ABSTRACT

Short-Term Load Forecasting Using System-Type Neural Network Architecture

Shu Du, M.S.

Mentor: Kwang Y. Lee, Ph.D.

This thesis presents a methodology for short-term load forecasting using a system-type neural network based on semigroup theory. A technique referred to as algebraic decomposition is used to decompose a distributed parameter system into a semigroup channel made of coefficient vectors and a function channel made of basis vectors. The actual load data is preprocessed by regression to become better correlated to daily time and temperatures. A rearrangement method based on the hourly temperature is developed to solve the problem of the roughness of the coefficient vector in the seimigroup channel. Interpolation or extrapolation of coefficient vector can be achieved for each hour using the historical temperatures and the temperature forecast. Recombination of the basis vector and predicted coefficient vector will give the next-day load forecasting. Load data from New England Independent System Operator is used to verify the capability of the proposed approach.

Short-Term Load Forecasting Using System-Type Neural Network Architecture

by

Shu Du, B.S.

A Thesis

Approved by the Department of Electrical and Computer Engineering

_____

Kwang Y. Lee, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Master of Science in Electrical and Computer Engineering

Approved by the Thesis Committee

_____

Kwang Y. Lee, Ph.D., Chairperson

_____

Robert J. Marks, Ph.D.

_____

Paul Grabow, Ph.D.

Accepted by the Graduate School
August 2009

_____

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANN – artificial neural network

AR – autoregressive

ARMA – autoregressive moving average

FRNN – fully connected recurrent neural network

RBFN – radial basis function network

LMS – least mean square

DRNN – diagonal recurrent neural network

SRN – simple recurrent network

PDE – partial differential equation

DPS – distributed parameter system

ISO – Independent System Operator

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to Professor Kwang Y. Lee for the guidance and support he has given me throughout my education and research. I also would like to thank Professor Robert J Marks and Professor Paul Grabow for reviewing my thesis and for their valuable comments that help me in completing this thesis.

My thanks also go to Professor Mike Thompson who has given me much help in completing degree program. I am thankful for the education of the Department of Electrical and Computer Engineering at Baylor University.

I would like to thank my uncle for his help after I came to the United States. Finally I would like to thank my beloved family in China for their support and encouragement.

CHAPTER ONE

Introduction

Forecasting load accurately plays a very important role for electric utilities in a competitive environment created by the electric industry deregulation. An electric company is confronted with many economical and technical problems in operation, planning, and control of an electric energy system since customers requires high quality electric energy to be supplied in a secure and economic manner [1]. Load forecasting helps an electric utility with making important decisions on generating, interchanging, and purchasing electric power, load switching, and infrastructure development. Besides load forecasting is crucial for energy suppliers, financial institutions, and others involved in electric energy generation, transmission, distribution, and markets [2].

Based on various time intervals, load forecasting can be divided into three main categories: short-term forecasting, medium-term forecasting and long-term forecasting. Short-term forecasting usually forecasts one hour to one week, medium-term forecasting concerns the future electric load from a week to a month, and long-term forecasting often predicts load of a year or even longer. The short-term forecasting is used for controlling and scheduling power generation, and also as inputs to load-flow study or contingency analysis [1]. Additionally, short-term load forecasting can assist in estimating load flows and in making decisions to prevent overloading of power system components. The medium-term and long-term forecasting are applied to determine the capacity of generation, transmission, or distribution system additions, the type of facilities required in

transmission expansion planning, annual hydrothermal maintenance scheduling, etc.

According to the work of Lee and Park [1], the load is a dynamic system which is mostly affected by two factors: time of the day and weather conditions. The time dependence of the load reflects the existence of a daily load pattern, which may vary for different weekdays and seasons. Among weather variables, temperature usually is the dominant weather factors influencing the load. Humidity and wind speed may also impact electric power consumption. For the models including weather variables, the total load may be decomposed into the non-weather sensitive load and the weather sensitive load which is usually predicted using correlation techniques. Besides, load series present different patterns for different types of customers. For instance, load consumed by residential and commercial customers usually shows a strong seasonal behavior as well as dependence on weather conditions. On the other hand, load with an industrial profile is much determined by operational decisions in a production or manufacturing facility.

Short-term load forecasting draws much attention. A variety of methods using statistical techniques or artificial intelligence algorithms, which include regression models, time series, neural networks, statistical learning algorithms, fuzzy logic, or expert systems, have been developed for short-term forecasting [1]. These methods all have been succeeded in short-term load forecasting problems. The success of a forecasting technique depends not only on the approach but also on the quality of input data which could contain proper patterns representing the system dynamics. In general, the load presents two distinct patterns: weekday and weekend load patterns. Weekday patterns include Tuesday through Friday and weekend patterns include Sunday through Monday. In addition, holiday patterns are different from non-holiday patterns.

Forecasting has been expected as one of the most promising application areas of artificial neural network (ANN). In the past, several authors have successfully applied the backpropagation learning algorithm to train ANNs for forecasting time series. However, there was also a negative opinion that the forecasting performance of the backpropagation algorithm was inferior to the simple linear regression. In order to address the importance of ANNs in power system engineering, The National Science Foundation organized a workshop and the results demonstrated that ANNs can be successfully used in short-term load forecasting with accepted accuracy [3].

The purpose of this thesis is to apply the proposed system-type neural network architecture based on semigroup theory to accomplish short-term load forecasting. Given the data from New England Independent System Operator, regression used for preprocessing the raw load data and a rearrangement step with respect to temperatures are both performed. A modeling technique, algebraic decomposition, decomposes the whole system into basis vector and coefficient vector which are correspondingly delivered to function channel and semigroup channel in the system-type neural network. The product of the predicted coefficient vector generated in semigroup channel and the basis vector in function channel supplies the next-day load forecasting.

This thesis consists of following chapters. Chapter two will introduce several approaches for short-term load forecasting. A few important neural networks are presented in Chapter three. Chapter four will give a general introduction to semigroup theory. The proposed approach in this thesis will be shown in Chapter five. Results are shown and discussed in Chapter six, and conclusions are drawn in Chapter seven.

CHAPTER TWO

Background

In general forecasting methods can be divided into two broad categories: parametric methods and artificial intelligence based methods. Based on analyzing qualitative relationships between the load and the factors affecting the load, the parametric methods formulate mathematical or statistical models of load. Then the parameters of the built model are estimated from historical data and the performance of the model is verified by analysis of forecast errors. Artificial intelligence based methods use artificial neural networks or fuzzy systems as load models. For both of the categories, several factors should be considered in short-term load forecasting, such as the time factor, weather data as well as possible customers' classes. The time factors influence the load hourly, daily and seasonally. Loads between weekdays and weekends, as well as holidays and non-holidays also show differences. Apparently the electric loads are dependent upon weather conditions significantly. Variations of dry-bulb temperature, dew point, wind speed, humidity, and cloud cover can change the load dynamics. This is especially true in residential areas. For those areas where the industry collects, temperature may not be an important variable any longer. It may be necessary to have information regarding operational decisions of plants taken into account as factors.

Parametric Load Forecasting Methods

Diverse statistical techniques have been developed for short-term load forecasting. One of the most widely used techniques is the regression method which is usually applied

to model the relationships of load consumption and other factors such as weather and non-weather variables influencing the electric load. Based on the summary made by Moghram and Rahman [4], the explanatory variables of the model are usually identified on the basis of correlation analysis on each of these (independent) variables with load (dependent) variable. Experience about the load to be modeled helps an initial identification of the possible influential variables. The estimation of the regression coefficients is usually achieved using the least square estimation technique. Statistical tests (such as the F-statistic test) can be performed to determine the significance of these regression coefficients.

Another approach is the time-series method, which treat the load pattern as a time-series signal with known seasonal, weekly, and daily periodicities. These periodicities give a rough prediction of the load at the given season, day of the week and time of the day. The difference between the predicted and the actual load is considered as a stochastic process. The analysis of this random process leads to a more accurate prediction [5]. Time-series models are based on the assumption that the data have an internal structure. The forecasting methods detect and explore such a structure. Techniques used to estimate the time-series signal of a load pattern include Autoregressive (AR) model, Autoregressive Moving Average (ARMA) model, spectral expansion technique and state estimation. The AR model has been used for decades in such fields as economics, digital signal processing, as well as electric load forecasting. In [5], an adaptive autoregressive modeling technique enhanced with partial autocorrelation analysis was presented. The ARMA model is used to express current value linearly in terms of past values. It has been extensively applied to load forecasting. An ARMA

model was obtained by identifying a finite order AR model for hourly load forecasting [6]. The spectral expansion technique utilizes Fourier series. Load pattern can be decomposed into a number of sinusoids with different frequencies. Each sinusoid with a specific frequency represents an orthogonal base. A linear combination of these orthogonal bases with proper coefficients can represent a perfectly periodic load pattern if the orthogonal bases span the whole signal space. The behavior of weather independent load component can be represented by Fourier series in terms of time functions [7]. Besides, state equations are used to model the load demand. The main reason is that the popular Kalman filtering theory can be applied to obtain the optimum forecasts. The identification of the model parameters is the main difficulty associated with this approach because Kalman filtering theory assumes that the model is exactly known beforehand [8]. Generally, techniques in time-series approach work well unless there is an abrupt change in the environmental or sociological variables which are believed to affect load pattern. If there is any change in those variables, the time-series techniques cannot provide accurate forecasting.

Artificial Intelligence Based Load Forecasting Methods

Among artificial intelligence based models, artificial neural networks (ANN) have probably received the most attention because of their straightforward implementation and relatively good performance. The ANN has also been applied in other power system problems such as security assessment, harmonic load identification, alarm processing, fault diagnosis, and topological observability [9]. It is known that one of the most promising application areas of ANN is the load forecasting. The application of artificial neural networks has been a widely studied electric load forecasting technique. Neural

networks reduce the computational burden within the parametric approach. Unlike the parametric methods, the great superiority of using neural networks is that they are able to learn the above mentioned dependencies directly from the historical data without the necessity of selecting an appropriate model.

The most popular architecture for short-term load forecasting is feedforward network with backpropagation training algorithm, which uses real valued functions and supervised learning. Several authors have applied the backpropagation learning algorithm [10] to train ANNs for forecasting time series. Application of this idea to the real world problem can be found in Werbos's work [11], where he applied the backpropagation algorithm to the recurrent gas market model.

There are a few types of neural networks that have been applied for load forecasting. A multi-layered feedforward network with one hidden neuron layer is most commonly used [1], [12]. A recurrent ANN trained by dynamic backpropagation algorithm is proposed as the methodology for electric load forecasting [13]. A nonlinear load model is suggested and the parameters of the nonlinear model are estimated using dynamic backpropagation algorithm. And a modified recurrent neural network, named diagonal recurrent neural network, is presented to overcome the training and convergence problems which arise from fully connected recurrent neural network (FRNN). This new architecture requires fewer weights than FRNN and rapid convergence has been demonstrated [14]. In [15], an architecture including two ANN forecasters, one predicts the base load and the other forecasts the change in load, is proposed. The final forecast is computed by an adaptive combination of these forecasts. Also a radial basis function network (RBFN) has the predictive capability and the ability to produce accurate

measures. A comparison between results from the RBFN and the backpropagation neural network shows that the former one performs better than the latter [16].

Other artificial intelligence based techniques, like fuzzy logic and support vector machines have been also applied, however, typically in conjunction with ANN or statistical models. Several hybrid models have been developed for load forecasting. A short-term load forecaster with an ANN and a fuzzy logic system is presented [17]. A genetic algorithm based approach is developed to automatically optimize the number of rules and the fuzzy membership functions. In [18], Senjyu et al. propose a hybrid model in which a fuzzy logic, based on similar days, corrects the neural network output to obtain the next day forecasted load.

CHAPTER THREE

Artificial Neural Networks

Overview

"A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use" [19]. The inspiration of neural networks was from examination of the central nervous system and the neurons. It imitates the brain at two aspects. First, the network acquires the knowledge from its environment by a learning process. Second, interneuron connection strengths, known as synaptic weights, function to store the obtained knowledge [19]. Neural networks appear to be developed recently. However, the earliest work in neural network science dates back to the 1940's. The neurophysiologist Warren McCulloch and the logician Walter Pitts introduced the first artificial neuron [20]. This started a completely new era within artificial intelligence. The next major development in neural network is made by Donald Hebb. He published a book "The Organization of Behavior" [21] which supported and further reinforced the theory of McCulloch and Pitts. Though, the future was not as bright as it had appeared at first glance. After an initial period of enthusiasm, the field of neural network underwent a period of frustration and disrepute. Minsky and Papert pointed out the limitations that the perceptron (two-layer network) has [22]. Such limitations led to the decline of the neural networks. However, this did not prevent some pioneers from keeping interest in neural networks. In the early 1980's, neural networks retrieved attention of researchers.

Nowadays, the study of the ANN models is receiving rapid and increasing importance because of their superiority on some of the problems which have been intractable by standard serial computers in computer science and artificial intelligence. Neural networks are better suited for achieving human-like performance in fields, such as aircraft control, voice synthesis, image recognition, machine vision, manipulator controllers, etc.

Conventional Neural Network

Neural network architecture can be broken down into two main categories: feedforward neural network which consists of single-layer or multi-layer networks, and feedback (recurrent) neural network which is in opposition to the former. Feedforward ANN was the first simplest type of neural network developed. The information travels in only one direction, forward, from input nodes to output nodes. There are no cycles or loops (feedbacks) in the network, which means the output of any layer does not affect that same or previous layer. Figure 1 shows a simple feedforward artificial neural network. It is composed of three layers of computational units: a layer of “input” units, a layer of “hidden” units, and a layer of “output” units. Each neuron in one layer has directed connections to the neurons of the subsequent layer. The hidden neuron is typically modeled with a nonlinear sigmoidal activation function. It also can be modeled with other activation functions to constitute other types of neural networks, such as Gaussian functions, in which case these are radial basis function networks (RBFN) [19].

Feedback network allows signals transporting in both directions so that it has closed loops in the network topology. Feedback is said to exist in a dynamic system whenever the output of an element in the system influences in part the input applied to

that particular element, thereby producing one or more closed paths for the transmission of signals around the system.

Feedback networks are dynamic; their states are changing continuously until they reach an equilibrium point. Once the input changes, the states leave the equilibrium point and a new equilibrium needs to be found. Feedback neural networks are developed to handle the time varying or time-lagged patterns and are very useful for the problems in which the dynamics of the process is complex. Examples of the recurrent neural networks are: Hopfield network, Regressive networks, diagonal recurrent neural networks, and Elman networks.

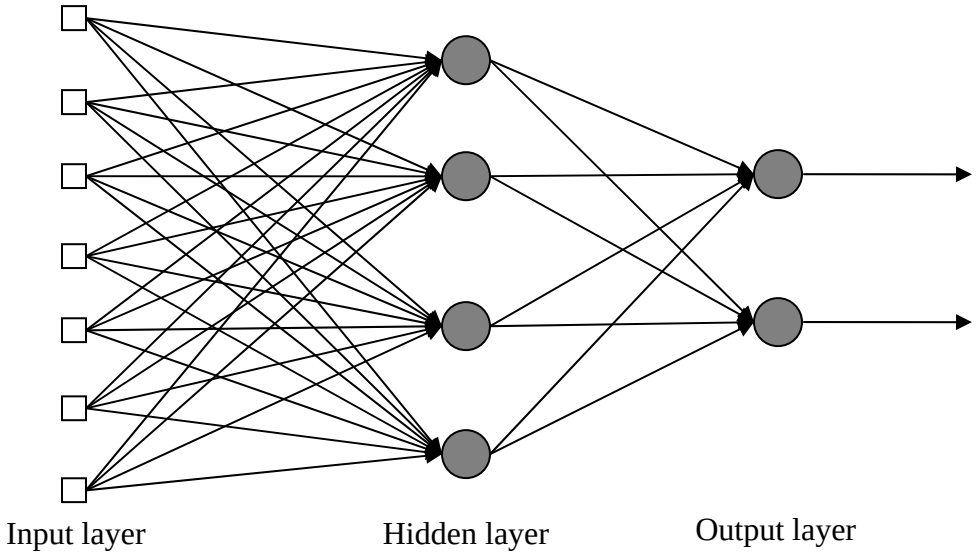

Figure 1. A simple feedforward neural network.

Overview of Learning Algorithms

"Learning is a process by which the free parameters of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the

parameter changes take place" [19]. There are three major learning paradigms, each corresponding to particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning. The task of the supervised learning is to predict the value of the function for any valid input object after having seen a number of training examples (i.e., pairs of input and target output). Examples of supervised learning algorithms include the least-mean-square (LMS) algorithm and its generalization known as the backpropagation algorithm. The LMS algorithm involves a single neuron, whereas the backpropagation algorithm involves a multi-layered interconnection of neurons. Normally, the backpropagation algorithm is more powerful in application than the LMS algorithm [19]. Unsupervised learning is distinguished from supervised learning in that the learner is given only unlabeled examples.[19] "Reinforcement learning is the on-line learning of an input-output mapping through a process of trial and error designed to maximize a scalar performance index called a reinforcement signal." [19]

The backpropagation learning algorithm is the most frequent-used method in training the feedforward neural networks. It is a generalization of the Widrow-Hoff error correction rule [23]. Basically, the backpropagation process consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern (input vector) is applied to the sensory nodes of the network, and its effect propagates throughout the network. Finally, the network produces a set of outputs as the actual responses. In the forward pass the synaptic weights of the network are all fixed. On the other hand, the synaptic weights are all adjusted in accordance with the error-correction rule during the backward pass. An error signal is produced by subtracting the actual response of the network from a desired (target)

response. This error signal is propagated backward through the network in the opposite direction of synaptic connections. The synaptic weights are adjusted so as to make the actual response of the network to be as close as much to the desired response [19].

Benefit of Neural Network

Having outstanding ability to derive meaning from complicated or imprecise data, neural networks can be applied to detect or extract highly complex patterns and trends. A well trained neural network can be thought of as an “expert” in handling the category of information it has been given. This expert can then be used to provide projections given new situations and answer “what if” questions.

“A neural network derives its computing power through its massively parallel distributed structure and its capability to learn and generalize” [19]. Generalization involves producing reasonable outputs for inputs not encountered during training. With these two information-processing capabilities, it is possible for neural networks to solve complex (large-scale) problems that are currently intractable. In practice, neural networks need to be integrated into a consistent system engineering approach rather than work by themselves alone. Specifically, a complex problem of interest is decomposed into a number of relatively simple tasks, and neural networks are assigned a subset of the tasks in which they can put their inherent capabilities to good use [19].

Failures and Shortcomings of Conventional Neural Networks

Recently, overall architecture of neural networks tends to shift from simple or component-type networks to system-type architectures. System-type architectures make use of combination of several neural networks to settle more complex tasks. The reasons

why emphasis shifts from component-type to system-type neural networks include: first, in the interests of advancing science, system-type neural networks are considered as the next stage; second, drawbacks have been found out in component-type neural networks [24]. Therefore, it is essential to exploit a system-type neural network which utilizes one or more components to learn individual functions, and another component to synthesize their contributions.

The first attempts marching toward system-type neural networks used various ad-hoc approaches mainly based on intuition. Recent efforts have been made to develop a disciplined approach in this area. The most popular architecture seems to be the “Modular Connectionist Architecture” which was advocated by Jacobs and Jordan [25]. One example of this architecture is shown in Figure 2 [26]. A group of expert networks which are trained individually are connected together through a component called the “gating network” element which is to decide the relative contributions to be made by each expert network component.

The most serious flaw in system-type neural networks is the shortage of a coherent discipline in the architecture design as well as the design of the learning algorithm. The entire design is completed on intuition. However, the proposed method depends on semigroup theory for the design of both architecture and the learning algorithm.

