

USING NEURAL NETWORKS TO DO TIME SERIES
FORECASTING

By

CHEN-CHOU LIN

Bachelor of Science

National Central University

Taiwan, R. O. C.

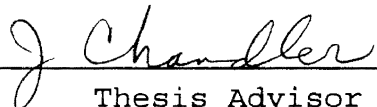
1986

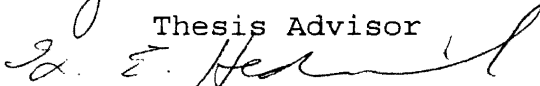
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the degree of
MASTER OF SCIENCE
May, 1993

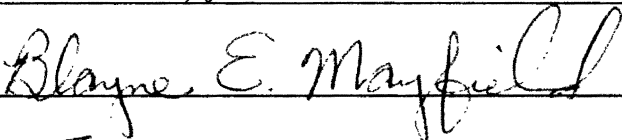
OKLAHOMA STATE UNIVERSITY

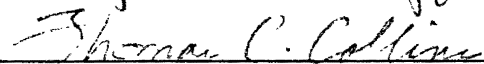
USING NEURAL NETWORKS TO DO TIME SERIES
FORECASTING

Thesis Approved:



Thesis Advisor






Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my grateful gratitude to Dr. J. P. Chandler for his guidance, encouragement and patience during this study. Likewise, sincere appreciation is expressed to the Advisory Committee members, Dr. G. E. Hedrich and Dr. B. Mayfield for their assistance.

I also wish to express appreciation to Dr. D. Miller for his help in understanding the theoretical concepts of neural networks and to Dr. R. Sharda in College of Business Administration for his help and providing the time series data.

A very special thanks and love go to my parents, Mong-shan Lin and Shunn-huey Shieh, my sister and brother for their constant support and understanding during my study overseas.

Special thanks is extended to my friend, Pei-ning Yang, for her encouragement and great understanding throughout this study. I also wish to express a deep thanks to Ryoko Oshima for her spiritual help.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION.	1
Neural Network Concepts.	2
General Statement of Forecasting	4
Objectives of the Study and Motivation	6
II. LITERATURE REVIEW AND RELEVANT STUDIES.	8
Literature Review and Related Studies.	8
Forecasting Using Neural Networks.	11
III. MODEL DEVELOPMENT AND METHODOLOGY	15
Why Neural Networks	15
Counterpropagation Training Algorithm.	16
Backpropagation Training Algorithm	20
Neural Network Architectures	24
Selection and Classification of Data Set	27
Method of Approach and Analysis.	28
Neural Network Implementations	31
IV. RESULTS, ANALYSIS AND DISCUSSION.	35
Learning Parameters and Weight Initialization	35
Results: Analysis of Yearly, Quarterly and Monthly Time Series.	39
Some Relative Examinations of Neural Networks	47
V. CONCLUSIONS AND FUTURE WORK	49
BIBLIOGRAPHY	57
APPENDIXES	63
APPENDIX A - TEST RESULTS OF BPN WITH YEARLY DATA	64
APPENDIX B - TEST RESULTS OF BPN WITH QUARTERLY DATA	67

Chapter	Page
APPENDIX C - TEST RESULTS OF BPN WITH MONTHLY DATA	72
APPENDIX D - TEST RESULTS OF CPN WITH YEARLY DATA	87
APPENDIX E - TEST RESULTS OF CPN WITH QUARTERLY DATA	90
APPENDIX F - TEST RESULTS OF CPN WITH MONTHLY DATA	95
APPENDIX G - TIME SERIES INFORMATION CONTENT. . .	110
APPENDIX H - EXPONENTIALLY WEIGHTED REGRESSION METHOD	114
APPENDIX I - COMPARISON OF DIFFERENT FORECASTORS AT 1_STEP FORECASTING.	118
APPENDIX J - BACKPROPAGATION PROGRAM.	125
APPENDIX K - COUNTERPROPAGATION PROGRAM	135
APPENDIX L - FIGURES OF SOME M-111 TIME SERIES. .	145

LIST OF TABLES

Table	Page
1. BPN and CPN network architectures	26
2. Comparison of the effects for different values of momentum and learning rate.	38
3. MAPE and M-APE values of yearly forecast of BPN at 1 step with arch. 2-2-1	65
4. MAPE and M-APE values of yearly forecast of BPN at 4 step with arch. 2-2-4	66
5. MAPE and M-APE values of quarterly forecast of BPN at 1 step with arch. 4-4-1.	68
6. MAPE and M-APE values of quarterly forecast of BPN at 4 step with arch. 4-4-4.	69
7. MAPE and M-APE values of quarterly forecast of BPN at 1 step with arch. 8-8-1.	70
8. MAPE and M-APE values of quarterly forecast of BPN at 4 step with arch. 8-8-4.	71
9. MAPE and M-APE values of monthly forecast of BPN at 1 step with arch. 12-12-1	73
10. MAPE and M-APE values of monthly forecast of BPN at 12 step with arch. 12-12-12	75
11. MAPE and M-APE values of monthly forecast of BPN at 1 step with arch. 24-24-1	81
12. MAPE and M-APE values of monthly forecast of BPN at 12 step with arch. 24-24-12	83
13. MAPE and M-APE values of yearly forecast of CPN at 1 step with arch. 3-6-3	88
14. MAPE and M-APE values of yearly forecast of CPN at 4 step with arch. 6-12-6.	89

Table	Page
15. MAPE and M-APE values of quarterly forecast of CPN at 1 step with arch. 5-10-5	91
16. MAPE and M-APE values of quarterly forecast of CPN at 4 step with arch. 8-16-8	92
17. MAPE and M-APE values of quarterly forecast of CPN at 1 step with arch. 9-18-9	93
18. MAPE and M-APE values of quarterly forecast of CPN at 4 step with arch. 12-24-12	94
19. MAPE and M-APE values of monthly forecast of CPN at 1 step with arch. 13-26-13	96
20. MAPE and M-APE values of monthly forecast of CPN at 12 step with arch. 24-48-24.	98
21. MAPE and M-APE values of monthly forecast of CPN at 1 step with arch. 25-50-25	104
22. MAPE and M-APE values of monthly forecast of CPN at 12 step with arch. 36-72-36.	106
23. Comparison of MAPE values of different forecastors at 1-step-ahead prediction.	121

LIST OF FIGURES

Figure	Page
1. Time series forecasting with neural net	16
2. Counterpropagation neural network	17
3. Activation function	21
4. Figures of some M-111 time series	146

CHAPTER I

INTRODUCTION

Artificial neural networks (ANNs) have been developed for a wide variety of applications in recent years. One of the most interesting computing applications is that of using neural network techniques to do time series forecasting. There are several useful and powerful learning algorithms, for example, backpropagation networks, counterpropagation networks, Hopfield nets, ART, ART2, etc. that can be used to predict the behavior of time series.

Many researches relative to prediction and decision making problems have proceeded for decades. This research focuses on the tests of the forecasting ability of two famous neural network algorithms, backpropagation and counterpropagation. A detailed comparison of the performances of backpropagation and counterpropagation models with NAIVE I ,NAIVE II, the exponentially weighted regression method [Chandler 1992] and the NN-PDP package used in Patil's thesis as well as the Autobox software [Patil 1990] is given in the Appendix I. The network constructions for both forecasting models are of three-layer type, one input layer, one hidden layer and one output

layer. More detail about the number of neurons in the different layers and the network architectures will be discussed in Chapter 3. The real-world data sets from the M-111 competition [Makridakis 1982] are taken as the testing data to examine the accuracy of prediction for the two neural network models.

Neural Network Concepts

The artificial neural network structures consist of fully interconnected layers (or slabs) or rows of processing elements (or neurons, units) [Hecht-Nielsen 1990]. Its prototype can be either single-layer (linear) or multi-layer networks. The former comprises only the input and output slabs while the latter can have different number of hidden slabs between input and output layers.

Each layer in the network contains a number of artificial neurons in a row. The neurons can receive any number of streams of incoming information through the connections (or links which have weights) and produce a single output signal. For single-layer neural networks, the nodes in the input layer serve only to distribute the input signals to the output layer and perform no computation. Therefore, the input layer will not be considered as a function layer. The neurons in the output layer simply output the weighted sum of the inputs to the network. The representational ability of single-layer (linear) networks

is severely limited, because it cannot provide the necessary nonlinearity [Wasserman 1990].

With regard to the multilayer artificial neural networks, the input layer functions the same as that of a single-layer neural network. The difference arises from the computational capability of the hidden layer(s) and the output layer. According to Wasserman (1990), multilayer networks provide no increase in computational power over a single-layer network unless there is a nonlinear activation function (or squashing function) between layers. The products of input information and the weights will be summed up as the Net and the Net will be processed by the transfer function (or the activation function) within the processing unit to emit an updated output signal to the next connecting layer.

There are two operations for artificial neural networks. One is the training of the networks, the other is the normal operation (application) of the networks. Neural nets perform a wide variety of pattern mappings and pattern representations after the networks are well-trained [Denning 1992]. The most important characteristic of neural networks is the cognitive behavior arising from the training of the networks. Through repeating input vectors to a network during training, a network can learn to memorize the input patterns and to represent the desired (or approximate) outputs. The learning capability is achieved by modifying

the interconnections (or weights) of the networks according to a predetermined procedure [Wasserman 1990]. A neural network is said to be well-trained if the errors (or noise) between the produced outputs and the desired results are minimized to an acceptable range.

The learning ability arising during the training procedure can be either supervised or unsupervised [Wasserman 1990]. Supervised learning requires the desired output (or target vector) be paired with each input vector to examine the behavior of the system. As an input vector is applied to the network system, a computed product is produced to compare with the corresponding target vector. The difference (or error) between the produced output and the target vector is propagated back through the network layer by layer to adjust the interconnecting weights. The unsupervised learning of the network requires no target vector for the outputs, and therefore, no comparisons to the desired result. When each input vector is applied to the network, no predetermined target output accompanies it. The neural network self-organizes to produce the ideal responses during the training.

General Statement of Forecasting

There are two major types of forecasting models: time-series and regression (causal) models [Makridakis and Wheelwright 1978]. For a time-series model, prediction of

the future is based on past values of a variable and/or past errors. Its objective is to discover the data series pattern of the past and extrapolate that pattern into the future. In the second type, it assumes that the factor to be forecast exhibits a cause-effect relationship with one or more independent variables. The purpose of the causal model is to discover the form of that relationship and use it to forecast future values of the dependent variable.

There are many time series forecasting techniques that have been invented and utilized widely in the human world, such as smoothing and decomposition methods, NAIVE methods, ARMA (Autoregressive/Moving Average) methods, Box-Jenkins method, ..., etc. [Makridakis and Wheelwright 1978]. The major concern in this thesis is to evaluate the forecasting ability of artificial neural networks over a wide range of data sets from the M-111 competition [Makridakis et al. 1982] and see if the neural network models are suitable to do the forecasting task or not. If the neural network models are able to discover the underlying structure changes of the environment and give an appropriate prediction to the future, then they can be used for the prediction job.

The problem of prediction can be generalized as the following equation:

$$\hat{Y}(t+N) = F(X(t), X(t-1), X(t-2), \dots, u(t+1) + \dots + u(t+N))$$

where,

$\hat{Y}(t+N)$ is the forecasting value for $X(t+N)$ with the

historical observations $X(t), X(t-1), X(t-2), \dots$;

N is the number of future values to be forecast.

F is the vector function containing random terms (or errors) $u(t+1), \dots, u(t+N)$;

$X(t)$ is the input vector of observation at time t .

The task here is to select and minimize the random terms (or errors) in predicting the output value $\hat{Y}(t+1)$ such that the function F can be used to forecast the future results of the data series $X(t), X(t-1), X(t-2), \dots$. Here, the data series $X(t), X(t-1), X(t-2), \dots$, is a large sample of the real-world time series from the M-111 competition data sets [Makridakis et al. 1982]. The vector function F will also be employed to simulate the forecasting environment of time series.

Objectives of the Study and Motivation

Considerable time and efforts have been spent in studying counterpropagation and backpropagation neural networks (this also includes various forecasting research papers using neural network algorithms). On reading these, the interest toward the studies of neural networks and time series forecasting has arisen. Therefore, the objectives of this research work are as follows:

- (i) Using a different neural network technique (counterpropagation networks) to evaluate the forecasting ability of neural networks with the 111 M-

competition data series, instead of backpropagation networks.

- (ii) Examining the learning parameters of different network architecture (for example, number of input neurons, number of neurons in the Kohonen layer and number of output neurons in the Grossberg layer, etc.), the training rate coefficients and the momentum introduced in the backpropagation algorithm.
- (iii) Analyzing and comparing the forecasting results of the two neural network models over the M-111 competition time series data with NAIVE I, NAIVE II, exponentially weighted regression and that of NN-PDP and Autobox in Patil's thesis.

There are many applications of neural networks (for example, data compression, image processing, pattern recognition, robotics central application, etc). Motivation comes from the interest in the learning ability of neural networks and time series forecasting as well as trying to estimate the performance of another neural network implementation.

CHAPTER II

LITERATURE REVIEW AND RELEVANT STUDIES

Literature Review and Related Studies

Researchers have designed artificial neural networks to simulate the organization and operation of the human brain. The original neural network model has only the input and output layers, which map a set of input vectors (patterns) directly to a set of output vectors. This construction restrains the learning ability of the neural network, since similar input patterns will lead to similar outputs or the same output. It is possible that such neural network cannot perform the necessary mapping. In order to avoid the limitation of single-layer networks, larger and more complex networks such as multilayer artificial neural nets with a nonlinear activation function between layers were invented to offer stronger computational capability. Through supervised or unsupervised training, artificial neural networks can learn to represent the desired outputs or at least give consistent results.

Artificial neural net models have been studied for many years. A wide range of applications in the fields of speech, image recognition [Lippmann 1987] and time-series

forecasting [Patil and Sharda 1989, 1990, Tang et al. 1991, Hill et al. 1991, 1992] have been explored by researchers.

In Rumelhart and McClelland's work (1988), multilayer networks with internal representation units (or hidden layer) were presented. The input information is recorded into an internal representation which generates the outputs. If there are enough hidden units, input patterns can always be encoded and the appropriate output patterns can be produced from any input pattern. This characteristic is similar to a backpropagation neural network.

Some neural networks such as Hopfield Nets suffer from a tendency to stabilize to a local rather than a global minimum of the energy (objective) function [Wasserman 1990, Knight 1990]. Hinton and Sejnowski (1988) and Wasserman (1990) presented a solution that starts with large steps and gradually reduces the size of the average random steps for this problem to approach the global minimum. This solution gets its idea from annealing of a metal; hence it is usually called the simulated annealing procedure with a probability density function.

The recent research has a rising interest in the selection for an optimal neural network. Fogel (1991) proposed one method to choose the "best" network. It is a modification of the Akaike's information criterion (AIC), the final information statistic (FIS).

In Masson and Wang's paper (1990), they described

several network models such as the Boltzmann machine, the Kohonen network and the Hopfield-Tank net. They applied the Kohonen self-organizing network to solve the Travelling Salesman Problem (TSP). Meanwhile, they also pointed out that the Boltzmann Machine is a time-consuming network.

Adaptive resonance architectures are neural networks that self-organize stable recognition codes in real time in response to arbitrary sequences of input patterns [Carpenter and Grossberg 1987]. In Carpenter and Grossberg's paper, they also indicated that such an architecture's adaptive search has the ability to discover and learn appropriate recognition codes without being entrapped in spurious memory states or local minima.

If time is the most important factor in the applications of neural networks, one can consider the utilization of the counterpropagation networks (CPN). CPN are not as general as backpropagation networks, but they provide a solution for those tasks that cannot tolerate tedious training procedures [Wasserman 1990]. In Hecht-Nielsen's paper (1987b), he pointed out that the advantages of a CPN are simplicity and that it can establish a good statistical model of its input vector environment. One of CPN's main applications is data compression.

By combining the self-organizing mapping of Kohonen learning and the outstar structure of Grossberg learning, a new type of mapping neural network (counterpropagation

network) is obtained [Hecht-Nielsen 1988]. In the report, Hecht-Nielsen described four major applications of counterpropagation neural nets. They are pattern recognition, function approximation, statistical analysis and data compression. The author also concluded that counterpropagation has advantages over other neural net approaches, such as development cost, computational savings, and the use of an explicitly parallelizable architecture.

Forecasting Using Neural Networks

In time series forecasting history, artificial neural networks have been utilized on the prediction for future tendency of data series. In Patil and Sharda's report (1989), they applied food product sales data series to train different configuration of the forecasting models. The models established were three-layered networks using a back error propagation (BEP) learning algorithm. The result shows that the neural network models can exhibit the seasonality property of the data series and give a reasonable analysis of the data patterns. Again, in 1990, Sharda and Patil made a comparison between a neural network approach and the conventional forecasting instrument, the Box-Jenkins model, with seventy-five time series. The neural network model being used is the well-known feedforward backpropagation network of a three-layered perceptron (with one hidden layer). The result suggests

that the simple neural net architecture can have a competitive performance with Autobox, a Box-Jenkins forecasting expert system.

Tang, Almeida and Fishwick (1991) have done an experiment with three time series of different complexity by using a backpropagation network implementation and the Box-Jenkins model in 1991. They concluded that both methods demonstrated comparable results for time series with long memory and that neural network models outperformed Box-Jenkins model for time series with short memory.

According to Hill, O'Connor and Remus's work (1991), they developed three different neural network configurations (NN-I, NN-II and NN-III) simulating backpropagation algorithm. The 111 time series data from the M-competition [Makridakis et al. 1982] were used to train and test the three neural network implementations. The result reveals that NN-II is the best neural network structure and it is used to compare with six classical models (Box-Jenkins, NAIIVE, Deseasonalized Exponential Smoothing, ..., etc.). Generally speaking, NN-II was better than the average of the other six reference methods for most periods. This is quite encouraging for the proponents of using neural networks as a forecasting tool.

Hill et al. (1992) argued more about the comparison between neural networks and the conventional statistical methods in prediction ability. Not only do they estimate

the performance difference between neural networks and time series forecasting models but also they evaluate causal forecasting models and decision models with neural networks. Across the areas of empirical studies, the authors summarize that neural networks seem to perform at least as well as classical models but there are still some problems left to be solved.

There are several important factors that need to be taken care of before and during the training of the neural networks. One is to normalize the input vectors before training; another is deciding on a method to adjust the connection weights appropriately; the other is employing a good strategy to determine when the training procedure should stop. In Logar, Corwin and Oldham's paper (1992), they consider the factors described above and use a modified backpropagation network to examine and predict the time series generated by measuring the acid concentration of waste water at given time periods. The experiment exhibits a satisfactory result to the researchers.

We also can use neural networks to forecast student behavior that used to be predicted by the student model, an important component of an intelligent tutoring system (ITS) [Mengel and Lively 1992]. The authors emphasize two significant abilities in the paper. The first is that neural networks can generalize over an input pattern set and can be used to predict the future behavior of the pattern

data. The second is that neural networks can be designed to learn from the pattern set and produce output patterns that it has not seen before. These talents make neural networks efficient and flexible for this application.

In Sastri, English and Wang's paper (1990), an empirical study on applying three known artificial neural networks (backpropagation, counterpropagation and the category learning with a single hidden layer) to identification of autoregressive time series models is presented. The evidence indicates that the backpropagation network was the best model identifier and the counterpropagation network was the best model change detector. The study also shows that the model change detection capability is related to a network's generalization, which depends on the number of processing elements in the hidden layer, the training sample and the learning algorithm employed.

CHAPTER III

MODEL DEVELOPMENT AND METHODOLOGY

Why Neural Networks

A neural network is a parallel, distributed information processing structure consisting of processing elements (i.e. nodes, neurons) and connections (i.e. interconnected weights, links) [Hecht-Nielsen 1990]. The ability to learn is the most intriguing aspect of all the neural networks' characteristics. The training of networks is accomplished by applying input vectors (or patterns) sequentially and adjusting the interconnected weights with a predetermined procedure [Wasserman 1990]. Once the network is trained, the error (or noise) is reduced to an acceptable interval and then the network can be used for testing. At this point, the weight vectors are not changed. Even if the given testing information is not complete, the neural network model still can accept it as an input and yield the appropriate output. This pattern distortion tolerance makes neural network capable of pattern recognition.

Since the output of each neuron (processing element) is the summation of the weighted inputs from previous neurons, we can analogize this relationship to be a time-series

forecasting procedure. That is, through a neural network model, the input data from previous time series can be used to generate the forecasting output for future time series. The relationship is depicted in Figure 1.

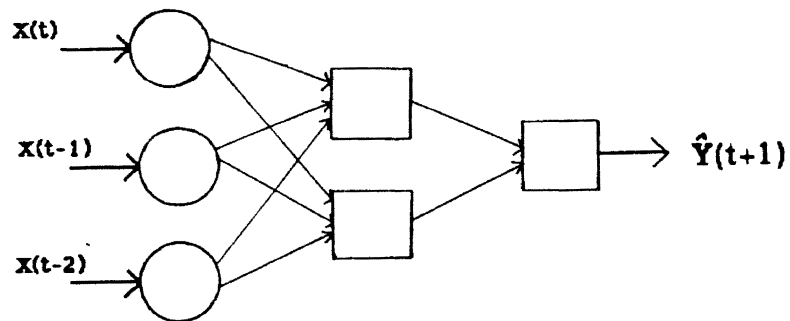


Figure 1. Time Series Forecasting with Neural Net

Counterpropagation Training Algorithm

The first neural network approach being employed in the forecasting experiment is the well-known counterpropagation network. The network combines a self-organization mapping Kohonen layer as the hidden layer and a Grossberg outstar layer as the output layer [Hecht-Nielsen 1990, Wasserman 1990]. The generalization property of the feedforward counterpropagation network allows it to output a proper (or even better, the desired and correct) result even though the input vector is partially incorrect (for instance, the input

vector is incomplete). The counterpropagation training algorithm is designed to approximate a continuous function $f: \text{ACR}^n \rightarrow \text{BCR}^m$, defined on a compact set A . The full network works best if f^{-1} exists and is continuous (i.e. if f is one-to-one and onto and if the inverse mapping $f^{-1}: \text{BCR}^m \rightarrow \text{ACR}^n$ is continuous) [Hecht-Nielsen 1990].

The network structure is depicted in Figure 2.

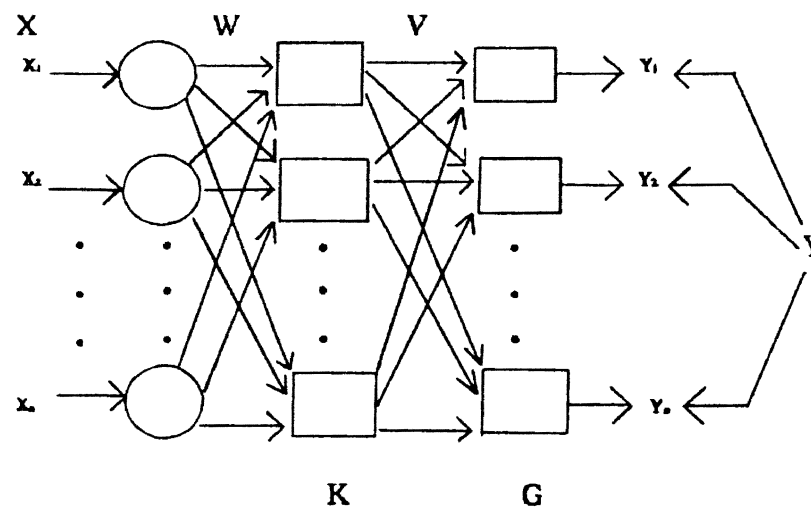


Figure 2. Counterpropagation Neural Network

Where

X = the input vector (x_1, x_2, \dots, x_n) ;

Y = the desired output vector (y_1, y_2, \dots, y_m) ;

W = the weight matrix comprising weight vectors $w_1,$

w_2, \dots, w_n (the conjunction between input layer and Kohonen layer);

V = the weight matrix comprising weight vectors v_1, v_2, \dots, v_m (the conjunction between Kohonen layer and Grossberg layer);

K = the Kohonen layer that contains p neurons;

G = the Grossberg layer that contains m neurons.

The objective of training Kohonen neurons is to separate the dissimilar input vectors into different groups (or clusters). During training, the Kohonen layer functions in a "winner-take-all" fashion (i.e. for an input vector, one and only one Kohonen neuron produces the logical one; the rest of the Kohonen neurons output a zero). The Kohonen neuron that has the largest summation of products of weighted input vectors wins the competition. The weight of the winner will then be adjusted according to the following formula:

$$w_{new} = w_{old} + \alpha (x - w_{old})$$

where α is a learning rate that may gradually be reduced during the training.

It is difficult to predict which Kohonen neuron will be activated (or be the winner) for a given input vector. In order to prevent the same Kohonen neuron from winning more than its fair share of the time (approximately $1/k$, where k is the number of Kohonen neurons in the Kohonen layer), I

added a "conscience" testing method in the program to check this implementation. This gives an opportunity to the other Kohonen neurons to be trained and provides a better result for time series forecasting. During the training of the neural network, the weight vector w will converge to (or approach to) the average value of the input vector x .

The only action in Grossberg layer is to activate the Grossberg neuron that is connected to the Kohonen neuron having a nonzero signal and output the connecting weight vector between them. Meanwhile, the weight of that Grossberg neuron is then modified with the following formula :

$$V_{ijnew} = V_{ijold} + \beta (y_j - V_{ijold}) k_i$$

where

β is the learning rate of Grossberg layer;

y_j is the j th component of the desired output vector;

k_i is the output of Kohonen neuron i .

As the neural network is trained, the weight vector v_{ij} will converge to the average value of the desired output y_j .

We can see that the Kohonen training operates under the unsupervised mode while the Grossberg training operates under the supervised mode, since the Kohonen layer produces outputs in an indeterminate way and the Grossberg-layer has desired outputs to which it trains. When the network

finishes its training, the most appropriate counterpropagation network model is applied to do time series forecasting.

Backpropagation Training Algorithm

The second neural network methodology being used in the predicting experiment is the popular backpropagation training algorithm. It is designed for training multilayer neural networks. The information processing operation that backpropagation networks are intended to carry out is the approximation of a bounded mapping or function $f: A \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$, from a compact subset A of n -dimensional Euclidean space to a bounded subset $f[A]$ of m -dimensional Euclidean space, by means of training on examples $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k), \dots$ of the mapping, where $y_k = f(x_k)$ [Hecht-Nielsen 1990].

Multilayer networks, such as feedforward backpropagation net, have greater power than single-layer networks only if a nonlinearity is supplied. As we can see in the formulas below, F is the squashing function or the activation function (or the sigmoid function) that provides the needed nonlinearity. x_i is the input signal from previous layer, w_i is the weight and OUT is the output signal of the neuron. I is the number of input neurons.

$$Net = \sum_{i=1}^I x_i w_i, \quad OUT = F(Net) = \frac{1}{(1 + e^{-Net})}$$

The activation function is depicted in Figure 3.

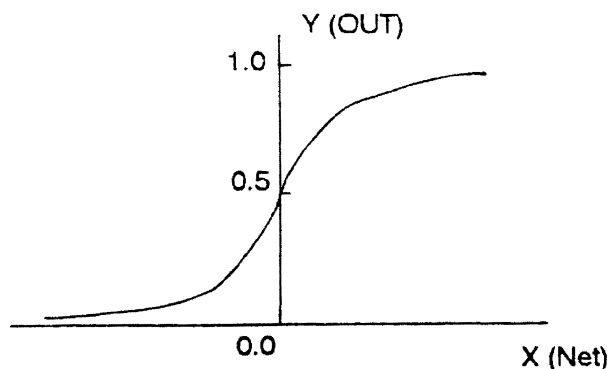


Figure 3. The Activation Function

The goal of training the backpropagation network is to modify the weights (or connections) such that the application of a set of inputs can produce the desired set of outputs. The training of this algorithm is supervised, so each set of inputs is paired with a target vector as the desired output. There is one important thing that should be mentioned here. Before starting the training process, all of the weight vectors ought to be initialized to small random numbers. This ensures that the network will not be affected by large values of weights and prevents certain other training pathologies [Wasserman 1990].

