1. INTRODUCTION

1.1 Forecasting

Forecasting is an activity to calculate or predict some future event or condition, usually as a result of rational study or analysis of pertinent data. Forecasting is widely used today in many fields, especially in industry, marketing, economy and finance. Such as in consumable product manufacturing, an accurate prediction of the future demand is very helpful in providing precise inventory, reducing transportation costs, then increasing profit (Markridakis, 1996).

Forecast methods may be broadly classified into qualitative and quantitative techniques. Qualitative methods are intuitive, largely educated guesses that may or may not depend on the past data. Quantitative methods use mathematical or statistical models to generate a reasonable prediction from the information of the past. Compared to qualitative methods, quantitative methods have the advantage of being supported by mathematical and statistical theory, and can be fully reproduced by any forecaster.

In general forecasting, especially time series forecasting, a primary type of data in business and economics, the quantitative methods are widely applied. A time series is a set of observations $\{y_t : t = 1, 2, \cdots, T\}$. Usually, time series is considered as discrete series which observations are recorded at predetermined, equal-interval time point such as hourly, daily, monthly, quarterly or yearly.

There are many quantitative forecast methods available today. In the M3-Competition (Makridakis 2002), the examined 24 methods are classified into six categories, which are naïve/simple, explicit trend models, decomposition, Autoregressive and Moving Average (ARIMA), expert system, and neural networks. Basically, the naïve/simple and explicit trend models are considered as simple forecasting methods while the ARIMA and neural networks are defined as statistically sophisticated and mathematically complex methods. In this paper, we explore these statistically sophisticated methods which are Dynamic Linear Model (DLM), ARIMA Model and Back Propagation Neural Networks (ANNs). .

1.2 The M3 Competition and Discussion

Reid (1969, 1972), Newbold and Granger (1974) published the first major papers regarding the forecasting method evaluation via a competition paradigm. These studies compared a large number of common time series with a limited number of paradigms to determine their post-sample forecasting accuracy. Makridakis and Hibon (1979) brought the forecasting competition to open debate with their paper. In this paper, they first compared a large number of quantitative forecasting methods across multiple time series. Since that time, many additional and larger studies have appeared, including the M-Competition (Makridakis et al., 1982), the M2-Competition (Makridakis et al., 1993), to determine which forecasting paradigm outperform others. Following the M and M2 trials, Spyros Makridakis and Michele Hibon presented their third forecasting study known as the M3-Competition at 1997. The M3-Competition utilizes a common database, which

contains 3003 mostly business and economic time series. An open invitation was given to all researchers willing to generate forecasts for all series. Their forecast results were then compiled and evaluated with various accuracy methods using a holdout sample observations. The purpose of the M3-Competition was to evaluate four hypotheses-the conclusions of the M and M2 competitions. Makridakis and Hibon (2000) concluded that the result of the M3-Competition confirmed the original conclusions of the last two M-Competition (Makridakis, 1982). The four confirmed conclusions are: (1) Statistically sophisticated or complex methods do not necessarily produce more accurate forecasts than simpler ones; (2) The rankings of the performance of the various methods vary according to the accuracy measure being used; (3) The accuracy of the combination of various methods outperforms, on average, the specific methods being combined and does well in comparison with other methods; (4) The performance of the various methods depends upon the length of the forecasting horizon.

The M3 project involved a large number of forecast paradigms in an attempt to be comprehensive. However, due to the fact they were limited in resources and relied on external researchers to provide their analysis of the series using researcher chosen paradigms, some paradigms were omitted. Many researchers chose to use commercially available implementations of various paradigms instead of standard textbook methods. For instance, there are a variety of designs and learning techniques available for forecaster to choose in the Neural Network paradigm. But in M-3 Competition, the Automated Artificial Neural Network was the only type of Artificial Neural Network paradigm that was involved. Moreover, neither the network architecture nor the training

algorithm of this Automated Artificial Neural Network was mentioned in the paper. In addition, some paradigms such as Dynamic Linear Model (DLM) were not included in M-3 Competition. In our research, we repeat the M-3 Competition among three types of paradigms: ARIMA, DLM, and ANNs and detail them. Some up-to-date technologies and different algorithms of these three paradigms will be employed in an attempt to improve the forecasting accuracy. One big discussion of the M-3 Competition conclusions is in the forecasting accuracy evaluation. To decide whether one method is better than the others, comparing only the average of the values of one accuracy measures is not convincible (Stekler, 2001). Hence we will apply standard statistical methodology, Mixed linear model, to identify the difference among different paradigms for various forecast horizons.

1.3 The M3 Data

The M3-Competition consists of 3003 series, which includes various types of time series data (micro, industry, macro, etc.) and different seasonal characters (yearly, quarterly, etc.). Table 1 shows the classification of the 3003 series. The yearly and monthly data contain time series from all catalogs which indicate that the forecast range of these two classifications are wider than the other two (Makridakis, 2000). The result of the M3-Competition shows that extending the application region of a specific forecast paradigm may decrease the forecast accuracy. Table 2 shows the detail of all the seasonal catalogs. The quarterly and the other data share the same forecast horizon as eight while the monthly need to be forecasted eighteen horizons ahead. Mostly, the short term

forecasting is more precise than the long term forecasting, such that the forecast result of the monthly data are supposed to be worse than the other three. The data length decides how much past information can be used to forecast ahead. The yearly data have a short average data length which looks much worse than the other data that has a long average data length but only need forecast eight horizons ahead. The seasonal periodic inside the quarterly and the monthly data provide more information for modeling, such that a seasonal adjustment should be considered and will help the model catch the real pattern.

Time interval between successive observations	Types of time series data						
	Micro	Industry	Macro	Finance	Demogr aphic	Other	Total
Yearly	146	102	83	58	245	11	645
Quarterly	204	83	336	76	57		756
Monthly	474	334	312	145	111	52	1428
Other	4			29		141	174
Total	828	519	731	308	413	204	3003

Table 1. The classification of the 3003 time series.

Seasonal Type	Data detail					
	Total Series	Min Length	Median Length	Max Length	Average Length	Forecast Horizon
Yearly	645	14	19	41	22	6
Quarterly	756	16	44	64	41	8
Monthly	1428	48	115	126	99	18
Other	174	60	63	96	69	8

Table 2. Data detail of all the seasonal catalogs.

1.4 Forecasting Approach

The forecasting process is an error-driven iterative approach consisting of four distinct phases: collect data for forecasting; identify a possible forecast model; estimate

parameters in tentative forecast model; and diagnostic checking (Figure 1). After the data was collected and the forecasting question was specified, a quick glance at the data structure and pattern characters should be applied to identify a possible suitable model. Once a model is identified, the chosen model is then diagnostically checked against the historical data to determine if it accurately describes the time series. For instance, in the ARIMA model, the diagnostic involves checking the residuals between the forecast and actual series and determine if they are small, randomly distributed, and uncorrelated, if so the chosen ARIMA model is said to be a good fit. However, if the chosen model is not satisfactory, the process will move backward to the identify stage and repeated with another model to replace the original one. This process is iterated until a satisfactory model is found.

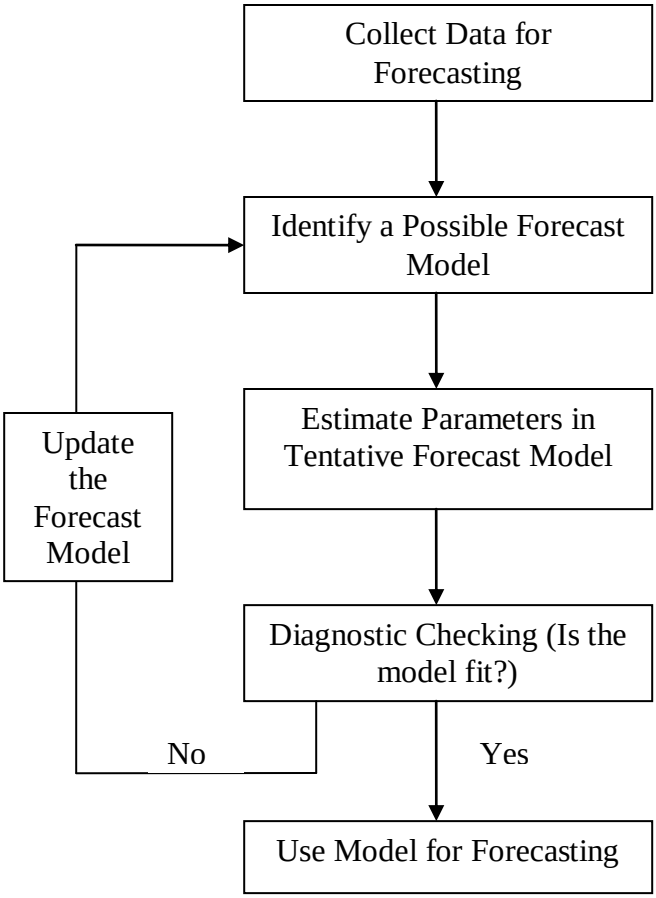

Figure 1. Forecasting Approach

1.5 Forecast Accuracy Measures

One of the conclusions confirmed by the M3-Competition is that the ranking of the performance of the various methods vary according to the accuracy measure being used. Regarding the accuracy measures used to evaluate which method gives the most accurate forecast; statisticians have given out a lot of heuristic comments. Koehler (2001) detected the asymmetry of the symmetry measures used in the M3-Competition and suggested that bounds on the forecast errors should be applied to evaluate how statistical

accurate the forecast is believed to be. Stekler (2001) indicated the necessity of performing statistical tests to determine whether there is any significant difference in the accuracy of the different forecasting methods. Following Stekler's suggestion, Koning et al (2004) finished research in using statistical multiple-comparison to test the significance among the results of various forecasting methods. They discovered that there are significant differences among the results obtained from the various accuracy measures that have been used in the M3-Competition.

We agree that rigorous statistical tests are necessary in the evaluation of the forecast results. In this paper, to make a valuable comparison between our forecasting results and the conclusions of the M3-Competition, we only employed one accuracy measures which was used in the M3-Competition: Symmetric mean absolute percentage error (SMAPE). Then, we used single mixed linear model to identify the differences among all the forecasting results in every forecasting horizon.

The SMAPE is defined as:

$$\frac{1}{n}\sum_{Series}\frac{|X-F|}{(X+F)/2}\times 100 \qquad 1.1$$

Where X is the real value and F is the forecast value, n is the number of the time series. The SMAPE is the average across all forecasts made for a given horizon in a specific type of time series data. Makridakis and Hibon (2000) considered that the SMAPE could help to avoid the problem of large errors when the actual values, X , are close to zero and the large difference between the absolute percentage errors when X is greater than F and vise versa. But actually, this SMAPE is not absolute symmetrical-it penalizes low forecasts more than high forecasts (Koehler, 2001). In the M3-Competition,

all time series data are strictly positive. To avoid the problem in the various SMAPE measures, a test was done on all the forecasted values, and all the negative value was substituted by zero to give a SMAPE as 200 (Makridakis, 2000).

2. THE OVERVIEW OF ARIMA

One mathematical approach to forecasting time series is known as the Box-Jenkins method and was suggested by Box and Jenkins (1970). Technically, the Box-Jenkins technique is an integration of the autoregressive and the moving average methods, so it is also named ARIMA (Autoregressive, Integrated, Moving Average) model. Since its first introduction, this ARIMA approach has become widely used in many fields such as specification, estimation, and diagnostic (Thomas 1983).

The ARIMA methodology is a statistical method for analyzing and building a forecasting model which best represents a time series by modeling the correlations in the data. In the empirical research, many advantages of the ARIMA model were found and support the ARIMA as a proper way in especially short term time series forecasting (Box, 1970; Jarrett, 1991). Taking advantage of its strictly statistical approach, the ARIMA method only requires the prior data of a time series to generalize the forecast. Hence, the ARIMA method can increase the forecast accuracy while keeping the number of parameters to a minimum. Some major disadvantages of ARIMA forecasting are: first, some of the traditional model identification techniques for identifying the correct model from the class of possible models are difficult to understand and usually computationally

expensive. This process is also subjective and the reliability of the chosen model can depend on the skill and experience of the forecaster. Second, the underlying theoretical model and structural relationships are not distinct as some simple forecasts models such as simple exponential smoothing and Holt-Winters (Thomas 1983). Moreover, the ARIMA models, as all forecasting methods, are essentially 'backward looking'. Such that, the long term forecast eventually goes to be straight line and poor at predicting series with turning points. In the next chapter, we briefly review the Autoregressive model and the moving average model, and then move foreword to ARIMA model.

2.1 Autoregressive Model

Autoregressive model are based on the assumption that each value of the time series Y_t depends only on the weighed sum of the product of the previous values $Y_{t-1}, Y_{t-2}, \cdots, Y_{t-p}$ and the regression coefficient $\phi_0, \phi_1, \phi_2, \cdots, \phi_p$ plus residual ε_t. An autoregressive model can be considered as a p-order autoregressive model, which takes the following form:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t \qquad 2.1$$

where Y_t is value of the series at time *t*, $Y_{t-1}, Y_{t-2}, \cdots, Y_{t-p}$ are dependent on the previous values of the variable at specified time periods, $\phi_0, \phi_1, \phi_2, \cdots, \phi_p$ are the regression coefficients and ε_t is the residual term that represents random events not explained by model.

The Autoregressive model is capable in a wide variety of time series forecasting by adjusting the regression coefficients ϕ_p. The difference between the Autoregressive models and other conventional regression models is with respect to the assumption of the independence of the error term. Since the independent variables are time-lagged values for the dependent variable, the assumption of uncorrelated error is easily violated.

2.2 Moving-Average Models

The basic idea of Moving-Average model is firstly finding the mean for a specified set of values and then using it to forecast the next period and correcting for any mistakes made in the last few forecasts. It takes this form:

$$Y_t = w_0 + \varepsilon_t - w_1\varepsilon_{t-1} - w_2\varepsilon_{t-2} - \ldots - w_q\varepsilon_{t-q} \qquad 2.2$$

where Y_t is the value of the series at time t, $w_0, w_1, w_2, \cdots, w_q$ are the weights applied to $\varepsilon_{t-1}, \varepsilon_{t-2}, \cdots, \varepsilon_{t-q}$ previous forecast errors and ε_t is the residual error.

