
      w4(i)=0.8*((w4(i)-0.0)/(40.45-0.0))+0.1
*********************************
* Wind Direction                *
*********************************
*     w5(i)=0.8*((w5(i)-sw5)/(lw5-sw5))+0.1

*********************************
* Solar Radiation L91/S90       *
*********************************
      w6(i)=0.8*((w6(i)-0.0)/(962.0-0.0))+0.1
*********************************
* Precipation L90/S90           *
*********************************
      w7(i)=0.8*((w7(i)-0.0)/(2.05-0.0))+0.1
250   continue

**********************************************************************
* Writing the file for nnffbp.f                                      *
**********************************************************************

      print *, 'OK, Starting to write file'
      print *, 'Enter the first hour Predicted'
      read *, h

      print *, 'Enter the starting day'
      print *, 'Monday = (1)'
      print *, 'Tuesday = (2)'
      print *, 'Wednesday = (3)'
```

```
print *, 'Thursday = (4)'
print *, 'Friday = (5)'
print *, 'Saturday = (6)'
print *, 'Sunday = (7)'
read *, d

print *, 'Enter the starting'
print *, 'First(1) , Second(2), Tenth(10), ... etc.'
read *, y

print *, 'Do you want to skip the averaging calcs?'
print *, '(1) yes'
print *, '(2) no'
read *, acf
print *, 'Enter a interger random # seed'
read *, jran

lc1=(26304-((y-1)*24+h))
lc2=(17520-((y-1)*24+h))

do 400 i=1,lc1

    hol=0
    d1=0
    d2=0
    d3=0
    d4=0
    d5=0
    d6=0
    d7=0

    if (d.gt.7) d=1

    if (h.gt.24) then
        h=1
        d=d+1
        y=y+1
    endif

    if (d.eq.1) d1=1
    if (d.eq.2) d2=1
    if (d.eq.3) d3=1
    if (d.eq.4) d4=1
    if (d.eq.5) d5=1
    if (d.eq.6) d6=1
    if (d.eq.7) d7=1

    if (y.eq.1.or.y.eq.148.or.y.eq.187.or.y.eq.246.or.
```

```
c               y.eq.326.or.y.eq.327.or.y.eq.359.or.y.eq.366.or.
c               y.eq.512.or.y.eq.551.or.y.eq.610.or.y.eq.697.or.
c               y.eq.698.or.y.eq.724.or.y.eq.731.or.y.eq.876.or.
c               y.eq.914.or.y.eq.981.or.y.eq.1061.or.y.eq.1062.or.
c               y.eq.1090) hol=1

****************************************
* Calculate daily weather averages     *
****************************************

        if (acf.eq.1) goto 390

        do 330 j=1,14
            if (j.eq.1) then
                a1=0
                a2=23
            else if (j.eq.2) then
                a1=24
                a2=47
            else if (j.eq.3) then
                a1=48
                a2=71
            else if (j.eq.4) then
                a1=72
                a2=95
            else if (j.eq.5) then
                a1=96
                a2=119
            else if (j.eq.6) then
                a1=120
                a2=143
            else if (j.eq.7) then
                a1=144
                a2=167
            else if (j.eq.8) then
                a1=168
                a2=191
                noise=0.03
            else if (j.eq.9) then
                a1=192
                a2=215
                noise=0.04
            else if (j.eq.10) then
                a1=216
                a2=239
                noise=0.05
            else if (j.eq.11) then
                a1=240
```

```
            a2=263
            noise=0.07
        else if (j.eq.12) then
            a1=264
            a2=287
            noise=0.10
        else if (j.eq.13) then
            a1=288
            a2=311
            noise=0.13
        else if (j.eq.14) then
            a1=312
            a2=335
            noise=0.17
        endif

        sum1=0.0
        sum2=0.0
        sum3=0.0
        sum4=0.0

        do 310 k=(i+a1),(i+a2)
            sum1=sum1+w1(k)
            sum2=sum2+w2(k)
            sum3=sum3+w4(k)
            sum4=sum4+w6(k)
310     continue

        if (j.le.7) then
            wa(1,j)=sum1/24
            wa(2,j)=sum2/24
            wa(4,j)=sum3/24
            wa(6,j)=sum4/24
        end if

        if (j.ge.8) then
            call pfore((sum1/24),noise,value)
            wa(1,j)=value
            call pfore((sum2/24),noise,value)
            wa(2,j)=value
             call pfore((sum3/24),noise,value)
            wa(4,j)=value
            call pfore((sum4/24),noise,value)
            wa(6,j)=value
        end if

330     continue
```

```
      do 380 j=1,6

         if (j.eq.1) then
              b1=0
              b2=3
         else if (j.eq.2) then
              b1=4
              b2=7
         else if (j.eq.3) then
              b1=8
              b2=11
         else if (j.eq.4) then
              b1=12
              b2=15
         else if (j.eq.5) then
              b1=16
              b2=19
         else if (j.eq.6) then
              b1=20
              b2=23
         endif

         do 335 z=1,14
              sum(1,z)=0.0
              sum(2,z)=0.0
              sum(3,z)=0.0
              sum(4,z)=0.0
335      continue

         do 340 k=1,14
              if (k.eq.1) di=0
              if (k.eq.2) di=24
              if (k.eq.3) di=48
              if (k.eq.4) di=72
              if (k.eq.5) di=96
              if (k.eq.6) di=120
              if (k.eq.7) di=144
              if (k.eq.8) di=168
              if (k.eq.9) di=192
              if (k.eq.10) di=216
              if (k.eq.11) di=240
              if (k.eq.12) di=264
              if (k.eq.13) di=288
              if (k.eq.14) di=312

         do 340 z=(i+b1),(i+b2)
              sum(1,k)=sum(1,k)+w1(z+di)
              sum(2,k)=sum(2,k)+w2(z+di)
```

```
                sum(3,k)=sum(3,k)+w4(z+di)
                sum(4,k)=sum(4,k)+w6(z+di)
340         continue

            do 350 z=1,14
                if (z.le.7) then
                    w(1,j,z)=sum(1,z)/4
                    w(2,j,z)=sum(1,z)/4
                     w(4,j,z)=sum(1,z)/4
                    w(6,j,z)=sum(1,z)/4
                endif

                if (z.eq.8) noise=0.03
                if (z.eq.9) noise=0.04
                if (z.eq.10) noise=0.05
                if (z.eq.11) noise=0.07
                if (z.eq.12) noise=0.10
                if (z.eq.13) noise=0.13
                if (z.eq.14) noise=0.17

                if (z.ge.8) then
                    call pfore((sum(1,z)/4),noise,value)
                    w(1,j,z)=value
                    call pfore((sum(2,z)/4),noise,value)
                    w(2,j,z)=value
                    call pfore((sum(3,z)/4),noise,value)
                    w(4,j,z)=value
                    call pfore((sum(3,z)/4),noise,value)
                    w(6,j,z)=value
                endif
350         continue
380     continue

************************************************
* Write to individual files                     *
************************************************

390     if (i.le.lc2) then
            wi=14
            su=15
        else if (i.gt.lc2) then
            wi=16
            su=17
        endif
```

```
**********************
* Winter               *
**********************

      if (y.lt.137.or.y.gt.257.and.y.lt.502.or.y.gt.622.and.
     c    y.lt.868.or.y.gt.988) write (wi,*)
     c    el(i+144),el(i+145),el(i+146),el(i+147),el(i+148),
     c    el(i+149),el(i+150),el(i+151),el(i+152),el(i+153),
     c    el(i+154),el(i+155),el(i+156),el(i+157),el(i+158),
     c    el(i+159),el(i+160),el(i+161),el(i+162),el(i+163),
     c    el(i+164),el(i+165),el(i+166),el(i+167),
     c    wa(1,7),wa(1,8),wa(1,9),wa(1,10),
     c    wa(1,11),wa(1,12),wa(1,13),wa(1,14),
     c    w(1,1,7),w(1,2,7),w(1,3,7),w(1,4,7),w(1,5,7),w(1,6,7),
     c    w(1,1,14),w(1,2,14),w(1,3,14),
     c    w(1,4,14),w(1,5,14),w(1,6,14),
     c    wa(4,7),wa(4,8),wa(4,9),wa(4,10),
     c    wa(4,11),wa(4,12),wa(4,13),wa(4,14),
     c    w(4,1,7),w(4,2,7),w(4,3,7),w(4,4,7),w(4,5,7),w(4,6,7),
     c    w(4,1,14),w(4,2,14),w(4,3,14),
     c    w(4,4,14),w(4,5,14),w(4,6,14),
     c    d1,d2,d3,d4,d5,d6,d7,
     c    hol,
     c    el(i+312),el(i+313),el(i+314),el(i+315),el(i+316),
     c    el(i+317),el(i+318),el(i+319),el(i+320),el(i+321),
     c    el(i+322),el(i+323),el(i+324),el(i+325),el(i+326),
     c    el(i+327),el(i+328),el(i+329),el(i+330),el(i+331),
     c    el(i+332),el(i+333),el(i+334),el(i+335)

**********************
* Summer               *
**********************

      if (y.gt.136.and.y.lt.258.or.y.gt.501.and.y.lt.623.or.
     c    y.gt.867.and.y.lt.989) write (su,*)
     c    el(i+144),el(i+145),el(i+146),el(i+147),el(i+148),
     c    el(i+149),el(i+150),el(i+151),el(i+152),el(i+153),
     c    el(i+154),el(i+155),el(i+156),el(i+157),el(i+158),
     c    el(i+159),el(i+160),el(i+161),el(i+162),el(i+163),
     c    el(i+164),el(i+165),el(i+166),el(i+167),
     c    wa(1,7),wa(1,8),wa(1,9),wa(1,10),
     c    wa(1,11),wa(1,12),wa(1,13),wa(1,14),
     c    w(1,1,7),w(1,2,7),w(1,3,7),w(1,4,7),w(1,5,7),w(1,6,7),
     c    w(1,1,14),w(1,2,14),w(1,3,14),
     c    w(1,4,14),w(1,5,14),w(1,6,14),
     c    wa(2,7),wa(2,8),wa(2,9),wa(2,10),
     c    wa(2,11),wa(2,12),wa(2,13),wa(2,14),
     c    w(2,1,7),w(2,2,7),w(2,3,7),w(2,4,7),w(2,5,7),w(2,6,7),
```

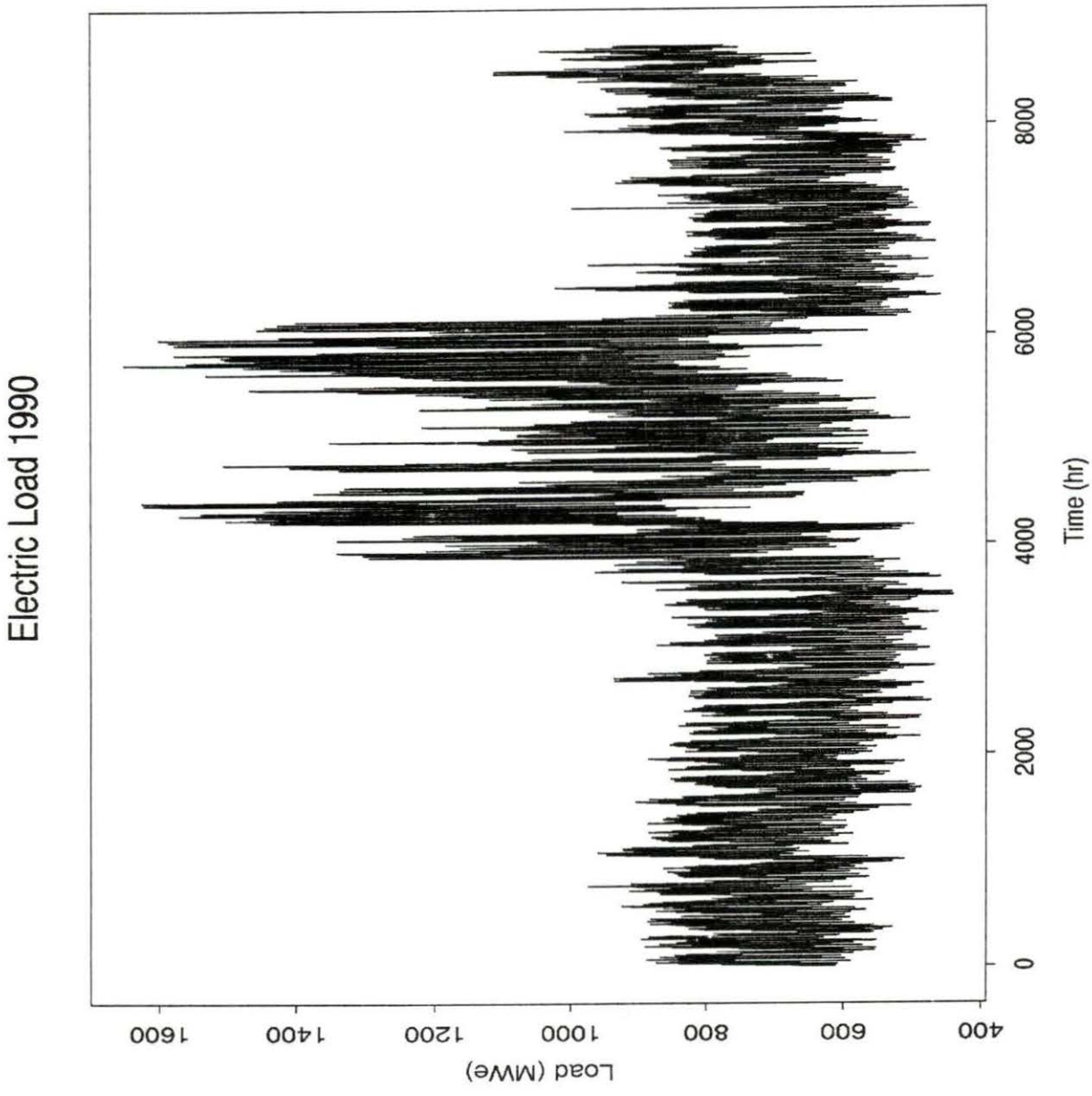
```
c           w(2,1,14),w(2,2,14),w(2,3,14),
c           w(2,4,14),w(2,5,14),w(2,6,14),
c           wa(6,7),wa(6,8),wa(6,9),wa(6,10),
c           wa(6,11),wa(6,12),wa(6,13),wa(6,14),
c           w(6,1,7),w(6,2,7),w(6,3,7),w(6,4,7),w(6,5,7),w(6,6,7),
c           w(6,1,14),w(6,2,14),w(6,3,14),
c           w(6,4,14),w(6,5,14),w(6,6,14),
c           d1,d2,d3,d4,d5,d6,d7,
c           hol,
c           el(i+312),el(i+313),el(i+314),el(i+315),el(i+316),
c           el(i+317),el(i+318),el(i+319),el(i+320),el(i+321),
c           el(i+322),el(i+323),el(i+324),el(i+325),el(i+326),
c           el(i+327),el(i+328),el(i+329),el(i+330),el(i+331),
c           el(i+332),el(i+333),el(i+334),el(i+335)

        h=h+1

400     continue

        close (12)
        close (13)
        close (14)
        close (15)
        close (16)
        close (17)

        end

**************************************************************************
*                                                                        *
* This Subroutine adds noise to actual weather to simulate               *
* forecasted weather                                                     *
*                                                                        *
**************************************************************************