The backpropagation training procedure is described as follows: 1. apply an input vector from the training set to

the network.

2. calculate the output of the network.
3. calculate the error between the target (desired) output and the network output.
4. adjust the connections (weights) of the network in a way to reduce the errors.
5. repeat steps 1 through 4 with all the training vectors until the error for the entire training set is reduced to an acceptable value.

After enough repetitions of these steps, the error (or noise) should be minimized to an acceptable range, and the network is said to be trained. Then the weights of the network model will not be changed and it will be used for recognition and forecasting tasks.

In the program, I included a trainable bias for each neuron in the hidden layer and the output layer. This helps the training process to converge more rapidly and provides a more accurate result in time series forecasting than that without using a bias. Therefore, the formula for the new Net becomes the original one plus theta (the bias). Theta can be considered as a weight connected to +1 [Freeman and Skapura 1990, Wasserman 1990].

$$Net = \sum_{i=1}^I x_i w_i + \theta$$

The input information is propagated from the input

layer through the hidden layers to the output layer in a parallel, distributed manner. The error message that comes from the difference between the actual value and the network output is propagated backward from the output layer to the hidden layers to adjust the weights within the network. The method used to modify the weights in the output layer is different from that used to modify the weights in the hidden layers. To adjust the connections between the output layer and the previous hidden layer, we utilize the generalized delta rule and the momentum strategy. The momentum strategy involves adding a term to the weight adjustment that is proportional to the amount of the previous weight change [Wasserman 1990]. The adjustment equations are as follows:

$$\delta = \text{OUT}_o(1 - \text{OUT}_o)(\text{Target} - \text{OUT}_o)$$

$$\Delta v(n+1) = \eta(\delta \text{OUT}_h) + \alpha[\Delta v(n)]$$

$$v(n+1) = v(n) + \Delta v(n+1)$$

where

α is the momentum coefficient,

η is the training rate coefficient,

OUT_o is the output of the network,

OUT_h is the output of the previous neuron in the
hidden layer,

$\Delta v(n)$ is the weight change at step n ,

$v(n)$ is the weight value at step n ,

$v(n+1)$ is the weight value at step $n+1$ (after
adjustment).

As we have mentioned earlier, for backpropagation network training the error message is propagated backward from the output layer through the network layer by layer to adjust the weights within the network. The value of delta from the output layer is passed back through the same connections to the previous hidden layer to generate a value of delta for each neuron in the hidden layer. The following equations illustrate the adjustment formulas:

$$\delta_H = \text{OUT}_H (1 - \text{OUT}_H) (\sum \delta v)$$

$$\Delta w = \eta \delta_H x_i$$

$$w(n+1) = w(n) + \Delta w$$

where

δ_H is the delta value of a certain neuron in the hidden layer,

δ is the delta value propagated back from the output layer,

η is the training rate coefficient,

Δw is the weight change at step n,

$w(n)$ is the weight value at step n, and

$w(n+1)$ is the new weight value after adjustment.

Neural Network Architectures

The neural network structures of backpropagation and counterpropagation being used in this experiment are different. For the backpropagation neural network model, the number of neurons in the input layer and the hidden

layer are the same, and the number of output neurons in the output layer is the same as the number of forecasting step(s). This selection is the same as that in Patil's thesis. The reason to choose this kind of architecture is to compare the results of the backpropagation nets in this thesis with Patil's NN-PDP results. Different architectures (for example, I-2I-N and I-(I/2)-N, where I is the number of neurons in the input layer and N is the forecasting horizon) also have been evaluated, but their results made no significant difference. In the counterpropagation neural network model, the first (or input) layer contains not only the input information but also the desired output target information. Therefore, the number of neurons in the input layer is the number of components of the input vector plus that of the output vector (the forecasting horizon). Usually, the number of components of the output vector is the forecasting horizon. Since the output of the counterpropagation network is the approximation of both the input information and the desired result, the output layer (or Grossberg layer) has the same number of neurons as that of the input layer. The number of neurons in the Kohonen layer (or the hidden layer) is flexible and always depends on the categories of the input patterns. Because the number of input patterns cannot be known in advance, it is very hard for the researchers to choose the optimal Kohonen structure beforehand. In the test of my program, I designed

the number of Kohonen processing units to be twice the number of input neurons and that turns out to be a satisfactory result.

Table 1 illustrates the neural network architectures being employed to do prediction for different types of time series:

TABLE 1
DIFFERENT BPN AND CPN NETWORK STRUCTURES USED
IN THE FORECASTING EXPERIMENT

	Backpropagation Network Arch.	Counterpropagation Network Arch.
Annual	2-2-1 2-2-2 2-2-4 2-2-6	3-6-3 4-8-4 6-12-6 8-16-8
Quarterly	4-4-1 8-8-1 4-4-2 8-8-2 4-4-4 8-8-4 4-4-6 8-8-6 4-4-8 8-8-8	5-10-5 9-18-9 6-12-6 10-20-10 8-16-8 12-24-12 10-20-10 14-28-14 12-24-12 16-32-16
Monthly	12-12-1 24-24-1 12-12-2 24-24-2 12-12-4 24-24-4 12-12-6 24-24-6 12-12-8 24-24-8 12-12-12 24-24-12 12-12-18 24-24-18	13-26-13 25-50-25 14-28-14 26-52-26 16-32-16 28-56-28 18-36-18 30-60-30 20-40-20 32-64-32 24-48-24 36-72-36 30-60-30 42-84-42

The neural network structure 12-12-8 in the backpropagation algorithm means twelve input neurons, twelve hidden neurons and eight output neurons. The network

structure 20-40-20 for monthly forecasting in the counterpropagation algorithm indicates twelve input signals plus eight desired output signals (i.e. twenty neurons) to the first layer (or the input layer), forty Kohonen neurons and twenty output signals for the approximation of the information to the input layer.

Selection and Classification of Data Sets

The data set selection and classification are the same as those of Sharda and Patil's early work (1990). The data series selected are from the M-111 competition [Makridakis et al. 1982] to estimate the performance of the two predicting models in this experiment. Out of 1001 series collected, only 111 time series are chosen and these data are different from the 111 time series used in Makridakis' previous work in 1979 [Makridakis and Hibon 1979]. These data series were from firms, industries and nations. The sample selection of 111 series from 1001 series are every ninth entry starting from series number 4 (a randomly selected starting point), i.e. 4, 13, 22, 31, ... , 994. All 111 time series will be used for the experiment. The M-111 studied used both the full set of 1001 series and also this subset of 111 series. They are classified into three groups according to the following rules :

annual data series : $1 \leq \text{series number} \leq 181$

quarterly data series : $182 \leq \text{series number} \leq 384$

monthly data series : 385 <= series number

There are 20 annual time series data, 23 quarterly time series data and 68 monthly time series data, among the 111 data series in the subset. This is different from the approach of the data sets classification used in Patil's thesis.

Method of Approach and Analysis

The experimental approach to those data series is to use the reduced sample, $n - k$ observations, to establish the two forecasting models (i.e. the backpropagation network model and the counterpropagation network model). Here, n is the total number of observations in a particular time series while k is the maximal number of future values to be forecast. We take $k = 6, 8, 18$ for yearly, quarterly and monthly data series, respectively. Obviously, we can interpret the N -step-ahead forecasting method in a briefer and clearer statement. That is, for N -step forecasting, N future values will be predicted beyond the first $n-k$ points (i.e. we predict points number $n-k+1, n-k+2, \dots, n-k+N$). Of course, N cannot be greater than k . Here, we take $N = 1, 2, 4, 6, 8, 12, 18$.

The generated predicted outputs will be analyzed by the comparison of MAPE (mean absolute percent error) and Median APE individually. The formula for MAPE is as follows:

$$MAPE_j = [1/(k-N+1)] \left[\sum_{i=1}^{k-N+1} |(A_{ij} - P_{ij}) / A_{ij}| * 100 \right]$$

where

k is equal to 6, 8, 18 for yearly, quarterly and monthly time series respectively, j = 1 to N;

N is the number of values to be predicted;

A_{ij} = the actual value;

P_{ij} = the forecast value.

The following example of the four-step-ahead prediction of yearly time series will demonstrate the above formula. The network architecture used is 2-2-4. A is the array of actual values in the testing set (in this case, there are six actual values, i.e. k=6). The one-dimensional array, A, can be transformed into a two-dimensional array. P is the array of forecasting values.

$$A = [1 \ 2 \ 3 \ 4 \ 5 \ 6] = \begin{matrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \end{matrix} = \begin{matrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \end{matrix}$$

$$P = \begin{matrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{matrix}$$

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \\ P_{11} & P_{12} & P_{13} & P_{14} & & \\ & P_{21} & P_{22} & P_{23} & P_{24} & \\ & & P_{31} & P_{32} & P_{33} & P_{34} \end{matrix} \quad \begin{matrix} i=1 \\ i=2 \\ i=3 \end{matrix}$$

Step	Y+1	Y+2	Y+3	Y+4
MAPE	MAPE1	MAPE2	MAPE3	MAPE4

Here, $MAPE_j = [1/(k-N+1)] * 100 * [(A_{1j} - P_{1j}) / A_{1j} + (A_{2j} - P_{2j}) / A_{2j} +$

$$(A_{3j} - P_{3j}) / A_{3j}]. \quad k - N + 1 = 6 - 4 + 1 = 3 \text{ and } j = 1 \text{ to } 4.$$

If the number of forecasting periods, N , is equal to k , the MAPE calculation would be as follows:

$$MAPE_j = [1 / (k - j + 1)] \left[\sum_{i=1}^{k-j+1} |(A_{ij} - P_{ij}) / A_{ij}| * 100 \right]$$

where $j = 1 \text{ to } k$.

The following example of the six-step-ahead prediction of yearly time series will demonstrate the above formula. The network architecture used is 2-2-6. A is the array of actual values in the testing set (in this case, there are only six actual values. i.e. $k=6$). The one-dimensional array, A , can be transformed into a two-dimensional array. P is the array of forecasting values.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} = \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 0 \\ 3 & 4 & 5 & 6 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 & 0 \\ 5 & 6 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 \end{matrix} = \begin{matrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & A_{16} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & A_{26} \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} & A_{36} \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} & A_{46} \\ A_{51} & A_{52} & A_{53} & A_{54} & A_{55} & A_{56} \\ A_{61} & A_{62} & A_{63} & A_{64} & A_{65} & A_{66} \end{matrix}$$

$$P = \begin{matrix} P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} \\ P_{21} & P_{22} & P_{23} & P_{24} & P_{25} & P_{26} \\ P_{31} & P_{32} & P_{33} & P_{34} & P_{35} & P_{36} \\ P_{41} & P_{42} & P_{43} & P_{44} & P_{45} & P_{46} \\ P_{51} & P_{52} & P_{53} & P_{54} & P_{55} & P_{56} \\ P_{61} & P_{62} & P_{63} & P_{64} & P_{65} & P_{66} \end{matrix}$$

1	2	3	4	5	6						
P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	$i=1$					
	P_{21}	P_{22}	P_{23}	P_{24}	P_{25}	P_{26}	$i=2$				
		P_{31}	P_{32}	P_{33}	P_{34}	P_{35}	P_{36}	$i=3$			
			P_{41}	P_{42}	P_{43}	P_{44}	P_{45}	P_{46}	$i=4$		
				P_{51}	P_{52}	P_{53}	P_{54}	P_{55}	P_{56}	$i=5$	
					P_{61}	P_{62}	P_{63}	P_{64}	P_{65}	P_{66}	$i=6$

Step	Y+1	Y+2	Y+3	Y+4	Y+5	Y+6
MAPE	MAPE1	MAPE2	MAPE3	MAPE4	MAPE5	MAPE6

$$\text{Here, } MAPE_j = [1/(k-j+1)] * 100 * [(A_{1j} - P_{1j})/A_{1j} + (A_{2j} - P_{2j})/A_{2j} + (A_{3j} - P_{3j})/A_{3j} + (A_{4j} - P_{4j})/A_{4j} + (A_{5j} - P_{5j})/A_{5j} + (A_{6j} - P_{6j})/A_{6j}]$$

$k = 6$ and $j = 1$ to 6 .

For Median APE (M-APE), all we have to do is to arrange all the APEs in order and pick the APE value of the middle term (i.e. the M-APE).

Neural Network implementations

If there are not enough data points in a time series, then that time series cannot be used to evaluate the forecasting ability of the network models. The detail of the implementations are shown in the program appendix. In the programs, I require that there should be at least three training patterns to precede the training task. Otherwise, the training of the network models would be of no use and not be able to do forecasting. The methodology of deciding the number of training patterns for N-period-ahead prediction of each algorithm is as follows:

backpropagation: $n - Y_{\max}$ (or Q_{\max} , M_{\max}) - $(I + N) + 1$

counterpropagation: $n - Y_{\max}$ (or Q_{\max} , M_{\max}) - $I + 1$

where

n is the total number of data points in a certain time series,

Y_{\max} , Q_{\max} and M_{\max} are the maximum numbers of

forecasting horizon (the number k) for yearly, quarterly and monthly time series respectively. (i.e. $Y_{max} = 6$, $Q_{max} = 8$ and $M_{max} = 18$)

I is the number of processing elements in the input layer.

The values of both above expressions must be greater or equal to three to maintain the training situation. For instance, ser265, a quarterly time series, has 60 data points so the number of training patterns in this series for the backpropagation training algorithm at four-period-ahead prediction is $60 - 8 - (8 + 4) + 1 = 41$. This series can be used to train and test the neural network model. Another quarterly time series, ser193, with only 20 data points has $20 - 8 - (8 + 4) + 1 = 1$ training patterns for the backpropagation training algorithm at four-step-ahead prediction. This series cannot be used to train and test the neural network model.

The number of training cycles of the backpropagation model was chosen to be 1000 so that the results of this model can be used to compare with the NN-PDP results in Patil's thesis. Different numbers of training cycles, such as 500, 2000 and 10000, also have been tested but their results made no significant difference. For example, the MAPE values for the prediction of yearly, quarterly and monthly data after 10000 training cycles are 6.90, 11.42 and 15.11 which are just slightly worse than the results (6.52,

10.74 and 15.19) of 1000 training cycles. The number of training cycles for the counterpropagation model was chosen according to its weight adjustment algorithm. The learning rates, α and β , are reduced gradually ($\alpha = \alpha/r$ and $\beta = \beta/r$, where $r > 1$ is the decreasing ratio) during the training, and the final values of both of them are decided to be 0.01 approximately. Different decreasing ratios, such as 1.001, 1.005 and 1.01, have been tested and the result for $r = 1.005$ turns out to be better than that of the others. For this reason, the number of training cycles is set to 850 for the counterpropagation model.

When the program reads in a time series from the input file, it checks if there are sufficient observations and processes the data points into two parts, training part and testing part. The training part (or training set) contains several training pairs (or training patterns). Each training pair is composed of an input vector and its desired output vector. Before starting training of the network model, the interconnecting weights must be initialized to small random numbers. It is highly desirable to normalize all input vectors before applying them to the network [Wasserman 1990]. The normalization methods for the two networks are different. For a counterpropagation net, the normalization converts the input training pattern into an n -dimensional unit vector with the same direction. Those weights starting from different input neurons in the input

layer and connecting to the same Kohonen neuron in the Kohonen layer are normalized together to be a unit vector. Meanwhile, those weights starting from the same Kohonen neuron and connecting to different Grossberg neurons in the Grossberg (output) layer are also normalized together to become a unit vector. The normalization formula can be expressed as:

$$x_i \text{ (normalized number)} = x_i / (x_1^2 + x_2^2 + \dots + x_n^2)^{1/2}$$

The normalization mean for a backpropagation neural network model is different from that of a counterpropagation neural network model. The method used in the program is the same as that in Patil's thesis [Patil 1990]. During the training, the program selects the maximum and minimum data points from each training pair (including the input and output values) and each training pattern is normalized by its own maximum and minimum observations. The normalization equation can be written as:

$$x_i \text{ (normalized number)} = (x_i - \text{minimum}) / (\text{maximum} - \text{minimum})$$

Hence the normalized x_i lies on the interval [0,1].

Note that the maximum and minimum numbers are the values over all data points of input vector and target vector in the training pair. But, in the case of testing, the maximum and minimum values are only found in the input vector since the model cannot know the actual testing values in advance.

CHAPTER IV

RESULTS, ANALYSIS AND DISCUSSION

During the testing of the programs, different forecasting periods (or horizons) have been used to evaluate the performance of the two neural network methods. That is, $N = 1, 2, 4, 6$ were used for yearly time series, $N = 1, 2, 4, 6, 8$ were used for quarterly time series and $N = 1, 2, 4, 6, 8, 12, 18$ were used for monthly time series. Here, N stands for the length of forecasting horizons. However, I select only the forecasting horizons of $N = 1, 4$, $N = 1, 4, 8$, and $N = 1, 4, 8, 12$ for yearly data, quarterly data and monthly data, respectively to present in the thesis.

After the programming design tasks are done, the analysis and discussion of the time series forecasting are based on several factors that affect the behavior of the neural networks. The factors are the range initialization of the weight vectors, the learning parameters (alpha, beta and eta), the training rate coefficients and the momentum, etc. Since the weight range initialization is of tremendous importance to the backpropagation and counterpropagation network models, it must be treated with care.

Learning Parameters and Weight Initialization

As far as the counterpropagation training algorithm is concerned, the interconnecting weights are designed to be in the closed interval $0.01 \leq \text{weights} \leq 0.5$. During the testing of the programs, the outputs of this implementation proved to excel that of the larger range $0.01 \leq \text{weights} \leq 0.95$. In Wasserman's book (1990) and Hecht-Nielsen's paper (1987a), they proposed weight adjustment formulas for counterpropagation nets and suggested some feasible values of the training rate coefficients (α , β , a and b).

According to their opinions, the training rate coefficient α is designed to be 0.7, β is assigned to be 0.1 and both a and b (A and B in the program) are equal to 0.3. Different values of those learning parameters, such as 0.1, 0.5, 0.6 and 0.9, have been tried during the testing of the programs. It is observed that the former set (i.e. $\alpha = 0.7$, $\beta = 0.1$, $a = b = 0.3$) of training rate coefficients turns out to be the optimal one.

For the backpropagation training algorithm, the weights were initialized between 0.01 and 0.99 at the beginning. Later on, the weight range was set to be within the interval $(-0.5, 0.5)$. This improved the forecasting performance of the backpropagation network model over the M-111 data series. The same interval has also been tried for the counterpropagation model, but the prediction capability of the network did not improve much and even got worse. According to my opinion, it is not necessary to say that a

certain weight initialization range is suitable for all the neural networks. Each neural network can have its own best choice of the weight range.

The training rate coefficient, eta, and the momentum of the backpropagation net were tested with three different numbers less than one (0.1, 0.5 and 0.9). Table 2 on the following page gives the idea of choosing the optimal values of training rate and momentum. The examination was carried out over time series ser58, ser94, ser229, ser337, ser679, ser769 and ser931. It is observed that the optimal values for eta and the momentum are 0.9 and 0.1, respectively. There is one thing needed to be noticed here: if the training rate coefficient, eta, is low (approximately 0.1), then the forecasting performance becomes poor, especially when both momentum and eta were set to be 0.1. This is different from the result in Patil's thesis.

In order to have an idea of how the effects of seasonality, noise and smooth trend affect the performance of backpropagation model and counterpropagation model over quarterly and monthly time series, the results (MAPE) of backpropagation net (BPN) and counterpropagation net (CPN) over six selected time series are illustrated as follows:

	ser211	ser562	ser229	ser787	ser310	ser832
BPN	33.46	19.43	2.42	4.89	1.30	0.95
CPN	17.04	24.86	14.45	8.22	0.87	0.96

TABLE 2

COMPARISON TABLE OF DIFFERENT VALUES OF MOMENTUM AND LEARNING RATE (MOMENTUM, LEARNING RATE) FOR SER58, SER94, SER229, SER337, SER679, SER769 AND SER931 WITH ONE STEP AHEAD FORECASTING

	ser58	ser94	ser229	ser337	ser679	ser769	ser931
(0.1,0.1)	7.13	3.87	7.72	6.27	30.68	12.13	38.94
(0.1,0.5)	6.94	2.92	3.60	4.10	17.66	10.49	16.67
(0.1,0.9)	6.90	2.73	2.42	3.62	16.34	10.68	16.38
(0.5,0.1)	7.06	3.85	7.46	6.23	28.96	12.14	37.53
(0.5,0.5)	6.92	2.81	3.05	4.03	16.33	11.02	16.18
(0.5,0.9)	6.89	2.68	2.58	4.02	17.19	11.56	17.20
(0.9,0.1)	7.01	3.76	7.06	6.07	26.75	11.99	34.82
(0.9,0.5)	6.90	2.74	2.83	4.00	15.67	11.63	16.66
(0.9,0.9)	6.87	2.65	2.60	4.09	16.94	12.29	18.26

Time series ser229 and ser787 are quarterly and monthly data series with seasonality. Time series ser211 and ser562 are quarterly and monthly data series with noise. Time series ser310 and ser832 are quarterly and monthly data series with smooth trend. It seems that the counterpropagation network model cannot pick up the seasonality of a time series as well as the backpropagation network model does. We can distinguish from the prediction results of ser229 and ser787 that the backpropagation training algorithm is capable of dealing with the seasonality trend of time series properly. Obviously, both training algorithms have trouble with the time series with lots of noise, but so must every method, if the noise is truly random. As regard to a time series with a smooth

trend, both of them can produce output nearly identical to the actual values.

Results: Analysis of Yearly, Quarterly and Monthly Time Series

As we have mentioned in the data selection and classification section, $1 \leq \text{series number} \leq 181$ are annual time series, $182 \leq \text{series number} \leq 384$ are quarterly time series and $385 \leq \text{series number} \leq 1001$ are monthly time series. Those data series have their own structure of trend and the available number of data points. The way to find out that a certain time series has sufficient observations or not is given in the neural network implementation section. If the time series does not have the required number of data points, it may be used to do short term forecasting but may not be able to do long term forecasting. Due to this constraint, a time series which appears in the examinations of the beginning several steps prediction of N-step-ahead forecasting may not appear in the examinations of last few steps prediction of N-step-ahead forecasting.

The training network architecture is the same as the testing network architecture. For the backpropagation model, there is only one network structure (2-2-N) for annual data. The quarterly and monthly data models selected have two different types of structures respectively. That is, 4-4-N and 8-8-N are used for the quarterly group while

12-12-N and 24-24-N are employed for the monthly category. Since the counterpropagation model has a different approach from that of the backpropagation model, the neural network architectures are constructed according to the number of prediction steps. That is, $(2+N)-(2*(2+N))-(2+N)$ is used for the yearly time series, $(4+N)-(2*(4+N))-(4+N)$ and $(8+N)-(2*(8+N))-(8+N)$ are selected for the quarterly data group and $(12+N)-(2*(12+N))-(12+N)$ and $(24+N)-(2*(24+N))-(24+N)$ are used for the monthly data category. As the size of the forecasting steps increased, the network architecture is enlarged, too.

The comparison of the forecasting ability at one-step of NAIIVE 1, NAIIVE 2, exponentially weighted regression method, Autobox, NN-PDP, BPN and CPN is given in Table 23. The results indicate that neural networks (NN-PDP and BPN) defeated the other forecasting techniques on yearly time series prediction. Exponentially weighted regression was better than any other competitor on quarterly and monthly data for N=1-step-ahead forecasting, the only N value for which the regression system was run. Except for the exponentially weighted regression method, it is observed that neural networks provide a better prediction of future values for those time series with short memory (i.e. ser211, ser292, ser400, ser922, etc) than the other competitors while NAIIVE 2 outperform others for the time series with long memory (i.e. ser337, ser454, ser508, ser616, etc).

Table 3 through Table 22 exhibit the multiple steps prediction capability of the backpropagation training algorithm and the counterpropagation training algorithm. Eventually, the backpropagation models outperform the counterpropagation models at most of the competitions except the one-step forecasting of the quarterly time series.

We also need to discuss the results from the following references in this thesis.

1. the M-111 study [Makridakis et al. 1982]
2. Patil's thesis [Patil 1990]
3. the computations by Chandler [Chandler 1992]
4. the paper by Hill, O'Connor and Remus (1991)

In the M-111 study, the given MAPE values of four selected methods for one-step ahead forecasting of the 111 time series are:

NAIVE 1	13.2
NAIVE 2	8.5
Box-Jenkins	10.3
D. Sing Exp	7.8

The NAIVE 1 method uses the last observation of the time series as the prediction for all future times. This means that the forecast of tomorrow's weather will always be the same as today's weather. The NAIVE 2 method deseasonalizes the data using the decomposition method of the ratio-to-moving averages [Makridakis and Wheelwright 1978] then applies the NAIVE method to the deseasonalized

data followed by reseasonalization. The Box-Jenkins forecasting in the M-111 study required a human expert to choose a model based on the standard Box-Jenkins methodology involving the examination of autocorrelation, partial autocorrelation and residual autocorrelation functions, etc. [Box and Jenkins 1976]. It required the most time (on the average about one hour of human effort per series) [Makridakis et al. 1982]. The D. Sing Exp stands for deseasonalized single exponential smoothing method, a simple automatic method. We list it because this method has the lowest MAPE for one-step-ahead prediction.

Patil compared neural networks (the Parallel Distributed Processing package using the backpropagation training algorithm) with the Box-Jenkins method (the Autobox software). Especially, the prediction results of annual time series of the M-111 competition data [Makridakis et al. 1982] in his thesis are much better than the outputs of the other competitors. However, there are a few mistakes in Patil's thesis that need to be corrected. The definitions (extents) of the first two types of time series in Makridakis et al.'s paper (1982) are different from those in Patil's thesis. Makridakis et al. (1982), Pack and Downing (1983) all agreed that the yearly time series group has 181 time series while the quarterly time series group has 203 time series. That is, the yearly time series in M-111 data series should run from ser4 to ser175 (instead of ser112

which is selected by Patil) and the quarterly time series in M-111 data series should start from ser184 (not seri21 selected by Patil) to ser382. The second one is that the mean values and the standard deviations of Table 32 in Patil's thesis seem to be incorrect. The original values of mean MAPEs and standard deviations in that table are as follows:

	Mean MAPE	Standard deviation
Autobox	15.94	15.18
NN-BM	14.92	15.12
NN-PDP	14.67	15.39

According to the correct calculation, the mean MAPE values and the standard deviations for Autobox, NN-BM and NN-PDP should be as follows:

	Mean MAPE	Standard deviation
Autobox	17.62	14.57
NN-BM	16.02	16.76
NN-PDP	12.50	14.29

These results make the advantage of neural networks that Patil claimed over Box-Jenkins methodology (Autobox) more significant. Meanwhile, there are two series on which NN-PDP outperformed Autobox most dramatically, ser58 and ser85. The MAPE values of these two series are shown as follows:

	Autobox	NN-BM	NN-PDP
ser58	72.57	16.17	1.34
ser85	37.00	7.53	0.29

These two series are smooth and easy to predict. These results raise the doubts of whether Autobox might have some bug, or whether Patil might have misused Autobox in some way. The Autobox software was designed according to the Box-Jenkins technique. The average value of the Box-Jenkins method in the M-111 competition paper [Makridakis et al. 1982] is 10.3 which is a lot better than the value, 17.62, of the Autobox software utilized in Patil's thesis. Although Autobox operates automatically and the Box-Jenkins results in the M-111 study were hand-tuned, Autobox has a good reputation and Patil's Autobox results appear to be questionable. Also, some of Patil's NN-PDP results on the annual series (for example, series 31, 49, 85) are hard to understand because I was unable to get close to these MAPE values using methods that should have been similar.

