Time-series Forecasting of Foreign Exchange Rates using Recurrent Neural Networks

A Comparative study with Statistical Models

Yip Chi Kin

MSc in Applied Mathematics for Science & Technology

THE HONG KONG
POLYTECHNIC UNIVERSITY

1998

Acknowledgment

I would like to thank Dr. Li Leong Kwan for his supervision and guidance during the course of this dissertation, and also Dr. James Liu for helpful introduction to the subject of neural networks. Lastly Dr. M.W. Mak gave some valuable comments.

Time-series Forecasting of Foreign Exchange Rates using Recurrent Neural Networks

A Comparative study with Statistical Models

Abstract of dissertation:

Time series are a special form of data where past values in the series may influence future values, depending on the presence of underlying deterministic forces. Whilst linear models, such as those based on regression techniques, have been the basis of traditional statistical forecasting models, their drawbacks have led to increased activity in nonlinear modeling. Neural networks are nonlinear models that can be trained to map past and future values of time series, and thereby extract hidden structure and relationships governing the data. Economy is a dynamic system that inherits non-linearity through long term trends, seasonal patterns, cyclical movements, and irregular factors.

This dissertation focus on the application of Recurrent Neural Networks (RNN) to predict Foreign Exchange (FX) time series. The RNN module performance is evaluated in terms of Direction matching, MAE, MAPE, and System stability using FX rate data set. This data set includes four currencies against United States Dollar (USD) as indicators that span over four years in daily price. The predictions are combined trading fuzzy decision making module to generate buy-sell-hold recommendations for the entire list of FX rates on a daily basis. Finally, a simple exponential smoothing (SES) model would be compared to RNN prediction. And the results are presented and concluded with a discussion on the ongoing research direction.

Submitted by Yip Chi Kin (Student No: 95613987G)

for MSc in Applied Mathematics for Science and Technology

at The Hong Kong Polytechnic University in January, 1998.

CONTENTS

Chapter One

SPECIFICATIONS

- Dissertation Aims & Brief
- Assumptions & Limitations
- Outline of Dissertation

1.1 DISSERTATION AIMS

Predictability is fundamental to the modern scientific view of nature. This dissertation considered time series prediction using recurrent neural networks (RNN) to forecast Foreign Exchange (FX) rates in capital market. The results focus on *US Dollars* (USD) against *UK Pound* (GBP), *Japanese Yen* (JPN), *Deutsche Mark* (DEM), and *Swiss Franc* (CHF) in **daily** behavior. The aim of this study is to investigate the potential of machine learning techniques for modeling **rise** / **fall** movements in foreign exchange rates, and make a decision for Selling, Buying or holding the currency.

1.2 ASSUMPTIONS & LIMITATIONS

1.2.1 Assumptions

(1) Currency prices are highly chaotic nature, and notoriously difficult to model with standard statistical methods such as regression.

(2) All the foreign exchange data are the closing price of markets with no time lagging.

(3) The global net turnover mainly affected by USD, DEM, GBP, CHF, & JPY, and neglect the interest rates.

(4) Global net turnover in the world's foreign exchange market is estimated to be about US$1 trillion per business day [Bi93].

(5) The transaction cost for buying and selling of FX is 1%.

1.2.2 Limitations

(1) Cannot collect daily foreign exchange option prices for input data set.

(2) Cannot consider the daily foreign exchange turnover for each currency.

(3) Real time effect for each currency forecasting.

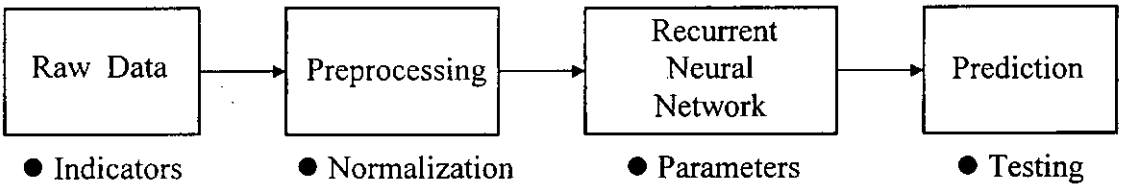
(4) The noise of the exchange rates system cannot be determined.

(5) Time constraint for doing the dissertation.

1.3 OUTLINE OF DISSERTATION

This dissertation describes how modern machine learning can be used to forecast short / long-term movements in exchange rates, and produces models suitable for use in trading. It compares the results achieved by some different techniques.

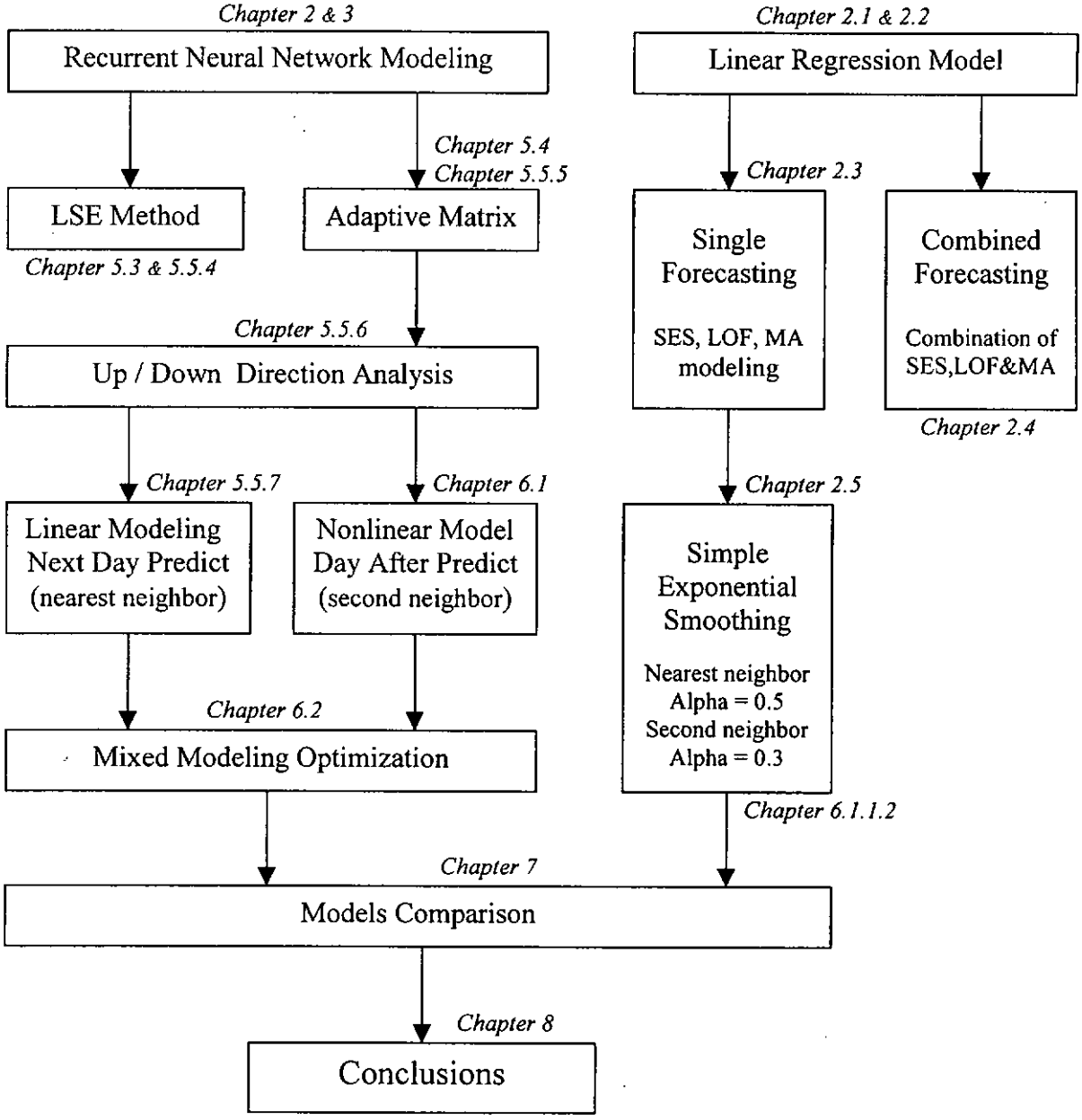
Forecasting Modeling



- This dissertation contains a total of nine chapters. In *chapter 2* some traditional statistical methods (e.g. such as **simple exponential smoothing**) are used to predict the currency rates. And then choose the best one to compare with the RNN forecasting
- In *chapter 3* there is a brief description of the RNN modeling techniques use in this work. Forecasting method is showed in above block diagram.
- In *chapter 4* **moving windowing** simulation technique is used. All implementations of the software details are described as pseudo-code and the block diagrams.
- The *chapter 5* the main factors to affect the results is preprocessing the data. And also the parameters and learning system should be selected by some strategies, which are described in this chapter. The main numerical optimizations of **Least square error** (LSE) and **Adaptive pseudo-inverse matrix** (APIM) are used.
- The *chapter 6* using **Taylor expansion** and **double interval** method to predict the second neighbor price (*day after price*). Analysis the system by mixed optimization *(linear & nonlinear)* to decide the sell / hold / buy marketing strategies.
- The *chapter 7* the results using **fuzzy decision** to compared the traditional modeling and RNN modeling. And determine the best parameters for forecasting FX.

■ Finally, *chapter 8* draws some conclusions from the work described in this dissertation, and suggest some future directions for the use of machine learning techniques in this area.

Block Diagram of Processes to make the conclusions



The above block diagram describes the processes with chapters layout in order to help the reader to understand the dissertation.

Chapter Two

STATISTICAL MODEL

- Overview
- Forecasting Criterions
- Forecasting Methods
- Combined Regression Model
- Models Comparison

2.1 OVERVIEW

2.1.1 Analysis Time Series

The raw data are plotted in a time series graph as shown in *Appendix A*. This graph shows the daily high low of FX rate of four currencies independently from 30-6-1993 to 30-6-1997. In the chart, the vertical axis represents the numerical value of the FX rate with USD, while the horizontal axis depicts the passage of time. This graph only conveys a sense of how a particular FX rate is changing over a fairly long period of time [Hs88]. To gain an insight into the time series should be modeled in mathematical form [Ti89]. This chapter illustrates some statistical modeling techniques for smoothing time series data to remove some effects of noise.

2.1.2 Data Processing

Normally, the raw data are used in traditional statistical method. It is easier to see the results by fitting the time series curves. But a more meaningful interpretation of what the FX market doing is obtained by observing a trend direction (up / down). Therefore, a common preprocessing method "*Relative Variable Difference*" (RVD) strategy is used as well.

The equation of RVD preprocessing is $X(t+1) - X(t)$. The given data set has 1041 observations from 30-6-1993 to 30-6-1997. The last 40 data use for forecasting checking. There are only 1001 observations needed for preprocessing, and the results have 1000 observations for finding the variables of models. The first data is omitted because it has no the previous observation for deduction of preprocessing.

There will be two methods of modeling used in this chapter. They are Single Forecasting and Combination Forecasting. Four currencies FX rate forecasted individually. It assumes independently.

2.2 FORECASTING CRITERIONS

2.2.1 Mean Forecast Error (ME)

$$\mathrm{ME} = \frac{1}{n}\sum_{t=m-n+1}^{m}(Y_t - \hat{Y}_t) \qquad (2.2.1)$$

2.2.2 Mean Square Error (MSE)

$$\mathrm{MSE} = \frac{1}{n}\sum_{t=m-n+1}^{m}(Y_t - \hat{Y}_t)^2 \qquad (2.2.2)$$

2.2.3 Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{n}\sum_{t=m-n+1}^{m}\left|Y_t - \hat{Y}_t\right| \qquad (2.2.3)$$

2.2.4 Mean Absolute Percentage Error (MAPE)

$$\mathrm{MAPE} = \frac{1}{n}\sum_{t=m-n+1}^{m}\left|\frac{Y_t - \hat{Y}_t}{Y_t}\right| \times 100\% \qquad (2.2.4)$$

where $\hat{Y}$ = the predicted value by forecasting
Y = actual data
m = the last day data from the data set
n = length of data set (daily)

Four common criterions are introduced. The main purpose of those criterions is comparison of modeling performance, which included statistics or RNN modeling. According to the following table, MAE criterion is the best method to analysis the results of the FX rate forecasting. Therefore, MAE is the main strategy of modeling selection in this chapter.

	ME	MSE	MAE	MAPE
Simple Computation	5	4	4	2
Actual behavior of Data	1	3	4	5
Independent of Data Set	1	4	5	2
Direction Matching (Up/Down)	1	4	5	2
Model Comparison	1	3	5	3

NB. 1:Worst 2:Bad 3:Fair 4:Good 5:Excellent

2.3 FORECASTING METHODS

2.3.1 Methodology

2.3.1.1 *Single Forecasting*

There are three simple single forecasting in this dissertation. They are Simple Exponential Smoothing (SES), Last Observation Forecast (LOF), and Moving Average (MA). Three models are used to predict the results separately. The parameters of *Alpha* and *Periods* were found by the data set of exchange rate USD/GBP only. Detailed model algorithms are formulated in *Chapter 2.3.2 to 2.3.4.*

2.3.1.2 *Combined Forecasting*

From the above SES, LOF, and MA forecasting results, we have $\mathbf{Y}_{SES}(t+1)$, $\mathbf{Y}_{LOF}(t+1)$ and $\mathbf{Y}_{MA}(t+1)$ respectively. A further four new forecast results can be derived from a combination of two single forecasting results, which work as the preprocessing function. Here is an example of modeling, $\mathbf{X}(t+1) = B_0 + B_1\mathbf{Y}_{SES}(t+1) + B_2\mathbf{Y}_{MA}(t+1)$. This newly combined forecast approach is a **Multiple Linear Regression Model**. The weights (B_0, B_1 and B_2) of regression model can be obtained from the following matrix equations.

$$\mathbf{Y} = \begin{bmatrix} 1 & Y_{11} & Y_{21} \\ 1 & Y_{12} & Y_{22} \\ 1 & Y_{13} & Y_{23} \\ \vdots & \vdots & \vdots \\ 1 & Y_{1n} & Y_{2n} \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix},$$

Hence: $\mathbf{b} = (\mathbf{Y'Y})^{-1}\,\mathbf{Y'X}$

The *Multiple Linear Regression* equation is $\hat{Y} = b_0 + b_1 Y_1 + b_2 Y_2$

Finally, the predict value of system is $X(t+1) = b_0 + b_1 Y_1 + b_2 Y_2$

The equation rewrote as $\mathbf{X}(t+1) = B_0 + B_1\mathbf{Y}_{SES}(t+1) + B_2\mathbf{Y}_{MA}(t+1)$

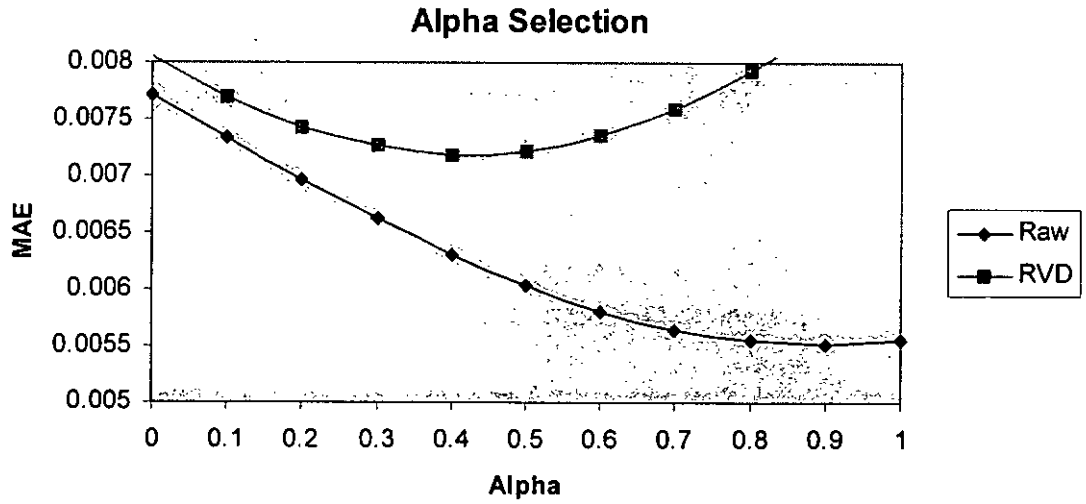
2.3.2 Simple Exponential Smoothing (SES)

Exponential smoothing is a convenient way of expressing the forecast $\hat{X}$ in terms of exponentially smoothed statistics. If the time series is not flat but has linear trends, second order exponential smoothing is used. This can model the trend and hence exponentially smooth the de-trended data set. The second order exponential smoothing model is given by:

$$\hat{X}(t+1) = \alpha X(t) + (1-\alpha) X(t-1) ; \qquad (2.3.2)$$

where $\hat{X}(t+1)$ is the predicted value of time t and $t-1$;
$X(t)$ is actual value at time t;
α is an exponential time constant, $0 \leq \alpha \leq 1$.

The statistic $X(t-1)$ is called the second smoothed statistic and is a smoothing of the outlier values. $X(t-1)$ gives an indication of the trend of the $X(t)$ over time. Hence, its inclusion in the model accounts for a linear trend of $X(t)$ with time.



The estimate values for the parameter of α is found by the comparison of MAE of the above model. The MAE results of SES model are found by GBP currency only as shown above. According to the results in the above figure, the single forecast methods do not need to preprocess the raw data. The predicted values of $X(t+1)$ are tabulated in *Table 2.3.2*.

Table 2.3.2

	Raw Data (α=0.9)				Relative Variable Difference (α=0.4)			
	ME (10^{-6})	MSE (10^{-6})	MAE (10^{-6})	MAPE %	ME (10^{-6})	MSE (10^{-6})	MAE (10^{-6})	MAPE %
USD/DEM	−17.5	16.4	2830.0	0.437	−6.65	24.6	3650.0	237
USD/BGP	165.0	61.6	5520.0	0.353	−7.3	90.8	7180.0	242
USD/CHF	23.8	31.2	3830.0	0.496	−9.3	46.4	4990.0	237
USD/JPY	0.532	0.0052	46.9	0.483	0.0338	0.00806	62.2	228

According to MAPE results, SES is not a good model for RVD to predict the currencies up or down. The SES model performance could not consider the MAE results only. It is because the currency rates change very little, such as USD/JPY. For example, MAE of USD/JPY is 46.9×10^{-6}, and the MAPE is almost the same as other currencies results. Therefore, the MAE is a useful tool for modeling comparison with same data set inputs. In the following chapters, this argument should be used in the same way.

2.3.3 Last Observation Forecast (LOF)

This is a very simple modeling. The predicted value equals to the last value of the observation. The formula is shown as following:

$$\hat{X}(t) = X(t-1) \qquad (2.3.3)$$

Where $\hat{X}$ = predicted value
X = the original observations

Table 2.3.3

	Raw Data				Relative Variable Difference			
	ME (10^{-6})	MSE (10^{-6})	MAE (10^{-6})	MAPE %	ME (10^{-6})	MSE (10^{-6})	MAE (10^{-6})	MAPE %
USD/DEM	−12.3	16.4	2830.0	0.438	−6.33	34.4	4390.0	300
USD/GBP	166.0	63.3	5570.0	0.357	−22.1	146.0	8830.0	319
USD/CHF	23.1	30.9	3830.0	0.495	−6.14	63.1	5890.0	317
USD/JPY	−0.59	0.0052	47.3	0.486	0.0154	0.0112	72.5	287

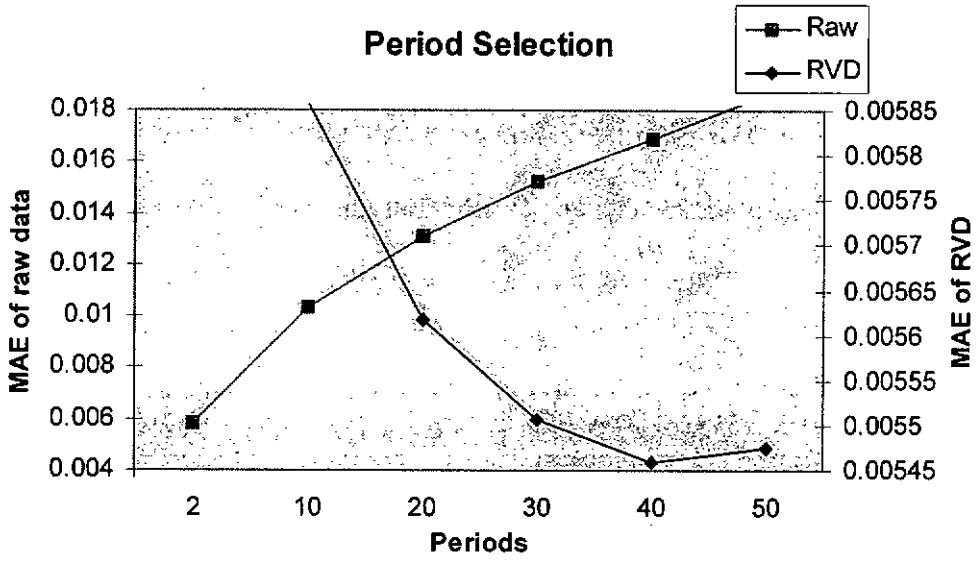
Although the results are not too bad, obviously the modeling cannot predict the actual values of the system. This method will combine others to be a preprocessing data function only for the multi-regression modeling in *Chapter 2.4.1*.