To specify a Moving-Average, the number and the value of the q moving average parameter w_1 through w_q have to be decided subject to the certain restrictions in value in order for the process to be stationary. The Moving-Average model works well with stationary data, a type of time series without trend or seasonality.

2.3 ARIMA Models

The AR and MA model can be mixed and, provide a third class of general models called ARMA, a particular $ARIMA(p,0,q)$ model. With non-seasonal differences d added to the model, the $ARIMA(p,d,q)$ model has the capability to handle the variety kind of time series forecasting questions. Here p is the number of autoregressive terms, d is the number of non-seasonal differences, and q is the number of lagged forecast errors in the prediction equation.

$$y_t = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t - w_1 \varepsilon_{t-1} - w_2 \varepsilon_{t-2} - \cdots - w_q \varepsilon_{t-q} \qquad 2.3$$

The $ARIMA(p,d,q)$ model use combinations of past values and past forecasting errors and offer a potential for fitting models that could not be adequately fitted by using an AR or an MA model alone. Furthermore, the addition of the differencing eliminates most non-stationarity in the series.

A significant difference between the ARIMA methodology and previous methods is that ARIMA does not make assumptions about the number of terms or the relative weights to be assigned to the terms. To specify the model, the analyst first selects the appropriate model, including the number of p,d,q terms; then calculates the coefficients and gives a refined suggestion of the model parameters by using a nonlinear least squares method (Hanke, 1995; Thomas, 1983). The Best ARIMA function in R utilizes Akaike Information Criterion (AIC) to choose the p,d,q value and identify the best ARIMA model.

2.4 Akaike Information Criterion (AIC)

The appropriate choice of p,d,q terms of $ARIMA(p,d,q)$ model has the potential of improving forecast accuracy. There are two ideas for the model selection: one is select one appropriate model for the series under consideration, the other is use a general selection methodology which will select the appropriate model for each series from a group of candidate models. Empirical Information Criterion (EIC) is a model selection method that is designed to be used in forecasting a large number of time series. There are many EIC available for forecaster to choose, one popular criterion is Akaike Information Criterion (AIC). In this paper, in order to choose the best $ARIMA(p,d,q)$ model for each time series, the AIC is applied in the model selection procedure. For a fitted ARIMA time series of length n, the AIC is defined to be:

$$AIC = \ln(\hat{\sigma}_{p,q}^2) + 2(p+q)/n \qquad 2.4$$

where $\hat{\sigma}_{p,q}^2$ is the residual error variance from the fitted model. When comparing fitted models, the basic idea is the smaller the AIC, the better the fit. Note that the AIC penalizes for additional model complexity with the addition of $2(p+q)/n$. The degree of differencing d is manually set subject to the seasonal pattern of the time series. The approach of these Information Criterion methods is that of penalized likelihood (Sakamoto,1986).

3. THE OVERVIEW OF BAYESIAN STATISTICS AND DLM

The basic assumption of Bayesian statistics is that all uncertainties should be represented and measured by probabilities. The extension of the Bayesian presupposition is that, in forecast field, the true of the future could be represented by the past with a

measurable probability. Bayesian methodology offers a comprehensive way of routine learning that is not dependent on any particular assumption, Such that, it provides consistent and intuitional results in forecasting.

Suppose a dynamic model $\mathcal{M}$ constructed by number models M, such that the prior probability $P(M)$ describing the likely of M to be selected in forecasting an uncertain quantity Y. Also, a conditional probability distribution $P(Y \mid M)$ is used to specify the likelihood of each member model M giving out a correct future value of Y conditional upon that particular M. By probability law, these two sets of probabilities combine to provide a joint probability distribution as:

$$p(Y, M) = p(Y \mid M)\, p(M). \tag{3.1}$$

From Bayesian theory

$$p(Y \mid M) = p(Y, M) / p(M) = p(M \mid Y)\, p(Y) / p(M) \tag{3.2}$$

and Therefore

$$p(M \mid Y) = p(Y \mid M)\, p(M) / p(Y). \tag{3.3}$$

When Y is deserved to take a value Y^*, the updated probability distribution for M given $Y = Y^*$ is defined by the conditional density

$$p(M \mid Y^*) \propto p(Y^* \mid M)\, p(M) \tag{3.4}$$

which is often expressed as

$$Posterior \propto Observed \quad likelihood \times prior$$

The DLM is a Bayesian paradigm for time-series analysis detailed in Pole et al. (1994) and West and Harrison (1997). Generally, DLM is defined as a

quadruple F_t, G_t, V_t, W_t , which contains four components: regression vector F_t , evolution matrix G_t , observation variance V_t , and evolution variance W_t . In order to specify a DLM, these four components must be specified for each period t . In the following chapter, from simple to complex, we first outline the First Order Polynomial Model, and then generalize the High Order Polynomial Model and the Dynamic Linear Model (West 1997).

3.1 Polynomial Model

The simplest and most widely used DLM, so called First Order Polynomial Model, is characterized by the quadruple $1, 1, V_t, W_t$. At time t , Y_t represents the corresponding value of the time series; μ_t represents the level of the series. The observational error v_t and the evolution error ω_t are internally independent, mutually independent, and independent of $(\mu_0 \mid D_0)$, the initial level μ_0 given the initial information set D_0 which is the information we have form the outside of the time series before we do the forecasting. In the first order polynomial model, the variance sequences V_t and W_t are known constants of the existing information. In brief, the time evolution is modeled as a simple random walk upon a locally constant mean μ_t .

Observation equation: $\mathbf{Y_t = \mu_t + \nu_t}$ $\mathbf{\nu_t \sim N[0, V_t]}$ 3.5

System equation: $\mathbf{\mu_t = \mu_{t-1} + \omega_t}$ $\mathbf{\omega_t \sim N[0, W_t]}$ 3.6

Initial information: $(\mu_0 \mid D_0) \sim N[m_0, C_0]$ 3.7

This First order polynomial model is used effectively in numerous applications, particularly in short-term forecasting for production planning and stock control.

To extend to the Second Order Polynomial Model, a growth component, which itself also drifts over time, was added to the local level of the First Order Polynomial Model. Therefore, the second order model equations can be formed as below.

Observation equation: $$\boldsymbol{Y_t = \theta_{1,t} + \nu_t} \qquad \boldsymbol{\nu_t \sim N[0, V_t]} \quad 3.8$$

System equation: $$\boldsymbol{\theta_{1,t} = \theta_{1,t-1} + \theta_{1,t-2} + \omega_{1,t}} \quad \boldsymbol{\omega_t \sim N[0, W_t]} \quad 3.9$$

The F_t and G_t are risen to corresponding 2×2 matrix:

$$F_t = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad G_t = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad 3.10$$

Following this idea, the n[th] order polynomial model could be produced by extending straightforward from the formulations above. The regression vector F_t and the evolution matrix G_t of the n[th] order polynomial model are written as

$$F_t = \begin{pmatrix} F_{1,t} \\ \vdots \\ F_{n,t} \end{pmatrix} \qquad G_t = \begin{pmatrix} G_{1,1} & \cdots & G_{1,t} \\ \vdots & \ddots & \vdots \\ G_{n,1} & \cdots & G_{n,t} \end{pmatrix}, \qquad 3.11$$

where $F_{i,j}$ and $G_{i,j}$ could be any number (West 1997).

3.2 The Dynamic Linear Model

Based on the fundamental concepts and important features of the general class of normal dynamic linear models and simple regression models, we go to the general

normal dynamic model (DLM). The general normal dynamic model (DLM) is characterized by a quadruples.

$$\{F, G, V, W\}_t = \{F_t, G_t, V_t, W_t\}$$

for each time t, where

(a) F_t is a known $(n \times r)$ matrix;

(b) G_t is a known $(n \times n)$ matrix;

(c) V_t is a known $(n \times n)$ variance matrix;

(d) W_t is a known $(n \times n)$ variance matrix.

This quadruple defines the model relating Y_t to the $n \times 1$ parameter vector θ_t sequence through time, the equations are as below.

Observation equation:	$Y_t = F_t' \theta_t + v_t$	$v_t \sim N[0, V_t]$	3.12
System equation:	$\theta_t = G_t \theta_{t-1} + w_t$	$w_t \sim N[0, W_t]$	3.13

The error sequence v_t and w_t are internally and mutually independent. Defined by the observation equation, the sampling distribution for Y_t is conditional on the quantity θ_t.

For time t

(1) F_t is the design matrix of known values of independent variable;

(2) θ_t is the state, or system, vector;

(3) $\mu_t = F_t' \theta_t$ is the mean response, or level;

(4) v_t is the observational error;

(5) G_t is the evolution, system, transfer or state matrix;

(6) w_t is the system, or evolution, error with evolution variance W_t;

The table below shows that the algorithm of the univariate DLM (Table 3) which is the foundation of our Matlab program. The F_t matrix displays the correlation of the known values which is the system inputs. Generally, for nonseasonal time series, each past data is considered to contribute a equal weight to the future, hence F_{ij} in F_t are all take the value as one; for seasonal time series, the F_t need to be identified as a proper matrix that represents the seasonal circle, e.g. F_t=[1,0,0,0]^t may fit the quarterly data. Considering the seasonal characteristic, a rotation matrix was picked for the transfer matrix G_t . The first system input vector m_t usually comes from the mean of certain past data, but in seasonal model, m_t vector takes the average of two data that are separated by one seasonal circle distance. The other parameters exist in the DLM model, such as w_t, d_t, S_t, W_1 , are optimized by utilizing a grid optimization method to seek the minimum values of the mean square error. In the grid optimization, we offer a wild range of members as candidates to the parameters being optimized, and then we test all the interactions and select the best one for each series.

Univariate DLM: unknown, constant variance V= ϕ^{-1}

Observation: $Y_t = F_t^{'}\theta_t + v_t \qquad v_t \sim N[0, V_t]$

System: $\theta_t = G_t\theta_{t-1} + w_t \qquad w_t \sim T_{n_{t-1}}[0, W_t]$

Information: $(\theta_{t-1} \mid D_{t-1}) \sim T_{n-1}\ [m_{t-1}, C_{t-1}]$

$$(\phi \mid D_{t-1}) \sim G\left[\frac{n_{t-1}}{2}, \frac{n_{t-1}s_{t-1}}{2}\right]$$

Forecast: $(Y_t \mid D_{t-1}) \sim T_{t-1}\ \ f_t, Q_t$

$(\theta \mid D_{t-1}) \sim T_{t-1}[a_t, R_t]$

where

$$R_t = G_t C_{t-1} G_t^{'} + W_t \qquad a_t = G_t m_{t-1}$$

$$Q_t = F_t^{'} R_t F_t + S_{t-1} \qquad f_t = F_t^{'} a_t$$

Updating Recurrence Relationships:

$$(\phi \mid D_t) \sim G\left[\frac{n_t}{2}, \frac{n_t s_t}{2}\right]$$

$(\theta_t \mid D_t) \sim T_{n_t}\ \ m_t, C_t$

With $e_t = Y_t - f_t$ and $A_t = R_t F_t / Q_t$

$$n_t = n_{t-1} + 1$$

$$S_t = S_{t-1} + \frac{s_{t-1}}{n_t}\left(\frac{e_t^2}{Q_t} - 1\right)$$

$$m_t = a_t + A_t e_t$$

$$C_t = \frac{S_t}{S_{t-1}}(R_t - A_t A_t^{'} Q_t)$$

Forecast Distributions: $k \geq 1$

$(\theta_{t+k} \mid D_t) \sim T_{n_t}[a_t(k),\ R_t(k)]$

$(Y_{t+k} \mid D_t) \sim T_{n_t}[\ f_t(k),\ Q_t(k)]$

Table 3. Univariate DLM

4. THE OVERVIEW OF ANTIFICIAL NEURAL NETWORKS (ANNs)

ANNs may be defined as "an information processing technology inspired by studies of the brain and nervous system" (Klimasauskas, 1991). In computer science, ANNs is a processor made up of massively parallel distributed simple processing units, which has a natural propensity for storing experiential knowledge, doing logical and quantitative analysis, and generalize new information from acquired knowledge. It is similar to the brain in two respects:

1. Acquire knowledge from its environment through a learning process.
2. Using synaptic weight to store the acquired knowledge.

The working of the ANNs may vary from different structures of the network, generally a series of connecting neuron weights are assigned to each inputs signal and are adjusted to fit this series of inputs to another series of known outputs which are the network target. When the weight of a particular neuron is continually updated to improve the network performance, it is said that the neuron is learning. The training is the process that neural network learns. A properly trained network tends to give reasonable answers when presented with inputs that they have never seen.

The most important advantage of neural networks is in solving problems that are too complex for conventional techniques. These kinds of problems include pattern recognition and data forecasting. Today ANNs has been widely applied to many real world problems: business, physical system control, engineering, statistics, also medical and biological fields (Haykin, 1994).

4.1 ANNs in Forecasting

The science of Artificial Neural Network (ANNs) has a history of about five decades but has been solid in application for only the past fifteen years. The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pitts. But the technology available at that time did not allow them to do any deep research. Today, with the further understanding of human brain and the huge progress of computer science, significant progress has been made in ANNs algorithms. Currently, ANNs are being used for a wide variety of tasks in many different fields of business, industry and science (Widrow, 1994).sd

One major area of application for ANNs is time series forecasting such as predicting stock price, future inventory, and sales marketing (Sharda, 1994). As a nonlinear, sophisticated forecasting method, ANNs has several special features which make it an attractive alternative tool for both forecasting researchers and practitioners.

ANNs are data-driven self-adaptive methods with few prior assumptions. They learn from existing information and capture faint relationships among the data even if the underlying relationships are unknown or difficult to describe in closed form. Therefore, ANNs is appropriate for problems whose solutions require knowledge that is difficult to specify but have enough data or observations available (White, 1898; Ripley, 1993). The adaptability, reliability and robustness of an ANNs only depend on the source, range, quantity and quality of the given data set. ANNs can generalize from learning the data

presented to them, similar to the human brain. ANNs can often correctly catch and infer the unseen part of a population, even if the sample data given contains noisy information (Perambur 1994). ANNs is a universal function approximation system; it can be set to approximate any continuous function to any desired accuracy (Cybenko, 1989; Funahashi, 1989). ANNs can do nonlinear data-driven approach; therefore it is not necessary to make any assumptions of the underlying distribution of the data. This important feature overcomes the weakness of the conventional approaches such as ARIMA that stands in the assumption that the given time series is generated from a linear process which is not always true for real world systems (Haykin, 1994). ANNs has the capability of performing nonlinear regression without knowing the relationship between input and output variables. This makes it a general and flexible modeling tool for the real data forecasting.