The MAPE value (8.50) of NAIVE 2 in Makridakis et al.'s report (1982) is better than that (10.24) of Chandler's (1992). Chandler [Chandler 1992] tried to reproduce the results for NAIVE 1 and NAIVE 2 in the M-111 report. The average MAPE values he obtained for all 111 series and for Patil's 72 series are as follows:

	111 series	72 series
NAIVE 1	13.66	13.25
NAIVE 2A	10.23	10.18
NAIVE 2B	10.52	10.54

Here, NAIVE 2A and NAIVE 2B use two different methods of deseasonalization. NAIVE 2B is presumed to be closer to the NAIVE 2 used in Makridakis and Wheelwright's report (1978). We can observe that the MAPE value of NAIVE 1 is different from that in M-111 study. It is very strange that the same method but utilized by different researchers should have different results. It is possible that Makridakis et al. might have included some personal judgement and selected the optimal outputs from both NAIVE 1 and NAIVE 2 and then concluded the outputs as the final result of NAIVE 2. Or possibly, a different set of points were predicted (maybe just one point per time series); the M-111 study is not completely clear on this.

Hill, O'Connor and Remus (1991) also used neural networks to do the forecasting experiment over the 111 data series from M-111 study. The results shown in Table 1 and Table 2 of the report were not very clear to the readers. They did not explain what is the forecasting horizon for the result values in the two tables. It would be ambiguous since the values could be the MAPES of one-step prediction or the average numbers of the MAPES of multiple-step prediction. However, from Figure 6 and Figure 7 of the paper, we can see that they obtained good results (approximately MAPE=9 or so) of the prediction for the first step out of the eight-step and eighteen-step predictions for quarterly and monthly time series respectively. There is

one thing that needs to be noted here. They eliminated ser949, which gave an MAPE value over 4000 %, from the 111 series (i.e. only 110 series were being tested in the experiment) to obtain a good result. If they included the series, ser949, in the examination, the result would be very poor. The authors explained that ser949 contained at least three major discontinuities and was therefore unsuitable for the investigation of the performance of the neural network models. Eliminating this series after a very poor prediction was obtained, however, seems to be unfair.

The best results of the neural networks (backpropagation and counterpropagation) employed in this thesis for one-step ahead forecasting may be summarized as follows:

	111 series	72 series
BPN 1	12.72	11.92
CPN 1	13.57	13.46
BPN 2	12.77	13.17
CPN 2	12.13	12.74

Here, the structures of BPN 1 and CPN 1 are different from those of BPN 2 and CPN 2. The details of the difference of the structures selected for BPN and CPN is specified at the beginning of this section. The structures of BPN 1 are 2-2-1, 4-4-1 and 12-12-1 for yearly, quarterly and monthly series respectively while the structures of BPN

2 are 2-2-1, 8-8-1 and 24-24-1 for yearly, quarterly and monthly series respectively. The structures of CPN 1 are 3-6-3, 5-10-5 and 13-26-13 for yearly, quarterly and monthly series respectively while the structures of CPN 2 are 3-6-3, 9-18-9 and 25-50-25 for yearly, quarterly and monthly series respectively. We can observe from the comparison of the results of this thesis with the results of NAIIVE 1, NAIIVE 2, Autobox, NN-PDP, etc. that BPN 1, 2 and CPN 1, 2 are better than NAIIVE 1 and Autobox as used by Patil but worse than the other forecasting technologies (NAIIVE 2 and Exponentially weighted regression method).

Some Relative Examinations of Neural Networks

In order to evaluate the ability of neural networks to capture the underlying trend (or mapping) of time series, two examples that consist of non-noise trend have been testified in this experiment. One is the linear equation, $y = x + 1$, the other is the example, $y = x^{1.37}$, being examined by Hill, O'Connor and Remus [Hill, O'Connor and Remus 1991]. Both the backpropagation net and the counterpropagation net have amazing results for these two instances. The linear equation is considered as yearly data while the equation, $y = x^{1.37}$, is taken as quarterly and monthly data. For the linear trend, the MAPE value of the backpropagation model is only 0.06 whereas the MAPE value of the counterpropagation

model is 2.41. The linear equation with seasonality has also been examined. The backpropagation net has an MAPE value of 3.86 while the counterpropagation net has 9.02. This indicates that the former model can approximate the underlying seasonality mapping more accurately than the latter. For the model, $y = x^{1.37}$, the MAPE value for the backpropagation model is 0.46 and the MAPE value for the counterpropagation model is 0.86.

We can observe an interesting possibility from the result above. It is easy for neural networks to identify (or learn) the functional forms automatically. In the Hill, O'Connor and Remus' paper (1991), they also specified that neural networks have the property to approximate the functional forms automatically but it would be difficult for a forecaster to correctly identify and capture it. The property is more likely to occur in the quarterly and the monthly data than in the annual data, since there is more chance for the quarterly and the monthly data to contain certain underlying pattern (or trend) [Hill, O'Connor and Remus 1991].

CHAPTER V

CONCLUSIONS AND FUTURE WORK

During the experiment of this research, not only the neural networks (backpropagation net and counterpropagation net) have been examined for the time series prediction but also several other forecasting methods (NAIVE 1, 2, Exponentially Weighted Regression, ..., etc). It is observed (see the summarized MAPE results below) that the neural network is the best forecaster on the yearly type time series (NN-PDP was the neural net package used in Patil's thesis, and BPN is the backpropagation net used in this thesis), whereas the exponentially weighted regression method dominates on the one-step-ahead prediction of quarterly and monthly data series. Every method has difficulty when dealing with the data series with noise and irregular trends (for example, ser481, ser715, etc. see Table 23).

Yearly time series. (BPN: 2-2-1, CPN: 3-6-3)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	10.25	10.25	24.65	1.88	6.38	7.57	7.02

Quarterly time series. (BPN: 8-8-1, CPN: 9-18-9)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	10.65	9.62	18.88	11.02	11.49	9.82	7.97

Monthly time series. (BPN: 24-24-1, CPN: 25-50-25)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	14.91	10.30	15.91	15.86	15.30	14.49	9.49

Quarterly time series. (BPN: 4-4-1, CPN: 5-10-5)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	10.65	9.62	18.88	10.47	10.74	10.43	7.97

Monthly time series. (BPN: 12-12-1, CPN: 13-26-13)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	16.48	10.10	15.91	12.52	13.45	16.12	9.37

In accordance with the experiments for counterpropagation neural nets, if the architecture has more processing elements in the hidden layer than that of the input layer then the network model can have better performance. If the forecasting horizon is of multiple-step length (i.e. more than two-step-ahead) then the backpropagation training algorithm is considered to be a more appropriate technique than the counterpropagation training algorithm. Meanwhile, the backpropagation model is able to detect the seasonality of a time series while the counterpropagation model cannot.

The following summarized MAPE values are the multiple-step forecasting results of backpropagation models and counterpropagation models over quarterly time series and monthly time series. The annual time series forecasting output are presented in Appendix A and Appendix D.

MAPE AND M-APE VALUES OF QUARTERLY FORECAST OF BPN
AT 8_STEP WITH ARCH. 4-4-8

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	8.42	6.20	10.46	9.89	11.39	9.75	13.20	12.53

step	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	13.89	10.18	15.43	15.05	15.01	12.64	15.89	15.89

MAPE AND M-APE VALUES OF QUARTERLY FORECAST OF BPN
AT 8_STEP WITH ARCH. 8-8-8

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	9.01	6.15	9.93	9.10	13.89	12.13	14.56	14.88

step	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	17.22	13.33	17.48	17.23	17.43	12.85	18.34	18.34

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF BPN
AT 4_STEP WITH ARCH. 12-12-4

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	14.93	11.19	15.56	11.96	14.56	10.68	16.38	12.46

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF BPN
AT 8_STEP WITH ARCH. 12-12-8

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	14.58	10.62	16.33	12.18	17.01	12.06	17.80	12.70

step	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	17.85	12.89	17.88	13.36	18.63	13.16	19.78	16.91

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF BPN
AT 4_STEP WITH ARCH. 24-24-4

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	14.78	11.42	15.36	13.11	15.76	13.51	17.12	15.39

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF BPN
AT 8_STEP WITH ARCH. 24-24-8

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	16.76	12.63	16.47	13.69	14.59	10.38	14.90	12.45

step	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	14.70	11.27	15.49	13.82	16.18	14.44	17.68	16.25

MAPE AND M-APE VALUES OF QUARTERLY FORECAST OF CPN
AT 8_STEP WITH ARCH. 12-24-12

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	9.25	6.36	12.80	11.32	15.02	11.73	14.48	14.00

step	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	17.64	11.57	19.05	19.18	22.93	16.15	24.98	24.98

MAPE AND M-APE VALUES OF QUARTERLY FORECAST OF CPN
AT 8_STEP WITH ARCH. 16-32-16

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	11.03	8.59	14.76	12.80	18.70	13.64	19.30	16.44

step	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	23.42	15.92	23.07	23.29	25.45	17.38	30.64	30.64

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF CPN
AT 4_STEP WITH ARCH. 16-32-16

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	15.91	11.31	20.31	15.36	20.72	17.02	22.65	18.64

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF CPN
AT 8_STEP WITH ARCH. 20-40-20

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	16.33	10.68	17.97	13.66	20.18	14.47	24.68	18.03

step	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	23.42	17.47	22.85	18.52	25.48	18.56	25.34	21.23

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF CPN
AT 4_STEP WITH ARCH. 28-56-28

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	14.90	9.90	17.86	13.18	19.23	14.97	20.82	16.93

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF CPN
AT 8_STEP WITH ARCH. 32-64-32

step	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	15.03	10.27	17.32	12.71	18.36	13.74	20.88	17.86

step	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
mean	22.01	17.65	22.62	18.04	23.71	17.66	25.00	21.73

There are a number of factors, such as the learning parameters, neural network structures and the training procedure, that can affect the performance of neural networks a lot. With a proper predetermined training procedure and an appropriate network architecture, a neural network is capable of approximating the underlying pattern of the time series and reproducing the underlying mapping properly.

This thesis has just accomplished a small part of the utilization of neural networks for time series forecasting. There is still much room for the researchers interested in the time series prediction to investigate and explore the potential of neural networks. The future works may be described as follows:

1. Employing some other neural networks, such as the Hopfield net, recurrent networks and so on, to execute the experiments of time series forecasting or decision making problems.
2. Evaluating the performance of neural networks with some other traditional prediction techniques.
3. Developing some other training procedures or learning methods and applying to the neural networks

to compare the result with the existing training algorithms.

BIBLIOGRAPHY

- Adam, Jr. E.E., "Individual item forecasting model evaluation", *Decision Sciences*, 4, 1973, 458-70.
- Brown, R.G., Smoothing, forecasting and prediction of discrete time series, Prentice-Hall, Englewood Cliffs, NJ, 1963.
- Box, G.E.P. and Jenkins, G.M., Time series analysis: forecasting and control, Holden-Day, San Francisco, CA. 1976.
- Carpenter, G. and Grossberg, S., "ART2 : self-organization of stable category recognition codes for analog input patterns", *Applied Optics*, 26(23), 1987, 4919-30.
- Carpenter, G. and Grossberg, S., Neural networks for vision and image processing, MIT Press, Cambridge, MA, 1992.
- Chandler, J.P., "The results of NAIVE I and NAIVE II over M-111 competition data", private communication, 1992.
- Chatterjee, S. and Bard, J.F., "A comparison of Box-Jenkins time series model with autoregressive processes", *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(2), March/April 1985, 252-9.
- Denning, P.J., "Neural Networks", *American Scientist*, 80(5), Sep-Oct 1992, 426-9.
- Fogel, D.B., "An information criterion for optimal neural

- networks selection", IEEE Transactions on Neural Networks, 2(5), Sep. 1991, 490-7.
- Freeman, J.A., "Neural networks for machine vision application: the spacecraft orientation demonstration", e^xponent: Ford Aerospace Technical Journal, Fall 1988, 16-20.
- Freeman, J.A. and Skapura, D.M., Neural Networks Algorithms, Applications, and Programming Techniques, Addison-Wesley, Reading, Massachusetts, 1990.
- Harrison, P.J. and Akram, M., Time series analysis: theory and practice 3, North-Holland, New York, 1982, 19-38.
- Hecht-Nielsen, R., "Counterpropagation networks", In Proceedings of the IEEE First International Conference on Neural Networks, 2, San Diego, CA: SOS Printing, 1987a, 19-32.
- Hecht-Nielsen, R., "Counterpropagation networks", Applied Optics, 26(23), 1987b, 4979-84.
- Hecht-Nielsen, R., "Applications of counterpropagation networks", Neural Networks, 1(2), 1988, 131-9.
- Hecht-Nielsen, R., Neurocomputing, Addison-Wesley, Reading, Massachusetts, 1990.
- Hill, T., et al, "Neural network model for forecasting and decision making", Working Paper, University of Hawaii, Feb. 28 1992.
- Hill, T., O'Connor, M. and Remus, W., "Neural network models for time series forecasts", Working Paper, University

- of Hawaii, Feb. 7 1991.
- Hinton, G.E. and Sejnowski, T.J., "Learning and relearning in Boltzmann machines", *Parallel Distributed Processing, Vol 1: Foundations*, Cambridge, MA: MIT Press, 1988, 282-317.
- Hopfield, J.J., "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of Natl. Acad. Sci. USA*, 79(2), 1982, 2554-8.
- Hopfield, J.J. and Tank, D.W., "Computing with neural circuits: a model", *Science*, 233, Aug. 1986, 625-34.
- Knight, K., "Connectionist ideas and algorithms", *Communications of the ACM*, 33(11), Nov. 1990, 59-74.
- Kohonen, T., Self-organization and associative memory, Third edition, New York: Springer-Verlag, 1989.
- Kosko, B., Neural networks and fuzzy system, Prentice-Hall, Englewood Cliffs, NJ, 1991.
- Lippmann, R.P., "An introduction to computing with neural nets", *IEEE ASSP Magazine*, 4(2), Apr. 1987, 4-23.
- Logar, A.M., Corwin, E.M. and Oldhan, W.J.B., "Predicting acid concentrations in processing plant effluent : an application of time series prediction using neural networks", *Proceedings of the 1992 ACM/SIGAPP SYMPOSIUM ON APPLIED COMPUTING*, 2, MAR. 1992, 651-8.
- Makridakis, S. and Hibon, M., "Accuracy of forecasting: an empirical investigation (with discussion)", *Journal of*

- the Royal Statistical Society, (A), 142, Part 2, 1979, 97-145.
- Makridakis, S. and Wheelwright, S.C., Forecasting, methods and applications, John Wiley and Sons, New York, 1978.
- Makridakis, S. and Wheelwright, S.C., Interactive forecasting: univariate and multivariate methods, Second edition, Holden-Day, San Francisco, CA. 1978.
- Makridakis, S., et al, The forecasting accuracy of major time series methods, John Wiley and Sons, New York, 1984.
- Makridakis, S., et al, "The accuracy of extrapolation (time series) methods: results of a forecasting competition", Journal of forecasting, 1, 1982, 111-53.
- Masson, E. and Wang, Y.J., "Introduction to computation and learning in artificial neural networks", European Journal of Operational Research, 47, 1990, 1-28.
- McLeod, G., Box-Jenkins in practice, Gwilym Jenkins and Partners, Lancaster (U.K.) 1983.
- Mehra, P. and Wah, B.W., Artificial neural networks: concepts and theory, IEEE Computer Society Press, ISBN 0-8186-8997-8, CA, 1992.
- Mengel, S. and Lively, W., "Using a neural network to predict student responses", Proceedings of the 1992 ACM/SIGAPP SYMPOSIUM ON APPLIED COMPUTING, 2, Mar. 1992, 669-76.
- Minsky, M. and Papert, S., Perceptrons, MIT Press, Cambridge

- MA, 1969.
- Montgomery, D. and Weatherby, G., "Modeling and forecasting time series using transfer function and intervention methods", *AIIE Transactions*, 12(4), Dec. 1980, 289-307.
- Pack, D.J. and Downing, D.J., "Why didn't Box-Jenkins win (again)?", *Proceedings, Third International Symposium on Forecasting*, Philadelphia, 1983.
- Page, E.W. and Tagliarini, G.A., "Algorithm development for neural networks", *Proceedings of SPIE-the International Society for Optical Engineering*, 880, 1988, 11-9.
- Patil, R., Neural networks as forecasting experts: test of dynamic modeling over time series data, M.S. Thesis, Computing and Information Sciences Department, Oklahoma State University, Stillwater, OK. 1990.
- Patil, R. and Sharda, R., "A test of forecasting using backpropagation", *Oklahoma Symposium for Artificial Intelligence*, 1989, 249-55.
- Rumelhart, D.E. and McClelland, J.L., "Learning internal representations by error propagation", *Parallel Distributed Processing, Vol 1: Foundations*, Cambridge, MA: MIT Press, 1988, 318-62.
- Sastri, T., English, J.R. and Wang, G.N., "Neural networks for time series model identification and change detection", *Texas A&M University Working Paper INEN/QC/WP/03/07-90*, 1990.
- Sharda, R. and Ireland, T.C., "A test of Box-Jenkins

- forecasting expert systems for microcomputers",
ORSA/TIMS Meeting, New Orleans, May 1987.
- Sharda, R. and Patil, R., "Neural networks as forecasting experts: an empirical test", International Joint Conference on Neural Networks, IJCNN-WASH-D.C., 2, Jan. 15-19 1990, 491-4.
- Suh, K., "An investigation of the use of feedforward neural networks for forecasting", Ph.D. Dissertation, Graduate School of Management, Kent State University, Kent, OH., Aug. 1991.
- Tang, Z., Almeida, C. and Fishwick, P.A., "Time series forecasting using neural networks vs. Box-Jenkins methodology", SIMULATION, Nov. 1991, 303-10.
- Vemuri, V.R., Artificial neural networks: concepts and control applications, IEEE Computer Society Press, ISBN 0-8186-9069-0, CA, 1992.
- Wasserman, P.D., Neural computing: theory and practice, New York, NY: Van Nostrand Reinhold, 1990.
- Weiss, S.M. and Kulikowski, C.A., Computer systems that learn, Morgan Kaufmann, San Mateo, CA, 1991.

APPENDIXES

APPENDIX A

TEST RESULTS OF BPN WITH YEARLY DATA

TABLE 3
 YEARLY FORECAST OF BPN AT
 1_STEP WITH ARCH. 2-2-1

series	y+1	
	mape	m-ape
ser4	11.23	6.77
ser13	7.46	6.70
ser22	0.23	0.11
ser31	9.22	6.36
ser40	4.27	3.64
ser49	26.16	25.21
ser58	6.90	3.06
ser67	3.55	3.20
ser76	5.26	3.15
ser85	3.86	2.62
ser94	2.73	0.91
ser103	1.44	1.02
ser112	0.66	0.55
ser121	2.76	2.28
ser130	10.66	9.58
ser139	6.96	6.00
ser148	5.62	3.29
ser157	11.74	12.83
ser175	3.18	1.60
mean	6.52	5.20

TABLE 4

MAPE AND M-APE VALUES OF YEARLY FORECAST OF BPN
AT 4_STEP WITH ARCH. 2-2-4

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser4	11.46	11.23	9.89	10.03	11.59	9.02	13.29	15.96
ser13	7.22	6.71	7.45	8.16	9.60	4.35	13.86	14.24
ser22	0.28	0.13	0.64	0.59	1.17	1.14	1.61	1.52
ser31	9.09	6.24	15.00	15.03	16.96	10.04	8.25	7.77
ser40	4.83	2.18	6.62	6.03	7.51	6.59	9.88	11.03
ser49	26.61	27.16	33.69	35.33	30.29	22.56	30.01	32.15
ser58	10.74	7.15	14.58	12.18	21.11	12.03	27.68	28.36
ser85	3.78	3.30	4.58	4.51	5.50	3.49	6.10	6.57
ser112	1.08	0.83	2.24	2.41	3.73	3.22	5.24	5.35
ser121	4.74	4.28	8.56	8.43	12.45	12.16	16.33	16.17
ser157	11.11	12.64	11.20	14.03	8.17	6.40	4.12	4.64
mean	8.27	7.44	10.40	10.61	11.64	8.27	12.40	13.07

APPENDIX B

TEST RESULTS OF BPN WITH QUARTERLY DATA

TABLE 5
 QUARTERLY FORECAST OF BPN AT
 1_STEP WITH ARCH. 4-4-1

series	y+1	
	mape	m-ape
ser184	12.44	8.76
ser193	31.86	9.06
ser202	4.59	1.99
ser211	33.46	29.22
ser220	33.77	21.69
ser229	2.42	1.48
ser238	5.54	5.79
ser265	3.40	3.57
ser283	2.49	1.05
ser292	10.30	5.91
ser301	2.56	2.00
ser310	1.30	1.11
ser319	3.35	2.63
ser328	3.51	2.35
ser337	3.62	1.98
ser346	16.93	5.74
ser355	2.74	2.57
ser364	2.38	2.63
ser382	27.37	29.17
mean	10.74	7.30

TABLE 6

MAPE AND M-APE VALUES OF QUARTERLY FORECAST OF BPN
AT 4_STEP WITH ARCH. 4-4-4

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser184	8.99	4.68	12.53	8.85	12.68	7.85	13.76	14.58
ser193	25.21	7.42	40.09	39.89	45.05	24.05	52.15	55.35
ser202	3.81	2.74	4.24	2.94	4.84	1.97	5.22	2.44
ser211	17.79	15.56	23.04	23.36	20.95	18.41	25.76	22.67
ser220	27.22	28.40	29.13	31.53	26.09	25.38	24.07	18.73
ser229	4.84	3.22	3.88	3.12	3.98	3.03	3.22	1.47
ser238	5.99	5.21	6.28	6.34	7.03	6.94	8.99	10.68
ser265	4.13	2.56	7.65	8.23	11.82	11.49	15.75	16.68
ser283	3.53	1.66	5.82	5.55	8.17	6.39	10.52	10.85
ser292	8.87	4.70	9.06	6.49	6.21	4.09	6.62	4.23
ser301	3.69	2.93	6.51	6.76	8.09	7.47	9.73	10.01
ser310	3.83	3.61	5.29	5.20	6.81	6.50	8.57	8.54
ser319	4.36	4.16	6.19	6.47	8.82	9.91	11.61	13.74
ser328	3.57	3.26	4.10	3.23	4.33	3.73	3.87	2.66
ser337	2.66	1.11	3.09	1.51	3.71	1.65	4.28	2.73
ser346	15.25	8.31	21.86	10.19	25.25	10.16	31.45	14.75
ser355	3.07	2.55	4.40	4.82	5.45	4.28	4.77	4.86
ser364	2.98	2.75	3.63	4.08	4.61	4.17	6.32	7.11
ser382	28.87	29.64	38.42	36.15	36.65	40.18	42.00	52.81
mean	9.40	7.08	12.38	11.30	13.19	10.40	15.19	14.47

TABLE 7
 QUARTERLY FORECAST OF BPN AT
 1_STEP WITH ARCH. 8-8-1

series	y+1	
	mape	m-ape
ser184	11.43	8.63
ser193	42.08	21.56
ser202	4.48	2.36
ser211	34.79	16.28
ser220	41.97	35.57
ser229	3.11	1.54
ser238	6.74	7.64
ser265	3.88	2.66
ser283	2.49	1.11
ser292	9.73	4.73
ser301	2.71	2.20
ser310	1.36	1.15
ser319	3.48	3.14
ser328	3.07	2.14
ser337	4.29	1.68
ser346	14.12	5.27
ser355	2.76	2.19
ser364	2.25	2.32
ser382	23.64	24.98
mean	11.49	7.74

TABLE 8

MAPE AND M-APE VALUES OF QUARTERLY FORECAST OF BPN
AT 4_STEP WITH ARCH. 8-8-4

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser184	8.11	4.98	9.69	8.98	10.77	9.65	16.47	16.51
ser202	3.68	1.61	4.32	3.08	5.03	2.41	5.32	2.63
ser211	26.02	19.94	16.42	12.89	22.52	13.67	16.56	12.85
ser220	35.07	27.06	36.34	35.85	43.14	41.09	36.07	34.30
ser229	3.05	2.94	3.20	2.16	2.79	1.63	2.02	1.78
ser238	6.95	7.63	9.12	8.98	10.53	8.70	10.97	13.16
ser265	4.48	3.72	8.33	9.11	12.40	11.66	14.47	15.03
ser283	4.37	2.52	6.41	6.22	8.43	6.69	10.62	10.96
ser292	9.50	3.42	6.20	2.63	7.06	4.58	5.25	4.84
ser301	4.51	3.94	5.51	6.12	7.26	6.69	9.20	9.57
ser310	5.43	5.20	6.24	6.01	7.08	6.79	8.60	8.58
ser319	5.03	4.68	7.32	7.31	9.46	11.24	11.83	13.45
ser328	2.79	1.27	3.41	2.61	3.69	2.47	4.09	3.53
ser337	2.94	1.42	3.49	1.90	4.16	2.71	4.76	3.68
ser346	13.41	7.58	15.60	8.29	22.50	9.33	31.00	13.86
ser355	6.34	4.79	11.18	12.51	12.82	12.57	11.88	12.06
ser364	2.35	2.35	3.14	2.55	4.49	3.66	6.10	7.16
ser382	25.96	25.18	33.75	35.19	36.73	36.70	38.90	51.24
mean	9.44	7.24	10.54	9.58	12.83	10.68	13.56	13.07

APPENDIX C

TEST RESULTS OF BPN WITH MONTHLY DATA

TABLE 9

MONTHLY FORECAST OF BPN AT
1_STEP WITH ARCH. 12-12-1

series	y+1	
	mape	m-ape
ser391	14.92	8.19
ser400	10.26	7.85
ser409	31.22	25.92
ser418	19.74	20.40
ser427	8.60	7.10
ser436	1.94	1.40
ser445	18.32	15.20
ser454	9.09	6.74
ser463	10.96	8.80
ser472	21.00	13.28
ser481	28.31	18.88
ser490	12.62	11.40
ser499	8.55	6.19
ser508	6.79	4.96
ser526	9.83	7.43
ser535	55.18	34.23
ser544	2.63	1.97
ser562	19.43	14.13
ser571	18.16	17.95
ser580	1.31	1.18
ser589	3.84	2.26
ser598	13.06	5.03
ser616	6.57	6.90
ser625	19.95	13.32
ser634	22.84	16.45
ser643	16.92	7.38
ser652	16.35	9.53
ser661	16.24	14.97
ser670	28.81	27.64
ser679	16.34	13.78
ser688	9.03	3.53
ser697	3.00	3.13
ser706	7.75	7.35
ser715	50.20	16.52
ser724	17.52	15.06
ser733	23.65	20.84
ser742	2.34	1.88
ser751	9.24	7.32
ser760	8.46	6.85
ser769	10.68	6.45
ser778	4.15	2.10
ser787	4.89	2.64

TABLE 9 (Continued)

series	y+1	
	mape	m-ape
ser796	21.39	19.46
ser805	1.30	1.35
ser814	0.86	0.54
ser823	0.68	0.47
ser832	0.95	0.86
ser841	11.26	7.95
ser850	5.52	4.09
ser868	9.44	6.38
ser877	5.30	3.73
ser886	33.52	33.88
ser895	16.26	14.95
ser904	4.63	2.18
ser913	36.60	38.70
ser922	4.79	4.45
ser931	16.38	13.17
ser940	27.31	22.30
ser949	81.19	80.12
ser958	8.08	5.95
ser967	6.64	4.97
ser985	33.31	16.34
ser994	10.59	9.64
mean	15.19	11.68