        Subroutine pfore(avg,noise,value)

        real avg,noise,value,rsum,ran

        common jran

        integer k,jran     rsum=0.0
        do 100 k=1,12
            rsum=rsum+(ran(jran))
100     continue
        value=avg+noise*avg*((rsum-6)/12)
        end
```

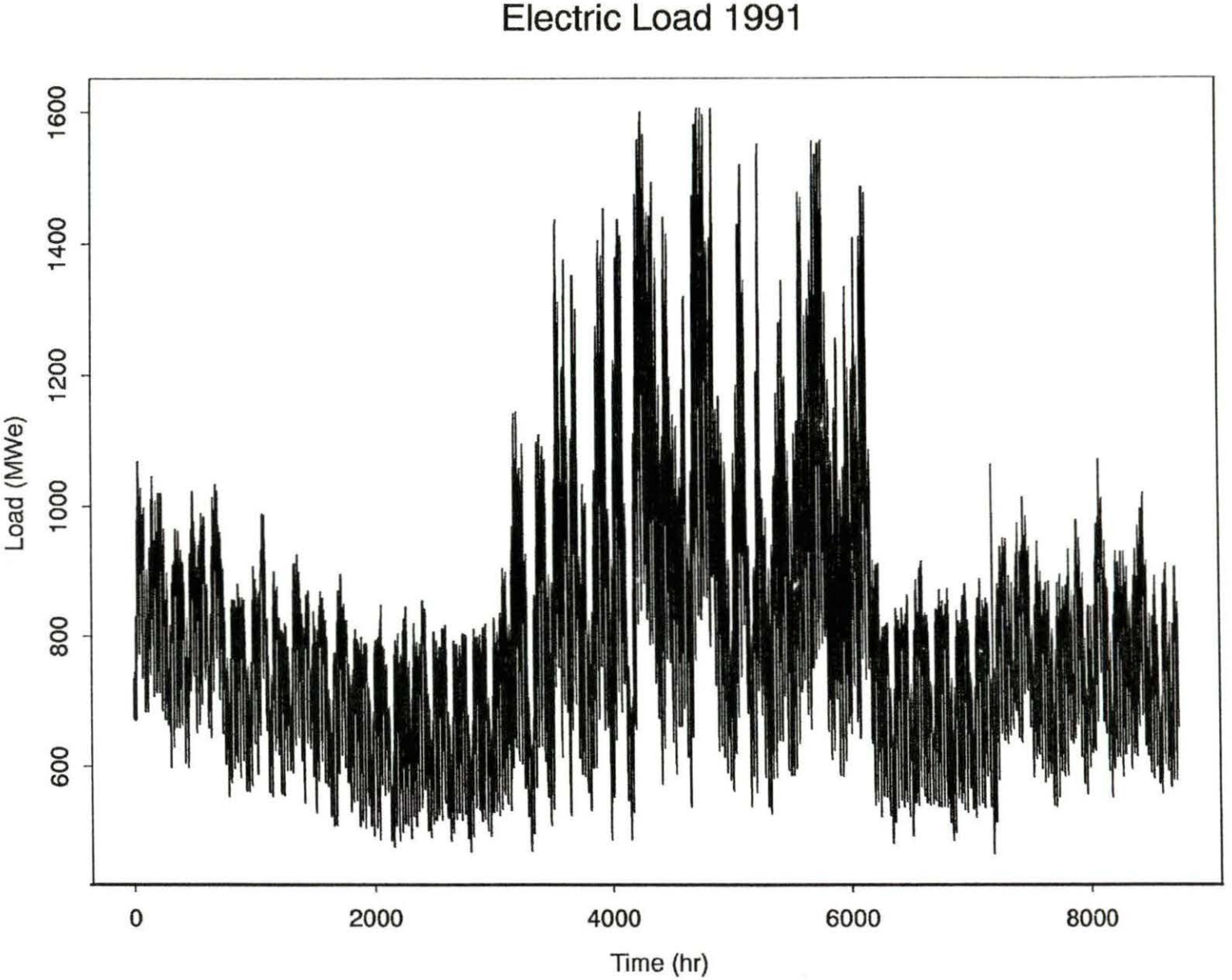
loadstrat.f

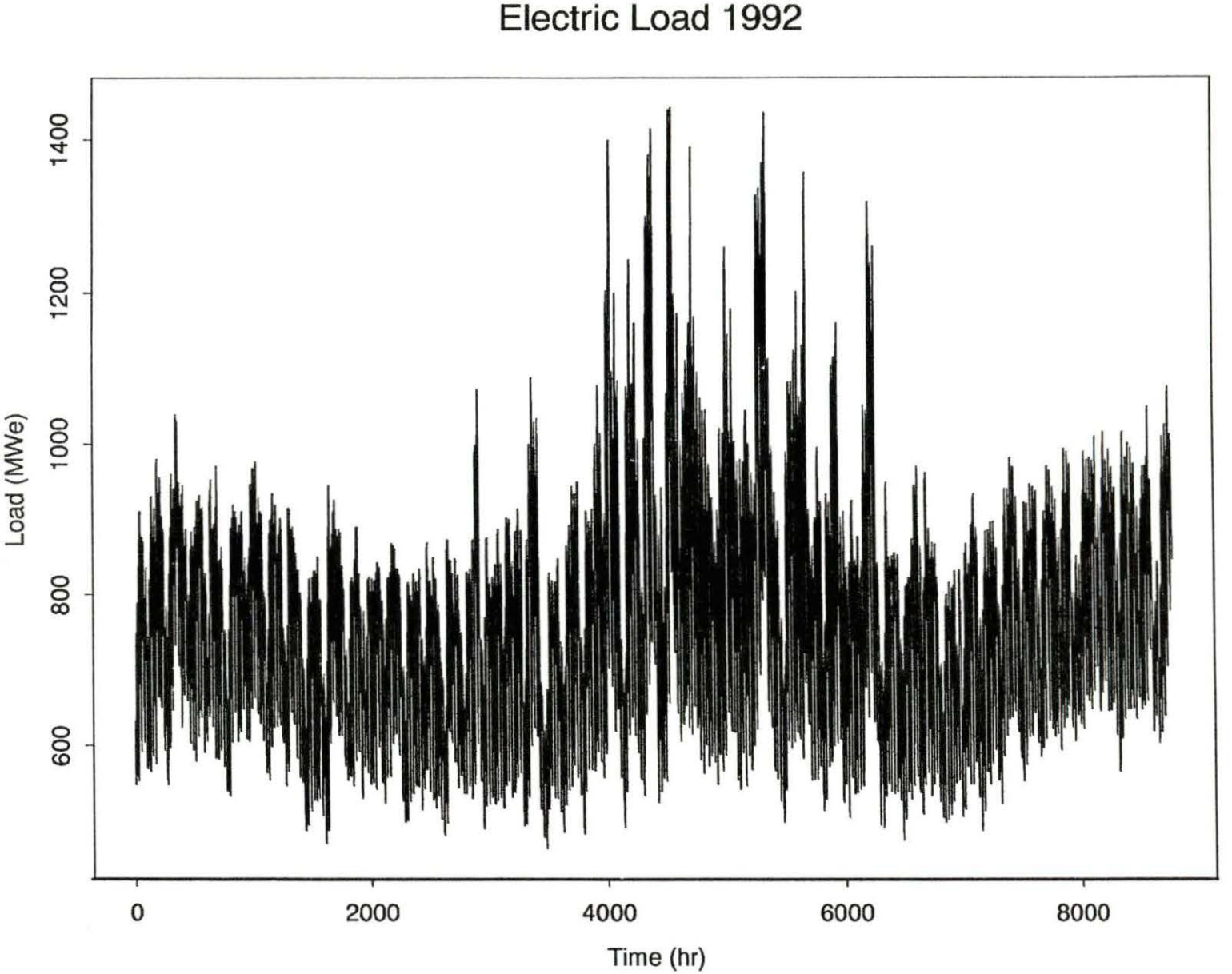
```
****************************************
* make io file smaller     real dum(15000,200)     *
****************************************


      integer i,j,k,nrp,s,np,x

      open (1,status='unknown',file='summer_test.dat')
      open (2,status='unknown',file='winter_test.dat')
      open (3,status='unknown',file='load_short.dat')

      j=0

      print *, 'Summer(1) or winter(2)?'
      read *, x
      print *, 'Enter the # of raw patterns'
      read *, nrp
      print *, 'Enter the # for statification'
      read *, s
      print *, 'Enter the # inputs/outputs'
      read *, np

      do 100 i=1,nrp
          read (x,*) (dum(i,k),k=1,np)
100   continue

      do 200 i=1,nrp
          j=j+1
          if (j.eq.1) write (3,*) (dum(i,k),k=1,np)
          if (j.eq.s) j=0
200   continue

      close (1)
      close (2)
      close (3)