When using the ANNs in forecasting, we should always be aware of some disadvantages. First, the individual relationship between the input variables and the output variables are not developed from mathematical deduction, so that the model tends to be a black box without a clear theoretical base. Secondly, a large sample size of data is required to obtain a stable and logical forecasting result. Finally, the ANNs forecasting can be time consuming. In some incarnations these ANNs may never converge; thus, training (learning) will continue for infinity.

4.2 Biological Structure

The human nervous system may be viewed as a three-stage system. Central to the system is the brain, represented by the neural net, which continually receives information, perceives it, and makes appropriate decisions. The receptors convert stimulation from the human body or the external environment into electrical impulses that convey information to the neural net. The effectors convert electrical impulses generated by the neural net into discernible responses as system output.

There are forward and backward arrows connecting these three stages. The forward arrows present the transmission of information-bearing signals through the system and the backward present the system feedback. (for more biological details see Perambur 1994).

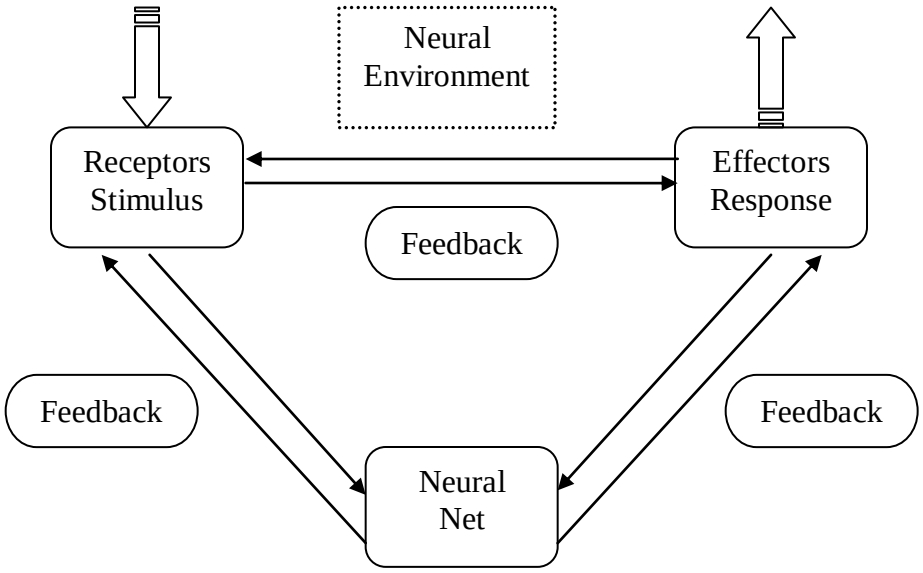

Figure 2. Three stages model of a ANNs

4.3 Single Network ANNs

The ANNs was developed in an effort to model the human neuron. The single artificial neuron, also called Perceptron, is depicted below (Figure 3). Inputs enter the neuron and are multiplied by their respective synaptic weight.

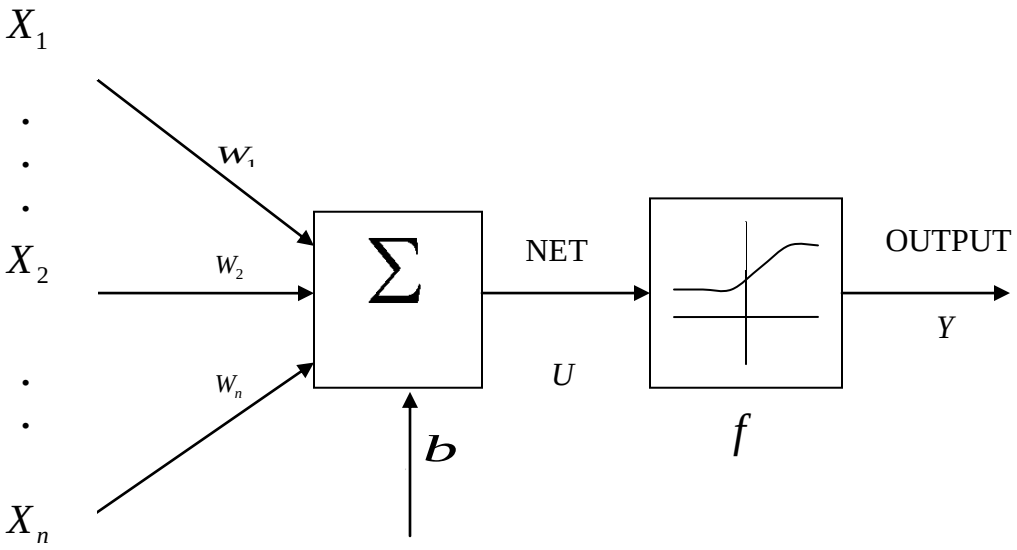

Figure 3. Single Network

Let $X_1, X_2, \cdots, X_n$ be input signals while $W_1, W_2, \cdots, W_n$ represent synaptic weights of neurons. The neuron will sum these weighted inputs and, with reference to a bias b as the input argument of the activation function f. The activation output Y is an input to the next layer or it is a response of the neural network if it is the last layer.

$$U = \sum_{1}^{n} W_i X_i + b \qquad 4.1$$

$$Y = f(U) \qquad 4.2$$

The activation function f bounds the neuron's output. There are various kinds of activation functions that could be chosen. Two common activation functions are the Pure Linear function

$$f(U) = U \qquad 4.3$$

and the Log-Sigmoid function (Haykin, 1994).

$$f(U) = 1/(1+e^{(-U)}) \qquad 4.4$$

4.4 Network Architecture

ANNs are networks with multiple layers and a large number of interconnected neurons. The ANNs architecture can be specified by four variables that are: the number of input nodes (n); the number of hidden layers (k) and hidden nodes (m); the number of output nodes (i) (Figure 4). Generally, the number of input nodes corresponds to the number of variables in the input vector which equal to the number of lagged observations used to forecast the future values. The number of output nodes corresponds to the problem to be answered which is the forecasting horizon in time series forecasting. The selection of the number of hidden layers and hidden nodes has a great effect on training, convergence, and forecast performance. Empirical research supports that one hidden layer may need a large number of hidden nodes for most forecasting cases, such that it

may consume a lot of computing time in network training (Cybenko, 1989; Hronik, 1989). Two hidden layers network were found to be more efficient in many time series forecasting purposes (Barron, 1994; Zhang 1994). Many papers supported that a network never needs more than two hidden layers in general forecasting problems (Cybenko, 1988; Lapedes, 1988). There are many discussions about how to specify the number of hidden nodes, but all of them only work well in specific or similar cases. Many trials was done to try to find a general rule in network architecture optimization, but none of them can guarantee the best architecture for all real forecasting problems. Hence, the empirical approach is still a common way in finding the best network architecture (Zhang 1998). Neural networks are usually fully connected. This means that each neuron is connected to every output from the preceding layer and each neuron has its output connected to every neuron in the succeeding layer. For the input layer, every neuron has one input from the external world. An ANN with well designed network architecture has the ability to learn or store knowledge in their synaptic weights and then generalize the population truth or future information. Thus ANNs have been applied successfully in time series.

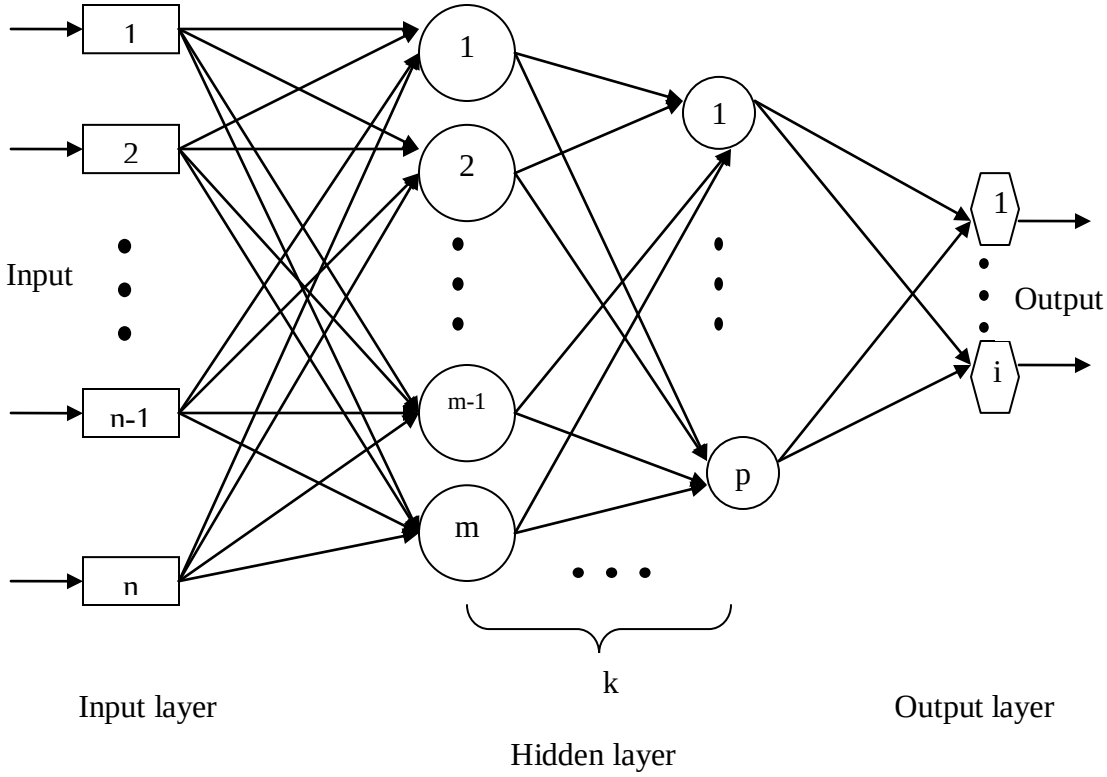

Figure 4. Network Architecture

4.5 Backpropagation Algorithm

In this paper we apply the Backpropagation Paradigm for a feed forward ANN to the M-3 Competition forecasting. Backpropagation algorithm was first proposed by Paul Werbos in the 1970's. In 1986, Rumelhart and McClelland rediscovered this algorithm and made it one of the most popular neural networks learning algorithms. Today, backpropagation network has been used successfully for wide variety of applications, such as forecasting, image pattern recognition, medical diagnosis, and automatic controls.

Backpropagation network made a tremendous progress from the single-layer perceptron. With a more sophisticated learning rule, backpropagation networks overcome the limitations that single-layer networks have which is the network can only approximate linear relationship between the inputs and the targets. Empirical reach shows that a backpropagation network with biases, a sigmoid layer, and a linear output layer are capable of approximating any function including linear and nonlinear. In our network design, we set output layer with linear transfer function and all the other layers with sigmoid transfer function to give the network the power for representing any functional relationship between the inputs and outputs.

Backpropagation network gets its name from its exclusive training procedure; the network feed forward the data from the input layer to the output layer through the hidden layers. The error signal between the outputs and the targets is backpropagated from the outputs to the inputs through the hidden layers in order to appropriately adjust the weights in each layer of the network until it can approximate a neuron weights function that can associate input vector with the specific output vector or narrow the total error into a defined value. A backpropagation network consists of at least three layers: one input layer, at least one hidden layer, and one output layer. Layers are feed forward connected with the input units fully connected to the hidden layer units and hidden units fully connected to the output layer units. Inside the backpropagation network flow cycle, the input nodes are propagated forward to the output nodes through the intervening input-to-hidden and hidden-to-output weights.

Standard backpropagation networks employ gradient descent algorithms to minimize the total error on the training set. Figure 5 illustrates the concept of gradient descent using a single weight. After the error on each pattern is computed, each weight is adjusted in proportion to the calculated error gradient backpropagated from the outputs to the inputs. The changes in the weights keep reducing the overall error until the performance goal is reached or the minimum gradient is met. The open-up parabola shows the relationship between the overall error and the changes in a single weight of a network (McClelland, 1988). In our network, we set the minimum performance gradient equal to 1×10^{-8} as a stop criterion of the network training.