2.3.4 Moving Average (MA)

The moving average strategy is quite simple. Given a time series on t points (X_1, … X_t).

$$\hat{X}_{t+1} = \frac{1}{n}\sum_{i=1}^{n} X_{t-i+1} \qquad (2.3.4)$$

Where $\hat{X}$ is the predicted value
n is the period; 2,10,20,30,40&50 are used



The best period n = 2 is selected for raw data and n = 40 after data preprocessing, means that the trend does not dependent on the long past periods.

Table 2.3.4

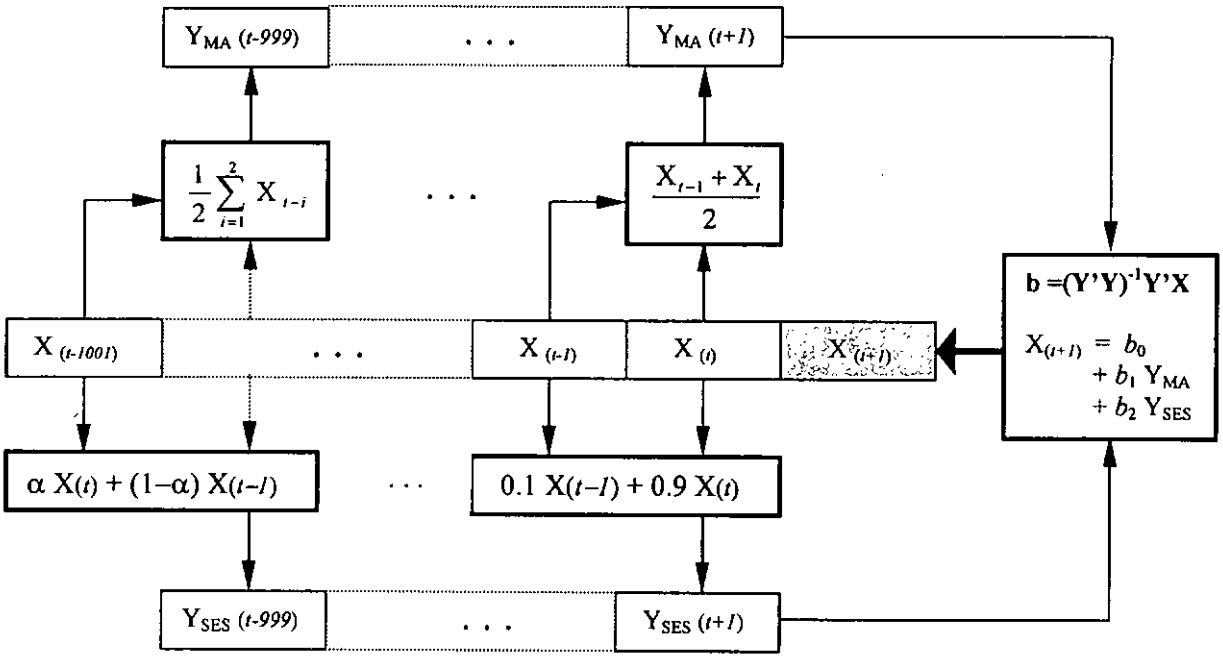
	Raw Data *(n = 2)* $\hat{X}_t = \frac{1}{2}\sum_{i=1}^{2} X_{t-i} = \frac{X_{t-1}+X_{t-2}}{2}$				Relative Variable Difference *(n = 40)* $\hat{X}_t = \frac{1}{40}\sum_{i=1}^{40} X_{t-i} = \frac{X_{t-1}+\cdots+X_{t-40}}{40}$			
	ME (10^{-6})	MSE (10^{-6})	MAE (10^{-6})	MAPE %	ME (10^{-6})	MSE (10^{-6})	MAE (10^{-6})	MAPE %
USD/DEM	-69.8	19.7	3100.0	0.477	-39.5	16.8	2850	112
USD/GBP	186.0	64.6	5810.0	0.371	-18.9	60.0	5460	114
USD/CHF	-30.7	38.5	4290.0	0.551	-49.2	32.0	3920	117
USD/JPY	-1.19	0.00597	51.1	0.525	0.166	0.00515	47	119

Depending on the above results of relative lengths of two MA strategies (Long MA and Short MA). The LMA gives a much smoother curve than the SMA. The decision rule for taking a position in FX market is straightforward. The SMA will be chosen in this dissertation.

2.4 COMBINED REGRESSION MODEL

2.4.1 Combined Regression Model Structure

The figure below shows the combined forecasting method, which is an example of MA & SES combined Regression model. Firstly, the MA & SES methods become some preprocessing function in order to give two new data sets for multiple regression modeling. The parameters of MA & SES are conformed by the previous *Chapter 2.3.2 & 2.3.4*, the SMA and α=0.9 are used. The raw data set contains 1001 observations, and two new data sets have 1000 input data only. Only one indicator in the system forecasts the result. Therefore, four separate linear models are needed to forecast four different FX rates.



There are four combinations of LOF & MA, MA & SES, LOF & SES and LOF & MA & SES modeling. This is one of the combined modelings, which can smooth the time series in order to filter out the noise. According to the above prediction system, the equation of predicted value would be $\mathbf{X}(t+1) = B_0 + B_1\mathbf{Y}_{MA}(t+1) + B_2\mathbf{Y}_{SES}(t+1)$.

2.4.1.1 *LOF & MA Regression Model*

$$\mathbf{X}(t+1) = B_0 + B_1\mathbf{Y}_{LOF}(t+1) + B_2\mathbf{Y}_{MA}(t+1)$$

$$= 0.0023 + 0.9243\ \mathbf{Y}_{LOF}(t+1) + 0.0721\ \mathbf{Y}_{MA}(t+1) \quad \text{for USD/DEM}$$

$$= 0.0152 + 0.6907\ \mathbf{Y}_{LOF}(t+1) + 0.2996\ \mathbf{Y}_{MA}(t+1) \quad \text{for USD/GBP}$$

$$= 0.0028 + 0.9797\ \mathbf{Y}_{LOF}(t+1) + 0.0167\ \mathbf{Y}_{MA}(t+1) \quad \text{for USD/CHF}$$

$$= 0.0000 + 0.8321\ \mathbf{Y}_{LOF}(t+1) + 0.1670\ \mathbf{Y}_{MA}(t+1) \quad \text{for USD/JPY}$$

(2.4.1.1)

2.4.1.2 *MA & SES Regression Model*

$$\mathbf{X}(t+1) = B_0 + B_1\mathbf{Y}_{MA}(t+1) + B_2\mathbf{Y}_{SES}(t+1)$$

$$= 0.0023 - 0.1590\ \mathbf{Y}_{MA}(t+1) + 1.1554\ \mathbf{Y}_{SES}(t+1) \quad \text{for USD/DEM}$$

$$= 0.0152 + 0.1269\ \mathbf{Y}_{MA}(t+1) + 0.8633\ \mathbf{Y}_{SES}(t+1) \quad \text{for USD/GBP}$$

$$= 0.0028 - 0.2282\ \mathbf{Y}_{MA}(t+1) + 1.2247\ \mathbf{Y}_{SES}(t+1) \quad \text{for USD/CHF}$$

$$= 0.0000 - 0.0410\ \mathbf{Y}_{MA}(t+1) + 1.0401\ \mathbf{Y}_{SES}(t+1) \quad \text{for USD/GBP}$$

(2.4.1.2)

2.4.1.3 *LOF & SES Regression Model*

$$\mathbf{X}(t+1) = B_0 + B_1\mathbf{Y}_{LOF}(t+1) + B_2\mathbf{Y}_{SES}(t+1)$$

$$= 0.0023 + 0.6359\ \mathbf{Y}_{LOF}(t+1) + 0.3605\ \mathbf{Y}_{SES}(t+1) \quad \text{for USD/DEM}$$

$$= 0.0152 - 0.5078\ \mathbf{Y}_{LOF}(t+1) + 1.4981\ \mathbf{Y}_{SES}(t+1) \quad \text{for USD/GBP}$$

$$= 0.0028 + 0.9129\ \mathbf{Y}_{LOF}(t+1) + 0.0835\ \mathbf{Y}_{SES}(t+1) \quad \text{for USD/CHF}$$

$$= 0.0000 - 0.1641\ \mathbf{Y}_{LOF}(t+1) + 0.8350\ \mathbf{Y}_{SES}(t+1) \quad \text{for USD/JPY}$$

(2.4.1.3)

The coefficients of *equation* *(2.4.1.1)* to *(2.4.1.3)* are calculated using program REGRESS.m, which included in the disk for this dissertation. The above equations have little improved with a constant term, except USD/JPY. For USD/JPY currency rate, B_0 is equal to zero, because USD/JPY FX rate is changed slightly in the raw data.

From the *equations* (*2.4.1.1*), (*2.4.1.2*), & (*2.4.1.3*), Three different kinds of combined regression models have same criterion results and tabulated in *Table 2.4.1*. It means that any one of two combinations does not affect the forecasting results, even they have different equations and coefficients.

Table 2.4.1

	ME	MSE	MAE	MAPE
USD/DEM	–0.000294	0.00000052	0.0028	0.4776
USD/GBP	0.0019	0.00000000114	0.0058	0.3550
USD/CHF	–0.0000737	0.00000168	0.0038	0.5474
USD/JPY	0.0000232	0.00000000028	0.0000692	0.8010

According to *Table 2.4.1*, the system stability of combined modeling does not perform very well. The MAPE is fluctuated up to 200%, see results of USD/GBP and USD/JPY. This implies that the MAPE of model is dependent upon raw data set. Therefore, the combined regression modeling may not be a good forecasting system.

2.4.1.4 *LOF & MA & SES Regression Model*

$$\mathbf{X}(t+1) = B_0 + B_1\mathbf{Y}_{LOF}(t+1) + B_2\mathbf{Y}_{MA}(t+1) + B_3\mathbf{Y}_{SES}(t+1)$$

$$= 0.0023 - 2.7637\,\mathbf{Y}_{LOF}(t+1) - 0.4380\,\mathbf{Y}_{MA}(t+1) + 0.9746\,\mathbf{Y}_{SES}(t+1) \quad \text{for USD/DEM}$$

$$= 0.0152 + 2.5576\,\mathbf{Y}_{LOF}(t+1) + 0.7268\,\mathbf{Y}_{MA}(t+1) - 1.4492\,\mathbf{Y}_{SES}(t+1) \quad \text{for USD/GBP}$$

$$= 0.0028 + 1.2500\,\mathbf{Y}_{LOF}(t+1) + 0.5078\,\mathbf{Y}_{MA}(t+1) - 8.3828\,\mathbf{Y}_{SES}(t+1) \quad \text{for USDCHF}$$

$$= 0.0000 + 0.8118\,\mathbf{Y}_{LOF}(t+1) + 0.1606\,\mathbf{Y}_{MA}(t+1) + 0.3506\,\mathbf{Y}_{SES}(t+1) \quad \text{for USD/JPY}$$

	ME	MSE	MAE	MAPE
USD/DEM	1.8813	0.0868	1.8813	322.3447
USD/GBP	–1.3827	0.0494	1.3827	84.3203
USD/CHF	5.3155	0.7014	5.3155	761.8876
USD/JPY	–0.0028	0.00000022	0.0028	32.0317

The results given by above equations, the MAE and MAPE are larger than other models. It can conclude that the model will be omitted in this dissertation.

2.5 MODEL COMPARISON

2.5.1 MAPE Criterion Analysis

According to *Table 2.3.2*, *Table 2.3.3*, *Table 2.3.4* & *Table 2.4.1*, combined the results of MAE & MAPE together and tabulated as following. The MAE criterion does not use for modeling selection, it becomes the reference for discussion only.

	SES		LOF		MA		MA & SES	
	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE
USD/DEM	0.00283	0.437	0.00283	0.438	0.00310	0.477	0.0028	0.4776
USD/GBP	0.00552	0.353	0.00557	0.357	0.00581	0.371	0.0058	0.3550
USD/CHF	0.00383	0.496	0.00383	0.495	0.00429	0.551	0.0038	0.5474
USD/JPY	0.00047	0.483	0.00047	0.486	0.00051	0.525	0.00007	0.8010
Total Error	0.01265	*1.769*	0.01270	1.776	0.01371	1.924	0.01247	2.1810

LOF has good result because the FX rates change very little daily. But the system should be unstable and cannot predict trend direction. MA is a simple model, which can simply predict a series without large upward or downward trend. Obviously, the combined model has less MSE. The best MSE of single forecast model is SES, but there have extreme difference of MSE by using different models.

According to above MAE results, it is not easy to select a suitable model. The FX time series is a fluctuation series by visual observation. Therefore LOF and MA are not a good single forecast model in this time series. It also fits into our common knowledge. If a combined forecast modeling techniques is used, the MAPE is unstable as shown in the above table. The model time series fluctuate not too much, so the SES is the best model.

According to above MAPE criterion, the best model for four FX rates is the Simple Exponential Smoothing Model. Therefore, the combined forecast is not a good prediction technique, even they normally got a less MAE and very close to the SES model. Finally, the SES of single forecast model will be compared with RNN model.

2.5.2 Summary

The SES single modeling is selected. Obviously, this forecasting equation is a time series equation. The time series cannot omit any outliers because it will break the time series, and make the data set series discontinuously.

By observation, the series has not a seasonal effect in one-year period or cyclic components. But the series fluctuate very large. So the preprocessing of data should be needed. If the time series is neither constant nor linear with time, it is best to use a triple exponential smoothing model. Higher-order exponential smoothing models exist but the difficulties in computing the forecasting equation for models of an order higher than three are considerable.

In addition, linear statistical models are well established and useful tools in the quantitative analysis. But the first target of this work is to show that the in-sample performance of the network gives a better fit than linear statistical regression (here is SES model). The in-sample performance of the RNN network is important because it determines its convergence ability and sets a target of feasible out-of-sample performance, which can be achieved by fine-tuning the network parameters and training discipline. Those processes will be described in following chapters.

Chapter Three

ALGORITHMS

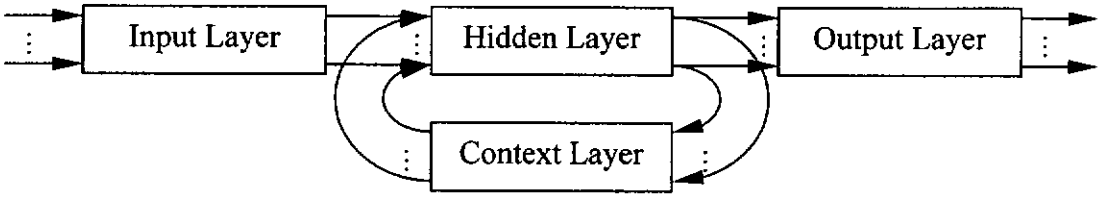
- Taxonomy
- Recurrent Neural Network
- Network Architecture
- Learning Algorithms

3.1 TAXONOMY

A *recurrent neural network*, also called a *feedback network*, is one in which self-loops and backward connections between nodes are allowed. One of the consequences of these connections is that dynamical behavior not possible with strictly feed-forward networks, such as limit cycles and chaos, can be produced with recurrent networks. Recurrent networks with symmetric weight connections always converge to a stable state as we have seen in Hopfield networks.

Unlike static feed-forward networks, RNNs exhibits dynamic behavior. They can perform mappings that are functions of time and / or space converge to one of a number of limit points. As this result, RNNs are capable of learning temporal pattern sequences, that is, sequences of patterns that are context or time dependent.

Some researchers, including [El91] and [Se91], have experimented with a class of networks having only partial feedback, which they call simple recurrent networks (SRN). In a SRN, the outputs of the hidden layer are allowed to feedback onto itself through a buffer or "context" layer. These are the only feedback connections in the network and weights from the hidden layer to the context layer are constant values. All other connections are feed-forward with adjustable weights.



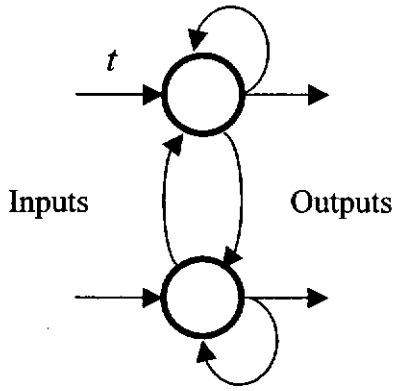
RNN has been used in a number of interesting applications including associative memories, control, optimization, forecasting and the generation of pattern sequences. The above diagram shows a simple RNN, which could be perform computational tasks equivalent to finite state automata [El91] as well as more general *Turing Machines* [Wi89].

3.2 RECURRENT NEURAL NETWORK

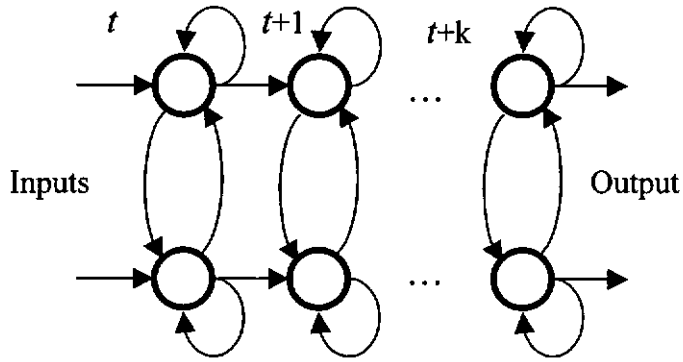
3.2.1 Fully Connected RNN

It was claimed in the above that RNN is generalizations of feed-forward networks. To show that they are in fact generalizations, one can always derive an equivalent *multi-layer feed-forward network* (MLFF) for any RNN. They are equivalent in that the two networks exhibit the same behavior. This can be accomplished through an unfolding-in-time process, where each time step t of the RNN corresponds to an additional layer of the MLFF. An example of the MLFF equivalent of simple two-unit fully connected RNN is illustrated in the figure below.

Simple Fully Connected RNN



Multi-layer Feed-forward Network

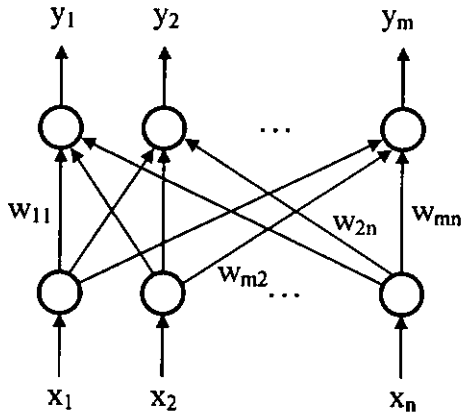


The MLFF equivalent network of the fully connected two-unit RNN has identical weights for all layers (the weights are the same as those of the two-unit RNN), but distinct perceptron outputs on each of its successive layers. Such a network can be taught to solve the nonlinear XOR problem, an impossible task for any MLFF network with only two perceptrons. Clearly, a form of error back-propagation can be used to train such networks.

3.2.2 Recurrent Signal Propagation

As illustrated in figure below, such a RNN takes an input pattern vector **x**, and produces an output pattern vector **y** by propagating **x** through its internal connections and transforming the resulting activation through the application of activation function. The process of computing the output pattern is analogous to a **vector-matrix multiplication**. The computation is performed by this simple network mathematically as $\mathbf{y}(t) = \Gamma[\mathbf{W}\mathbf{x}(t)]$. Where $\Gamma[\ ]$ represent the application of the activation function to each component of the vector-matrix multiplication between the input pattern x(t) and the weight matrix **W**.

Specifically, $\mathbf{W}=\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$.