      end
```

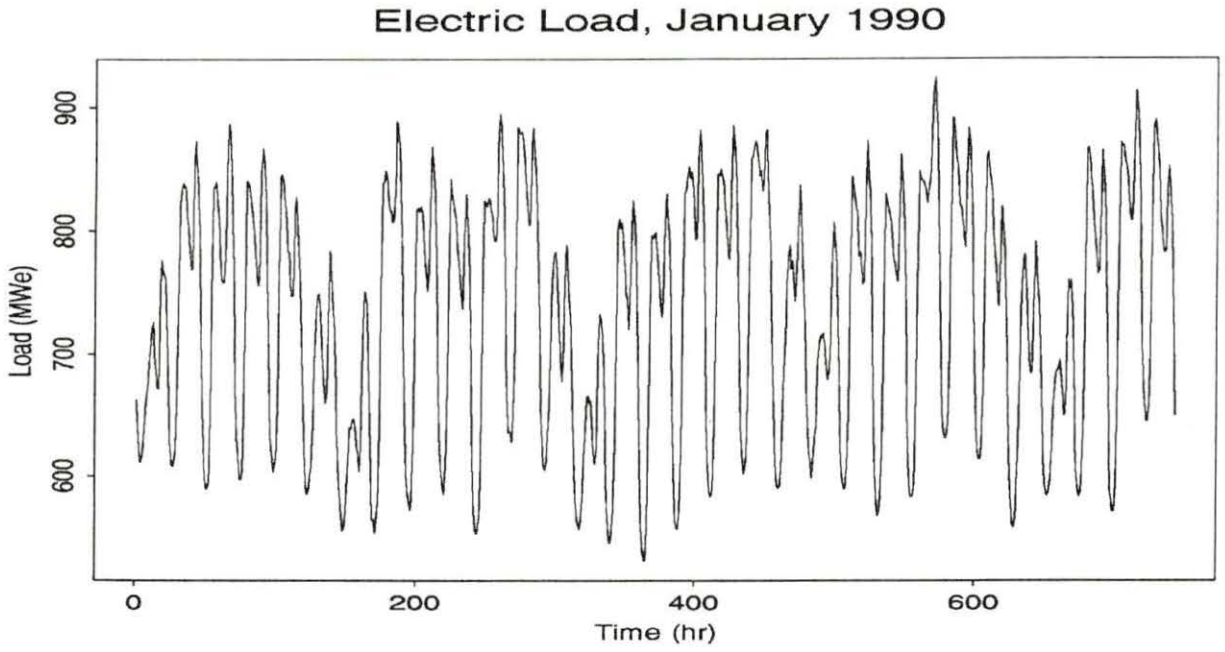
Appendix B. ELECTRIC LOAD CURVES

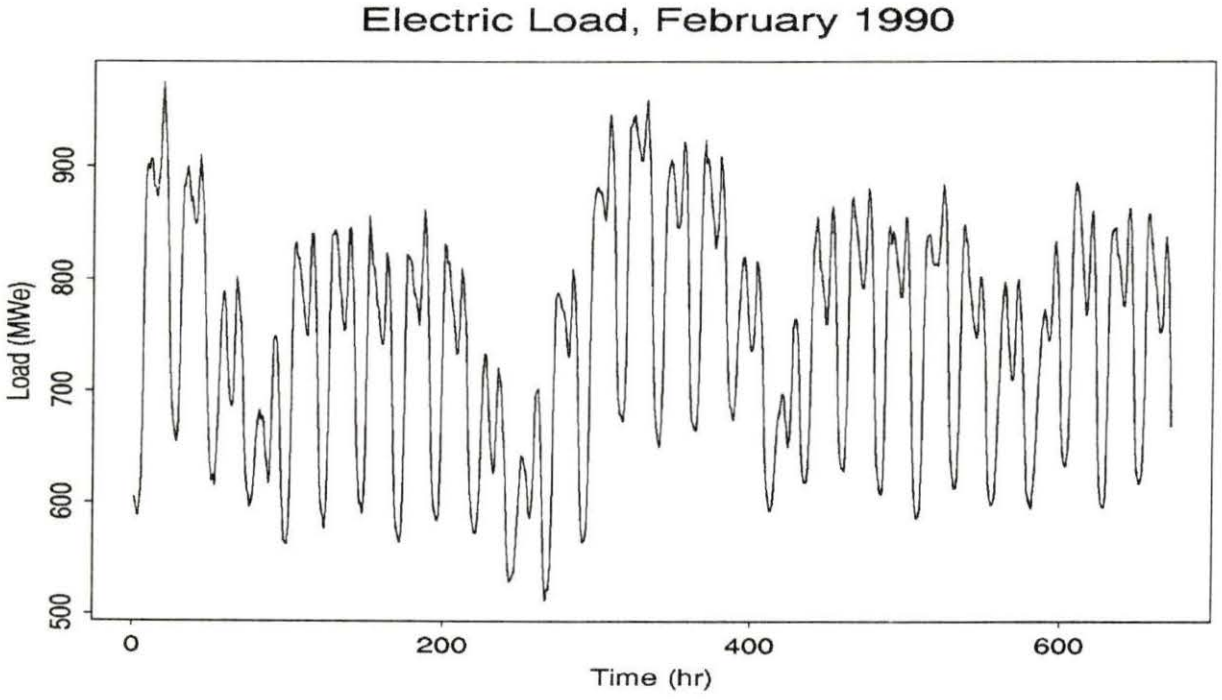
The following section illustrates the electric load data which is plotted by years and by months.

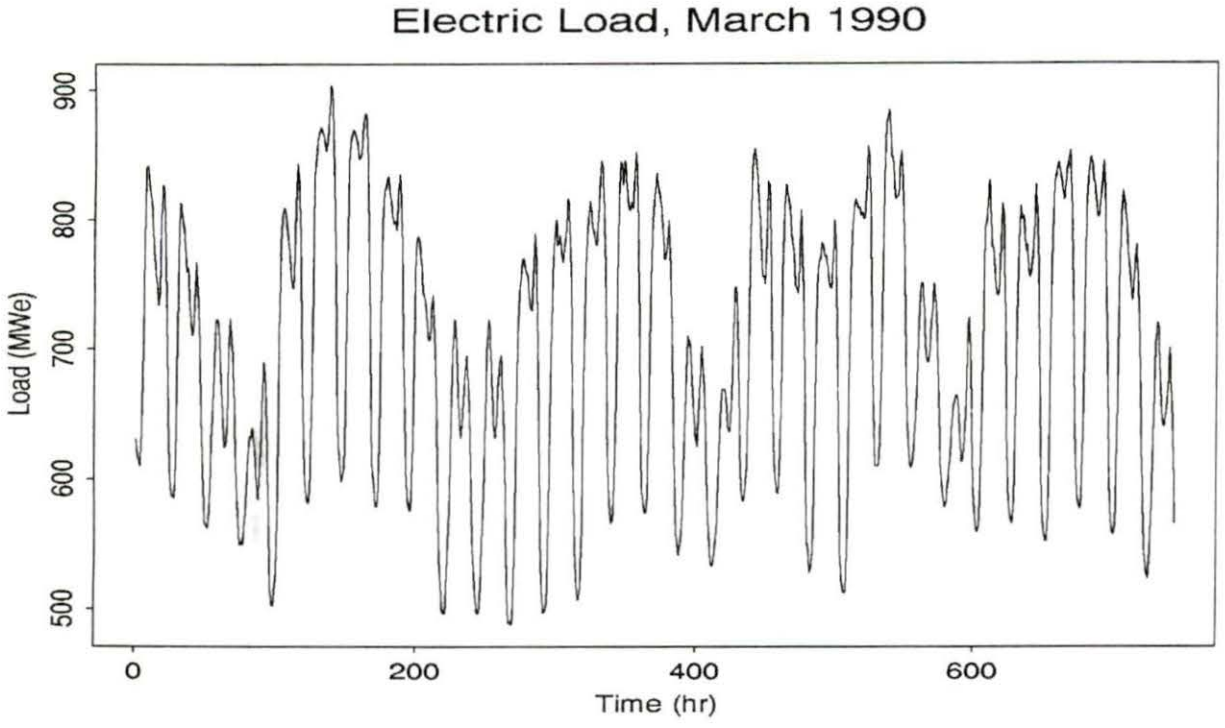


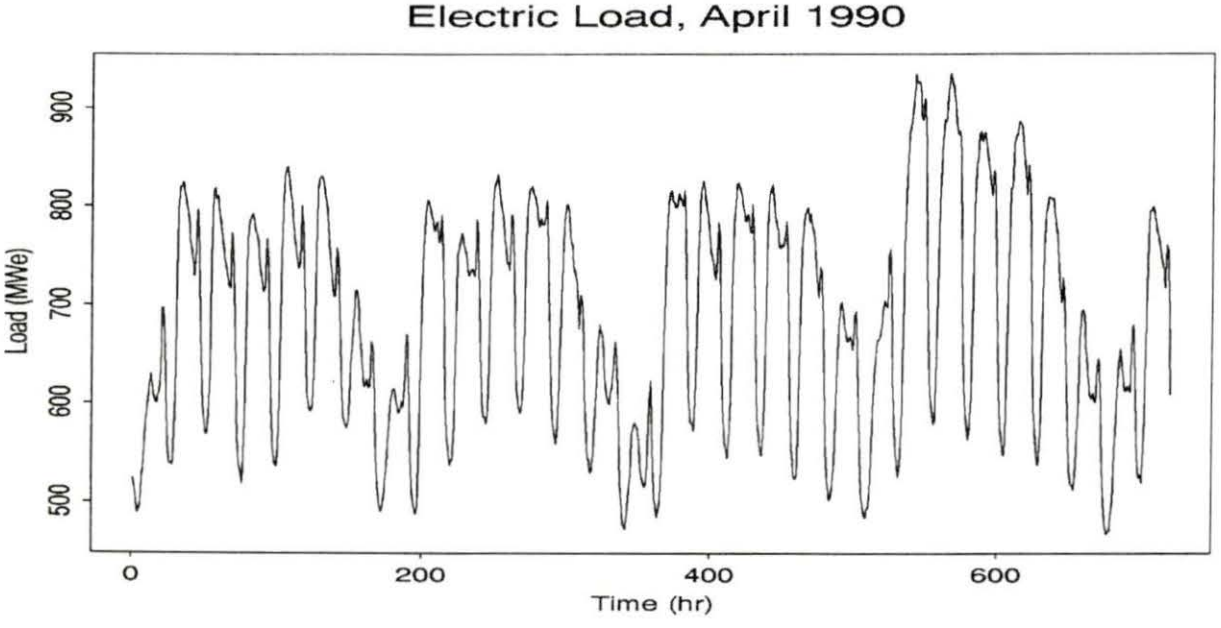

Electric Load 1991
Load (MWe)
600
800
1000
1200
1400
1600
0
2000
4000
6000
8000
Time (hr)

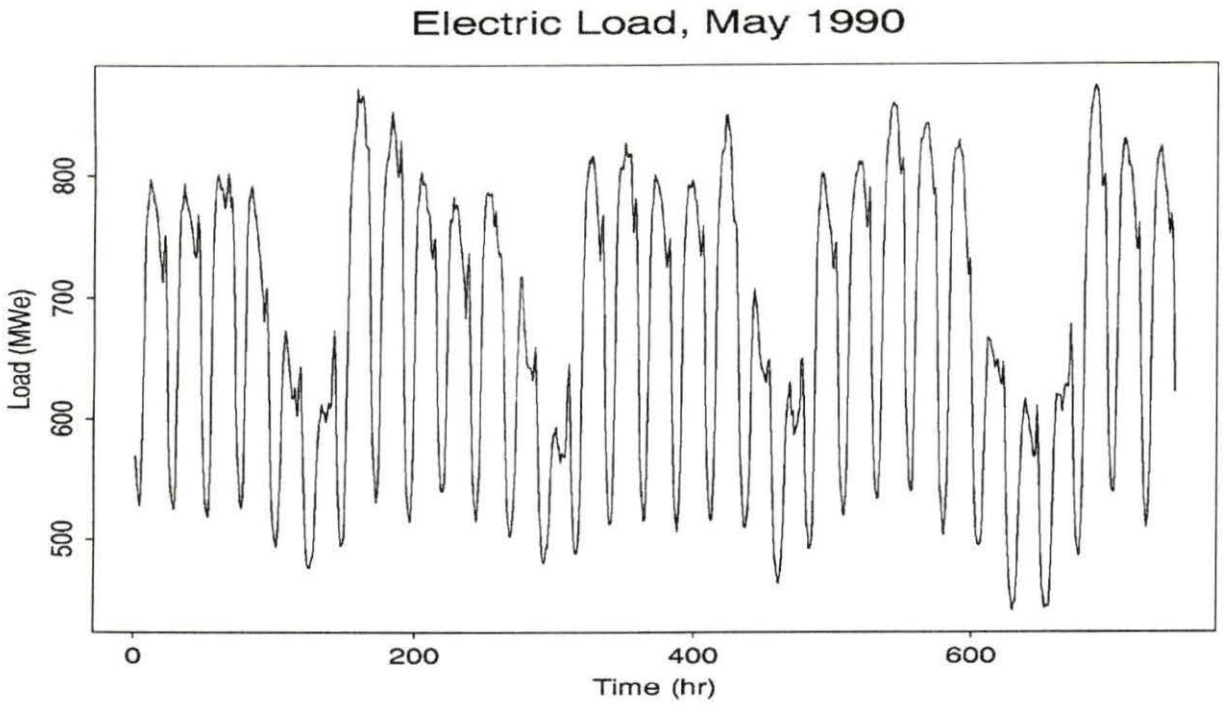

Electric Load 1992
Load (MWe)
600
800
1000
1200
1400
0
2000
4000
6000
8000
Time (hr)

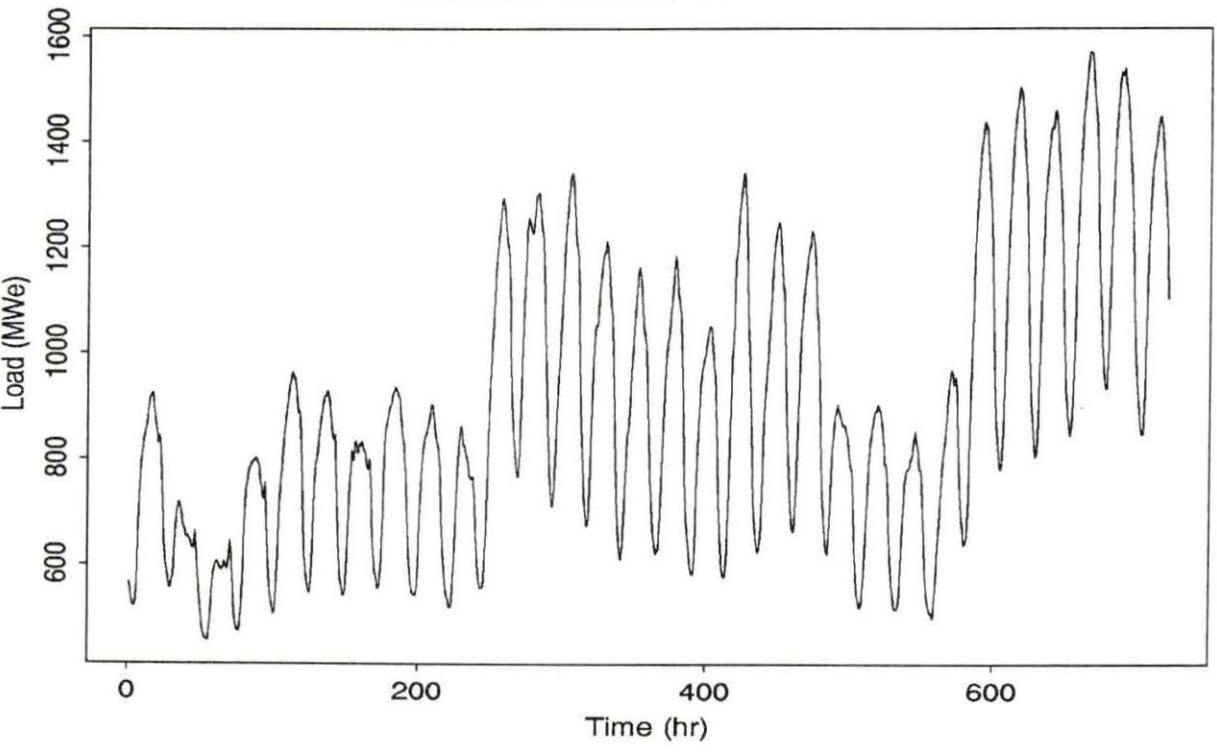

Electric Load, January 1990
Load (MWe)
600
700
800
900
0
200
400
600
Time (hr)

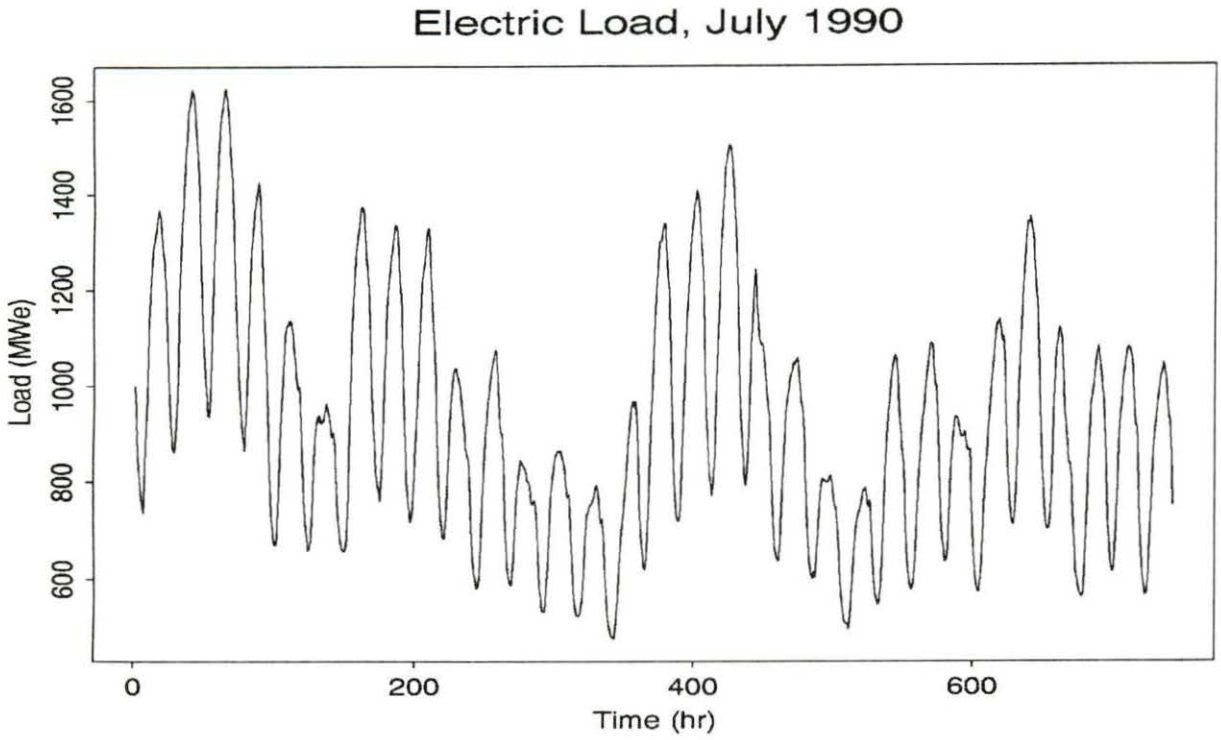

Electric Load, February 1990
Load (MWe)
500
600
700
800
900
0
200
400
600
Time (hr)

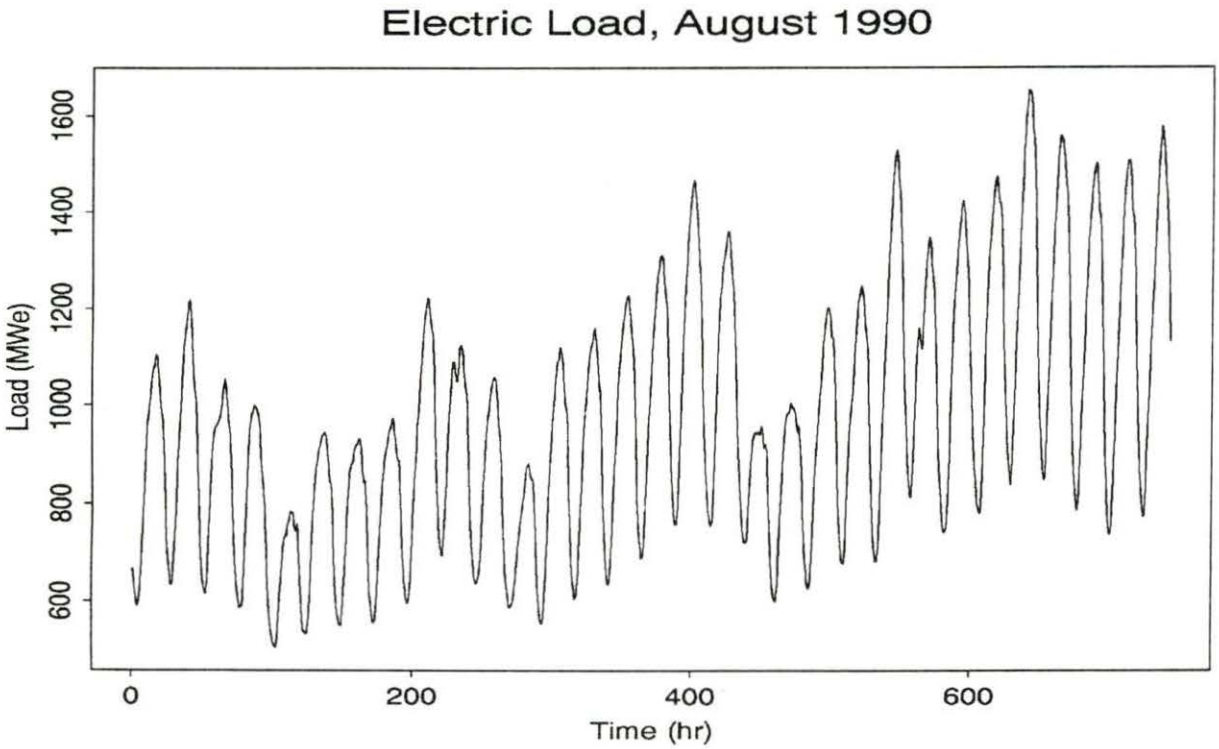

Electric Load, March 1990
Load (MWe)
500
600
700
800
900
0
200
400
600
Time (hr)

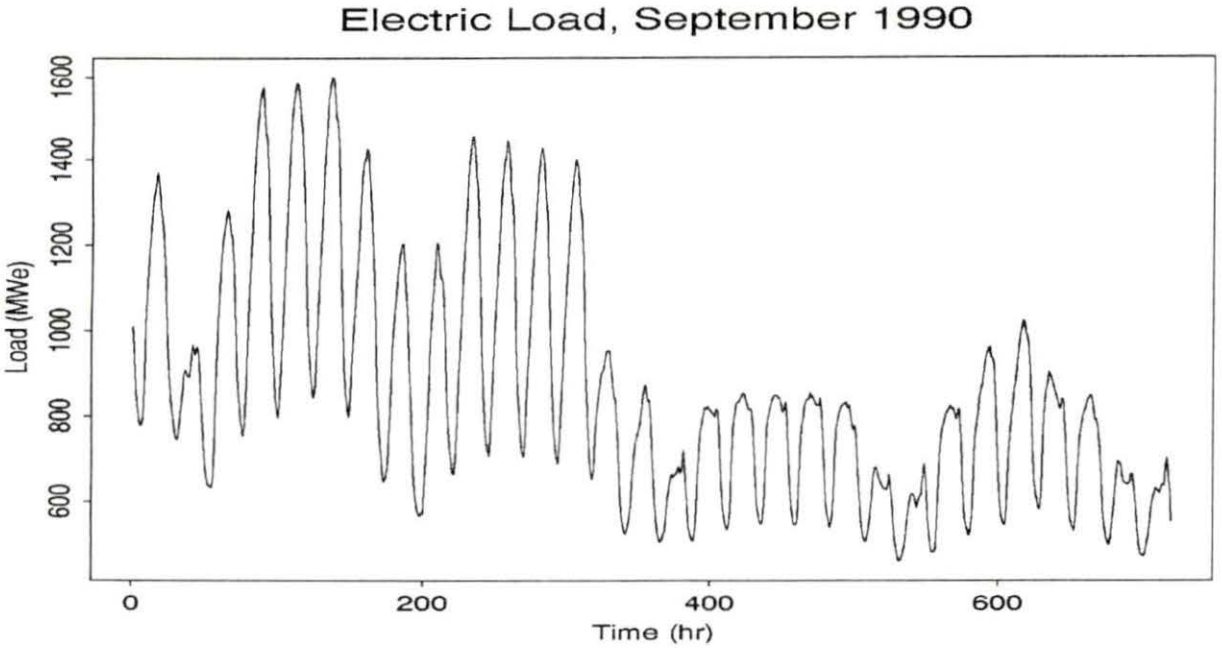

Electric Load, April 1990
Load (MWe)
500
600
700
800
900
0
200
400
600
Time (hr)

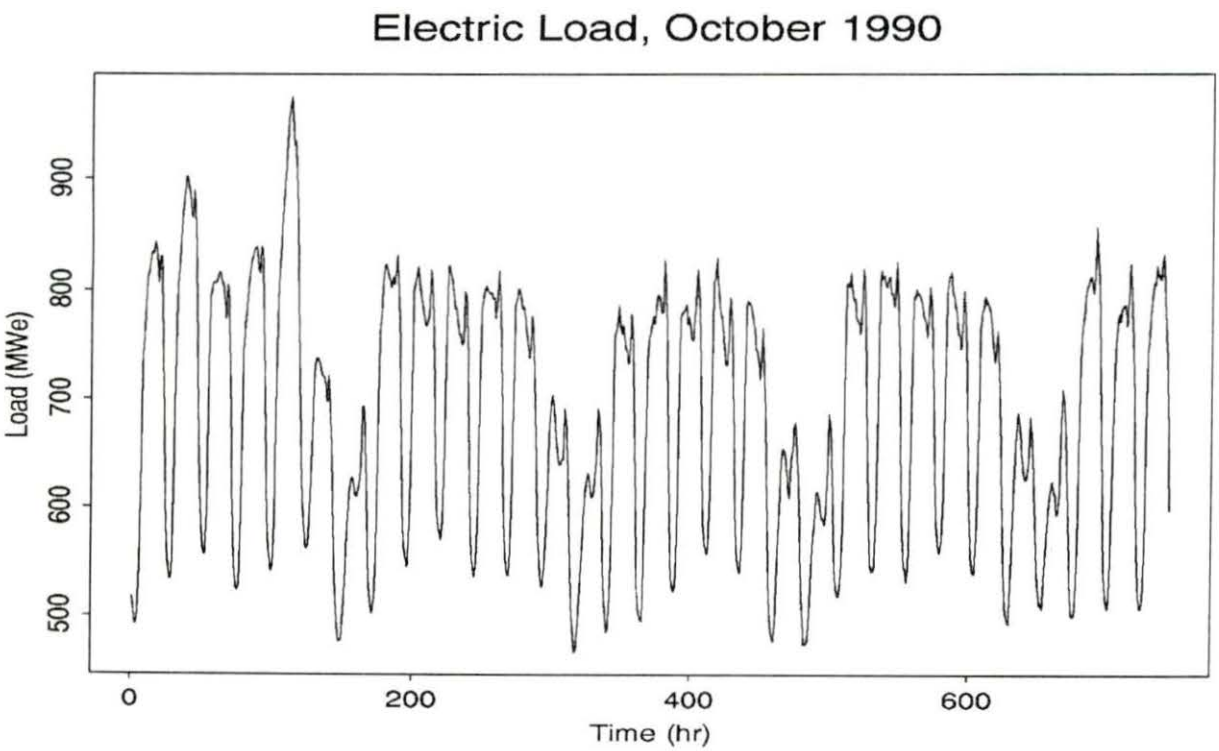

Electric Load, May 1990
Load (MWe)
500
600
700
800
0
200
400
600