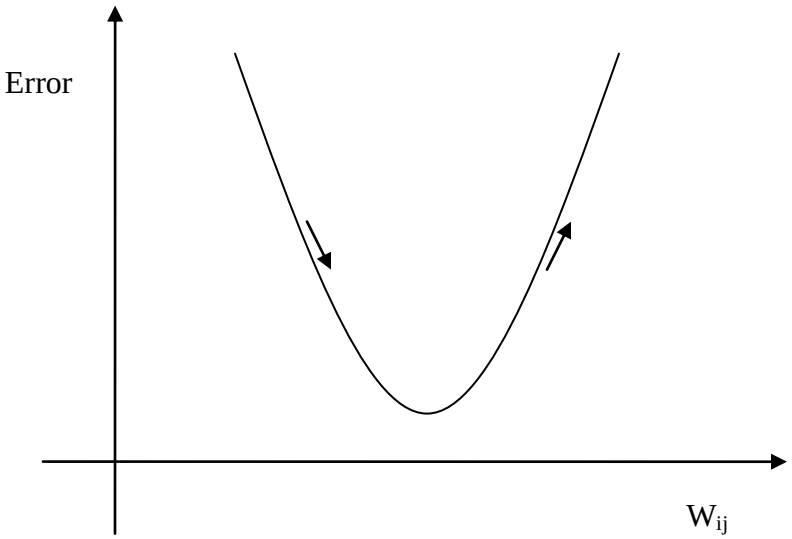

Figure 5. Gradient Descent Algorithm

The network training is set up following four steps. First, assemble the training data. Suppose a time series has n numbers of data and k of these n data are reserved for forecasting. Following the instruction of the M-3 Competition data, we specify a delay d as the number of the past data that are used in the training for the target. Horizon h is

the steps need to be forecast ahead. Second, create the network object. For a specify horizon, the target t is set to be the number that is one horizon ahead the last delay series. Following this procedure, the first target is the $d+h$ number of the training series while the last training target is the last number in the training series. Such that for each training circle, the input layer always has d neurons while the output layer has one neuron (Figure 6). Third, train the network. After the input series and the output series are set up, we go to the training process. There are several different training algorithms available in backpropagation network. These different algorithms have variety of computation and memory requirements. The selection of algorithm depends upon the problem at hand. Considering the increase in training speed and the reduction in memory requirement, we chose the Levenberg-Marquardt algorithm as the training algorithms of our network. The Levenberg-Marquardt iteration method is a variation of the Newton iteration. Newton's approach starts from an initial value x_0 and refines this value using the assumption that f is locally linear. A first order approximation of $f(x_0+\Delta)$ yields:

$$f(x_0+\Delta) = f(x_0) + J\Delta \tag{4.5}$$

with J the Jacobian matrix and Δ a small displacement. Under these assumptions minimizing $\hat{e}_0 - J\Delta$ can be solved through linear least-squares. An augmented equation yields from the simple derivation

$$N\Delta = J^T J\Delta = J^T \hat{e} \tag{4.6}$$

In Levenberg-Marquardt iteration, this augmented equation is changed to

$$N'\Delta = J^T \hat{e} \tag{4.7}$$

where

$$N'_{ij} = (1+\delta_{ij}\lambda)N_{ij} \qquad 4.8$$

with δ_{ij} the Kronecker delta . At the beginning of the iteration, the value λ is initialized to a small value. If the value obtained for Δ reduces the error, the increment is accepted and λ is divided by a certain number before the next iteration. On the other hand, if the error increases then λ is multiplied by a certain number and the augmented normal equations are solved again, until an increment is obtained that reduces the error. A large λ will give a steep descent in the approaches then reduce the convergent time. Once the training is done, we go to the fourth step, forecasting, which is simulating the network response to the new inputs. In the simulation, the input series is always set up to be the last d number of the training data and the target is aimed at the h number of the reserved data. The maximum h is the length of the reserved data k (Figure 7).

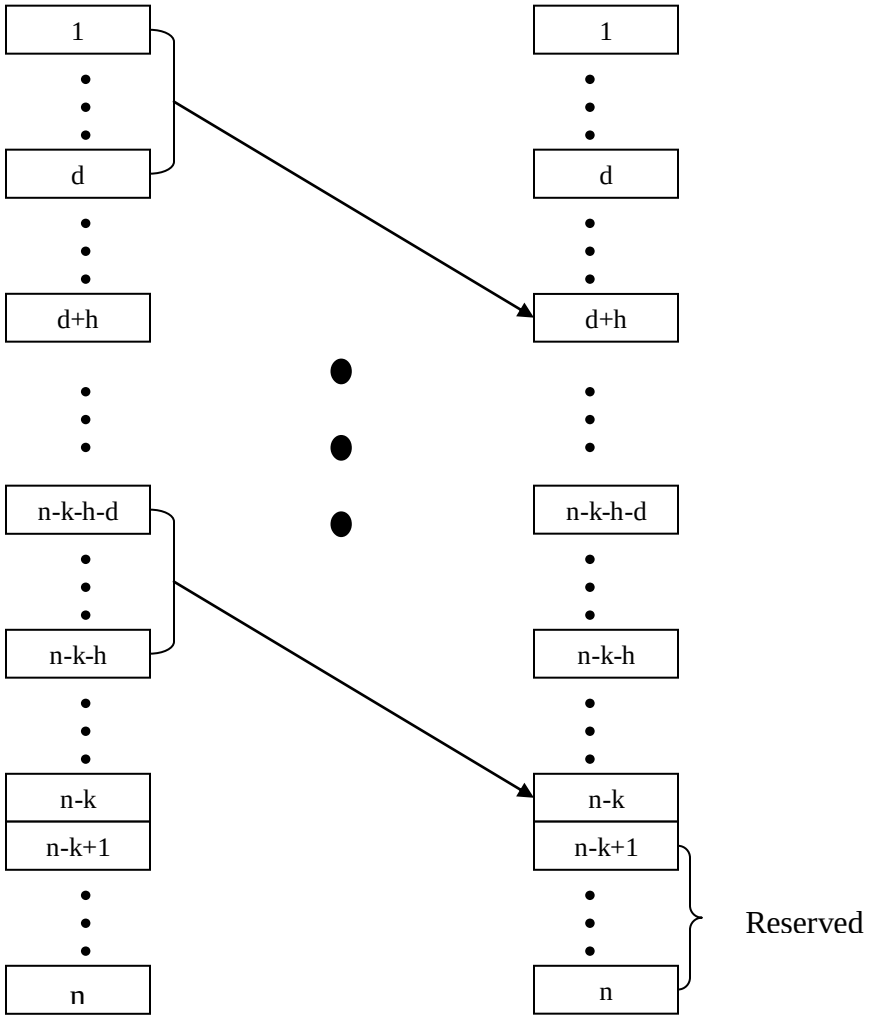

Figure 6. Network Training

n: Length of Time Series; k: Reserved Data;
d: Delay; h: Forecast Horizon.

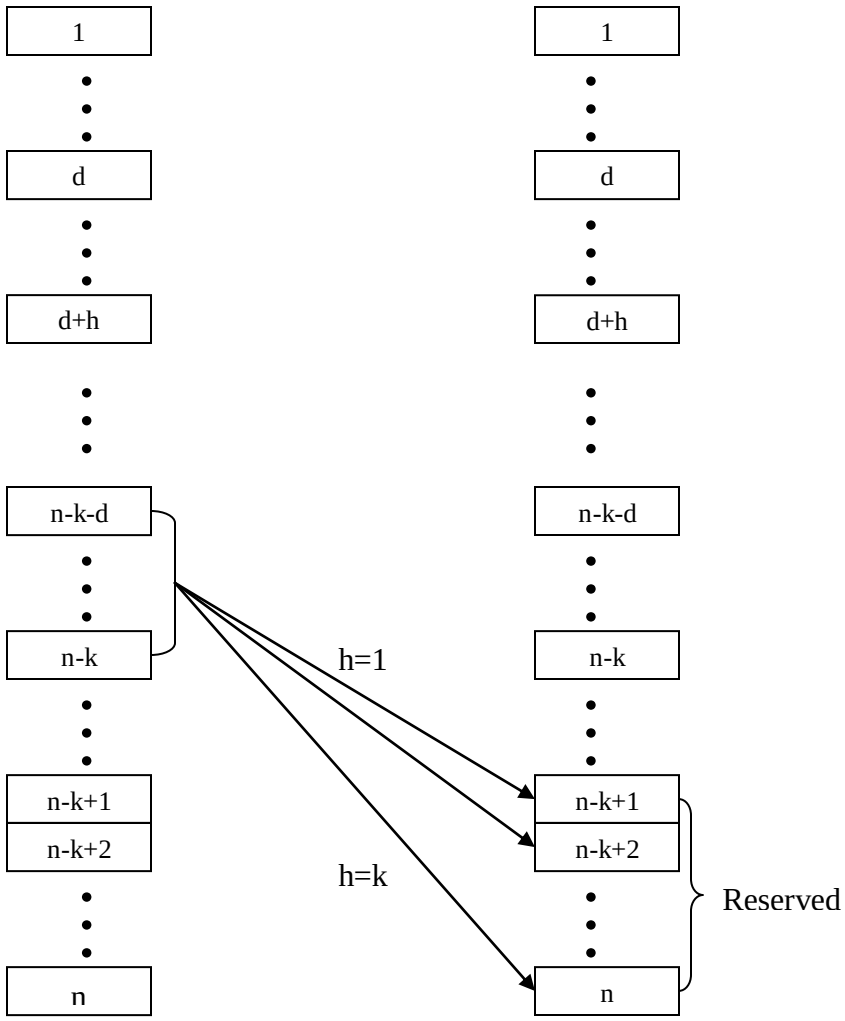

Figure 7. Forecasting

n: Length of Time Series; k: Reserved Data;
d: Delay; h: Forecast Horizon.

As we described before, the ANNs could be a time consuming algorithm, such that the computation time is considered as an important criterion of the network efficiency. In this paper, we explore the relationship between the elapsed time and the network structure from empirical test. In the next tables, the network configuration was specify by numbers inside a bracket, the first and the last number represent the number of neurons in the input and output layer, the numbers in the middle represent the neurons in each hidden layer. We describe the three conclusions we discover and take the "Other Data" set the of M-3 Competition as an instance: First, in a particular network, the elapsed time appears to be the quadratic function with respect to the delay number used in generalizing the future. A very short delay may cost a lot of elapsed time in training since the deficiency of the past information available for the network leads to a long time approximation (Figure 6). On the other hand, a very long delay may provide too much past information to the network thus causing an increase in training time. Therefore, if the performance is close for networks with different delay, we could minimize the elapsed time by choosing a proper delay. In Figure 6, twelve was selected to be the delay which empirical testing showed consumed less time than the others. Second, as the number of hidden layers increased, the numbers of neurons increase causing longer training times(Figure 7 and 8). Hence, in the neural network design, a network with a small number of hidden layer should be considered first. Most of the time, one or two hidden layers were sufficient and neural network with two hidden layers were found to be more efficient in many time series forecasting purposes (Zhang, 1994).