TABLE 10

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF BPN
AT 12_STEP WITH ARCH. 12-12-12

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	9.17	8.03	11.16	11.23	12.27	12.10	11.67	10.44
ser409	22.91	20.90	21.53	15.23	17.29	15.21	33.00	22.82
ser418	22.85	21.51	24.59	23.57	15.31	8.43	15.21	8.62
ser427	7.23	6.95	9.98	8.63	8.82	4.69	10.14	8.14
ser436	2.87	2.47	2.67	2.37	2.76	2.77	3.74	3.97
ser445	14.36	13.04	18.37	16.91	19.31	19.37	12.95	7.28
ser454	19.11	13.91	23.65	20.54	21.99	19.68	21.94	21.90
ser463	18.72	18.73	19.41	15.65	27.33	25.72	20.34	14.98
ser472	22.65	21.69	21.26	12.96	23.68	10.47	26.45	10.60
ser481	22.96	7.68	34.26	19.82	45.04	24.74	38.89	27.57
ser490	14.28	11.33	14.01	12.16	14.08	12.28	14.84	12.84
ser499	13.56	12.55	12.32	11.43	13.69	14.17	14.46	17.69
ser508	14.45	15.22	15.48	16.56	15.55	16.60	12.81	12.81
ser526	10.71	7.16	12.28	9.75	17.65	17.53	14.42	13.04
ser544	4.65	3.20	6.25	4.84	6.59	4.96	6.64	6.27
ser562	43.56	23.80	50.24	33.60	39.01	28.92	40.80	30.38
ser571	19.45	18.51	21.46	21.83	17.50	17.33	20.61	20.38
ser580	1.40	1.28	1.88	1.82	2.57	2.73	2.97	3.29
ser589	5.22	2.77	7.38	6.15	8.33	7.60	9.30	10.67
ser598	11.23	5.38	11.10	6.64	13.96	8.22	14.86	8.27
ser616	7.64	7.04	7.12	5.55	6.67	5.37	7.95	8.78
ser634	13.84	12.17	13.91	11.02	20.66	15.66	18.48	10.90
ser643	18.32	10.45	21.41	18.85	21.03	11.65	18.90	12.01
ser652	16.31	7.58	16.57	7.19	18.84	10.63	20.54	19.24
ser661	15.48	10.00	17.33	13.97	20.36	19.33	22.34	14.64
ser670	33.94	32.90	24.72	13.97	32.92	26.67	40.30	36.93
ser679	43.30	22.90	45.96	24.80	37.19	21.05	30.04	18.04
ser688	10.21	6.86	12.48	7.94	11.92	7.91	15.62	7.10
ser697	2.96	2.34	4.50	4.88	2.03	1.63	2.89	2.99
ser706	3.11	2.85	3.39	2.30	3.97	3.38	4.16	2.97
ser715	61.32	9.21	64.54	21.17	56.47	19.77	64.80	36.32
ser724	18.19	14.05	21.54	19.13	19.62	14.22	20.41	14.26
ser733	17.51	6.92	16.99	8.75	16.83	8.65	16.48	7.38
ser742	2.65	1.60	2.97	2.44	3.28	2.70	3.07	2.62
ser751	8.40	6.53	7.60	9.20	7.29	6.97	8.47	7.14
ser760	11.24	9.99	11.48	10.14	9.95	7.49	9.98	8.80
ser769	8.41	4.08	9.58	7.37	10.57	9.54	9.32	7.19
ser787	6.82	5.80	6.02	4.53	6.57	4.10	8.24	8.63
ser796	15.87	10.51	16.17	14.31	17.62	13.13	17.49	12.62
ser805	1.76	1.72	2.93	2.90	4.02	3.95	5.29	5.36
ser823	0.85	0.79	0.97	0.81	1.09	0.74	1.23	0.97
ser832	4.90	4.58	5.49	5.07	6.36	5.77	7.15	6.52

TABLE 10 (Continued)

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser877	3.72	2.89	4.34	2.70	4.23	3.49	3.92	2.33
ser904	5.67	4.04	9.07	7.58	5.59	3.92	7.45	6.42
ser913	30.45	10.95	38.21	33.77	41.60	39.04	41.77	47.69
ser922	5.31	4.95	6.66	7.33	7.80	7.49	8.35	8.20
ser958	23.31	24.12	24.64	26.24	17.60	14.78	9.06	5.54
ser967	15.65	10.16	20.19	15.84	26.09	21.40	32.55	38.65
mean	14.76	10.08	16.17	12.11	16.27	12.17	16.71	13.26

TABLE 10 (Continued)

series	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	9.95	8.23	9.45	8.36	9.85	10.30	7.28	7.68
ser409	46.97	30.44	20.78	21.23	13.03	8.08	19.14	14.19
ser418	15.92	14.56	11.49	7.79	18.84	14.83	20.89	17.67
ser427	10.71	8.63	11.16	9.46	10.86	10.29	6.54	5.54
ser436	5.06	5.46	5.96	6.20	6.89	7.08	6.63	6.63
ser445	16.61	14.85	25.38	25.47	24.30	21.60	17.48	16.71
ser454	18.54	13.11	20.40	18.57	19.43	9.08	18.64	19.92
ser463	22.03	18.76	30.78	30.69	19.39	16.42	18.01	10.26
ser472	27.09	18.04	19.22	13.70	13.98	9.91	11.91	10.11
ser481	27.59	11.28	52.84	32.71	47.00	33.14	32.18	24.04
ser490	14.60	17.17	14.74	14.22	13.69	13.78	11.22	10.21
ser499	17.28	18.17	18.34	21.74	16.43	17.28	19.81	22.40
ser508	12.24	11.37	11.55	11.94	13.38	12.48	11.38	9.76
ser526	16.22	15.04	16.35	15.74	12.75	10.17	10.48	10.37
ser544	6.68	5.46	6.65	5.48	6.21	4.11	5.70	5.38
ser562	40.73	34.73	46.62	35.85	34.46	23.93	31.32	32.23
ser571	16.20	15.01	15.68	14.44	16.20	14.14	17.35	14.54
ser580	3.37	3.50	4.36	4.16	4.81	4.18	5.05	5.62
ser589	10.66	12.90	11.63	12.43	12.07	10.05	12.67	8.36
ser598	15.99	9.84	15.67	12.66	19.47	11.31	20.11	18.36
ser616	7.27	7.20	7.51	6.42	7.06	4.86	7.05	6.74
ser634	16.30	8.29	19.83	20.03	11.69	8.75	20.37	24.07
ser643	14.70	9.08	22.39	23.89	10.41	6.90	15.07	15.36
ser652	17.72	14.59	16.76	15.47	25.18	22.91	27.23	25.40
ser661	25.32	14.49	23.94	18.79	20.30	19.10	14.25	14.48
ser670	32.24	26.83	32.45	29.56	33.82	28.68	41.82	39.80
ser679	13.62	8.49	20.26	12.64	12.13	9.03	34.39	32.81
ser688	22.21	6.97	17.67	5.54	13.97	7.60	14.21	8.87
ser697	2.16	2.03	2.78	2.33	2.30	1.83	3.51	3.81
ser706	5.55	4.62	6.80	6.63	6.65	6.14	8.57	7.45
ser715	61.27	16.05	65.96	18.11	68.91	16.67	72.52	17.09
ser724	22.35	17.93	22.73	19.64	21.78	14.98	21.51	20.01
ser733	16.06	7.50	16.64	5.90	18.82	5.27	17.75	8.46
ser742	3.76	3.29	3.46	3.11	3.15	2.01	3.16	2.85
ser751	8.57	7.49	9.42	9.23	9.55	9.64	10.90	8.64
ser760	10.19	7.61	11.26	9.42	8.67	10.99	12.80	14.06
ser769	9.80	9.34	12.38	11.51	15.82	15.32	11.66	10.34
ser787	14.01	13.81	10.09	10.38	8.59	7.64	8.55	6.92
ser796	31.68	24.91	16.61	13.92	17.33	14.20	19.30	7.82
ser805	6.33	6.63	7.18	6.89	8.08	8.16	8.97	8.71
ser823	1.31	1.18	1.39	1.34	1.29	0.75	1.19	1.24
ser832	7.95	7.19	8.99	8.79	9.87	9.56	10.36	10.42
ser877	3.80	2.76	3.31	2.77	2.94	2.39	3.85	3.15
ser904	4.97	2.47	5.02	2.72	6.28	3.45	5.13	2.60
ser913	43.98	46.72	54.49	60.28	65.46	70.11	73.73	76.47

TABLE 10 (Continued)

series	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser922	10.44	11.25	14.15	13.51	19.03	17.72	24.06	26.18
ser958	8.63	3.09	7.77	5.09	6.90	5.51	18.64	21.23
ser967	33.51	37.49	34.92	39.72	37.00	31.88	36.96	30.49
mean	16.87	12.83	17.60	14.72	16.79	13.01	17.74	15.11

TABLE 10 (Continued)

series	y+9		y+10		y+11		y+12	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	10.29	9.47	8.16	7.36	11.04	7.83	9.39	8.17
ser409	26.98	17.96	22.58	23.10	13.97	13.41	14.79	15.21
ser418	15.69	13.07	16.60	13.48	19.95	9.49	14.76	9.40
ser427	9.35	8.73	5.17	4.44	4.57	3.68	2.88	2.86
ser436	7.47	7.30	8.13	8.63	8.18	7.43	8.13	7.85
ser445	21.62	22.45	25.46	25.02	28.02	31.14	22.54	23.72
ser454	18.90	15.66	16.56	22.49	19.74	22.40	17.90	22.87
ser463	19.69	12.93	23.27	19.94	16.16	17.15	18.35	20.16
ser472	29.34	30.06	14.99	13.00	13.56	5.29	14.61	6.96
ser481	19.31	13.55	35.56	35.46	25.91	18.13	30.32	26.08
ser490	10.20	12.63	10.00	12.77	6.46	4.46	5.17	5.34
ser499	20.85	23.09	20.44	20.39	24.60	24.58	23.78	21.98
ser508	14.03	14.40	15.49	15.56	13.34	13.12	14.45	13.59
ser526	10.96	10.57	9.83	10.86	14.47	14.62	8.86	9.57
ser544	5.43	4.73	5.31	5.45	5.37	3.88	5.50	6.65
ser562	43.50	34.52	28.28	28.89	23.95	20.46	22.14	12.86
ser571	17.19	12.91	12.34	10.84	7.12	5.15	8.31	8.63
ser580	6.04	6.01	6.78	6.90	6.40	5.80	6.71	7.45
ser589	12.67	8.12	13.20	16.72	15.08	16.33	17.86	20.87
ser598	21.46	18.43	23.63	23.87	26.50	23.74	29.67	26.37
ser616	7.71	7.26	7.62	6.12	7.18	4.72	7.36	7.15
ser634	22.53	25.60	24.25	26.71	21.69	21.86	21.58	22.49
ser643	14.71	11.90	17.11	16.04	9.21	4.75	6.13	3.70
ser652	15.88	13.36	23.07	22.34	16.88	3.66	21.01	20.55
ser661	21.53	20.39	6.62	7.70	16.99	18.60	17.85	13.73
ser670	40.83	37.14	42.63	34.80	41.80	31.83	45.25	40.73
ser679	33.99	34.08	20.94	21.43	34.99	34.72	44.92	44.70
ser688	11.35	7.96	10.67	7.42	7.03	4.21	6.98	5.52
ser697	3.35	3.36	2.64	2.55	2.85	3.54	2.94	3.30
ser706	8.74	7.91	9.52	10.61	10.35	11.17	10.77	11.95
ser715	75.10	29.85	80.33	25.31	82.88	7.60	92.38	11.66
ser724	23.72	20.05	14.94	15.02	16.97	17.18	17.05	15.08
ser733	19.25	9.57	20.83	11.96	21.58	7.17	24.00	15.77
ser742	2.78	1.22	2.87	2.50	2.87	1.86	3.04	2.05
ser751	9.27	7.26	8.01	5.58	7.74	4.51	7.30	6.21
ser760	11.25	8.39	8.87	7.87	8.45	7.48	8.24	7.12
ser769	15.40	12.57	14.83	12.50	9.35	6.59	11.17	11.59
ser787	7.89	5.09	7.83	4.56	9.60	6.25	6.29	7.11
ser796	17.94	11.33	19.76	16.83	26.31	23.16	15.90	11.61
ser805	10.00	9.52	11.16	10.95	11.87	11.92	12.48	12.23
ser823	1.06	0.75	1.00	0.65	1.19	0.80	1.35	1.25
ser832	11.08	11.22	11.91	12.19	12.80	12.52	13.78	13.70
ser877	2.87	2.60	4.95	3.18	2.85	1.49	3.74	1.55
ser904	2.87	2.22	2.23	2.47	1.85	0.67	2.22	2.83
ser913	75.36	75.22	73.59	72.58	77.23	76.04	75.36	76.92

TABLE 10 (Continued)

series	y+9		y+10		y+11		y+12	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser922	20.94	20.03	26.60	25.67	32.70	30.20	32.59	34.78
ser958	9.73	8.56	12.99	10.33	14.88	12.56	13.28	12.28
ser967	34.75	26.98	39.39	40.44	36.53	36.94	37.25	36.17
mean	18.18	15.16	17.67	15.86	17.72	14.00	17.88	15.21

TABLE 11

MONTHLY FORECAST OF BPN AT
1_STEP WITH ARCH. 24-24-1

series	y+1	
	mape	m-ape
ser400	8.04	6.76
ser409	28.34	23.78
ser418	23.17	21.02
ser427	8.75	7.15
ser436	2.05	1.81
ser445	16.22	17.16
ser454	10.66	10.00
ser463	20.38	14.94
ser472	12.21	9.37
ser481	44.31	35.26
ser490	10.95	11.87
ser499	9.05	7.96
ser508	6.55	4.47
ser526	12.87	9.23
ser544	4.71	3.16
ser562	64.22	53.03
ser571	6.64	2.89
ser580	3.37	2.83
ser589	8.09	6.08
ser598	10.84	3.75
ser616	5.96	4.50
ser634	22.23	14.73
ser643	18.54	7.28
ser652	15.83	16.58
ser661	20.34	19.34
ser670	22.90	16.91
ser679	37.60	41.17
ser688	11.06	7.87
ser697	3.20	2.98
ser706	10.73	11.45
ser715	57.52	33.95
ser724	18.87	12.13
ser733	17.19	10.00
ser742	2.57	2.28
ser751	11.84	10.77
ser760	7.85	6.86
ser769	7.99	4.47
ser787	4.23	2.27
ser796	16.36	18.59
ser805	1.06	1.08
ser823	0.65	0.41
ser832	0.98	0.88

TABLE 11 (Continued)

series	y+1	
	mape	m-ape
ser877	6.48	6.03
ser904	4.82	2.74
ser913	58.25	57.25
ser922	4.74	4.62
ser949	34.36	35.16
ser958	12.22	10.82
ser967	21.11	16.53
mean	15.69	12.90

TABLE 12

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF BPN
AT 12_STEP WITH ARCH. 24-24-12

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	9.80	8.66	10.82	10.01	10.19	9.75	12.12	10.60
ser409	17.99	14.51	18.47	18.67	16.88	15.20	62.53	50.62
ser418	20.88	18.90	22.58	21.68	18.93	12.58	18.12	10.19
ser427	9.21	8.41	8.52	6.57	7.29	5.91	9.51	7.31
ser436	2.18	1.53	2.61	2.37	2.97	2.57	3.49	3.74
ser445	14.07	14.65	13.01	8.71	26.11	26.22	23.91	22.66
ser454	11.37	7.92	22.71	20.23	22.32	22.31	14.62	11.67
ser463	18.82	15.28	26.05	26.29	24.03	20.09	28.19	30.79
ser472	20.97	16.11	16.14	12.45	17.09	14.83	17.27	7.54
ser481	24.62	15.26	32.55	17.56	19.89	10.70	21.52	10.49
ser490	12.32	10.05	12.34	12.57	13.76	10.72	13.80	10.96
ser499	16.63	16.91	18.15	18.72	16.61	16.53	13.55	14.68
ser508	10.02	9.88	13.91	14.11	10.83	9.22	8.82	9.27
ser526	12.58	9.90	18.86	13.74	19.87	11.72	16.53	10.86
ser544	13.08	15.17	13.27	14.94	12.95	12.93	6.36	6.08
ser562	47.46	28.33	32.34	29.05	53.37	37.00	47.86	59.31
ser571	17.64	16.37	19.08	18.01	14.63	14.27	16.80	15.67
ser598	9.97	6.32	11.00	7.42	14.46	7.07	12.42	6.99
ser616	6.76	5.01	7.11	5.75	6.52	5.22	7.25	4.06
ser634	18.17	15.89	16.49	11.86	16.13	14.30	17.25	12.06
ser643	18.74	11.41	20.47	15.17	19.53	13.78	17.64	15.16
ser652	18.69	16.38	12.61	12.15	15.54	14.15	19.02	21.60
ser661	29.70	27.13	31.31	29.06	27.44	20.62	23.51	21.28
ser670	33.87	33.28	35.02	35.66	35.95	36.17	37.93	34.92
ser679	78.58	61.68	61.03	52.11	50.85	42.80	44.37	43.19
ser688	17.41	7.93	24.26	14.93	22.00	9.16	22.68	7.73
ser697	6.04	6.18	7.32	6.94	6.64	7.69	6.42	6.76
ser706	3.35	1.63	4.71	4.15	2.52	1.78	3.03	2.19
ser715	53.96	11.25	59.97	39.32	67.71	18.91	58.20	16.04
ser724	22.03	18.67	22.49	18.77	22.41	13.77	25.29	21.16
ser733	16.94	8.73	17.18	12.40	16.37	6.10	15.55	7.20
ser742	4.52	4.16	4.88	4.76	4.08	3.44	4.15	4.36
ser751	6.03	5.31	6.90	6.47	8.69	8.08	7.15	6.86
ser760	8.37	7.10	8.00	6.66	8.42	6.61	7.26	6.90
ser769	11.29	6.98	11.75	11.40	14.42	13.02	10.94	8.71
ser787	6.09	3.87	5.23	3.70	7.22	3.62	7.47	5.17
ser796	12.93	7.38	12.81	7.76	12.02	7.10	12.09	8.51
ser805	3.88	4.25	4.56	4.72	7.16	7.51	7.20	7.07
ser823	1.37	1.48	1.26	1.33	1.31	1.28	1.03	1.06
ser832	5.74	5.19	6.06	5.40	6.92	6.24	7.70	7.02
ser877	4.74	4.17	5.40	4.30	4.49	3.10	4.72	3.56

TABLE 12 (Continued)

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser904	4.85	2.73	4.75	3.25	4.72	2.01	4.48	2.37
ser922	6.60	5.44	10.74	8.93	19.09	17.44	12.36	12.18
mean	16.05	12.03	16.62	13.72	16.98	12.64	17.03	13.64

TABLE 12 (Continued)

series	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	13.86	11.38	11.89	9.40	10.67	9.34	8.48	9.30
ser409	20.48	17.54	16.52	13.22	18.12	11.72	26.30	18.09
ser418	12.63	8.43	17.07	16.40	21.29	19.19	21.05	23.65
ser427	7.33	4.06	8.91	7.83	7.01	5.46	7.90	7.71
ser436	4.11	4.13	5.03	4.99	5.51	5.02	5.54	5.73
ser445	18.90	19.60	21.83	21.97	27.52	29.68	21.29	22.62
ser454	17.69	14.18	27.46	29.22	20.31	20.37	17.63	17.60
ser463	24.71	15.27	21.07	17.19	37.94	18.33	31.09	15.46
ser472	21.67	12.23	23.24	21.82	16.19	17.14	22.63	20.33
ser481	20.22	10.64	23.43	11.13	17.57	11.28	19.77	17.52
ser490	12.72	11.45	11.41	9.72	10.61	9.62	9.40	9.16
ser499	18.42	17.99	18.50	22.17	19.56	21.06	24.59	29.00
ser508	11.17	9.45	12.23	12.56	12.09	11.12	16.64	15.62
ser526	12.92	8.51	16.98	12.10	15.49	10.18	9.82	8.15
ser544	4.96	3.76	5.30	4.80	4.88	4.23	4.80	3.76
ser562	72.76	66.24	91.05	84.81	69.71	66.76	44.03	29.40
ser571	19.31	16.53	17.40	14.74	19.53	17.29	19.65	15.16
ser598	12.49	4.63	14.50	8.99	19.16	11.22	18.30	16.50
ser616	7.88	7.81	7.49	6.21	7.08	6.33	7.84	8.16
ser634	16.04	10.81	21.27	18.65	20.10	18.20	15.67	14.71
ser643	12.59	7.97	23.89	22.49	19.22	18.88	14.82	12.90
ser652	16.88	15.13	29.58	31.50	23.95	20.08	23.08	20.33
ser661	20.15	18.67	16.81	13.95	15.09	10.36	16.77	13.50
ser670	35.59	34.60	35.50	34.33	38.74	37.77	36.57	35.51
ser679	36.32	32.80	28.72	33.64	19.07	21.98	15.77	15.48
ser688	17.81	5.76	20.12	12.20	22.93	5.58	21.80	8.22
ser697	6.58	7.07	6.44	7.55	5.61	5.93	4.99	5.74
ser706	4.07	3.00	6.45	7.68	6.99	6.56	7.55	8.03
ser715	43.38	27.35	80.35	85.45	69.68	81.68	71.77	75.13
ser724	23.80	16.70	23.83	22.27	23.13	14.80	23.31	24.02
ser733	14.56	7.08	17.54	12.27	18.76	5.64	18.75	9.10
ser742	3.56	3.29	4.49	4.51	3.08	2.54	3.57	3.37
ser751	10.78	10.46	10.57	8.68	9.76	7.25	8.55	8.68
ser760	8.43	6.47	9.26	8.08	9.44	8.18	8.95	8.71
ser769	10.29	7.77	7.45	6.60	10.31	7.95	9.45	8.46
ser787	7.84	5.81	7.94	6.61	7.95	6.16	5.55	4.40
ser796	13.33	6.27	12.09	8.89	16.20	11.65	11.68	7.22
ser805	7.53	7.79	8.18	7.74	9.31	9.40	9.97	9.64
ser823	0.84	0.92	0.80	0.74	0.66	0.39	0.69	0.66
ser832	8.55	7.66	8.90	8.61	9.83	9.48	10.50	10.55
ser877	6.64	6.81	7.55	7.20	2.91	2.70	4.55	3.72
ser904	4.19	1.71	4.39	1.66	3.77	1.15	5.04	3.12
ser922	12.40	11.80	18.50	18.76	15.95	15.48	21.00	20.71
mean	15.73	12.27	18.18	16.73	17.28	14.77	16.44	14.53

TABLE 12 (Continued)

series	y+9		y+10		y+11		y+12	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	8.86	6.38	8.88	8.76	11.35	9.13	15.36	15.95
ser409	24.76	13.57	34.06	28.80	24.82	14.32	28.02	27.05
ser418	18.75	15.05	19.71	18.81	20.82	15.52	19.53	13.31
ser427	4.18	1.59	7.96	8.62	5.96	3.53	2.90	2.84
ser436	6.26	6.00	6.80	6.69	7.27	7.00	7.42	7.34
ser445	56.13	58.60	33.99	35.09	29.88	29.27	40.44	43.14
ser454	20.16	17.03	19.73	20.89	17.57	20.58	17.45	22.07
ser463	32.74	11.45	40.84	20.96	14.77	9.37	30.22	32.96
ser472	9.78	7.27	24.19	17.93	25.54	17.19	23.03	18.05
ser481	20.39	11.33	28.68	21.19	33.26	28.68	43.73	43.92
ser490	10.29	5.59	9.51	9.80	5.98	3.86	4.88	3.66
ser499	21.35	24.21	24.97	26.85	22.57	22.77	23.00	21.17
ser508	15.52	16.37	14.43	15.98	13.52	13.46	14.19	13.50
ser526	9.37	6.87	16.25	13.38	13.07	8.22	12.86	11.59
ser544	5.74	5.63	6.12	4.68	5.62	4.26	5.53	6.60
ser562	23.92	14.88	44.12	33.13	62.93	64.94	65.14	71.27
ser571	16.32	12.11	10.54	9.82	8.61	6.00	8.27	8.14
ser598	17.71	15.62	14.59	10.73	26.19	23.17	28.50	22.80
ser616	7.49	5.94	7.39	6.20	7.16	4.73	7.24	6.03
ser634	21.23	15.86	16.55	15.42	15.81	17.52	18.59	17.93
ser643	13.19	8.46	7.88	6.93	7.11	6.56	3.90	3.18
ser652	38.81	45.03	28.90	28.56	45.56	44.77	31.56	25.15
ser661	16.22	12.87	16.08	12.68	16.75	16.67	25.46	27.00
ser670	39.92	38.38	48.47	47.81	41.43	38.66	43.42	36.56
ser679	9.56	5.00	41.09	41.28	50.07	45.67	56.72	58.24
ser688	15.58	10.55	9.24	6.51	6.66	4.09	7.82	7.90
ser697	4.57	4.43	3.61	3.63	2.74	1.72	4.43	5.15
ser706	7.01	7.15	7.87	9.27	9.70	10.59	9.43	10.54
ser715	80.91	89.14	75.41	82.42	88.20	38.17	86.51	26.55
ser724	21.02	18.88	14.21	9.49	16.12	13.76	16.57	15.76
ser733	17.36	6.78	19.24	12.52	23.80	9.34	24.53	18.89
ser742	3.68	2.45	2.74	0.75	2.79	2.01	3.20	2.63
ser751	8.04	6.12	10.75	9.25	6.16	3.12	6.57	6.68
ser760	7.81	4.72	6.81	4.57	6.43	5.53	7.35	6.66
ser769	11.87	9.25	19.89	18.60	10.88	7.17	16.36	15.13
ser787	8.64	6.26	13.86	14.31	9.35	6.39	10.44	10.21
ser796	13.62	9.01	11.86	8.97	15.28	6.90	18.07	11.67
ser805	10.74	10.23	11.46	11.23	11.97	12.06	12.16	11.89
ser823	0.83	0.52	1.10	0.90	0.96	0.67	1.06	1.04
ser832	11.22	11.36	12.04	12.31	12.84	12.56	13.80	13.72
ser877	7.61	6.46	4.43	4.21	6.12	6.19	8.23	6.78
ser904	10.12	9.76	2.21	2.28	2.26	1.79	2.24	2.96
ser922	22.37	21.65	13.71	13.12	20.39	18.16	22.35	21.89
mean	17.02	14.32	17.96	16.17	18.35	14.79	19.73	17.80

APPENDIX D

TEST RESULTS OF CPN WITH YEARLY DATA

TABLE 13

YEARLY FORECAST OF CPN AT
1_STEP WITH ARCH. 3-6-3

series	y+1	
	mape	m-ape
ser4	15.49	13.24
ser13	11.95	8.91
ser22	0.23	0.14
ser31	9.07	6.36
ser40	4.12	3.43
ser49	29.38	29.97
ser58	6.11	2.90
ser67	3.47	2.42
ser76	6.22	5.38
ser85	3.05	1.55
ser94	6.87	7.83
ser103	1.41	0.99
ser112	1.08	1.03
ser121	2.02	1.69
ser130	12.04	11.40
ser139	8.93	8.09
ser148	6.27	5.77
ser157	9.30	3.54
ser175	3.71	1.88
mean	7.41	6.13