Time (hr)

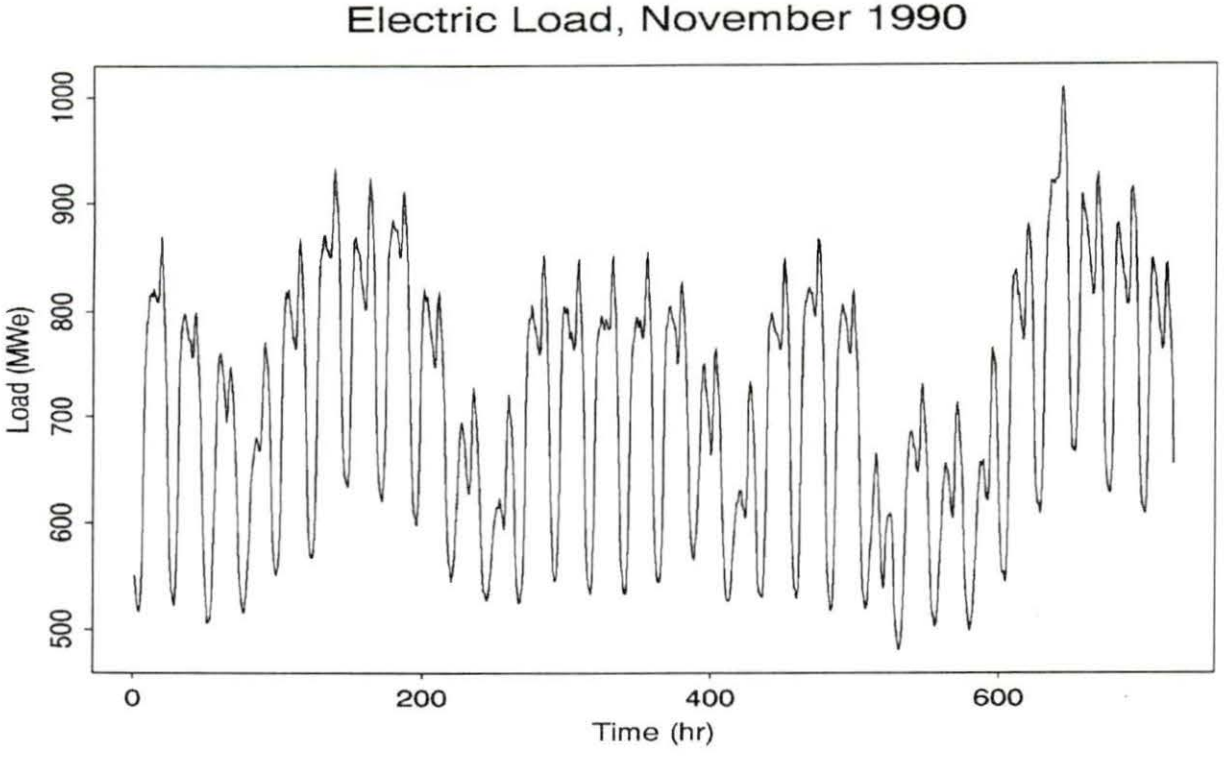

Electric Load, June 1990
Load (MWe)
600
800
1000
1200
1400
1600
0
200
400
600
Time (hr)

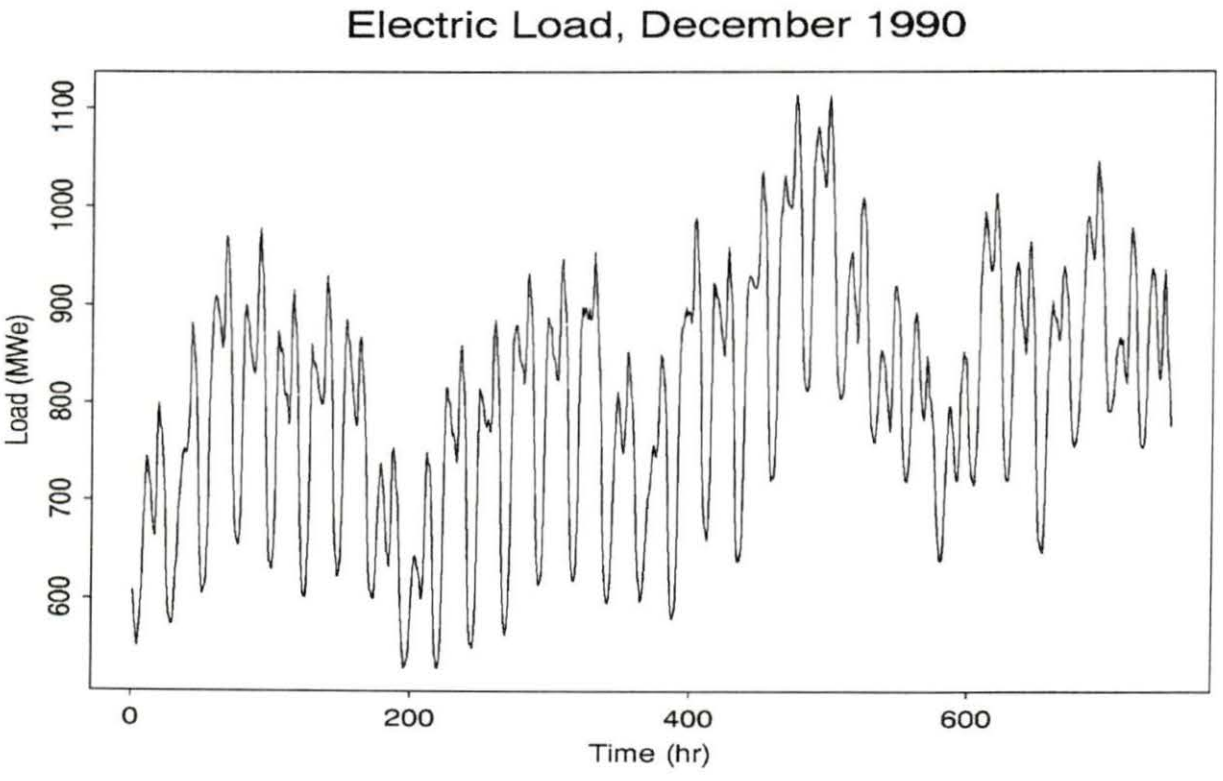

Electric Load, July 1990
Load (MWe)
600
800
1000
1200
1400
1600
0
200
400
600
Time (hr)

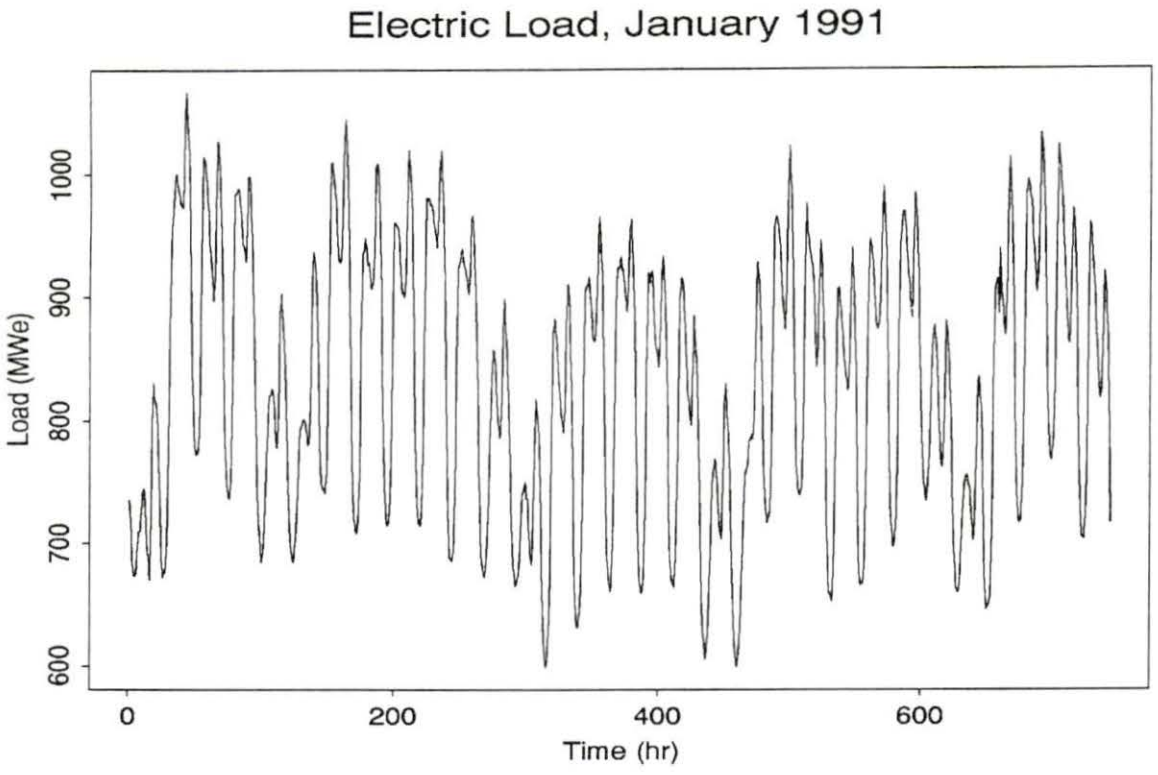

Electric Load, August 1990
Load (MWe)
600
800
1000
1200
1400
1600
0
200
400
600
Time (hr)

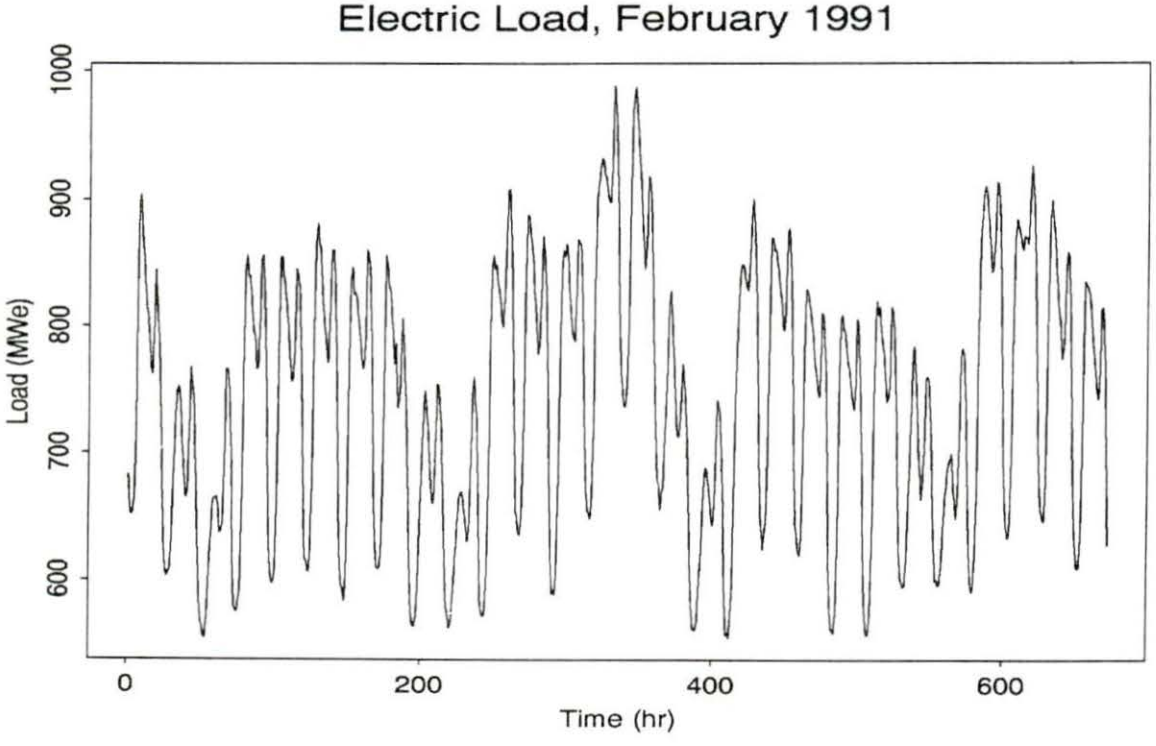

Electric Load, September 1990
Load (MWe)
600
800
1000
1200
1400
1600
0
200
400
600
Time (hr)

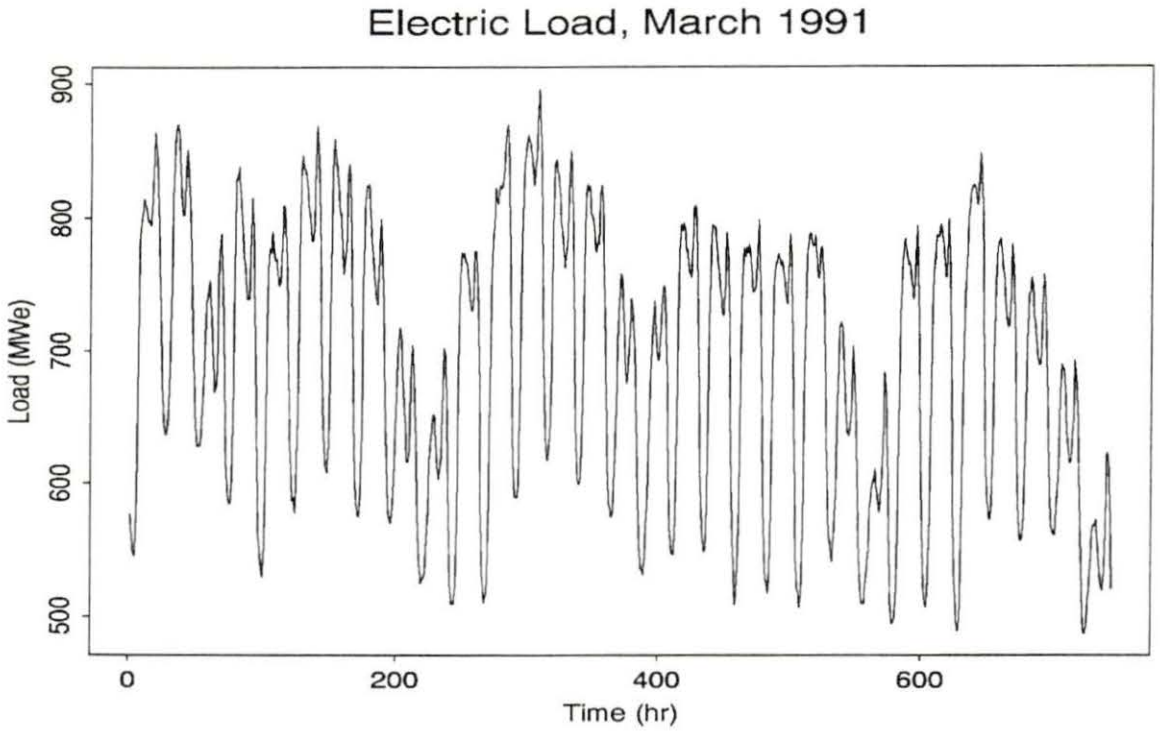

Electric Load, October 1990
Load (MWe)
500
600
700
800
900
0
200
400
600
Time (hr)

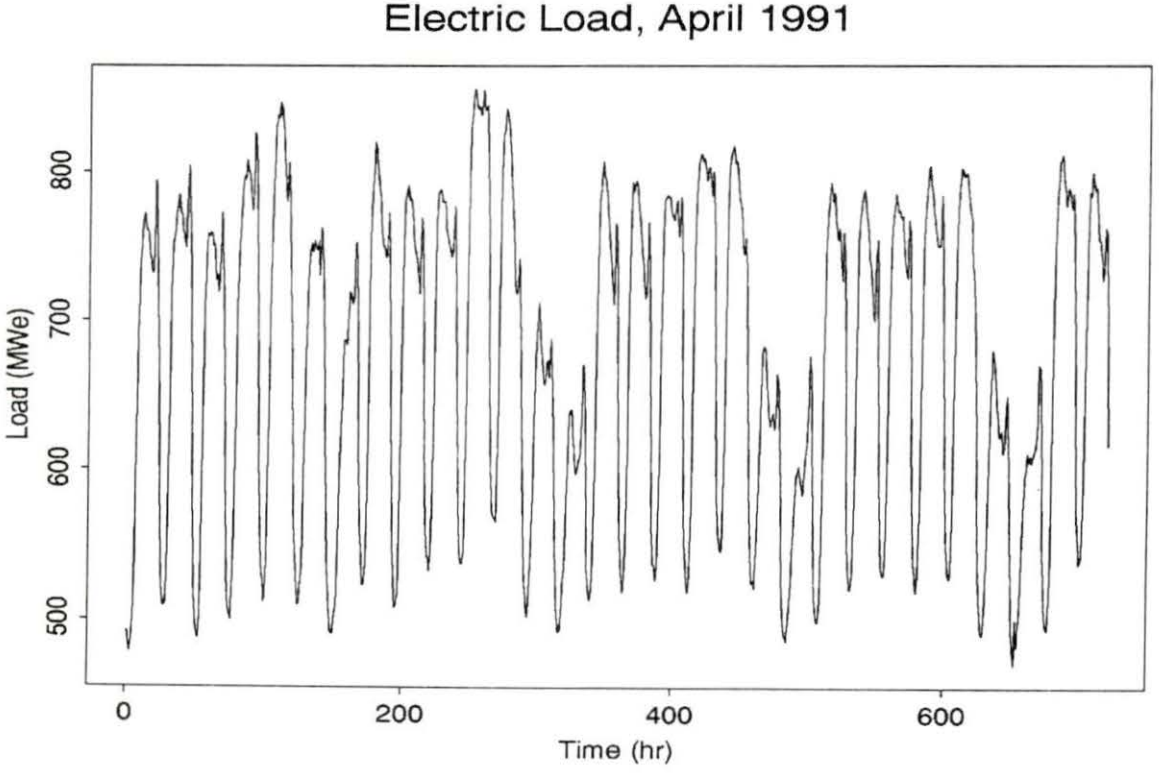

Electric Load, November 1990
Load (MWe)
500
600
700
800
900
1000
0
200
400
600
Time (hr)

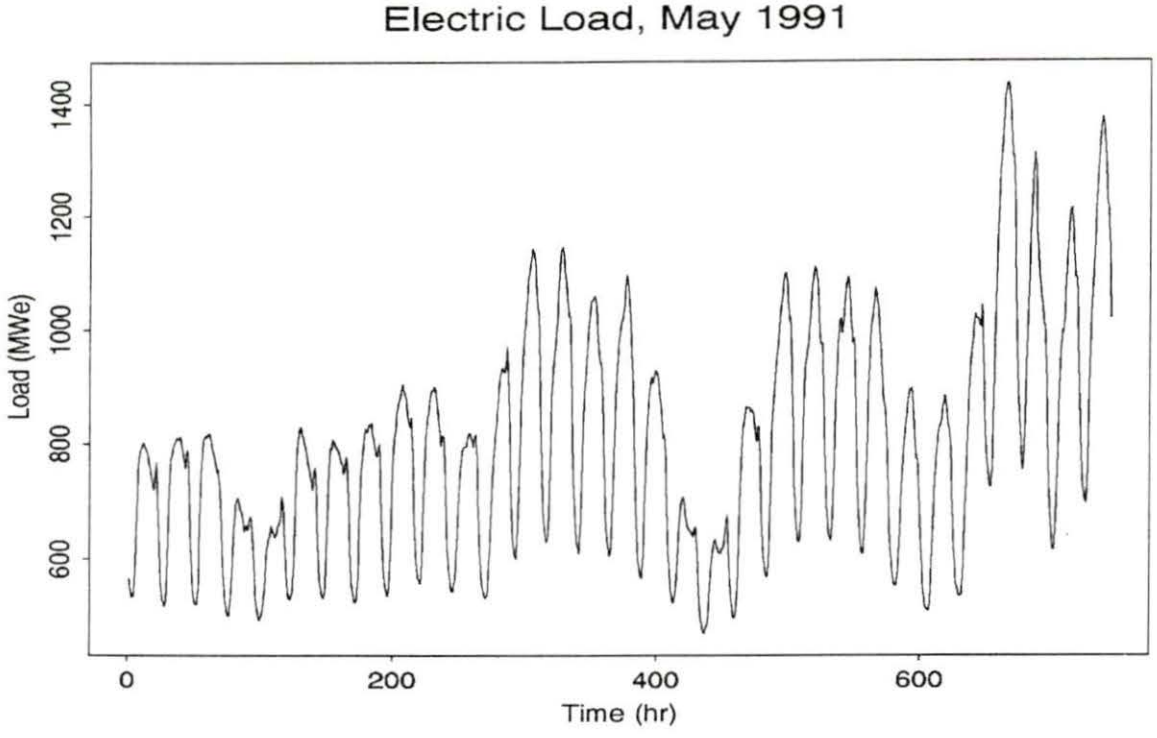

Electric Load, December 1990
Load (MWe)
600
700
800
900
1000
1100
0
200
400
600
Time (hr)

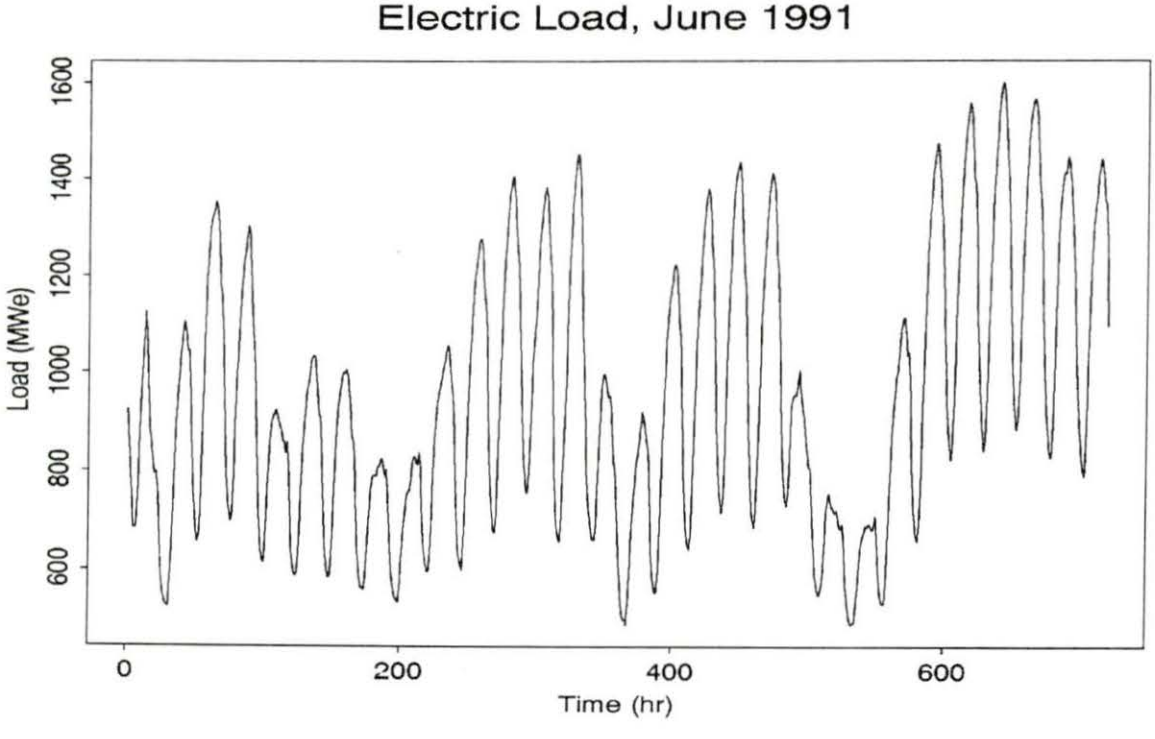

Electric Load, January 1991
Load (MWe)
600
700
800
900
1000
0
200
400
600
Time (hr)

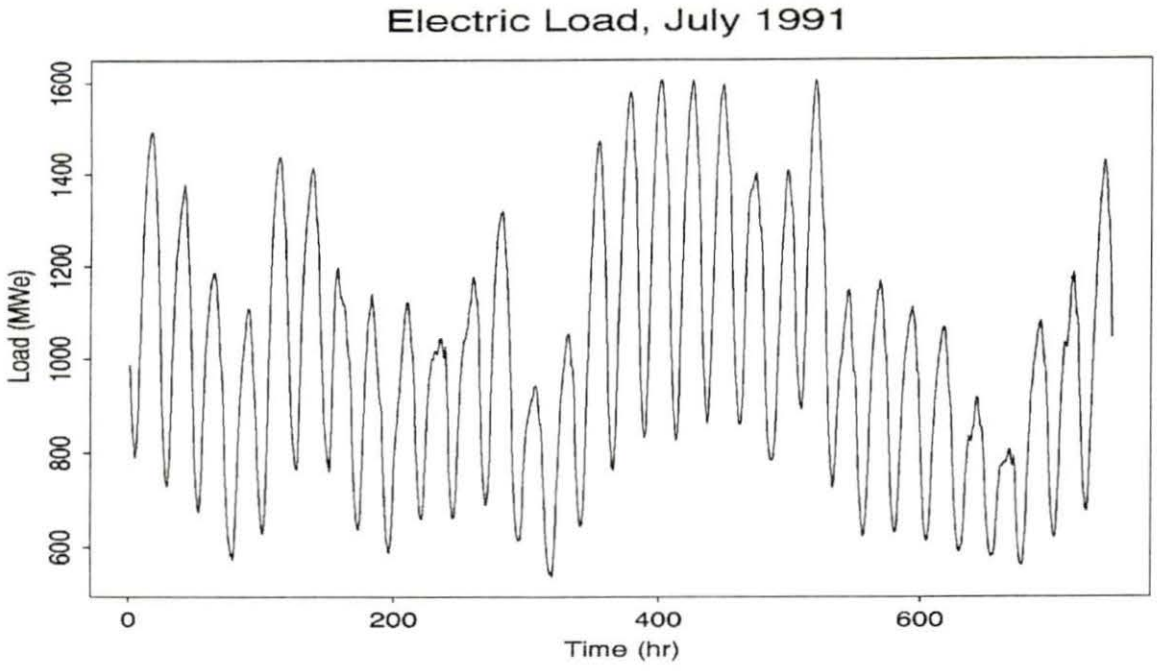

Electric Load, February 1991
Load (MWe)
600