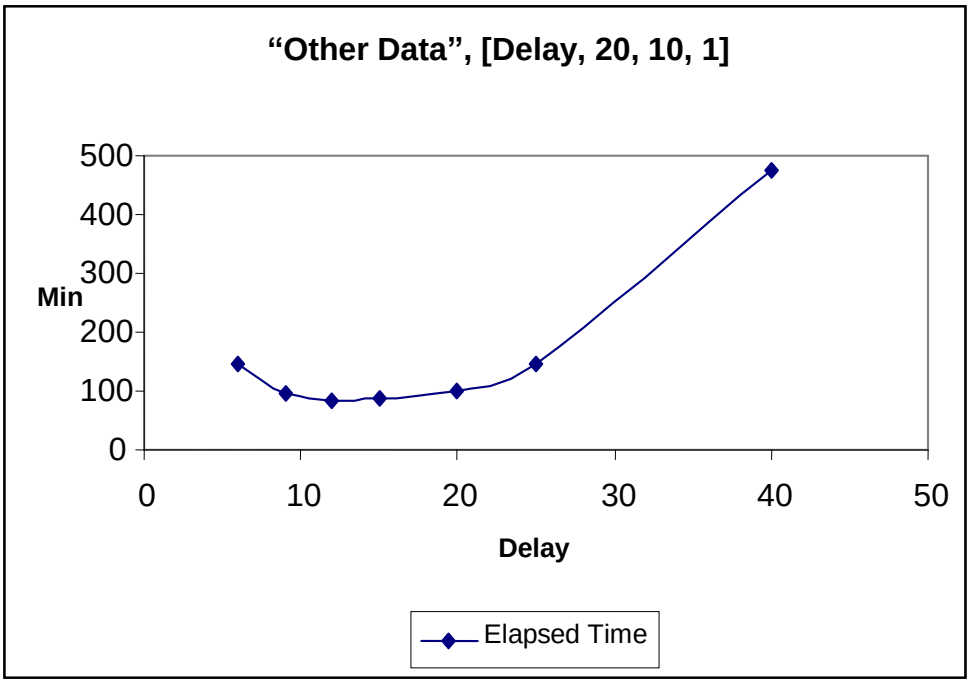

Figure 8. Elapsed Time and Delay

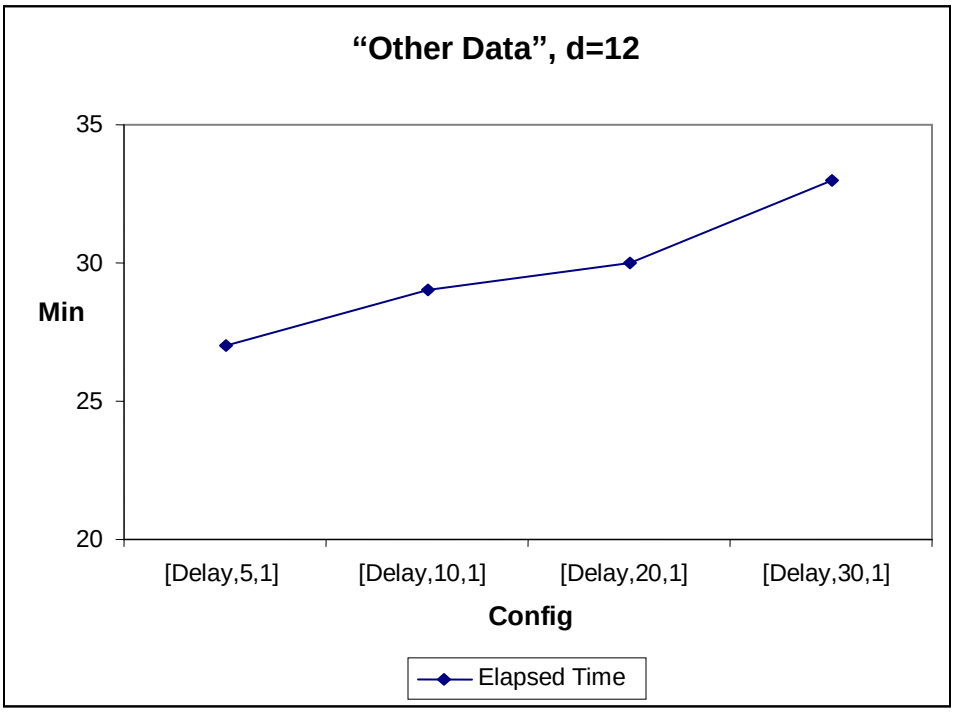

Figure 9. Elapsed Time and Network Configuration A

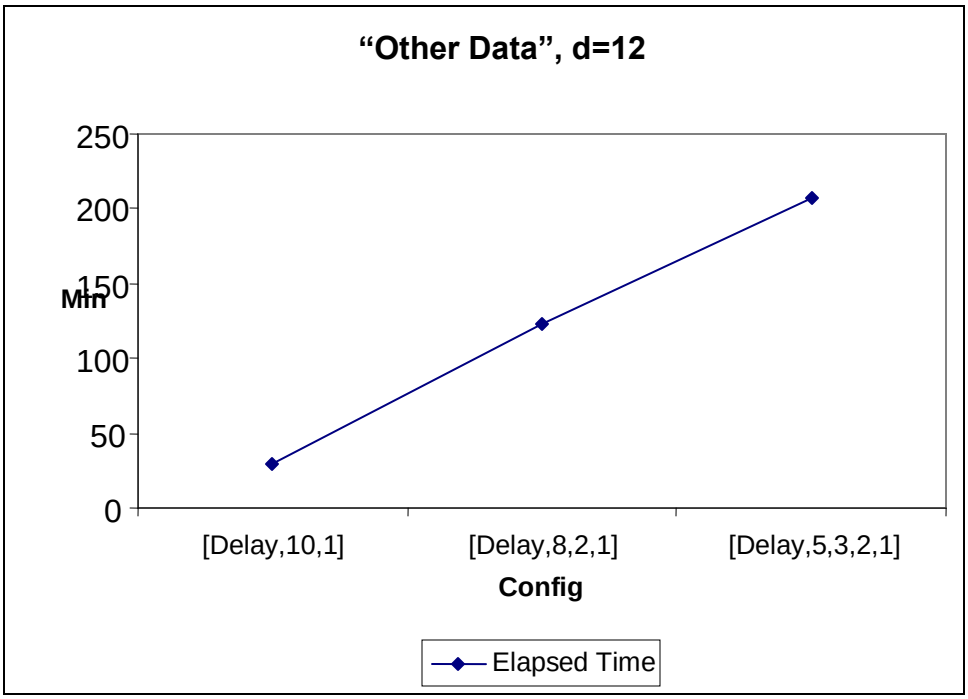

Figure 10. Elapsed Time and Network configuration B

The network configuration affects the network performance. To illustrate this relationship, we picture the performance of different network configuration in every single horizon (Figure 11). Generally, the forecast error increased as the forecast horizon H increased, the performance is approximately linear as seen in figure 11. But the same network configuration doesn't strictly monotonically increase over increasing horizon. For example, some network configuration, e.g. [20,10,10,10,1] (twenty neurons in input layer, ten neurons in each one of three hidden layers, one neuron in output layer) performs slightly better than the others in short term forecasting but not as well as some others, e.g. [9,10,10,10,1] in long term forecasting and vice versa. Some network configuration works well in the middle term forecasting but perform slightly worse in both ends, e.g. [12,20,10,1]. To assess the performance of different networks

configuration and identify the best network design for each forecast horizon, a statistics test is necessary. The mixed linear model is selected for the assessment in this work. The result of the mixed linear model will be discussed in detail in the next chapter.

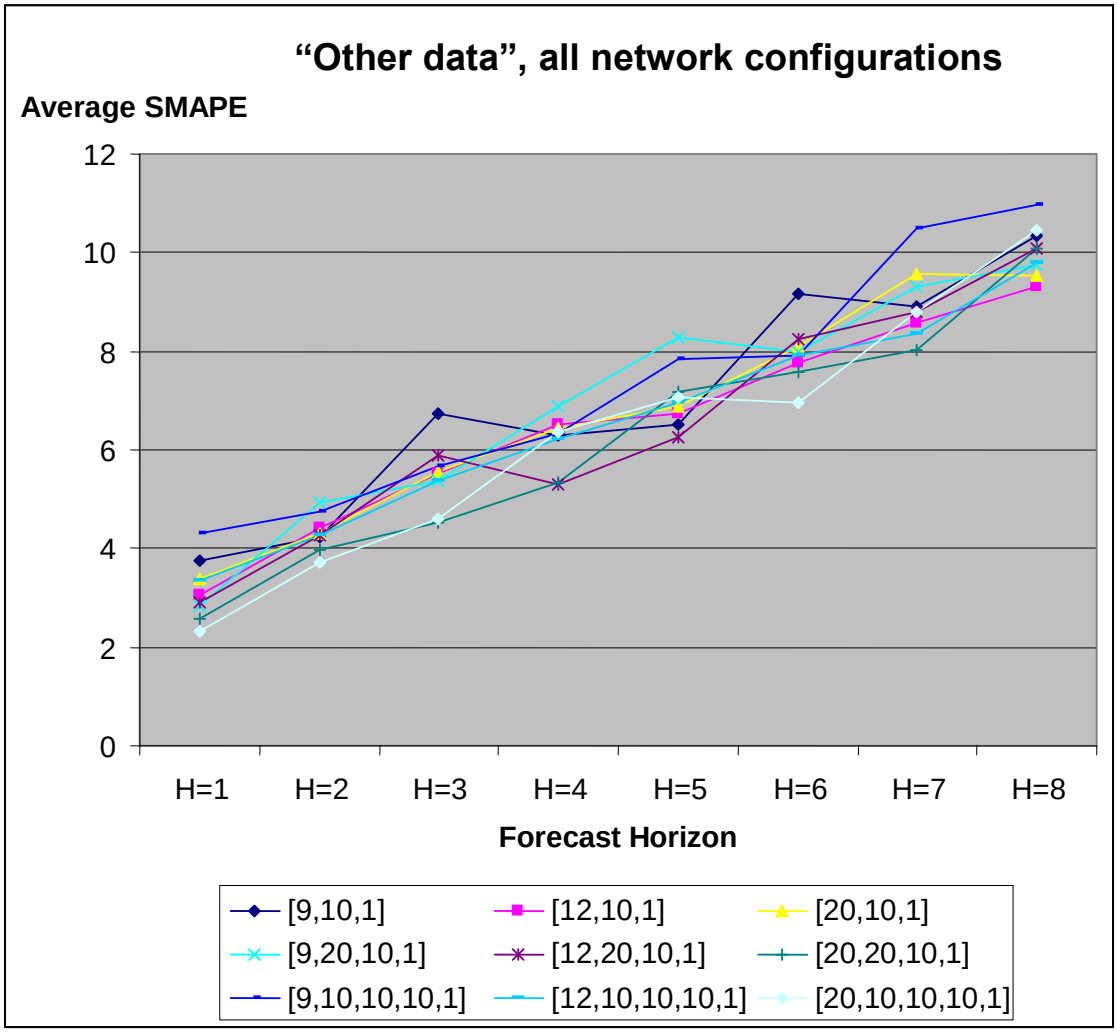

Figure 11. Average SMAPE of Different ANNs Configurations in All Horizons

5. MIXED LINEAR MODEL AND DISCUSSION

To explore where paradigms are statistically different we use the mixed linear model. The model we are using is defined by

$$A_{ij} = \mu + \tau_i + \beta_j + \varepsilon_{ij} \qquad 5.1$$

where A_{ij} is the accuracy measure associated with the i^{th} paradigm using the j^{th} time series, μ is the overall mean, τ_i is the i^{th} paradigm effect, β_j is the effect of the j^{th} time series where $\beta_j \overset{iid}{\sim} N(0,\sigma_\beta^2)$ is a random effect and ε_{ij} is the random error term where $\varepsilon_{ij} \overset{iid}{\sim} N(0,\sigma^2)$. By using this model to determine where the differences in paradigms exist we can account for the correlation induced by applying more than one forecasting paradigm to the same time series.

The mixed model analysis was performed in SAS using 'PROC MIXED'. The Mixed linear model tests the null hypothesis that all the selected paradigms produce the same SMAPE mean in every horizon. Therefore, the alternative hypothesis is that at least one SMAPE mean of one paradigm in a specific horizon is different from the others. With the Mixed linear model test, we are able to tell which paradigm produces smallest SMAPE mean and which paradigm produces the biggest SMAPE at a specific horizon and whether these differences is significant at the 0.05 confident level. The paradigms we test in the mixed ANOVA and the results are listed in the tables below. In Tables 4, 6, 8, 10, the average SMAPE and the standard error of each forecast horizon for each paradigm are listed. The result of the Theta method that was strongly recommended by

the M3-Competition as a simple and efficient method is also listed at the top of these tables for comparison. In Tables 5, 7, 9, 11, all paradigms are ranked in the order of descending SMAPE; the best and the worst paradigms are shaded in different colors. Paradigms that are statistically different are shaded in the same color.

Table 4 and Table 5 show that for yearly data, the First Order DLM generates the over all best performance in every forecast horizon but is not significantly different than the Best ARIMA at the first two forecast horizons at 0.05 confident levels. All the ANNs don't perform as well as the other paradigms; the ANN [3, 10, 1] performs significantly worst than other ANNs; Second Order DLM has the worst performance at the second horizon with three ANNs. Subject to the network training algorithm, yearly series with short series length, e.g. 14, are not be able to provide enough past information to the network training for long term forecasting. To explore how the ANNs works at long term forecasting for short time series, a further work on discovering new training algorithms is needed.

Table 6 and Table 7 show that the average SMAPE of the Best ARIMA is significantly smaller than all the other paradigms except at the fourth horizon on where four ANNs appear to have not significant difference with it. Seasonal DLM performs significantly worse at all horizons. Different ANNs generate close average SMAPE no matter what the network architecture it is. ANNs with delay match the quarterly pattern are slightly better than the other ANNs. This phenomenon indicates that ANNs are capable of recognizing the seasonal pattern when a proper delay is assigned to the

network configuration. With the same delay, ANNs with two hidden layers outperform ANNs with only one hidden layer; ANNs with one hidden layer and short delay appear to be inconsistent in forecast performance, e.g. [4,20,10,1] has a jump at horizon 4.

Table 8 and Table 9 show that for the monthly data, ANNs with suitable delay that match the series seasonal cycle (twelve for monthly data) outperform all the other paradigms at every single horizon at 0.05 significant level except that at the fifth horizon where the ANN[12,20,1] shows no difference with the Best ARIMA. Generally, ANNs with two hidden layers are showing better performance than ANNs with one hidden layer even though some differences are not statistical significant. On the other hand, ANNs with improper network architecture generates significantly bigger forecast error in forecasting. The Seasonal DLM doesn't perform as well as the Best ARIMA. Our research also confirm that the ANNs especially ANNs with more hidden layers and hidden nodes did cost a lot of time in network training , e.g. the ANN [12, 20, 10, 1] takes a fast computer three days to finish the 1428 monthly series. Such that, considering the forecast efficiency, we prefer ANN[12,20,1] to ANN[12,20,10,1]. The average SMAPE of the ANN[12,20,10,1] are better than the results of the Theta method used in the M3-Competition. If we assume that the SMAPE values of the theta method used in M3-Competition and the AMAPE values of ANN[12,20,10,1] used in our research share the same distribution and have close variance, then since ANN[12,20,10,1] generate a smaller SMAPE, it could be the overall best paradigms in monthly data forecasting among M3-Competition and our work. This rejects of the conclusion confirmed by the

M3-Competition saying that statistically sophisticated or complex methods do not necessary provide more accurate forecasts than simple ones (Makridakis, 2000).

Table 10 and Table 11 show that the Best ARIMA performs the best in every horizon and the difference is significant at 0.05 confidence level. ANNs with different network architecture generate close average SMAPE value at all horizons, ANNs with longer delay perform slightly better but consume much longer computation time (Figure 9). The average SMAPE of the First Order DLM is smaller than ANNs at all horizons but has no significant difference. The results of two Second Order DLM paradigms are significant worse in the other data set forecasting.

In summary, we reach four conclusions: first, different paradigms perform diversity in different categories of time series. First Order DLM performs best in yearly data while Best ARIMA works well with the other and quarterly data. ANNs gives out impressed performance in monthly data forecasting; second, unlike the one conclusion of the M3-Competition that statistically sophisticated paradigm are not as well as simple paradigm in time series forecasting. We discover statistically sophisticated paradigms, such as ANNs, is likely to produces better forecast accuracy then simple paradigms in monthly time series; Third, The length of the time series affect the ANNs performance. We consider this as the main reasons why the ANNs perform so much difference in different categorical time series, since the forecast performance of the ANNs rely on how much past information is available for the training process. The more past data offered, the better forecast accuracy received. Finally, complex DLM models are not necessary

better than simple DLM models in forecasting. This phenomenon shows in all time series forecasting. In our research, we also confirm one conclusion of the M3-Competition which is that the forecast performance depends upon the length of the forecasting horizon. Generally, the forecast error increase when horizon was increased, but there is some exclusion in ANNs when the model is not stable.

Finally, we provide a few comments in selecting a proper and efficient paradigm for time series forecasting. The Best ARIMA paradigm has proven to be good at short term forecasting for middle length time series, e.g. "Other" data in M3-Copetition. It also has the capability to catch the seasonal pattern of the time series, e.g. "Quarterly" data in M3-Competition. As for short term forecasting, when the time series is short and has no seasonal pattern, e.g. "Yearly" data in the M3-Competition, we strongly recommend the First Order DLM algorithm. In this case, the First Order DLM provides a simple paradigm for fast, stable, and accurate forecasting. In our competition, a well designed ANNs shows good performance in forecasting long time series, even with seasonal pattern at all forecast horizons. We discovered that, ANNs used in this work have poor performance when the training data is sparse/short. This proves that, to acquire a stable and logical forecasting result, a large number of sample are required by the ANNs used in this work. When using the ANNs in time series forecasting, forecasters should always be aware of computation time consumed in training process.

Following research is suggested to focus on:

1) Test and evaluate the long term forecast accuracy of First Order DLM in short nonseasonal time series.

2) Explore more effective technique in the parameter optimization step in DLM; Introduce the discount factor to the DLM paradigm to make the forecast result more adaptive and fit the real curve.

3) Explore a new training method for the ANNs to make it be able to do long term forecasting for short time series so that the entire forecast horizon required in the yearly data could be finished.

4) Explore how the G_t and F_t matrix affect the DLM model in seasonal time series forecasting, hence improve the DLM capability in catch the seasonal pattern.

5) Screen out the quarterly series that the ANNs generate abnormal forecast error or unstable forecast result. Then Check if the forecast performance could be improved by fixed the training method or/and change the value of network parameters.