TABLE 14

MAPE AND M-APE VALUES OF YEARLY FORECAST OF CPN
AT 4_STEP WITH ARCH. 6-12-6

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser4	35.52	29.39	53.09	47.92	63.76	53.10	67.84	56.06
ser13	42.06	38.90	45.42	46.65	61.18	59.82	43.42	40.43
ser22	0.32	0.29	0.33	0.28	0.25	0.20	0.31	0.22
ser31	9.04	5.00	13.48	12.69	16.98	16.33	11.50	12.63
ser40	4.16	3.51	5.07	5.03	2.16	0.85	4.89	5.20
ser49	25.66	21.15	34.27	32.62	31.28	24.77	35.32	35.69
ser58	10.47	6.21	11.99	9.08	14.67	5.77	17.87	17.53
ser85	3.88	1.68	12.91	14.35	19.76	16.26	26.53	25.85
ser112	0.97	0.86	1.32	1.22	1.58	1.16	1.97	2.10
ser121	0.98	0.52	1.89	1.84	3.26	3.08	5.76	5.87
ser157	10.74	5.97	13.63	12.84	11.41	7.76	9.86	8.82
mean	13.07	10.32	17.58	16.77	20.57	17.19	20.48	19.13

APPENDIX E

TEST RESULTS OF CPN WITH QUARTERLY DATA

TABLE 15
 QUARTERLY FORECAST OF CPN AT
 1_STEP WITH ARCH. 5-10-5

series	y+1	
	mape	m-ape
ser184	18.52	13.51
ser193	22.62	21.39
ser202	3.04	2.10
ser211	17.04	15.35
ser220	30.17	16.88
ser229	14.45	8.63
ser238	5.29	3.19
ser265	2.59	2.75
ser283	2.40	1.14
ser292	21.53	8.87
ser301	1.66	1.12
ser310	0.87	0.76
ser319	3.62	3.29
ser328	3.63	2.47
ser337	10.12	11.40
ser346	16.12	7.57
ser355	2.55	1.83
ser364	1.40	0.72
ser382	20.53	23.08
mean	10.43	7.69

TABLE 16

MAPE AND M-APE VALUES OF QUARTERLY FORECAST OF CPN
AT 4_STEP WITH ARCH. 8-16-8

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser184	16.59	19.09	17.60	11.18	25.68	18.73	10.72	12.68
ser193	24.74	24.88	35.55	14.41	44.83	26.18	65.07	42.67
ser202	2.97	1.96	5.88	4.35	7.55	3.34	7.00	2.38
ser211	35.20	34.94	36.37	25.66	38.94	25.78	24.66	28.28
ser220	27.41	8.72	38.97	26.71	25.44	3.83	35.89	24.92
ser229	16.20	12.68	13.55	10.84	16.60	14.72	6.34	6.85
ser238	4.29	1.60	7.81	6.16	11.04	3.74	9.41	12.13
ser265	2.96	1.67	2.88	2.12	2.25	0.74	3.64	2.72
ser283	2.20	0.93	4.08	4.07	6.38	4.70	8.39	8.75
ser292	18.80	9.21	11.19	6.09	19.59	12.86	8.28	2.70
ser301	1.82	1.33	2.89	3.23	5.22	4.65	5.68	6.03
ser310	1.35	1.24	2.77	2.69	3.19	3.02	4.38	4.31
ser319	2.96	2.49	5.03	4.80	8.24	9.72	9.76	11.52
ser328	3.31	2.91	2.71	1.10	3.89	1.93	5.39	5.93
ser337	8.79	6.70	2.89	1.62	9.60	7.96	3.72	2.27
ser346	11.71	7.57	16.49	11.15	18.59	16.75	23.52	12.95
ser355	5.43	5.82	8.69	9.19	7.47	7.28	5.75	7.55
ser364	1.32	0.61	2.28	2.00	4.37	2.69	6.16	6.08
ser382	23.33	20.80	42.63	36.47	43.10	47.29	42.90	54.32
mean	11.13	8.69	13.70	9.68	15.89	11.36	15.09	13.42

TABLE 17
 QUARTERLY FORECAST OF CPN AT
 1_STEP WITH ARCH.9-18-9

series	y+1	
	mape	m-ape
ser184	15.70	10.82
ser193	24.56	23.77
ser202	3.16	1.95
ser211	16.62	7.98
ser220	28.14	16.72
ser229	15.36	11.34
ser238	5.11	2.45
ser265	3.38	3.28
ser283	2.06	0.79
ser292	14.89	10.73
ser301	1.42	0.84
ser310	1.45	1.34
ser319	3.94	4.24
ser328	3.31	3.56
ser337	8.68	10.01
ser346	12.72	7.64
ser355	4.57	4.76
ser364	1.27	0.52
ser382	20.23	16.33
mean	9.82	7.32

TABLE 18

MAPE AND M-APE VALUES OF QUARTERLY FORECAST OF CPN
AT 4_STEP WITH ARCH. 12-24-12

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser184	23.04	16.24	14.93	10.18	19.92	16.23	20.15	21.92
ser202	2.94	2.38	5.11	3.32	6.40	4.39	6.99	2.36
ser211	18.13	13.82	24.98	25.74	31.00	23.07	25.66	32.03
ser220	28.15	13.12	36.83	29.43	26.39	5.73	35.70	26.67
ser229	13.28	3.48	11.65	1.96	20.13	10.88	1.43	1.69
ser238	5.12	2.75	6.12	5.66	9.08	5.55	10.03	11.91
ser265	3.52	3.08	6.77	6.63	9.72	8.79	12.19	12.78
ser283	2.66	1.40	5.02	5.01	7.36	5.69	10.03	10.38
ser292	15.67	10.41	7.84	6.85	22.36	15.39	7.81	7.09
ser301	2.10	1.71	3.25	3.73	6.03	5.46	7.90	8.25
ser310	1.18	1.08	2.36	2.28	3.99	3.82	4.53	4.47
ser319	3.35	2.73	4.72	4.86	6.83	7.55	7.31	8.87
ser328	3.29	3.36	3.68	2.42	4.95	3.95	7.90	10.00
ser337	10.50	8.17	4.35	3.44	11.26	9.98	4.67	3.37
ser346	17.14	8.19	26.09	10.22	46.72	27.71	72.14	53.80
ser355	4.70	5.51	8.06	8.56	7.61	7.42	4.89	5.78
ser364	1.16	0.63	2.34	2.07	3.74	2.07	5.79	5.70
ser382	18.65	11.34	34.32	24.40	40.54	41.95	39.94	45.69
mean	9.70	6.08	11.58	8.71	15.78	11.42	15.84	15.15

APPENDIX F

TEST RESULTS OF CPN WITH MONTHLY DATA

TABLE 19

MONTHLY FORECAST OF CPN AT
1_STEP WITH ARCH. 13-26-13

series	y+1	
	mape	m-ape
ser391	20.47	12.28
ser400	7.30	5.97
ser409	30.96	24.78
ser418	22.76	16.53
ser427	11.14	10.87
ser436	1.76	1.45
ser445	17.29	10.97
ser454	24.17	21.47
ser463	21.21	17.22
ser472	22.08	8.34
ser481	33.15	28.53
ser490	17.36	14.66
ser499	9.86	7.02
ser508	9.06	5.83
ser526	15.09	9.21
ser535	26.71	25.27
ser544	2.26	2.34
ser562	24.86	23.01
ser571	12.85	8.60
ser580	0.71	0.68
ser589	5.55	3.50
ser598	6.37	5.64
ser616	10.76	5.53
ser625	24.39	12.11
ser634	17.31	12.05
ser643	16.74	13.96
ser652	30.00	28.71
ser661	21.63	10.31
ser670	31.77	21.03
ser679	19.27	16.90
ser688	13.24	8.68
ser697	2.83	2.77
ser706	3.36	2.78
ser715	66.49	45.41
ser724	16.50	9.28
ser733	27.81	15.63
ser742	5.76	5.70
ser751	9.78	6.79
ser760	19.19	12.61
ser769	12.71	10.79
ser778	2.25	2.10
ser787	8.22	4.85

TABLE 19 (Continued)

series	y+1	
	mape	m-ape
ser796	40.13	22.17
ser805	1.09	1.01
ser814	0.65	0.49
ser823	0.50	0.33
ser832	0.96	0.85
ser841	19.74	18.49
ser850	15.57	8.06
ser868	10.50	7.06
ser877	10.35	7.12
ser886	9.48	8.33
ser895	4.27	3.60
ser904	5.00	1.71
ser913	25.87	15.70
ser922	4.23	3.61
ser931	47.01	43.67
ser940	26.74	16.61
ser949	21.53	13.59
ser958	6.27	2.48
ser967	6.53	4.42
ser985	52.25	21.26
ser994	19.49	14.61
mean	16.37	11.55

TABLE 20

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF CPN
AT 12_STEP WITH ARCH. 24-48-24

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	8.24	5.99	11.62	10.97	12.73	13.95	13.04	14.90
ser409	20.31	17.46	46.97	34.85	34.69	32.79	66.32	50.40
ser418	25.15	19.64	21.59	20.22	21.84	19.60	19.74	14.88
ser427	10.50	8.43	14.18	11.49	16.75	9.33	22.24	15.21
ser436	1.79	1.60	2.11	1.83	3.46	3.23	5.00	4.76
ser445	16.87	10.75	21.79	19.62	17.63	9.91	19.53	13.71
ser454	22.64	15.48	34.30	16.72	35.46	18.16	39.83	20.96
ser463	26.21	27.12	22.93	19.84	23.99	20.04	31.56	29.69
ser472	22.76	11.54	25.70	8.39	28.76	13.33	30.75	11.08
ser481	37.50	32.01	33.46	22.03	32.01	22.75	43.55	26.38
ser490	12.64	9.99	19.05	14.65	10.76	7.52	14.60	13.51
ser499	10.86	6.96	11.68	8.99	8.54	6.99	11.91	11.96
ser508	8.75	7.17	9.22	8.02	7.97	6.13	9.54	6.56
ser526	12.78	6.86	13.43	5.20	13.33	9.17	14.44	7.37
ser535	21.61	13.91	33.85	32.51	39.94	39.62	42.30	39.59
ser544	2.27	2.43	3.64	3.21	3.33	2.48	3.61	2.91
ser562	28.34	22.10	25.03	26.55	25.43	25.87	24.47	19.71
ser571	13.43	8.56	17.83	7.69	18.46	4.21	20.99	9.92
ser580	0.63	0.57	1.32	1.29	2.52	2.31	4.42	4.48
ser589	5.96	3.88	11.06	8.58	14.10	11.45	14.60	13.43
ser598	6.62	4.65	8.48	7.36	7.65	4.22	11.18	12.74
ser616	11.36	9.16	12.40	9.86	10.49	6.63	9.40	4.78
ser634	19.03	13.65	18.75	17.68	15.15	14.71	19.85	21.08
ser643	16.72	13.86	21.54	18.70	24.32	15.73	20.97	19.26
ser652	24.11	19.95	26.61	20.57	24.25	12.12	25.92	9.98
ser661	24.94	9.35	39.01	20.68	41.75	13.47	50.57	22.87
ser670	26.56	11.20	34.84	11.25	33.52	13.15	34.20	20.59
ser679	14.72	13.30	20.79	19.99	24.88	25.93	30.43	26.68
ser688	14.72	10.48	25.54	11.82	35.18	9.72	32.93	16.37
ser697	3.93	3.64	3.94	3.09	3.40	2.18	4.57	4.30
ser706	6.66	5.62	5.55	4.37	6.15	6.00	5.44	4.54
ser715	58.45	6.44	71.07	22.05	62.36	22.41	56.59	25.97
ser724	18.48	12.70	18.67	10.19	18.08	17.71	18.96	12.17
ser733	26.56	15.56	27.56	17.14	22.96	8.60	29.19	18.18
ser742	5.18	4.33	8.11	7.92	11.88	12.52	13.56	13.78
ser751	9.71	6.71	10.63	9.11	11.58	8.84	14.34	15.93
ser760	20.27	17.23	22.27	24.21	12.28	5.55	23.24	21.94
ser769	13.07	9.55	18.27	15.92	19.64	16.55	18.08	20.94
ser787	6.56	2.67	7.96	2.23	8.77	3.57	11.28	7.72
ser796	42.98	20.14	47.97	30.11	41.63	38.92	36.09	40.62
ser805	0.99	0.81	1.51	1.29	2.11	1.89	2.65	2.60
ser823	0.58	0.52	0.74	0.57	0.84	0.75	1.05	0.94

TABLE 20 (Continue)

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser832	0.84	0.73	1.65	1.52	2.37	2.02	3.17	2.94
ser877	9.46	4.34	13.24	10.82	13.86	14.03	12.62	12.27
ser904	5.45	2.07	6.20	2.68	7.16	2.67	7.76	4.07
ser913	24.34	17.45	35.45	39.48	39.71	35.76	42.44	40.31
ser922	4.99	5.22	10.63	10.65	14.76	13.46	19.46	20.23
ser949	18.28	11.21	18.19	18.43	24.28	19.41	25.64	23.19
ser958	10.23	8.48	15.02	17.14	13.51	7.61	10.19	7.10
ser967	5.95	4.79	9.23	8.23	10.90	10.23	12.93	12.27
mean	15.22	9.98	18.85	13.55	18.74	12.91	21.14	15.96

TABLE 20 (Continued)

series	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	11.94	8.60	14.18	11.89	14.76	13.03	13.10	13.76
ser409	62.41	54.00	53.94	42.98	57.88	50.21	59.89	54.88
ser418	19.14	14.33	17.42	22.16	20.02	18.64	18.03	17.88
ser427	25.96	16.03	27.52	20.75	27.51	21.55	25.80	19.12
ser436	6.49	7.18	7.87	7.64	8.47	8.25	8.86	9.13
ser445	19.27	11.04	17.06	8.47	18.30	11.88	14.21	12.83
ser454	31.61	20.54	28.51	13.67	35.57	19.47	47.78	31.52
ser463	33.22	24.33	29.30	22.70	30.73	26.64	32.05	29.80
ser472	31.26	13.24	28.55	12.22	26.81	17.54	28.92	23.46
ser481	41.05	32.27	32.70	24.34	31.03	25.29	35.13	33.84
ser490	19.35	12.98	13.68	12.57	16.33	14.25	13.09	6.94
ser499	9.98	7.05	11.41	14.40	14.20	12.92	9.97	7.47
ser508	7.43	2.26	6.99	6.93	8.94	6.96	6.67	1.56
ser526	16.28	8.70	13.49	6.11	15.29	8.07	16.25	14.97
ser535	49.03	36.56	54.85	31.86	49.12	27.67	41.76	35.12
ser544	4.77	3.64	5.69	4.03	5.62	4.73	6.23	8.46
ser562	22.44	21.96	31.61	32.35	21.31	14.89	24.77	21.87
ser571	21.00	8.76	22.83	8.43	27.67	15.49	26.79	13.52
ser580	4.58	4.55	4.53	4.46	5.25	4.58	6.49	6.68
ser589	14.29	8.56	14.91	9.06	15.91	9.33	15.65	7.55
ser598	9.76	8.46	17.50	11.94	17.19	14.01	24.34	24.09
ser616	9.63	6.56	12.23	13.39	12.89	8.83	11.70	7.73
ser634	21.39	21.62	20.67	19.97	19.68	14.46	22.75	19.47
ser643	19.81	20.05	16.97	17.60	16.26	12.51	19.01	15.64
ser652	29.92	23.12	34.08	32.48	30.16	27.96	32.46	27.00
ser661	41.11	29.45	37.50	33.75	38.54	25.80	29.96	34.28
ser670	25.48	16.25	34.21	28.06	31.03	23.59	36.09	21.64
ser679	32.12	30.33	30.48	34.04	27.36	11.66	25.00	14.93
ser688	26.63	12.89	32.14	20.72	32.80	21.90	33.25	24.65
ser697	4.61	4.14	5.88	5.22	5.64	5.63	6.11	6.71
ser706	8.67	8.55	11.66	11.20	11.21	9.45	18.14	20.02
ser715	54.98	29.61	59.44	42.27	81.73	52.12	73.75	47.34
ser724	20.05	19.55	22.39	19.04	20.90	16.14	26.02	23.98
ser733	32.49	18.35	29.37	26.36	26.64	20.87	24.88	15.71
ser742	13.76	14.16	11.44	8.43	11.23	8.48	11.03	9.36
ser751	14.04	12.13	17.20	14.82	14.50	7.79	16.87	13.97
ser760	23.55	23.56	13.06	11.03	22.68	23.19	24.40	17.66
ser769	15.00	15.04	15.29	12.53	21.39	25.26	20.45	20.08
ser787	7.85	2.25	12.81	11.03	9.50	3.38	8.04	4.61
ser796	33.32	14.35	25.71	29.68	34.44	23.53	31.64	35.14
ser805	3.02	3.47	3.96	3.60	4.75	4.97	5.93	5.63
ser823	1.42	1.15	1.57	1.54	1.48	0.83	1.30	1.17
ser832	4.02	3.31	4.83	4.50	5.52	5.17	6.29	6.31

TABLE 20 (Continued)

series	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser877	8.36	4.99	6.55	3.86	7.73	5.93	14.24	14.30
ser904	9.04	5.19	8.06	4.57	8.99	4.21	10.24	5.72
ser913	45.73	42.88	50.27	53.74	63.61	65.22	70.77	79.03
ser922	27.49	28.19	31.04	29.51	33.15	32.29	33.11	33.57
ser949	26.54	23.34	30.73	27.44	33.05	35.99	44.13	44.28
ser958	7.46	7.19	14.66	15.68	13.17	13.26	11.27	10.66
ser967	13.78	11.08	14.79	11.37	16.61	17.96	18.70	20.04
mean	20.85	15.56	21.27	17.53	22.49	17.48	23.27	19.90

TABLE 20 (Continued)

series	y+9		y+10		y+11		y+12	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	10.37	7.50	8.09	4.65	8.07	6.38	8.96	7.05
ser409	62.51	47.63	59.36	53.42	40.61	26.66	17.96	11.81
ser418	17.09	9.25	24.91	24.48	27.02	21.28	15.33	11.08
ser427	22.04	13.20	18.47	19.26	14.85	12.26	4.71	3.69
ser436	9.68	9.11	10.22	10.26	10.75	10.85	10.70	10.85
ser445	14.38	12.40	24.66	19.04	16.91	16.43	24.71	25.17
ser454	32.31	19.70	27.48	23.47	14.75	13.59	10.78	9.33
ser463	28.18	25.76	39.79	39.06	33.22	32.58	31.77	34.92
ser472	26.96	21.90	38.04	33.97	42.42	41.45	32.05	30.07
ser481	31.83	15.21	25.82	25.06	51.70	48.48	42.93	34.90
ser490	12.98	14.02	16.99	12.32	20.42	13.80	15.48	16.34
ser499	12.62	10.92	13.69	12.02	10.01	5.59	7.43	6.38
ser508	10.01	8.56	10.30	4.43	9.37	5.86	3.51	2.37
ser526	15.36	13.06	16.90	15.10	20.83	16.37	9.32	8.36
ser535	35.08	27.81	46.55	39.63	43.89	34.22	39.56	34.97
ser544	8.59	8.30	8.11	8.00	8.32	7.86	9.20	9.03
ser562	27.19	25.90	19.98	10.26	30.60	27.85	28.21	35.25
ser571	20.90	6.02	17.50	14.66	8.96	9.79	3.79	1.69
ser580	6.83	7.17	7.51	8.14	8.24	7.90	9.11	9.83
ser589	14.38	8.28	13.12	5.53	14.13	7.35	16.94	17.55
ser598	24.44	19.92	23.43	24.15	34.39	34.97	31.54	28.77
ser616	10.21	11.04	9.72	7.73	11.64	6.55	3.07	2.60
ser634	16.51	14.59	24.56	32.24	32.56	33.06	28.39	25.14
ser643	21.19	22.28	15.62	19.37	17.05	15.38	15.48	13.41
ser652	34.20	29.24	27.94	18.26	19.20	6.76	12.38	9.45
ser661	34.42	30.03	31.06	29.69	19.17	12.52	10.01	7.99
ser670	43.97	27.75	42.75	33.06	39.86	38.58	28.80	30.79
ser679	27.69	18.79	33.19	34.15	35.17	34.60	42.96	43.74
ser688	19.31	11.96	17.24	17.24	12.90	9.98	12.66	11.97
ser697	5.31	3.55	5.20	4.96	5.28	5.02	3.71	4.20
ser706	21.91	22.27	25.19	26.76	23.69	23.99	23.64	26.79
ser715	60.46	60.11	61.96	64.79	78.13	49.80	32.26	34.64
ser724	27.70	25.91	23.34	19.50	22.84	17.14	23.26	25.24
ser733	36.15	24.20	25.57	17.12	24.22	20.03	24.74	25.84
ser742	10.34	10.09	6.83	7.21	3.69	2.52	2.22	0.86
ser751	13.80	10.27	10.69	9.01	8.90	7.90	7.82	6.00
ser760	11.12	7.52	23.49	13.75	23.68	16.21	9.54	9.27
ser769	20.81	19.64	19.00	17.91	14.19	12.24	9.38	6.50
ser787	8.90	2.98	8.85	5.12	7.84	2.24	1.65	1.09
ser796	41.04	25.71	57.93	22.88	43.74	26.36	11.03	8.50
ser805	6.86	6.40	8.01	7.79	9.27	9.47	10.13	9.83
ser823	1.34	0.74	1.71	0.84	1.61	0.55	1.12	0.38
ser832	7.11	7.30	7.98	8.29	8.87	8.58	9.80	9.71

TABLE 20 (Continued)

series	y+9		y+10		y+11		y+12	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser877	14.54	14.46	13.30	6.85	9.99	4.57	4.94	5.05
ser904	7.61	4.43	8.09	6.11	5.41	3.77	6.31	6.83
ser913	76.30	81.02	76.37	80.64	78.64	79.55	80.77	80.28
ser922	32.01	32.48	33.00	31.76	33.68	31.90	30.47	30.89
ser949	47.34	50.33	58.39	58.95	66.13	68.17	73.98	74.05
ser958	13.35	14.71	12.71	9.66	16.68	10.84	14.49	15.21
ser967	21.85	18.82	23.94	29.66	27.00	25.06	29.96	30.34
mean	22.74	18.80	23.69	20.95	23.41	19.70	18.78	18.12

TABLE 21

MONTHLY FORECAST OF CPN AT
1_STEP WITH ARCH. 25-50-25

series	y+1	
	mape	m-ape
ser400	8.24	5.09
ser409	31.56	22.58
ser418	20.57	15.05
ser427	9.23	9.78
ser436	1.79	1.99
ser445	15.97	13.00
ser454	21.36	17.81
ser463	18.62	17.24
ser472	21.67	10.26
ser481	36.54	33.00
ser490	16.75	13.41
ser499	13.59	10.92
ser508	8.91	7.12
ser526	14.16	9.53
ser535	17.30	7.73
ser544	2.25	2.41
ser562	29.49	15.54
ser571	10.35	6.73
ser580	0.54	0.45
ser589	3.21	1.22
ser598	5.97	5.63
ser616	11.32	9.09
ser634	23.73	20.73
ser643	16.16	11.95
ser652	22.15	16.05
ser661	21.24	12.38
ser670	26.44	6.57
ser679	16.92	13.28
ser688	14.24	11.20
ser697	3.44	3.03
ser706	3.54	2.36
ser715	59.37	17.69
ser724	18.69	12.43
ser733	26.29	19.10
ser742	4.30	4.07
ser751	9.01	4.42
ser760	17.96	14.36
ser769	13.66	10.12
ser787	9.96	8.70
ser796	25.41	14.80
ser805	1.38	1.35
ser823	0.64	0.60

TABLE 21 (Continued)

series	y+1	
	mape	m-ape
ser832	0.84	0.73
ser877	9.12	7.52
ser904	5.00	1.54
ser913	22.62	18.59
ser922	3.89	3.69
ser949	32.70	30.19
ser958	8.11	6.97
ser967	9.46	8.97
mean	14.91	10.38

TABLE 22

MAPE AND M-APE VALUES OF MONTHLY FORECAST OF CPN
AT 12_STEP WITH ARCH. 36-72-36

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	7.91	4.69	11.55	11.32	12.73	14.05	12.87	16.37
ser409	24.03	20.55	40.58	42.41	49.42	34.88	30.01	25.46
ser418	23.88	17.94	21.62	23.12	21.77	20.44	23.83	18.72
ser427	9.87	7.76	11.55	6.50	15.42	11.75	17.39	13.87
ser436	1.77	1.31	1.83	1.28	1.68	1.55	2.30	2.05
ser445	18.98	9.04	24.36	17.94	18.69	15.32	24.69	16.39
ser454	22.02	15.38	28.88	13.60	27.71	15.64	33.94	22.29
ser463	16.15	6.46	21.55	14.54	23.35	18.72	26.93	22.28
ser472	21.47	9.99	24.79	9.27	27.42	14.65	29.61	13.59
ser481	35.92	31.26	44.79	35.17	22.04	24.09	37.52	23.84
ser490	17.46	15.46	17.92	13.27	9.98	6.74	13.27	12.85
ser499	10.29	8.38	11.74	8.27	8.00	4.55	12.60	12.99
ser508	8.60	7.78	8.23	3.67	7.87	4.56	10.32	8.95
ser526	14.30	10.05	13.47	6.77	13.29	9.74	13.90	9.53
ser544	4.96	5.05	4.49	3.98	3.15	2.38	2.86	3.92
ser562	29.16	23.04	29.07	25.77	21.99	11.59	23.47	18.80
ser571	9.12	2.26	17.07	4.72	18.29	3.51	22.47	12.69
ser598	5.74	4.11	7.04	5.80	8.00	5.09	10.22	10.98
ser616	11.42	9.12	12.86	11.02	15.07	11.32	17.89	15.46
ser634	19.04	14.64	19.99	14.11	17.83	14.50	24.24	21.44
ser643	18.58	14.49	24.31	23.46	29.67	27.15	21.78	17.37
ser652	25.15	16.47	26.07	21.91	29.41	21.37	29.02	14.14
ser661	20.54	7.41	28.68	11.35	36.02	15.11	32.90	12.75
ser670	29.81	18.22	36.38	12.53	34.13	13.05	27.00	11.16
ser679	15.79	11.40	26.16	26.20	29.45	28.13	33.68	34.29
ser688	16.16	9.66	24.00	10.58	27.22	7.94	28.54	13.93
ser697	8.33	7.81	7.75	9.35	6.60	6.38	9.52	9.79
ser706	7.73	7.07	5.76	5.53	10.70	9.90	13.54	13.13
ser715	60.26	24.36	62.48	39.48	66.46	49.49	67.68	64.68
ser724	18.50	19.97	16.72	12.75	20.41	19.02	17.59	17.29
ser733	27.56	19.78	29.60	23.83	25.61	23.77	27.85	20.13
ser742	4.86	3.71	7.34	6.94	10.99	11.59	13.85	14.71
ser751	9.82	7.32	10.74	10.57	11.80	7.23	14.64	12.81
ser760	10.17	5.55	11.67	9.58	12.12	9.64	15.41	13.91
ser769	12.68	11.45	18.99	17.66	20.76	17.83	18.07	20.45
ser787	6.82	3.80	9.35	5.01	15.05	11.39	13.55	10.50
ser796	36.82	23.15	30.53	26.49	34.52	34.01	31.72	38.74
ser805	1.43	1.40	2.39	2.25	3.18	2.99	4.21	4.16
ser823	0.63	0.57	0.85	0.87	0.84	0.72	1.07	0.99
ser832	0.64	0.53	1.60	1.48	1.61	1.26	2.43	2.20
ser877	8.90	4.19	12.58	11.46	13.72	13.93	13.21	11.15