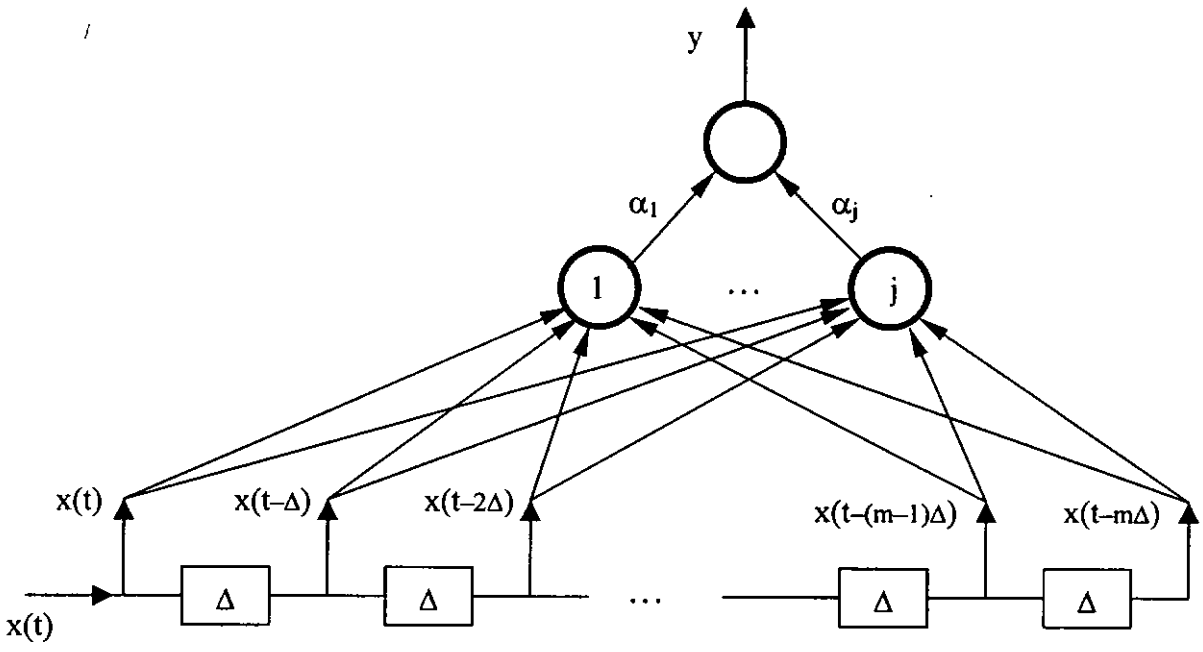


Now, given an input vector x(t), the function performed by the operator $\Gamma[\ ]$ is described by a vector-matrix multiply, followed by application of the activation function to each component of the resulting activation vector.

Equation $\mathbf{y}(t) = \Gamma[\mathbf{W}\mathbf{x}(t)]$ describes the general process of computing an output from an input performed by layer of network units. However, because dealing with recurrence, it considers the operation of the network in discrete time series. At every time step, t, the input received by a unit is the combination of the original input vector, x(t), and the output produced by layer at an earlier time, as seen through the feedback connections to the unit $\mathbf{x}(t-1)\mathbf{W}$. For the purpose of simplifying the analysis, considering the delay to be equivalent to single time step, which is also the amount of time required by each unit to compute an output from the current input. Such a network is depicted in *Chapter 3.4*.

3.3 NETWORK ARCHITECTURE

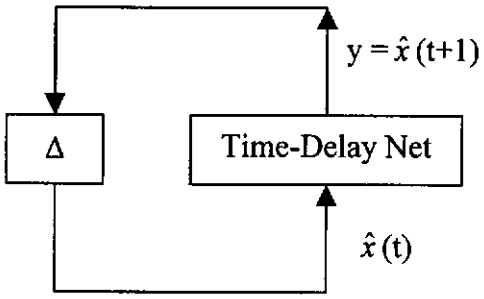
Recurrent networks appears to performs better than more commonly used tapped delay line feed-forward network in prediction of financial time series [El90][We90]. Consider a tapped time-delay neural network architecture shown in the figure below, This network maps a finite time sequence $\{x(t), x(t-\Delta), x(t-2\Delta), \ldots, x(t-m\Delta)\}$ into a single output y (this architecture also can be generalized for the case when x and/or y are vectors). One may view this neural network as a discrete-time nonlinear filter.



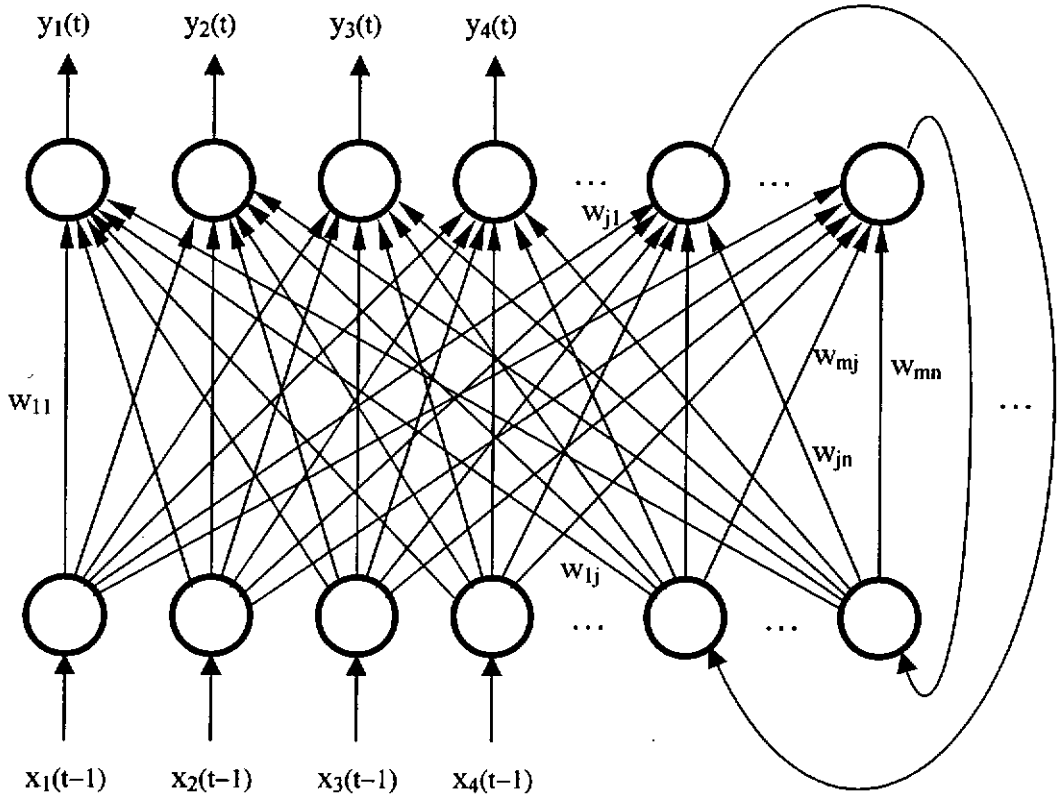
The architecture receives the (m+1)-dimensional "spatial" pattern x generated by a tapped-delay line preprocessor from a temporal sequence. Thus, the target values for the output unit is specified for various times t.

The time-delay neural net has been applied successfully to the problem of speech recognition and *time series* prediction. Here, time series prediction is discussed because it captures the spirit of the type of processing done by time-delay neural net. Given observed values of the state x of a (nonlinear) dynamical system at discrete time less than t, the goal is to use these values to accurately predict x(t+p), where p is some prediction time step into the future. Then $x(t+p) = g[x(t), x(t-\Delta), x(t-2\Delta), \ldots, x(t-m\Delta)]$.

A simple modification to the time-delay net makes it suitable for sequence reproduction. The training procedure is identical to the one for the preceding prediction network. However, during retrieval, the output y [predicting x(t+1)] is propagated through a single delay element, with the output of this delay element connected to the input of the time-delay net as is shown in above figure. The sequence reproduction net will only work if the prediction $y = \hat{x}(t+1)$ is very accurate, since any error in the predicted signal has a multiplicative effect due to iterated scheme employed. Finally, a simple RNN will replaced to time-delay net as shown below.



Fully Connected RNN Architecture



The $x_1(t-1)$, $x_2(t-1)$, $x_3(t-1)$, and $x_4(t-1)$ are represented as four currencies in this dissertation. And the $y_1(t)$, $y_2(t)$, $y_3(t)$, and $y_4(t)$ are the prediction values of four currencies.

3.4 LEARNING ALGORITHM

Learning methods for ANN can be classified as one of three basic types: supervised, reinforcement, or unsupervised. In this dissertation, the supervised learning of recurrent neural network is used.

3.4.1 Supervised Learning

In supervised learning a teacher is assumed to be present during the learning process and each example pattern used to train the network includes an input pattern together with a target or desired output pattern, the correct answer. During the learning process, a comparison can be made between the computed output by the network and the correct output to determine the error. The error can then be used to change network parameters, which result in an improvement in performance. The weight matrices connecting the layers are usually initialized by setting all weights to zero or to small random real-valued numbers.

The input training pattern vectors x^p , $p = 1, 2, \ldots$ are then presented to the network one at a time and a corresponding output pattern y^p is computed. This computed output pattern is compared to the desired or target output pattern t^p and an error $e^p = y^p - t^p$ is determined. The resultant error is then used through some form of computation and feedback to adjust the individual weights to reduce the error for each training pair. After iteratively adjusting weights for all training patterns, the weight values may converge to a set of values needed to perform the required pattern recalls. Learning has been achieved when the errors for all training patterns ($p = 1, 2, \ldots p$) have been reduced to some acceptable level for all new patterns not in the training set.

3.4.2 Widrow-Hoff LMS Learning

The Widrow-Hoff or Least Mean Square algorithm [Wi87][Wi90] uses a linear rule for training a perceptron. Let $d(k)$ be the desired real-valued output for the augmented pattern $\mathbf{y}(k)$, and suppose that the corresponding weight vector is $\mathbf{w}(k)$. The error at the kth iteration is $e_k = d(k) - \mathbf{w}(k)\bullet\mathbf{y}(k)$. The LMS algorithm is also a gradient descent algorithm requiring the squared error function, $e^2_k = [d(k) - \mathbf{w}(k)\bullet\mathbf{y}(k)]^2 = [d(k) - \mathbf{w}^T(k)\mathbf{y}(k)]^2$, to be minimized at the kth iteration, and $E = \sum e^2_k$. The difference between the LMS algorithm and perceptron algorithm is that the error minimized by the LMS algorithm is a continuous quantity. Both can be applied to the same neural network architecture, whose formulation is described in *Chapter 5.3.1*. The true gradient $\nabla_k = \dfrac{\partial E(e_k^2)}{\partial w(k)}$, by applying the gradient descent rule it gets

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\alpha e_k \mathbf{y}(k) \qquad (3.4.2.1)$$

where α is a suitable learning parameter that influences the stability and convergence rate. Because the gradient estimate is unbiased, the LMS algorithm could be made a true gradient descent algorithm in the limiting sense, by estimating ∇_k at each step but not updating the weights until many estimates have been accumulated.

The optimal solution $\mathbf{w}^*$ that minimizes the mean squared error is obtained by setting $\nabla_k = \dfrac{\partial E(e_k^2)}{\partial w(k)} = 0$. Provided $\mathbf{Q}$ is not singular, this solution is

$$\mathbf{w}^* = \mathbf{Q}^{-1}\mathbf{P} . \qquad (3.4.2.2)$$

where $\mathbf{Q} = E(\mathbf{y}(k)\mathbf{y}^T(k))$ is the N×N auto-correlation matrix of all the N training pattern
$\mathbf{P} = E(d(k)\mathbf{y}(k))$ is a vector of cross-correlation.

The optimal solution of $\mathbf{w}^* = \mathbf{Q}^{-1}\mathbf{P}$ requires a priori knowledge of the signal statistics conveyed through the matrices $\mathbf{Q}$ and $\mathbf{P}$. Usually, that information is not completely available. The LMS algorithm enables one to obtain an accurate estimate of the true solution by computing the pseudo-inverse matrix.

3.4.3 Pseudo-Inverse Solution

A form of correlative weight adjustment of the correlative matrices wound be correlative learning. The learning in this case is accomplished in a straightforward manner. The input patterns x^p and corresponding desired output patterns t^p are used to compute the weight matrix **W** as the sum of p superimposed pattern matrices $\mathbf{W}^p$ ($p = 1, 2, \ldots p$), where each $\mathbf{W}^p$ is computed as the outer product or *correlation matrix*, $\mathbf{W}^p = \mathbf{x}^p(\mathbf{y}^p)^T$ (the superscripted T denotes vector transpose).

Unfortunately, the correlative matrix of **A,B,C,D,E** in *equation (5.2.1.6)* may not have an inverse to find the correlative matrices. Therefore, pseudo-inverse matrix method is introduced as followings.

Pseudo-Inverse Matrix Optimization

Consider the possibility of providing the input / output description of the associative memory through the linear transformation

$$\mathbf{y} = \mathbf{W}\mathbf{x},$$

where **y** and **x** are respectively, representative vectors from the output and input spaces, whereas **W** is a ($p \times r$) weight matrix that characterizes approximately the associative memory. One approach to the solution for **W** is obtained by expressing in the form

$$\mathbf{W}[\, x_1\, x_2 \ldots x_n\,] = [\, y_1\, y_2 \ldots y_n\,] \qquad (3.4.3.1)$$

and then solving the foregoing system of linear equations by the method of generalized inverse. To solve for **W** in *equation (3.4.3.1)*, a generalized inverse **G** of

$$\mathbf{X} = [\, x_1\, x_2 \ldots x_n\,] \qquad (3.4.3.2)$$

G is usually *non-unique* and satisfies the matrix equation

$$\mathbf{XGX} = \mathbf{X} \qquad (3.4.3.3)$$

The matrix **G** in *equation (3.4.3.3)* is sometimes referred to as a **pseudo-inverse** to distinguish it from cases in which more constraints are imposed on a generalized inverse than the one conveyed through *equation (3.4.3.3)*. In fact, a generalized inverse can be made unique by the imposition of certain additional constraints. This unique generalized inverse, called the Moore-Penrose inverse, gives the minimum-norm least squares solution to any system of linear equations. That is, among all solutions to *equation (3.4.3.1)* for **W** that minimize the squared error,

$$E(\mathbf{W}) = \sum_{k=1}^{n} [y_k - Wx_k]^H [y_k - Wx_k]$$

The Moore-Penrose inverse $\mathbf{X}^+$ for **X** gives a weight matrix $\mathbf{W}_0 = \mathbf{YX}^+$ of minimum norm $\|\mathbf{W}\|^2$ where for $\mathbf{W} = [w_{ij}]$,

$$\|\mathbf{W}\|^2 = \sum_{i=1}^{p} \sum_{j=1}^{r} |w_{ij}|^2 .$$

When the matrix **X** in *equation (3.4.3.2)* is of full rank, $\mathbf{XX}^H$ is nonsingular, and the solution for **W** in *equation (3.4.3.1)*, obtained after computing a pseudo-inverse or in this case the Moore-Penrose inverse,

$$\mathbf{X}^+ = \mathbf{X}^H(\mathbf{XX}^H)^{-1}$$

of **X**, is

$$\mathbf{W} = \mathbf{YX}^+ = \mathbf{YX}^H(\mathbf{XX}^H)^{-1} \qquad (3.4.3.4)$$

where $\mathbf{X} = [\, x_1\ x_2 \ldots x_n \,]$
$\mathbf{Y} = [\, y_1\ y_2 \ldots y_n \,]$

Note that most applications of the entries of matrices and vectors are real-valued, the superscript H may be replaced by the notation T for transpose, and *equation (3.4.3.4)* is then changed to below.

$$\mathbf{W} = \mathbf{YX}^T(\mathbf{XX}^T)^{-1} \qquad (3.4.3.5)$$

Chapter Four

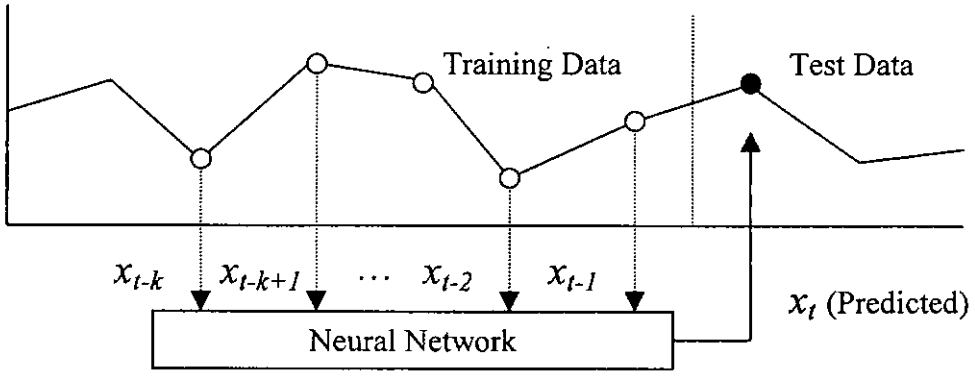
IMPLEMENTATION

- Moving Window Simulation
- Data Normalization
- Software Implementation

4.1 MOVING WINDOW SIMULATION

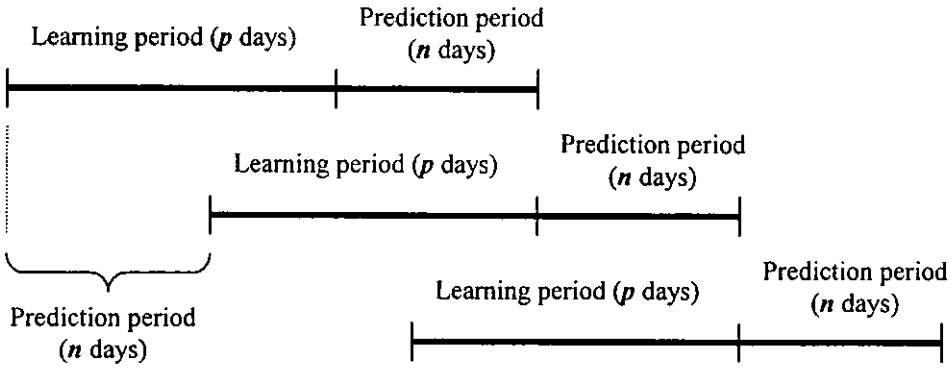
4.1.1 FX Time Series Windowing

The problem of forecasting time series can be stated as follows: Given a number of values (observations) of a time dependent variable x_t (t = 1, 2, ...), we wish to accurately predict the value of the variable at some future time $t + h$. For simplicity, we assume the time series is **stationary**, that is, the underlying probability distribution or dynamics driving the system remains constant over time. To be able to forecast the series using a RNN network, the appropriate training scheme have been devised in *Chapter 3.4*. The most common method is to use the RNN as a time delay neural network. We assume some historical time series data is available to train the network. Using this data, a sequence of $k \geq 2$ consecutive discrete time values of the series are used for input to the network. Usually the series values are equally spaced in time (hourly, daily, weekly, monthly). For input, the k values x_{t-k} x_{t-k+1} ... x_{t-1} are used with a single target or desired value of x_t (or x_{t+h}). The input time delay "**window**" is then shifted one or more time points later in the series and second training vector is formed using the values x_{t-k+1} x_{t-k+2} ... x_t with target value x_{t+1} . The window and target values are shifted later in time again for the next point and so on. This training set is used over some chosen time interval repeatedly to adjust the weights by LSE and adaptive matrix method in this dissertation. The training and testing samples are illustrated in following figure.



4.1.2 Moving Simulation

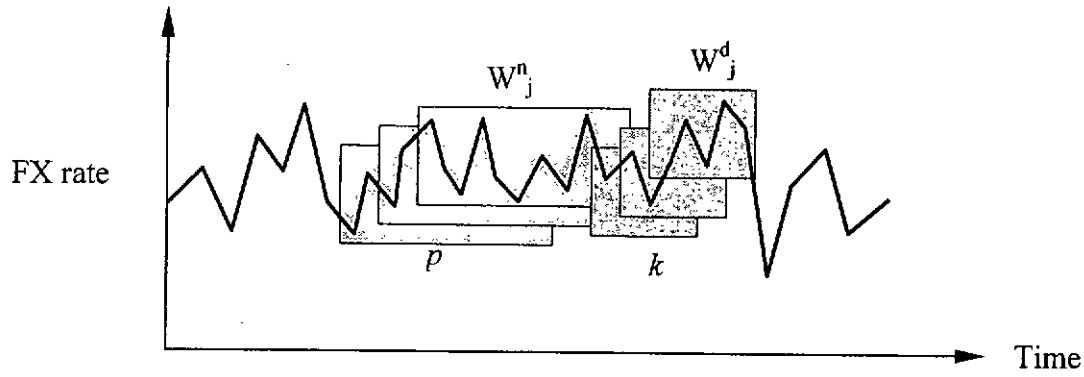
For prediction of an economic system, such as exchange rate, in which the prediction rules are changing continuously, learning and prediction must follow the changes. In this study, a prediction method called moving simulation is used. In this system, prediction is done by simulation while moving the objective learning and prediction periods. This moving simulation predicts as shown in the figure below. The system learns data for the past ***p*** days (training patterns) and then predicts for the next ***n*** days. The system advances while repeating this.



4.1.3 Windowing Architecture

The network architecture at the input and output levels is largely determined by the application. The method of identifying regularities within a data set (training patterns) contaminated by noise is windowing. The basic idea is to use two windows W^n and W^d of fixed sizes, p and k respectively, to look into the data set. For a given window size the assumption is that the sequence of values W^n_0 , ... W^n_p is somehow related to the following sequence W^d_0 , ... W^d_k , and that this relationship, although unknown, is defined entirely within the data set. Various methods can then be used to correlate the two sets of values. In this case of neural networks, $W^n \rightarrow W^d$ can be used as a training vector. In this investigation,

W^n is training patterns and W^d is (Time delay) variables of matrix. Both windows are shifted along the time series using a fixed size *s* as shown in below figure.