Method	Forecasting Horizon					
	1	2	3	4	5	6
Theta Method in M3-Competition	8	12.2	16.7	19.2	21.7	23.6
Best ARIMA	11.06	17.61	18.38	21.64	25.12	22.42
Standard Error	9.39	11.77	11.53	12.64	16.06	65.94
First Order DLM	10.01	16.59	17.26	20.67	23.50	26.01
Standard Error	15.69	20.30	19.41	22.43	23.08	25.34
Second Order DLM	18.80	26.93	30.098	37.27	43.62	51.14
Standard Error	33.57	36.42	35.30	36.59	37.90	40.17
ANN:						
[3,10,1]	25.05	26.03	32.82	34.29		
Standard Error	99.05	42.94	107.09	55.53		
[4,10.,1]	17.58	23.92	25.89			
Standard Error	28.81	31.68	41.52			
[3,20,10,1]	16.34	23.88	25.33	29.12		
Standard Error	29.96	30.68	30.74	32.57		
[4,20,10,1]	17.98	26.12	28.32			
Standard Error	34.83	41.84	61.57			
[3,10,10,10,1]	17.67	26.18	27.12	29.38		
Standard Error	36.96	57,87	50.32	33.62		
[4,10,10,10,1]	16.33	24.77	25.92			
Standard Error	24.30	41.87	44.40			

Table 4. Average Symmetric MAPE: Yearly Data.

Rank	Forecasting Horizon					
	1	2	3	4	5	6
1	First Order DLM	First Order DLM	First Order DLM	First Order DLM	First Order DLM	Best ARIMA
2	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	First Order DLM
3	ANN [4,10,10,10,1]	ANN [3,20,10,1]	ANN [3,20,10,1]	ANN [3,20,10,1]	Second Order DLM	Second Order DLM
4	ANN [3,20,10,1]	ANN [4,10,1]	ANN [4,10,1]	ANN[3,10, 10,10,1]		
5	ANN [4,10,1]	ANN [4,10,10,10,1]	ANN[4,10, 10,10,1]	ANN [3,10,1]		
6	ANN [3,10,10,10,1]	ANN [3,10,1]	ANN[3,10, 10,10,1]	Second Order DLM		
7	ANN [4,20,10,1]	ANN [4,20,10,1]	ANN[4,20, 10,1]			
8	Second Order DLM	ANN [3,10,10,10,1]	Second Order DLM			
9	ANN [3,10,1]	Second Order DLM	ANN [3,10,1]			

Table 5. Results of Mixed Model: Yearly Data

Method	Forecasting Horizon							
	1	2	3	4	5	6	7	8
Theta Method in M3-Competition	5	6.7	7.4	8.8	9.4	10.9	Non	12
Best ARIMA	5.49	6.88	7.71	8.17	10.13	11.00	12.36	13.67
Standard Error	5.25	5.87	6.32	6.12	7.39	7.74	8.31	9.73
Seasonal DLM								
Ft=[10001]	28.25	27.45	26.72	30.30	24.80	31.24	29.58	29.19
Standard Error	27,09	27.07	26.65	28.04	23.78	27.56	28.34	25.51
Ft=[1000]	27.36	30.60	27.25	28.57	30.52	32.64	29.31	30.56
Standard Error	29.13	32.96	27.06	26.04	30.03	32.71	28.34	26.97
ANN								
[4,20,10,1]	14.44	13.22	18.50	43.04	20.41	20.94	21.56	19.73
Standard Error	47.16	22.13	25.52	30.28	24.85	29.83	34.71	27.43
[6,20,10,1]	9.52	11.32	13.19	14.89	15.67	18.20	17.15	18.43
Standard Error	15.46	15.80	21.04	24.46	20.16	24.40	20.55	21.66
[8,20,10,1]	12.53	14.29	17.10	15.97	18.40	20.15	20.66	23.32
Standard Error	20,01	17.90	23.09	21.97	18.30	21.77	25.41	27.43
[4,10,10,10,1]	10.62	18.99	14.57	19.68	19.74	17.08	20.29	21.31
Standard Error	29.00	19.40	26.56	27.48	37.06	21.35	33.78	35.98
[6,10,10,10,1]	9.74	11.84	14.02	13.72	15.29	16.57	17.20	19.80
Standard Error	17.17	18.36	29.82	19.72	20.29	21.43	20,31	26.88
[8,10,10,10,1]	13.22	13.50	16.04	18.90	18.53	20.11	21.48	24.52
Standard Error	22.24	16.58	22.13	29.24	18.08	21.03	20.67	27.53

Table 6. Average Symmetric MAPE: Quarterly Data

Rank	Forecasting Horizon							
	1	2	3	4	5	6	7	8
1	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA
2	ANN[6, 20,10,1]	ANN[6, 20,10,1]	ANN[6, 20,10,1]	ANN[6, 10,10,10, 1]	ANN[6, 10,10,10, 1]	ANN[6, 10,10,10, 1]	ANN[6, 20,10,1]	ANN[6, 20,10,1]
3	ANN[6, 10,10,10, 1]	ANN[6, 10,10,10, 1]	ANN[6, 10,10,10, 1]	ANN[6, 20,10,1]	ANN[6, 20,10,1]	ANN[4, 10,10,10, 1]	ANN[6, 10,10,10, 1]	ANN[4, 20,10,1]
4	ANN[4, 10,10,10, 1]	ANN[4, 20,10,1]	ANN[4, 10,10,10, 1]	ANN[8, 20,10,1]	ANN[8, 20,10,1]	ANN[6, 20,10,1]	ANN[4, 10,10,10, 1]	ANN[6, 10,10,10, 1]
5	ANN[8, 20,10,1]	ANN[8, 10,10,10, 1]	ANN[8, 10,10,10, 1]	ANN[8, 10,10,10, 1]	ANN[8, 10,10,10, 1]	ANN[8, 10,10,10, 1]	ANN[8, 20,10,1]	ANN[4, 10,10,10, 1]
6	ANN[8, 10,10,10, 1]	ANN[8, 20,10,1]	ANN[8, 20,10,1]	ANN[4, 10,10,10, 1]	ANN[4, 10,10,10, 1]	ANN[8, 20,10,1]	ANN[8, 10,10,10, 1]	ANN[8, 20,10,1]
7	ANN[4, 20,10,1]	ANN[4, 10,10,10, 1]	ANN[4, 20,10,1]	DLM Ft= [1000]	ANN[4, 20,10,1]	ANN[4, 20,10,1]	ANN[4, 20,10,1]	ANN[8, 10,10,10, 1]
8	DLM Ft= [1000]	DLM Ft= [10001]	DLM Ft= [10001]	DLM Ft= [10001]	DLM Ft= [10001]	DLM Ft= [10001]	DLM Ft= [1000]	DLM Ft= [10001]
9	DLM Ft= [10001]	DLM Ft= [1000]	DLM Ft= [1000]	ANN[4, 20,10,1]	DLM Ft= [1000]	DLM Ft= [1000]	DLM Ft= [10001]	DLM Ft= [1000]

Table 7. Results of Mixed ANOVA: Quarterly Data

Method	Forecasting Horizon									
	1	2	3	4	5	6	8	12	15	18
Theta Method in M3	11.2	10.7	11.8	12.4	12.2	12.4	12.7	13.2	16.2	18.2
Best ARIMA	12.64	11.96	12.86	13.79	14.81	14.79	15.72	15.81	18.69	21.25
Standard Error	13.82	12.35	10.65	12.90	9.70	9.80	11.10	11.35	15.28	25.73
Seasonal DLM	25.85	26.13	26.75	26.00	26.03	25.78	24.15	30.40	30.63	29.21
Standard Error	27.06	26.99	28.73	28.87	27.78	28.91	29.54	29.89	30.12	29.55
ANN										
[8,20,1]	36.92	34.13	31.68	36.93	53.82	42.29	68.23	38.09	38.43	36.50
Standard Error	172.8	112.1	72.71	146.4	486.3	173.6	864.0	152.9	148.9	109.5
[12,20,1]	10.05	11.45	11.56	13.74	12.31	12.92	12.90	14.02	16.07	18.89
Standard Error	50.31	17.67	48.33	161.4	144.6	23.90	25.14	25.06	30.61	34.16
[15,20.,1]	29.45	29.93	34.84	34.46	35.88	35.51	32.92	36.14	30.77	
Standard Error	36.54	27.52	28.26	34.82	29.90	43.99	32.53	28,97	29.53	
[8,20,10,1]	31.26	29.45	29.93	34.84	34.46	35.88	34.82	31.64	31.87	31.23
Standard Error	35.64	65.82	86.33	132.4	343.2	147.1	453.6	145.2	84.21	93.83
[12,20,10,1]	8.53	8.80	9.87	9.76	10.19	9.90	10.53	12.73	13.74	16.11
Standard Error	15.47	12.18	19,79	14.31	21.48	14.17	20,42	30.31	94,76	111.8
[15,20,10,1]	17.04	18.59	18.52	20.28	18.79	19.14	21.21	19.67	20.61	
Standard Error	24.27	50.60	42.24	29.32	23.13	23.07	39.06	25.16	32.24	

Table 8. Average Symmetric MAPE: Monthly Data

Rank	Forecasting Horizon									
	1	2	3	4	5	6	8	12	15	18
1	ANN [12,20, 10,1]	ANN [12,20, 10,1]	ANN [12,20, 10,1]	ANN [12,20, 10,1]	ANN [12,20, 10,1]	ANN [12,20, 10,1]	ANN [12,20, 10,1]	ANN [12,20, 10,1]	ANN [12,20, 10,1]	ANN [12,20, 10,1]
2	ANN [12,20, 1]	ANN [12,20, 1]	ANN [12,20, 1]	ANN [12,20, 1]	ANN [12,20, 1]	ANN [12,20, 1]	ANN [12,20, 1]	ANN [12,20, 1]	ANN [12,20, 1]	ANN [12,20, 1]
3	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA
4	ANN [15,20, 10,1]	ANN [15,20, 10,1]	ANN [15,20, 10,1]	ANN [15,20, 10,1]	ANN [15,20, 10,1]	ANN [15,20, 10,1]	ANN [15,20, 10,1]	ANN [15,20, 10,1]	ANN [15,20, 10,1]	DLM
5	DLM	DLM	DLM	DLM	DLM	DLM	DLM	DLM	DLM	ANN [8,20, 10,1]
6	ANN [15,20, ,1]	ANN [8, 20,10, 1]	ANN [8, 20,10, 1]	ANN [15,20, 1]	ANN [8,20,1 0,1]	ANN [15,20, 1]	ANN [15,20, 1]	ANN [15,20, 1]	ANN [15,20, 1]	ANN [8,20,1]
7	ANN [8, 20,10, 1]	ANN [15,20, 1]	ANN [8, 20,1]	ANN [8, 20,10, 1]	ANN [15,20, 1]	ANN [8,20,1 0,1]	ANN [8,20,1 0,1]	ANN [8,20,1 0,1]	ANN [8,20,1 0,1]	
8	ANN [8, 20,1]	ANN [8, 20,1]	ANN [15,20, 1]	ANN [8, 20,1]	ANN [8,20,1]	ANN [8,20,1]	ANN [8,20,1]	ANN [8,20,1]	ANN [8,20,1]	

Table 9. Results of Mixed Model: Monthly Data

Method	Forecasting Horizon							
	1	2	3	4	5	6	7	8
Theta Method in M3-Competition	1.8	2.7	3.8	4.5	5.6	5.2	Non	6.1
Best ARIMA	1.59	2.81	3.40	4.15	4.27	4.75	4.89	5.75
Standard Error	1.42	3.50	3.85	4.28	3.27	3.46	3.17	3.57
First Order DLM	2.15	3.82	4.69	5.84	6.23	7.06	7.70	9.12
Standard Error	4.22	9.13	9.75	10.56	7.42	7.97	7.43	8.32
Second Order								
Ft=[1,1], Gt=[1001]	21.82	22.04	22.43	23.25	24.50	25.22	26.28	26.69
Standard Error	14.39	14.38	14.74	15.38	15.81	16.59	17.20	17.92
Ft=[1,1], Gt=[1010]	21.27	21.48	21.87	22.71	23.96	24.68	25.72	26.15
Standard Error	14.19	14.18	14.52	15.12	15.60	16.40	17.02	17.72
ANN								
[9,10,1]	3.77	4.23	6.72	6.31	6.52	9.17	8.89	10.36
Standard Error	7.13	5.85	15.79	10.53	7.44	18.04	9.68	11.47
[12,10,1]	3.04	4.42	5.51	6.52	6.73	7.75	8.58	9.30
Standard Error	4.38	9.18	8.13	8.10	7.71	10.25	10.08	9.45
[20,10.,1]	3.38	4.32	5.55	6.43	6.89	8.08	9.57	9.53
Standard Error	5.23	5.95	6.47	11.40	10.69	9.99	11.93	9.27
[9,20,10,1]	2.83	4.95	5.39	6.90	8.28	7.99	9.33	9.76
Standard Error	3.60	8.21	6.91	10.80	12.89	9.18	11.03	10.92
[12,20,10,1]	2.91	4.28	5.90	5.29	6.27	8.25	8.79	10.09
Standard Error	4.62	5.48	11.41	7.29	6.66	9.91	10.16	10.54
[20,20,10,1]	2.58	3.98	4.54	5.33	7.19	7.57	8.01	10.07
Standard Error	4.38	7.75	6.14	6.32	13.52	8.41	7.43	11.62
[9,10,10,10,1]	4.32	4.76	5.66	6.34	7.83	7.90	10.50	10.96
Standard Error	6.66	5.67	7.63	9.23	8.24	8.58	16.56	13.44
[12,10,10,10,1]	3.34	4.26	5.36	6.23	6.94	7.91	8.34	9.79
Standard Error	4.99	6.95	7.89	8.05	7.51	11.92	9.06	13.65
[20,10,10,10,1]	2.31	3.72	4.60	6.37	7.05	6.94	8.81	10.47
Standard Error	4.48	6.59	6.93	8.58	12.70	8.12	7.65	12.55