Radial Basis Function Network

Radial basis functions were first introduced in the solution of the real multivariate interpolation problem. The early work on this subject is surveyed by Powell [27]. In [19], the statement of the interpolation problem is supplied in its strict sense:

Given a set of N different points $\{x_i \in R^{m_0} \mid i=1,2,\cdots,N\}$ and a corresponding set of N real numbers $\{d_i \in R^1 \mid i=1,2,\cdots,N\}$, find a function $F: R^N \to R^1$ that satisfies the interpolation condition:

$$F(x_i)=d_i, i=1,2,\cdots,N \tag{1}$$

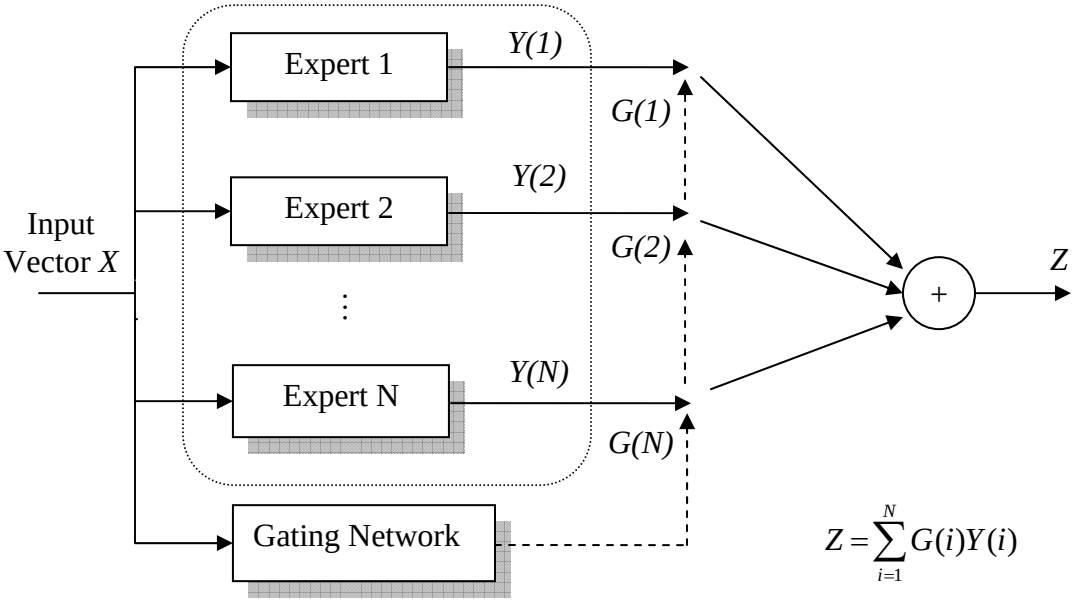

Figure 2. Modular connectionist architecture [26].

The radial basis function technique involves choosing a function F that has the following form:

$$F(x)=\sum_{i=1}^{N} w_i \varphi(\| x-x_i \|) \tag{2}$$

where $\{\varphi(\| x-x_i \|) \mid i=1,2,\cdots,N\}$ is a set of N arbitrary (generally nonlinear) functions, known as radial basis functions, and || || denotes a norm that is usually taken to be Euclidean. The known data points $x_i \in R^{m_0}$, $i=1,2,\cdots,N$ are taken to be the centers of the radial basis functions [19].

The structure of a radial basis function network in its most basic form involves three completely different layers. The first layer is an input layer containing source nodes. The second layer is a hidden layer of high enough dimension, whose served purpose is different from that in a multilayer perceptron. The output layer responds to the activation patterns applied to the input layer. From the input space to the hidden-unit space, the transformation is nonlinear, whereas from the hidden-unit space to the output space it is linear [19]. Design of radial basis function network may need more neurons when compared to standard feedforward networks. However, radial basis networks can be designed in a fraction of the time spent on training standard feedforward networks. Figure 3 shows the architecture of a radial basis function network.

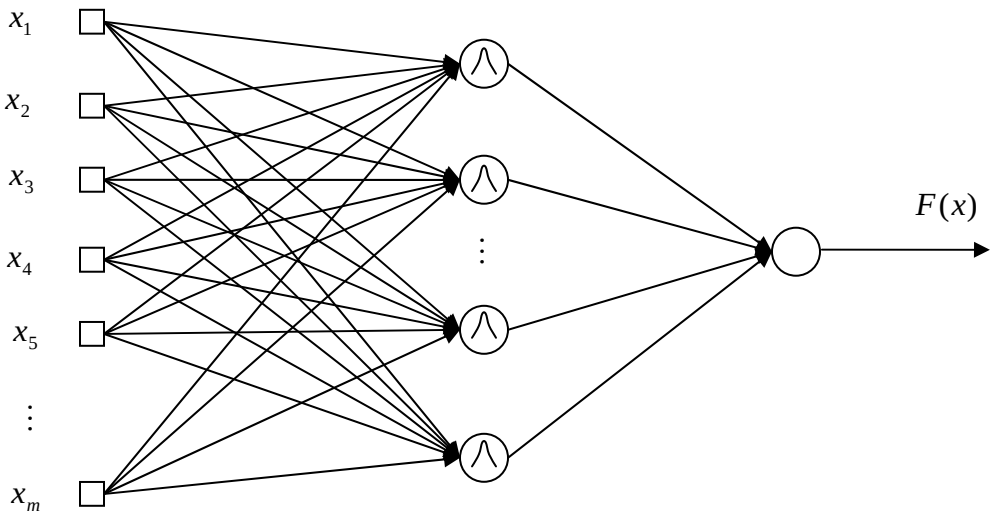

Figure 3. Radial basis function network [19].

A mathematical justification for the rationale of a nonlinear transformation followed by a linear transformation may be dated back to an early paper by Cover [28]. It is noted from this paper that a pattern classification problem cast in a high dimensional space is more likely to be linearly separable than in a low dimensional space. Hence this

is the reason for designing high dimension of the hidden-unit space in an RBF network. Nevertheless, through careful design it is still possible to reduce the dimension of the hidden-unit space, especially when the centers of the hidden units are made adaptive [19]. In addition, it is important to note that the dimension of hidden-unit space is directly influencing the performance of the network to approximate a smooth input-output mapping [29]. The higher the dimension of the hidden space is, the more accurate the approximation is.

Diagonal Recurrent Neural Network

The diagonal recurrent neural network (DRNN) was developed by Ku and Lee [30]. The architecture of DRNN is shown in Figure 4. Since there are no interlinks among neurons in the hidden layer, the DRNN has considerably fewer weights than the fully connected recurrent neural network and the network is simplified considerably. Therefore the DRNN requires a shorter training time.

The mathematical model for DRNN is represented as follows:

$$o(k) = f_o(r(k)) \tag{3}$$

$$r(k) = \sum_j w_j^O x_j(k),\ x_j(k) = f(s_j(k)) \tag{4}$$

$$s_j(k) = w_j^D x_j(k-1) + \sum_i w_{ij}^I i_i(k) \tag{5}$$

where $i_i(k)$ is the i^{th} input to the DRNN, $s_j(k)$ is the sum of inputs to the j^{th} recurrent neuron, $x_j(k)$ is the output of the j^{th} recurrent neuron and $o(k)$ is the output of the DRNN. Here $f_o(\cdot)$ and $f(\cdot)$ are the common sigmoid functions, and w^I, w^D, and w^O are input, recurrent, and output weight vectors, respectively, in R^{n^i}, R^{n^d}, and R^{n^o} [30].

On the basis of the similar procedure from Ku and Lee, the following Lemma can be derived.

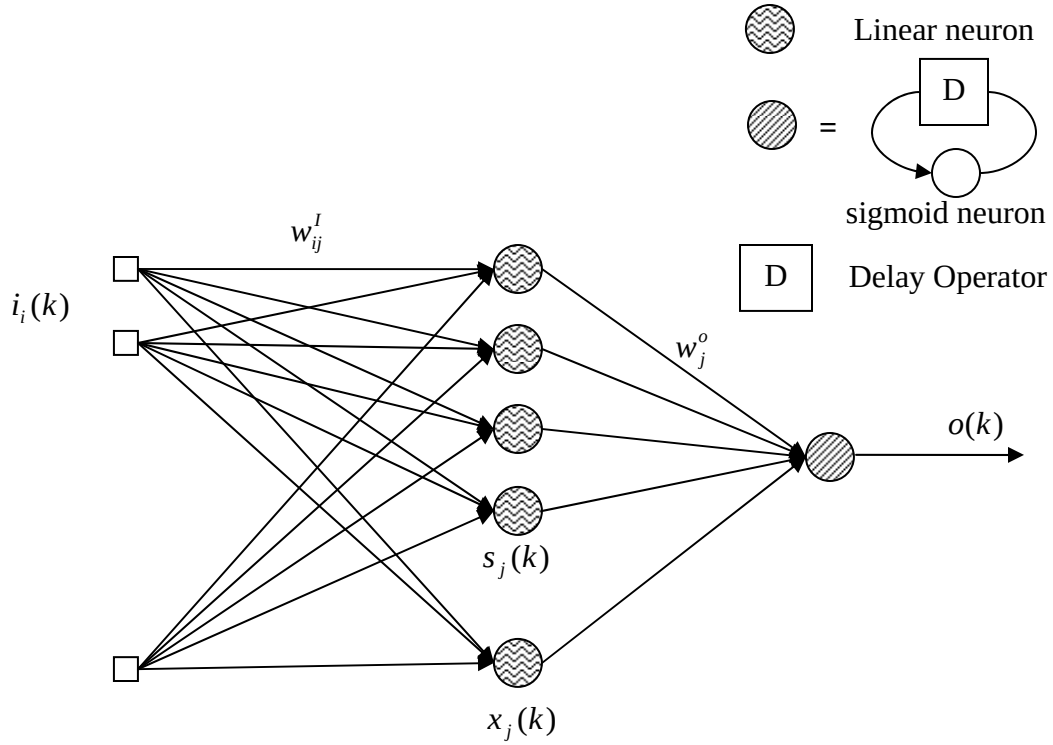

Figure 4. Diagonal recurrent neural network [30].

Lemma: Given the DRNN and described by (3), (4), and (5), the output gradients with respect to output, recurrent, and input weights, respectively, are given by

$$\frac{\partial o(k)}{\partial w_j^O} = f_o^{'}(r(k))x_j(k) \tag{6}$$

$$\frac{\partial o(k)}{\partial w_j^D} = f_o^{'}(r(k))w_j^O p_j(k) \tag{7}$$

$$\frac{\partial o(k)}{\partial w_{ij}^I} = f_o^{'}(r(k))w_j^O q_{ij}(k) \tag{8}$$

where $p_j(k) \equiv \frac{\partial x_j(k)}{\partial w_j^D}$ and $q_{ij}(k) \equiv \frac{\partial x_j(k)}{\partial w_{ij}^I}$, and satisfy the dynamic equations

$$p_j(k) = f'(s_j)(x_j(k-1) + w_j^D p_j(k-1)),\ p_j(0) = 0 \tag{9}$$

$$q_{ij}(k) = f'(s_j)(i_i(k) + w_j^D q_{ij}(k-1)),\ q_{ij}(0) = 0 \tag{10}$$

The error function is defined as follows:

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \tag{11}$$

Let E_p be the measure of error on pattern p and let $E = \sum_p E_p$ be the overall measure of the error, where t_{pj} is the target output for j^{th} component of the pattern p and o_{pj} is the corresponding output for the j^{th} component.

The output gradients can be used for DRNN to obtain the negative error gradient. Hence the weight update rule can be obtained as follows:

$$\Delta w^O(n+1) = \eta^O\left(-\frac{\partial E_P}{\partial w^O}\right) + \alpha \Delta w^O(n) \tag{12}$$

$$\Delta w^D(n+1) = \eta^D\left(-\frac{\partial E_P}{\partial w^D}\right) + \alpha \Delta w^D(n) \tag{13}$$

$$\Delta w^I(n+1) = \eta^I\left(-\frac{\partial E_P}{\partial w^I}\right) + \alpha \Delta w^I(n) \tag{14}$$

where η^O, η^D, and η^I are the learning rate for DRNN weights w^O, w^D, w^I, respectively, and α is the momentum constant to determine the effect of past weight changes." The update rules require a proper choice of learning rate η. A small value of η ensures the convergence but the speed may be very slow; on the other hand, large η will make the algorithm unstable [30].

Simple Recurrent Network

Simple recurrent network (SRN) is demonstrated in [31] and the structure of a SRN is shown in Figure 5.

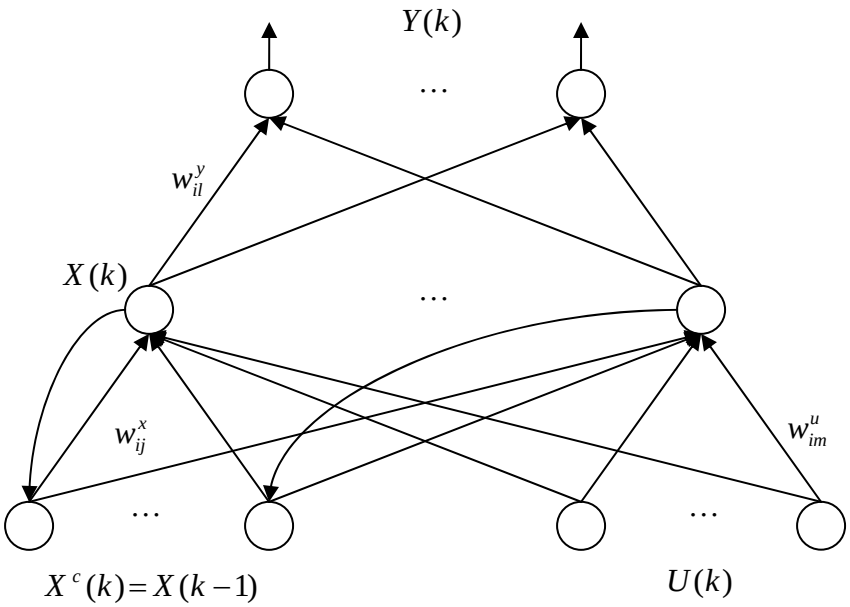

Figure 5. Simple recurrent network [32].

Besides the typical connections from input layer to hidden layer and from hidden layer to output layer, the SRN contains additional recurrent connections from the hidden neurons to a layer of context units with a fixed weight of one. These context units always maintain copies of the previous outputs of the hidden neurons, and then transport them back to the hidden layer. Thus the hidden neurons maintain a sort of their prior states, which allows the network to perform learning tasks that extend over time. Therefore, at each time cycle the hidden unit activations are copied to the context units; till the next time cycle, the context combined with the new inputs activates the hidden units. The hidden units then take on the job of mapping new inputs and prior states to the output so

as to respond to the external stimulus. Due to the nature of feedback between the hidden layer and context layer, hidden neurons may continue recycling information through the network over multiple time steps so that abstract representation of time can be discovered.

The mathematical model of the simple recurrent neural network is developed in [32] and shown in Figure 5. It can be formulated as follows:

$$v_i(k)=\sum_{j=1}^{N} w_{ij}^{x} x_j^{c}(k)+\sum_{m=1}^{M} w_{im}^{u} u_m(k),\ i=1,\cdots\cdots,N \tag{15}$$

$$x_i(k)=f(v_i(k)) \tag{16}$$

$$x_j^{c}(k)=x_j(k-1) \tag{17}$$

$$y_l(k)=\sum_{i=1}^{N} w_{il}^{y} x_i(k),\ l=1,\cdots\cdots,L \tag{18}$$

where $i=1,\cdots\cdots,N$ is the number of neurons in hidden layer, $j=1,\cdots\cdots,N$ is the number of neurons in context layer, $m=1,\cdots\cdots,M$ is the number of neurons in input layer, and $l=1,\cdots\cdots,L$ is the number of neurons in output layer.

Denote three weight matrices as following:

$$W_x=[w_{ij}^{x}]_{N\times N},\ W_u=[w_{im}^{u}]_{N\times M},\ W_y=[w_{il}^{y}]_{N\times L} \tag{19}$$

then,

$$X(k)=f(W_x X(k-1)+W_u U(k)),\ Y(k)=W_y^{T} X(k) \tag{20}$$

where $Y(k)=[y_1(k),\cdots,y_L(k)]^T$, $U(k)=[u_1(k),\cdots,u_M(k)]^T$, $X(k)=[x_1(k),\cdots,x_N(k)]^T$.

From (15), (16), (17), and (18), the output of SRN can be represented as follows:

$$y_l(k)=\sum_{i=1}^{N} w_{il}^{y} f\left(\sum_{j=1}^{N} w_{ij}^{x} x_j(k-1)+\sum_{m=1}^{M} w_{im}^{u} u_m(k)\right) \tag{21}$$

CHAPTER FOUR

Semigroup Theory

Overview

It is well known that differential equations play an important role in engineering and many areas of social sciences. These equations may have various forms in diverse problems, such as functional differential equations, partial differential equations, and even combination of interacting systems with ordinary and partial differential equations. Semigroup theory can be used to study some problems in the field of partial differential equations. However, some partial differential equations can be regarded as ordinary differential equations on abstract spaces by the semigroup approach. This is the area where semigroup theory demonstrates its usefulness. Additionally, semigroup theory has been extensively applied in the study of Markov process, ergodic theory, and approximation theory [33].

General Group Theory

"A group is a finite or infinite set of elements together with an operation which combines any two elements to form another element" [32]. The set and operation must satisfy four fundamental group axioms including closure, which means the result of operation with any two elements in the group is also in the group; associativity, which means for any elements in the group, $(a \cdot (b \cdot c)) = (a \cdot b) \cdot c)$ holds; identity, which means there exists an element e such that $e \cdot a = a \cdot e = a$ for all elements in the group; and inverse element, which means for any element a, there must be an element b such that

$a \cdot b = b \cdot a = e$. A semigroup differs from a group in that there may not be an inverse for each element nor an identity element. For example, the set of all positive and negative integers forms a group under addition, and only forms a semigroup under multiplication since these does not exist inverse element under multiplication. A more complicated example is the set of all $n \times n$ real matrices under the operation of matrix multiplication. In general, this will represent a semigroup of operators from R^n to R^n.

Theory of Semigroups of Linear Operators

Definition: Let X be a Banach space. A one parameter family $T(t)$, $0 \le t < \infty$, of bounded linear operators from X to X is a semigroup of bounded linear operator on X if

$$\text{(i) } T(0) = I \text{, } (I \text{ is the identity operator on } X) \tag{22}$$

$$\text{(ii) } T(t+s) = T(t)T(s) \text{ for every } t,\ s \ge 0 \text{ (the semigroup property)} \tag{23}$$

A semigroup of bounded linear operators, $T(t)$, is uniformly continuous if

$$\lim_{t \to 0} \| T(t) - I \| = 0 \tag{24}$$

The linear operator A defined by

$$D(A) = \left\{ x \in X : \lim_{t \to 0} \frac{T(t)x - x}{t} \text{ exists} \right\} \tag{25}$$

and

$$Ax = \lim_{t \to 0} \frac{T(t)x - x}{t} = \left. \frac{d^+ T(t)x}{dt} \right|_{t=0} \text{ for } x \in D(A) \tag{26}$$

is the infinitesimal generator of the semigroup $T(t)$, $D(A)$ is the domain of A [34].

Theorem: A linear operator A is the infinitesimal generator of a uniformly continuous semigroup if and only if A is a bounded linear operator [34].

Consider the equation

$$\begin{aligned} &\frac{d}{dt}x(t) = Ax(t) \text{ for } t > 0 \\ &x(0) = x_0 \in R^n \end{aligned} \tag{27}$$

Whether A is a constant operator or a time varying operator, it is always a bounded operator; therefore belongs to the space of bounded operators that map R^n to R^n, denoted by $B(R^n)$. In this case, A is just an $n \times n$ matrix. Because $B(R^n)$ is an algebra, (not merely a vector space) sums and products of A belong to $B(R^n)$. Therefore, the exponential matrix $e^{At} = \sum_{k=0}^{\infty} \frac{(At)^k}{k!}$ is well defined, $e^{At} \in B(R^n)$ for all $t \in R$ since the series convergences absolutely in the Banach space $B(R^n)$. Then, for all t:

$$\begin{aligned} \frac{d}{dt}e^{At}x_0 &= \lim_{h\to 0}\frac{1}{h}(e^{A(t+h)}x_0 - e^{At}x_0) \\ &= \lim_{h\to 0}\frac{1}{h}\left(\lim_{N\to\infty}\left(\sum_{k=0}^{N}\frac{(t+h)^k - t^k}{k!}A^k\right)x_0\right) \\ &= \lim_{N\to\infty}\left(\sum_{k=1}^{N}\frac{t^{k-1}}{(k-1)!}A^k\right)x_0 \\ &= \lim_{N\to\infty} A\sum_{k=0}^{N-1}\frac{(At)^k}{k!}x_0 \\ &= Ae^{At}x_0 \end{aligned} \tag{28}$$

This shows that $x(t) = e^{At}x_0$ is the solution to (27). Note that the operator $T : R^n \to R^n$, given by $T(t) = e^{At}$, possesses the following properties:

$$T(0) = I \tag{29}$$

$$T(t+s) = T(t)\,T(s), \text{ for all } t,\ s \tag{30}$$

Another property is that the mapping $t \to T(t)x_0$ from R into X is differentiable for all $x_0 \in X$. Notice that the procedure can be reversed. That is if the set $T(t)$ exists with the properties of (29) and (30), then it generates A, i.e.,

$$\lim_{t\to 0} A_t = \lim_{t\to 0} \frac{T(t)-I}{t} = \lim_{t\to 0} \frac{T(t)-T(0)}{t} \equiv \frac{d}{dt}[T(t)]\Big|_{t=0} = \frac{d}{dt} e^{At}\Big|_{t=0} = A \tag{31}$$

where $A_t = \dfrac{T(t)-I}{t}$, and $T(t)$ is termed as the infinitesimal generator of A. If we allow A to be any bounded operator on a Banach space X, exactly the same calculation will show that if $e^{At}x_0$ solves the equation

$$\begin{aligned} &\frac{d}{dt}x(t) = Ax(t) \\ &x(0) = x_0 \in X \end{aligned} \tag{32}$$

then it is straightforward to represent the solution to the equation

$$\begin{aligned} &\frac{d}{dt}x(t) = Ax(t) + f(t) \\ &x(0) = x_0 \in X \end{aligned} \tag{33}$$

where $f : R \to X$ is continuous, by the variation of parameters formula $x(t) = e^{At}x_0 + \int_0^t e^{A(t-s)} f(s)ds$ [32].