700
800
900
1000
0
200
400
600
Time (hr)

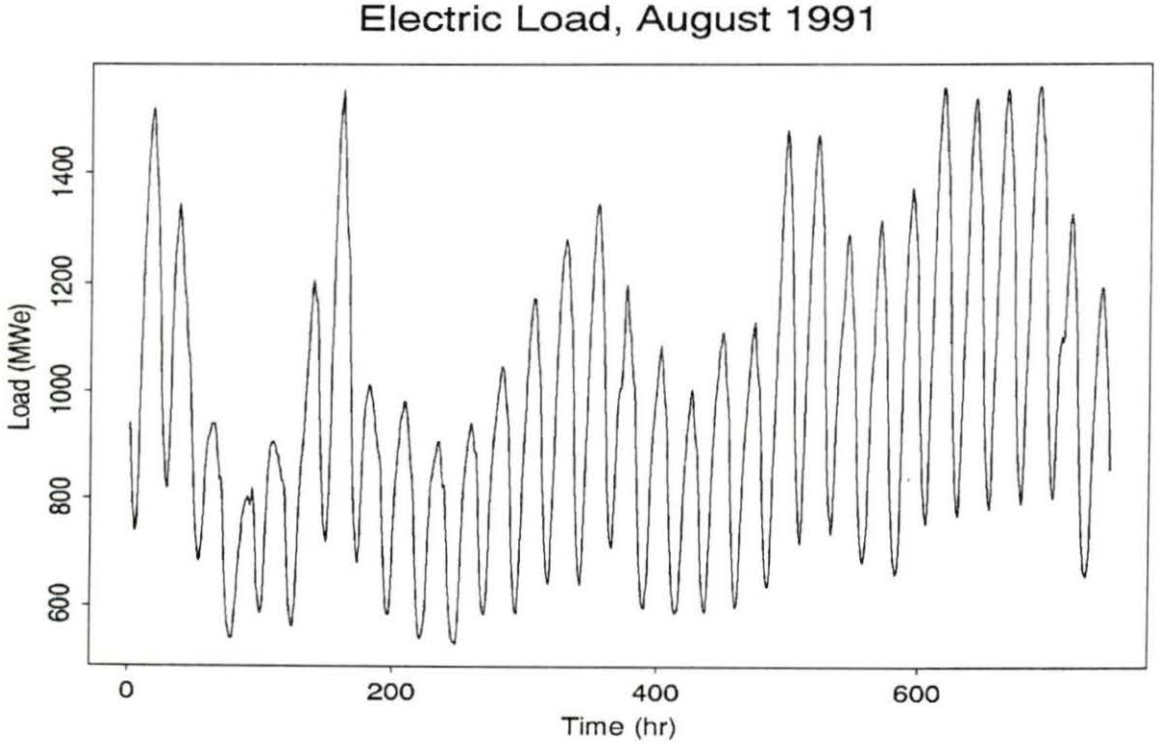

Electric Load, March 1991
Load (MWe)
500
600
700
800
900
0
200
400
600
Time (hr)

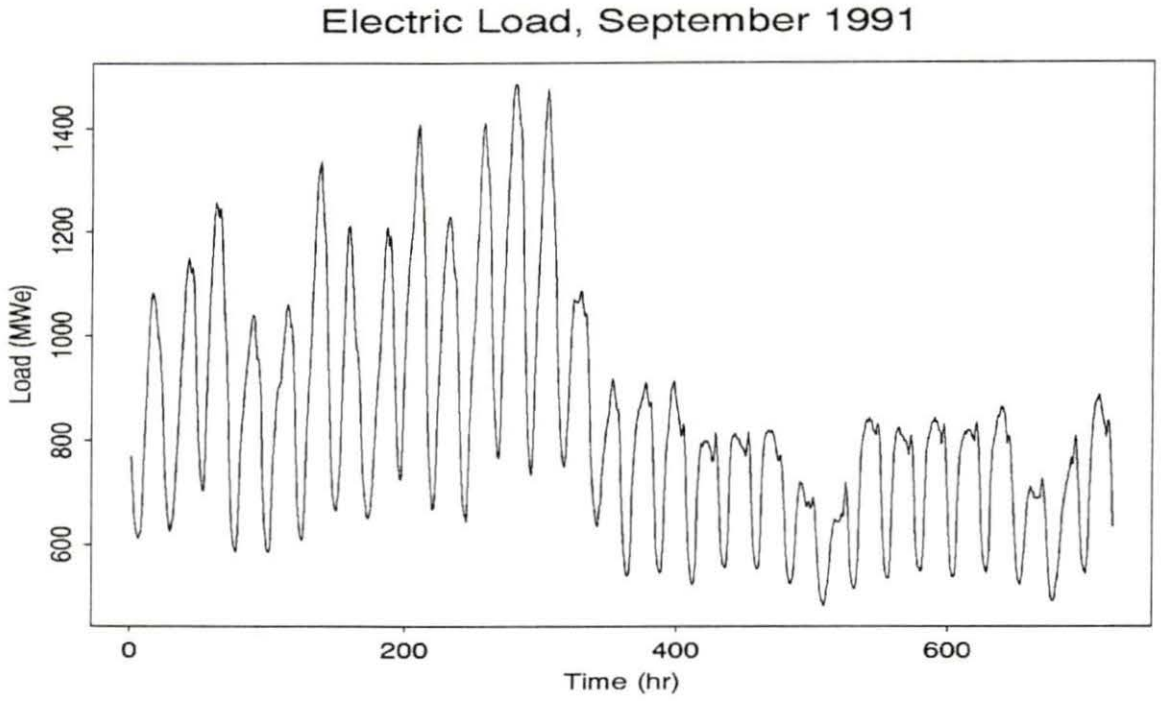

Electric Load, April 1991
Load (MWe)
500
600
700
800
0
200
400
600
Time (hr)

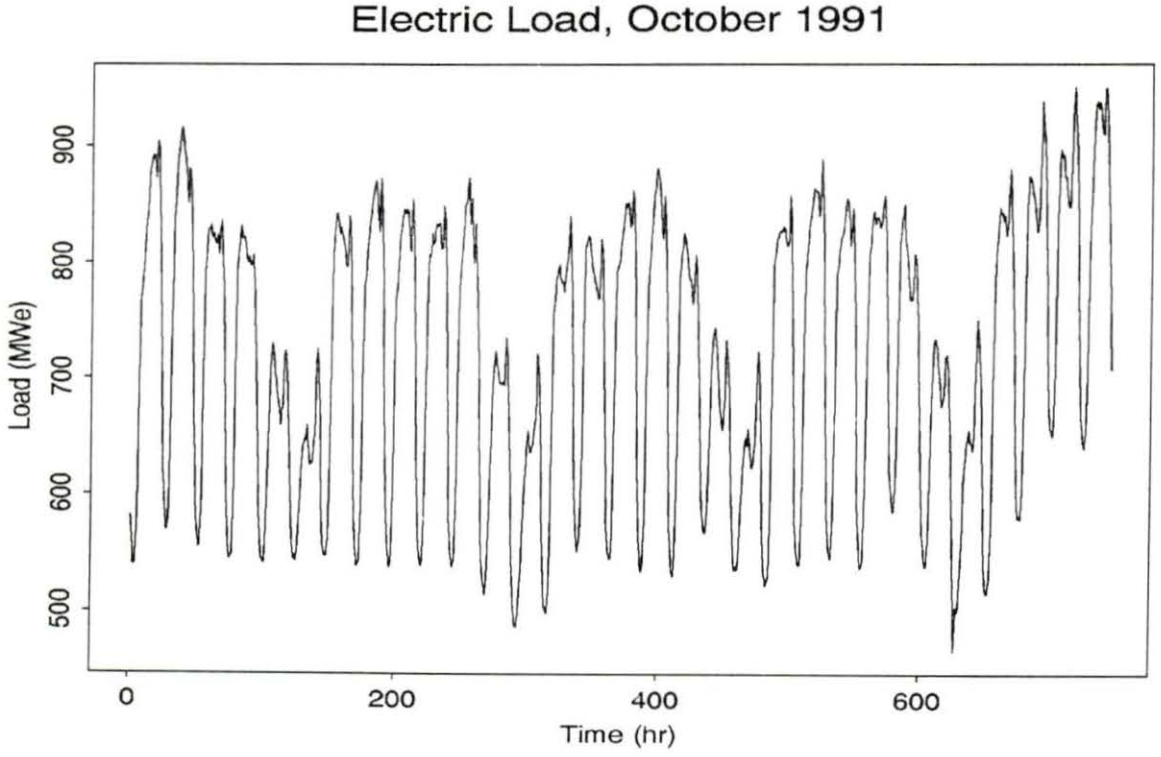

Electric Load, May 1991
Load (MWe)
600
800
1000
1200
1400
0
200
400
600
Time (hr)

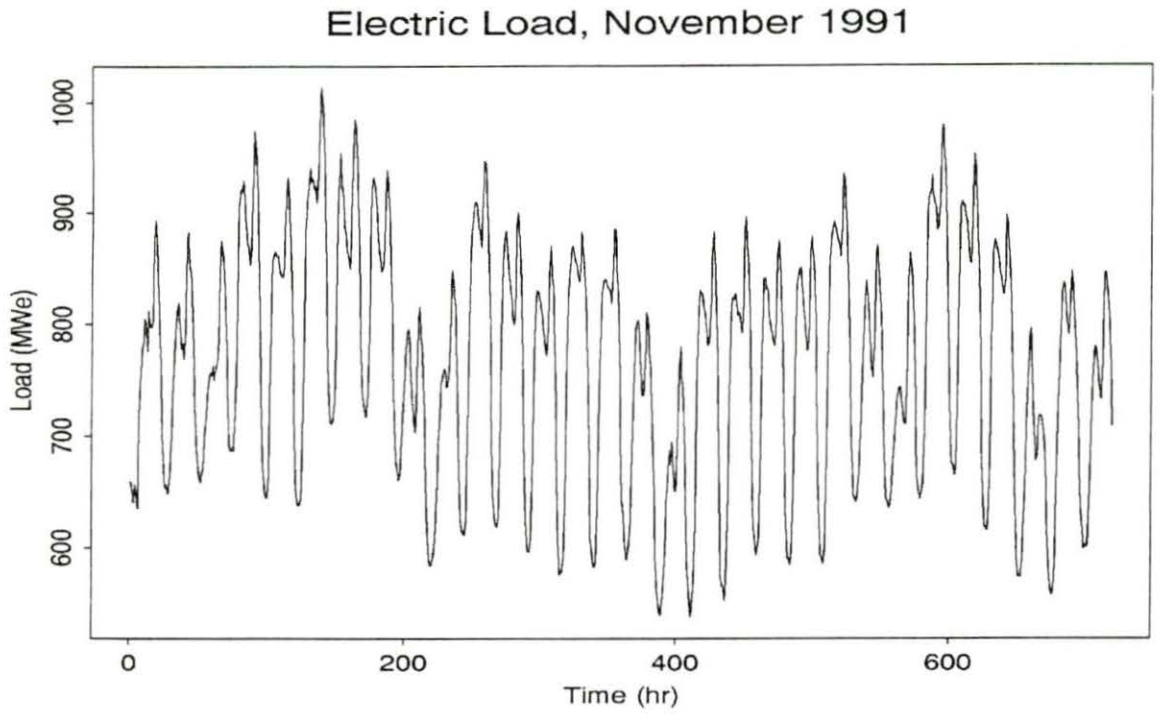

Electric Load, June 1991
Load (MWe)
600
800
1000
1200
1400
1600
0
200
400
600
Time (hr)

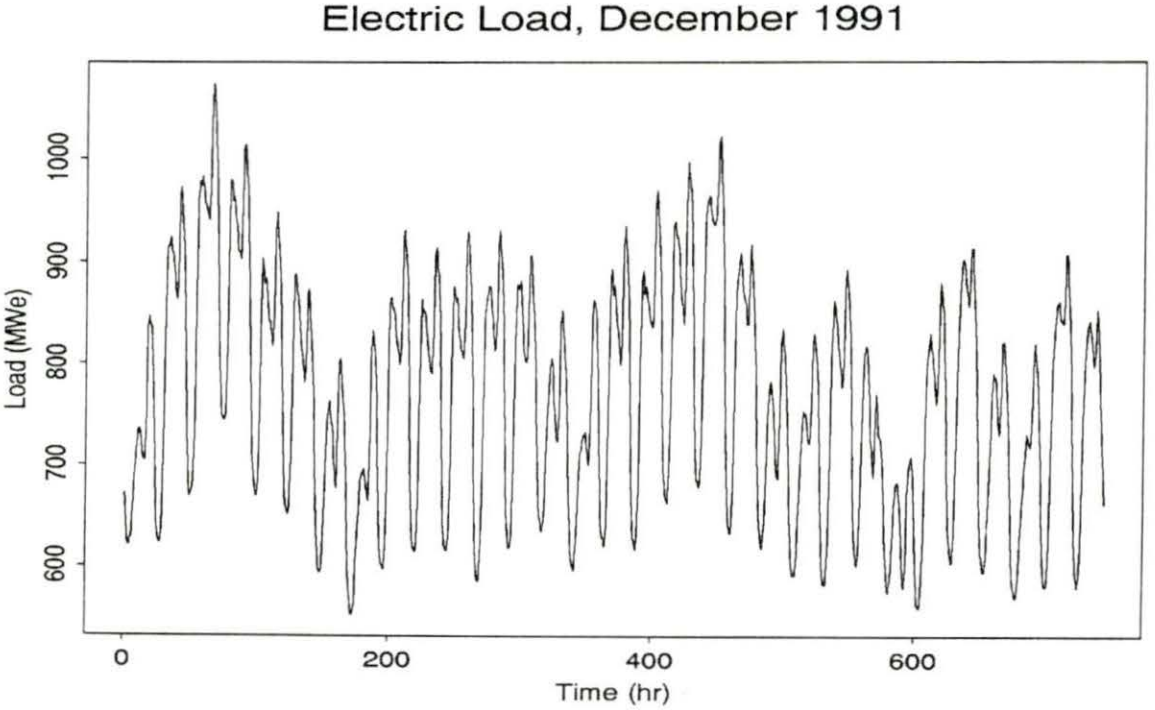

Electric Load, July 1991
Load (MWe)
600
800
1000
1200
1400
1600
0
200
400
600
Time (hr)

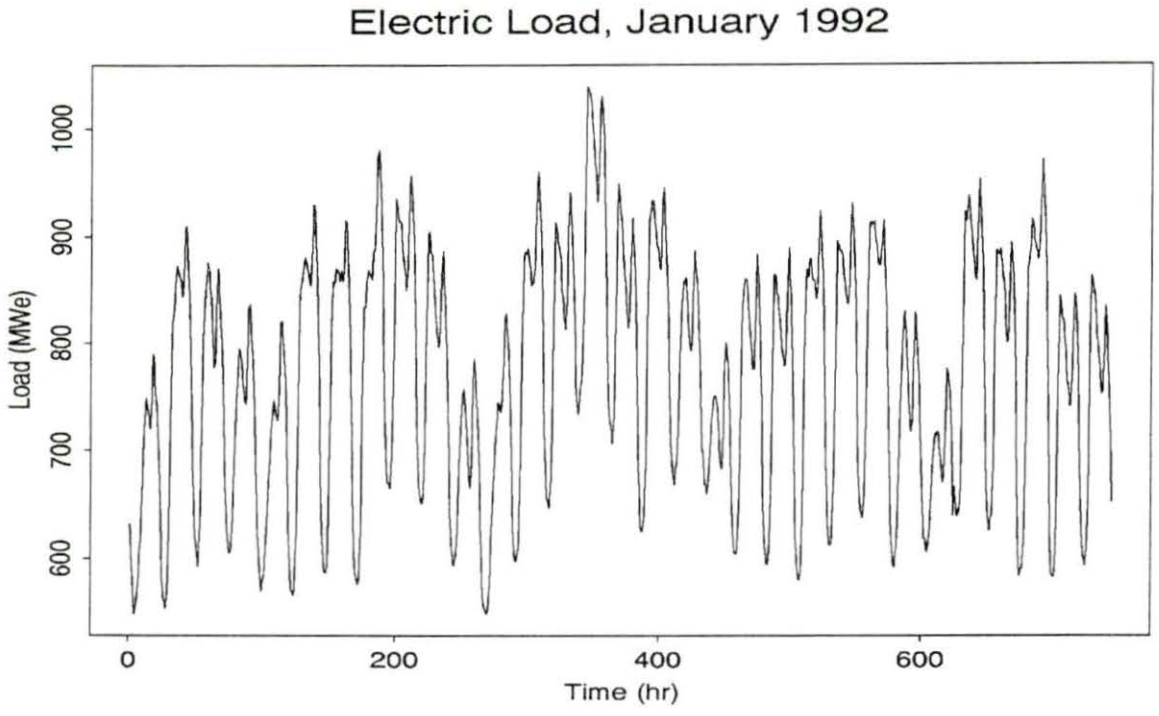

Electric Load, August 1991
Load (MWe)
600
800
1000
1200
1400
0
200
400
600
Time (hr)

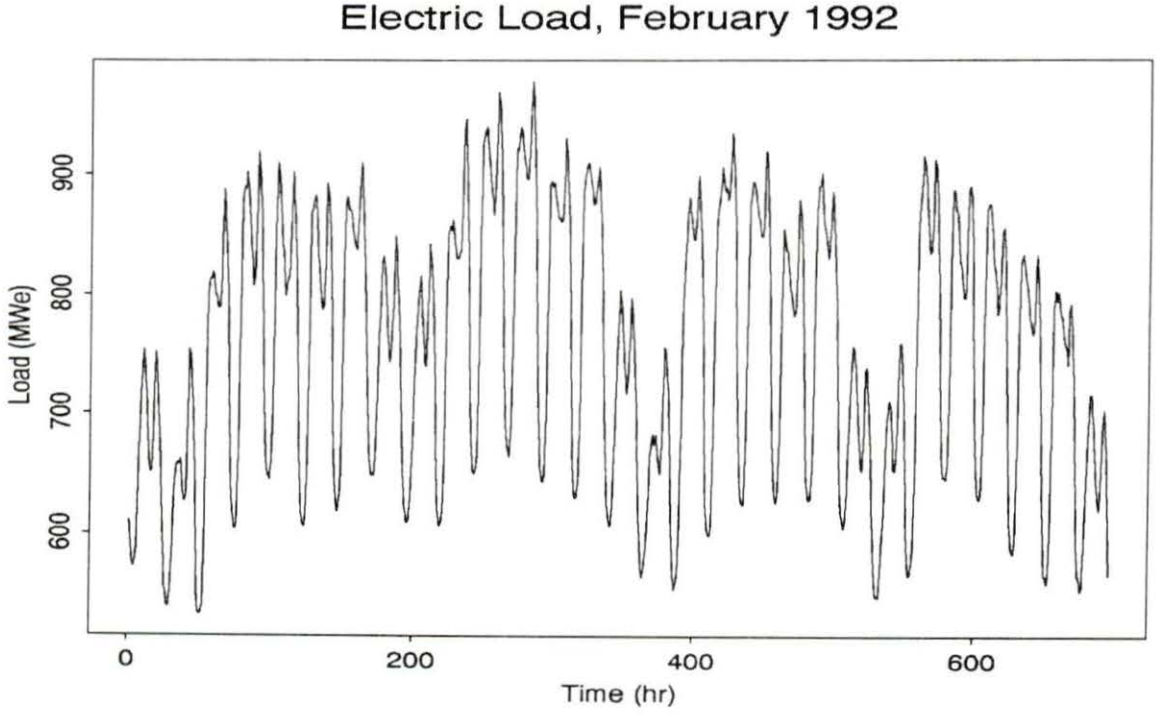

Electric Load, September 1991
Load (MWe)
600
800
1000
1200
1400
0
200
400
600
Time (hr)

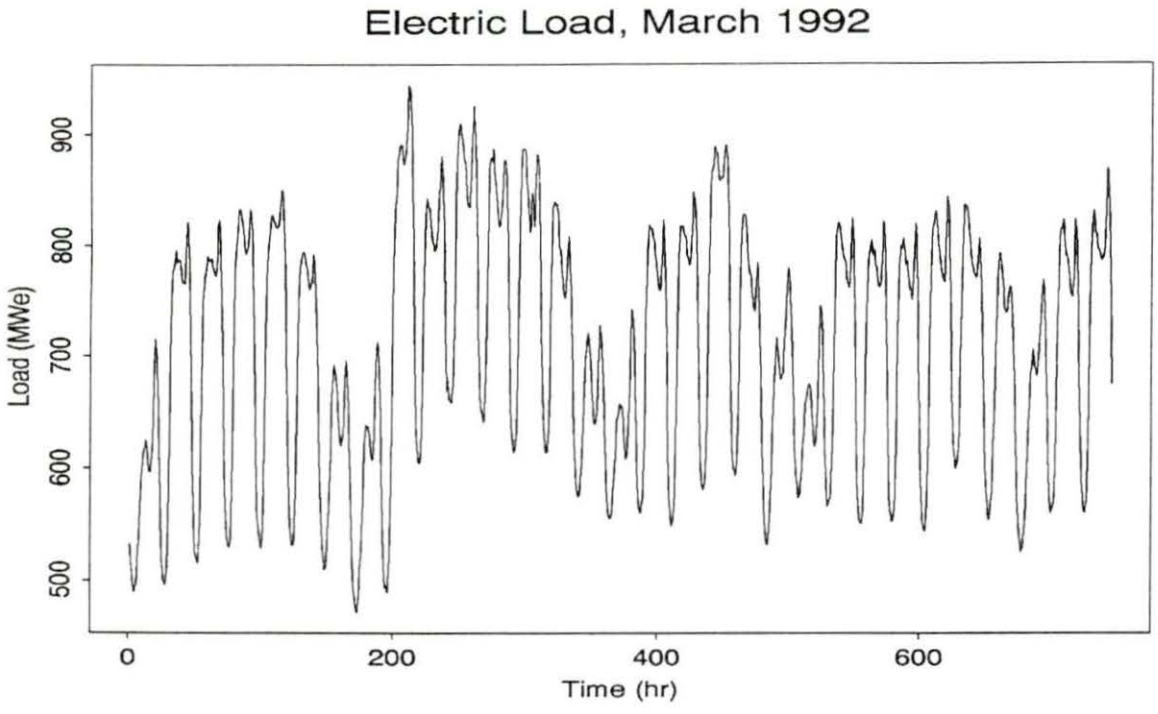

Electric Load, October 1991
Load (MWe)
500
600
700
800
900
0
200
400
600
Time (hr)

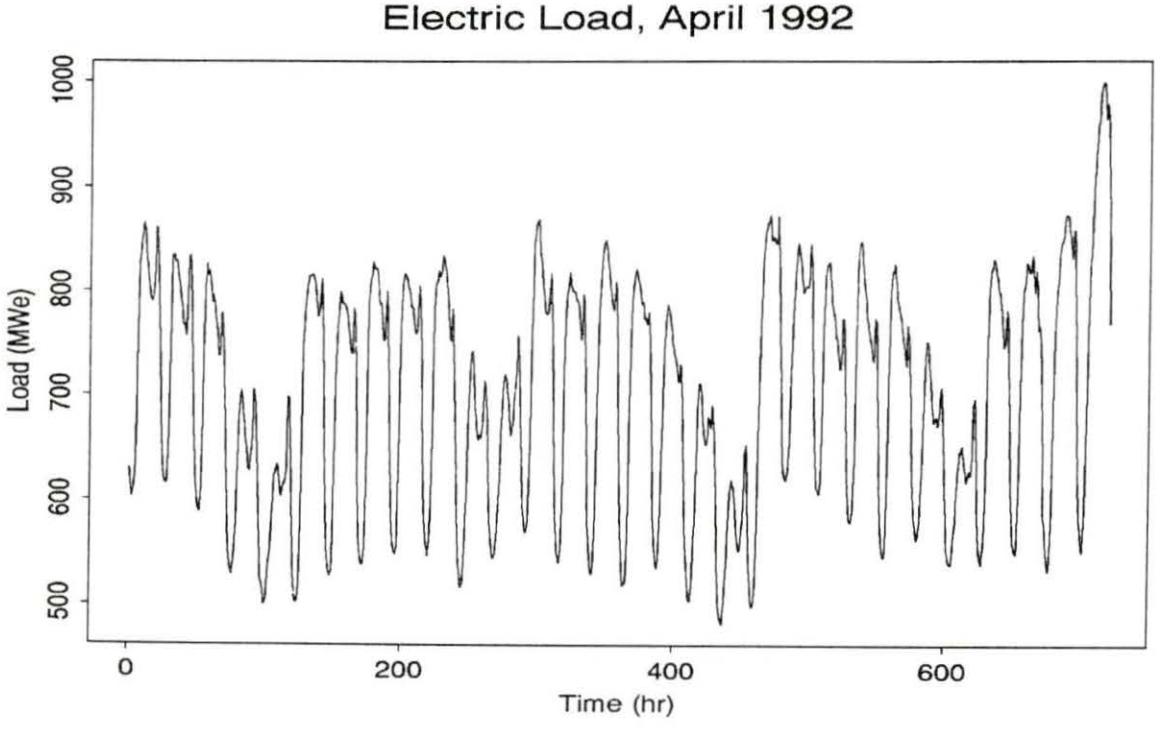

Electric Load, November 1991
Load (MWe)
600