Table 10. Average Symmetric MAPE: Other Data

Rank	Forecasting Horizon							
	1	2	3	4	5	6	7	8
1	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA	Best ARIMA
2	First Order DLM	ANN[20, 10,10,10, 1]	ANN[20, 20,10,1]	ANN[12, 20,10,1]	First Order DLM	ANN[20, 10,10,10, 1]	First Order DLM	First Order DLM
3	ANN[20, 10,10,10, 1]	First Order DLM	ANN[20, 10,10,10, 1]	ANN[20, 20,10,1]	ANN[12, 20,10,1]	First Order DLM	ANN[20, 20,10,1]	ANN[12, 10,1]
4	ANN[20, 20,10,1]	ANN[20, 20,10,1]	First Order DLM	First Order DLM	ANN[9,1 0,1]	ANN[20, 20,10,1]	ANN[12, 10,10,10, 1]	ANN[20, 10,1]
5	ANN[9, 20,10,1]	ANN[9, 10,1]	ANN[12, 10,10,10, 1]	ANN[12, 10,10,10, 1]	ANN[12, 10,1]	ANN[12, 10,1]	ANN[12, 10,1]	ANN[9, 20,10,1]
6	ANN[12, 20,10,1]	ANN[12, 10,10,10, 1]	ANN[9, 20,10,1]	ANN[9, 10,1]	ANN[20, 10,1]	ANN[9, 10,10,10, 1]	ANN[12, 20,10,1]	ANN[12, 10,10,10, 1]
7	ANN[12, 10,1]	ANN[12, 20,10,1]	ANN[12, 10,1]	ANN[9, 10,10,10, 1]	ANN[12, 10,10,10, 1]	ANN[12, 10,10,10, 1]	ANN[20, 10,10,10, 1]	ANN[20, 20,10,1]
8	ANN[12, 10,10,10, 1]	ANN[20, 10,1]	ANN[20, 10,1]	ANN[20, 10,10,10, 1]	ANN[20, 10,10,10, 1]	ANN[9, 20,10,1]	ANN[9,1 0,1]	ANN[12, 20,10,1]
9	ANN[20, 10,1]	ANN[12, 10,1]	ANN[9, 10,10,10, 1]	ANN[20, 10,1]	ANN[20, 20,10,1]	ANN[20, 10,1]	ANN[9,2 0,10,1]	ANN[9, 10,1]
10	ANN[9, 10,1]	ANN[9,1 0,10,10,1]	ANN[12, 20,10,1]	ANN[12, 10,1]	ANN[9, 10,10,10, 1]	ANN[12, 20,10,1]	ANN[20, 10,1]	ANN[20, 10,10,10, 1]
11	ANN[9, 10,10,10, 1]	ANN[9, 20,10,1]	ANN[9, 10,1]	ANN[9, 20,10,1]	ANN[9, 20,10,1]	ANN[9, 10,1]	ANN[9,1 0,10,10,1]	ANN[9, 10,10,10, 1]
12	DLM Ft=[1,1] Gt=[1,0, 1,0]	DLM Ft=[1,1] Gt=[1,0, 1,0]	DLM Ft=[1,1] Gt=[1,0, 1,0]	DLM Ft=[1,1] Gt=[1,0, 1,0]	DLM Ft=[1,1] Gt=[1,0, 1,0]	DLM Ft=[1,1] Gt=[1,0, 1,0]	DLM Ft=[1,1] Gt=[1,0, 1,0]	DLM Ft=[1,1] Gt=[1,0, 1,0]
13	DLM Ft=[1,1] Gt=[1,0, 0,1]	DLM Ft=[1,1] Gt=[1,0, 0,1]	DLM Ft=[1,1] Gt=[1,0, 0,1]	DLM Ft=[1,1] Gt=[1,0, 0,1]	DLM Ft=[1,1] Gt=[1,0, 0,1]	DLM Ft=[1,1] Gt=[1,0, 0,1]	DLM Ft=[1,1] Gt=[1,0, 0,1]	DLM Ft=[1,1] Gt=[1,0, 0,1]

Table 11. Results of Mixed Model: Other Data

REFERENCE

[1] Akaike, H. Statistical predictor identification, Annals of Institute of Statistical Mathematics, (1970) 22, 203–217.

[2] Akiaike, H. Information theory and an extension of the maximum likelihood principle,in B.N. Petrov and F. Csaki (eds.), Second International Symposium on InformationTheory, Akademiai Kiado: Budapest, (1973) 267–281.

[3] Azoff, E.M., 1994. Neural Networks Time series Forecasting of Financial Markets. John Wiley and Sons, chichester.

[4] Armstrong, J. Scott and Fred Collopy. Error Measures For Generalizing About Forecasting Methods: Empirical Comparisons. International Journal of Forecasting 8 (1992), 69-80.

[5] Armstrong, J. Scott. Should we redesign forecasting competitions? International Jounal of Forecasting 17 (2001) 537-584.

[6] Box, G.E.P. and G.M. Jenkins (1970) Time series analysis: Forecasting and control, San Francisco: Holden-Day.

[7] Chatfield, D. A personal view of the M2-Competition. International Journal of Forecasting 9, No (1), 23-24.

[8] Clements, Michael P. and Hendry, David F. Explaining the results of the M3-Competition. International Journal of Forecasting 17 (2001) 537-584.

[9] Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Mathematical Control Signals Systems 2, 303-314.

[10] Funahashi, K., 1989. On the approximate realization of continuous mapping by neural networks. Neural Networks 2, 183-192.

[11] Hanke, John E. and Arthur G. Reitsch. Business Forecasting, Fifth Edition, Prentice Hall (1995)

[12] Haykin, Simon. Neural Networks, A Comprehensive Foundation, Second Edition, Prentice Hall (1999).

[13] Hornik, K., Stinchcombe, M., white, H., 1989. Multilayer feedforward networks are universal approximators. Nerual Netowrks 2, 395-366.

[14] Jeffrey, Jarrett. Business Forecasting Methods, Second Edition, Basil Blackwell (1990).

[15] Keith Ord. Commentaries on the M3-Competition, An introduction, some comments and a Scorecard. International Journal of Forecasting 17 (2001) 537-584.

[16] Keith Ord. Michele Hibon, Spyros Makridakis. The M3-Competition. International Journal of Forecasting 16 (2000) 433-436.

[17] Klimasauskas, C.C., 1991. Applying neural networks. Part 3: Training a neural network, PC-AI, May/June, 20-24.

[18] Koehler, Anne B. The asymmetry of the sAPE measure and other comments on the M3-Competition. International Journal of Forecasting 17 (2001) 537-584.

[19] Koning, Alex J. Philip Hans Franses, Michele Hibon and H. O. Stekler.The M3 competition: Statistical tests of the results. International Journal of Forecasting xx (2004) xxx-xxx.

[20] Lapedes, A., Farber, R., 1987. How neural nets work. In: Anderson, D.Z., (Ed.), Nerual Information Processing Systems, American Institute of Physics, New York, pp. 442-456.

[21] Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E., & Winkler, R. (1982). The accuracy of extrapolation (time series) methods: results of a forecasting competition. Journal of Forecasting 1, 111–153.

[22] Makridakis, S., Chatfield, C., Hibon, M., Lawrence, M., Mills, T., Ord, K., & Simmons, L. F. (1993). The M-2 Competition: a real-time judgmentally based forecasting study. International Journal of Forecasting 9, 5–23.

[23] Makridakis,Spyros Forecasting: its role and value for planning and strategy. International Journal of Forcasting 12 (1996) 513-537.

[24] Makridakis, S., & Hibon, M. (1979). Accuracy of forecasting: an empirical investigation (with discussion). Journal of the Royal Statistical Society A 142, 97–145.

[25] Makridakis, Spyros, Hibon Michele and Claus Moser. Accuracy of Forecasting: An Empirical Investigation. Journal of the Royal Statistical Society. Series A (General), Vol. 142, No, 2 (1997), 97-145.

[26] Makridakis Spyros and Hibon Michele. Response to the commentaries on 'The M3-Competition: results, conclusions and implications'. International Journal of Forecasting 17 (2001) 537-584.

[27] Makridakis Spyros and Michele Hibon. The M3-Competition: results, conclusions and implications. International Journal of Forecasting 16 (2000) 451-476.

[28] Makridakis, Spyros Wheelwright Steven C. and McGee, Victor E. forecasting: Methods and Applications, Second Edition, John Wiley & Sons (1983).

[29] McClelland, J. L., Rumelhart, D. E. Explorations in parallel distributed processing: A handbook of models, programs, and exercises (1988).

[30] Newbold, P., & Granger, C.W. J. (1974). Experience with forecasting univariate time series and the combination of forecasts (with discussion). Journal of Royal Statistical Society A 137, 131–165.

[31] Perambur S. Neelakanta and Dolores F. De Groff. Neural Network Modeling, Statistical Mechanics and Cybernetic Perspectives, CRC Press (1994).

[32] Phillip, M. Yelland and Lee Eunice. Forecasting Product Sales with Dynamic Linear Mixture Models.

[33] Pole,A. M. West and P.J. Harrison (1994) Applied Bayesian Forecasting and Time Series Analysis. Chapman-Hall, New York.

[34] Reid, D.J. (1969). A comparative study of time series prediction techniques on economic data. PhD Thesis, Department of Mathematics, University of Nottingham.

[35] Reid, D. J. (1975). A review of short term projection techniques. In: Gordon, H. D. (Ed.), Practical aspects of forecasting, Operational Research Society, London, pp. 8–25.

[36] Rumelhart, D. E., McClelland, J. L.. Parallel distributed processing: Explorations in the microstructure of cognition (Vol. 1). Cambridge, MA: MIT Press (1986).

[37] Sandy D. Balkin. The value of nonlinear models in the M3-Competition. International Journal of Forecasting 17 (2001) 537-584.

[38] Sakamoto, Y., Ishiguro, M., and Kitagawa G. (1986). Akaike Information Criterion Statistics. D. Reidel Publishing Company.

[39] Sharda, R., 1994. Neural networks for the MS/OR analyst: An application bibliography. Interfaces 24 (2), 116-130.

[40] Sheldon, M. Ross. Introduction to Probability Models, Eighth Edition, Academic Press (2003).

[41] Stekler, Herman. The M3-Competition: the need for formal statistical tests. International Journal of Forecasting 17 (2001) 537-584.

[42] Tashman, Leonard J. Out-of-sample tests of forecasting accuracy: an analysis and review. International Journal of Forecasting 16 (2002) 437-450.

[43] Thomas M. O'Donovan. Short Term Forecasting, An introduction to the Box-Jenkins Approach, John Wiley & Sons (1983).

[44] West, Mike and Harrison, Jeff. Bayesian Forecasting and Dynamic Models, Second Edition, Springer (1997).

[45] Widrow, B., Rumelhart, D.E., lehr, M.A., 1994. Neural Networks: Applications in industry, business and science. Communications of the ACM 37 (3), 93-105.

[46] White, H., 1998. Learning in artificial neural networks: A statistical perspective. Neural Computation 1, 425-464.

[47] Zhang, Guoqing B. Eddy Patuwo and Michael Y. Hu. Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting 14 (1998) 35-62.

APPENDIX

Appendix A. Best ARIMA Model Code (R 2.0.1)

```
Library ("MASS")      # Upload MASS Package
Library ("forecast")    # Upload Forecast Package

path1 <- "C:\\Documents and Settings\\My Documents\\sam\\m3\\M3data3003\\"
# Specify the path of the data list folder

files1 <-  read.table("C:\\Documents and Settings\\cas\\My
Documents\\sam\\m3\\monthly_list1428.csv",header=FALSE,sep=",")
# Read the monthly list



n1 <- nrow(files1)          #Get the length of the list
SMAPE <- rep(0,n1)       #Initial the matrix of the SMAPE

for (i in 1:n1){               # Set up loop for the whole list

  path2 <- files1[i,1]        #Get the name of each series
  nval1 <- files1[i,2]        #Get the  length of valid numbers of each series
  nfct1 <- files1[i,3]        #Get the length of reserved numbers of each series
  path3 <- paste(path1,path2,sep="")     # get the series' path
  X1 <- read.table(path3,header=FALSE,sep=",")    #Read in one series
  valid1 <- X1[(nval1-nfct1+1):nval1,1]    # Read in the reserved data
  train1 <- X1[1:(nval1-nfct1),1]              # Read in the training data


  fit <- best.arima(train1,d=1,D=12,max.p=3,max.q=3,max.Q=3,alpha=0.05)

  # Apply the best ARIMA model to train series 'train1'
    # Set the order of first-differencing d equal to 1
    # Set the order of seasonal-differencing, for monthly data, D equal to 12
    # Set the maximum value of p equal to 3
    # Set the maximum value of q equal to 3
    # Set the maximum value of Q equal to 3
    # Set the Level for unit-root tests used to determine the order d of differencing


  fcst1<-forecast(fit,h=nfct1)   # Use the trained model to forecast h horizon ahead

  SMAPE <- t(abs(fcst1$mean-valid1)/((fcst1$mean+valid1)/2))  # Calculate the SMAPE
  X12 <- data.frame(path2,SMAPE)  # Creates data frames to store the value of SMAPE
```

```
write.table(X12,"C:\\Documents and Settings\\cas\\My
Documents\\sam\\result\\AllLagsMAPEmonthly.csv",append=TRUE,sep=",",row.names
=FALSE,col.names=FALSE)}
# Save the results to a specify folder in CSV format
plot(forecast(fit,h=nfct1))  # plot the forecast result
```