Partial Differential Equations

To give an idea of a semigroup property being possessed by a mapping involving a PDE, consider the following steady state heat flow model in Cartesian coordinates:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{34}$$

If we set $T(x,y)=C(y)^T E(x)=c_1(y)e_1(x)+c_2(y)e_2(x)$, where $C(y)=[c_1(y),c_2(y)]^T$, $E(x)=[e_1(x),e_2(x)]$, and e_i are orthonormal basis, then by substitution, $\ddot{c}_1e_1+\ddot{c}_2e_2=-c_1\ddot{e}_1-c_2\ddot{e}_2$. This, in turn implies

$$\begin{aligned}\ddot{c}_1+\ddot{c}_2<e_2,e_1>&=-c_1<\ddot{e}_1,e_1>-c_2<\ddot{e}_2,e_1>\\ \ddot{c}_1<e_1,e_2>+\ddot{c}_2&=-c_1<\ddot{e}_1,e_2>-c_2<\ddot{e}_2,e_2>\end{aligned} \text{ or } \begin{bmatrix}\ddot{c}_1\\ \ddot{c}_2\end{bmatrix}=A\begin{bmatrix}c_1\\ c_2\end{bmatrix} \qquad (35)$$

for a suitable matrix A, which leads to a semigroup for $C(y)$. Notice that the semigroup property is not reflected in the original data; it appears only in the coefficient vector portion of the system description [32].

CHAPTER FIVE

Short-Term Load Forecasting Using System-Type Neural Network Architecture

Neural Network Architecture

Neural networks are being applied for distributed parameter systems (DPS), i.e., described by PDE's [35]. In previous papers [36]-[41], a system-type neural network had been proposed for implementing extrapolation. With this method, the distributed parameter system (DPS) surface determined by a given data set was expanded by being extrapolated along one axis to predict data in the unknown region. With respect to load forecasting, the load is in general a function, $Load = f(Day, Hour, Weather, Customer\,classes)$, and is often considered as $Load = f(Day, Hour)$, which is parameterized by weather and customer classes [41]. In this thesis, however, the load is modeled as $Load = f(Temperature, Hour)$ due to the importance of the temperature among many factors affecting the load. It will be shown that the load profile can be formulated in this form: $L(Temperature, Hour) = C(T)^T E(H)$.

Figure 6 shows the system-type architecture of the proposed neural network which implements an arbitrary load function $L(T, H)$. Instead of using one neural network to realize the mapping $L(T, H)$ like those conventional neural networks, the proposed architecture shows a system-type approach with two neural network channels, a Function Channel and a Semigroup Channel. The Function Channel outputs a vector of basis functions $E(H)$, while the Semigroup Channel supplies the Function Channel with a smoothened coefficient vector $\tilde{C}(T)$ as a function of the index T. Applying the

coefficient vector to the basis set $E(H)$ from the Function Channel causes the Function Channel to operate as one specific load function within a vector space of functions. These two channels bring about a semigroup-based implementation of the mapping in the following form:

$$L(T,H)=\tilde{C}(T)^T E(H) \tag{36}$$

where $\tilde{C}(T)=[\tilde{c}_1(T),\cdots,\tilde{c}_n(T)]$, $E(H)=[e_1(H),\cdots,e_n(H)]$.

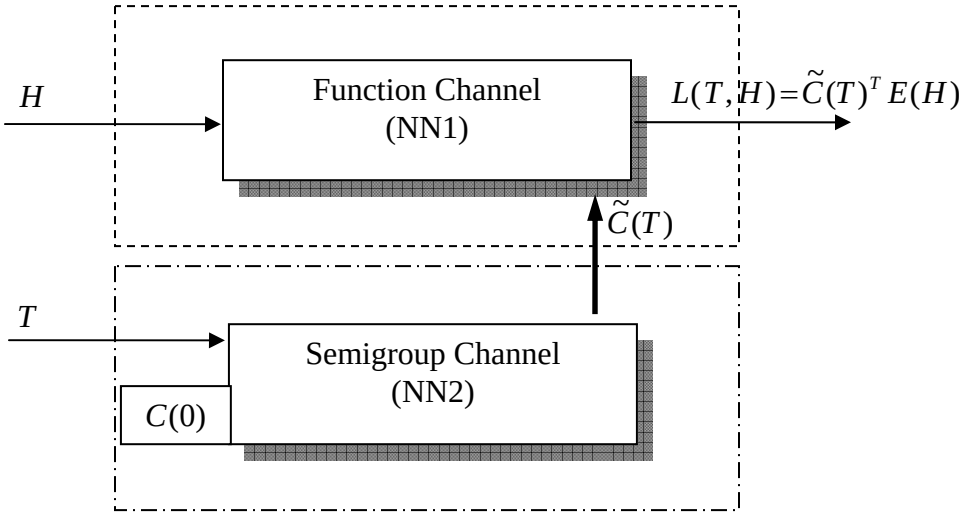

Figure 6. System-type architecture [36].

Function Channel

The Function Channel is of Radial Basis Function (RBF) architecture [19]. The Channel, in which there are n RBF networks, concerns with implementing each of the n orthonormal basis functions in $E(H)$ by each network. The outputs of the two channels are (internally) linearly summed so as to span an *n*-dimensional function space. Up to this step, the operation of the Function Channel is identical with the idea used by Phan and Frueh [42]. However, there are some essential differences between their approach and the

proposed approach, of which one is that the former requires prior engineering knowledge to select the basis vectors, and the presented approach does not. The RBF network is chosen because of several advantages it has, which excel other architectures. One advantage is that the functionality of RBF network can be given an explicit mathematical expression in which the neuron activation functions operate as Green's functions. Another advantage is that they function as universal approximators [19]. And also RBF networks can be designed rather than trained. The well known universal approximation theorem of RBF networks is given as follows.

Theorem (Universal Approximation Theorem): For any continuous input-output mapping function $f(x)$ there is a RBF network with a set of centers $\{t_i\}_{i=1}^{m_1}$ and a common width $\sigma > 0$ such that the input-output mapping function $F(x)$ realized by the RBF network is close to $f(x)$ in the L_p norm, $p \in [1, \infty]$ [43].

The universal approximation theorem for *n*-RBF networks is derived by Kim [32]. In the following Corollary, for any positive integer r, R^r denotes the normed linear space of real *r*-vectors and $L_p(R^r)$ denotes the usual spaces of *R*-valued functions f defined on R^r such that f is *p*-th power integrable and N is the number of RBF networks. Let S_{K_i} be the family of RBF networks consisting of functions $q_i : R^r \to R$ defined by

$$q_i(x) = \sum_{j=1}^{M} w_j K_i\left(\frac{x - z_j}{\sigma}\right), \; i = 1, \cdots, N \tag{37}$$

where $M \in \mathbb{N}$, $\sigma > 0$, $w_j \in R$, and $z_j \in R^r$ for $j = 1, \cdots, M$, and $\mathbb{N}$ is the natural number set. Let S be the family of linear combination of $\{S_{K_1}, \cdots, S_{K_N}\}$. Then for any $q \in S$,

$$q(x)=\sum_{i=1}^{N} q_i(x)=\sum_{i=1}^{N}\sum_{j=1}^{M} w_j K_i\left(\frac{x-z_j}{\sigma}\right) \tag{38}$$

Corollary: Suppose that for all $i\in\{1,\cdots,N\}$, N is the number of RBF networks, $K_i : R^r \to R$ is an integrable bounded function such that K_i is continuous almost everywhere and $\int_{R^r} K_i(x)dx\neq 0$. Then the linear combination of the family S is dense in $L_p(R^r)$ for every $p\in[1,\infty)$" [32].

Semigroup Channel

The Semigroup Channel can be adapted based upon the Diagonal Recurrent Neural Network (DRNN) [30] or the Simple Recurrent Network (SRN) architecture [31]. In this thesis, the SRN is applied in the Semigroup Channel. The input, that is the preliminary coefficient vector $C(T)$, is separated into a dynamic scalar component T and one static vector component, the initial vector $C(0)$. The output of the channel is a smoothened vector $\tilde{C}(T)$, which is associated with the dynamic input T and the static input $C(0)$ by the semigroup property: $\tilde{C}(T)=\Phi(T)C(0)$, where $\Phi(T_1+T_2)=\Phi(T_1)\Phi(T_2)$.

The internal weight structure of the Semigroup Channel is shown in Figure 7 which includes four weight spaces in the SRN: input weight corresponding to dynamic component W_i, input weights corresponding to static component W_c, feedback weights W_{fb}, and output weights W_o.

The following mathematical representations from [32] illustrate the nature of the weight pattern convergence with reviewing the SRN architecture in Figure 5. Let the

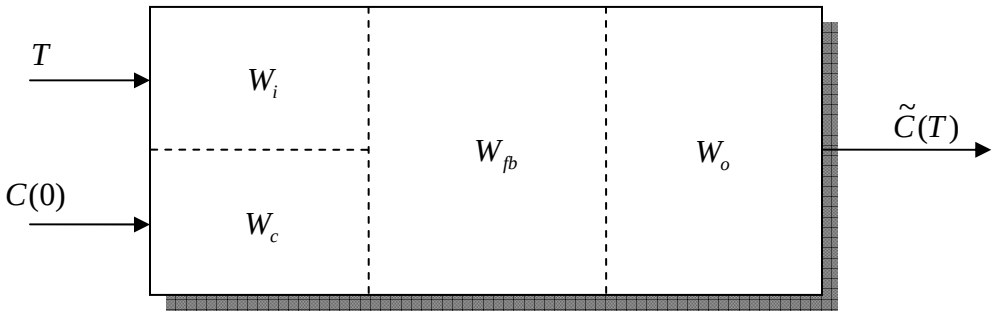

Figure 7. Internal weight architecture for Semigroup Channel [32].

error function be defined as follows, where p is the number of input data and $p_{\min} < p < p_{\max}$:

$$\begin{aligned} e_l(k) &= y_{d,l}(k) - y_l(k) \\ E^p &= \frac{1}{2}\sum_{l=1}^{L}[e_l(k)]^2 \\ E &= \sum_p E_p \end{aligned} \tag{39}$$

where y_d is desired output vector and $l = 1, \cdots, L$, and L is the number of neurons in output layer.

Based on the gradient-descent method, the following equations are obtained by differentiating the error function with respect to feedback, input, and output weights w_{ij}^x, w_{im}^u, and w_{il}^y, respectively in the SRN:

$$-\frac{\partial E}{\partial w_{il}^y} = \frac{\partial E}{\partial y_l(k)}\frac{\partial y_l(k)}{\partial w_{il}^y} = -\sum_p (y_{d,l}(k) - y_l(k))x_i(k) \tag{40}$$

$$-\frac{\partial E}{\partial w_{im}^u} = \frac{\partial E}{\partial y_l(k)}\frac{\partial y_l(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial v_i(k)}\frac{\partial v_i(k)}{\partial w_{im}^u} = -\sum_p [(y_{d,l}(k) - y_l(k))w_{il}^y]\frac{\partial x_i(k)}{\partial v_i(k)}u_m(k) \tag{41}$$

$$-\frac{\partial E}{\partial w_{ij}^x} = \frac{\partial E}{\partial y_l(k)}\frac{\partial y_l(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial v_i(k)}\frac{\partial v_i(k)}{\partial w_{ij}^x} = -\sum_p [(y_{d,l}(k) - y_l(k))w_{il}^y]\frac{\partial x_i(k)}{\partial v_i(k)}x_j(k-1) \tag{42}$$

From (40), (41) and (42), since $\Delta w = -\eta \frac{\partial E}{\partial w}$,

$$\begin{aligned}
\Delta w_{il}^{y} &= \eta \sum_{p} (y_{d,l}(k) - y_{l}(k)) x_{i}(k) \\
\Delta w_{im}^{y} &= \eta \sum_{p} [(y_{d,l}(k) - y_{l}(k)) w_{il}^{y}] \frac{\partial x_{i}(k)}{\partial v_{i}(k)} u_{m}(k) \\
\Delta w_{ij}^{x} &= \eta \sum_{p} [(y_{d,l}(k) - y_{l}(k)) w_{il}^{y}] \frac{\partial x_{i}(k)}{\partial v_{i}(k)} x_{j}(k-1)
\end{aligned} \tag{43}$$

After the weight pattern converges, the input can be split into a static component $u(0)$, and a dynamic component k, which corresponds to the static and dynamic weights, respectively. Therefore, (21) can be reformulated as follows by assuming the first input neuron is always used for the dynamic component:

$$y_{l}(k) = \sum_{i=1}^{N} w_{il}^{y} f\left(\sum_{j=1}^{N} w_{ij}^{x} x_{j}(k-1) + \sum_{m=2}^{M} w_{im}^{u,static} u_{m}(0) + w_{im}^{u,dynamic} k \right) \tag{44}$$

The Proposed Training Method

The first component of the system, the Function Channel, can be designed rather than trained due to the attribute of RBF networks. The first and significant step is determining the algebraic dimensionality of the orthonormal basis set $E(H)$, say n, then n RBF networks need to be designed to emulate those n basis functions which are the generations of the algebraic decomposition.

The second component, the Semigroup Channel, needs to be trained in a successive way shown in Figure. 8. In each of trainings, the Semigroup Channel receives a preliminary coefficient vector $C(T)$ as input composed by the dynamic scalar component and the static initial vector component, and outputs a smoothened coefficient

vector $\tilde{C}(T)$. Therefore the primary objective of training in Semigroup Channel is to replicate and smoothen the vector $C(T)$ with a vector $\tilde{C}(T)$, which has the semigroup property: $\tilde{C}(T)=\Phi(T)\tilde{C}(0)$, where $\tilde{C}(T)=[\tilde{c}_1(T),\tilde{c}_2(T),\cdots,\tilde{c}_n(T)]$ and $\Phi(T)$ is an $n\times n$ matrix that satisfies: $\Phi(T_1+T_2)=\Phi(T_1)\Phi(T_2)$.

However, there is a secondary objective of training requiring that the channel must also replicate the semigroup property of the trajectory by gradually acquiring a semigroup property of its own in the weight space. In order to obtain this gradual acquisition of the semigroup property, the training should occur in a gradual manner, as shown in Figure 8. Note that N is the number of data points in the coefficient vector. According to Figure 8, the proposed training method slices the entire trajectory into many nested sub-trajectories, each of which is composed by the previous sub-trajectory and one new data point. Therefore two kinds of convergence will happen in the training. Firstly during training of each sub-trajectory, the network weights, W_i, where $i=1,\cdots,N$, must converge. This means that the SRN is able to duplicate the sub-trajectory up to this data point. This convergence must reoccur for each subsequent training step. At certain future point, besides the weight convergence at each step, there will be a convergence in the overall pattern of weights. This second convergence will be referred to as the "the weight pattern convergence" which actually is the basis of extrapolation.

In this proposed training approach, each sub-trajectory is trained using a conventional (batch) method, and the resulting weight is recorded. After all sub-trajectories have been trained, the sequence of resulting weights is examined for the purpose of the weight pattern convergence. Only if the weight pattern convergence is achieved, can extrapolation start.

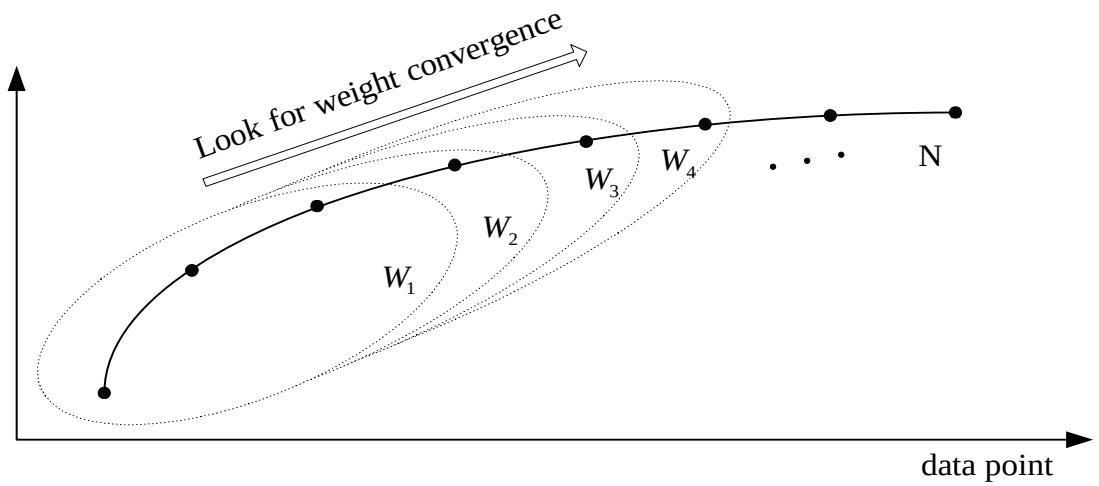

Figure 8. Overview of new training algorithm [37].

System Modeling

In the previous applications [36]-[41], system modeling is realized by a technique referred to as algebraic decomposition which aims to approximate and model the given data set. It is a mathematical operation which represents the given function $L(T,H)$ in such a form: $L(T,H)=L_T(H)=C(T)^T E(H)$, where $L_T(H)$ is the parameterized set of $L(T,H)$. The generation of this process is the coefficient vector $C(T)^T=[c_1(T),c_2(T),\cdots,c_f(T)]^T$ which is a representation of the parameterized function $L_T(H)$ with respect to $E(H)$, and also $E(H)$ provides the algebraic basis for the representation of each element in this parameterized function. Technically algebraic decomposition starts by finding a low dimensional basis set whose sources are directly drawn from the given family of functions $\{L_T(H)\}$. For example, $\{L_{T=1}(H),L_{T=6}(H),L_{T=9}(H)\}$ is found to form a basis set such that any arbitrary element $L_{T=i}(H)$ can be expressed as a linear combination of these basis:

$$L_{T=i}(H)=c_1(T)L_{T=1}(H)+c_2(T)L_{T=6}(H)+c_3(T)L_{T=9}(H) \quad (45)$$

In a word, algebraic decomposition is the process that forms an approximation surface $\tilde{L}(T,H)$ to $L(T,H)$ and can be summarized in following steps: (1) Parameterize $L(T,H)$ as $\{L_T(H)\}$, $T=T_i,\cdots,T_f$. (2) Determine the dimensionality of $\{L_T(H)\}$ as n. (3) Choose n elements from $\{L_T(H)\}$ and orthonormalize them to form a basis set $E(H)=[e_1(H),e_2(H),\cdots,e_n(H)]$ using the Gram-Schmidt process. (4) For each element of the parameterized function family $\{L_T(H)\}$, determine the particular linear combination of the basis set using the least squares method. This step determines $C(T)$, and the product of $C(T)$ and $E(H)$ provides the approximation $\tilde{L}(T,H)$ of the original load function, where $\tilde{L}(T,H)=L_T(H)=C(T)^T E(H)$.

Regression Method

Regression method is one of the most widely used approaches for load forecasting. Usually the forecasting model is developed by identifying a normal or weather-insensitive load component and a weather-sensitive load component. Regression method can be applied to each component separately or to the total load. However, regression method is not playing the leading role to perform forecasting in this thesis. It is well known that the electric load in a complex system is influenced by many factors. It can be represented as the following form:

$$L_{total}=L_{base}+L_{weather}+L_{other\ factors} \quad (46)$$

where L_{base} is the base load component which is caused by time factor; $L_{weather}$ is the weather sensitive load component which is due to weather variables such as temperature,

dew point, wind speed, cloud cover, etc. Temperature is usually dominant among various weather variables; and $L_{other\ factors}$ is a component of the load resulting from other factors.