The choice of window and step sizes is critical to the ability of any prediction system to identify high-order regularities in time series and thus approximate the hidden relationship accurately. Quite often some preprocessing of data set is required to obtain a sensible starting point for W^n, W^d, *p*, *k*, and *s*. Those parameters are determined in *Chapter 5 & 6*.

The rationale behind this is that a year's trading (p=260) would capture the underlying regularities quite accurately and would eliminate much of the noise. The rationale behind the weekly size of $k = 5$ is to considered *weekly effect*. Moreover, the overall effect of $s = 2$ is to take the moving average over a 2-day period. This would dampen the effect of trading near the beginning and the end of the week where the processing of backlogs might affect the markets and introduce further noise. The resulting training set consists of overlapping snapshots of the time series each of length 260 daily (yearly), moving along the curve at an interval of 2 days, where time delay is $t_0, \ldots, t_j$. More formally:

$$
\begin{aligned}
t_0 &: \; W^n_0 = \{ V_0, \ldots, V_{259} \} \quad \rightarrow W^d_0 = \{ V_{260} \} \\
t_1 &: \; W^n_1 = \{ V_2, \ldots, V_{261} \} \quad \rightarrow W^d_1 = \{ V_{262} \} \\
&\cdots\cdots \\
t_j &: \; W^n_j = \{ V_{j+s}, \ldots, V_{j+p} \} \quad \rightarrow W^d_j = \{ V_{j+p+1} \}
\end{aligned}
\qquad (4.1.3)
$$

The intermediate size of training set and test set makes the problem nontrivial and also allows for extensive tests of learning speed and generalization of performance.

4.1.4 Single / Double Step Prediction

There are two types of step methods (single-step and multi-step) in FX rate forecasting. The single-step for next day prediction and double-step for the day after prediction are used in this dissertation.

4.1.4.1 *Nearest neighbor Windowing*

For the single-step prediction, the network predicts the FX rate value one step ahead of time, here is next day price. The window moves one step only (daily). Single-step prediction serves two purposes. *Firstly*, it is a good mechanism for evaluating the adaptability and robustness of the prediction system by showing that even when its prediction is wrong, it is not dramatically wrong and that the system can use the actual value to correct itself. *Secondly*, it can act as an indictor generator that would allow traders to buy or sell in advance of a price increase or decrease respectively.

Single-step method is used to predict next day price in the present investigation.

4.1.4.2 *Second neighbor Windowing*

For the double-step prediction, there is a requirement for long-term forecasting that aims to identify general trends and major turning points in a FX rate. In multi-step (or double-step) prediction, the system uses a set of current values to predict the value of the exchange rate for a fixed period. The prediction is then fed back to the network to forecast next period. The Taylor expansion method (*Chapter 5.2.2*) of RNN has the smooth interpolation properties, which can be used to produce the second neighbor price of FX rate. Details are studied in *Chapter 6.1*.

Double-step method is used to predict the price on the day after in this study.

4.2 DATA NORMALISATION

4.2.1 Indicators Selection

Geographical factor is a main distribution of FX trading in the world. FX market was almost entirely the result of the above average growth of trading in the United Kingdom, which took its share of global trading up from 25% in 1989 to nearly 30% in 1992. The combined share of the three major countries (the United Kingdom, the United States and Japan) is about 60%. The four most important centers, Singapore, Switzerland, Hong Kong and Germany, accounted together for around a further 25% of the total FX volume. The Deutsche Mark is now the second most widely traded currency in the FX market. And the Swiss Franc also as a shelter in the political risk.

From the ***Datastream*** in HKPU Library, I collected four major currencies: the Deutsche Mark, Sterling, the Swiss Franc and Japanese Yen quoted against United States Dollar (using the notation of the International Organization for Standardization, respectively USD/DEM, USD/GBP, USD/CHF and USD/JPY) exchange rates for my study. The basic input indicators behavior will be shown in *Appendix A*. The data set of FX rates covers the period from 30th June 1993 to 30th June 1997 resulting in 1044 records. Over this period, there are some missing data (i.e. London bank holidays), which are replaced by the yesterday FX rate.

For five trading day of a week, the one-month training pattern of RNN is 22, and one-year training pattern is 260. The number of observation periods (forecasting time) is either 1000 when one considers the one-month (training pattern) maturity or 750 when on considers the one-year maturity. For the one-month time horizon, the total number of forecasting time and training pattern should be less than the total number of data set (1036 record), 8 records for preprocessing and calculation.

4.2.2 Raw Data Preprocessing

Series of raw FX prices sampled at regular time intervals (*here is daily*) and filtered by some means can form one set of possible inputs. However, application of this type of inputs in the FX market has many problems. First, FX price movements are generally non-stationary and quite random in nature, and therefore not very suitable for learning purpose. Second, raw price inputs give rise to scaling problems; however, this issue is less serious and can be addressed with little manipulation.

4.2.2.1 *Raw Data*

Raw data are seldom adequate for neural network training. Therefore, the raw data set is used in *chapter 2* only to select a best statistical model for comparison with RNN model. Data analysis and transformations are necessary to enhance information that provides a better descriptor of the trends or processes present in the raw data.

4.2.2.2 *Relative Variable Difference*

If $x_1, x_2, \ldots, x_t$ is a price sequence sampled at regular intervals, then $x_2 - x_1$, $x_3 - x_2$, $\ldots$, $x_t - x_{t-1}$ are defined as the first differences of the price sequence. First differences are perhaps the most effective way to generate data sets for neural network learning. This is true both for intra-day and inter-day data series. Two advantages of taking first differences can be cited here. First, it helps in making the time series stationary, which is a useful statistical property. Second, it overcomes the scaling problem, which is associated with rates. Therefore, the input vectors of *Relative variable Difference* (RVD) strategy is

$$X_{(t)} = \frac{[x(t) - x(t-1)]}{SD} \qquad (4.2.2.2)$$

where SD is standard deviation of the whole data set
$X_{(t)}$ is RNN value
$x(t)$ is original value

4.2.3 Scaling Learning Vectors

The FX raw data were used in three different ways. First, the original FX raw data without any preprocessing were used in traditional statistic method. Obviously, this may cause problems because the ratios have very different means and standard deviations. The other two methods are formulated as followings.

4.2.3.1 *Nonlinear Transformation*

Therefore, a simple standardization procedure was used as one kind of data preprocessing. It is always used in conjunction with a nonlinear transformation in order to undermine the impact from outlying data:

$$\text{Symmetric Sigmoid function } f(\sigma) = \frac{1-e^{-\sigma}}{1+e^{-\sigma}} \qquad (4.2.3.1.1)$$

$$\text{Logistic Activation function } g(\sigma) = \frac{1}{1+e^{-\sigma}} \qquad (4.2.3.1.2)$$

In the later stages of my study, two activation functions of *equation* (*4.2.3.1.1*) & (*4.2.3.1.2*) are used. The best results are obtained using the MAE analysis method and standardization in conjunction with the nonlinear transformation of the input data. Details of tables in *Chapter 5*, show the results obtained here. The best results of squashing raw data is function $f(\sigma)$. It is because the function $f(\sigma)$ initializes each raw data an interval signal between –1 and 1, which fits to direction matching characteristics.

4.2.3.2 *Unique Interval*

It is necessary to transform all input and output data for the RNN into a range from 0 to 1. A recommended method is the use of a simple mapping function as below. The function requires the maxima and minima for all data.

$$x_1 = \frac{x - \min_x}{\max_x - \min_x} \qquad \text{where } x \text{ is source value and } x_1 \text{ is RNN value.}$$

It is also important to give due consideration to the output transfer function. If it is one with asymptotic limits such as the sigmoid, which only reaches the limits [0,1] for inputs at infinity, then typical outputs may only reach values in the range [–1,1] in order to present the direction of FX rate. A common way of scaling the data is by a linear transformation given *equation (4.2.3.2)*.

$$SCALE = \frac{MAX - MIN}{\max_x - \min_x}$$

$$OFFSET = MAX - \frac{MAX - MIN}{\max_x - \min_x}\max_x$$

$$x_1 = x \times SCALE + OFFSET$$

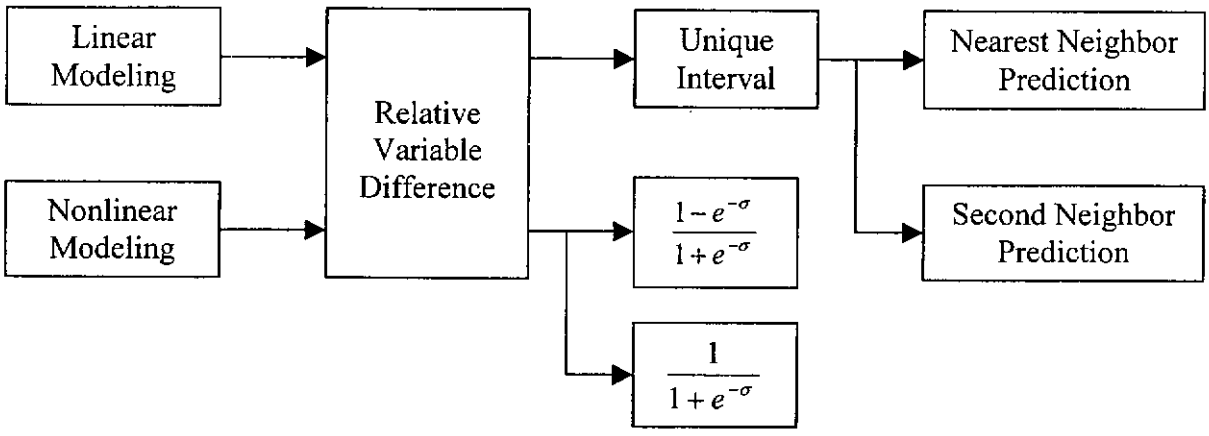
$$= (MAX - MIN)\frac{x - \max_x}{\max_x - \min_x} + MAX \qquad (4.2.3.2)$$

where x_1 is the scaled variable;
x is the original variable;
$\min_x$,$\max_x$ are the minimum and maximum values of original variable of x;
MIN, *MAX* are the values of the target range, that is [–1,1] or even [–0.9,0.9].

4.2.4 Transformation Selection

According to the results in *chapter 5*, the linear modeling has a better result using unique interval transformation. But the nonlinear transformation is suitable for nonlinear modeling. The final result in *Chapter 6*, the unique interval transformation is used.

Block Diagram of Data Transformation Testing



4.3 SOFTWARE IMPLEMENTATION

4.3.1 Data Set Structure

The following preprocessing example is a part of data set, which are prepared for SES and RNN modeling. There some steps as shown below:

4.3.1.1 *Raw Data Set*

Table 4.3.1.1

Currency	USD/DEM	USD/GBP	USD/CHF	USD/JPY
30/6/93	0.5857	1.4915	0.6601	0.009337
1/7/93	0.5901	1.5145	0.6625	0.009313
2/7/93	0.5996	1.508	0.662	0.009217
⋮	⋮	⋮	⋮	⋮
30/6/97	0.5729	1.6645	0.6842	0.008722

The raw data is retrieved from HKPU *Datastream*. The data structure included the date and one FX rate only. Using EXCEL® software transfer Four FX rates to a text file named "yspm.dat".

4.3.1.2 *Taking Difference*

Table 4.3.1.2

Currency	USD/DEM	USD/GBP	USD/CHF	USD/JPY
30/6/93	-----	-----	-----	-----
1/7/93	+0.0044	+0.0230	+0.0024	–0.000024
2/7/93	+0.0095	–0.0065	–0.0005	–0.000096
⋮	⋮	⋮	⋮	⋮
30/6/97	–0.0022	0.0	–0.0040	–0.000008

Take RVD function to scaling the data set with *equation* (*4.2.2.2*). The results will be obtained +ve and –ve values, means *upward* trend and *downward* trend of the FX rate.

4.3.1.3 *Rearrange Data Set*

Table 4.3.1.3

Currency	30/6/97	29/6/97	28/6/97	...	1/7/93
USD/DEM	–0.0022	–0.0043	–0.0009	...	+0.0044
USD/GBP	0.0	–0.0005	+0.0017	...	+0.0230
USD/CHF	–0.0040	–0.0079	+0.0007	...	+0.0024
USD/JPY	–0.000008	–0.000112	+0.000055	...	–0.000024

The data set is rearranged in order to write the **MATLAB®** programs. The indexing of data set are simple to refer all the formula with reverse the sign of the time index. For example $x(t-2)$ becomes $x(t+2)$.

4.3.1.4 *Squashing Data*

The whole data set will squashing by *equation* (*4.2.3.1.1*), (*4.2.3.1.2*) or (*4.2.3.2*). For comparison with statistical SES model. All input variables are used to predict the rating (which is scaled to fall in the interval [–1,+1]. The first two data are prepared for testing. The first one is *Day After* FX price, and the second one is *Next Day* FX rate.

4.3.2 Program Lists

User Guide:

All the programs are written by **MATLAB®** software, and the source codes are provided by a given disk. The user should start the **MATLAB®** text files from the windows Environment. Before run the m-files, change the parameters, which will be tested. By testing the programs, all the results are recorded to tables in this dissertation.

File Name	Description	Parameters / Results	Details
● *Traditional statistical modeling*			
Regress.m	Linear Regression Modeling	Single Forecasting SES Model	Table 2.5.1
Updown4.m	Alpha Selection	Alpha = 0.5 & 0.3	Table 6.1.1.2
Predict4.m	Buy, Hold or Sell Decision	53.4% of success matching	Table 9.4.4
● *LSE parameters selection*			
Mul_test.m	Matrix Multiplication Method	$A\Sigma PP^T + B\Sigma PQ^T + C\Sigma PR^T + D\Sigma PS^T + E\Sigma PT^T = \Sigma PY^T$	Table 5.5.4.1
LSE_dela.m	Time Delay Selection	3 days time delay	Table 5.5.4.2
LSE_simu.m	3 Matrices Selection	A,B,C simultaneously	Table 5.5.4.3
LSE_lin.m	Linear / Nonlinear Selection	Nonlinear (Short term = 22)	Table 5.5.4.4
LSE_h.m	h Parameter Selection	Nonlinear model (h = 0.7)	Table 5.5.4.5
● *APIM parameters selection*			
AM_seq.m	5 Matrix Sequence Selection	Finding sequence A→B→C→D→E	Table 5.5.5.1
AM_delay.m	Time Delay Selection	5 days (weekly)	Table 5.5.5.2
AM_term.m	Short / Long Term Selection	Long term = 260 (Linear)	Table 9.4.3
AM_line.m	Linear / Nonlinear Selection	Linear model & Unique interval	Table 5.5.5.4
● *Modeling Selection*			
AM_LSE.m	APIM / LSE method	APIM Optimization better than LSE	Table 5.5.6
Updown_u.m	Unique interval (Up / Down)	Taylor Expansion method	Table 6.1.2
Updown_n.m	Nonlinear Transformation (U/D)	Taylor Expansion method	Table 6.1.2
● *Source code in Appendix C*			
Predict.m	Buy / Hold / Sell	Mixed modeling (Nonlinear & Linear)	Table 9.4.5 Table 9.4.6

4.3.3 Pseudo-Code of PREDICT.m

Read the currency daily raw data

Take the *Relative Variable Difference* of raw data

Unique interval of RVD raw data from 1 to –1

Event case is not end of **forecasting Testing**

{ Event case is a **New Training Set** window

{ Get the *Nearest Neighbor* X(t+1) and *Second Neighbor* X(t+2)

Rearrange the *Training Pattern* as X(t), X(t–1), … X(t–k), … vectors

Take the *Inverse Hyperbolic Tangent* function of X(t+1) vector

Initialize the 5 *Vector Matrices* of training pattern is 390

Compute correlation matrix **A** by using APIM optimization

Calculate the $Error_a$ of $\hat{X}$ (t+1) by using matrix **A**

Compute correlation matrix **B** with $Error_a$ using APIM method

Calculate the $Error_{ab}$ of $\hat{X}$ (t+1) by using matrix **A,B**

Compute correlation matrix **C** with $Error_{ab}$ using APIM method

Calculate the $Error_{abc}$ of $\hat{X}$ (t+1) by using matrix **A,B,C**

Compute correlation matrix **D** with $Error_{abc}$ using APIM method

Calculate the $Error_{abcd}$ of $\hat{X}$ (t+1) by using matrix **A,B,C,D**

Compute correlation matrix **E** with $Error_{abcd}$ using APIM method

}

Compute the *Predicted Nearest Neighbor* rate $\hat{X}$ (t+1)

Compute the predicted $\hat{X}$ (t+2) with $\hat{X}$ (t+1) using *Taylor Expansion* method

Take the difference of X(t+1) – $\hat{X}$ (t+1) and X(t+2) – $\hat{X}$ (t+2)

Sum up the MAE and MAPE

Sum up the success of *Direction Prediction* using Fuzzy Decision

}

Printout the *Total* MAE and MAPE of 4 currencies

Printout the *Total* success of Direction Matching of 4 currencies

Chapter Five

MODELS SELECTION

- Introduction
- Formulation
- LSE Optimization
- APIM Optimization
- Models Selection Strategy

5.1 INTRODUCTION

5.1.1 Forecasting Criterions

While the MAE & MAPE are a perfectly acceptable measure of performance, in practice the ultimate goal of any testing strategy is to reassure the modeler that his or her results are robust and to measure the profitability of the system. It is therefore important to design a testing strategy can be determined the future trending of the FX rates. In fact, the MAE could not measure the FX rates direction movement, it is only for models comparison.

To determine whether a market is trending, RVD will be calculated for each discrete time series. Specifically, the high and low values of market at the current time are compared with high and low values at the previous time. The difference values obtained indicate the plus directional movement (+DM) and minus directional movement (–DM), respectively. While the system to determine the suitable parameters, the **Direction Matching** should be used for major criterion. If the predicted value matches to the actual value (e.g. both are +DM or –DM), the prediction will be counted for success. This Direction matching analysis is a simple method for testing trending.

5.1.2 Methodology

The FX movement can be obtained by RVD of raw data. *Nearest neighbor* forecast and *Second neighbor* forecast are two major factors for evaluating the FX trending in order to make recommendation of buy-hold-sell strategy. Two predictions are combined with fuzzy rules to make a trading decision, details in *Chapter 6.3.2*.

Two *numerical optimization* methods are LSE and APIM, which are used to find the coefficients weights of correlation matrix. Their formulation details are derived in following chapters. The testing is determining the best numerical optimization method with the best parameters setting.

5.2 FORMULATION

5.2.1 Next Day Forecasting (Nearest Neighbor)

The recurrent neural networks are capable of modeling non-linear dynamic systems. RNNs have powerful capabilities for modeling many computational structures. The real-time recurrent learning algorithm is based on a training method by Williams and Zipser [Wi89]. Then the pre-processed trajectory is approximated by the following equation [Li96]:

$$y_i(t+1) = (1-h)\, y_i(t) + h[\sigma(\sum_{j=1}^{n} W_{ij} y_j(t) + \theta_i)] \qquad (5.2.1.1)$$

where h is the parameters of the system;
θ is the thresholds of the system.

After learning of RNN, the model of forecasting is

$$y_i(t+1) = (1-h)\, x_i(t) + h[\sigma(\sum_{j=1}^{n} W_{ij} x_j(t) + \theta_i)] \qquad (5.2.1.2)$$

where $W \in \mathbf{R}^{4\times4}$, and $\theta \in \mathbf{R}^4$.

The optimal of weights W in RNN is found by Least Square Error (LSE) method.

$$\text{LSE} = \sum_{j=1}^{n} [W_{ij} x_j(t) - x_j(t-1)]^2 \qquad (5.2.1.3)$$

where $x(t)$ is the state of the system.

The fully-connected RNN training could be computed directly by a dynamical system as following non-linear differential equations. The discrete time forecasting model can be formulated by $y(t+1) = y(t) + h\frac{dy(t)}{dt}$ (5.2.1.4)

If $h = 1$, J = 0 and σ is the identity mapping, then $y(t+1) = \mathrm{W}y(t) + \theta$ is the multi-regression model. This model will be compared with the following non-linear RNN model.

$$x(t+1) = x(t) + h\,\{ -x(t) + \sigma\,[\, \mathbf{S}x_i(t) + \theta\,] + \mathrm{J}\,\}, \qquad (5.2.1.5)$$

where $\mathbf{S}x_i(t)$ is the time delay of the system, then the equation rewrote as

$$x(t+1) = \underbrace{(1-h)\,x(t)}_{\text{Linear Part}} + \underbrace{h\,\{\sigma\,[\,\mathbf{A}x(t) + \mathbf{B}x(t-1) + \mathbf{C}x(t-2) + \mathbf{D}x(t-3) + \mathbf{E}x(t-4) + \theta\,] + J\}}_{\text{Non-linear Part}} \qquad (5.2.1.6)$$

According to above RNN equation the correlation matrix **A**, **B**, **C**, **D** and **E** are found by numerical optimization method.