TABLE 22 (Continued)

series	y+1		y+2		y+3		y+4	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser904	5.54	2.29	6.11	2.34	6.99	2.70	7.91	4.46
ser922	4.04	3.82	7.51	6.85	12.16	10.67	17.12	17.91
mean	15.42	10.43	18.16	13.26	18.68	13.73	19.83	15.89

TABLE 22 (Continued)

series	y+5		y+6		y+7		y+8	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	11.94	8.41	14.64	11.93	14.48	11.68	13.00	13.80
ser409	47.85	38.25	54.16	45.38	56.71	34.82	73.32	44.11
ser418	16.17	11.46	19.61	18.09	22.10	21.80	20.21	27.14
ser427	17.71	13.11	19.52	19.89	15.01	14.23	15.65	13.70
ser436	3.13	2.96	3.42	3.04	3.57	3.20	4.76	5.03
ser445	19.25	14.33	25.93	21.29	19.53	18.71	15.28	14.70
ser454	25.23	13.89	28.84	18.45	31.51	9.20	40.82	27.01
ser463	29.36	16.93	26.71	18.16	26.09	20.07	29.75	26.55
ser472	30.81	17.07	31.15	19.33	30.31	22.91	32.93	30.57
ser481	30.26	26.77	28.03	15.19	35.28	20.92	23.89	23.51
ser490	16.47	9.95	12.70	10.66	15.58	13.14	12.53	8.93
ser499	10.14	7.87	11.08	12.98	13.31	9.93	8.88	8.73
ser508	8.03	4.58	6.99	6.93	8.43	4.34	7.82	5.22
ser526	17.23	14.10	13.72	6.59	23.25	23.28	15.93	14.90
ser544	4.02	3.81	4.78	4.65	7.88	7.25	11.63	14.46
ser562	24.76	19.93	29.59	23.07	20.92	17.32	21.03	17.72
ser571	22.36	12.01	23.70	6.76	26.18	7.44	26.87	9.81
ser598	12.31	12.51	11.46	6.84	12.66	7.05	15.37	13.00
ser616	16.54	16.60	15.70	12.71	17.76	15.66	14.41	11.64
ser634	29.39	19.94	30.99	20.38	32.35	29.14	22.88	19.38
ser643	20.29	19.57	20.44	15.43	19.98	14.78	20.26	12.69
ser652	31.37	23.38	29.64	21.57	25.17	16.96	29.48	30.77
ser661	31.41	8.68	27.67	11.13	32.83	28.76	27.34	30.92
ser670	24.37	10.11	26.01	13.40	30.95	17.98	35.07	11.51
ser679	33.10	34.04	30.21	36.33	31.06	12.92	36.10	23.50
ser688	28.33	14.69	33.18	25.96	37.56	23.88	40.36	37.28
ser697	8.95	9.00	9.52	9.25	8.05	7.59	8.09	6.74
ser706	21.71	22.63	22.23	22.02	20.29	18.16	25.42	28.26
ser715	82.86	67.20	70.07	49.93	85.79	44.61	95.77	39.53
ser724	19.24	16.99	23.70	21.67	22.62	21.16	24.78	20.07
ser733	26.17	19.02	26.45	24.35	25.76	18.04	32.47	29.70
ser742	15.36	14.74	14.96	14.07	13.77	10.50	12.64	9.32
ser751	14.71	11.36	17.16	19.39	15.37	14.90	16.37	14.53
ser760	12.40	11.93	11.58	8.04	15.45	11.61	14.27	13.98
ser769	15.02	12.47	13.28	13.69	15.98	14.02	16.26	14.58
ser787	13.44	9.90	11.48	9.38	11.36	7.05	9.42	8.25
ser796	39.38	35.07	26.49	27.55	35.06	34.10	32.50	26.39
ser805	4.79	5.23	6.25	5.94	7.72	7.93	7.86	7.56
ser823	1.94	1.98	2.54	2.76	2.45	1.82	3.27	3.04
ser832	4.19	3.48	4.80	4.47	5.73	5.39	6.66	6.68
ser877	8.60	5.37	6.80	4.93	7.18	3.28	13.01	10.93
ser904	8.90	4.91	8.05	4.54	9.07	4.34	10.30	5.97
ser922	21.06	21.83	26.13	24.49	33.21	32.35	36.09	36.53
mean	20.48	15.54	20.50	16.11	21.98	15.91	22.81	18.11

TABLE 22 (Continued)

series	y+9		y+10		y+11		y+12	
	mape	m-ape	mape	m-ape	mape	m-ape	mape	m-ape
ser400	10.76	5.89	8.91	5.22	8.18	6.62	8.62	8.72
ser409	46.39	27.32	45.69	45.90	38.69	22.49	33.47	32.16
ser418	20.76	11.70	14.62	9.27	26.47	19.51	21.80	25.49
ser427	15.00	16.69	13.85	6.83	13.87	10.24	9.14	8.05
ser436	7.33	6.75	9.21	9.25	10.55	10.64	11.22	11.37
ser445	18.48	12.43	25.80	26.63	27.32	21.93	27.19	25.02
ser454	32.63	14.39	15.43	9.82	12.29	7.26	10.41	10.95
ser463	24.48	20.31	31.54	25.82	38.50	37.91	31.76	34.92
ser472	42.22	40.95	39.63	36.61	39.16	35.58	41.69	39.98
ser481	32.55	25.95	48.75	36.72	51.97	45.63	59.93	47.56
ser490	10.90	8.25	19.03	19.59	16.37	17.17	6.60	6.19
ser499	12.97	11.59	15.47	12.01	10.72	8.48	6.05	3.47
ser508	8.97	5.39	10.41	4.87	10.01	6.29	3.60	1.40
ser526	17.87	15.68	28.43	23.00	17.28	16.07	14.89	14.60
ser544	13.71	13.44	14.18	14.30	16.42	15.93	20.18	19.99
ser562	23.57	19.26	17.55	9.60	23.41	14.70	23.66	25.93
ser571	25.84	8.69	16.94	9.09	7.32	6.16	10.57	8.16
ser598	16.95	11.05	18.01	17.50	24.38	23.86	26.84	24.16
ser616	13.44	11.44	15.67	16.95	14.49	12.30	7.56	6.68
ser634	13.17	8.09	23.40	29.09	28.41	34.83	14.92	12.38
ser643	23.97	26.18	19.37	19.58	17.84	10.18	17.96	15.87
ser652	32.24	30.78	24.19	19.99	19.12	15.06	13.78	13.90
ser661	28.44	27.17	26.07	23.38	20.97	19.36	8.13	3.47
ser670	30.16	14.53	39.94	27.83	24.05	8.90	23.82	24.03
ser679	44.59	36.35	54.24	55.93	63.10	63.53	71.01	71.41
ser688	35.44	26.80	28.87	22.49	23.58	22.07	14.08	14.38
ser697	8.08	6.87	6.84	7.51	7.97	7.70	5.06	6.12
ser706	28.82	27.00	34.54	36.84	36.64	35.88	37.31	36.68
ser715	72.31	34.43	73.20	30.00	105.67	41.84	35.43	42.08
ser724	29.89	28.97	20.57	19.29	23.59	20.45	26.80	27.62
ser733	33.67	32.81	25.15	16.61	23.82	11.17	25.27	22.05
ser742	9.47	6.36	8.06	7.16	4.36	2.84	3.06	1.82
ser751	14.47	11.58	15.29	15.42	8.33	5.51	5.99	3.86
ser760	11.37	9.00	10.57	2.89	9.89	10.59	5.93	6.27
ser769	20.13	22.78	19.01	18.02	16.07	14.12	11.58	9.18
ser787	9.43	7.48	8.30	5.04	9.71	5.65	6.09	5.56
ser796	38.90	30.74	39.54	44.90	31.18	28.46	10.27	5.75
ser805	8.11	7.66	8.39	8.17	8.02	8.22	8.66	8.36
ser823	2.92	2.35	2.86	2.14	2.62	1.65	2.15	1.63
ser832	6.76	6.94	7.90	8.21	8.77	8.49	9.87	9.78
ser877	13.48	14.23	12.65	11.36	9.58	4.24	5.70	6.13
ser904	7.57	4.39	7.85	5.48	5.08	3.90	6.26	6.76
ser922	37.81	38.25	38.18	37.03	37.79	36.13	38.46	38.83
mean	22.23	17.28	22.42	18.91	22.18	17.66	18.20	17.41

APPENDIX G

TIME SERIES INFORMATION CONTENT

TIME SERIES INFORMATION CONTENT

This section is quoted directly from Pack and Downing's paper from p.13 through p.18 in 1983. It provides the necessary information concerning the two characters specified for each time series in Table 23.

Uninformative Series

- I. The stationary nonseasonal ARIMA model will produce forecasts equal to the estimated mean after relatively few time periods (i.e., for relatively short lead time).
- II. The models in this group are constant mean (157), random walk (400, 778, 895) and simple linear trend (319, 814). The forecasts produced are constant for all lead times in the first two models, and a simple trend line in the third model.
- III. The models in this group are of the form the ARIMA school generally associates with exponential smoothing [(0,1,1) is simple exponential smoothing]. The (0,1,q) model forecasts are constant for lead times greater than or equal to q if there is no trend in the model. Other models in the group, including (0,2,1), produce simple linear trend forecasts for lead times beyond the indicated moving

average order.

- IV. The first difference models in this group produce forecasts that are constant for sufficiently long lead times (dependent on the strength of AR), or simple linear trend forecasts for those time series that indicate "trend".

The point is that in all these cases the forecast patterns produced are relatively simple. They are not very likely to describe what really happens beyond a relatively short lead time. These forecast patterns would be easily matched by a number of other extrapolative methods.

Informative Series

- I. This model is dubiously called "informative". It represents the weakest form of seasonal structure, MA structure in the absence of AR or differencing.
- II. Order s (quarterly $s=4$, yearly $s=12$) relationships are present, but order 1 relationships are not, considering AR or differencing, irrespective of MA.
- III. In contrast to II, order s and order 1 relationships appear multiplicatively in the model in the AR and/or differencing structure. The 23 series listed in this group suggest this may be a common seasonal structure, and our experience suggests that it is. Any extrapolative method which depends primarily on an autoregression involving lags 1 , s , and $s+1$ should forecast competitively with ARIMA on these

time series.

IV. About half of these cases are like those in III, but reflect higher orders of autoregression (e.g., 652, 661). The other cases are the unique cases of historical pattern present in the 111 M-competition time series. In principle, one might expect the ARIMA identification process to perform most successfully relative to other extrapolative methods on this group of 16 time series.

PS: short, no analysis. Pack and Downing, and also Makridakis et al. agree that these series are seasonal.

UM: unsuitable, no analysis. Makridakis et al. consider these series to be seasonal, but Pack and Downing do not.

SH: short, no analysis, and not type PS (see above).

UN: unsuitable (contains structural change(s)), no analysis, and not type UM (see above).

APPENDIX H

EXPONENTIALLY WEIGHTED REGRESSION METHOD

EXPONENTIALLY WEIGHTED REGRESSION METHOD

A regression forecasting method simply fits a curve or model to the historical data of a given time series, and then extrapolates the curve or model into the future to obtain predicted values. The fitting process minimizes the sum of squares of the differences between the data values in the historical training data and the fitted value at each datum point.

The problem with doing this is that "old" data points near the beginning of the training set have just as much influence over the fitted model as do the "recent" data near the end of the training set. This is quite unreasonable: the points to be forecast are nearer the "recent" data, which should be more predictive of the future than the "old" data. If we want to forecast next year's sales, the past year's sales are much more important than ancient data from five or ten years ago. The model needs to ignore or discount old data in some systematic way. Exponential weighting accomplishes this [Harrison and Akram 1982]. In the least squares fit, each residual (the difference between the data and the fit) is squared and added into the sum of squares, but each squared residual is multiplied by a weight before the addition:

$$S(a) = \sum_i w_i (y_i - f(a, t_i))^2$$

Here, t_i is the i -th time abscissa, y_i is the i -th data ordinate, a is the vector of parameters of the model, $f(a, t_i)$ is the model being fitted to the y_i , w_i is the weight for the i -th datum point, and $S(a)$ is the sum of squares to be minimized. The exponentially weighted regression method takes w_i to be an exponential function of the time t_i :

$$w_i = b^{t_i},$$

where b is a real number greater than one, to be determined along with the parameter a in the fit. Because b is greater than one, the weights corresponding to the larger (later) t_i are larger than those corresponding to the earlier points.

In the exponentially weighted regression results used in this thesis for comparison with NAIIVE and neural net models, the model f was simply a straight line:

$$f(a, t_i) = a_1 + a_2 t_i$$

(A logistic curve might be more reasonable and has been used in other time series forecasting, but the nonlinear least squares fitting process sometimes breaks down and must be replaced by an even slower optimization problem.) For quarterly and monthly series, the model was multiplied by fitted seasonality factors. Also applied to the forecast values were fitted "shrinking factors", one factor that shrinks the forecast toward the data ordinate y_i in the fitted set, and one factor that shrinks the forecast toward

zero. Without shrinking, regression models tend to be a little wild and should not be used. In the M-111 study, a linear regression model without exponential weighting or shrinking was one of the worst methods tested; with exponential weighting, seasonality factors, and shrinking, regression becomes competitive with some of the best methods. Exponentially weighted regression is related to exponential smoothing models and to Box-Jenkins ARIMA models.

APPENDIX I

COMPARISON OF DIFFERENT FORECASTORS
AT 1_STEP FORECASTING

COMPARISON OF DIFFERENT FORECASTORS
AT 1_STEP FORECASTING

This section presents a comparison of the performance of seven forecasting methods. The results of NAIVE 1, 2, and exponential regression are provided by Chandler (1992). The Autobox and the NN-PDP are the Box-Jenkins software and the Parallel Distributed Processing package used in Patil's thesis (1990), respectively. BPN and CPN are the backpropagation neural network and the counterpropagation neural network in this thesis.

The numbers in the parentheses inside the tables below are the available numbers of data points. The two letters in the parentheses inside the tables below indicate the information of that time series (see APPENDIX G). The parentheses that follow the "yearly time series", the "Quarterly time series" and the "Monthly time series" in the tables below show the neural network architectures used by BPN and CPN. Table 23 shows the comparison of the MAPE values of these seven prediction implementations over 72 M-competition data series. We can observe that Patil's Autobox results are poor when compared with the other methods and NN-PDP does an extremely good job on the annual time series.

Since the extents of yearly and quarterly time series in Makridakis et al.'s report (1982) (i.e. ser4 <= yearly <= ser175, ser184 <= quarterly <= ser382) are different from those (i.e. ser4 <= yearly <= ser112, ser121 <= quarterly <= ser382) in Patil's thesis, the comparison tables below present only those time series shown on both extents. That is, the annual time series table contains the results from ser4 through ser112 and the quarterly time series table contains the outputs from ser184 through ser382.

TABLE 23

MAPE COMPARISON OF NEURAL NETWORKS, NAIVE I, II, EXPONENTIAL REGRESSION AND BOX-JENKINS AT 1_STEP FORECASTING

Yearly time series. (BPN: 2-2-1, CPN: 3-6-3)

series	NAIVE1	NAIVE2	Autobox	NN-PDP	BPN	CPN	Expreg
4 (23,SH)	15.51	15.51	23.53	6.43	11.23	15.49	11.80
13 (52,UN)	7.76	7.76	6.58	2.59	7.46	11.95	7.40
22 (14,SH)	0.47	0.47		0.04	0.23	0.23	0.36
31 (22,SH)	9.08	9.08	18.07	0.55	9.22	9.07	8.80
40 (21,SH)	4.51	4.51	10.16	1.03	4.27	4.12	7.23
49 (19,SH)	28.39	28.39	24.77	7.65	26.13	29.38	27.70
58 (18,SH)	11.21	11.21	72.58	1.34	6.90	6.11	8.29
67 (13,SH)	3.96	3.96		0.93	3.55	3.47	4.19
76 (13,SH)	6.10	6.10		2.27	5.26	6.22	6.11
85 (20,SH)	3.76	3.76	37.00	0.29	3.86	3.05	3.03
94 (13,SH)	3.32	3.32		0.76	2.73	6.87	3.11
103 (13,SH)	4.66	4.66		0.41	1.44	1.41	1.49
112 (38,U4)	1.77	1.77	4.55	0.10	0.66	1.08	1.77
mean	14.34	14.34	24.65	2.50	8.72	10.03	9.50

Quarterly time series. (BPN: 8-8-1, CPN: 9-18-9)

series	NAIVE1	NAIVE2	Autobox	NN-PDP	BPN	CPN	Expreg
184 (48,I3)	18.82	6.34	20.61	6.92	11.43	15.70	6.32
193 (20,SH)	24.00	21.71	44.43	32.52	42.08	24.56	24.42
202 (24,SH)	2.96	3.34	21.51	3.95	4.48	3.16	4.00
211 (30,SH)	21.85	32.46	26.31	36.03	34.79	16.62	19.43
220 (34,SH)	28.52	32.48	37.10	47.59	41.97	28.14	29.08
229 (32,PS)	13.22	1.68	42.19	4.19	3.11	15.36	1.65
238 (57,I3)	5.16	3.22	14.22	4.79	6.74	5.11	2.64
265 (60,U3)	4.92	3.17	21.15	5.52	3.88	3.38	3.61
283 (76,U4)	2.77	2.16		2.90	2.49	2.06	1.41
292 (38,I3)	22.37	32.60	12.67	13.94	9.73	14.89	20.36
301 (56,I4)	2.12	1.36	2.91	2.25	2.71	1.42	1.12
310 (52,U4)	2.49	0.95	4.46	2.90	1.36	1.45	0.74
319 (52,U2)	3.18	2.71	14.30	4.39	3.48	3.94	3.11
328 (36,I3)	3.37	1.67	8.02	3.14	3.07	3.31	1.87
337 (56,I4)	10.44	2.38	8.52	3.86	4.29	8.68	2.53
346 (40,UN)	11.75	16.86	20.23	11.15	14.12	12.72	12.86
355 (51,UN)	3.15	2.73	4.75	3.92	2.76	4.57	2.65
364 (36,U4)	1.16	1.45	3.73	1.25	2.25	1.27	0.75
382 (40,U1)	20.08	13.46	32.78	18.25	23.64	20.23	12.87
mean	11.09	10.03	18.88	11.47	11.99	10.25	8.33

TABLE 23 (Continued)

Monthly time series. (BPN: 24-24-1, CPN: 25-50-25)

series	NAIVE1	NAIVE2	Autobox	NN-PDP	BPN	CPN	Expreg
400(90,U2)	7.87	9.05	14.14	11.42	8.04	8.24	10.28
409(78,I2)	27.46	30.86	42.80	32.09	28.34	31.56	20.63
418(63,I1)	21.95	16.24	19.23	27.37	23.17	20.57	13.76
427(105,I2)	10.76	7.24	9.89	8.53	8.75	9.23	6.55
436(101,I4)	1.87	0.68	7.09	1.85	2.05	1.79	0.69
445(75,I3)	15.78	11.75	25.71	12.87	16.22	15.97	10.86
454(101,I2)	26.10	10.44	8.48	6.85	10.66	21.36	8.45
463(124,I3)	16.07	7.37	6.19	19.95	20.38	18.62	6.86
472(80,I4)	21.59	8.10	19.21	20.76	12.21	21.67	7.59
481(56,U4)	38.20	39.24	32.36	31.00	44.31	36.54	32.16
490(80,I3)	18.12	7.63	7.90	8.71	10.95	16.75	6.71
499(105,U4)	10.24	8.33	13.50	10.02	9.05	13.59	6.47
508(105,I4)	8.60	5.67	16.11	5.89	6.55	8.91	4.85
526(79,I4)	14.44	10.60	9.95	14.75	12.87	14.16	8.26
544(57,I4)	2.28	2.00	2.72	3.98	4.71	2.25	2.21
562(80,U1)	25.72	19.55		77.15	64.22	29.49	16.57
571(64,I3)	9.12	6.44	5.57	5.82	6.64	10.35	7.14
580(52,U1)	0.55	0.66	3.03	3.28	3.37	0.54	0.62
589(54,U1)	3.15	3.63	9.66	8.20	8.09	3.21	3.63
598(84,I4)	6.08	3.33	23.49	6.36	10.84	5.97	2.98
616(120,I4)	11.65	5.89	7.22	4.62	5.92	11.32	4.25
634(87,I3)	16.71	16.73	13.70	18.67	22.33	23.73	12.87
643(64,I3)	18.02	15.37	18.92	20.29	18.54	16.16	17.06
652(67,I4)	24.19	11.47	15.23	15.54	15.83	22.15	9.94
661(66,I4)	21.54	12.60	19.80	20.52	20.34	21.24	11.53
670(65,I3)	29.26	11.79	28.39	30.27	22.90	26.44	11.98
679(66,U3)	10.85	16.14	20.42	40.01	37.60	16.92	17.57
688(65,I3)	16.80	5.49	12.86	11.12	11.06	14.24	6.04
697(66,I3)	3.31	2.60	4.13	3.63	3.20	3.44	1.66
706(65,I3)	3.37	3.61	20.11	9.66	10.73	3.54	3.83
715(66,U1)	64.55	61.50	61.31	83.83	57.52	59.37	66.19
724(62,U3)	18.39	17.37	17.43	22.44	18.87	18.69	16.08
733(66,I4)	29.12	19.56	14.21	21.52	17.19	26.29	16.71
742(66,I3)	5.51	2.56	5.00	2.05	2.57	4.30	2.18
751(63,I3)	9.95	5.00	5.19	8.17	11.84	9.01	3.88
760(66,I3)	22.81	4.44	7.75	8.05	7.85	17.96	4.32
769(66,I3)	13.55	7.35		10.46	7.99	13.66	6.62
787(111,I3)	6.58	1.91	3.17	1.87	4.23	9.96	1.93
796(114,I4)	43.75	18.08	17.68	15.96	16.36	25.41	15.34
805(114,U3)	1.19	0.85	7.93	1.18	1.06	1.38	0.74
823(114,U3)	0.47	0.57	1.29	0.65	0.65	0.64	0.54
832(114,U3)	1.29	0.70	7.96	1.29	0.98	0.84	0.33
877(102,I4)	9.50	3.48	3.73	3.38	6.48	9.12	2.58
904(90,I3)	5.00	2.11	3.88	3.22	4.82	5.00	2.04
913(51,UM)	21.86	22.32	58.53	48.90	58.25	22.62	27.67

TABLE 23 (Continued)

Monthly time series. (BPN: 24-24-1, CPN: 25-50-25)

series	NAIVE1	NAIVE2	Autobox	NN-PDP	BPN	CPN	Expreg
922 (126, U3)	3.89	4.67	26.35	5.22	4.74	3.89	4.66
958 (54, UM)	6.81	4.05	13.67	11.36	12.22	8.11	3.92
967 (54, UN)	5.94	7.33	38.74	20.28	21.11	9.46	5.76
mean	14.70	10.16	15.91	14.65	14.40	14.18	9.40

The above mean values of each method (except Autobox) are the means of the selected series corresponding to those series available for the Autobox (i.e. 8 for yearly, 18 for quarterly, and 46 for monthly). The actual mean values of each method are presented in the following tables.

Yearly time series. (BPN: 2-2-1, CPN: 3-6-3)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	10.25	10.25	24.65	1.88	6.38	7.57	7.02

Quarterly time series. (BPN: 8-8-1, CPN: 9-18-9)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	10.65	9.62	18.88	11.02	11.49	9.82	7.97

Monthly time series. (BPN: 24-24-1, CPN: 25-50-25)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	14.91	10.30	15.91	15.86	15.30	14.49	9.49

The following tables are the comparisons of these methods but with different neural network architecture. The first two tables are the results corresponding to the selected time series available for Autobox while the last

two are the comparison tables of actual means.