In terms of parameterization of the given load, load data is represented as a function with respect to two major variables, time and temperature. Therefore, regression method is used here to filter the load and remove the load component caused by other factors. In this thesis, following regression formulas are used to perform filtering:

$$L_i(t) = A + B \cdot T_d(t) \tag{47}$$

$$\text{or } L_i(t) = A + B \cdot T_d(t) + C \cdot T_d(t)^2 \tag{48}$$

where $L_i(t)$ is the load at hour t in *i*-th day; A, B, and C are the regression coefficients which are assumed constant for different time intervals; $T_d(t)$ is the temperature at hour t, in deg. F. Therefore, using regression method is to make the original load more correlated to time and temperature, that is, to form the function $L(T, H)$, which is ready to be decomposed.

Rearrangement of Load

Based on the research done by Kim, Velas, and Lee [44], the smoothness of the given data surface is the prerequisite for performing the extrapolation. They tested two different surfaces, one is a bivariate sinusoidal function with smoothness and another is the shape of pyramid including sharp edges. The results turn out that the modeling of the two surfaces can be perfectly achieved by algebraic decomposition, but extrapolation of the coefficient vector can only be implemented for the first surface due to the smoothness of the coefficient vector. Regarding the pyramid surface, algebraicly decomposing the surface with sharp transitions and discontinuities generates the coefficient vector with the

same attributes. Therefore the extrapolation in the second case is impossible on account of the discontinuities in the coefficient vector.

As mentioned above, the load is already filtered to be more correlated to time and temperature. The filtered load, namely regression load, rises and falls mainly due to the fluctuation of the hourly temperatures for different days. The differences in temperatures for a given time (hour), which bring about differences in electric load, result in an non-smooth load surface. Therefore, it is necessary to rearrange the regression load according to the hourly temperatures so that a smooth load surface can be obtained. Rearrangement of the load is applied for each hour based on either increasing or decreasing temperatures. Figure 9 illustrates the rearrangement of the load based on the temperature for the first hour. The same step can be applied for other hours. The darkest circle represents the load data with the highest temperature.

Extrapolation Test

The semigroup property should be ultimately realized in the Semigroup Channel as a sequence of weight changes that occurs after the weight pattern convergence takes place. The sequence of weight changes must follow a rule that leads to the semigroup property. A permissible rule for the j^{th} weight change, $\Delta w_j(k+1)=\alpha_j \cdot \Delta w_j(k)$, has been successfully applied in the previous applications [36]-[41]. This rule can derive the form $\Delta w_j(k)=\alpha_j^k \cdot \Delta w_j(0)$ which has the semigroup property.

However, before performing the extrapolation using this weight change rule with semigroup property, there are still two more steps needed to accomplish. First step is to confirm that the weight pattern convergence already happened in the Semigroup Channel. Identifying this convergence can be completed based upon observation of the following

two facts: (1) the output weights, the feedback weights, and those input weights with respect to the static input component undergo no significant changes for all steps after the weight pattern convergence occurs; (2) the only weight changes springs from those input weights $w_i^{dynamic}$ which associates with the dynamic input component, that is, these weights must vary by differential amounts. Therefore, this results in $w_i^{dynamic}(k+1)=(w_i^{dynamic}(k)+\Delta w_i(k))$ where $\Delta w_i(k)$ is the change between $w_i^{dynamic}(k+1)$ and $w_i^{dynamic}(k)$.

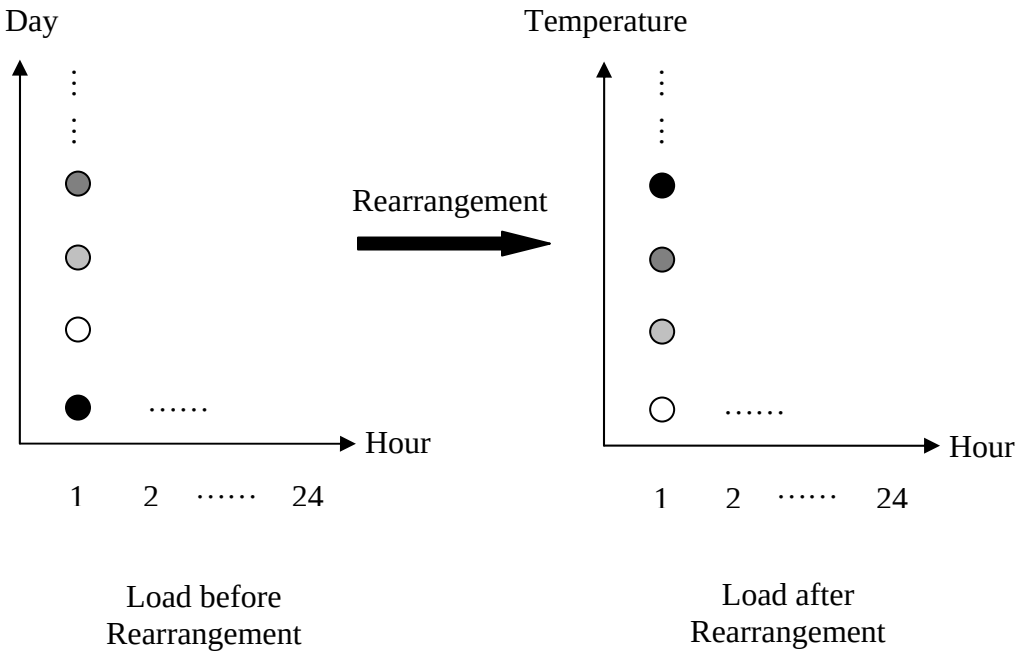

Figure 9. Rearrangement of the regression load.

Another step right before the extrapolation is the validation of the proposed weight change rule for the dynamic input component, and this step is called extrapolation test. In order to complete this, M consecutive points in the coefficient vector are selected as an observation window. Based on the assumptions that the actual weight change

produced by training for weight " j " at point " k " is $\Delta w_j(k)$, and at the start point of the observation window there is an initial weight change of $\Delta w_j(0)$, a weight change sequence approximation is computed by

$$\Delta w_j(k+1) = \alpha_j \Delta w_j(k) \tag{49}$$

To check if this estimated weight change sequence can replace the actual weight change sequence which is produced by training the simple recurrent network over the observation window, the two trajectories of coefficient vector, which are the simulation outputs of the SRN using the actual and estimated weight change sequences, should be compared. If the error between the two trajectories is small, the estimated weight change sequence rule is acceptable.

Thus, from this test, after the weight pattern converges, the weight change for the dynamic input component k can be substituted for $\Delta w_i(k) = \alpha_i^k \Delta w_i(0)$, where the α_i^k satisfies the semigroup property. Note that here $\Delta w_i(0)$ represents the weight change immediately following the convergence which has the initial weight $w_i(0)$. Therefore, the final output of Semigroup Channel in (43) becomes

$$y_l(k) = \sum_{i=1}^{N} w_{il}^{y} f\left(\sum_{j=1}^{N} w_{ij}^{x} x_j(k-1) + \sum_{m=2}^{M} w_{im}^{u,static} u_m(0) + \left(w_{i1}^{u,dynamic}(0) + \alpha_i^k \Delta w_{i1}^{u,dynamic}(0) \right) k \right) \tag{50}$$

Extrapolation

Extrapolation involves only the coefficient vector and the Semigroup Channel. At the uppermost level, the idea is to train the neural network to replicate the coefficient vector in such a way that it is additionally replicating the semigroup property, which is responsible generating the coefficient vector by acquiring a semigroup property of its

own in the weight space. However, the implementation of extrapolation is not completely same with previous works [36]-[41]. Because the smoothened coefficient vector $\tilde{C}(T)$ is a function of the index T and the temperature forecasting for the next day is assumed to be known already, the need to extrapolating coefficient vector totally depends on whether or not the temperature forecast at a given hour exceeds the historical temperature bounds in the same hour group. If it does, the coefficient vector which is going to be supplied to the Function Channel can be obtained through extrapolation. The load forecasting at this hour is achieved by recombining the extrapolated coefficient vector with the basis set. Forecasting load for other hours follows the same procedure when extrapolation is necessary. Figure 10 shows the extrapolation of the coefficient vector for the first hour. The white circle represents historical load data. The circle with stripes represents the load data to be forecasted.

Interpolation

Since the system for load forecasting is considered as DPS, the smoothened coefficient vector $\tilde{C}(T)$ can be counted as a continuous function with the variable of temperature, although the vector is composed of individual points. Therefore, the interpolation of coefficient vector based on the hourly temperatures can be performed when the temperature forecast at a given hour falls within the historical temperature range in the same hour group. The load forecasting at this hour is acquired by recombining the interpolated coefficient vector with the basis set. Forecasting load for other hours follows the same procedure if interpolation is needed. Figure 11 shows the interpolation of the coefficient vector for the first hour. The white circle represents historical load data. The circle with stripes represents the load data to be forecasted.

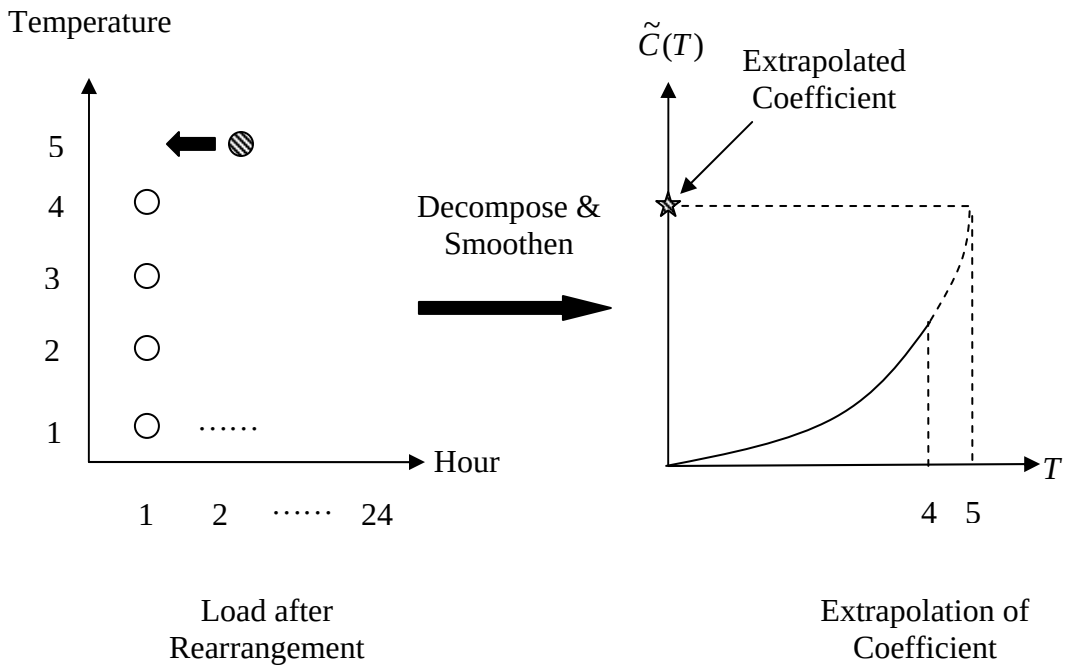

Figure 10. Extrapolation.

Summary of Processes

The following is to give a summary of the processes required for short-term load forecasting using the proposed approach. Assume that a load data set is available.

Step 1: Regression. Let the original load data go through the specified regression method to obtain the regression load in the defined time interval.

Step 2: Rearrangement. Arrange the regression load along the temperature coordinate. That is, rearrange the regression load $L(Day, Hour)$ to $L(Temperature, Hour)$ based on the magnitudes of temperatures for 24 different hours.

Step 3: Algebraic Decomposition. Choose *n* vectors from the parameterized set $\{L_T(H)\}$ to form a set of basis vectors $\{v_1, \cdots, v_n\}$. Orthonormalize each vector $\{v_1, \cdots, v_n\}$ using Gram-Schmidt procedure to form the orthonormal set of basis vectors $E(H)=[e_1(H), e_2(H), \cdots, e_n(H)]$. Design a radial basis function (RBF) network for each

of the orthonormalized vectors in the basis set. The number of hidden neurons is equal to the length of each basis vector. Determine the preliminary coefficient vector $C(T)$ using the least squares method.

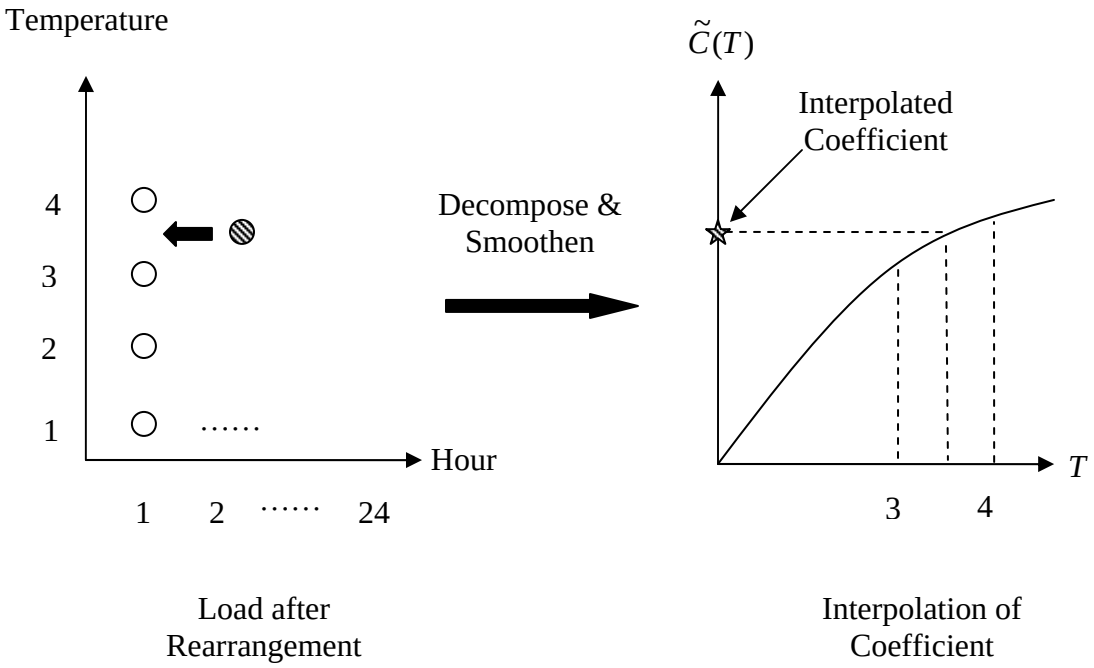

Figure 11. Interpolation.

Step 4: Semigroup Channel. Design a simple recurrent network (SRN) with static inputs $C(0)$ and dynamic input T. It has been found through experimentation that the number of hidden neurons is approximately $25\times$(number of basis vectors + 1). Train the SRN in the gradual manner that increases the number of input points by 1 in each training.

Step 5: Determine whether or not extrapolation or interpolation is needed. Based upon the relationships of the temperature forecasts and historical temperature among 24 hours, the need of extrapolation or interpolation can be decided. If extrapolation is

required, then continue the following steps. Otherwise, skip to Step 8 if only interpolation is required.

Step 6: Weight pattern convergence check. Check if the weight pattern of the SRN converges. Then perform extrapolation test to confirm if the test result is good.

Step 7: Extrapolation. Extrapolate the smoothened coefficient vector $\widetilde{C}(T)$.

Step 8: Recombination. Multiply the interpolated or extrapolated $\widetilde{C}(T)$ by the set of basis vector $E(H)$ for each individual hour, and then only keep the forecasting load value at the corresponding hour after each multiplication. Group the 24 hours' forecasting load values to form the final one-day-ahead forecasting result.

CHAPTER SIX

Simulation Studies

Forecasting Procedure

The proposed forecasting approach is tested by using the past load profile obtained from New England Independent System Operator (ISO). The hourly temperatures of each day are weighted average values of 8 weather stations in the New England area in degrees Fahrenheit. In the simulation, load data for the year 2002 is chosen for demonstrating the capability of the proposed approach. The simulation uses load and temperature data in a moving window of previous four weeks for each forecasting target day. Usually the load profile has two distinct patterns: weekday and weekend patterns. In general, Monday load is classified to weekend pattern which includes Saturday and Sunday since the level of Monday load in the early morning is low influenced by Sunday load. However, in this thesis Monday load is grouped to weekday pattern due to its similarity to weekday pattern comparing with weekend pattern. Therefore, previous weekday or weekend pattern load in the window of four weeks are selected as historical data to forecast next weekday or weekend load, respectively.

Regression

Here we choose an arbitrary weekday as the forecasting day to show the simulation procedures. The weekdays' data of previous four weeks forms the moving window. At the very beginning, the actual raw data in the moving window should pass through the regression filter to generate regression load which is ready to be rearranged

and decomposed. Figure 12 shows the actual load at 8am on this chosen day. Then it is found that using $L_i(t) = A + B \cdot T_d(t) + C \cdot T_d(t)^2$ can perform well in the curve fitting based on the least squares method. Figure 13 illustrates the actual load and the regression load after the filtering. It shows that the regression method not only removes the load caused by other unknown factors but also makes up for the load component which is offset by other factors.

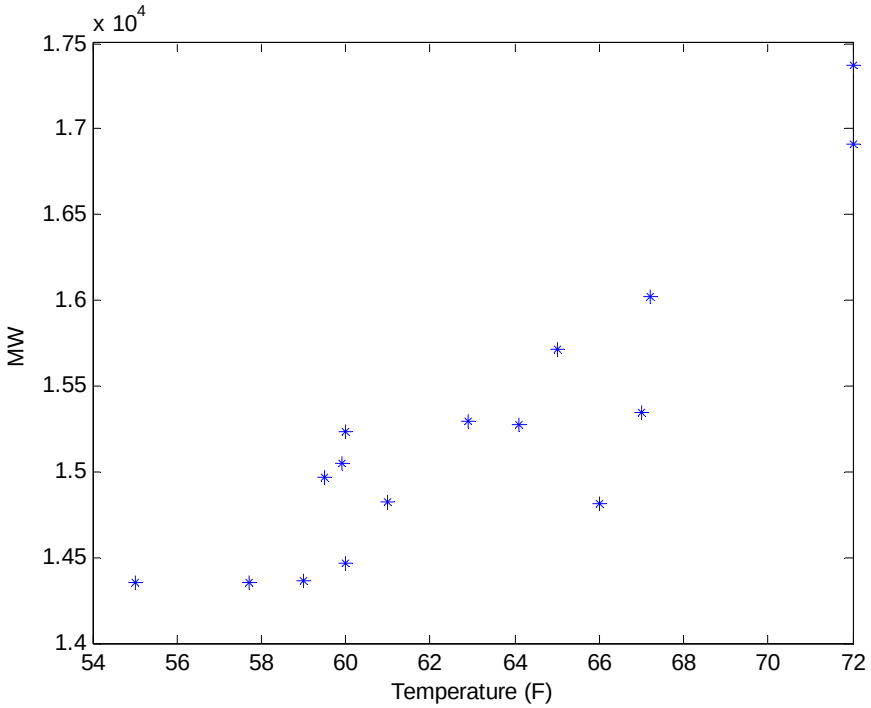

Figure 12. Actual load at hour 8.

Rearrangement

Instead of the regression load in the form $L(Day, Hour)$, the load in the form of $L(Temperature, Hour)$ is preferred because the historical load surface needs to be smooth as much as possible. Therefore the objective of the process called rearrangement

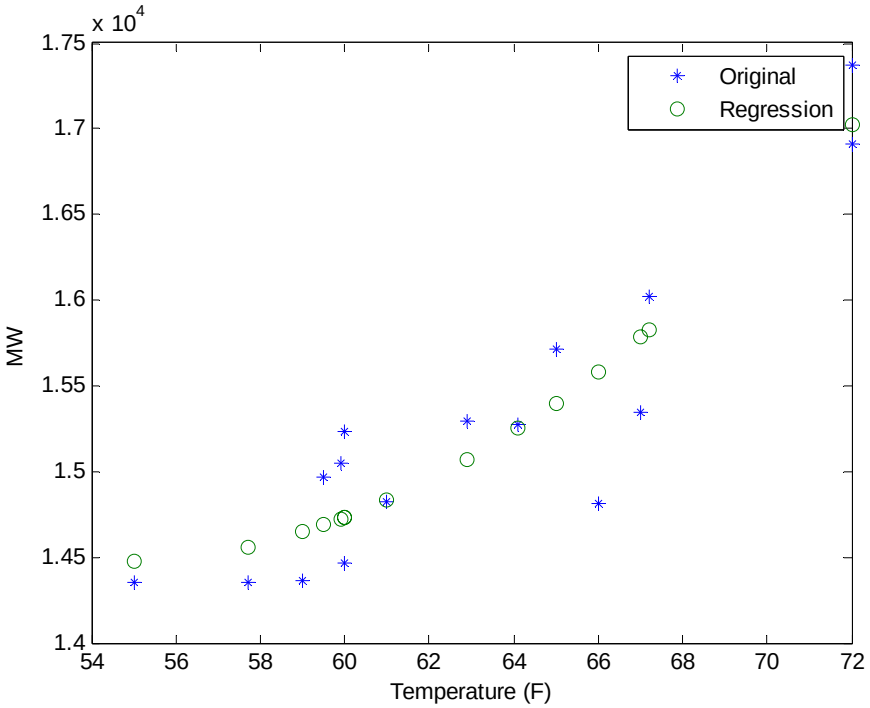

Figure 13. Actual load and regression load at hour 8.

is to sort the load data for each hour along the temperature coordinate to meet the smoothness requirement. At each individual hour, the loads of the historical days are sorted in temperature ascending or descending order. Figure 14 and Figure 15 show the regression load before and after the rearrangement, respectively. The one after the rearrangement is much smoother than that before this process since regression load which is highly correlated to the hourly temperatures is sorted by temperatures. Then a success of acquiring a smooth coefficient vector is expected.