700
800
900
1000
0
200
400
600
Time (hr)

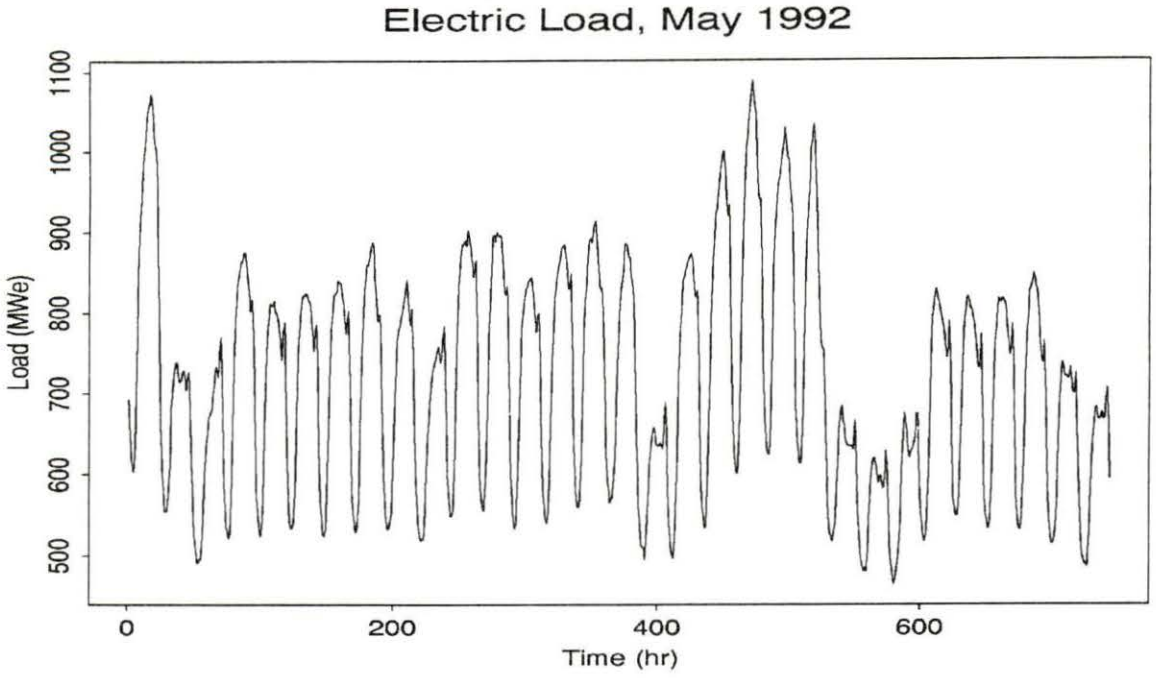

Electric Load, December 1991
Load (MWe)
600
700
800
900
1000
0
200
400
600
Time (hr)

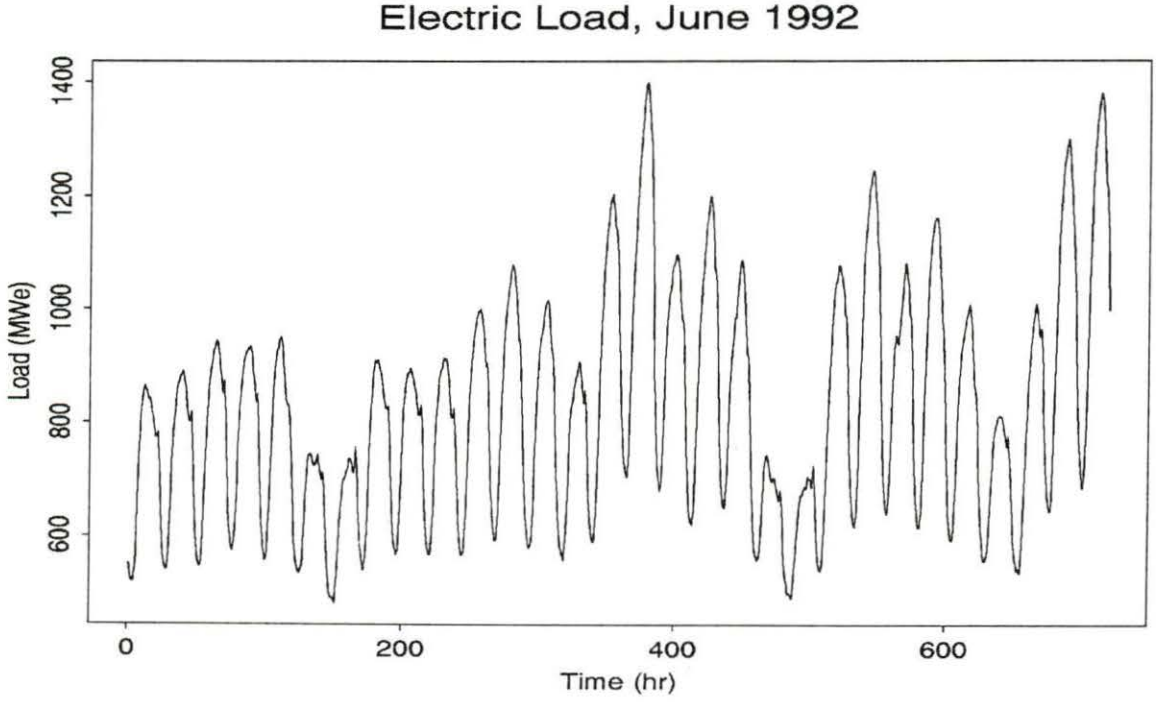

Electric Load, January 1992
Load (MWe)
600
700
800
900
1000
0
200
400
600
Time (hr)

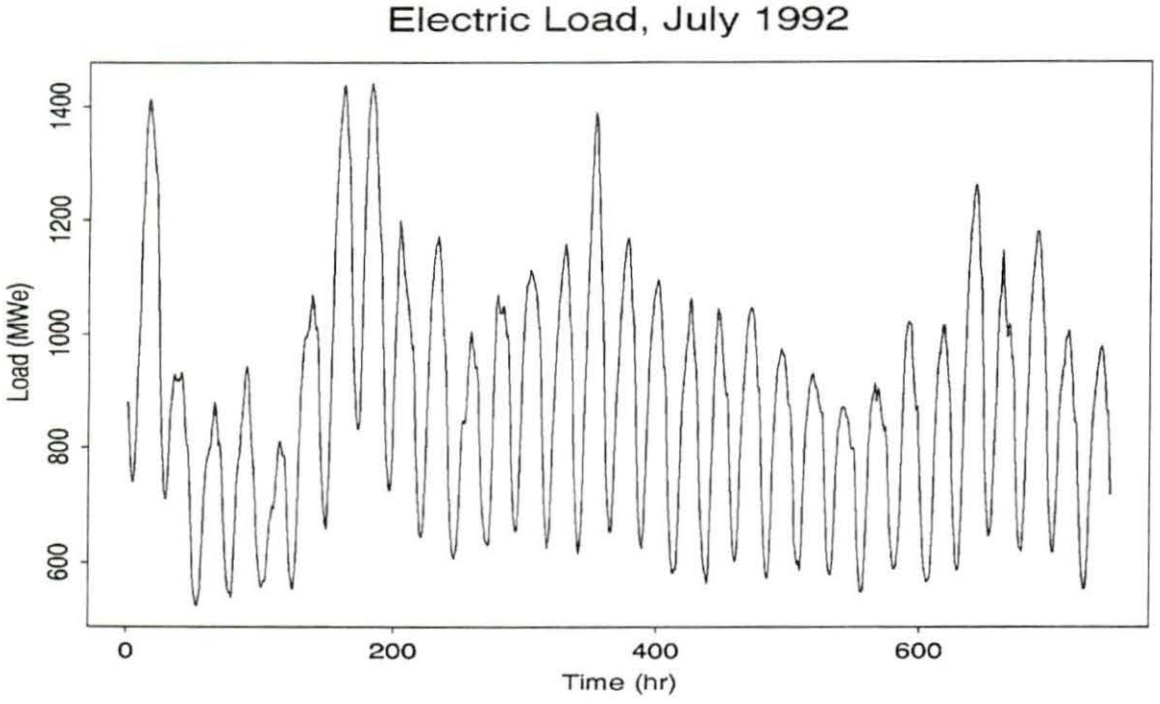

Electric Load, February 1992
Load (MWe)
600
700
800
900
0
200
400
600
Time (hr)

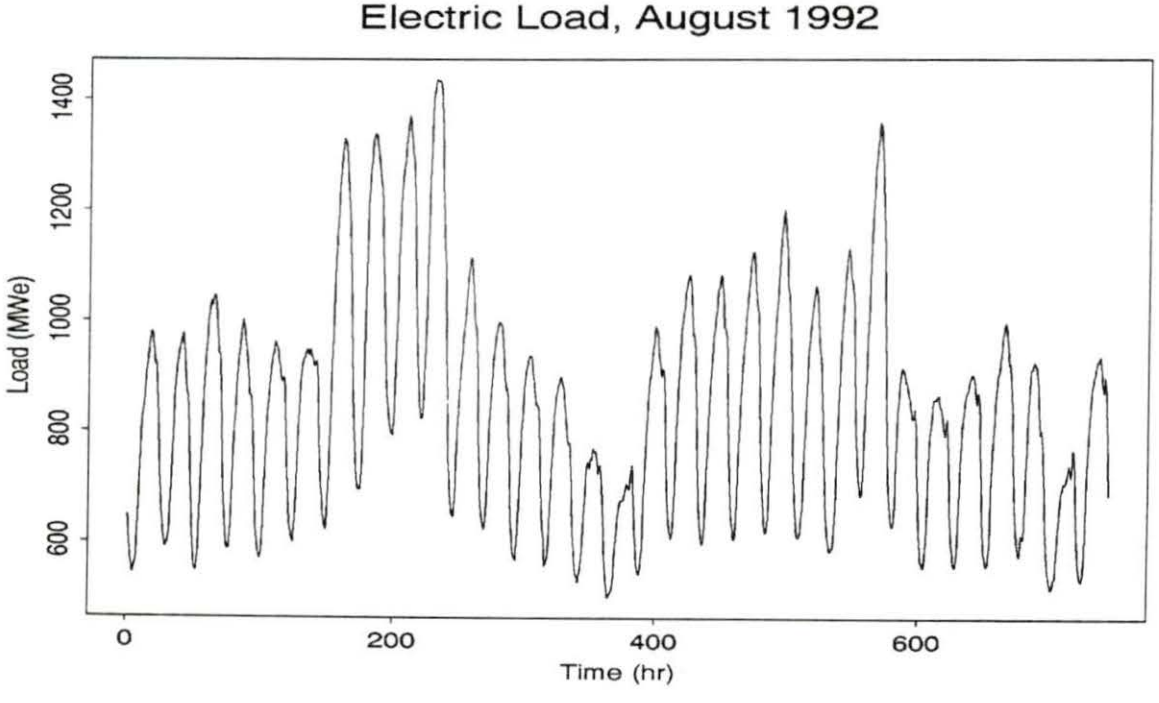

Electric Load, March 1992
Load (MWe)
500
600
700
800
900
0
200
400
600
Time (hr)


Electric Load, April 1992
Load (MWe)
500
600
700
800
900
1000
0
200
400
600
Time (hr)

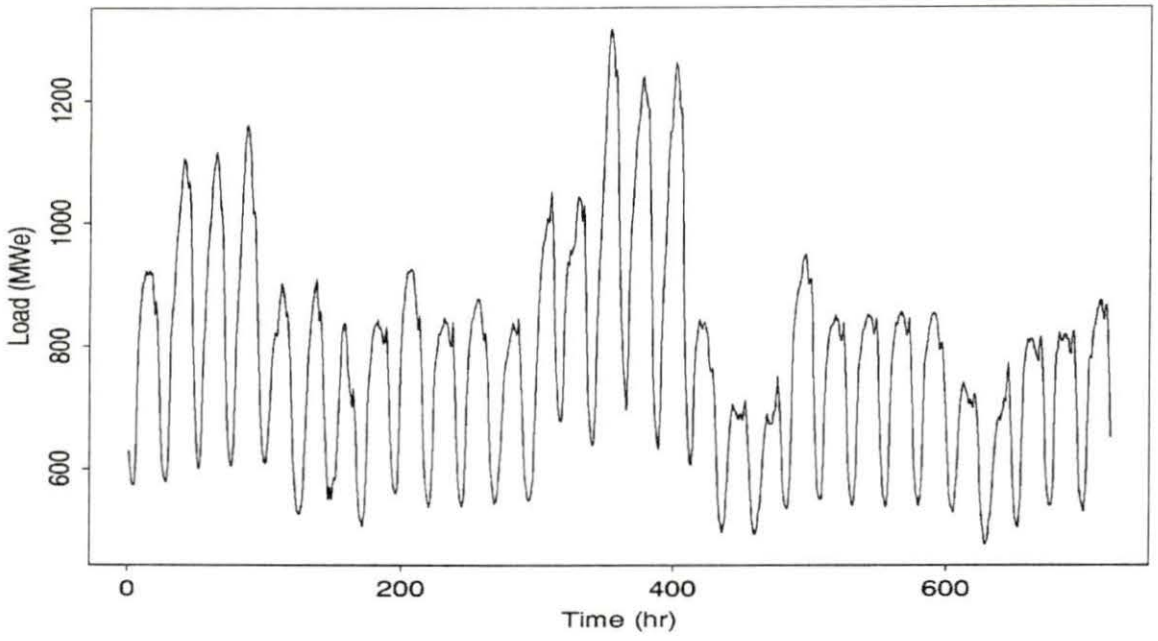

Electric Load, May 1992
Load (MWe)
500
600
700
800
900
1000
1100
0
200
400
600
Time (hr)


Electric Load, June 1992
Load (MWe)
600
800
1000
1200
1400
0
200
400
600
Time (hr)

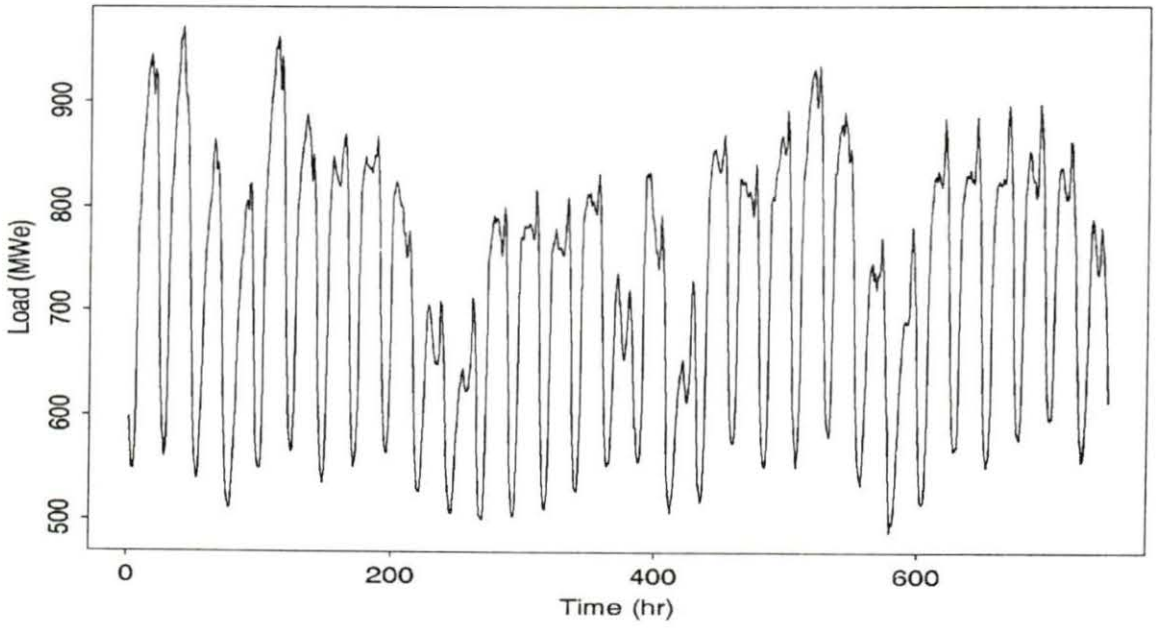

Electric Load, July 1992
Load (MWe)
600
800
1000
1200
1400
0
200
400
600
Time (hr)

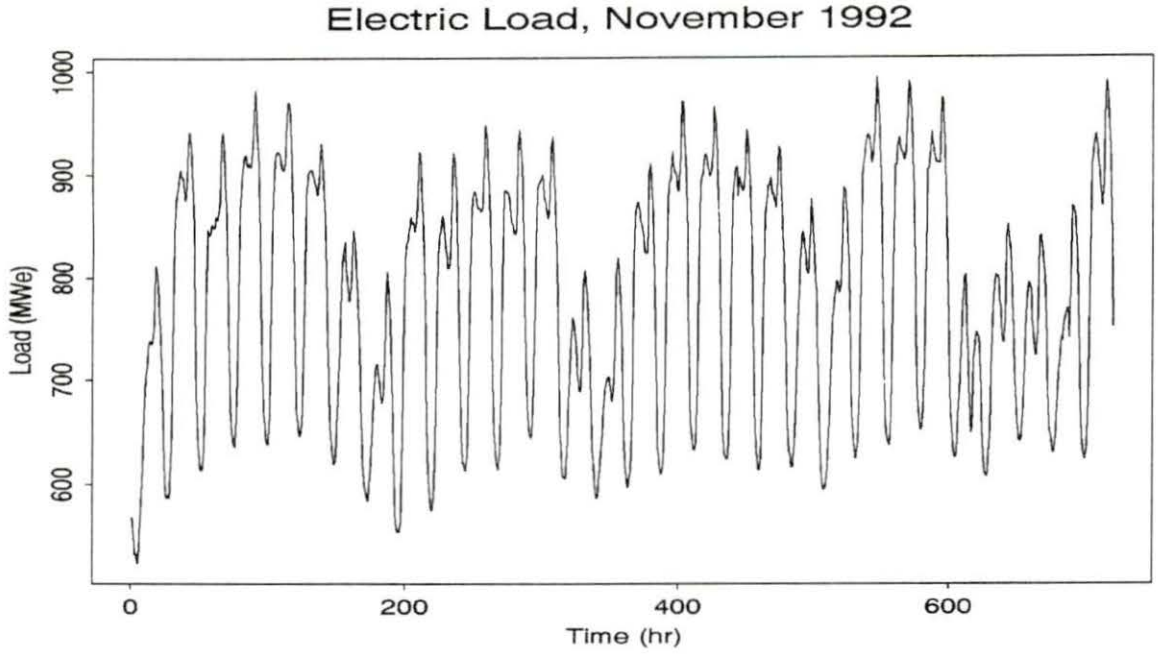

Electric Load, August 1992
Load (MWe)
600
800
1000
1200
1400
0
200
400
600
Time (hr)

Electric Load, September 1992