Quarterly time series. (BPN: 4-4-1, CPN: 5-10-5)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	11.09	10.03	18.88	10.89	11.20	10.88	8.33

Monthly time series. (BPN: 12-12-1, CPN: 13-26-13)

method	NAIVE I	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	14.70	10.16	15.91	11.53	12.76	15.06	9.40

Quarterly time series. (BPN: 4-4-1, CPN: 5-10-5)

method	NAIVE 1	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	10.65	9.62	18.88	10.47	10.74	10.43	7.97

Monthly time series. (BPN: 12-12-1, CPN: 13-26-13)

method	NAIVE I	NAIVE 2	Autobox	NN-PDP	BPN	CPN	Exp-reg
mean	16.48	10.10	15.91	12.52	13.45	16.12	9.37

APPENDIX J
BACKPROPAGATION PROGRAM

```

#include <math.h>
#define prime 200
/* prime the random number generator 200 times */

#define maxape 70
/* maxape is the maximum total # of apes */

#define bias 1.0
#define momentum 0.1
#define rate 0.9 /* 0.01 <= rate <= 1.0 */

/* momentum is the factor to improve BPN training time */
/* rate is the training rate coefficient of the weight change */

#define Yser 20
#define Qser 23
#define Mser 68

/* There are 13, 30 and 68 time series for yearly, quarterly and
monthly data, respectively. */

#define Ymax 6
#define Qmax 8
#define Mmax 18

/* The maximum # of forecasting steps for yearly, quarterly and
monthly time series is 6, 8, and 18, respectively. */

#define maxset 50
#define numerr 30
#define maxobs 150

#define UPbound 0.99
#define LOWbound 0.01

/* UPbound and LOWbound are the upper delimiter and lower
delimiter of weight vectors */

#define TRUE 1
#define FALSE 0

float MAX=0.0,MIN=9999999.0;
/* initial values of MAX and MIN for normalization of
observations */

float W_change[50][50];
/* W_change is the previous weight change */

float w[100][100],v[100][100];

/* wx[],wy[] are the weight matrices connecting Input, Hidden

```

```

layers vx[],vy[] are the weight matrices connecting Hidden, Output
layers */

float err_set[50][30];
long seed = 1.0;
int I,H,O;
/* I,H,O are the # of neurons in Input, Hidden, Output layers*/

int N; /* N_step ahead forecasting */
int Pass;

/* Pass is the number of times to select the training patterns
and adjust the weight matrices */

float mape[30][70],m_mape[30][70];

main()
{
float random();
float actual[150];
/* actual[] is the array of actual values for training and
testing */

int time=1;
/* 1 <= time <=13 for yearly , 14 <= time <= 43 for quarterly
44 <= time <= 111 for monthly */

int numofobs;
/* numofobs is the # of observations in a certain time series */
int i,done=0,add;
int j;
char sernum[7];
FILE *fp,*fopen();
printf("\nPlease enter\n");
printf("the number of neurons for Input layer : ");
scanf("%d",&I);
printf("%d\n",I);
printf("the number of neurons for Hidden layer : ");
scanf("%d",&H);
printf("%d\n",H);
printf("the number of neurons for Output layer : ");
scanf("%d",&O);
printf("%d\n",O);
printf("the number of values to be forecast (N = 1,2,4,6,8,12,18)
: ");
scanf("%d",&N);
printf("%d\n",N);
printf("the number of train passes : ");
scanf("%d",&Pass);
printf("%d\n",Pass);
printf("\n\n");
if (O != N) {

```

```

    printf("The future values cannot be predicted!\n");
    exit(1);
};
printf("I = %d , H = %d , O = %d\n",I,H,O);
printf("%d_step ahead forecasting\n",N);

for(i=0;i<prime;i++) random();
fp = fopen("data","r");

for(i=0;i<N;i++)
    for(j=0;j<maxape;j++) {
        mape[i][j] = 0.0;
        m_mape[i][j] = 0.0;
    };
printf("\n***** Yearly time series forecasting
*****\n");
while (time <= Yser && !feof(fp)) {
    /* for yearly time series */
    for(i=0;i<maxobs;i++)
        actual[i] = 0.0;
    for(i=0;i<maxset;i++)
        for(j=0;j<numerr;j++)
            err_set[i][j] = 0.0;
    fscanf(fp,"%s",sernum);
    fscanf(fp,"%d",&numofobs);
    for(i=0;i<numofobs;i++)
        fscanf(fp,"%f",&actual[i]);
    /*
    printf("\n%s %d\n",sernum,numofobs);
    if (numofobs < I || numofobs <= Ymax) {
        printf("There is no enough data points in the time
series : %s\n",sernum);
    }
    else if (N > Ymax) {
        printf("This %d step forecasting cannot be done ",N);
        printf("with time series %s\n",sernum);
    }
    else done = trainwork(numofobs-Ymax,numofobs,sernum,actual);
    if (done) {
        testwork(Ymax,numofobs,sernum,actual,time-1);
    };
    done = 0;
*/
    MAX = 0.0;
    MIN = 9999999.0;
    time++;
};
/*
if (N <= Ymax) {
    MEAN(Yser);
    M_MEDIAN(Yser);
};

```

```

*/
for(i=0;i<N;i++)
    for(j=0;j<maxape;j++) {
        mape[i][j] = 0.0;
        m_mape[i][j] = 0.0;
    };
time = 1;
printf("\n***** Quarterly time series forecasting
*****\n");
while (time <= Qser && !feof(fp)) {
    /* for quarterly time series */
    for(i=0;i<maxobs;i++)
        actual[i] = 0.0;
    for(i=0;i<maxset;i++)
        for(j=0;j<numerr;j++)
            err_set[i][j] = 0.0;
    fscanf(fp,"%s",sernum);
    fscanf(fp,"%d",&numofobs);
    for(i=0;i<numofobs;i++)
        fscanf(fp,"%f",&actual[i]);
    /*
    printf("\n%s %d\n",sernum,numofobs);
    if (numofobs < I || numofobs <= Qmax) {
        printf("There is no enough data points in the time
series : %s\n",sernum);
    }
    else if (N > Qmax) {
        printf("This %d_step forecasting cannot be done ",N);
        printf("with time series %s\n",sernum);
    }
    else done = trainwork(numofobs-Qmax,numofobs,sernum,actual);
    if (done) {
        testwork(Qmax,numofobs,sernum,actual,time-1);
    };
    done = 0;
*/
    MAX = 0.0;
    MIN = 9999999.0;
    time++;
};
/*
if (N <= Qmax) {
    MEAN(Qser);
    M_MEDIAN(Qser);
};
*/
for(i=0;i<N;i++)
    for(j=0;j<maxape;j++) {
        mape[i][j] = 0.0;
        m_mape[i][j] = 0.0;
    };
time = 1;

```

```

printf("\n***** Monthly time series forecasting
*****\n");
while (time <= Mser && !feof(fp)) {
/* for monthly time series */
for(i=0;i<maxobs;i++)
actual[i] = 0.0;
for(i=0;i<maxset;i++)
for(j=0;j<numerr;j++)
err_set[i][j] = 0.0;
fscanf(fp,"%s",sernum);
fscanf(fp,"%d",&numofobs);
printf("\ns %d\n",sernum,numofobs);
for(i=0;i<numofobs;i++)
fscanf(fp,"%f",&actual[i]);
if (numofobs < I || numofobs <= Mmax) {
printf("There is no enough data points in the time
series : %s\n",sernum);
}
else if (N > Mmax) {
printf("This %d_step forecasting cannot be done ",N);
printf("with time series %s\n",sernum);
}
else done = trainwork(numofobs-Mmax,numofobs,sernum,actual);
if (done) {
testwork(Mmax,numofobs,sernum,actual,time-1);
};
done = 0;
MAX = 0.0;
MIN = 9999999.0;
time++;
};

if (N <= Mmax) {
MEAN(Mser);
M_MEDIAN(Mser);
};

fclose(fp);
}

/*-----*/
/* This is the random number generator function from the */
/* paper below. Park and Miller, "Random number generators:*/
/* good ones are hard to find. Comm. ACM 31, 10(Oct. 1988),*/
/* 1192-1201. */
float random()
{
float a=16807.0,m=2147483647.0,q=127773.0,r=2836.0;
float lo,test;
int hi;
hi = (int)(seed/q); lo = seed - q*hi;

```

```

test = a*lo - r*hi;
if (test > 0.0) seed = test;
else seed = test + m;
return (seed/m);
}

/*-----*/
/* This is the function to train BPN network with the */
/* training set from the time series data. */
trainwork(train,numofobs,sernum,actual)
int train,numofobs;
char sernum[7];
float actual[];
{
float random(),normalization(),root[150];
float activation();
float obs[150][50]; /* obs[] is the array of observations */
float set[150][50];
float product[50],H_net[50],O_net[50];
float H_OUT[50],O_OUT[50];

/* product is the product of observation and weight. H_OUT[] is
the net outputs for Hidden neurons while O_OUT[] for Output
neurons. */

float err[150],error,diff,forecast[50];
float act[150][50],min[150];
int i,j,h,o,c=0;
int ind,in,pass;
int b,prev[150],flag;

printf("----Training for time series : %s\n",sernum);
train = train - I - N;
if (train <= 1) {
printf("There is no enough data points in the time series :
%s\n",sernum);
return(0);
};

for(i=0;i<=train;i++) {
for(j=0;j<I;j++) {
obs[i][j] = actual[j+i];
};
for(b=j;b<I+N;b++) {
obs[i][b] = actual[b+i];
};
};

init weight();
for(I=0;i<50;i++)
for(j=0;j<50;j++)

```



```

        W_change[i][j] = 0.0;
while (c <= train) {
    root[c] = normalization(obs[c],I+N);
    min[c] = MIN;

/* the root=(maxobs - minobs) of the observations is saved for
denormalization purpose when comes to the testing process save
the normalized observations for training */

    for(i=0;i<I+N;i++) {
        set[c][i] = obs[c][i];
    };

/* operations between Input and Hidden layers */

    for(h=0;h<H;h++) {
        for(i=0;i<I;i++)
            product[i] = 0.0;
        for(i=0;i<I;i++) {
            if (i == I)
                product[i] = bias * w[i][h];
            else product[i] = obs[c][i] * w[i][h];
        };
        H_net[h] = 0.0;
        for(i=0;i<I;i++)
            H_net[h] = H_net[h] + product[i];
        H_OUT[h] = activation(H_net[h]);
    };

/* operations between Hidden and Output layers */

    for(o=0;o<O;o++) {
        for(h=0;h<=H;h++)
            product[h] = 0.0;
        for(h=0;h<=H;h++) {
            if (h == H)
                product[h] = bias * v[h][o];
            else product[h] = H_OUT[h] * v[h][o];
        };
        O_net[o] = 0.0;
        for(h=0;h<=H;h++)
            O_net[o] = O_net[o] + product[h];
        O_OUT[o] = activation(O_net[o]);

        act[c][o] = actual[o+c+I];
    };

    adjust_weight(obs[c],O_OUT,H_OUT);
/* init_adjust(obs[c]);*/
    c++;

```

```

    };
b=0;
for(pass=1;pass<=Pass-c;pass++) {
    ind = (int)(random() * c);

    for(h=0;h<H;h++) {
        for(i=0;i<=I;i++) {
            if (i == I)
                product[i] = bias * w[i][h];
            else product[i] = set[ind][i] * w[i][h];
        };
        H_net[h] = 0.0;
        for(i=0;i<=I;i++)
            H_net[h] = H_net[h] + product[i];
        H_OUT[h] = activation(H_net[h]);
    };

    for(o=0;o<O;o++) {
        for(h=0;h<=H;h++) {
            if (h == H)
                product[h] = bias * v[h][o];
            else product[h] = H_OUT[h] * v[h][o];
        };
        O_net[o] = 0.0;
        for(h=0;h<=H;h++)
            O_net[o] = O_net[o] + product[h];
        O_OUT[o] = activation(O_net[o]);

        forecast[o] = O_OUT[o] * root[ind] + min[ind];
        diff = act[ind][o] - forecast[o];
        if (diff < 0)
            diff = forecast[o] - act[ind][o];
        err[o] = diff / act[ind][o];
    };
    error = 0.0;
    for(i=0;i<N;i++) error += err[i];
    error /= N;

    adjust_weight(set[ind],O_OUT,H_OUT);

    if (error <= 0.000001)
        return(1);
};
printf("The training of BPN network for time series ts is
finished.\n",senum);
return(1);
}

```

```

-----*/
/* This is the function to initialize the connecting weight */
/* vectors in the BPN network. */
init_weight()
{
float random(),delimiter=1.0;
/* delimiters are the bounds for weights */

float sqr_sum=0.0;
int i,h,o;
int ind;

/* initialize the weights between Input and Hidden layers */
for(h=0;h<H;h++) {
for(i=0;i<I;i++) {
while (delimiter > UPbound || delimiter < LOWbound)
delimiter = random();
w[i][h] = delimiter - 0.5;
delimiter = 1.0;
};
};

/* initialize the weights between Hidden and Output layers */
for(o=0;o<O;o++) {
for(h=0;h<H;h++) {
while (delimiter > UPbound || delimiter < LOWbound)
delimiter = random();
v[h][o] = delimiter - 0.5;
delimiter = 1.0;
};
};
}

-----*/
/* The function adjusts the weight vectors in the BPN */
/* networks according to the methods from the papers of */
/* Hecht-Nielsen[1987a]. */
adjust_weight(x,out,h_out)
float x[];
float out[],h_out[];
{
float target[50];
float hdelta[50],odelta[50];
int i,h,o;

for(i=0;i<50;i++)
odelta[i] = 0.0;
for(i=0;i<N;i++)
target[i] = x[i+I];

```

```

/* adjust the weights connecting Hidden and Output layers */
for(o=0;o<O;o++)
odelta[o] = out[o] * (1 - out[o]) * (target[o] - out[o]);
for(o=0;o<O;o++) {
for(h=0;h<=H;h++) {
if (h == H) {
v[h][o] = v[h][o] + rate*odelta[o]*bias;
v[h][o] += (momentum*W_change[h][o]);
W_change[h][o] = rate*odelta[o]*bias;
W_change[h][o] += (momentum*W_change[h][o]);
}
else {
v[h][o] = v[h][o] + rate*odelta[o]*h_out[h];
v[h][o] += (momentum*W_change[h][o]);
W_change[h][o] = rate*odelta[o]*h_out[h];
W_change[h][o] += (momentum*W_change[h][o]);
};
};
};

/* adjust the weights connecting Input and Hidden layers */
for(h=0;h<H;h++) {
hdelta[h] = 0.0;
for(o=0;o<O;o++)
hdelta[h] = hdelta[h] + odelta[o]*v[h][o];
hdelta[h] = hdelta[h] * h_out[h] * (1 - h_out[h]);
};
for(h=0;h<H;h++) {
for(i=0;i<I;i++)
if (i == I)
w[i][h] += (rate*hdelta[h]*bias);
else w[i][h] = w[i][h] + rate*hdelta[h]*x[i];
};
}

-----*/
/* This function normalizes the input vectors and weight */
/* vectors. */
float normalization(x,b)
float x[];
int b;
{
int i;
MAX = 0.0;
MIN = 9999999.0;
/*float sqr sum = 0.0;
for(i=0;i<I;i++)
sqr_sum = sqr_sum + x[i]*x[i];

```

```

for(i=0;i<I;i++)
  x[i] = x[i] / sqrt(sqr_sum);*/
for(i=0;i<b;i++)
  if (MAX <= x[i])
    MAX = x[i];
for(i=0;i<b;i++)
  if (MIN >= x[i])
    MIN = x[i];
for(i=0;i<b;i++)
  if (MAX != MIN)
    x[i] = (x[i] - MIN) / (MAX - MIN);
  else x[i] = 1.0;

return(MAX-MIN);
)

/*-----*/
/* This is the function to test the trained BPN network */
/* model with the remaining k data points (observations) */
/* of a certain time series and evaluate its forecasting */
/* ability. */
testwork(n,numofobs,sernum,actual,t)
int n,numofobs;

/* n : the maximum # of forecasting steps for yearly, */
/* quarterly and monthly time series (i.e. 6,8,18) */

char sernum[7];
float actual[];
int t;
{
float normalization(),xroot;
float compare();
float obs[50][50],H_net[50],O_net[50],product[100];
float H_OUT[50],O_OUT[50];
float error,m_err[30];
float actualval[50];
float forecast[50][50];
float sum_err,sum,average;
float modfactor;
int c=0,o,i,j,h;
int ind,begin,end;

printf("----Testing for time series : %s\n",sernum);
end = numofobs - n;
begin = end - I;
sum = 0.0;
/*
for(i=end-n;i<end;i++)
  sum += actual[i];
average = sum / n;

```

```

modfactor = actual[--i]/average;
*/
sum = actual[end-1] - actual[end-n];
modfactor = sum / (n-1);

for(i=0;i<50;i++)
  actualval[i] = 0.0;
for(i=0;i<n;i++)
  actualval[i] = actual[i+end];
for(i=0;i<n;i++)
  for(j=0;j<I;j++) {
    obs[i][j] = actual[j+begin+i];
  };

while (c < n) {
  xroot = normalization(obs[c],I);

/* the xroot=(maxobs - minobs) of the observations is saved for
denormalization purpose when comes to the testing process*/

/* operations between Input and Hidden layers */

for(h=0;h<H;h++) {
  for(i=0;i<=I;i++) {
    if (i == I)
      product[i] = bias * w[i][h];
    else product[i] = obs[c][i] * w[i][h];
  };
  H_net[h] = 0.0;
for(i=0;i<=I;i++)
  H_net[h] = H_net[h] + product[i];
  H_OUT[h] = activation(H_net[h]);
};

/* operations between Hidden and Output layers */

for(o=0;o<O;o++) {
  for(h=0;h<=H;h++) {
    if (h == H)
      product[h] = bias * v[h][o];
    else product[h] = H_OUT[h] * v[h][o];
  };
  O_net[o] = 0.0;
for(h=0;h<=H;h++)
  O_net[o] = O_net[o] + product[h];
  O_OUT[o] = activation(O_net[o]);

  if (xroot == 0.0)
    forecast[c][o] = MIN;
  else forecast[c][o] = (O_OUT[o]*xroot+MIN)+modfactor;

if ((c > n-N) && (o >= n-c))

```

```

        forecast[c][o] = 0.0;
/*
printf("output[%d] = %lf\n",o,O_OUT[o]);
printf("forecast[%d] = %lf\n",o,forecast[c][o]);
printf("-----\n");
*/
};
/*printf("error[%d] = %5.2f\n",c,error[c]);*/
c++;
};

compute_err(forecast,actualval,c);

printf("The %d_step ahead forecasting MAPE \n",N);
for(i=0;i<N;i++) {
    if ((i % 10) == 0 && i != 0)
        printf("\n Y+%d ",i+1);
    else printf(" Y+%d ",i+1);
};
printf("\n");
for(i=0;i<N;i++) {
    if ((i % 10) == 0 && i != 0)
        printf("\n-----");
    else printf("-----");
};
printf("\n");
sum_err = 0.0;
for(j=0;j<N;j++) {
    error = 0.0;
    for(i=0;i<c;i++) {
        error = error + err_set[i][j];
    };
    mape[j][t] = error / (c-j);
    if ((j % 10) == 0 && j != 0)
        printf("\n%5.2f ",mape[j][t]);
    else printf("%5.2f ",mape[j][t]);
    sum_err = 0.0;
};
printf("\n");
/*printf("sum_err = %f\n",sum_err);*/
printf("The %d_step ahead forecasting Median-APE \n",N);
for(i=0;i<N;i++) {
    if ((i % 10) == 0 && i != 0)
        printf("\n Y+%d ",i+1);
    else printf(" Y+%d ",i+1);
};
printf("\n");
for(i=0;i<N;i++) {
    if ((i % 10) == 0 && i != 0)
        printf("\n-----");
    else printf("-----");
};
};

```

```

printf("\n");
for(j=0;j<N;j++) {
    for(i=0;i<c-j;i++) {
        m_err[i] = err_set[i][j];
    };
    m_mape[j][t] = compare(m_err,c-j);
    if ((j % 10) == 0 && j != 0)
        printf("\n%5.2f ",m_mape[j][t]);
    else printf("%5.2f ",m_mape[j][t]);
};
printf("\n");
printf("The testing of BPN network for time series %s is
finished.\n",senum);
}

```

```

/*-----*/
/* This is the function to calculate the Mean Absolute */
/* Percent Error for each time series of N_step ahead */
/* forecasting. */
compute_err(fore,act,c)
float fore[50][50],act[];
int c;
{
    float sum_err=0.0;
    float diff;
    int i,j;

    for(i=0;i<N;i++)
        for(j=0;j<c;j++) {
            diff = act[j+1] - fore[j][i];
            if (diff < 0.0) diff = fore[j][i] - act[j+1];
            if (diff == 0.0) err_set[j][i] = 0.0;
            else err_set[j][i] = diff * 100 / act[j+1];
/* printf("e[%d]= %5.2f ",i,err_set[c][i]);*/
        };
/*printf("\n");*/
}

```

```

/*-----*/
/* This function decides the output value of each hidden */
/* neuron through the activation function in the neurons */
/* of hidden layer. */
float activation(net)
float net;
{
    float out;
    /*
    if (net > 92.0)
        return(0.849999);
    */
}

```

```

else if (net < -92.0)
    return(-0.150001);
else {
    out = 1 + 1 / exp(net);
    return(1/out-0.15);
};
*/
out = 1 + 1 / exp(net);
return(1/out);
}

```

```

/*-----*/
/* The function calculates the mean of MAPEs for each */
/* forecasting step. */
MEAN(t)
int t;
{
float mean;
int i,j,c;
printf("\nThe following table is the list of the means of
MAPEs\n");
for(i=0;i<N;i++) {
    if ((i % 10) == 0 && i != 0)
        printf("\n M%d ",i+1);
    else printf(" M%d ",i+1);
};
printf("\n");
for(i=0;i<N;i++) {
    if ((i % 10) == 0 && i != 0)
        printf("\n-----");
    else printf("-----");
};
printf("\n");
for(i=0;i<N;i++) {
    mean = 0.0;
    c = 0;
    for(j=0;j<t;j++) {
        if (mape[i][j] != 0.0)
            mean = mean + mape[i][j];
        else c++;
    };
    j = j-c;
    mean = mean / j;
    if ((i % 10) == 0 && i != 0)
        printf("\n%5.2f ",mean);
    else printf("%5.2f ",mean);
};
printf("\n");

```

```

}

```

```

/*-----*/
/* This function calls the sorting function to find out */
/* the middle term of the Absolute Percent Errors. */
float compare(x,c)
float x[];
int c;
{
    sorting(x,0,c-1);
    if ((c % 2) == 0)
        return(x[(c-1)/2]);
    else return(x[c/2]);
}

```

```

/*-----*/
/* The function arranges the input data in order by calling */
/* itself recursively. */
sorting(node,left,right)
float node[];
int left,right;
{
    float x,y;
    int i,j;
    i = left;
    j = right;
    x = node[(left+right)/2];
    /* compare the middle term with the numbers of its right side and
left side */
do {
    while (node[i] < x && i < right) i++;

    /* if the # on the left side is greater than the middle term
then stop */
    while (x < node[j] && j > left) j--;

    /* if the # on the right side is smaller than the middle term
then stop */

    if (i <= j) { /* swap the two #s */
        y = node[i];
        node[i] = node[j];
        node[j] = y;
        i++; j--;
    };
}while (i <= j);

/* compare the #s on the left side of the middle term */

```

```

if (left < j)
    sorting(node,left,j);

/* compare the #s on the right side of the middle term */
if (i < right)
    sorting(node,i,right);
}

/*-----*/
/* This is the function to pick the middle term of Median */
/* Absolute Percent Errors for the time series of yearly, */
/* quarterly and monthly. */
M MEDIAN(t)
int t;
{
float median,mean;
float compare();
float temp[80];
int i,j,a,c;
printf("\nThe following table is the list of the medians of
M APEs'\n");
for(i=0;i<N;i++) {
    if ((i % 10) == 0 && i != 0)
        printf("\n M_m%d ",i+1);
    else printf("M_m%d ",i+1);
};
printf("\n");
for(i=0;i<N;i++) {
    if ((i % 10) == 0 && i != 0)
        printf("\n-----");
    else printf("-----");
};
printf("\n");
for(i=0;i<N;i++) {
    mean = 0.0;
    c = 0;
    for(j=0;j<t;j++) {
        if (m_mape[i][j] != 0.0)
            mean = mean + m_mape[i][j];
        else c++;
    };
    j = j-c;
    mean = mean / j;
    if ((i % 10) == 0 && i != 0)
        printf("\n%5.2f ",mean);
    else printf("%5.2f ",mean);
};
/*for(i=0;i<N;i++) {
    a = 0;
    for(j=0;j<t;j++)

```

```

        if (m_mape[i][j] != 0.0) {
            temp[a] = m_mape[i][j];
            a++;
        };
        call the compare() function to rearrange the Median APEs in
order
        and return the middle term
        median = compare(temp,a);
        if ((i % 10) == 0 && i != 0)
            printf("\n%5.2f ",median);
        else printf("%5.2f ",median);
    };*/
printf("\n");
}

```

APPENDIX K
COUNTERPROPAGATION PROGRAM

```

#include <stdio.h>
#include <math.h>
#define prime 200
/* prime the random number generator 200 times */

#define maxape 70
/* maxape is the maximum total # of apes */

#define Yser 20
#define Qser 23
#define Mser 68
/* There are 13, 30 and 68 time series for yearly, quarterly
and monthly data, respectively. */

#define Ymax 6
#define Qmax 8
#define Mmax 18
/* The maximum # of forecasting steps for yearly, quarterly
and monthly time series is 6, 8 and 18, respectively */

#define maxset 50
#define numerr 30
#define maxobs 150

#define UPbound 0.5
#define LOWbound 0.01
/* UPbound and LOWbound are the upper delimiter and lower
delimiter of weight vectors */

#define TRUE 1
#define FALSE 0

float w[100][300],v[300][100];
/* wx[],wy[] are the weight matrices connecting Input, Kohonen
layers */
/* vx[],vy[] are the weight matrices connecting Kohonen,
Grossberg layers */

float alpha=0.7,beta=0.1;
float A=0.3,B=0.3;
/*float alpha,beta,UPbound,LOWbound,A,B;*/
float denorm;

float err_set[50][30];
long seed = 1.0;
int I,K,G;
/* I,K,G are the # of neurons in Input, Kohonen, Grossberg
layers*/

int N; /* N_step ahead forecasting */
int Pass;

/* Pass is the number of times to select a certain training
pattern and adjust the weight matrices */

float mape[30][70],m_mape[30][70];

main()
{
float random();
float actual[200];
/* actual[] is the array of actual values for training and
testing */

float Alpha,Beta,AA,BB;
int time=1;
int numofobs;
/* 1 <= time <=13 for yearly , 14 <= time <= 43 for quarterly
44 <= time <= 111 for monthly */
/* numofobs is the # of observations in a time series */

int i,done=0,add;
int j;
char sernum[7];
FILE *fp,*fopen();
printf("\nPlease enter\n");
printf("the number of neurons for Input layer : ");
scanf("%d",&I);
printf("%d\n",I);
printf("the number of neurons for Kohonen layer : ");
scanf("%d",&K);
printf("%d\n",K);
printf("the number of neurons for Grossberg layer : ");
scanf("%d",&G);
printf("%d\n",G);
printf("the number of values to be forecast (N = 1,2,4,6,8,12,18)
: ");
scanf("%d",&N);
printf("%d\n",N);
/*
printf("the upperbound of weight vectors : ");
scanf("%f",&UPbound);
printf("%f\n",UPbound);
printf("the lowerbound of weight vectors : ");
scanf("%f",&LOWbound);
printf("%f\n",LOWbound);
printf("the training rate coefficient alpha : ");
scanf("%f",&Alpha);
printf("%f\n",Alpha);
printf("the training rate coefficient beta : ");
scanf("%f",&Beta);
printf("%f\n",Beta);
printf("the training rate coefficient A : ");
scanf("%f",&AA);
*/
}

```



```

printf("%f\n",AA);
printf("the training rate coefficient B : ");
scanf("%f",&BB);
printf("%f\n",BB);
*/
printf("the number of train passes : ");
scanf("%d",&Pass);
printf("%d\n",Pass);
printf("\n");
if (I != G || I <= N) {
    printf("The future values cannot be predicted!\n");
    exit(1);
};
printf("I = %d , K = %d , G = %d\n",I,K,G);
printf("%d_step ahead forecasting\n",N);

for(i=0;i<prime;i++) random();
fp = fopen("data","r");

/*alpha = Alpha; beta = Beta;
A = AA; B = BB;*/
for(i=0;i<N;i++)
    for(j=0;j<maxape;j++) {
        mape[i][j] = 0.0;
        m_mape[i][j] = 0.0;
    };
printf("\n***** Yearly time series forecasting
*****\n");
while (time <= Yser && !feof(fp)) {
    /* for yearly time series */
    fscanf(fp,"%s",sernum);
    fscanf(fp,"%d",&numofobs);
    for(i=0;i<numofobs;i++)
        fscanf(fp,"%f",&actual[i]);
    /*
    printf("\n%s %d\n",sernum,numofobs);
    if (numofobs < I || numofobs <= Ymax) {
        printf("There is no enough data points in the time
series : %s\n",sernum);
    }
    else if (N > Ymax) {
        printf("This %d step forecasting cannot be done ",N);
        printf("with time series %s\n",sernum);
    }
    else done = trainwork(numofobs-Ymax,numofobs,sernum,actual);
    if (done) {
        testwork(Ymax,numofobs,sernum,actual,time-1);
    };
    for(i=0;i<maxobs;i++)
        actual[i] = 0.0;
    for(i=0;i<maxset;i++)
        for(j=0;j<numerr;j++)

```

```

err set[i][j] = 0.0;
done = 0;
alpha = 0.7; beta = 0.1;
A = 0.3; B = 0.3;
*/
time++;
};
/*
if (N <= Ymax) {
    MEAN(Yser);
    M_MEDIAN(Yser);
};
*/
alpha = 0.7; beta = 0.1;
for(i=0;i<N;i++)
    for(j=0;j<maxape;j++) {
        mape[i][j] = 0.0;
        m_mape[i][j] = 0.0;
    };
time = 1;
printf("\n***** Quarterly time series forecasting
*****\n");
while (time <= Qser && !feof(fp)) {
    /* for quarterly time series */
    fscanf(fp,"%s",sernum);
    fscanf(fp,"%d",&numofobs);
    for(i=0;i<numofobs;i++)
        fscanf(fp,"%f",&actual[i]);
    /*
    printf("\n%s %d\n",sernum,numofobs);
    if (numofobs < I || numofobs <= Qmax) {
        printf("There is no enough data points in the time
series : %s\n",sernum);
    }
    else if (N > Qmax) {
        printf("This %d step forecasting cannot be done ",N);
        printf("with time series %s\n",sernum);
    }
    else done = trainwork(numofobs-Qmax,numofobs,sernum,actual);
    if (done) {
        testwork(Qmax,numofobs,sernum,actual,time-1);
    };
    for(i=0;i<maxobs;i++)
        actual[i] = 0.0;
    for(i=0;i<maxset;i++)
        for(j=0;j<numerr;j++)
            err set[i][j] = 0.0;
    done = 0;
    alpha = 0.7; beta = 0.1;
    A = 0.3; B = 0.3;
*/
time++;