Implementation of the System-Type Neural Network

In this application, the regression load profile $L(T,H)$ is parameterized as $\{L_T(H)\}$, where for each T there are 24 points corresponding to hours $H=1,2,\cdots,24$. Following the implementation procedure, two vectors $\{L_{T=i}(H)\}$ are chosen as the basis

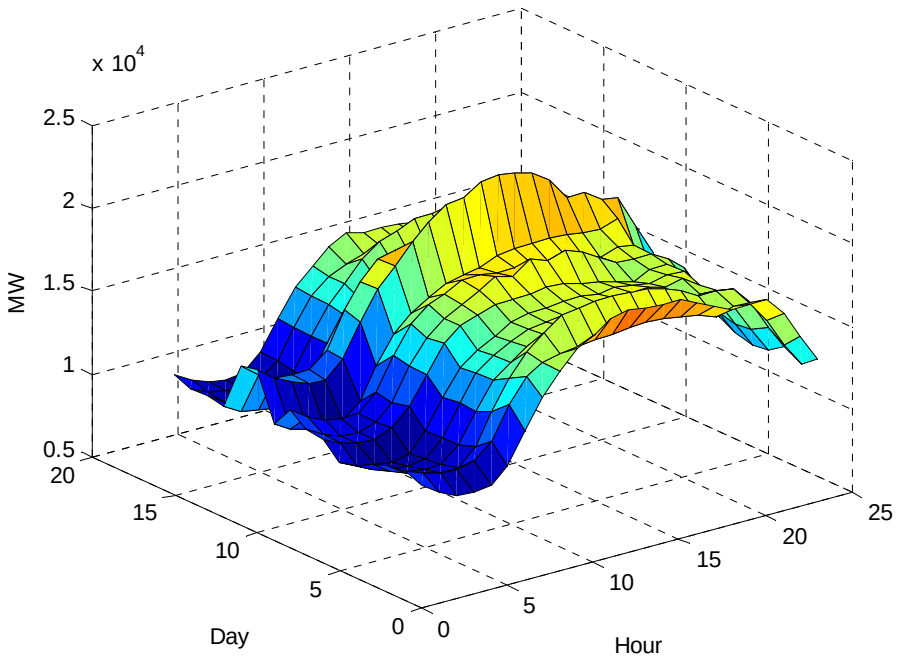

Figure 14. Regression load before rearrangement.

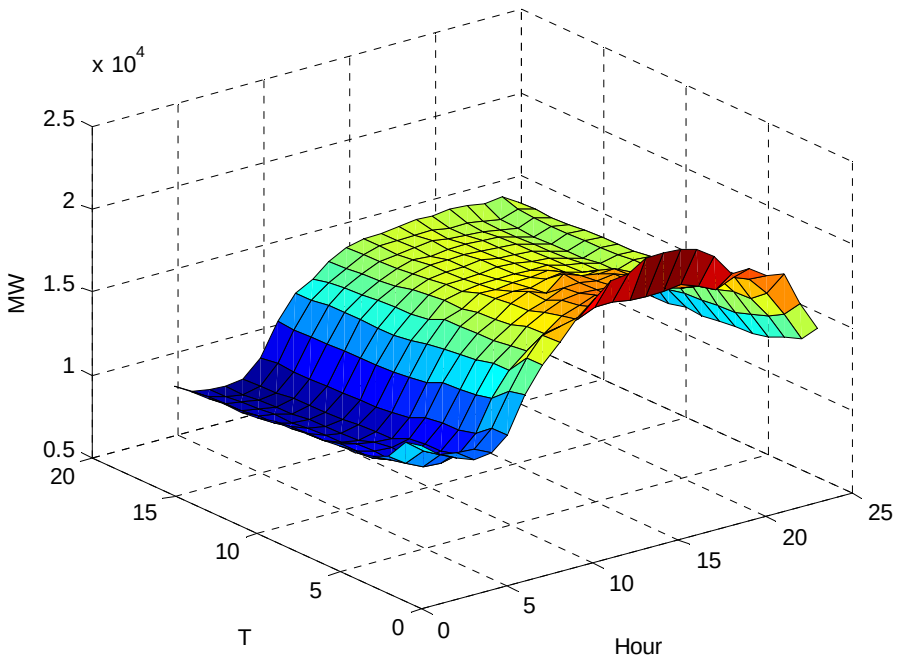

Figure 15. Regression load after rearrangement.

vectors to procure a good duplicate of the regression load, that is, the dimensionality n is two. The larger the n is, the more precise the modeling is. But the reason why n is set to two here is that the coefficient vector will be non-smooth when n is equal or greater than three. With these basis vectors, the coefficient vector is obtained using the least squares method. Therefore, the Function Channel consists of two RBF networks. The number of hidden neurons in each RBF network is 24 because the length of the basis vector is 24, corresponding to the 24 hours. Then each individual RBF network is used to implement each basis vector. All regression load data have been used for duplicating the empirical load in the form of the product of coefficient vector and the set of basis vectors. The Semigroup Channel has one SRN with 3 input neurons, one for a dynamic scalar component T and other two for the static coefficient vector $C(0)=[c_1(0), c_2(0)]$. The output of the SRN is the smoothened coefficient vector $\tilde{C}(T)=[\tilde{c}_1(0), \tilde{c}_2(0)]$. Since the number of input is 3, the number of hidden neurons in the SRN is chosen as $25\times 3 = 75$.

Algebraic Decomposition

The chosen set of basis vectors will be orthonormalized to produce the orthonormal set of basis vectors: $\{e_1(H), e_2(H)\}$. The preliminary (rough) coefficient vector $C(T)$ is produced by the algebraic decomposition. The preliminary coefficient vector is shown in Figure 16 and Figure 17. Figure 18 and Figure 19 show the basis vectors produced by the RBF networks. The product of this rough coefficient vector together with the set of basis vectors will produce the computed regression load which is the modeling of the empirical regression load. Figure 20 shows the computed regression load. The errors between empirical and computed regression load is shown in Figure 21,

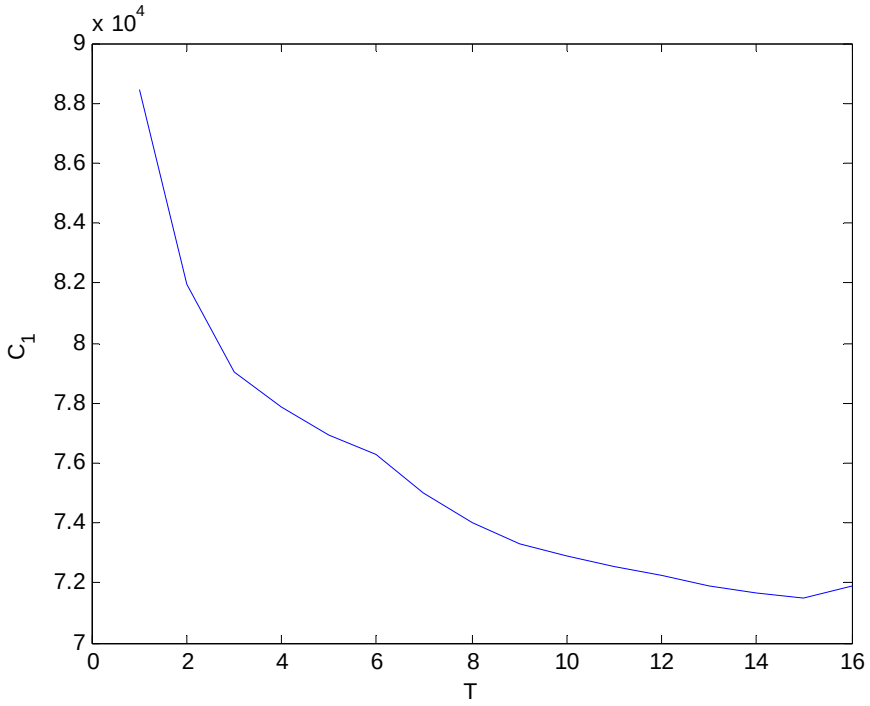

Figure 16. Preliminary coefficient C_1.

and Figure 22 shows the percent error. The average percent error is 0.92% which explains the sufficiency of the algebraic decomposition.

Semigroup Channel Smoothing

The preliminary coefficient vector needs to be smoothened. Therefore, the SRN is trained using the proposed successive training algorithm with the initial coefficient vector $C(T)=[c_1(0),c_2(0)]$ and dynamic scalar component T as the input and the preliminary coefficient vector $C(T)=[c_1(T),c_2(T)]$ as output. Figure 23 and Figure 24 show the smoothened coefficient vector generated by the trained SRN. The preliminary coefficient vector is well replicated and smoothened.

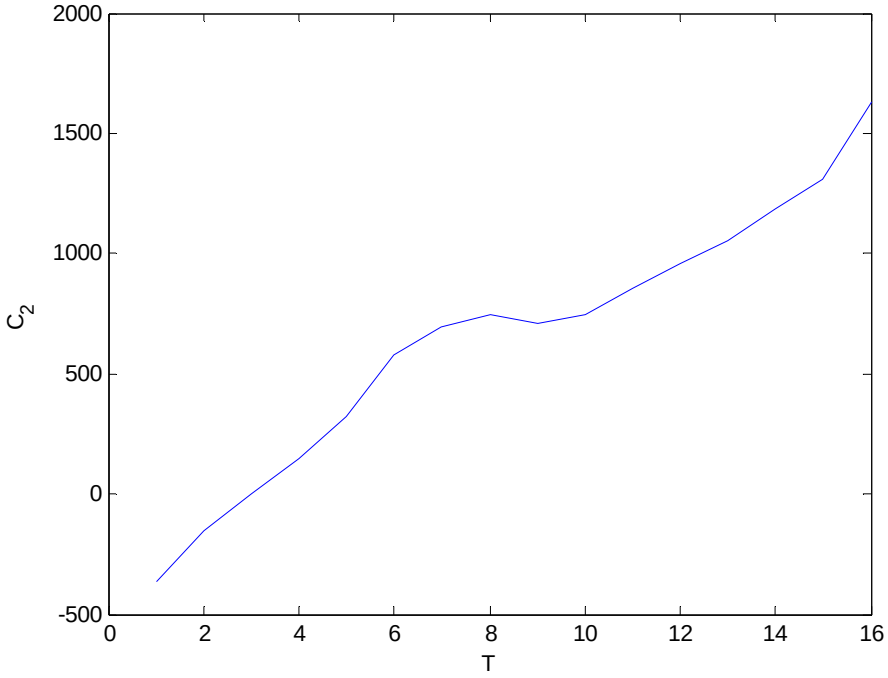

Figure 17. Preliminary coefficient C_2.

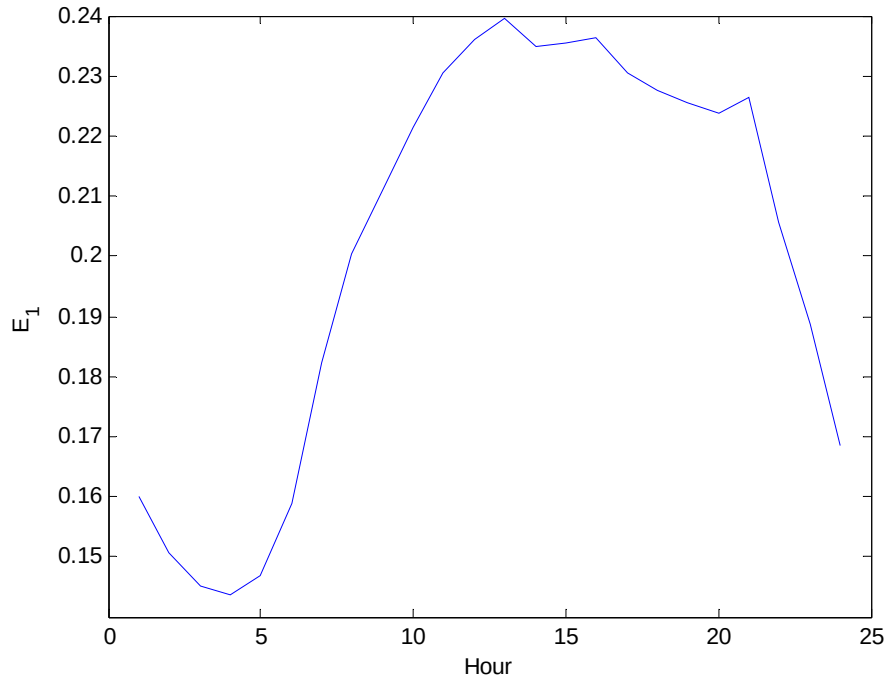

Figure 18. Orthonormalized basis vector E_1.

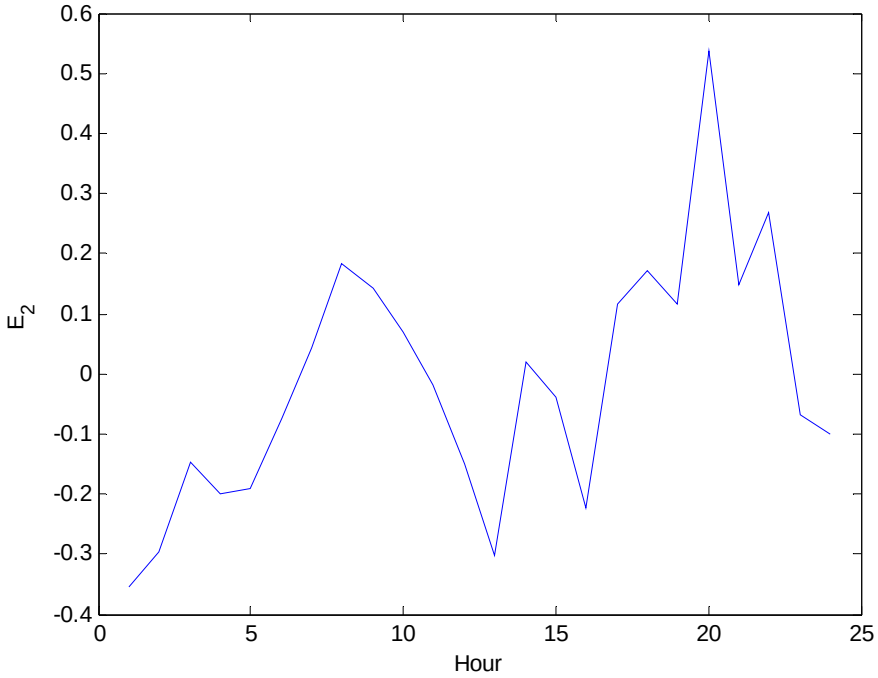

Figure 19. Orthonormalized basis vector E_2.

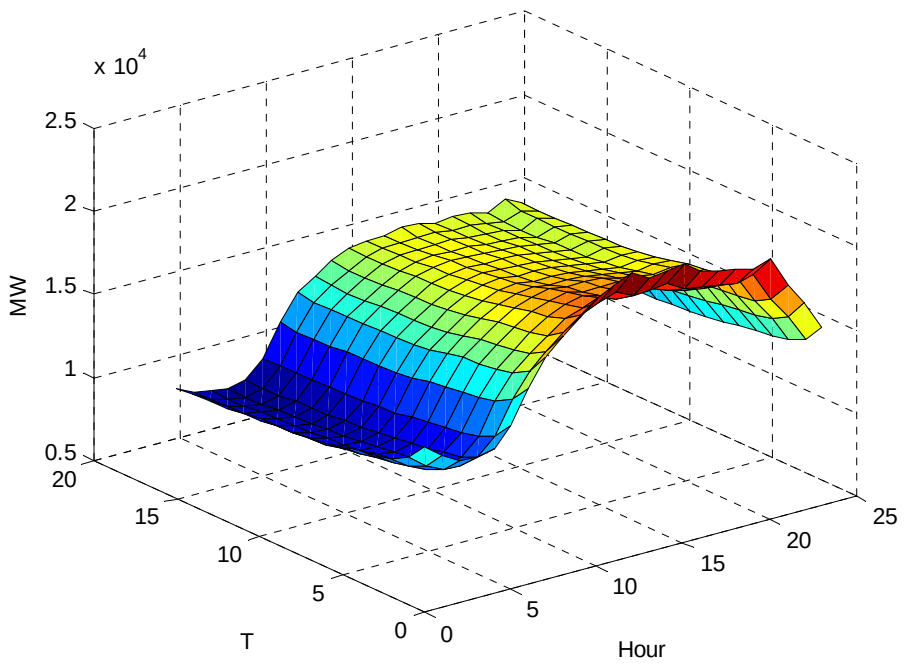

Figure 20. Computed regression load.

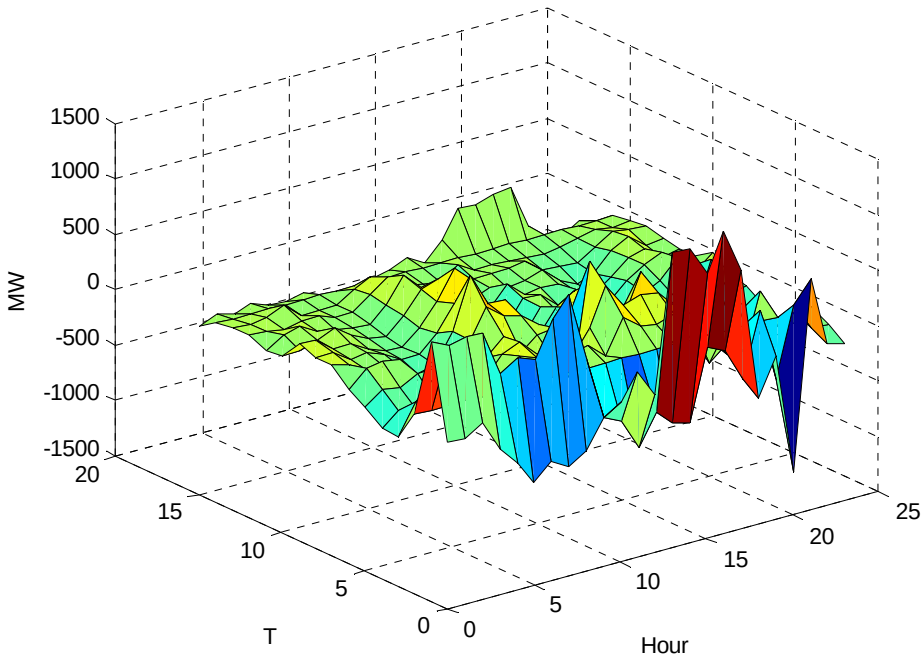

Figure 21. Error between empirical and computed regression load.

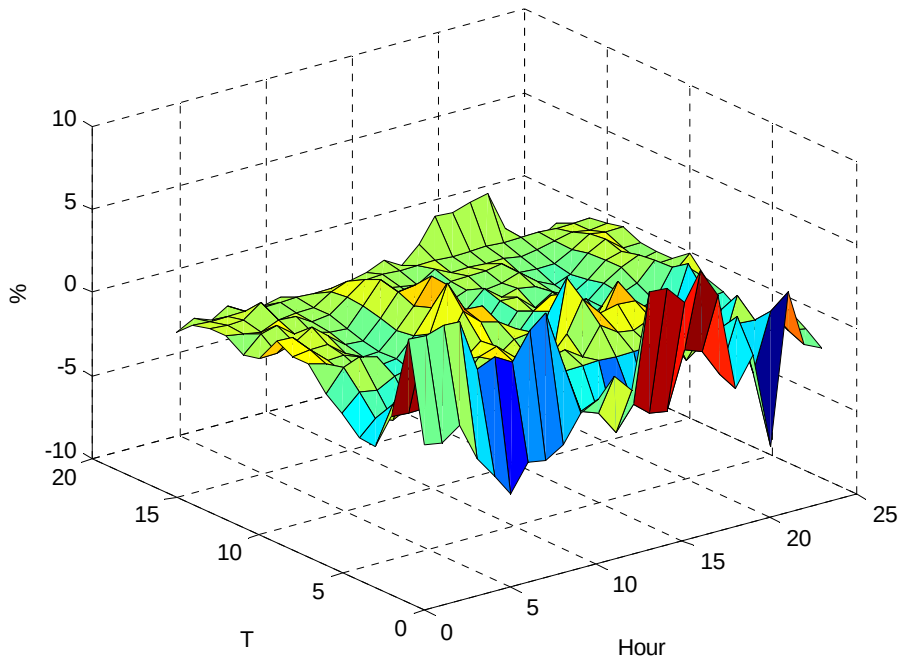

Figure 22. Percent error.