Electric Load, October 1992




Electric Load, November 1992
Load (MWe)
600
700
800
900
1000
0
200
400
600
Time (hr)

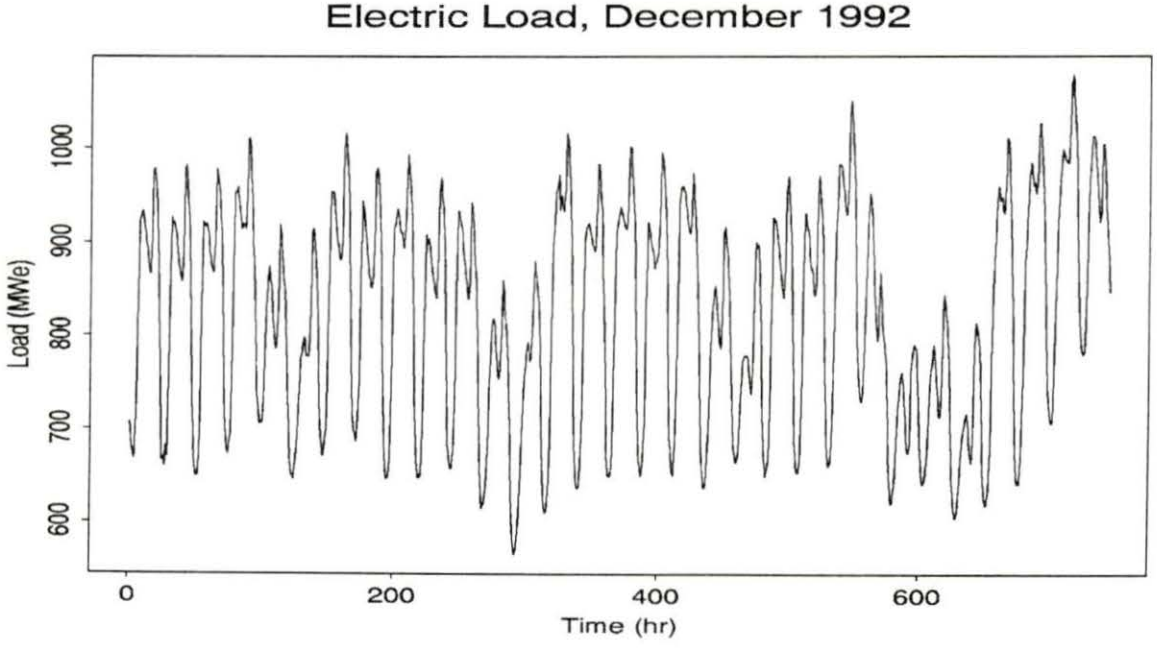

Electric Load, December 1992
Load (MWe)
600
700
800
900
1000
0
200
400
600
Time (hr)

Table 5.2 Comparison of other ANN models

Comparison of Other ANN Models					
#	Ref.	Prediction Period(s)	Data Set	% error	Notes
1	[39]	a. 24hr, one week ahead b. 24hr ahead (T–F)	Tr. – Winter peak Utility Te. – a. 5–mo. period from Feb.1 b. 1000 hr. period (T–F) from Jan. 1	a. 3.38 b. 2.09	a. Performed decomposition of load and used 5 Adaline to model. Each day had own set of ANNs.
2	[13]	Next days peak	a. Tr. 1/12–1/25 Te. 1/26–2/1 b. Tr. 1/19–2/1 Te. 2/2–2/8	a. 1.15 b. 1.22	Nonfully connected ANN, Pseudo forecasted weather.
3	[38]	a. peak load b. total load c. hourly load	Tr. 11/1/88 – 1/30/89 , except test Te. 1. 1/23–1/30 2. 11/9–11/17 3. 11/18–11/29 4. 12/8–12/15 5. 12/27–1/4	Average a. 2.04 b. 1.68 c. 1.40	Focus – normal weekdays, i.e. no holidays or weekends.
4	[16]	One hour ahead	Tr. 20 days Te. 25 days	4.1	Used adaptive NN, no weekends.