```

```

};
/*
if (N <= Qmax) {
    MEAN(Qser);
    M_MEDIAN(Qser);
};
*/
alpha = 0.7; beta = 0.1;
for(i=0;i<N;i++)
    for(j=0;j<maxape;j++) {
        mape[i][j] = 0.0;
        m_mape[i][j] = 0.0;
    };
time = 1;
printf("\n***** Monthly time series forecasting
*****\n");
while (time <= Mser && !feof(fp)) {
    /* for monthly time series */
    fscanf(fp,"%s",sernum);
    fscanf(fp,"%d",&numofobs);
    printf("\n%s %d\n",sernum,numofobs);
    for(i=0;i<numofobs;i++)
        fscanf(fp,"%f",&actual[i]);
    if (numofobs < I || numofobs <= Mmax) {
        printf("There is no enough data points in the time
series : %s\n",sernum);
    }
    else if (N > Mmax) {
        printf("This %d_step forecasting cannot be done ",N);
        printf("with time series %s\n",sernum);
    }
    else done = trainwork(numofobs-Mmax,numofobs,sernum,actual);
    if (done) {
        testwork(Mmax,numofobs,sernum,actual,time-1);
    };
    for(i=0;i<maxobs;i++)
        actual[i] = 0.0;
    for(i=0;i<maxset;i++)
        for(j=0;j<numerr;j++)
            err_set[i][j] = 0.0;
    done = 0;
    alpha = 0.7; beta = 0.1;
    A = 0.3; B = 0.3;
    time++;
};

if (N <= Mmax) {
    MEAN(Mser);
    M_MEDIAN(Mser);
};

fclose(fp);

```

```

}

/*-----*/
/* This is the random number generator function from the */
/* paper below. Park and Miller, "Random number generators:*/
/* good ones are hard to find. Comm. ACM 31, 10(Oct. 1988),*/
/* 1192-1201. */
float random()
{
    float a=16807.0,m=2147483647.0,q=127773.0,r=2836.0;
    float lo,test;
    int hi;
    hi = (int)(seed/q); lo = seed - q*hi;
    test = a*lo - r*hi;
    if (test > 0.0) seed = test;
    else seed = test + m;
    return (seed/m);
}

/*-----*/
/* This is the function to train CPN network with the */
/* training sets from the time series data. */
trainwork(train,numofobs,sernum,actual)
int train,numofobs;
char sernum[7];
float actual[];
{
    float random(),normalization(),root[100];
    float obs[150][100]; /* obs[] is the array of observations */
    float set[150][100];
    float product[100],K_net[300],G_net[100];

    /* product is the product of observation and weight. K_net[] is
the net outputs for Kohonen neurons while G_net[] for Grossberg
neurons. */

    float act[100][50],diff,err[50],error,fore[50];
    int i,j,k,g,c=0;
    int ind,in,pass;
    int b,prev[150],flag;
    int count[300];

    /* count[] is an array of counters of winning times for each
Kohonen neuron */

    printf("----Training for time series : %s\n",sernum);
    train = train - I;
    if (train <= 1) {
        printf("There is no enough data points in the time series :
%s\n",sernum);
    }
}

```

```

    return(0);
};

for(i=0;i<=train;i++)
    for(j=0;j<I;j++) {
        obs[i][j] = actual[j+1];
    };

init_weight();
for(i=0;i<300;i++) count[i] = 0;

while (c <= train) {
    root[c] = normalization(obs[c]);

    /* the square root of sum of squares of the observations is saved
    for denormalization purpose when comes to the testing process */

    /* operations between Input and Kohonen layers */

    for(i=0;i<I;i++) {
        /* save the normalized observations for training */
        set[c][i] = obs[c][i];
    };
    for(k=0;k<K;k++) {
        for(i=0;i<I;i++) {
            product[i] = obs[c][i] * w[i][k];
        };
        K_net[k] = 0.0;
        for(i=0;i<I;i++)
            K_net[k] = K_net[k] + product[i];
    };
    j = decide_winner(K_net,count,train+1);
    b=0;

    /* operations between Kohonen and Grossberg layers */
    for(g=0;g<G;g++) {
        for(k=0;k<K;k++) {
            if (k == j)
                product[k] = v[k][g];
            else product[k] = 0.0;
        };
        G_net[g] = 0.0;
        for(k=0;k<K;k++)
            G_net[g] = G_net[g] + product[k];
        if (g >= G-N) {
            act[c][b] = actual[c+G-N+b];
            b++;
        };
    };

    adjust_weight(obs[c],j);
    /* int adjust(obs[c]); */

```

```

    c++;
};

b=0;
for(pass=1;pass<=Pass-c;pass++) {
    ind = (int)(random() * c);

    for(k=0;k<K;k++) {
        for(i=0;i<I;i++) {
            product[i] = set[ind][i] * w[i][k];
        };
        K_net[k] = 0.0;
        for(i=0;i<I;i++)
            K_net[k] = K_net[k] + product[i];
    };

    j = decide_winner(K_net,count,train+1);
    b=0;
    for(g=0;g<G;g++) {
        for(k=0;k<K;k++) {
            if (k == j)
                product[k] = v[k][g];
            else product[k] = 0.0;
        };
        G_net[g] = 0.0;
        for(k=0;k<K;k++)
            G_net[g] = G_net[g] + product[k];
        if (g >= G-N) {
            fore[b] = root[ind] * G_net[g];
            b++;
        };
    };

    for(i=0;i<N;i++) {
        diff = act[ind][i] - fore[i];
        if (diff < 0.0)
            diff = fore[i] - act[ind][i];
        err[i] = diff / act[ind][i];
    };
    error = 0.0;
    for(i=0;i<N;i++) error += err[i];
    error /= N;
    if (error <= 0.000001)
        return(1);
    adjust_weight(set[ind],j);
};

printf("The training of CPN network for time series %s is
finished.\n",sernum);
return(1);
}

```

```

/*-----*/
/* This is the function to initialize the connecting weight */
/* vectors in the CPN network. */
init_weight()
{
float random(),delimiter=1.0;
/* delimiters are the bounds for weights */
float sqr_sum=0.0;
int i,k,g;
int ind;

/* initialize the weights between Input and Kohonen layers */
for(k=0;k<K;k++) {
for(i=0;i<I;i++) {
while (delimiter > UPbound || delimiter < LOWbound)
delimiter = random();
w[i][k] = delimiter;
delimiter = 1.0;
};

/* normalization of the weights between Input and Kohonen
layers */
for(i=0;i<I;i++)
sqr_sum = sqr_sum + w[i][k]*w[i][k];
for(i=0;i<I;i++)
w[i][k] = w[i][k] / sqrt(sqr_sum);
};

/* initialize the weights between Kohonen and Grossberg
layers */
for(k=0;k<K;k++) {
for(g=0;g<G;g++) {
while (delimiter > UPbound || delimiter < LOWbound)
delimiter = random();
v[k][g] = delimiter;
delimiter = 1.0;
};

/* normalization of the weights between Kohonen and Grossberg
layers */
sqr_sum = 0.0;
for(g=0;g<G;g++)
sqr_sum = sqr_sum + v[k][g]*v[k][g];
denorm = sqrt(sqr_sum);
for(g=0;g<G;g++)
v[k][g] = v[k][g] / denorm;
};
}

/*-----*/
/* The function adjusts the weight vectors in the CPN */

```

```

/* networks according to the methods from the papers of */
/* Hecht-Nielsen[1987a]. */
adjust_weight(x,j)
float x[];
int j;
{
int i,k,g;

/* adjust the weights connecting Input and Kohonen layers */

for(i=0;i<I-N;i++) {
w[i][j] = w[i][j] + alpha*(x[i] - w[i][j]);
};
for(i=I-N;i<I;i++) {
w[i][j] = w[i][j] + beta*(x[i] - w[i][j]);
};
alpha = alpha / 1.005;
beta = beta / 1.005;

/* adjust the weights connecting Kohonen and Grossberg layers */
for(g=0;g<G-N;g++) {
v[j][g] = v[j][g] + (B*x[g] - A*v[j][g]);
};
for(g=G-N;g<G;g++) {
v[j][g] = v[j][g] + (B*x[g] - A*v[j][g]);
};
/*A = A / 1.001;
B = B / 1.001;*/
}

/*-----*/
/* This function normalizes the input vectors and weight */
/* vectors. */
float normalization(x)
float x[];
{
int i;
float sqr_sum = 0.0;
for(i=0;i<I;i++)
sqr_sum = sqr_sum + x[i]*x[i];
for(i=0;i<I;i++)
x[i] = x[i] / sqrt(sqr_sum);
return(sqrt(sqr_sum));
}

/*-----*/
/* This is the function to test the trained CPN network */
/* model with the remaining k data points (observations) */
/* of a certain time series and evaluate its forecasting */
/* ability. */

```

```

testwork(n,numofobs,sernum,actual,t)
int n,numofobs;
char sernum[7];
float actual[];
int t;
{
float normalization(),xroot;
float compare();
float obs[50][50],K_net[300],G_net[100],product[100];
float error,m_err[30];
float actualval[50];
float winner=0.0,forecast[50][50];
float sum_err,sum,average,modfactor;
float mod;
int c=0,g,i,j,k;
int a,ind,begin,end;
int f;

printf("----Testing for time series : %s\n",sernum);
end = numofobs - n;
begin = end - (I-N);
sum = 0.0;
sum = actual[end-1] - actual[end-n];
modfactor = sum / (n-1);

for(i=0;i<50;i++)
  actualval[i] = 0.0;
for(i=0;i<n;i++)
  actualval[i] = actual[i+end];
for(i=0;i<n;i++)
  for(j=0;j<I-N;j++) {
    obs[i][j] = actual[j+begin+i];
  };
a = j;
while (c < n) {
  f = 0;
  for(ind=0;ind<N;ind++)
    obs[c][ind+a] = 0.0;
  xroot = normalization(obs[c]);

  /* the square root of sum of squares of the observations is saved
  for denormalization purpose in the testing procedure */

  /* operations between Input and Kohonen layers */
  for(k=0;k<K;k++) {
    for(i=0;i<I;i++)
      product[i] = obs[c][i] * w[i][k];
    K_net[k] = 0.0;
    for(i=0;i<I;i++)
      K_net[k] = K_net[k] + product[i];
  };

```

```

for(k=0;k<K;k++)
  if (winner <= K_net[k]) {
    winner = K_net[k];
    j = k;
  };
winner = 0.0;
xroot = xroot * obs[c][I-N-1];
/* operations between Kohonen and Grossberg layers */

for(g=0;g<G;g++) {
  for(k=0;k<K;k++) {
    if (k == j)
      product[k] = v[k][g];
    else product[k] = 0.0;
  };
  G_net[g] = 0.0;
  for(k=0;k<K;k++)
    G_net[g] = G_net[g] + product[k];
  if (g == (G-N-1))
    xroot /= G_net[g];
  if (g >= G-N) {
    /*
    forecast[c][f] = modfactor + G_net[g] * xroot;
    */
    if (N >= I-N)
      forecast[c][f] = G_net[g] * xroot;
    else forecast[c][f] = G_net[g] * xroot;
    if ((c > n-N) && (f >= n-c))
      forecast[c][f] = 0.0;
    /*
    printf("output[%d] = %f\n",g,G_net[g]);
    printf("forecast[%d] = %f\n",f,forecast[c][f]);
    */
    f++;
  };
  c++;
};

compute_err(forecast,actualval,c);

printf("The %d step ahead forecasting MAPE \n",N);
for(i=0;i<N;i++) {
  if ((i % 10) == 0 && i != 0)
    printf("\n Y+%d ",i+1);
  else printf(" Y+%d ",i+1);
};
printf("\n");
for(i=0;i<N;i++) {
  if ((i % 10) == 0 && i != 0)

```

```

        printf("\n-----");
        else printf("-----");
    };
    printf("\n");
    sum_err = 0.0;
    for(j=0;j<N;j++) {
        error = 0.0;
        for(i=0;i<c;i++) {
            error = error + err_set[i][j];
        };
        mape[j][t] = error / (c-j);

        if ((j % 10) == 0 && j != 0)
            printf("\n%5.2f ", mape[j][t]);
        else printf("%5.2f ", mape[j][t]);

        sum_err = 0.0;
    };
    printf("\n");

    printf("The %d step ahead forecasting Median-APE \n",N);
    for(i=0;i<N;i++) {
        if ((i % 10) == 0 && i != 0)
            printf("\n Y+%d ",i+1);
        else printf(" Y+%d ",i+1);
    };
    printf("\n");
    for(i=0;i<N;i++) {
        if ((i % 10) == 0 && i != 0)
            printf("\n-----");
        else printf("-----");
    };
    printf("\n");
    for(j=0;j<N;j++) {
        for(i=0;i<c-j;i++) {
            m_err[i] = err_set[i][j];

            /* assign err_set[][] to a one-dimension array m_err[] */
            /* for the convenience of comparison of errors */

        };
        m_mape[j][t] = compare(m_err,c-j);

        if ((j % 10) == 0 && j != 0)
            printf("\n%5.2f ", m_mape[j][t]);
        else printf("%5.2f ", m_mape[j][t]);

    };
    printf("\n");
    printf("The testing of CPN network for time series %s is
    finished.\n",sernum);

```

```

/*-----*/
/* This is the function to calculate the Mean Absolute */
/* Percent Error for each time series of N_step ahead */
/* forecasting. */
compute_err(fore,act,c)
float fore[50][50],act[];
int c;
{
    float sum_err=0.0;
    float diff;
    int i,j;

    for(i=0;i<N;i++)
        for(j=0;j<c;j++) {
            diff = act[j+1] - fore[j][i];
            if (diff < 0.0) diff = fore[j][i] - act[j+1];
            if (diff == 0.0) err_set[j][i] = 0.0;
            else err_set[j][i] = diff * 100 / act[j+1];
        };
}

/*-----*/
/* This function decides the winner neuron of Kohonen */
/* layer by winner-take-all rule. */
decide_winner(K_net,count,c)
float K_net[];
int count[];
int c;
/* count[] is an array of counters of winning times */
/* for each Kohonen neuron */
{
    float winner = 0.0;
    int j,k;

    /* if a certain Kohonen neuron has won 1/K times of the */
    /* total number of trainings then it will be held and give */ /*
    /* chance to the second winner to adjust the weight vectors */

    for(k=0;k<K;k++)
        if ((winner <= K_net[k]) && (count[k] <= ((Pass+c)/K + 1))) {
            winner = K_net[k];
            j = k;
        };
    count[j]++;
    return(j);
}

```

```

/*-----*/
/* The function calculates the mean of MAPEs for each */
/* forecasting step. */
MEAN(t)
int t;
{
float mean;
int i,j,c;
printf("\nThe following table is the list of the means of
MAPEs'\n");
for(i=0;i<N;i++) {
if ((i % 10) == 0 && i != 0)
printf("\n M%d ",i+1);
else printf(" M%d ",i+1);
};
printf("\n");
for(i=0;i<N;i++) {
if ((i % 10) == 0 && i != 0)
printf("\n-----");
else printf("-----");
};
printf("\n");
for(i=0;i<N;i++) {
mean = 0.0;
c = 0;
for(j=0;j<t;j++) {
if (mape[i][j] != 0.0)
mean = mean + mape[i][j];
else c++;
};
j = j-c;
mean = mean / j;
if ((i % 10) == 0 && i != 0)
printf("\n%5.2f ",mean);
else printf("%5.2f ",mean);
};
printf("\n");
}

/*-----*/
/* This function calls the sorting function to find out */
/* the middle term of the Absolute Percent Errors. */
float compare(x,c)
float x[];
int c;
{
sorting(x,0,c-1);
if ((c % 2) == 0)
return(x[(c-1)/2]);
else return(x[c/2]);
}

```

```

}

/*-----*/
/* The function arranges the input data in order by */
/* calling itself recursively. */
sorting(node,left,right)
float node[];
int left,right;
{
float x,y;
int i,j;
i = left;
j = right;
x = node[(left+right)/2];

/* compare the middle term with the numbers of
its right side and left side */

do {
while (node[i] < x && i < right) i++;

/* if the # on the left side is greater than the middle term
then stop */

while (x < node[j] && j > left) j--;

/* if the # on the right side is smaller than the middle term
then stop */

if (i <= j) { /* swap the two #s */
y = node[i];
node[i] = node[j];
node[j] = y;
i++; j--;
};
}while (i <= j);

/* compare the #s on the left side of the middle term */

if (left < j)
sorting(node,left,j);

/* compare the #s on the right side of the middle term */

if (i < right)
sorting(node,i,right);
}

/*-----*/
/* This is the function to calculate the mean of Median */

```

```

/* Absolute Percent Errors for the time series of yearly, */
/* quarterly and monthly. */
M_MEDIAN(t)
int t;
{
float median,mean;
float compare();
float temp[80];
int i,j,a,c;
printf("\n\nThe following table is the list of the medians of
M_APEs'\n");
for(i=0;i<N;i++) {
if ((i % 10) == 0 && i != 0)
printf("\n M_m%d ",i+1);
else printf(" M_m%d ",i+1);
};
printf("\n\n");
for(i=0;i<N;i++) {
if ((i % 10) == 0 && i != 0)
printf("\n-----");
else printf("-----");
};
printf("\n\n");
for(i=0;i<N;i++) {
mean = 0.0;
c = 0;
for(j=0;j<t;j++) {
if (m_mape[i][j] != 0.0)
mean = mean + m_mape[i][j];
else c++;
};
j = j-c;
mean = mean / j;
if ((i % 10) == 0 && i != 0)
printf("\n%5.2f ",mean);
else printf("%5.2f ",mean);
};
printf("\n\n");
}

/*-----*/
init_adjust(x)
float x[];
{
int i,k,g;
/* adjust the weights connecting Input and Kohonen layers */
for(i=0;i<I-N;i++) {
for(k=0;k<K;k++)
w[i][k] = w[i][k] + alpha*(x[i] - w[i][k]);
};
}

```

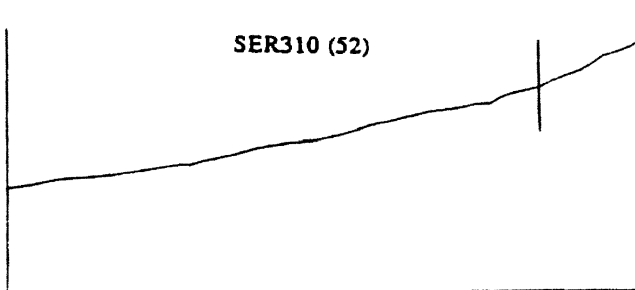
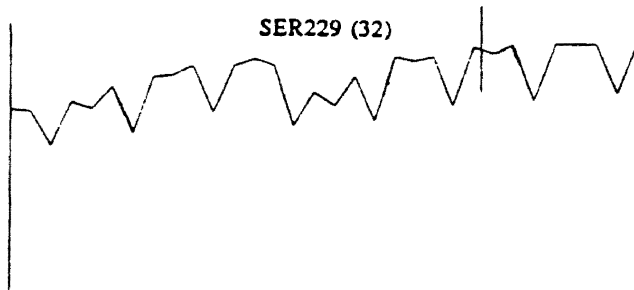
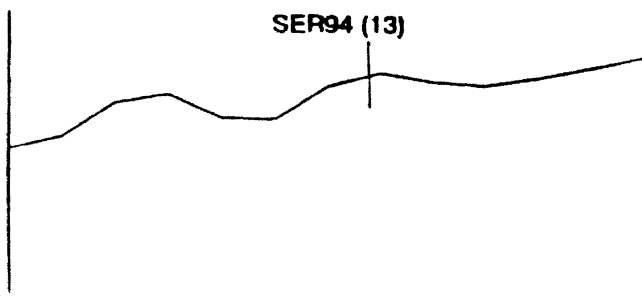
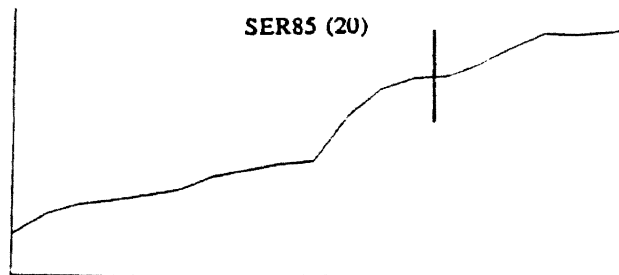
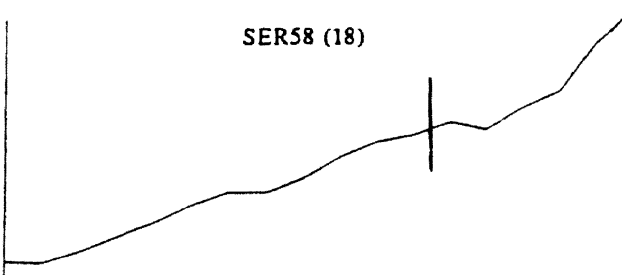
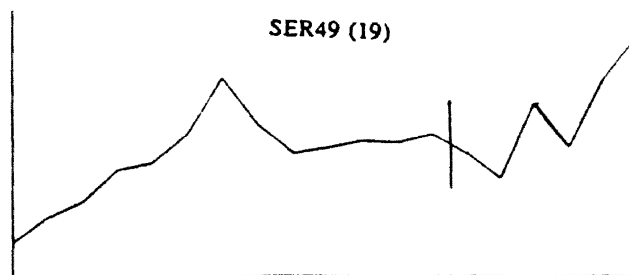
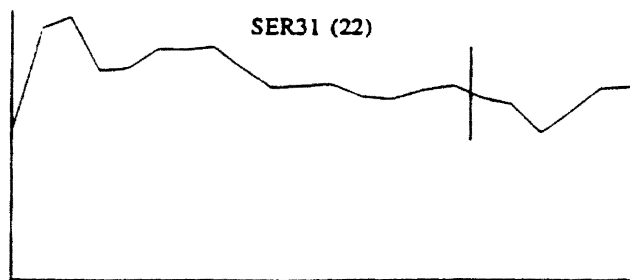
```

for(i=I-N;i<I;i++) {
for(k=0;k<K;k++)
w[i][k] = w[i][k] + beta*(x[i] - w[i][k]);
};
alpha = alpha / 1.005;
beta = beta / 1.005;

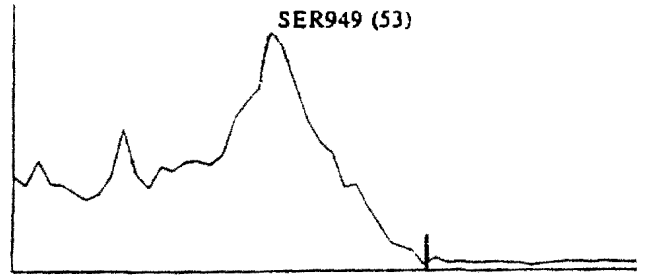
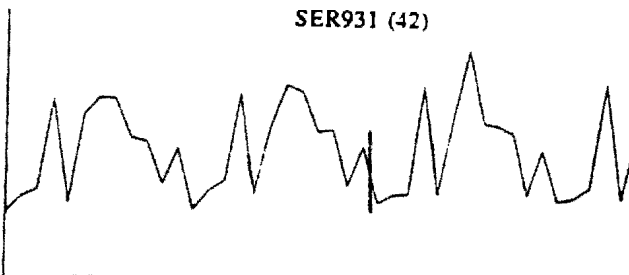
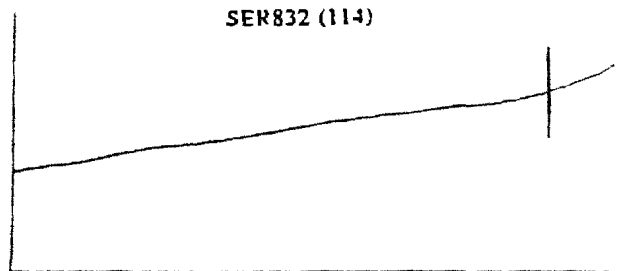
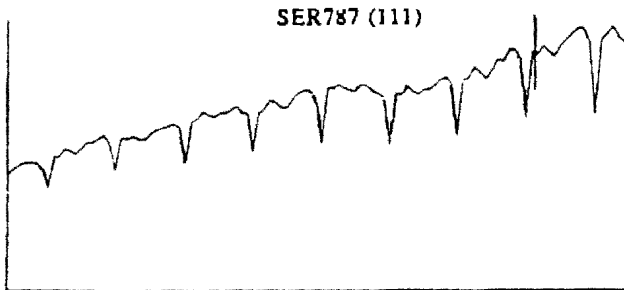
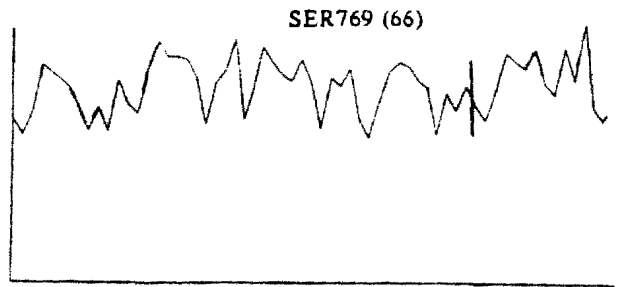
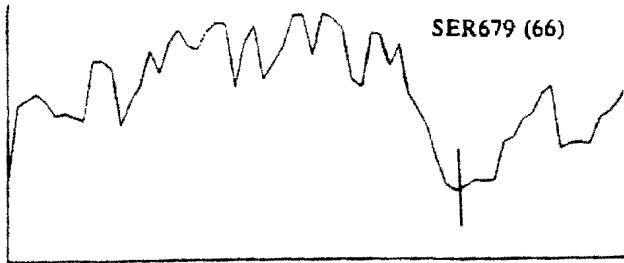
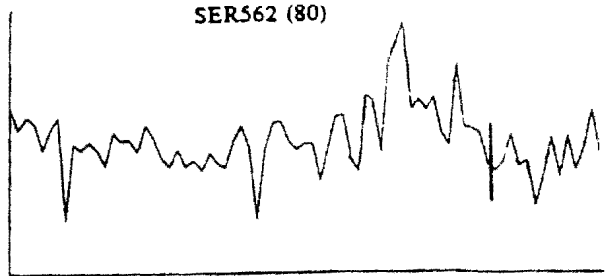
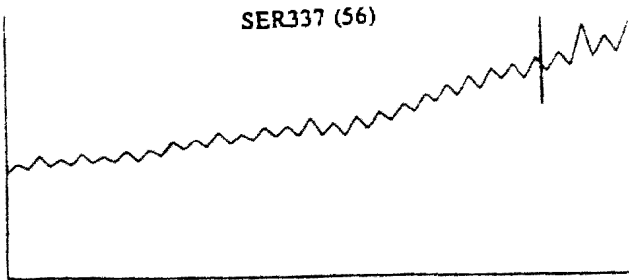
/* adjust the weights connecting Kohonen and Grossberg layers */
for(g=0;g<G-N;g++) {
for(k=0;k<K;k++)
v[k][g] = v[k][g] + (B*x[g] - A*v[k][g]);
};
for(g=G-N;g<G;g++) {
for(k=0;k<K;k++)
v[k][g] = v[k][g] + (B*x[g] - A*v[k][g]);
};
/*A = A / 1.001;
B = B / 1.001;*/
}

```


APPENDIX L
FIGURES OF SOME M-111 TIME SERIES



The short vertical line on each graph divides the time series into two parts: the left part is the training set while the right part is the testing set.



VITA

Chen-chou Lin

Candidate for the degree of

Master of Science

Thesis: USING NEURAL NETWORKS TO DO TIME SERIES FORECASTING

Major Field: Computer Science

Biographical:

Personal Data: Born in Pingtung, Taiwan, R. O. C.,
February 7, 1964, the son of Mong-shan Lin and
Shunn-huey Shieh.

Education: Graduated from Chen Jen High School,
Kaohsiung, Taiwan, R. O. C. in June 1981; received
Bachelor of Science degree from National Central
University in June 1986; completed the
requirements for the Master of Science degree at
Oklahoma State University in May, 1993.