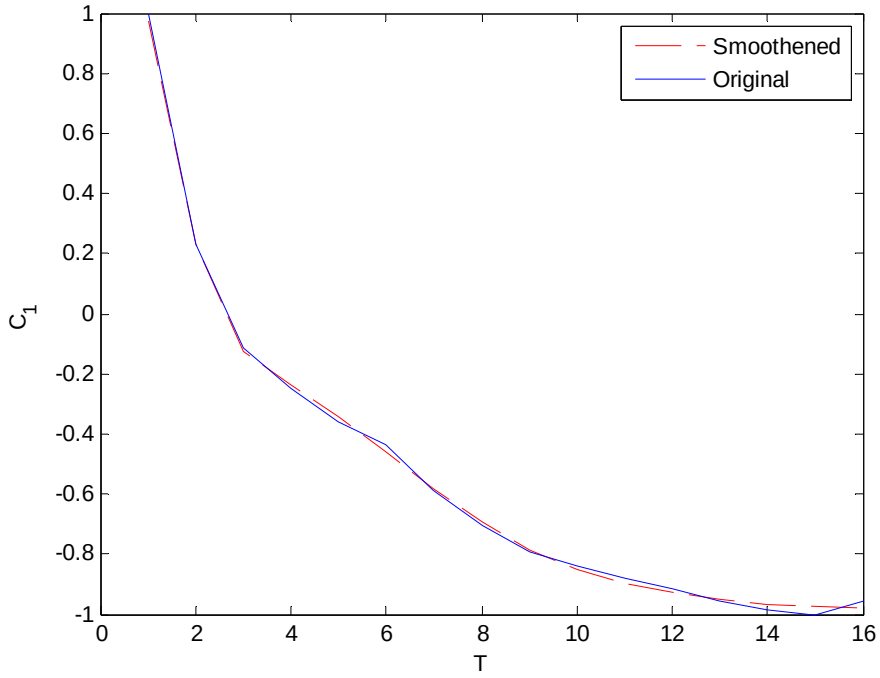

Figure 23. Comparison of original and smoothened coefficient vector C_1.

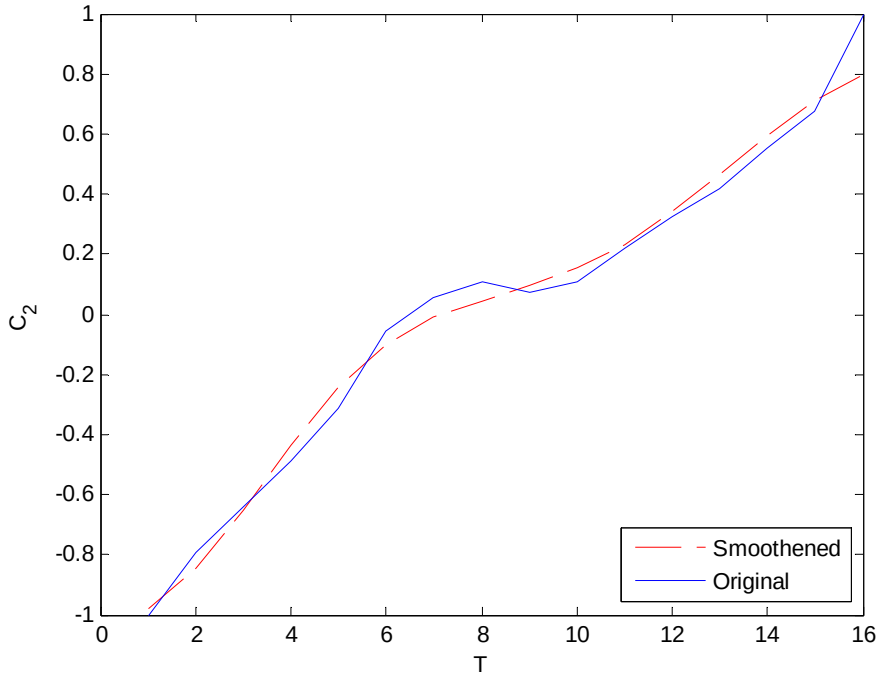

Figure 24. Comparison of original and smoothened coefficient vector C_2.

Extrapolation

The smoothened coefficient vector is observed to change dynamically over the temperature axis T. This dynamics is assumed to continue by the semigroup property and the extrapolation can be performed when it is needed. The dynamics of the coefficient vector is modeled by the SRN through the weight changes.

The Weight Change Convergence Check

The possibility for extrapolation is checked by observing the convergence of the weight change sequence as training is performed along the coefficient vector. In this case, weight convergence occurs as the training is repeated successively over longer intervals. It is this weight convergence that becomes the basis for extrapolation. There are 75 hidden neurons connected to 3 inputs and 2 outputs through input and output weights, respectively. The output of each neuron is also connected back to itself and other neurons through feedback weight. The feedback weight changes become zero almost immediately and therefore are not shown. The output weight changes are shown in Figure 25 and Figure 26. The output weight changes are shown only for the weights connected to the first 15 neurons for each coefficient. It is observed that the output weight changes converge to zero after a sufficient number of data is trained and the output weights remain almost fixed.

The input weight changes for dynamic component T are shown in Figure 27 and the integral of the input weight changes is shown in Figure 28. The input weight changes for static component $C(0)$ are shown in Figure 29 and Figure 30. The input weight changes are shown only for the weights connected to the first 15 neurons. It is observed that the input weight changes become zero after a sufficient number of data is trained.

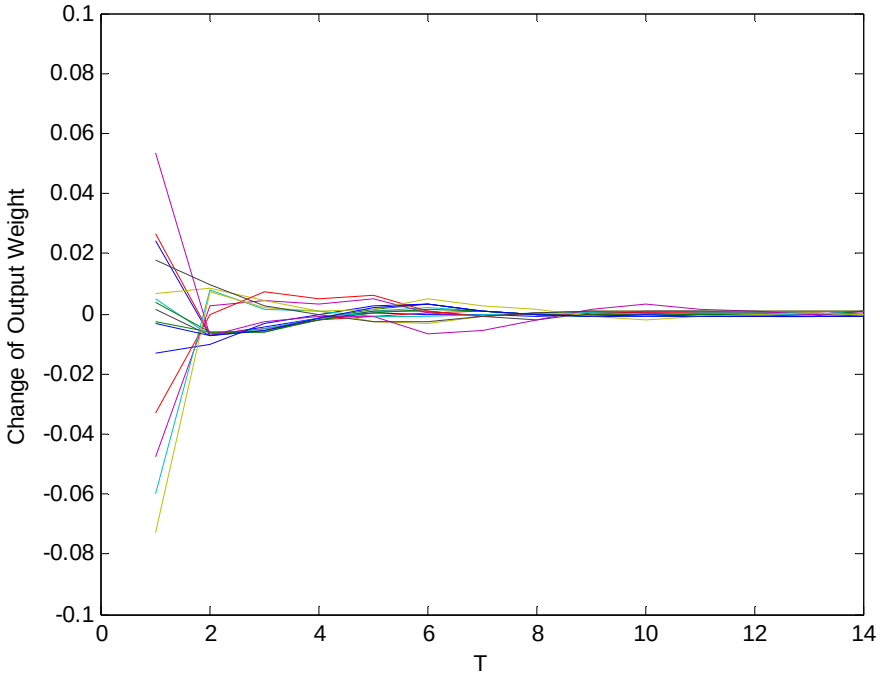

Figure 25. Output weight change for C_1.

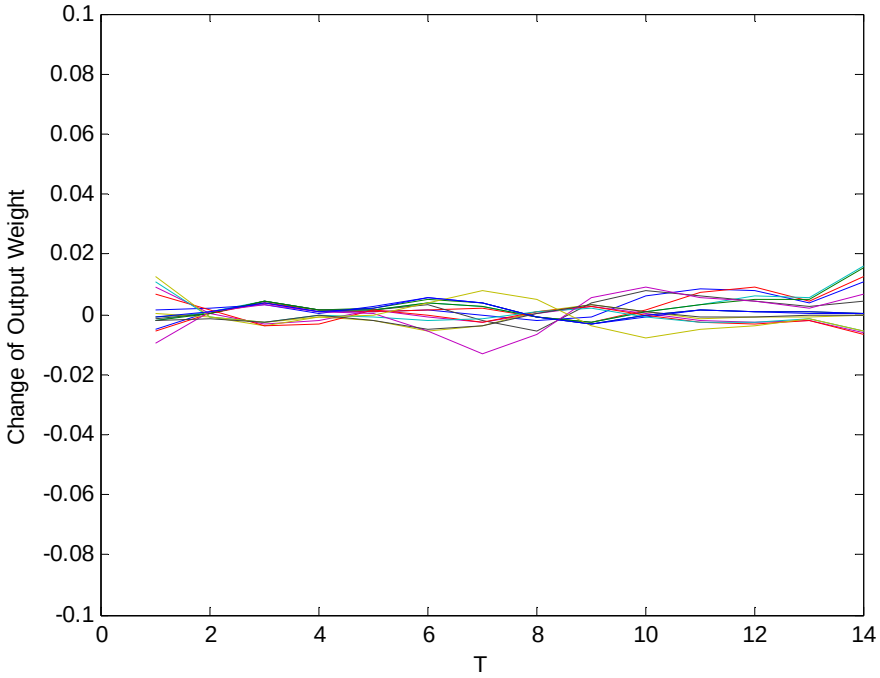

Figure 26. Output weight change for C_2.

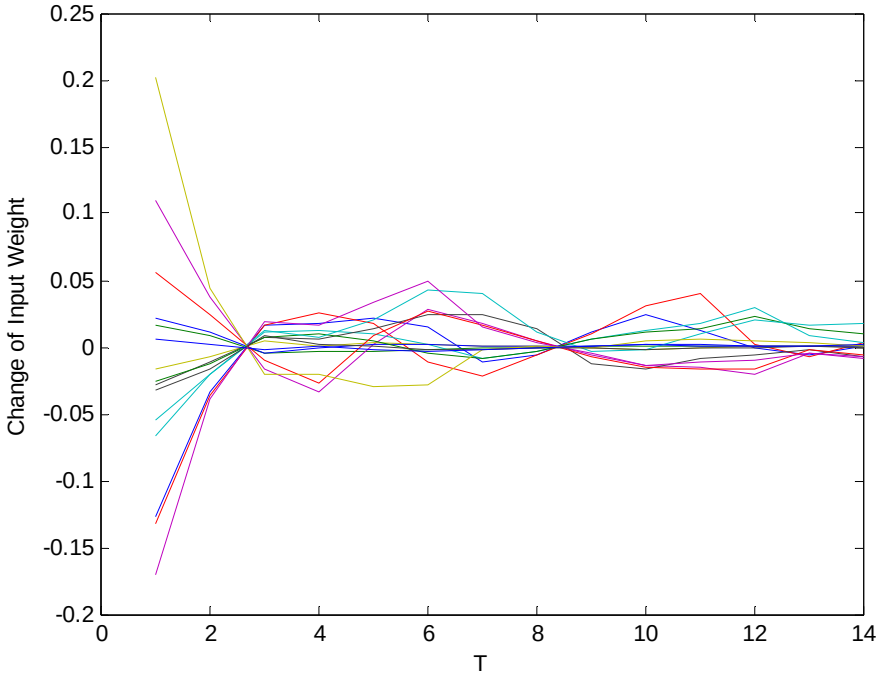

Figure 27. Input weight change for dynamic component *T*.

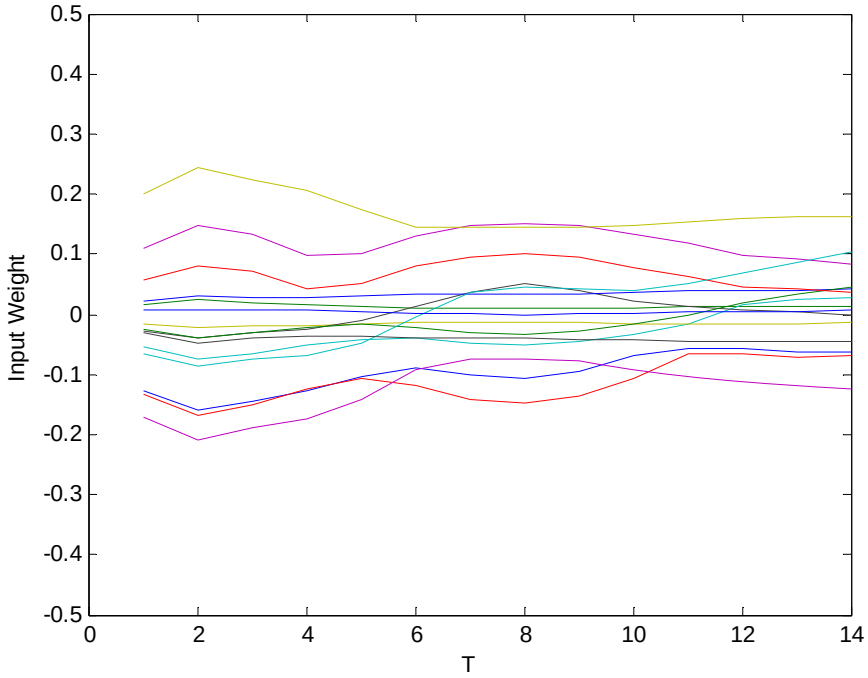

Figure 28. Integral of input weight change for dynamic component *T*.

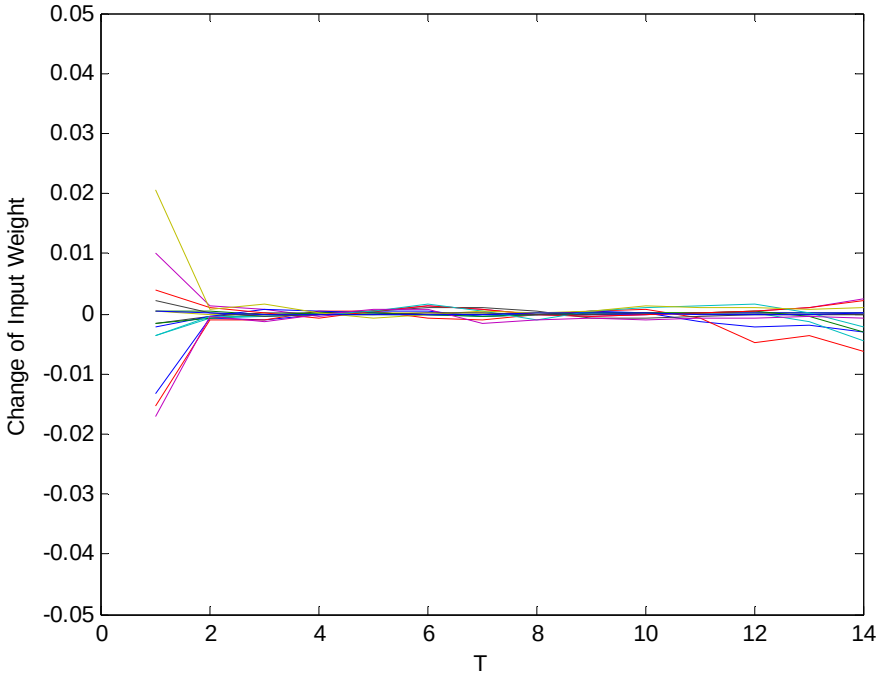

Figure 29. Input weight change for static component $C_1(0)$.

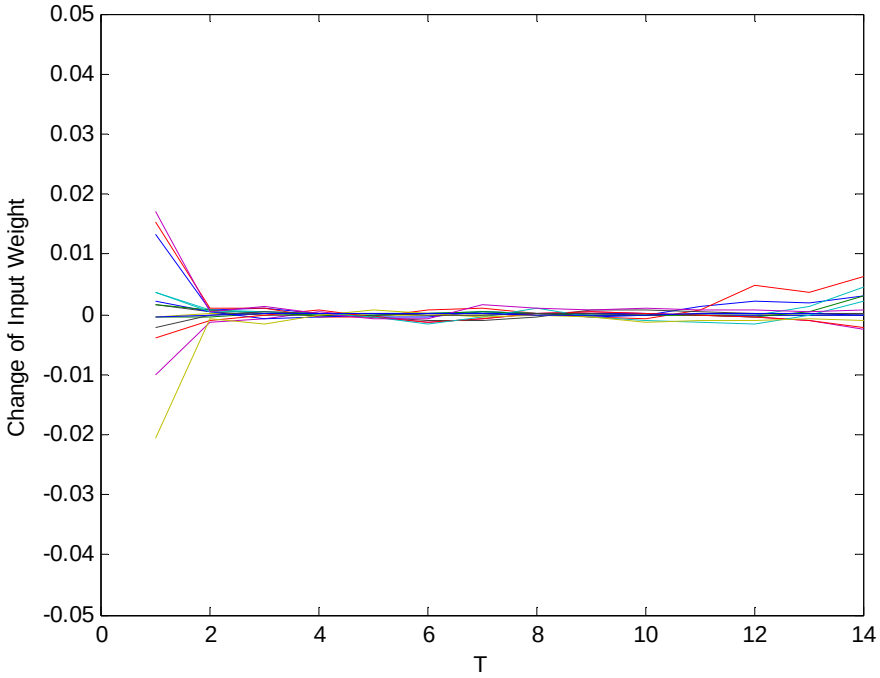

Figure 30. Input weight change for static component $C_2(0)$.

However, there is an exception for weights corresponding to the dynamic input T. Although the scale does not reveal it in Figure 28, the dynamic input weights are changing by non-zero amounts compared to the static input weights.

Extrapolation Test

Because of the smoothness of the coefficient vectors, the possibility for extrapolation exists and the next step is to apply an extrapolation test in which the trailing end of the weight change sequence (produced by training) is replaced by an equivalent weight change sequence based on a rule that generates a semigroup. Based upon an observation of the weight change sequence on the interval in the observation window, a semigroup-based rule of weight change for dynamic weights is modeled according to (49) and used in place of the original dynamic weights on the interval in the extrapolation test window. For each dynamic weight, a model for the weight change in the form of (49) is derived which, when inserted into the neural network, produces the same results as the actual weight change sequence over the extrapolation test interval. Figure 31 and Figure 32 show the extrapolation test for each coefficient. The similarity of the trajectories in the extrapolation test window supports that the weight change model can be applied and the extrapolation can be performed.

Extrapolation

Extrapolation (to the region where no data were assumed) consists of the autonomous continuation of the rule for the weight change which was modeled and tested during the extrapolation test. Since the weight change model in extrapolation test is

successful, the same modeling scheme can be applied to extrapolation. Figure 33 and Figure 34 show the extrapolation result for the next point of T .

Interpolation

The interpolation of coefficient vector is also needed at a given hour when the forecasting hourly temperature does not surpass the historical temperature boundaries. Assuming the coefficient vector is a continuous function with respect to the temperature variable, the interpolation becomes possible. Therefore, the interpolation can be implemented using the known historical temperatures and the forecasting temperature at a given hour. Figure 35 and Figure 36 show the interpolation results at a certain hour. The interpolated coefficient values can be obtained based upon the two adjacent points.