5.2.2 Day After Forecasting (Second Neighbor)

The Day after forecasting FX rate is used Taylor expansion as followings:

Using the basic formula : $y(t+1) = y(t) + h\dfrac{dy(t)}{dt}$

Taylor Series of $y(t+1)$ is $\quad y_{t+1} = y_t + \dfrac{h^1}{1}y_t' + \dfrac{h^2}{2!}y_t'' + \dfrac{h^3}{3!}y_t''' + O(h^4)$

Taylor Series of $y(t-1)$ is $\quad -) \quad y_{t-1} = y_t - \dfrac{h^1}{1}y_t' + \dfrac{h^2}{2!}y_t'' - \dfrac{h^3}{3!}y_t''' + O(h^4)$

And the difference of $\quad y_{t+1} - y_{t-1} = 2hy_t' + \dfrac{2h^3}{3!}y_t'''$

$$y_{t+1} = y_{t-1} + 2hy_t' + O(h^3)$$

ignoring the $O(h^3)$ terms, $\quad y_{t+2} = y_t + 2hy_{t+1}'$

$$y(t+2) = y(t) + 2h\frac{dy(t+1)}{dt} \qquad (5.2.2.1)$$

Finally,

$$x(t+2) = x(t) + 2h\,\{\,-x(t+1) + \sigma\,[\,\mathbf{S}x_{i+1}(t+1) + \theta\,] + J\,\},$$

$$x(t+2) = x(t) + 2h\{-x(t+1) + \sigma\,[\,\mathbf{A}x(t+1) + \mathbf{B}x(t) + \mathbf{C}x(t-1) + \mathbf{D}x(t-2) + \mathbf{E}x(t-3) + \theta\,] + J\,\} \qquad (5.2.2.2)$$

The vectors of $x(t+1)$ and $\mathbf{A}x(t+1)$ are found by *equation* (5.2.1.6) which are the predicted values of the system.

5.3 LEAST SQUARE ERROR OPTIMIZATION

5.3.1 Least Square Error (LSE)

The LSE is a common method of numerical optimization. The following **A**, **B**, **C**, **D** & **E** are 4×4 correlation matrix. Using windowing technique in *equation (4.1.3)*, the matrix of Y, P, Q, R, S & T are obtained by $x(t)$, $x(t-1)$, … , & $x(t-5)$ vectors respectively. Where $x(t)$ is a $m\times 1$ vector with m currency rates, Y is a $m\times p$ matrix with p training patterns. All the matrices of Y, P, Q, R, S & T are non-square matrices. Grouping vectors of $x(t)$, $x(t-1)$ … $x(p)$ to form the matrix Y, where $Y = [x(t)\ \ x(t-1)\ \ \ldots\ \ x(t-p+1)]$.

Five-day time delay (*weekly effects*) is used in the following examples.

$$x(t) = \mathbf{A}\,x(t-1) + \mathbf{B}\,x(t-2) + \mathbf{C}\,x(t-3) + \mathbf{D}\,x(t-4) + \mathbf{E}\,x(t-5) \qquad (5.3.1.1)$$

$$Y = \mathbf{A}\,P + \mathbf{B}\,Q + \mathbf{C}\,R + \mathbf{D}\,S + \mathbf{E}\,T \qquad (5.3.1.2)$$

The squared norm of matrix is $Z(A,B,C,D,E) = \sum_{i=1}^{n} \|Y - P - Q - R - S - T\|^2$

$$Z = \sum_{i=1}^{n} \|Y - AP - BQ - CR - DS - ET\|^2 \qquad (5.3.1.3)$$

Z is the sum of the squared errors for training pattern over all output units and the output Y. In order to minimize the total error Z with respect to the weights, it is required to have the partial derivatives of *equation (4.3.1.3)* set to zero.

$$\begin{cases} \dfrac{\partial Z}{\partial A} = 2\sum_{i=1}^{n}(Y - AP - BQ - CR - DS - ET)(-P) = 0 \\ \dfrac{\partial Z}{\partial B} = 2\sum_{i=1}^{n}(Y - AP - BQ - CR - DS - ET)(-Q) = 0 \\ \dfrac{\partial Z}{\partial C} = 2\sum_{i=1}^{n}(Y - AP - BQ - CR - DS - ET)(-R) = 0 \\ \dfrac{\partial Z}{\partial D} = 2\sum_{i=1}^{n}(Y - AP - BQ - CR - DS - ET)(-S) = 0 \\ \dfrac{\partial Z}{\partial E} = 2\sum_{i=1}^{n}(Y - AP - BQ - CR - DS - ET)(-T) = 0 \end{cases} \qquad (5.3.1.4)$$

To find the correlation matrix **A**, **B**, **C**, **D** & **E** by using pseudo-inverse matrix. Because of not all matrices have an inverse. But every matrix has a pseudo-inverse, a modified limit form of original matrix. The pseudo-inverse provides an alternative approach to the solution of a system of simultaneous *equation* (*5.3.1.5*). It arises in the context of minimizing a sum of squares which will be minimized is typically the errors in the neural network output.

Calculating the partial derivatives from *equations* (*5.3.1.4*), we get

$$\begin{cases} A\Sigma PP^T + B\Sigma PQ^T + C\Sigma PR^T + D\Sigma PS^T + E\Sigma PT^T = \Sigma PY^T \\ A\Sigma QP^T + B\Sigma QQ^T + C\Sigma QR^T + D\Sigma QS^T + E\Sigma QT^T = \Sigma QY^T \\ A\Sigma RP^T + B\Sigma RQ^T + C\Sigma RR^T + D\Sigma RS^T + E\Sigma RT^T = \Sigma RY^T \\ A\Sigma SP^T + B\Sigma SQ^T + C\Sigma SR^T + D\Sigma SS^T + E\Sigma ST^T = \Sigma SY^T \\ A\Sigma TP^T + B\Sigma TQ^T + C\Sigma TR^T + D\Sigma TS^T + E\Sigma TT^T = \Sigma TY^T \end{cases} \quad (5.3.1.5)$$

There is an alternative method of matrix multiplication as shown below:

$$\begin{cases} A\Sigma PP^T + B\Sigma QP^T + C\Sigma RP^T + D\Sigma SP^T + E\Sigma TP^T = \Sigma YP^T \\ A\Sigma PQ^T + B\Sigma QQ^T + C\Sigma RQ^T + D\Sigma SQ^T + E\Sigma TQ^T = \Sigma YQ^T \\ A\Sigma PR^T + B\Sigma QR^T + C\Sigma RR^T + D\Sigma SR^T + E\Sigma TR^T = \Sigma YR^T \\ A\Sigma PS^T + B\Sigma QS^T + C\Sigma RS^T + D\Sigma SS^T + E\Sigma TS^T = \Sigma YS^T \\ A\Sigma PT^T + B\Sigma QT^T + C\Sigma RT^T + D\Sigma ST^T + E\Sigma TT^T = \Sigma YT^T \end{cases}$$

There are four different matrix multiplication methods, which will be selected by the program MUL_TEST.m and get the results in *Table 5.5.4.1*. Finally, the solution of the correlation matrices **A**, **B**, **C**, **D** & **E** could be formulated in matrix notation *equation* (*5.3.1.6*). For 5 Days Time Delay, matrix notation *equation* (*5.3.1.5*) may be written as:

$$\begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} = \begin{bmatrix} \sum PP^T & \sum PQ^T & \sum PR^T & \sum PS^T & \sum PT^T \\ \sum QP^T & \sum QQ^T & \sum QR^T & \sum QS^T & \sum QT^T \\ \sum RP^T & \sum RQ^T & \sum RR^T & \sum RS^T & \sum RT^T \\ \sum SP^T & \sum SQ^T & \sum SR^T & \sum SS^T & \sum ST^T \\ \sum TP^T & \sum TQ^T & \sum TR^T & \sum TS^T & \sum TT^T \end{bmatrix}^{-1} \begin{bmatrix} \sum PY^T \\ \sum QY^T \\ \sum RY^T \\ \sum SY^T \\ \sum TY^T \end{bmatrix} \quad (5.3.1.6)$$

where all variables of P, Q, R, S & T are non-square matrices.

5.4 APIM OPTIMIZATION

There is still another approach, which can be applied to minimize the total error in retrieved patterns. In this case the correlation matrix is obtained by direct computation using the pseudo-inverse of the input pattern matrix. The input matrix is n×m matrix X consisting of m columns of pattern vectors X^m, where each vector is n-dimensional. The pseudo-inverse of a matrix is derived in *Chapter 3.4.3* and it is shown that the pseudo-inverse of X always exists, even when X is a non-square matrix. Thus, one can compute directly the pseudo-inverse of X to solve the linear system of equations as following:

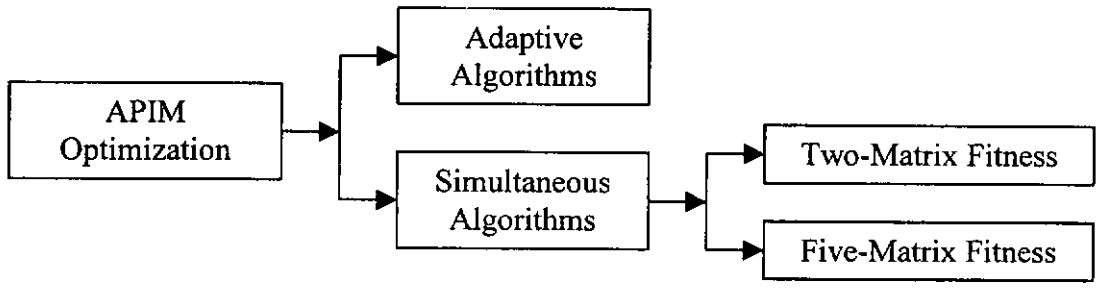
$$\mathbf{W} = \mathbf{Y}\mathbf{X}^{\mathbf{T}} (\mathbf{X}\mathbf{X}^{\mathbf{T}})^{-1}. \qquad (5.4.1.1)$$

Where **Y** is the matrix of target vector associated with the input matrix **X**

The solution of matrix W, gives a least square solution in the sense of retrieval errors the same as the *Delta Rule* when the input vectors form a linearly independent set. When more than one solution exists, the pseudo-inverse will be the one with the smallest sum of squares. When **X** has normal inverse, $\mathbf{X}^{-1}$, the pseudo-inverse will be identical to $\mathbf{X}^{-1}$. This method is not computationally appealing, however, since to compute each weight all other weights must be used. They are all interdependent.

There are two methods (LSE and APIM optimizations) show in next chapters to find the correlation matrices of RNN. Three approaches of APIM method is shown below. The solutions are obtained by pseudo-inverse matrix calculation.

Block diagram of Adaptive Pseudo-Inverse Matrix Optimization



5.4.1 Adaptive Algorithm

According to *equation* (*5.3.1.2*), to find **A**, **B**, **C**, **D** & **E** by using pseudo-inverse matrices. This adaptive method eliminates the errors step by step, and all the matrices formats as same as in *Chapter 5.3.1*. The first correlation matrix **A** is found by using *equation* (*5.4.1.1*), and assume correlation matrices **B**, **C**, **D** & **E** are zero. The equation becomes as follows:

$$\mathbf{Y} = \mathbf{A}\,\mathbf{P}$$

By using Pseudo-Inverse matrix method, the correlation matrix **A** is

$$\mathbf{A} = \mathbf{Y}\mathbf{P}^{T}(\mathbf{P}\mathbf{P}^{T})^{-1} \qquad (5.4.1.2)$$

After correlation matrix **A** has been found, the error compared with true value, which will be forecased. The new vector Y is created to be applied pseudo-inverse matrix optimization. The *equation* (*5.4.1.1*) is used to find another correlation matrix **B** again as followings.

$$x(t) - \mathbf{A}\,x(t-1) = \mathbf{B}\,x(t-2)$$

In matrix notation $\mathbf{Y}_a = \mathbf{B}\,\mathbf{Q}$ Where $\mathbf{Y}_a = x(t) - \mathbf{A}\,x(t-1)$, $\mathbf{Q} = x(t-2)$

Hence, $$\mathbf{B} = \mathbf{Y}_a\mathbf{Q}^{T}(\mathbf{Q}\mathbf{Q}^{T})^{-1}$$

Similarly correlation matrices **C**, **D** & **E** are found, as followings:

$$x(t) - \mathbf{A}\,x(t-1) - \mathbf{B}\,x(t-2) = \mathbf{C}\,x(t-3)$$

$$\mathbf{Y}_b = \mathbf{C}\,\mathbf{R}$$

$$\mathbf{C} = \mathbf{Y}_b\mathbf{R}^{T}(\mathbf{R}\mathbf{R}^{T})^{-1}$$

$$x(t) - \mathbf{A}\,x(t-1) - \mathbf{B}\,x(t-2) - \mathbf{C}\,x(t-3) = \mathbf{D}\,x(t-4)$$

$$\mathbf{Y}_c = \mathbf{D}\,\mathbf{S}$$

$$\mathbf{D} = \mathbf{Y}_c\mathbf{S}^{T}(\mathbf{S}\mathbf{S}^{T})^{-1}$$

$$x(t) - \mathbf{A}\,x(t-1) - \mathbf{B}\,x(t-2) - \mathbf{C}\,x(t-3) - \mathbf{D}\,x(t-4) = \mathbf{E}\,x(t-5)$$

$$\mathbf{Y}_d = \mathbf{E}\,\mathbf{T}$$

$$\mathbf{E} = \mathbf{Y}_d\mathbf{T}^{T}(\mathbf{T}\mathbf{T}^{T})^{-1}$$

5.4.2 Simultaneous Algorithms

5.4.2.1 *Two Matrix Fitness*

If the time delay is two, there are an alternative method to estimate the correlation matrix of **A** & **B**. The equation as shown below:

$$x(t) = \mathbf{A}\,x(t-1) + \mathbf{B}\,x(t-2)$$

In matrix notation, $\mathbf{Y} = \mathbf{A}\,\mathbf{P} + \mathbf{B}\,\mathbf{Q}$ *(5.4.2.1.1)*

By using different pseudo-inverse matrices substitute into the *equation (5.4.2.1.1)*. Two equations are obtained.

$$\begin{cases} \mathbf{Y}\mathbf{P}^{T} = \mathbf{A}\,\mathbf{P}\mathbf{P}^{T} + \mathbf{B}\,\mathbf{Q}\mathbf{P}^{T} \\ \mathbf{Y}\mathbf{Q}^{T} = \mathbf{A}\,\mathbf{P}\mathbf{Q}^{T} + \mathbf{B}\,\mathbf{Q}\mathbf{Q}^{T} \end{cases}$$

$$\begin{cases} \mathbf{Y}\mathbf{P}^{T}(\mathbf{Q}\mathbf{P}^{T})^{-1} = \mathbf{A}\,\mathbf{P}\mathbf{P}^{T}(\mathbf{Q}\mathbf{P}^{T})^{-1} + \mathbf{B} \\ \mathbf{Y}\mathbf{Q}^{T}(\mathbf{Q}\mathbf{Q}^{T})^{-1} = \mathbf{A}\,\mathbf{P}\mathbf{Q}^{T}(\mathbf{Q}\mathbf{Q}^{T})^{-1} + \mathbf{B} \end{cases} \quad (5.4.2.1.2)$$

Solving correlation matrices by simultaneous equations method, and gives the results as following:

$$\mathbf{A} = (\mathbf{Y}\mathbf{P}^{T}(\mathbf{Q}\mathbf{P}^{T})^{-1} - \mathbf{Y}\mathbf{Q}^{T}(\mathbf{Q}\mathbf{Q}^{T})^{-1})\,(\mathbf{P}\mathbf{P}^{T}(\mathbf{Q}\mathbf{P}^{T})^{-1} - \mathbf{P}\mathbf{Q}^{T}(\mathbf{Q}\mathbf{Q}^{T})^{-1})^{-1}$$

$$\mathbf{B} = \mathbf{Y}\mathbf{Q}^{T}(\mathbf{Q}\mathbf{Q}^{T})^{-1} - \mathbf{A}\,\mathbf{P}\mathbf{Q}^{T}(\mathbf{Q}\mathbf{Q}^{T})^{-1} \quad (5.4.2.1.3)$$

5.4.2.2 *Five Matrix Fitness*

This method could be solved for any numbers of time delay, normally the time delay is equal to five. According to *equation (5.3.1.2) & (5.4.1.1)* some different transpose of matrices could be added to every coefficient weight of correlation matrices **A**, **B**, **C**, **D** & **E**, and then five different equations are obtained as following.

$$\begin{cases} YP^T = APP^T + BQP^T + CRP^T + DSP^T + ETP^T \\ YQ^T = APQ^T + BQQ^T + CRQ^T + DSQ^T + ETQ^T \\ YR^T = APR^T + BQR^T + CRR^T + DSR^T + ETR^T \\ YS^T = APS^T + BQS^T + CRS^T + DSS^T + ETS^T \\ YT^T = APT^T + BQT^T + CRT^T + DST^T + ETT^T \end{cases}$$

By adding suitable pseudo-inverse matrices, the equations become as follows:

$$\begin{cases} YP^T(PP^T)^{-1} = \mathbf{A} + BQP^T(PP^T)^{-1} + CRP^T(PP^T)^{-1} + DSP^T(PP^T)^{-1} + ETP^T(PP^T)^{-1} \\ YQ^T(QQ^T)^{-1} = APQ^T(QQ^T)^{-1} + \mathbf{B} + CRQ^T(QQ^T)^{-1} + DSQ^T(QQ^T)^{-1} + ETQ^T(QQ^T)^{-1} \\ YR^T(RR^T)^{-1} = APR^T(RR^T)^{-1} + BQR^T(RR^T)^{-1} + \mathbf{C} + DSR^T(RR^T)^{-1} + ETR^T(RR^T)^{-1} \\ YS^T(SS^T)^{-1} = APS^T(SS^T)^{-1} + BQS^T(SS^T)^{-1} + CRS^T(SS^T)^{-1} + \mathbf{D} + ETS^T(SS^T)^{-1} \\ YT^T(TT^T)^{-1} = APT^T(TT^T)^{-1} + BQT^T(TT^T)^{-1} + CRT^T(TT^T)^{-1} + DST^T(TT^T)^{-1} + \mathbf{E} \end{cases}$$

Rewrite the equations in matrix form.

$$\begin{bmatrix} YP^T(PP^T)^{-1} \\ YQ^T(QQ^T)^{-1} \\ YR^T(RR^T)^{-1} \\ YS^T(SS^T)^{-1} \\ YT^T(TT^T)^{-1} \end{bmatrix} = \begin{bmatrix} 1 & QP^T(PP^T)^{-1} & RP^T(PP^T)^{-1} & SP^T(PP^T)^{-1} & TP^T(PP^T)^{-1} \\ PQ^T(QQ^T)^{-1} & 1 & RQ^T(QQ^T)^{-1} & SQ^T(QQ^T)^{-1} & TQ^T(QQ^T)^{-1} \\ PR^T(RR^T)^{-1} & QR^T(RR^T)^{-1} & 1 & SR^T(RR^T)^{-1} & TR^T(RR^T)^{-1} \\ PS^T(SS^T)^{-1} & QS^T(SS^T)^{-1} & RS^T(SS^T)^{-1} & 1 & TS^T(SS^T)^{-1} \\ PT^T(TT^T)^{-1} & QT^T(TT^T)^{-1} & RT^T(TT^T)^{-1} & ST^T(TT^T)^{-1} & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix}$$

The correlation matrices of **A**, **B**, **C**, **D** & **E** are obtained as following:

$$\begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} = \begin{bmatrix} YP^T(PP^T)^{-1} \\ YQ^T(QQ^T)^{-1} \\ YR^T(RR^T)^{-1} \\ YS^T(SS^T)^{-1} \\ YT^T(TT^T)^{-1} \end{bmatrix} \begin{bmatrix} 1 & QP^T(PP^T)^{-1} & RP^T(PP^T)^{-1} & SP^T(PP^T)^{-1} & TP^T(PP^T)^{-1} \\ PQ^T(QQ^T)^{-1} & 1 & RQ^T(QQ^T)^{-1} & SQ^T(QQ^T)^{-1} & TQ^T(QQ^T)^{-1} \\ PR^T(RR^T)^{-1} & QR^T(RR^T)^{-1} & 1 & SR^T(RR^T)^{-1} & TR^T(RR^T)^{-1} \\ PS^T(SS^T)^{-1} & QS^T(SS^T)^{-1} & RS^T(SS^T)^{-1} & 1 & TS^T(SS^T)^{-1} \\ PT^T(TT^T)^{-1} & QT^T(TT^T)^{-1} & RT^T(TT^T)^{-1} & ST^T(TT^T)^{-1} & 1 \end{bmatrix}^{-1} \quad (5.4.2.2)$$

5.5 MODELS SELECTION STRATEGY

Several questions need to be answered before attempting to forecast with RNN. How many training samples are required? How many input data points should be used, that is, what window size is best? What prediction horizon should be chosen (how far into future to predict), how should the test and training data be divided, what network configuration should be used (type of activation functions) and so on. Therefore this dissertation make use of some strategies to solve these problems as followings.