Appendix B. ANN Code in Matlab 7.0.1

```
% Read in the file for the matches and probes.
clear all;
tic;
% Start measure elapsed time
[seriesfile,length,hold] = textread('C:\Documents and Settings\Administrator\My
Documents\sam\m3\monthly_list1428.txt','%s %d %d');
% Get the series name, series length and the number of reserve data
path1 = 'C:\Documents and Settings\Administrator\My
Documents\sam\m3\M3data3003\';
% Specify the path of the data folder
outfilename = 'C:\Documents and Settings\Administrator\My
Documents\sam\karl_nn_rand_monthly_2010_d=10.csv';
% Create the output file
fid = fopen(outfilename,'w');
% Open the output file for reading and writing
% Set the horizon and delay
nfiles = size(seriesfile,1);
% Get the size of the whole data list
horiz=18;
% Set up the horizon, yearly=6, quarterly=8, monthly=18, other=8.
delay=10;
% set up the delay, could be various, but was bounded subjuct to the length of the series.
        % for monthly maxdelay=11,  other maxdelay=46, yearly
        % delay=1,2,3,4, Maxhroiz=6,5,4,3
        % for quarterly, there is a problem, maxdelay=16-8-8=0
seriesleng=length-hold;
% Get the length of the training data
minseriesleng=min(seriesleng);
% Check the minimum length of training data of all series
neuroconfig = [delay,20,10,1];
% Set the network config, 4 layers network
% Set the size of each layer, Inputlayer = delay, hidenlayer1 = 20, hidenlayer2 = 10,
outputlayer = 1
trainalgo = 'trainlm';
% Specify the training function, ‘trainlm’ represents the Levenbery-Marquardt
algorithm. %  also could use trainrp, traingdx, traincgp, etc.
```

```
for i = 1:nfile1;          % Set up the loop for the data;
    for j=1:horiz;          % Forecasting all required horizon
       file1 = char(seriesfile(i));   % Get a series name
       path2 = [ path1, file1 ];      % Get a series
       tseries = csvread(path2);    % Read in a series
       datasize = length(i,:);        % Get the length of the series
       datahold = hold(i,:);          % Get the length the data reserved
     datavalid = datasize-datahold;  % Get the number of training data
     trgnum = datavalid+j;
     % Get the target number in the series for accuracy evaluation
     % begian Neural Network

     [P,T]=createInputTarget(tseries(1:datavalid,1),tseries(1:datavalid,1),delay,j);
     % Apply createInputTarget function to reate the input and target for training
     [pn,meanp,stdp,tn,meant,stdt]=prestd(P,T);
     % Normalize the original inputs and targets into a standard normal distribution
     % or [pn,minp,maxp,tn,mint,maxt]=premnmx(P,T); % Normalize the data in [-1,1]
     net = newff(minmax(pn),neuroconfig,{'tansig','tansig','tansig','purelin'},trainalgo);
     % Create a feed-forward backpropagation network
     % Set Parameters for NN;
     net.trainParam.goal=1e-6;   % Set up the networks goal.
     net.trainParam.show = 300;
     net.trainParam.lr = 0.2;
     % Set up the learning rate lr. If the lr is set too big, the algorithm may oscillate and
     % become unstable. If the lr is too small, the algorithm will take too long to converge.
     %Test lr=0.005, 0.01, 0.05, 0.1, 0.2, 0.25
     net.trainParam.mem_reduc=2;   % Decrease the amount of memory needed
     net.trainParam.min_grad=1e-8;  % Set the min gradient.
     net.trainParam.epochs = 2000;   % Set the max epochs
     % Apply random function to make the training randomly
     [pnran,tnran]=randomFn(pn,tn,delay);
     net=init(net);  % Initializing networks weigh and bias before training
     [net,tr] = train(net,pnran,tnran); % Train the network with random input and target
     simin=sim(net,pn);    % Simulate result
     focin=poststd(simin,meant,stdt);  % Retrun the simulation result in original units
     % or  simresults=postmnmx(an,mint,maxt);
     pnew=tseries(datavalid-delay+1:datavalid); % Pick up the data used in forecasting
     pnewn=trastd(pnew,meanp,stdp);            % Preprocess the data pnew
     anpnewn=sim(net,pnewn);                  % Simulation the preprocessed data pnew
     fcstpnew(i,j)=poststd(anpnewn,meant,stdt);
     % Return the forecasting result in original units
     if fcstpnew(i,j) < 0  % Refine the forecast value make the negative results equal to 0;
       fcstpnew(i,j) = 0;
     end;
     %madin(i,j)= MAD_nn(simresults,tseries(trgnum,1));
     % mean absolute error for each series
```

```
      smapein(i,j) = SMAPE_nn(fcstpnew(i,j),tseries(trgnum,1));
      % Symmetric mean absolute percentage error for each series
     end
     outputsmape(i,:)=cat(2,delay,smapein(i,:)); % Concatenate the output
     fprintf(fid,'%s, %1.0f, %8.4f, %8.4f, %8.4f, %8.4f, %8.4f, %8.4f, %8.4f,
    %8.4f, %8.4f, %8.4f, %8.4f, %8.4f, %8.4f, %8.4f, %8.4f, %8.4f,
    %8.4f\n',file1,outputsmape(i,:));
    % Output the SMAPE of all horizon for each series, column size should be equal to
the horizon+1
end     % End of loop
toc;     % Stop the clock of the elapsed time measuring
t=toc   % Output the elapsed time
SMAPEnn(delay,:)=mean(smapein(:));
% The mean of SMAPE of the entire seasonal data set
allfoc=cat(2,focin,fcstpnew(1,:));
%  Concatenate the last in-sample and out-sample forecasting results
figure;plot(delay+horiz:datasize,tseries(delay+horiz:datasize),'b',delay+horiz:datasize,allf
oc,'r-','linewidth',2);
% Grahp the last series all simulation result
title(sprintf('%s: %d %d %s %s',
char(seriesfile(i)),delay,horiz,trainalgo,num2str(neuroconfig)));
%  Title and legend
fclose(fid);  % Close the file after writing
```

ANN Functions:

1. CreateInputTarget Function:

```
function [in,tgt] = createInputTarget(I,T,delay,horiz)
numtrainpts = size(T,1);
ilength = numtrainpts-delay-horiz+1;
tgt = transpose(T(delay+horiz:numtrainpts,1));
   for i=1:delay;
      in(i,:)=I(i:ilength+i-1);
   end
```

Random Function:

2.
```
function  [randinput,randoutput] = randomFn(inputmatrix, outputmatrix,inputrowsize)

pnsize=size(inputmatrix,2);  % Get the loop size
rannum=randperm(pnsize);  % Get the random number list
pnran=zeros(inputrowsize,pnsize);
% Get the matrix frame of random input and   output
 tnran=zeros(1,pnsize);
         for k = 1:pnsize
```

```
    randinput(:,k)=inputmatrix(:,rannum(k));
    % Randomly rearrange the input and  the target
    randoutput(:,k)=outputmatrix(:,rannum(k));
  end
```

3. SMAPE function

```
function result = SMAPE_nn(fcst,act)
res = 100*abs(fcst - act)/((fcst+act)/2);
result = res;
```

Appendix C. DLM Code in Matlab 7.0:

```
% Read in the file for the matches and probes.
clear all;
tic;
[seriesfile,length,hold] = textread('C:\Documents and Settings\Owner\My
Documents\asheng\school\Thesis stuff\Mdata\m3\monthly_list1428.txt','%s %d %d');
path1 = 'C:\Documents and Settings\Owner\My Documents\asheng\school\Thesis
stuff\Mdata\m3\M3data3003\';
outfilename = 'C:\Documents and Settings\Owner\My Documents\asheng\school\Thesis
stuff\results\DLM_yearly.csv';
fid = fopen(outfilename,'w');   % Open the file for reading and writing

nfiles = size(seriesfile,1);  % Get the size of the whole data list
horiz=18;   % Set up the horizon, yearly=6,quarterly=8,monthly=18,other=8.

% Set Parameters for DLM;
Ft = [ 1; 0];    % Sensitive, possible [1 0],[1,1],[0 1], second number control forecast
mean
Gt = [ 1 0; 1 1];     % The third has to be zero, the others not sensitive
Ct = [1 1; 0 1]*100000000;   % Sensitive, affect the vibration especially the at the
beginning when pick huge or extremely small value
% W1 = eye(2,2)*1000000;   %  Sensitively contral the vibration  when the number is
biger enought.

    W1parcand =
[.0000001,.000001,.00001,.0001,.001,.1,10,100,10000,10000,100000,1000000,10000000
,100000000];
    wtOptCand =
[.00000001,.000001,.0001,.001,.1,1,10,100,1000,10000,10000,1000000];
    dtOptCand = [.00000001,.000000,.0001,.001,.1,1,10,100,1000,10000,1000000];
    StOptCand =
[.0001,.001,.1,10,100,10000,10000,100000,1000000,10000000,100000000];
```

```
    wtO = 1;
    dtO = 1;
    StO = 1;
    DLMMSEinit = 100000000; % Initial the MSE with a huge value

for i = 3;  % for the entire data set use i=1:nfiles

    file1 = char(seriesfile(i));    % Get a series name
    path = [ path1, file1 ];        % Get a series
    tseries = csvread(path);        % Read in a series
    datasize=length(i,:);           % Get the length of the series
    datahold=hold(i,:);             % Get the length the data hold
    datavalid= datasize - datahold;    % The number of data could be used in forecasting
    trgnum=tseries(datavalid+1:datasize,1);
    %trgnum= datavalid + horiz;       % Set the target number in the series for accuracy
evaluation
    mt = [tseries(1);0];

   % Optimize W1, wt, dt, and St
   for W1C = W1parcand; W1 = eye(2,2)*W1C;
    for wt = wtOptCand;
     for dt = dtOptCand;
      for St = StOptCand;
       DLMMSE = DLMFNMSE(tseries(1:datavalid),Ft,Gt,mt,Ct,dt,St,W1,wt);
        while DLMMSE < DLMMSEinit   % Get the min MSE
          DLMMSEinit = DLMMSE;
          wtO = wt;
          dtO = dt;
          StO = St;
          W1O = W1C;

        end
      end
     end
    end
   end
wtO % Output the value of the optimal parameter
dtO
StO
W1O
W1=eye(2,2)*W1O;

      % begin DLM
      mt = [mean(tseries(1:2));mean(tseries(1:2))]; %initialize the first value
      [f,f1] = DLMFN2(tseries(1:datavalid),Ft,Gt,mt,Ct,dtO,StO,W1,wtO,horiz); %call
the DLM function
```

```
        finleng=size(f,2);
        smapein = SMAPE_DLM(f1,trgnum'); % Symmetric mean absolute percentage
error for each serie
        outputsmape(i,:)=cat(2,horiz,smapein);

fprintf(fid,'%s, %d, % 8.4f, % 8.4f, % 8.4f, % 8.4f, % 8.4f, % 8.4f, % 8.4f, % 8.4f, %
8.4f, % 8.4f, % 8.4f, % 8.4f, % 8.4f, % 8.4f, % 8.4f, % 8.4f, % 8.4f, %
8.4f\n',file1,outputsmape(i,:));      % The output size should equal to the horizon

end
        allfoc=cat(2,f(1:finleng-1),f1);  % Concatenate the last in-sample and out-sample
forecasting results for the figure
        figure;plot(2:datasize,tseries(2:datasize),'b-',2:datasize,allfoc,'r-','linewidth',2);
% Grahp the last all series simulation
        title({['yearly-DLM-Char: ', file1]; 'blue:actual  red:forecast'});

   toc;
   fclose(fid);
   t=toc
```

DLM Functions:

1. DLMFNMSE Function

```
function result = DLMFNMSE(z,Ft,Gt,mt,Ct,dt,St,W1,wt1)
% Use this function to train and optimize.
wt = eye(2,2)*wt1;   %wt = eye(2,2)*wt1*(1-delta)/delta;
W = eye(2,2)*W1;  % ?
Rtk = Ct*1;          %Rtk = Ct*1/delta;
zlen = size(z);
zlen = zlen(1);
ft = mean(z(1:2));
et2 = 0;

for i=1:zlen
    Rt = Gt*Ct*Gt' + W;
    Qt = Ft'*Rt*Ft + St;
    et = z(i) - ft;
    dt = dt + St*et^2/Qt;
    At = Rt*Ft/Qt;
    zi=z(i);
    at = Gt*mt;
    ft = Ft'*at;
    mt = at + At*et;
    St1 = St*1;
```

```
    St = St + St/i*(et^2/Qt - 1);
    Ct = St/St1*(Rt-At*At'*Qt);
    Rtk = Gt*Rtk*Gt + wt;
    Q1 = Ft'*Rtk*Ft + St;
    et2 = et2 + et^2;
end
result = et2;
```

2. DLMFUN2 Function:

```
function [fin,fout] = DLMFN2(z,Ft,Gt,mt,Ct,dt,St,W1,wt1,horizon)
% Use this function to train and optimize.
wt = eye(2,2)*wt1; %wt = eye(2,2)*wt1*(1-delta)/delta;
W = eye(2,2)*W1;
Rtk = Ct*1;        %Rtk = Ct*1/delta;
et = 0;
zlen = size(z);
zlen = zlen(1);
ft = mean(z(1:2));
f(1)=ft;
for i=1:zlen
    Rt = Gt*Ct*Gt' + W;
    Qt = Ft'*Rt*Ft + St;
    et = z(i) - ft;
    dt = dt + St*et^2/Qt;
    At = Rt*Ft/Qt;
    zi=z(i);
    at = Gt*mt;
    ft = Ft'*at;
    mt = at + At*et;
    St1 = St*1;
    St = St + St/i*(et^2/Qt - 1);
    Ct = St/St1*(Rt-At*At'*Qt);
    Rtk = Gt*Rtk*Gt + wt;
    Q1 = Ft'*Rtk*Ft + St;
    f(i+1)=ft;
end
fin=f(1:i);
for i=1:horizon
fout(i) = Ft'*Gt^(i)*mt;
end
```

3. SMAPE_DLM Function:

```
function result = SMAPE_DLM(fcst,act)
   for i = 1:size(fcst,2);
   res(1,i) = 100*abs(fcst(i) - act(i))/((fcst(i)+ act(i))/2);
   end
result = res;
```

Appendix C. SAS Code for One-way ANOVA

```
/* Import data sheet from excel*/
PROC IMPORT OUT= WORK.other_h1
          DATAFILE= "C:\Documents and Settings\Owner\My Documents\
          asheng\school\Thesis stuff\ANOVA_raw\other\each horizon\other_H1.xls"
          DBMS=EXCEL2000 REPLACE;
    RANGE="Sheet1$";
    GETNAMES=YES;
RUN;

/* Run mix ANOVA model at every forecast horizon*/
PROC MIXED DATA=other_h1;
  CLASS paradigm series;
  MODEL SMAPE=paradigm;
  RANDOM series;
  lsMEANS paradigm/pdiff;
TITLE "Mixed ANOVA for other data, h=1";
RUN;
```