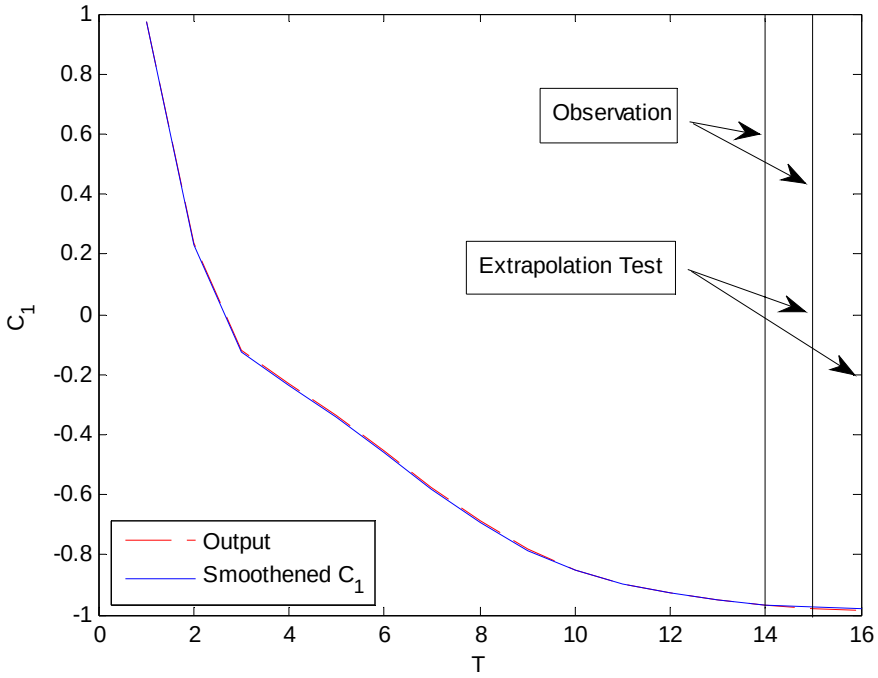

Figure 31. Extrapolation test for C_1.

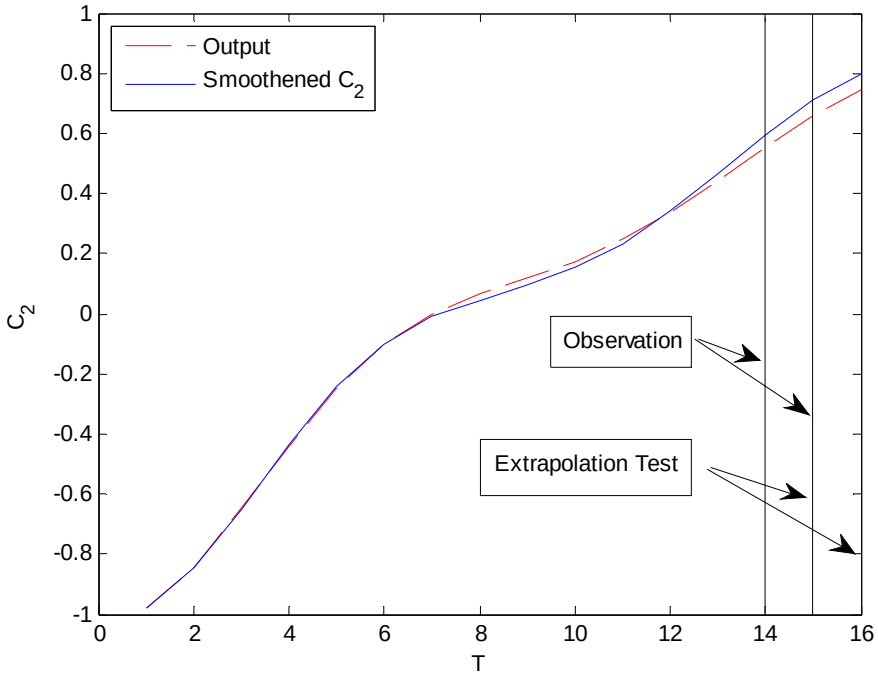

Figure 32. Extrapolation test for C_2.

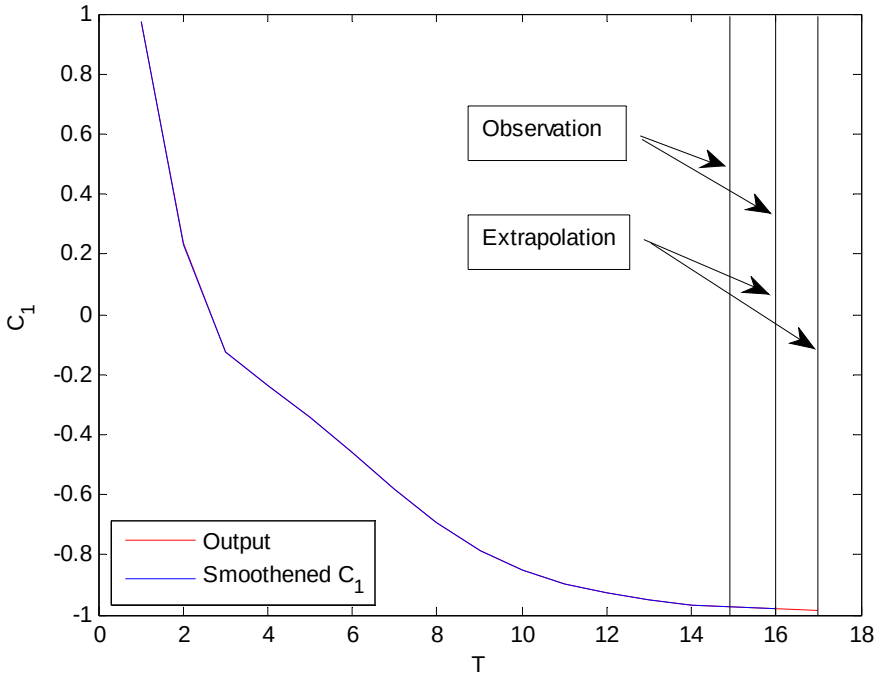

Figure 33. Extrapolation for C_1.

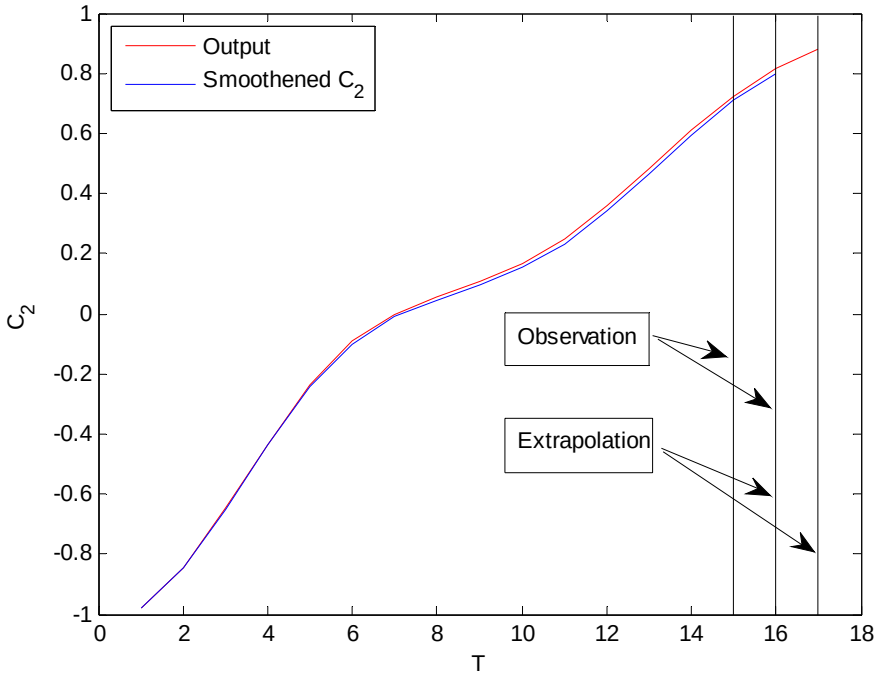

Figure 34. Extrapolation for C_2.

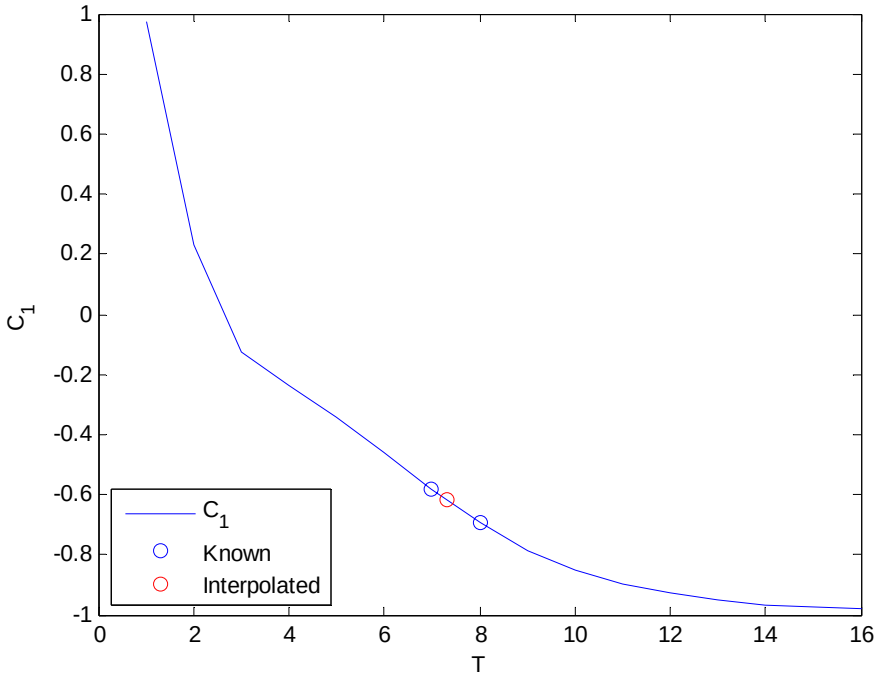

Figure 35. Interpolation of C_1.

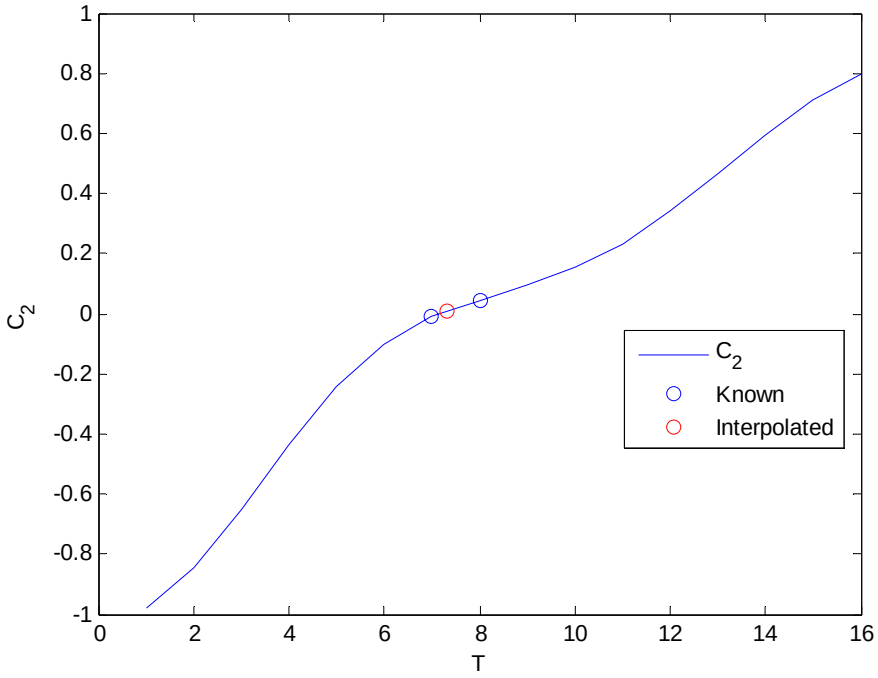

Figure 36. Interpolation of C_2.

Simulation Results

The proposed method is carried out for a one-day-ahead forecasting of hourly electric loads. The results are analyzed by the following formulas:

(i) Standard Deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{d=1}^{N}[L(d,h) - \hat{L}(d,h)]^2} \tag{51}$$

where $L(d,h)$ is empirical load data for a given day (d) and hour (h), $\hat{L}(d,h)$ is the corresponding load forecast.

(ii) Percent Error:

$$Error = \left|L(d,h) - \hat{L}(d,h)\right| \Big/ L(d,h) \times 100 \tag{52}$$

The simulation results include percent error and standard deviation at each hour, both being calculated from the regression load and the actual load. The regression load of the forecasting day can be derived from the historical regression model based on the assumption that the temperature forecasting has been completed. Table 1 shows the results of the regression load averaged for the whole year. Among the daily forecasting, Friday shows the best forecasting results and Saturday shows the worst forecasting results. For Friday, the average percent error is 1.05%, and the largest error is 1.25% at 19:00 and 20:00 and the least error is 0.76% at 9:00. For Saturday, the average percent error is 1.41%, and the largest error is 1.83% at 1:00 and the least error is 1.03% at 23:00. The total average error for daily forecasting is 1.20% and the average standard deviation is 280 MW. The errors of Monday, Wednesday, Thursday, and Friday are below the total average level, with respective errors 1.15%, 1.17%, 1.13% and 1.05%. The errors of Tuesday, Saturday, and Sunday are above the total average level, with respective errors 1.21%, 1.41%, and 1.25%.

Table 2 shows the forecasting results of the actual load averaged for the whole year. Wednesday and Thursday show the best forecasting results and Saturday still shows the worst. For Wednesday, the average percent error is 3.24%, and the largest error is 4.32% at 18:00 and the least error is 2.61% at 10:00. For Thursday, the average percent error is 3.24%, and the largest error is 4.63% at 18:00 and the least error is 2.44% at 24:00. For Saturday, the average percent error is 4.61%, and the largest error is 6.18% and the least error is 3.53% at 1:00. The total average error for daily forecasting is 3.75% and the average standard deviation is 782 MW. The errors of Tuesday, Wednesday, Thursday, and Friday are below the total average level, with respective errors 3.42%,

3.24%, and 3.50%. The errors of Monday, Saturday, and Sunday are above the total average level, with respective errors 4.01%, 4.61% and 4.26%.

Table 1. Statistics of average forecasting results of regression load for the year 2002.

	Mon.		Tue.		Wed.		Thr.		Fri.		Sat.		Sun.	
Hour	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.
1	1.30	209	1.29	313	1.22	208	1.07	184	1.00	168	1.83	314	1.48	256
2	1.12	194	1.35	293	1.08	205	1.08	171	1.01	167	1.80	279	1.50	274
3	1.25	200	1.30	283	1.27	238	1.17	174	0.99	176	1.66	244	1.37	233
4	1.37	198	1.45	295	1.35	259	1.27	187	1.10	204	1.44	207	1.28	211
5	1.40	196	1.61	298	1.32	224	1.44	211	1.18	223	1.49	224	1.64	247
6	1.43	227	1.38	284	1.41	275	1.40	226	1.24	228	1.40	244	1.36	196
7	1.39	270	1.22	269	1.21	247	1.24	259	1.17	220	1.44	253	1.27	206
8	1.21	264	0.82	181	0.97	205	1.06	288	0.81	166	1.60	255	1.13	186
9	1.09	249	0.91	222	0.78	171	1.11	262	0.76	173	1.55	281	1.06	199
10	1.06	258	0.95	218	0.79	203	0.96	251	0.93	227	1.14	209	0.80	171
11	0.90	254	0.85	260	0.75	203	0.87	247	0.89	250	1.28	269	0.88	178
12	0.77	186	0.81	223	0.87	203	0.97	318	0.88	239	1.51	316	1.00	222
13	0.89	224	1.01	271	1.02	334	1.21	402	0.98	239	1.51	328	1.26	293
14	0.91	252	1.04	291	1.09	335	0.86	222	1.08	255	1.44	311	1.32	352
15	0.91	231	1.18	342	1.12	375	0.93	315	1.16	314	1.40	268	1.05	242
16	1.02	272	1.19	348	1.32	438	1.13	323	1.08	310	1.51	286	1.20	301
17	1.23	333	1.04	245	1.11	421	1.16	311	1.18	303	1.60	311	1.29	342
18	1.27	379	1.13	323	1.32	412	1.20	392	1.51	403	1.39	278	1.20	315
19	1.13	316	1.61	927	1.50	404	1.39	471	1.25	337	1.29	259	1.04	242
20	1.13	352	1.36	868	1.25	325	1.20	348	1.25	343	1.12	214	1.03	207
21	1.11	318	1.33	753	1.41	426	0.97	295	1.09	320	1.09	254	1.10	232
22	1.24	294	1.17	521	1.19	338	1.10	287	0.98	252	1.06	215	1.29	289
23	1.17	244	1.42	487	1.37	367	1.12	275	0.90	186	1.03	184	1.55	346
24	1.34	250	1.53	373	1.46	300	1.14	231	0.89	159	1.16	189	1.87	376
Avg	1.15	257	1.21	370	1.17	296	1.13	277	1.05	244	1.41	258	1.25	255

Table 3 and Table 4 show the monthly forecasting results of the regression load. January shows the least error 0.55% and August shows the largest error 2.01%. For January, the largest error is 0.76% at 24:00 and the least error is 0.30% at 11:00. For August, the largest error is 2.65% at 1:00 and the least error is 1.31% at 8:00. The total average error is 2.65% at 1:00 and the least error is 1.31% at 8:00%. The total average

Table 2. Statistics of average forecasting results of actual load for the year 2002.

	Mon.		Tue.		Wed.		Thr.		Fri.		Sat.		Sun.	
Hour	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.
1	5.11	693	3.35	644	2.72	461	2.79	467	2.38	399	3.53	559	3.66	561
2	4.92	645	3.46	657	2.84	462	2.93	490	2.42	386	3.71	565	3.54	541
3	4.61	626	3.34	575	2.84	451	2.67	417	2.43	375	3.89	578	3.45	499
4	4.46	623	3.12	517	2.96	469	2.69	412	2.46	378	3.88	577	3.41	487
5	4.25	607	3.03	470	3.21	499	2.67	425	2.59	403	4.23	590	3.88	537
6	4.29	664	3.20	550	3.55	616	2.71	478	2.75	454	4.69	688	4.33	598
7	4.97	966	3.10	658	3.40	789	3.07	681	3.16	625	5.29	874	5.48	720
8	4.85	1123	3.06	769	3.01	859	3.18	831	2.97	683	6.18	1083	6.36	844
9	3.88	996	2.94	727	2.75	742	2.82	750	2.67	621	6.10	1126	6.25	937
10	3.55	993	2.76	701	2.61	674	2.89	751	2.88	674	5.65	1105	5.36	901
11	3.50	934	2.97	764	2.78	697	3.08	781	3.02	759	5.50	1128	4.61	870
12	3.41	913	3.07	798	3.02	731	3.48	879	3.30	860	4.86	1067	4.32	841
13	3.40	795	3.10	779	3.18	776	3.55	889	3.23	819	4.59	1030	4.28	860
14	3.79	922	3.35	868	2.96	756	3.82	947	3.50	864	4.78	1058	4.27	848
15	3.89	925	3.57	875	3.22	795	4.16	1032	3.74	952	4.74	1046	4.33	876
16	4.17	975	3.53	832	3.40	832	4.56	1159	3.74	900	4.97	1079	4.46	865
17	4.48	1026	4.14	996	3.97	950	4.40	1105	4.74	1105	5.14	1095	4.30	857
18	4.74	1163	4.39	1147	4.32	1090	4.63	1141	5.64	1311	4.93	1047	4.46	906
19	3.74	913	4.38	1174	4.30	1069	3.76	1037	5.78	1278	4.35	933	4.28	907
20	3.45	799	4.37	1193	3.97	974	3.43	919	6.15	1286	3.98	835	3.79	844
21	3.10	710	4.05	1044	3.51	852	2.80	709	5.53	1166	3.55	728	3.57	739
22	3.26	776	3.61	859	3.13	683	2.67	618	3.91	833	3.70	737	3.34	651
23	3.32	699	3.24	695	3.05	661	2.56	515	2.76	536	4.32	843	3.21	615
24	3.15	572	3.04	554	3.07	580	2.44	458	2.36	414	3.96	694	3.17	613
Avg	4.01	836	3.42	785	3.24	728	3.24	746	3.50	753	4.61	878	4.26	747

error for monthly forecasting is 1.19%. The errors of January, February, March, May, June, November and December are lower than the total average level, with respective errors 0.55%, 0.93%, 0.82%, 0.89%, 0.97%, 1.05% and 0.86%. The errors of April, July, August, September and October are higher than the total average level, with respective errors 1.20%, 1.86%, 2.01%, 1.57% and 1.61%.

Table 5 and Table 6 show the monthly forecasting results of the actual load. February shows the least error 2.70% and September shows the largest error 6.10%. For

Table 3. Statistics of monthly forecasting results of regression load for the year 2002.