5.5.1 Selection Strategy

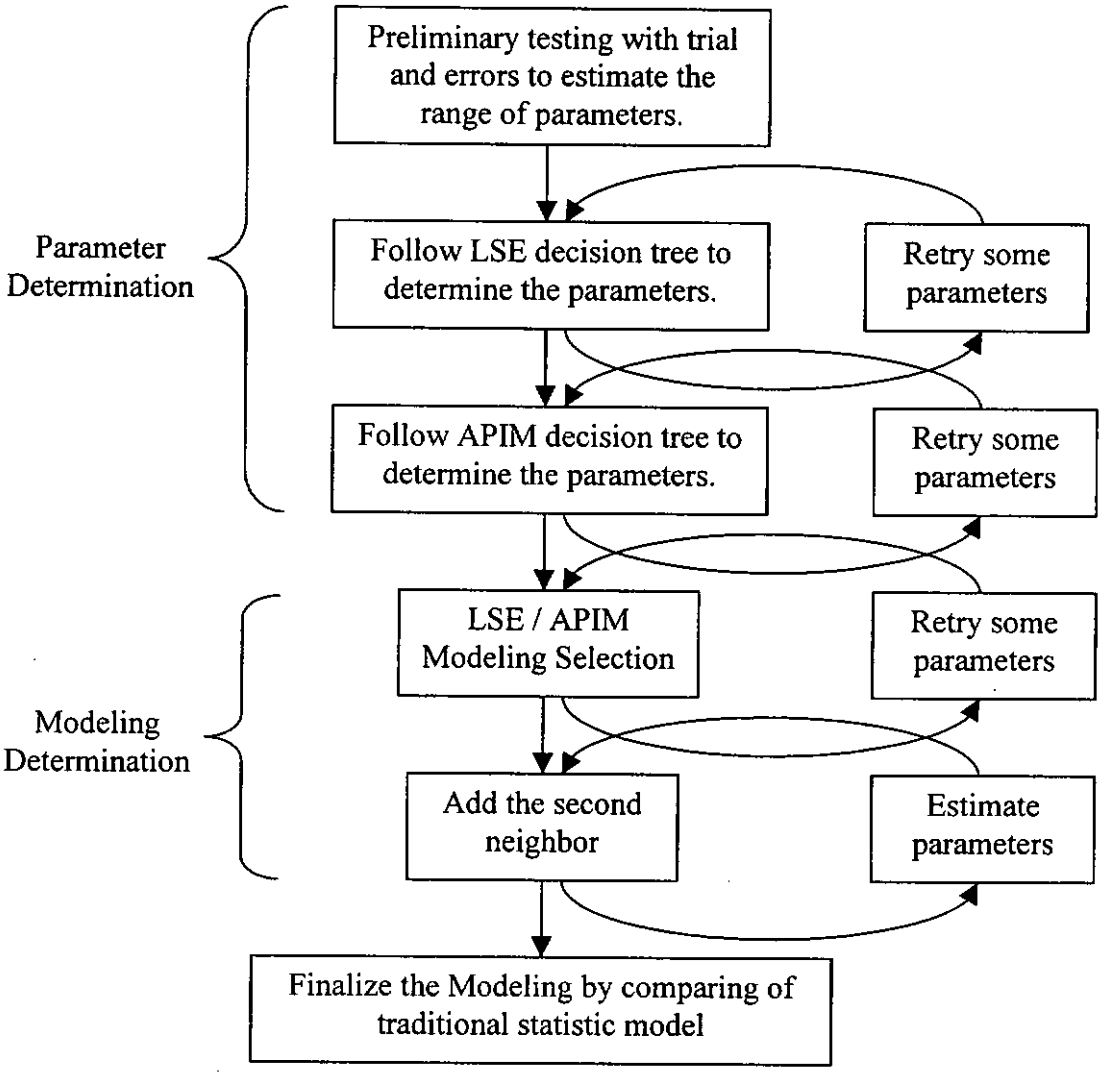
There are too many parameters for selection. So a preliminary test should be used, and a cycling testing method is introduced in the next section. Two decision trees are set up for testing. At the bottom of the trees, it shows the results of the testing, the testing processes are described in *Chapter 5.5.4* and *5.5.5*. Both of LSE and APIM decision trees are the **Numerical Optimization** method for finding the correlation matrices.

Two criterions will be used for selection the parameters in the processing. They are MAE & Direction Matching (Up/Down). The direction analysis is the major selection criterion of the system. This is a threshold error monitor for targets that take on fixed discrete values. This method is monitored by percentage accuracy. An alternative method uses MAE as monitor, accumulating the error only for those outputs that are on the wrong side of the threshold, measuring the error as the difference between the observed output and the threshold value, rather than the target.

The total results of *Direction matching* of different parameters or modeling are calculated into percentage. MAPE can determine the accuracy of the system.

5.5.2 Cycling Testing Strategy

Because of the parameters where are correlated one another, it is very difficult to find a starting point for testing. Therefore, a **Cycling Testing** method is used for modeling selection in this dissertation. It divides into two main parts, the first one is *Parameters Determination*, and the second part is *Modeling Determination*. The normalization methods and modeling parameters will be retest again in decision tree processes during decision making. The cycling processes are described as following diagram.



In *Chapter 5.5.4 and 5.5.5*, the decision trees illustrate the processes to determine the parameters by top-down scheme. The cycling testing method is used by input different parameters as well.

5.5.3 Parameters Assumption / Range

There are two main groups of parameters in the RNN to be chosen. The first area is in the preprocessing stage. The second area is modeling parameters which affects the results of RNN.

5.5.3.1 *Data Normalization*

Relative Variable Difference (RVD) = $\frac{X(t)-X(t-1)}{SD}$

Nonlinear Transformation is $\frac{1-e^{-\alpha}}{1+e^{-\alpha}}$ or $\frac{1}{1+e^{-\alpha}}$

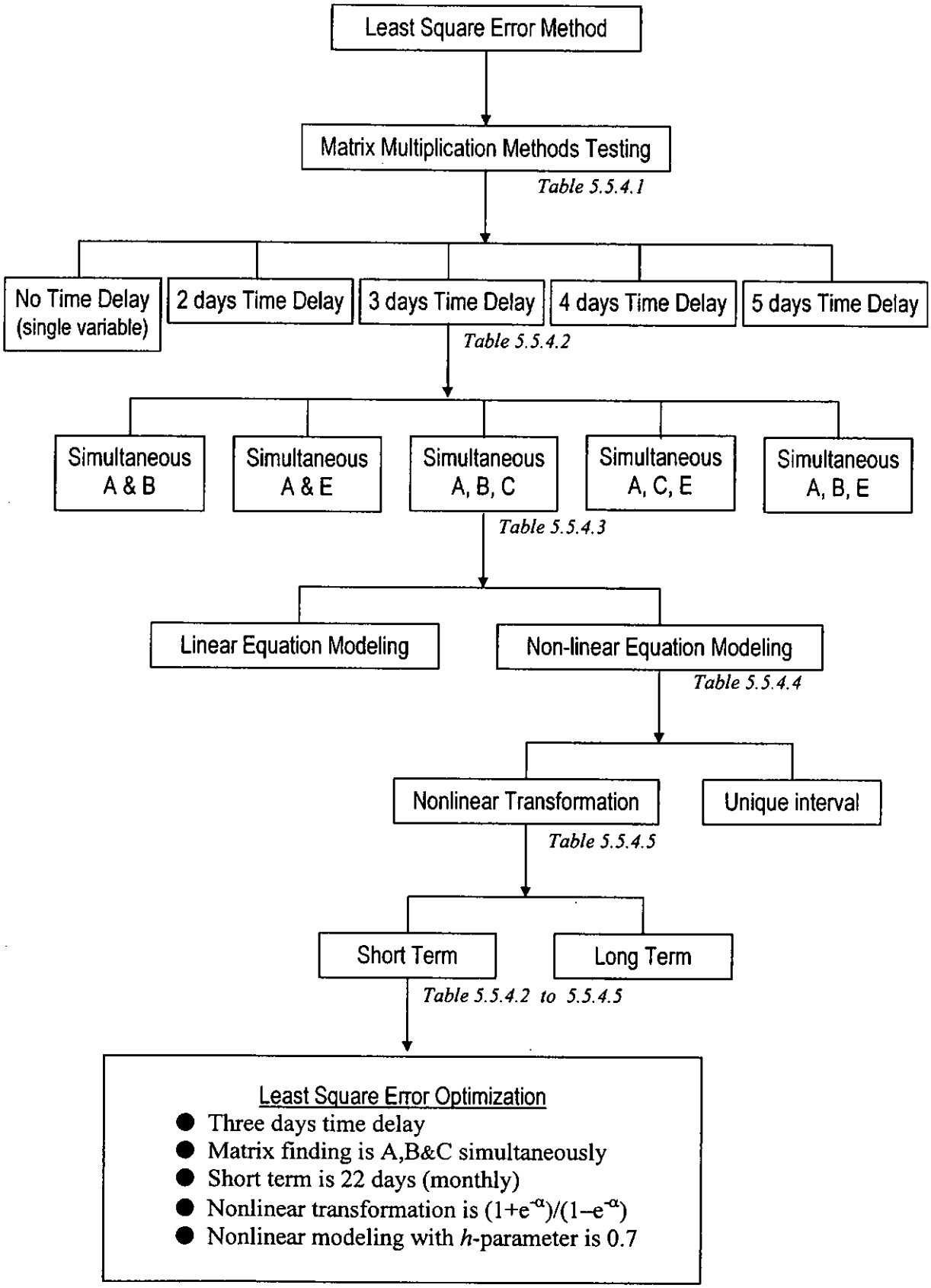
Unique interval [–1,+1] is $(MAX-MIN)\frac{x-\max_x}{\max_x-\min_x}+MAX$

5.5.3.2 *Modeling Parameters*

- The training pattern is the length of the training set which determined into Short / Long Term. Mainly the short term is a week and long term is a year. The values of term should be fitted to FX market system. The FX markets open from Monday to Friday only. Therefore the characteristics of values of terms are weekly = 5, monthly = 22, season = 65 and yearly = 260.
- Time delay is the variable selection from 1 to 5 in the programs. Also the time delay sequence of the correlation matrix **A**, **B**, **C**, **D** & **E** will be determined.
- *h*-parameter range is 0.5, 0.6, 0.7, 0.75, 0.8, 0.9, if h = 1, the system will become a Multiple Regression modeling.
- Times of forecasting are 50×4 or 100×4 in testing the system. The total records of results are 200 or 400 respectively, including four currencies.
- σ = tanh, $\theta = 0$, $J = 0$.

5.5.4 Decision Tree of LSE Method

The Top-down decision illustrates the testing procedure, and the results are tabulated accordingly.



5.5.4.1 *Matrix Multiplication Method*

Program name:	MAT_TEST.m
Parameters:	Time delay = 5 days
	Training pattern = 260 (yearly)
	Forecasting testing = 100×4 times
Preprocessing:	RVD is $[X(t)-X(t-1)]/SD$
	Unique interval of LSE is form –1 to +1
Formula:	shown in *Chapter 5.3*

MAE analysis	Time Delay					Total
	1	2	3	4	5	
$\Sigma YP^T = A\Sigma PP^T + B\Sigma QP^T + C\Sigma RP^T + D\Sigma SP^T + E\Sigma TP^T$	0.6048	0.6623	0.6473	0.6574	0.6903	3.2621
$\Sigma PY^T = A\Sigma PP^T + B\Sigma QP^T + C\Sigma RP^T + D\Sigma SP^T + E\Sigma TP^T$	0.5642	0.5461	0.5630	0.6480	0.7837	3.1050
$\Sigma PY^T = A\Sigma PP^T + B\Sigma PQ^T + C\Sigma PR^T + D\Sigma PS^T + E\Sigma PT^T$	0.6048	0.6680	0.6765	0.6913	0.7018	3.3424
$\Sigma YP^T = A\Sigma PP^T + B\Sigma PQ^T + C\Sigma PR^T + D\Sigma PS^T + E\Sigma PT^T$	0.5642	0.5415	0.5703	0.6463	0.7608	3.0831

Table 5.5.4.1

Direction matching (Up / Down)	Time Delay					Total
	1	2	3	4	5	(%)
$\Sigma YP^T = A\Sigma PP^T + B\Sigma QP^T + C\Sigma RP^T + D\Sigma SP^T + E\Sigma TP^T$	265	265	255	254	251	64.5
$\Sigma PY^T = A\Sigma PP^T + B\Sigma QP^T + C\Sigma RP^T + D\Sigma SP^T + E\Sigma TP^T$	250	255	251	240	234	61.5
$\Sigma PY^T = A\Sigma PP^T + B\Sigma PQ^T + C\Sigma PR^T + D\Sigma PS^T + E\Sigma PT^T$	265	266	255	253	257	***64.8***
$\Sigma YP^T = A\Sigma PP^T + B\Sigma PQ^T + C\Sigma PR^T + D\Sigma PS^T + E\Sigma PT^T$	250	254	250	248	233	61.75

From the LSE *decision tree* in *Chapter 5.5.4*, The first step is determining the matrix multiplication method in calculation. After preliminary testing, the initial values of parameters are used in this process. According to the above table, They are divided into two main groups (1&3, 2&4). The 1&3 groups have more MAE and success matching, on the other hands, the 2&4 groups have less MAE and matching. The trading profit depends on direction trend (*Upward/Downward*) forecasting. And the MAE are very close to each others, they have maximum 8.4% difference only. Therefore, the best direction matching the third multiplication method is selected, even the MAE is the biggest.

The following multiplication method is applied to the whole dissertation:

$$\begin{cases} A\Sigma PP^T + B\Sigma PQ^T + C\Sigma PR^T + D\Sigma PS^T + E\Sigma PT^T = \Sigma PY^T \\ A\Sigma QP^T + B\Sigma QQ^T + C\Sigma QR^T + D\Sigma QS^T + E\Sigma QT^T = \Sigma QY^T \\ A\Sigma RP^T + B\Sigma RQ^T + C\Sigma RR^T + D\Sigma RS^T + E\Sigma RT^T = \Sigma RY^T \\ A\Sigma SP^T + B\Sigma SQ^T + C\Sigma SR^T + D\Sigma SS^T + E\Sigma ST^T = \Sigma SY^T \\ A\Sigma TP^T + B\Sigma TQ^T + C\Sigma TR^T + D\Sigma TS^T + E\Sigma TT^T = \Sigma TY^T \end{cases}$$

5.5.4.2 *Time Delay Selection*

Program name:	LSE_DELA.m
Parameters:	Forecasting testing = 100×4 times
Preprocessing:	RVD is $[x(t)-x(t-1)]/SD$
	Squashing data of LSE is $(1-e^{-\alpha})/(1+e^{-\alpha})$
Formula:	*Equation* (*5.3.1.6*) for time delay sequence

MAE analysis	Training Patterns *(Length of Training Data Set)*					Total
	22	65	130	195	260	
No Time Delay	1.7122	1.3639	1.2883	1.2951	1.2606	6.9201
2 Days Time Delay	2.5013	1.5178	1.3515	1.3132	1.2728	7.9566
3 Days Time Delay	3.1774	1.6951	1.4945	1.3818	1.3110	9.0598
4 Days Time Delay	4.8273	1.9099	1.6543	1.5496	1.4583	11.3994
5 Days Time Delay	9.5049	2.2090	1.7767	1.6580	1.5582	16.7068

Table 5.5.4.2

Direction matching (Up / Down)	Training Patterns *(Length of Training Data Set)*					Total
	22	65	130	195	260	(%)
No Time Delay	200	205	198	205	206	50.7
2 Days Time Delay	205	199	198	205	212	50.95
3 Days Time Delay	212	208	204	210	207	***52.05***
4 Days Time Delay	206	200	203	198	194	50.05
5 Days Time Delay	203	199	204	200	192	49.9

According to the above results, the MAE are stable in long term of training pattern and almost get the same MAE results. In long term, the MAE are in the range approximate ±10% difference. And also, the majority selection is *Direction matching*. There are some oscillation occuring in direction matching results. The 3 days time delay is more stable and gets a more success matching results, but it has the maximum matching in short term.

The time delay is considered from 1 to 5 only. If time delay more than 5 days, the weekly effect will be overlapped by windowing technique, and the training time increasing exponentially. And also over-training will happen if the time delay is too long. From the *table 5.5.4.2* above, the 3 Days Time Delay is selected and the training pattern tends to short term windowing. Finally, the result fits to above arguments.

5.5.4.3 *Three Variables Sequence*

Program name: LSE_SIMU.m
Parameters: Forecasting testing = 100×4 times
Preprocessing: RVD is $[x(t)-x(t-1)]/SD$
Squashing data of LSE is $(1-e^{-\alpha})/(1+e^{-\alpha})$
Formula: *Equation (5.3.1.6)*

MAE analysis		Training Patterns *(Length of Training Data Set)*					Total
		22	65	130	195	260	
Matrix	A & B	1.8112	1.3563	1.2851	1.3026	1.2672	7.0223
	A & E	2.3554	1.5505	1.3947	1.3447	1.3060	7.9513
	A, B, C	3.1764	1.6945	1.4940	1.3813	1.3105	9.0567
	A, C, E	3.0647	1.7573	1.5157	1.3969	1.3365	9.0711
	A, B, E	3.7175	1.6854	1.4396	1.3704	1.3241	9.5370

Table 5.5.4.3

Direction matching (Up / Down)		Training Patterns *(Length of Training Data Set)*					Total (%)
		22	65	130	195	260	
Matrix	A & B	199	207	204	208	207	51.25
	A & E	208	200	208	203	205	51.2
	A, B, C	212	208	204	210	207	***52.05***
	A, C, E	206	217	205	205	202	51.75
	A, B, E	206	207	212	205	204	51.7

If "**A**" is Monday, then referring to correlation matrix **A**, **B**, **C**, **D** & **E** which are Monday, Tuesday, Wednesday, Thursday & Friday respectively. Although the 3 time delay was obtained in *Chapter 5.5.4.2*, the 2 time delay was tested again also, because of the use of cycling testing strategy. There is no actual sequence in forecasting. Results are obtained simultaneously with extract some meaningful time delay only.

In my work three categories were tested. They were nearest time delay (e.g. **A**,**B**,**C** and **A**,**B**), weekend effect (e.g. **A**,**C**,**E** and **A**,**E**), and combined both characteristics (e.g. **A**,**B**,**E**). Finally, according to direction analysis the 3 time delay with nearest forecasting day got more success matching percentage. Therefore, the **A**,**B**,**C** time delay was selected in LSE optimization.

5.5.4.4 *Linear / Nonlinear Modeling Selection*

Program name: LSE_LIN.m
Parameters: Time delay = 3 days
Forecasting testing = 100×4 times
Preprocessing: RVD is $[x(t)-x(t-1)]/SD$
Squashing data of LSE is $(1-e^{-\alpha})/(1+e^{-\alpha})$ or $1/(1+e^{-\alpha})$
Formula: *Equation (5.3.1.6)*

MAE analysis		Training Patterns *(Length of Training Data Set)*					Total
		22	65	130	260	390	
Linear	$(1-e^{-\alpha})/(1+e^{-\alpha})$	3.1764	1.8945	1.4940	1.3105	1.2947	8.9701
	$1/(1+e^{-\alpha})$	4.6189	1.6558	1.6623	1.7047	1.6277	11.2694
$h = .7$	$(1-e^{-\alpha})/(1+e^{-\alpha})$	3.3030	3.0598	2.9422	2.8196	2.4447	14.5693
	$1/(1+e^{-\alpha})$	2.7050	2.7146	2.7582	2.7931	3.0470	14.0179

Table 5.5.4.4

Direction matching (Up / Down)		Training Patterns *(Length of Training Data Set)*					Total (%)
		22	65	130	260	390	
Linear	$(1-e^{-\alpha})/(1+e^{-\alpha})$	212	208	204	207	191	51.1
	$1/(1+e^{-\alpha})$	212	205	204	207	211	51.95
$h = .7$	$(1-e^{-\alpha})/(1+e^{-\alpha})$	211	210	209	215	206	***52.55***
	$1/(1+e^{-\alpha})$	214	204	205	207	207	51.85

The best result of LSE method is shaded. It is a nonlinear model with a short term training windowing. From the above table, The length of training pattern is directly proportional to the MAE results, but it is inversely proportional to Direction matching results. This means that the shorter training pattern of time series will give a better forecasting result. But the training windowing is too short, the system becomes a linear model. By observation, the training windowing more than one year period may be increase percentage of success.

The h-parameter determines the system either a linear system or a nonlinear system. In the case, $h = 0$, the nonlinear part of *equation (5.2.1.6)* will become equal to zero. It remains the linear part. The system becomes a linear model. If $h = 1$, the linear part will be deleted and the nonlinear part becomes a multiple regression model. Therefore, the h-parameter testing should be found more accurately in next chapter.