5	[2]	One half hour ahead	16 months of data from 4/90–7/91 forecasted for one year	Average Win.–3.16 Spr.–3.67 Sum–2.69 Aut.–4.24	Modular ANN design partitioned into season and day types, total of 48 ANNs. No holidays.
6	[39]	Total daily load	Tr. 1 yr. Te. 1 yr. x is total of both years	2.95	ANN retrained each day because if forecasting period =x, then training cases = x–1. Data divided into 5 subsets for day types.
7	[1]	Peak load on Thursday	6/83–4/94 16 patterns for training	3.11	
8	[29]	a. 24hr ahead b. 1hr ahead	Forecasted on 6 mo. Feb. to Jul.	a. 1.89 b. 1.84	a. Weekend and weekday models weekends grouped into 5 types.
9	[31]	a. 1hr ahead b. 24hr ahead c. peak load	One year or data divided into 12 train/test groups. Tr. 1 to 2 mo. Te. 7 days	a. 2.28 b. 3.08 c. 2.44	Errors averaged for same experiments for different companies. 12 ANN models, 1 for each mo.
10	[27, 28]	peak and valley	Tr. 10 previous similar days Te. the next similar day	p. 1.57 v. 0.20	Day identification w/self–organizing ANN. Retraining for each day.
11	[26]	peak and valley	Tr. 30 previous similar patterns Te. next 7 similar days	P. 0.10 v. 0.26	Day type identification. Only 4 examples presented.