	Jan.		Feb.		Mar.		Apr.		May		Jun.	
Hour	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.
1	0.57	93	1.25	198	0.81	133	1.23	172	1.10	179	1.13	176
2	0.52	79	1.14	181	0.87	144	1.31	159	0.92	120	0.79	130
3	0.60	88	1.37	194	0.85	144	1.25	154	0.91	116	0.56	83
4	0.59	85	1.30	186	0.95	160	1.39	172	0.91	111	0.60	82
5	0.70	95	1.45	215	1.15	176	1.70	216	0.89	108	0.92	115
6	0.73	105	1.33	221	1.04	156	1.80	265	0.85	116	0.95	142
7	0.66	121	1.12	233	1.00	176	1.18	203	1.08	165	1.09	221
8	0.54	104	0.81	184	0.84	151	1.21	218	1.10	193	1.30	335
9	0.48	98	0.72	150	0.77	143	0.83	161	0.90	174	1.38	298
10	0.41	88	0.63	138	0.66	125	0.68	127	0.73	187	1.18	301
11	0.30	65	0.57	112	0.58	131	0.61	123	0.78	196	0.79	215
12	0.38	81	0.85	160	0.73	180	0.80	182	0.95	230	0.93	213
13	0.42	90	0.73	145	0.71	151	1.15	280	1.04	269	1.09	356
14	0.52	104	0.67	146	0.67	159	1.11	264	1.00	242	0.82	185
15	0.56	119	0.75	142	0.66	149	1.12	248	0.93	209	0.87	213
16	0.60	142	0.72	143	0.84	183	1.24	288	0.82	176	1.12	334
17	0.59	143	0.61	121	1.08	228	1.14	234	1.00	197	1.01	256
18	0.56	148	0.67	152	0.89	219	1.11	210	1.03	201	1.03	310
19	0.58	143	0.65	145	0.69	179	1.80	377	1.12	228	1.25	512
20	0.45	101	0.59	136	0.78	193	1.40	285	1.06	195	0.85	169
21	0.56	110	0.81	181	0.68	159	0.97	227	0.55	105	0.96	216
22	0.59	107	0.92	192	0.70	132	0.99	201	0.48	97	0.83	168
23	0.63	102	1.19	250	0.69	122	1.18	221	0.53	92	0.86	183
24	0.76	114	1.50	321	0.95	151	1.56	248	0.67	97	1.08	202
Avg	0.55	105	0.93	177	0.82	160	1.20	218	0.89	167	0.97	226

February, the largest error is 3.99% at 7:00 and 8:00 and the least error is 2.07% at 23:00. For September, the largest error is 7.85% at 2:00 and the least error is 5.07% at 8:00. The total average error for monthly forecasting is 3.75%. The errors of January, February, March, April, May, October and November are lower than the total average level, with respective errors 2.91%, 2.70%, 2.79%, 3.20%, 3.10%, 2.85% and 3.19%. The errors of June, July, August, September and December are higher than the total average level, with respective errors 4.52%, 3.97%, 5.28%, 6.10% and 4.40%.

From the results, it can be observed that the percent errors with respect to regression load are most below 2% which meets the requirement in the power industry. However, due to ignoring the load which caused by other unknown factors, the results in terms of actual load don't achieve the short-term load forecasting requirement.

Table 4. Statistics of monthly forecasting results of regression load for the year 2002.

	July		Aug.		Sep.		Oct.		Nov.		Dec.	
Hour	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.
1	1.69	260	2.65	508	1.45	280	1.71	289	0.95	160	1.20	177
2	1.70	275	2.48	440	1.89	369	1.74	268	0.89	165	1.03	147
3	1.66	294	2.59	426	1.88	327	1.83	268	0.97	159	0.96	141
4	1.99	356	2.22	377	1.70	287	2.01	296	1.22	187	1.02	150
5	1.75	301	1.92	334	2.07	340	2.06	311	1.42	216	1.27	194
6	1.69	319	1.70	332	1.89	332	2.01	311	1.56	254	0.95	175
7	1.39	277	1.70	330	1.70	310	1.86	339	1.68	297	0.91	180
8	1.18	221	1.31	247	1.04	226	1.51	282	1.39	278	0.81	166
9	1.42	290	1.40	292	1.50	312	1.25	265	1.13	237	0.68	149
10	1.42	336	1.56	341	1.08	225	1.20	243	1.07	215	0.77	168
11	1.90	488	1.61	372	1.02	212	1.11	229	0.91	173	0.73	177
12	1.40	357	1.78	470	1.02	250	1.16	223	0.89	172	0.73	206
13	1.72	464	2.14	537	1.71	377	1.31	268	0.91	182	0.57	160
14	1.71	447	1.96	472	1.72	440	1.60	353	0.86	170	0.61	165
15	1.78	557	2.12	552	1.29	295	1.55	332	0.87	176	0.71	181
16	1.78	505	2.32	603	1.77	440	1.62	321	0.80	162	0.72	186
17	1.36	441	2.35	590	1.89	482	1.88	385	1.16	252	0.67	216
18	1.65	451	2.31	597	2.26	540	1.67	378	1.51	479	0.72	231
19	2.41	1236	1.97	462	1.89	425	1.48	342	1.08	281	0.81	223
20	2.89	1245	2.06	482	1.23	286	1.14	235	0.94	226	0.84	220
21	2.83	1106	2.39	622	1.14	230	1.33	284	0.80	211	0.95	226
22	2.31	761	2.08	484	1.79	381	1.46	361	0.74	178	0.95	219
23	2.55	712	2.10	395	1.33	324	1.84	413	0.68	145	1.04	206
24	2.39	510	1.63	288	1.41	341	2.32	423	0.84	145	1.10	189
Avg	1.86	509	2.01	440	1.57	335	1.61	309	1.05	213	0.86	186

Table 5. Statistics of monthly forecasting results of actual load for the year 2002.

	Jan.		Feb.		Mar.		Apr.		May		Jun.	
Hour	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.
1	2.35	347	2.61	370	2.59	386	2.80	382	3.00	394	3.89	646
2	2.41	337	2.46	335	2.52	358	2.77	357	3.03	363	3.71	634
3	2.47	346	2.36	307	2.40	329	2.69	351	2.86	335	3.66	597
4	2.56	355	2.50	330	2.57	348	2.74	365	2.79	327	3.59	579
5	2.86	392	2.64	358	2.56	343	2.96	384	2.71	332	3.76	594
6	3.21	492	2.87	500	2.61	378	3.33	441	3.29	450	3.62	566
7	4.28	808	3.99	903	3.40	558	3.44	527	3.77	709	3.83	619
8	4.67	985	3.99	1036	3.51	592	3.32	570	3.82	854	4.11	754
9	4.11	888	3.13	860	3.40	605	3.03	543	3.37	766	3.88	804
10	3.47	794	2.60	704	3.24	597	2.73	523	3.07	692	4.02	928
11	3.20	722	2.21	599	3.25	599	2.78	547	3.03	678	4.16	1006
12	3.02	650	2.20	566	3.10	576	3.01	569	3.07	698	4.67	1074
13	2.96	602	2.23	543	2.95	529	3.25	608	3.14	721	4.90	1102
14	3.07	612	2.27	556	2.98	525	3.20	628	3.29	779	5.50	1237
15	3.17	619	2.42	569	2.92	519	3.43	672	3.47	814	5.68	1281
16	3.16	604	2.74	604	3.11	536	3.62	719	3.53	827	5.93	1336
17	3.20	632	3.62	751	3.45	605	3.43	681	3.60	798	5.92	1329
18	2.23	525	3.84	819	3.98	735	3.93	735	3.78	790	6.13	1344
19	2.12	501	2.46	631	2.41	527	5.17	920	3.63	737	5.61	1211
20	2.15	491	2.45	619	2.17	452	4.03	733	3.96	719	5.40	1177
21	2.37	484	2.65	616	2.02	404	3.16	621	2.53	543	4.23	906
22	2.46	440	2.42	520	1.77	333	2.83	537	2.03	412	4.13	851
23	2.37	393	2.07	395	1.97	324	2.56	439	1.77	330	4.04	806
24	2.02	312	2.13	343	2.10	316	2.57	390	1.91	303	4.08	770
Avg	2.91	555	2.70	576	2.79	478	3.20	552	3.10	599	4.52	923

Table 6. Statistics of monthly forecasting results of actual load for the year 2002.

	July		Aug.		Sep.		Oct.		Nov.		Dec.	
Hour	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.
1	3.66	617	4.22	707	7.15	1072	2.78	358	2.06	297	3.24	508
2	3.56	570	4.71	741	7.85	1097	2.57	334	2.18	311	3.12	495
3	3.37	550	4.61	690	7.55	994	2.61	331	2.28	321	2.94	484
4	3.38	541	4.12	615	6.99	962	2.63	334	2.37	344	3.12	507
5	3.52	559	4.04	601	6.83	912	2.81	360	2.56	382	3.61	570
6	3.62	574	4.12	672	6.73	973	3.00	434	2.85	470	4.52	778
7	3.78	647	3.96	687	5.29	950	3.42	576	3.51	732	6.05	1215
8	4.16	827	3.89	800	5.07	1070	3.55	676	3.96	868	6.60	1399
9	3.96	863	4.41	1007	5.64	1163	3.09	611	3.52	765	5.39	1166

Table 6. Continued

	July		Aug.		Sep.		Oct.		Nov.		Dec.	
Hour	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.	Per. Err.	Std. Dev.
10	3.80	874	5.39	1203	5.20	1214	2.83	613	3.23	698	4.40	952
11	3.90	931	6.01	1367	5.28	1220	2.91	651	2.98	673	3.72	815
12	3.40	922	6.90	1531	5.20	1263	2.84	605	2.85	648	3.25	720
13	3.58	1028	6.26	1451	5.39	1127	2.64	563	2.79	657	3.27	710
14	3.79	1072	6.30	1437	5.43	1211	2.91	570	3.02	759	3.56	780
15	3.86	1084	6.04	1420	6.10	1331	3.11	555	3.28	849	3.83	826
16	3.87	1146	6.03	1383	5.92	1264	3.35	584	3.68	946	4.37	911
17	3.99	1210	6.34	1489	6.18	1264	3.44	651	5.58	1238	4.74	1065
18	5.22	1492	6.72	1544	6.91	1464	3.43	712	5.75	1385	4.87	1195
19	5.83	1533	6.29	1430	6.69	1442	3.00	607	3.92	1031	5.19	1242
20	5.99	1501	5.65	1306	6.12	1351	2.72	589	3.59	895	5.65	1281
21	5.01	1248	5.52	1180	5.77	1257	2.53	514	3.28	701	5.83	1248
22	3.81	878	4.82	1034	5.58	1196	2.37	426	2.88	567	5.50	1076
23	3.37	716	5.43	1061	5.72	1089	2.03	347	2.29	431	4.81	838
24	2.82	540	5.02	853	5.82	1006	1.80	293	2.09	368	4.11	652
Avg	3.97	913	5.28	1092	6.10	1162	2.85	512	3.19	681	4.40	893

CHAPTER SEVEN

Conclusions

In this thesis, a new neural network approach for short-term load forecasting is proposed. The proposed method investigates a mathematical approach referred to as algebraic decomposition to obtain an analytic modeling of empirical load data, which is in the form of the product of a coefficient vector with temperature variable and a set of basis vectors with time variable. The new concept of a combination of Radial Basis Function (RBF) networks and a Simple Recurrent Network (SRN) provides the external basis for the interpolation and extrapolation. The new training algorithm in the SRN offers an existence of semigroup property in the weight space.

Whether or not the coefficient vector is smooth is directly related to the possibility of implementing extrapolation. Therefore the assumption that actual load is mainly influenced by time and temperature factors is made, and the use of regression method makes the actual load to be more correlated to the expected two variables. An effort called rearrangement based upon the hourly temperature data is carried out to assure that a smooth regression load surface can be obtained with time and temperature coordinates. Finally according to historical temperatures and the known forecasting temperatures, we can perform interpolation or extrapolation for different hours when either is required. According to the obtained results, the percent errors with respect to regression load meet the short-term load forecasting objective, that is, normally errors should be below 2%. However, since actual load data used here represent load

consumption in a large area and the temperature data are average weighted values, the load includes various components caused by many factors besides time and temperature and correlation between load and temperature is not substantially strong. If we are given load and temperature which are highly correlated to each other, it is expected that much better results in regard to actual load can be achieved and the regression procedure can be removed.

BIBLIOGRAPHY

[1] K. Y. Lee, Y. T. Cha, and J. H. Park, "Short-Term Load Forecasting Using An Artificial Neural Network," *IEEE Trans. on Power Systems*, vol. 7, pp. 124-132, Feb. 1992.

[2] J. H. Chow, Applied Mathematics for Restructured Electric Power Systems: Optimization, Control, and Computational Intelligence. New York: Springer-Verlag, 2005, ch. 12.

[3] K. Y. Lee, Y. T. Cha, and J. H. Park, "Artificial Neural Network Methodology for Short-Term Load Forecasting," *NSF Workshop on Artificial Neural Network Methodology in Power System Engineering*, Clemson University, SC, Apr. 9-10, 1990.

[4] I. Moghram and S. Rahman, "Analysis and Evaluation of Five Short-Term Load Forecasting Techniques," *IEEE Trans. on Power Systems*, vol. 4, pp. 1484-1491, Nov. 1989.

[5] A.A. El-Keib, X. Ma, and J. Ma, "Advancement of Statistical Based Modeling Techniques for Short-Term Load Forecasting," *Electric Power Systems Research*, pp. 51-58, Mar. 1995.

[6] S. Vemuri, Wen Liang Huang, and D. J. Nelson, "On-Line Algorithms for Forecasting Hourly Loads of An Electric Utility," *IEEE Trans. on Power Apparatus and Systems*, vol. PAS-100, No. 8, pp. 3775-3784, Aug. 1981.

[7] J. Y. Fan and J. D. McDonald, "A Real-Time Implementation of Short-Term Load Forecasting for Distribution Power Systems," *IEEE Trans. on Power Systems*, vol. 9, No. 2, pp. 988-994, May 1994.

[8] M. A. Abu-El-Maqd and N. K. Sinha, "Short-Term Load Demand Modeling and Forecasting: A Review," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 12, pp. 370-382, May 1982.

[9] M. J. Damborg, M. A. El-Sharkawi, M. E. Aggoune, and R. J. Marks II, "Potential of Artificial Neural Networks in Power System Operation," *Proc. of 1990 IEEE International Symposium on Circuits And Systems*, pp. 2933-2937, New Orleans, Louisiana, May 1990.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, vol. 1, pp. 318-362, Cambridge, MA: MIT Press, 1986.

[11] P. Werbos, "Generalization of Backpropagation with Application to Recurrent Gas Market Model," *Neural Networks*, vol. 1, pp. 339-356, 1988.

[12] D. 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 Trans. on Power Systems*, vol. 6, pp. 442-449, May 1991.

[13] K. Y. Lee, Y. T. Cha, and C. C. Ku, "A Study on Neural Networks for Short-Term Load Forecasting," *Proc. of the First International Forum on Applications of Neural Networks to Power Systems*, pp. 26-30, 1991.

[14] K. Y. Lee, T. I. Choi, C. C. Ku, and J. H. Park, "Short-Term Load Forecasting Using Diagonal Recurrent Neural Network," *Proc. of the Neural Networks to Power Systems*, pp. 227-232, 1993.

[15] A. Khotanzad, R. Afkhami-Rohani, and D. Maratukulam, "ANNSTLF-Artificial Neural Network Short-Term Load Forecaster Generation Three," *IEEE Trans. on Power Systems*, vol. 13, pp. 1413-1422, 1998.

[16] D. K. Ranaweera, N. F. Hubele, and A. D. Papalexopoulos, "Application of Radial Basis Function Neural Network Model for Short-Term Load Forecasting," *IEE Proceedings-Generation, Transmission and Distribution*, vol. 142, No. 1, pp. 45-50, Jan. 1995.

[17] A. Khotanzad, Enwang Zhou, and H. Elragal, "A neuron-fuzzy approach to short-term load forecasting in a price sensitive environment," *IEEE Trans. on Power Systems*, vol. 17, No. 4, pp. 1273-1282, Nov. 2002.

[18] T. Senjyu, P. Mandal, K. Uezato, and T. Funabashi, "Next Day Load Curve Forecasting Using Hybrid Correction Method," *IEEE Trans. on Power Systems*, vol. 20, No. 1, pp. 102-109, Feb. 2005.

[19] S. Haykin, *Neural Networks*, 2nd ed., Prentice Hall, N.J., 1999.

[20] W. S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.

[21] D. O. Hebb, "The Organization of Behavior: A Neuropsychological Theory," Wiley-Interscience, New York, 1949.

[22] M. L. Minsky and S. A. Papert, *Perceptrons*. MIT Press, 1969.

[23] B. Widrow and M. E. Hoff, "Adaptive Switching Networks," *Parallel Distributed Processing*, vol. 1, pp. 123-134, Cambridge, MA: MIT Press, 1986.

[24] M. Omid, L. Omidvar, and C. Wilson, *Progress in Neural Networks*, vol. 5: Architecture. Edited by: Ablex Publishing Co., 1997, N.J.

[25] R. Jacobs and M. Jordan, "A Competitive Modular Connectionist Architecture," *Advances in Neural Information Processing Systems*, vol. 3, pp. 767-773, Morgan-Kaufmann, Cal., 1991.

[26] A. Atiya, R. Aiyad, and S. Shaheen, "A Practical Gated Expert Neural Network," *IEEE International Joint Conference on Neural Networks*, vol. 1, pp. 419-424, 1998.

[27] M. J. D. Powell, "Radial Basis Functions for Multivariable Interpolation: A Review," *IMA Conference on Algorithms for the Approximation of Functions and Data*, pp. 143-167, RMCS, Shrivenham, England, 1985.

[28] T. M. Cover, "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition," *IEEE Trans. on Electronic Computers*, vol. EC-14, pp. 326-334, 1965.

[29] H. M. Mhaskar, "Neural Networks for Optimal Approximation of Smooth and Analytic Functions," *Neural Computation*, vol. 8, pp. 1731-1742, 1996.

[30] C. C. Ku and K. Y. Lee, "Diagonal Recurrent Neural Networks for Dynamic Systems Control," *IEEE Trans. on Neural Networks*, vol. 6, pp. 144-156, Jan. 1995.

[31] J. Elman, "Finding Structure in Time," *Journal of Cognitive Science*, vol. 14, pp. 179-211, 1990.

[32] B. H. Kim, Development of System-Type Neural Network Architectures for Distributed Parameter Systems Using Algebraic Decomposition, Ph. D. Thesis, The Pennsylvania State University, PA, 2007.

[33] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, vol. I and II, New York: Wiley Interscience, 1970.

[34] A. Pazy, Semigroups of Linear Operators and Applications to Partial Differential Equations, New York: Springer-Verlag, 1983.

[35] R. Padhi and S. N. Balakrishnan, "Proper Orthogonal Decomposition Based Feedback Optimal Control Synthesis of Distributed Parameter Systems Using Neural Networks," *Proceedings of the 2002 American Control Conference*, vol. 6, pp. 4389-4394, May 2002.

[36] K. Y. Lee, J. P. Velas, and B. H. Kim, "Development of An Intelligent Monitoring System with High Temperature Distributed Fiberoptic Sensor for Fossil-Fuel Power Plants," *IEEE Power Engineering Society General Meeting*, pp. 1350-1355, Jun. 2004.

[37] B. H. Kim, J. P. Velas, and K. Y. Lee, "Development of Intelligent Monitoring System for Fossil-Fuel Power Plants Using System-Type Neural Networks and Semigroup Theory," *IEEE Power Engineering Society General Meeting*, pp. 2949-2954, 2005.

[38] B. H. Kim, J. P. Velas, and K. Y. Lee, "Semigroup Based Neural Network Architecture for Extrapolation of Enthalpy in A Power Plant," *Proc. of the ISAP*, pp. 291-296, 2005.

[39] B. H. Kim, J. P. Velas, and K. Y. Lee, "Semigroup Based Neural Network Architecture for extrapolation of Mass Unbalance for Rotating Machines in Power Plants," *International Federation of Automatic Control (IFAC) Symposium on Power Plants and Power Systems Control*, in CD, 2006.

[40] B. H. Kim, J. P. Velas, and K. Y. Lee, "Short-Term Load Forecasting Using System-Type Neural Network Architecture," *International Joint Conference on Neural Networks*, pp. 2619-2626, 2006.

[41] N. J. Hobbs, B. H. Kim, and K. Y. Lee, "Long-Term Load Forecasting Using System-Type Neural Network Architecture," *International Conference on Intelligent Systems Applications to Power Systems*, pp. 1-7, Nov. 2007.

[42] M. Q. Phan and J. A. Frueh, "Learning Control for Trajectory Tracking Using Basis Functions," *Proc. of the 35th IEEE Conference on Decision and Control*, pp. 2490-2492, Dec. 1996.

[43] J. Park and I. W. Sandberg, "Universal Approximation Theorem Using Radial Basis Function Networks," *Neural Computation*, vol. 3, pp. 246-257, 1991.

[44] B. H. Kim, J. P. Velas, and K. Y. Lee, "Development of System-Type Neural Network Architecture for Distributed Parameter System Modeling and Extrapolation".