5.5.4.5 *h-parameter Determination*

Program name: LSE_H.m
Parameters: Forecasting testing = 100×4 times
Preprocessing: RVD is $[x(t)-x(t-1)]/SD$
Squashing data of LSE is $(1-e^{-\alpha})/(1+e^{-\alpha})$ or $1/(1+e^{-\alpha})$
Unique interval of APIM is from −1 to +1
Formula: *Equation (5.3.1.6)*

MAE Analysis		Adaptive Matrix Modeling				LSE Modeling				Total
		$(1-e^{-\alpha})/(1+e^{-\alpha})$		$1/(1+e^{-\alpha})$		$(1-e^{-\alpha})/(1+e^{-\alpha})$		$1/(1+e^{-\alpha})$		
		22	260	22	260	22	260	22	260	
Training Patterns	$h=.5$	3.1764	1.3105	4.6189	1.7047	2.2740	2.5161	2.1478	1.9649	19.7133
	$h=.6$	3.1764	1.3105	4.6189	1.7047	2.5288	2.5340	2.4200	2.4019	20.6952
	$h=.7$	3.1764	1.3105	4.6189	1.7047	3.3030	2.8196	2.7050	2.7931	22.4312
	$h=.8$	3.1764	1.3105	4.6189	1.7047	3.0473	2.0080	3.1329	3.4235	22.4222
	$h=.9$	3.1764	1.3105	4.6189	1.7047	2.9763	2.0863	3.3581	4.3122	23.5434
	$h=1$	3.1764	1.3105	4.6189	1.7047	2.4523	1.3957	3.7365	3.1218	21.5168

Table 5.5.4.5

Direction matching (Up /Down)		Adaptive Matrix Modeling				LSE Modeling				Total
		$(1-e^{-\alpha})/(1+e^{-\alpha})$		$1/(1+e^{-\alpha})$		$(1-e^{-\alpha})/(1+e^{-\alpha})$		$1/(1+e^{-\alpha})$		
		22	260	22	260	22	260	22	260	(%)
Training Patterns	$h=0.5$	212	207	212	207	197	205	202	193	51.09
	$h=0.6$	212	207	212	207	212	208	188	203	51.53
	$h=0.7$	212	207	212	207	211	215	214	207	***52.66***
	$h=0.8$	212	207	212	207	209	206	204	201	51.81
	$h=0.9$	212	207	212	207	211	208	208	211	52.38
	$h=1.0$	212	207	212	207	217	210	204	202	52.22

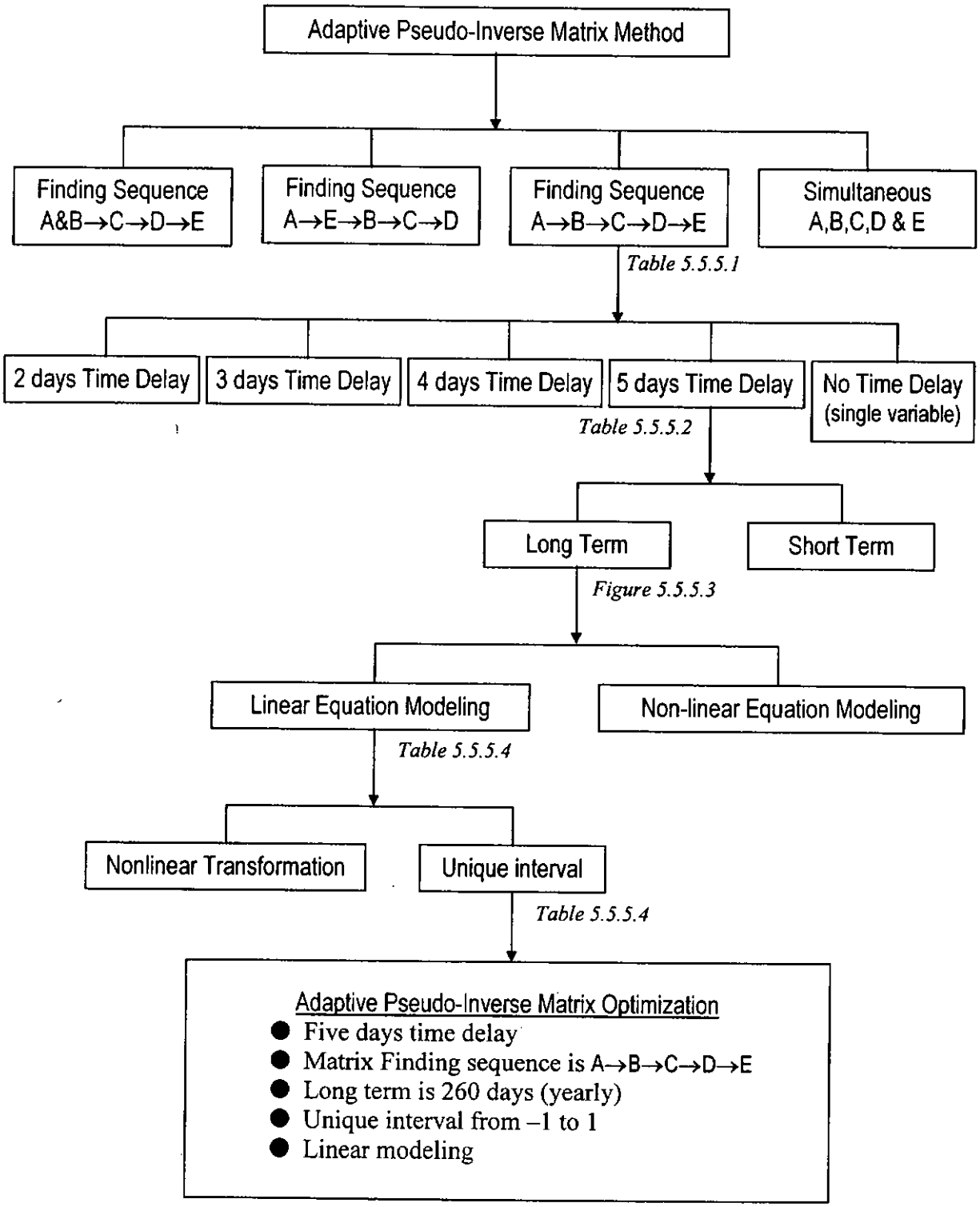
$h=0.7$ is selected, but the smaller h will be used again in my work for eliminate MAE.

5.5.4.6 *Summary*

LSE method is a very common method in numerical optimization. This algorithm is based on the simply minimizes the instantaneous error squared, thereby reducing the storage requirement to the minimum possible. Normally the MAE errors are small in this chapter using LSE method, but the prediction of price direction has only achieved 53% successful rate. It is not a good result. Therefore, another numerical optimization method (APIM) should be tested again with the same time series.

5.5.5 Decision Tree for Adaptive Matrix Method

The initial parameters are based on LSE results in *Chapter 5.5.4*. But some parameters will be changed and tested again in order to get better results. Each stage of the tree is tested by a MATLAB® program, and the results are tabulated in this chapter.



5.5.5.1 *Variables Sequence Analysis of Adaptive Matrix*

Program name:	AM_SEQ.m
Parameters:	Forecasting testing = 50×4 times or 100×4 times
Preprocessing:	RVD is [x(*t*)–x(*t–1*)]/SD
	Unique interval of APIM is from –1 to +1
Formula:	*Equation* (*5.4.2.1.1*) for 2 variables A&B
	Equation (*5.4.2.2*) for 5 variables A,B,C,D&E

MAE analysis

Term	Short (*monthly*) = 22				Long (*yearly*) = 260				Total
Times of Forecast	50×4		100×4		50×4		100×4		
h-parameter	0.7	1	0.7	1	0.7	1	0.7	1	
Finding Sequence									
A→B→C→D→E	0.6892		0.6701		0.4760		0.4799		2.3152
A→E→B→C→D	0.6904		0.6586		0.4722		0.4791		2.3003
A & B→C→D→E	13.0443		10.9034		56.1538		76.7375		156.8399
A, B, C, D & E	26.0056		20.9093		0.5884		0.5766		48.0799

Table 5.5.5.1

Direction matching (Up / Down)

Term	Short (*monthly*) = 22				Long (*yearly*) = 260				Total
Times of Forecast	50×4		100×4		50×4		100×4		
h-parameter	0.7	1	0.7	1	0.7	1	0.7	1	(%)
Finding Sequence									
A→B→C→D→E	119	119	242	242	142	142	273	273	***64.67***
A→E→B→C→D	118	118	242	242	138	138	270	270	64.00
A & B→C→D→E	111	111	204	204	104	104	228	228	53.92
A, B, C, D & E	98	98	193	193	132	132	258	258	56.75

APIM requires a *Finding Sequence* to find correlation matrix **A**, **B**, **C**, **D** & **E** in the *equation* (*5.2.1.6*). Finding sequence is the basic factor of the FX time series, because of the windowing size, training pattern and time delay will be affected.

According to the results, the value of *h*-parameter does not affect the prediction. A *Five Matrix Fitness* (see details in *Chapter 5.4.2.2*) method was used. This method solves the matrices simultaneously with *equation* (*5.4.2.2*) of PIM optimization. The best MAE result is **A→E→B→C→D** sequence, but the price up/down direction is the most important analysis in marketing practice. Finally, the matrix finding sequence **A→B→C→D→E** is selected.

5.5.5.2 *Time Delay*

Program name:	AM_DELAY.m
Parameters:	Forecasting testing = 50×4 times
Preprocessing:	RVD is $[x(t)-x(t-1)]/SD$
	Unique interval of APIM is from −1 to +1
Formula:	1 to 5 days sequence is shown in *Chapter 5.4.1*
	Equation (*5.4.2.1.1*) for 2 days (no sequence)

MAE analysis	Training Patterns *(Length of Training Data Set)*					Total
	130	195	260	325	390	
No Time Delay	0.4701	0.4660	0.4674	0.4643	0.4641	2.3319
2 Days Time Delay	0.4724	0.4681	0.4696	0.4675	0.4663	2.3439
3 Days Time Delay	0.4724	0.4656	0.4653	0.4646	0.4637	2.3316
4 Days Time Delay	0.4753	0.4677	0.4679	0.4672	0.4698	2.3479
5 Days Time Delay	0.4882	0.4769	0.4760	0.4694	0.4680	2.3785
2 Days (No Sequence)	57.2452	38.1361	71.3336	50.2006	52.9557	269.8712

Table 5.5.5.2

Direction matching (Up / Down)	Training Patterns *(Length of Training Data Set)*					Total
	130	195	260	325	390	(%)
No Time Delay	137	137	136	139	142	69.1
2 Days Time Delay	134	137	138	141	140	69.0
3 Days Time Delay	137	134	141	140	144	69.6
4 Days Time Delay	135	135	140	139	139	68.8
5 Days Time Delay	136	139	142	142	139	***69.8***
2 Days (No Sequence)	108	88	98	98	100	49.2

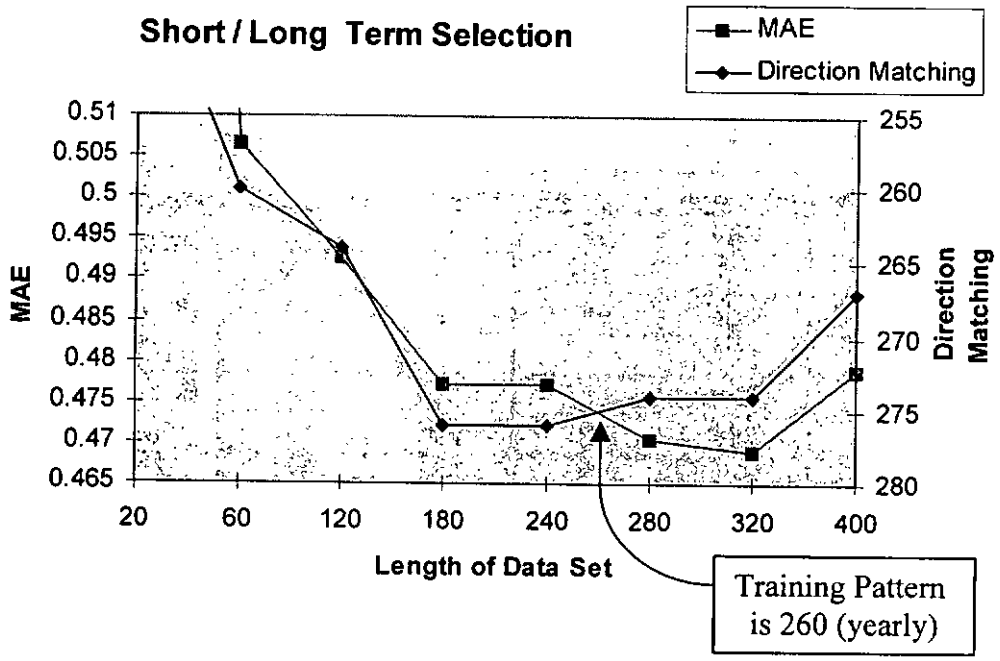
In *table 5.5.5.1* and *table 5.5.5.2*, the prediction result of *Two Matrix Fitness* is the worst optimization method in FX time series. These results extremely difference with other numerical optimization method. From these two testing, it can be concluded that the *Two Matrix Fitness* will not be adopted in my work further.

The preliminary test of the parameters is in *Chapter 5.5.5.1*. According to *table 5.5.5.1*, The long term is the best performance of the system. Therefore, the training pattern was chosen from 130(half year) to 390(one and half years). From the MAE results, the time delay affects the predicted value very little. The best result of MAE is 3 days time delay, but 5 days time delay is the best result in direction matching analysis. The direction matching is the main consideration. Therefore, the 5 days time delay is selected.

5.5.5.3 *Long / Short Term Determination*

Program name:	AM_TERM.m
Parameters:	Time delay = 5 days
	Forecasting testing = 100×4 times
Preprocessing:	RVD is [x(*t*)–x(*t*–*1*)]/SD
	Unique interval of APIM is from –1 to +1
Formula:	5 variables A,B,C,D&E is showed in *Chapter 5.4.1*

According to *chapter 5.5.4.6*, the training pattern (short/long term) will be re-tested. The other parameters are based on the results of *Chapter 5.5.5.1 & 5.5.5.2*, but the testing samples are increased to 100×4 times. The results are tabulated in table 9.4.3, and two curves results are illustrated as below.



From the figure, the best result of MAE or Direction matching is the middle of the curve. It means the best result is not the extreme short or long term. This result is the intercept of MAE and Direction matching curves. The final selection of the term is 260(yearly) from the above figure.

5.5.5.4 *Linear / Nonlinear Selection*

Program name: AM_LINEA.m
Parameters: Time delay = 5 days
Forecasting testing = 100×4 times
Preprocessing: RVD is $[x(t)-x(t-1)]/SD$
Squashing data of LSE is $(1-e^{-\alpha})/(1+e^{-\alpha})$ or $1/(1+e^{-\alpha})$
Unique interval of APIM is from −1 to +1
Formula: 5 variables A,B,C,D&E is showed in *Chapter 5.4.1*

MAE analysis		Training Patterns *(Length of Training Data Set)*					Total
		22	65	130	260	325	
Linear $h=0$	Unique interval	0.6701	0.5094	0.4928	0.4799	0.4695	2.6217
	$(1-e^{-\alpha})/(1+e^{-\alpha})$	1.6218	1.3343	1.2905	1.2599	1.2308	6.7373
	$1/(1+e^{-\alpha})$	1.0903	0.8647	0.8457	0.8277	0.8226	4.4510
Nonlinear $h=0.7$	Unique interval	0.9620	0.6276	0.5876	0.5702	0.5678	3.3152
	$(1-e^{-\alpha})/(1+e^{-\alpha})$	2.0431	1.5123	1.3860	1.3363	1.2958	7.5735
	$1/(1+e^{-\alpha})$	1.3140	0.9879	0.9397	0.8735	0.8730	4.9881
Multiple $h=1$	Unique interval	1.3800	0.8280	0.7525	0.7328	0.7292	4.4225
	$(1-e^{-\alpha})/(1+e^{-\alpha})$	2.3070	1.6965	1.5181	1.4400	1.3844	8.3460
	$1/(1+e^{-\alpha})$	1.7678	1.1210	0.9654	0.8582	0.8449	5.5573

Table 5.5.5.4

Direction matching (Up / Down)		Training Patterns *(Length of Training Data Set)*					Total (%)
		22	65	130	260	325	
Linear $h=0$	Unique interval	242	263	264	273	274	***65.8***
	$(1-e^{-\alpha})/(1+e^{-\alpha})$	198	193	189	185	198	48.15
	$1/(1+e^{-\alpha})$	166	179	167	172	169	42.65
Nonlinear $h=0.7$	Unique interval	193	238	247	234	238	57.5
	$(1-e^{-\alpha})/(1+e^{-\alpha})$	188	208	202	211	209	50.9
	$1/(1+e^{-\alpha})$	179	185	184	185	183	45.8
Multiple $h=1$	Unique interval	175	194	192	168	174	45.15
	$(1-e^{-\alpha})/(1+e^{-\alpha})$	195	203	203	212	212	51.25
	$1/(1+e^{-\alpha})$	179	184	182	175	172	44.6

5.5.5.5 *Summary*

Once again determining a linear or nonlinear model of system is the main target in this chapter. Three different *h*-parameter performance, unique interval and nonlinear transformation of normalization are re-tested. Obviously, the linear modeling is better than nonlinear modeling. And the symmetric sigmoid function $(1-e^{-\alpha})/(1+e^{-\alpha})$ is suitable for direction matching of FX time series, but it gets more MAE in the nonlinear modeling.

5.5.6 Adaptive Matrix / LSE Method

Program name: AM_LSE.m
Parameters: Forecasting testing = 100×4 times
Preprocessing: RVD is $[x(t)-x(t-1)]/SD$
Squashing data of LSE is $(1-e^{-\alpha})/(1+e^{-\alpha})$
Unique interval of APIM is from –1 to +1
Formula: 3 variables A,B&C is showed in *equation (5.3.1.6)*
5 variables A,B,C,D&E is showed in *Chapter 5.4.1*

MAE analysis				Training Patterns *(Length of Training Data Set)*					Total
				22	65	130	260	390	
Time Delay	3	APIM	Linear	0.6471	0.5438	0.5294	0.5162	0.5079	2.7444
			$h = 0.7$	1.1039	0.7016	0.6589	0.6258	0.6041	3.6943
		LSE	Linear	3.1764	1.6945	1.4940	1.3105	1.2947	8.9701
			$h = 0.7$	3.3030	3.0598	2.9422	2.8196	2.4447	14.5693
	5	APIM	Linear	0.7446	0.5660	0.5476	0.5332	0.5157	2.9071
			$h = 0.7$	1.5312	0.8987	0.7994	0.7365	0.7277	4.6935
		LSE	Linear	9.5011	2.2082	1.7761	1.5577	1.5084	16.5515
			$h = 0.7$	2.8152	3.0982	2.9772	3.0032	2.6506	14.5444

Table 5.5.6

Direction matching (Up / Down)				Training Patterns *(Length of Training Data Set)*					Total (%)
				22	65	130	260	390	
Time Delay	3	APIM	Linear	257	266	270	281	285	***67.95***
			$h = 0.7$	221	255	259	164	265	58.2
		LSE	Linear	212	208	204	207	191	51.1
			$h = 0.7$	211	210	209	215	206	52.55
	5	APIM	Linear	242	263	264	273	279	***66.05***
			$h = 0.7$	189	228	240	222	219	54.9
		LSE	Linear	203	199	204	192	183	49.05
			$h = 0.7$	205	213	207	208	195	51.4

For the best model from the above table is linear modeling with 3-day time delay. It has contradiction with the previous results *Table (5.5.5.2)*. By observation from *Table (5.5.5.1)* & *(5.5.5.2)*, although some parameters are different, 3-day and 5-day having very closely results. From the above table, the 5-day results of direction matching are more than 66%, and only less than 2% of 3-day results. And other models are below 58%. They have an obvious advantage from others. Therefore, the time delay may be 3 or 5 days in APIM method. According to weekly effect of FX time series, the 5-day is preferred to use.

5.5.7 Summary

In this chapter, two numerical optimization methods were introduced by using the same data set. The Direction Matching is the main selection strategy because the performance of MAE is dependent on the range of normalization of data. For instance, the normalization of data unique interval range [–1,+1] has more MAE than the range [–0.1,+0.1]. Therefore, MAE is a tool to check the extreme error from the system, such as the result of finding sequence **A→E→B→C→D** in *Chapter 5.5.5.1*, or the second criterion of the system, an example as in *Chapter 5.5.5.3*.

In past studies, the usual procedure was to select a group of independent parameters or variables and reduce the parameters' range of the system. By following the decision tree in *Chapter 5.5.4 & 5.5.5*, the results of parameters are repeated testing again and again. Although the results are very close correlation to each of others, the final result is obtained that the LSE method is suitable for nonlinear system, and APIM method is suitable for linear system. By observations of tables in this chapter, better results may be achieved by different parameters' range. It means the parameters could be in a certain range, for instance, long term windowing range is 260 to 390.

By the using Adaptive Pseudo-Inverse Matrix (APIM) numerical optimization, the linear system is very stable for output FX nearest neighbor prediction. According to next day FX trend, we cannot make a decision to buy or sell a currency. Therefore, a further studies of day after prediction (second neighbor) or option price should be needed for recommendation to buy/hold/sell a currency. This important issue will be discussed in following chapters.

Chapter Six

FORECASTING

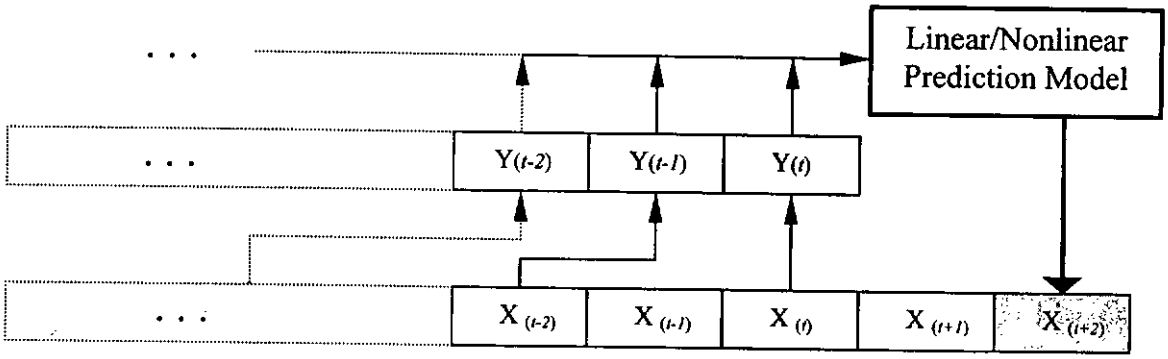
- Second Neighbor Prices
- Mixed Optimization
- Fuzzy System Decision

6.1 SECOND NEIGHBOR PRICE

A day after price is needed for making a buy/hold/sell decision. There are two RNN modeling methods introduced to predict the second neighbor rate of FX time series. They are Taylor Expansion method and Double Interval method, which are compared in *Chapter 6.1.3*. The traditional statistical model will be evaluated in *Table 9.4.4*.

6.1.1 Double interval (DI) Method

The below modeling structure could be used in RNN modeling or statistical modeling. The time lag is double of the nearest neighbor system, which was described in previous chapters. A new data set as $Y(t)$, $Y(t-1)$, $Y(t-2)$, ... vectors is created by double interval of raw data set as $X(t)$, $X(t-2)$, $X(t-4)$, ... vectors. Therefore, a double number of raw data is needed for data normalization.



6.1.1.1 *RNN Modeling*

The parameters of RNN are determined by previous studies in *Chapter 5.5.4 & 5.5.5*. The above method proposed here provides a simple technique for general case of data normalization. The modeling is the same as the nearest neighbor prediction. If the data interval increase to five, the prediction becomes a weekly option price of FX. But different parameters, windowing and normalization of raw data will be needed, such as moving average is a better method for longer interval. In my works, the *equation (5.2.1.6)* is used for the nearest neighbor prediction, and the results are tabulated in *Table 6.1.3*.

6.1.1.2 *Simple Exponential Smoothing*

Although this is one kind of time series equations, it is a *separate modeling*. Each univariate FX time series USD/DEM, USD/GBP, USD/CHF & USD/JPY, was analyzed separately, without utilizing their interdependencies. For example, only the values of $X(t)_{USD/GBP}$ & $X(t-1)_{USD/GBP}$ were used to predict $\hat{X}(t+1)_{USD/GBP}$. The equation is as following:

$$\hat{X}(t+1)_{USD/GBP} = \alpha X(t)_{USD/GBP} + (1-\alpha) X(t-1)_{USD/GBP} \qquad (6.1.1.2.1)$$

Table 6.1.1.2

Program name:	UPDOWN4.m
Parameters:	Forecasting testing = 1000×4 times
Preprocessing:	RVD is $[x(t)-x(t-1)]/SD$
	Squashing data of RVD is $(1-e^{-\alpha})/(1+e^{-\alpha})$
	Unique interval of APIM is from −1 to +1
Formula:	*Equation (6.1.1.2.1)*

Direction matching (Up / Down)		Alpha								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\frac{1-e^{-\alpha}}{1+e^{-\alpha}}$	Nearest neighbor	1977	1966	1952	1919	1893	1858	1832	1831	1821
	Second neighbor	2055	2047	2045	2044	2052	2037	2029	2035	2016
Unique Interval	Nearest neighbor	2641	2655	2677	2687	2692	2677	2649	2634	2603
	Second neighbor	2677	2706	2731	2718	2720	2720	2697	2666	2654

The new alpha should be found again because RVD was used to normalize data in a range [−1,+1], in order to get up/down direction. The above results are presented by success matching of up/down direction. The comparison with RNN will be in *Chapter 7.1.1*.

From the above *Table 6.1.1.2*, $\alpha = 0.5$ for nearest neighbor prediction

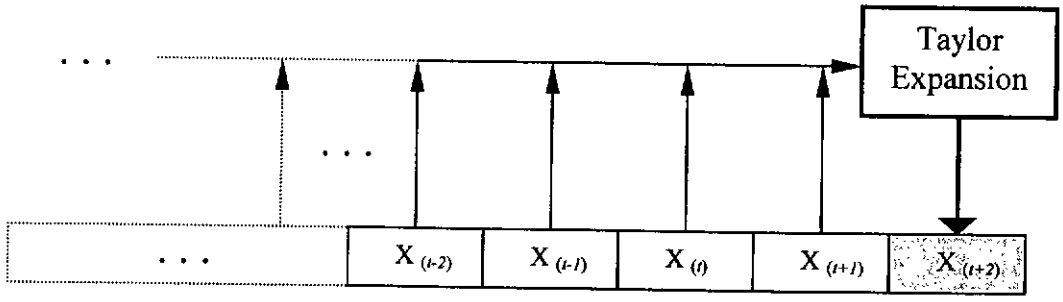
$$\hat{X}(t+1) = \frac{1}{2}[X(t) + X(t-1)] \qquad (6.1.1.2.2)$$

The above equation is the same as MA equation. This means in a long run, SES or MA modeling have the same results in nearest neighbor.

From the above *Table 6.1.1.2*, $\alpha = 0.3$ for second neighbor prediction

$$\hat{X}(t+1) = 0.3\,X(t) + 0.7\,X(t-1) \qquad (6.1.1.2.3)$$

6.1.2 Taylor Expansion (TE) Method



The following equation derived from *Chapter 5.2.2* and *equation* (*5.2.2.1*).

$$x(t+2) = x(t) + 2h\frac{dx(t+1)}{dt} \qquad (6.1.2.1)$$

$$x(t+2) = x(t) + 2h\{-x(t+1) + \sigma\,[\,\mathbf{A}x(t+1) + \mathbf{B}x(t) + \mathbf{C}x(t-1) + \ldots + \theta\,] + J\,\}$$

From above equation, nearest neighbor data $X_{(t+1)}$ should be given by prediction value in practice. Here, the vectors of $x(t+1)$ and $\mathbf{A}x(t+1)$ are found by *equation* (*5.2.1.6*) which are the predicted values of the system. According to Chapter 5.5.6, the predicted value of nearest neighbor data $X_{(t+1)}$ is predicted by a linear system using APIM numerical optimization.

The Advantage of this nonlinear modeling method is using the same data set of FX time series, and all the parameters, windowing size, ... etc, could be the same as nearest neighbor prediction. According to double interval method, half set of training data is lost in the FX time series. But Taylor series is a complete FX time series data set, not just an odd/even selected interval data set. Therefore, it makes the system simpler and robustness.

In the nonlinear part of equation (*6.1.2.1*), the *h*-parameter value is double of the nearest neighbor prediction, hence it may be less than 0.7. The cycling testing strategy will use again in *Table 9.4.6*. Finally, the results of comparison with double interval method are tabulated in *Table 6.1.3*.

6.1.3 Second Neighbor Prediction

Program name: UPDOWN_U.m and UPDOWN_N.m
Parameters: Time delay = 5 days
Forecasting testing = 100×4 times
Preprocessing: RVD is $[x(t)-x(t-1)]/SD$
Squashing data is $(1-e^{-\alpha})/(1+e^{-\alpha})$
Unique interval is from −1 to +1
Formula: *Equation* (*5.2.1.6*) for Double Interval method
Equation (*5.2.2.2*) for Taylor Expansion method

MAE analysis			Training Patterns *(Length of Training Data Set)*					Total
			22	65	130	260	390	
Unique interval	Taylor	$h = 0.7$	1.2312	0.8667	0.7981	0.7683	0.7436	4.4079
	Double Interval	Linear	1.2988	0.9132	0.8526	0.8016	0.7708	4.6370
		$h = 0.7$	1.0005	0.7440	0.6840	0.7177	0.7116	3.8578
$\frac{1-e^{-\alpha}}{1+e^{-\alpha}}$	Taylor	$h = 0.7$	7.4127	6.0955	6.0129	5.1778	4.0190	28.7179
	Double Interval	Linear	3.1274	1.7961	1.6200	1.5576	1.3935	9.4946
		$h = 0.7$	3.1258	3.2289	2.8733	3.3267	2.6544	15.2091

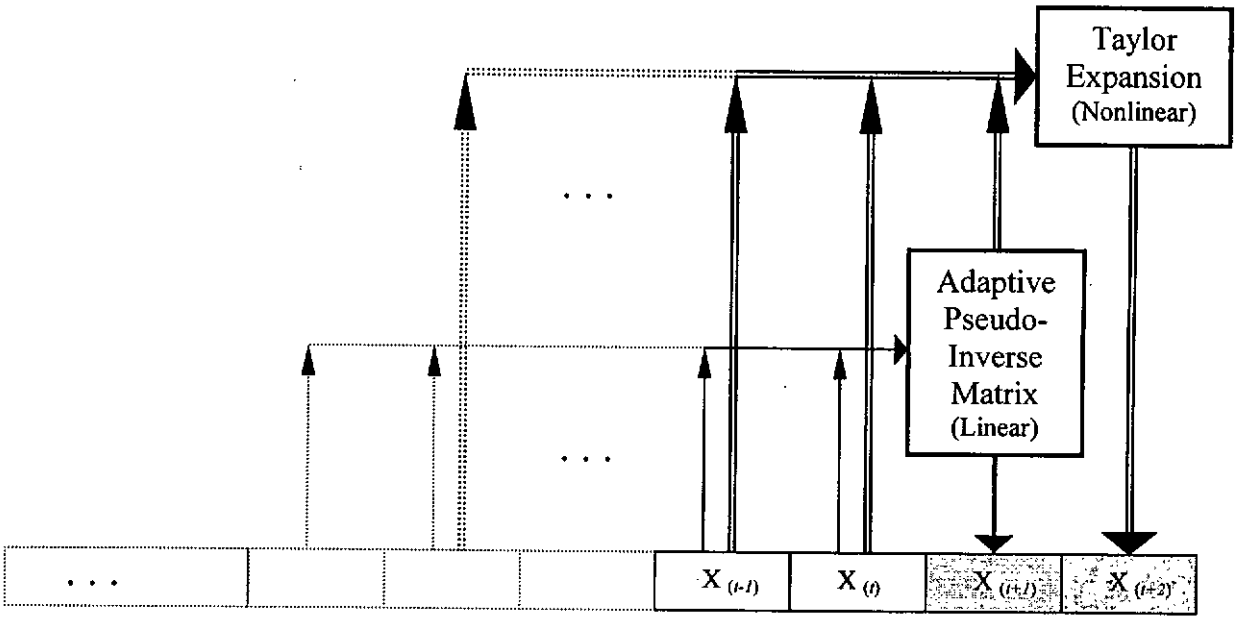
Table 6.1.3

Direction matching (Up / Down)			Training Patterns *(Length of Training Data Set)*					Total (%)
			22	65	130	260	390	
Unique interval	Taylor	$h = 0.7$	222	250	239	243	255	***60.45***
	Double Interval	Linear	219	207	211	209	203	52.45
		$h = 0.7$	213	233	231	214	213	55.2
$\frac{1-e^{-\alpha}}{1+e^{-\alpha}}$	Taylor	$h = 0.7$	215	205	212	216	217	53.25
	Double Interval	Linear	201	196	192	189	182	48.0
		$h = 0.7$	194	194	199	199	199	49.25

The above results are given by two MATLAB® programs "UPDOWN_U.m" and "UPDOWN_N.m". Unique interval ("UPDOWN_U.m" program) and nonlinear transformation ("UPDOWN_N.m" program) normalization are used for testing the performance of RNN. The training pattern is long term in nonlinear modeling, but short term in linear modeling.

Obviously, the nonlinear system is the best optimization model for second neighbor prediction, either the *Taylor Expansion* method or *Double Interval* method. The final selection of the predict is Taylor Series Expansion method, which is up to 60% in success direction matching.

6.2 MIXED OPTIMIZATION



According to previous studies, a mixed optimization method is used. It includes linear APIM method model and nonlinear TE method model. The above figure combines the nearest and second neighbor prediction system as a whole. Single lines are data flow of nearest neighbor prediction, and double lines for second neighbor prediction.

This combined modeling obtained a considerably improved performance using information from the same completed data set of FX time series, which includes noisy data. This is a multivariate time series consists of sequences of values of contemporaneous USD/DEM, USD/GBP, USD/CHF & USD/JPY rates changing with time (daily). And this FX time series is significantly correlated, for example, when similar attributes are being measured at different geographic locations.

Because of using a cycling strategy testing (see details in *Chapter 5.5.2*), a testing program could be used to arrange parameters in order to find a suitable parameters' range of the system. This MATLAB® program named "PREDICT.m" and the testing results are tabulated in *Table 9.4.5 & 9.4.6.*

6.3 FUZZY SYSTEM DECISION

6.3.1 Fuzzy Logic

Predicate logic is a two-valued logic based on traditional set theory. Predicates define classes of objects and objects that satisfy a given predicate are members of the respective class. Inferences in predicate logic are performed using inferring rules. If P and Q are predicates and the symbol "→" is the implication connective (read as IF ... THEN), then it can be summarized as followings:

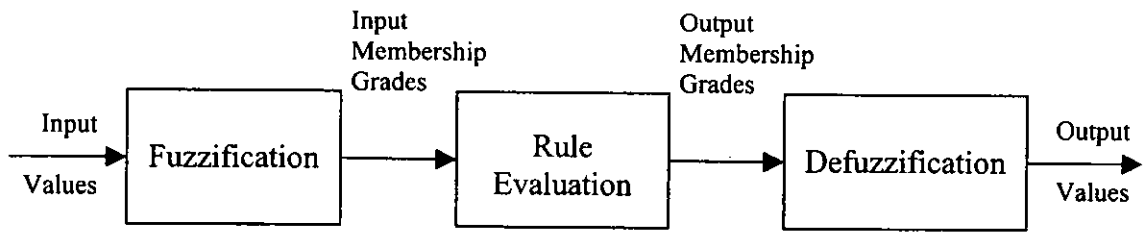
P	assumed true
$P \rightarrow Q$	also assumed true
Q	then conclude

Similar inferring rules have been defined for fuzzy logic based on fuzzy relations among fuzzy subsets. The formal definitions are omitted here. Instead, a fuzzy FX system might be interpreted as:

Premise:	USD/GBP is low
Implication:	USD/GBP and USD/JPY are approximately equal
Conclusion:	USD/JPY is more or less low

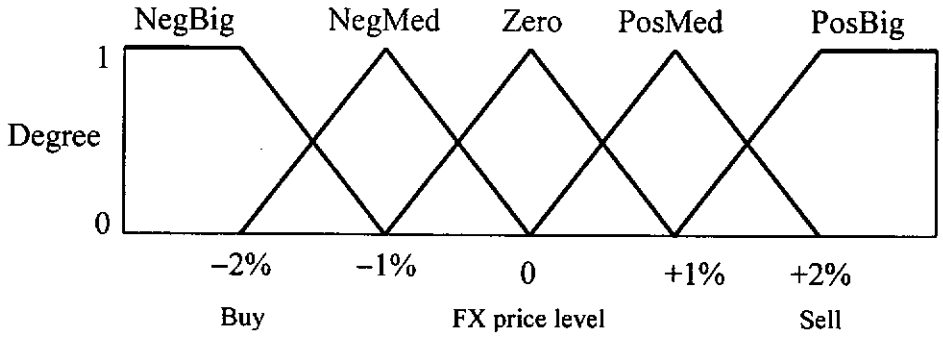
6.3.2 Fuzzy Expert Systems

A popular class of expert systems use "if ... then" rules to represent "chunks" of knowledge. Such systems are known as rule-based systems. Similar expert systems have been implemented with fuzzy "if ... then" rules. These systems are called fuzzy expert systems. They are often able to model commonsense reasoning better than conventional rule-based systems. The basic fuzzy expert system operates in three stages: converting input variable values back to "crisp" output variable values. The process is illustrated in figure below.



A fuzzy decision support system for FX rate is used in the present investigation. The membership functions define for the *next day price* level of FX rate. Nearest neighbor forecast is illustrated in following figure where the triangular shaped fuzzy membership values of negative big (NegBig), medium negative (NegMed), zero, medium positive (PosMed) and positive big (PosBig) are defined.

Membership Function for Nearest Neighbor FX rate

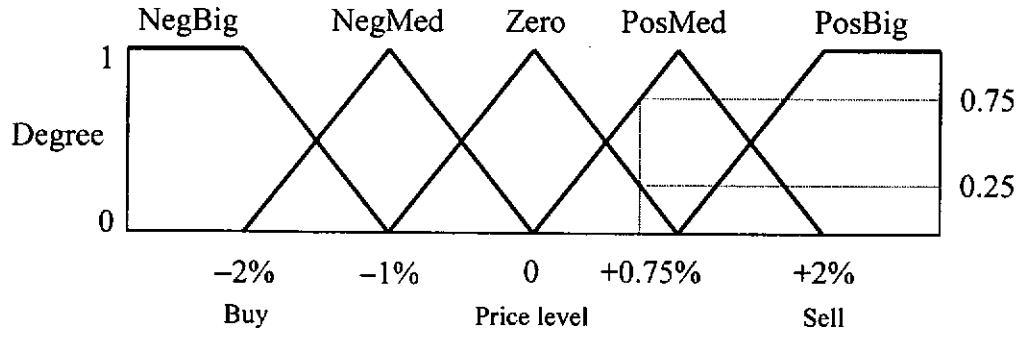


Fuzzification of price level is accomplished by mapping from price value to membership function value. For example, if the price of FX rate is +0.75% the input grades are given by:

LV label	Grade (Value)
PosBig (PB)	0
PosMed (PM)	0.75
Zero (Z)	0.25
NegMed (NM)	0
NegBig (NB)	0

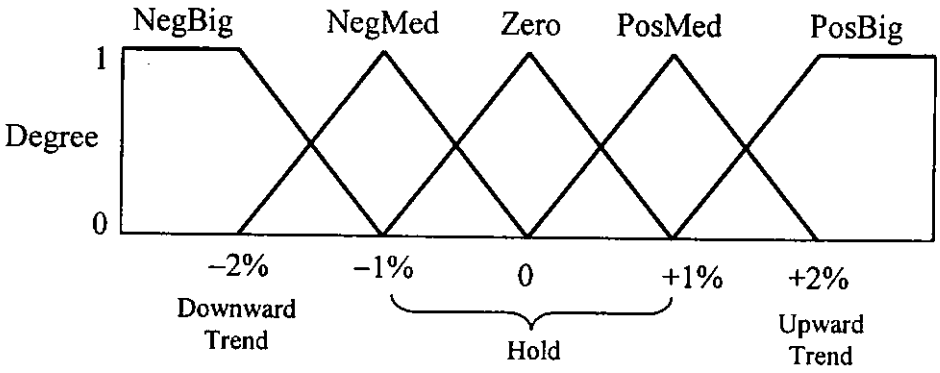
The mapping is illustrated in figure below with doted lines.

The Fuzzification Mapping Process



The other input variable is second neighbor (day after) FX rate. Assume the transaction prices (1%) are the same as the nearest neighbor price (next day). Therefore, the membership function is shown in the following figure.

Membership Function for Second Neighbor FX rate



The next stage of processing is rule evaluation. The fuzzy rules used have two conjuncts (if conditions) in the rule premise corresponding to two fuzzy input variables and one output variable (action). The rules have the following form:

If *Neareat_neighbor* is PosBig and *Second_neighbor* is NegBig Then *FX_price* PosBig

⋮ ⋮ ⋮

If *Neareat_neighbor* is NegBig and *Second_neighbor* is PosBig Then *FX_price* NegBig

A complete decision table for the rules are illustrated in *Table 6.3.2* where the row and column entries are the rule conjuncts and the table value is the corresponding rule action. The final stage of processing is defuzzification, mapping from the "then" part of the rule membership function to variable values —— the recommended action of the system.

Fuzzy Decision Table for FX rate system

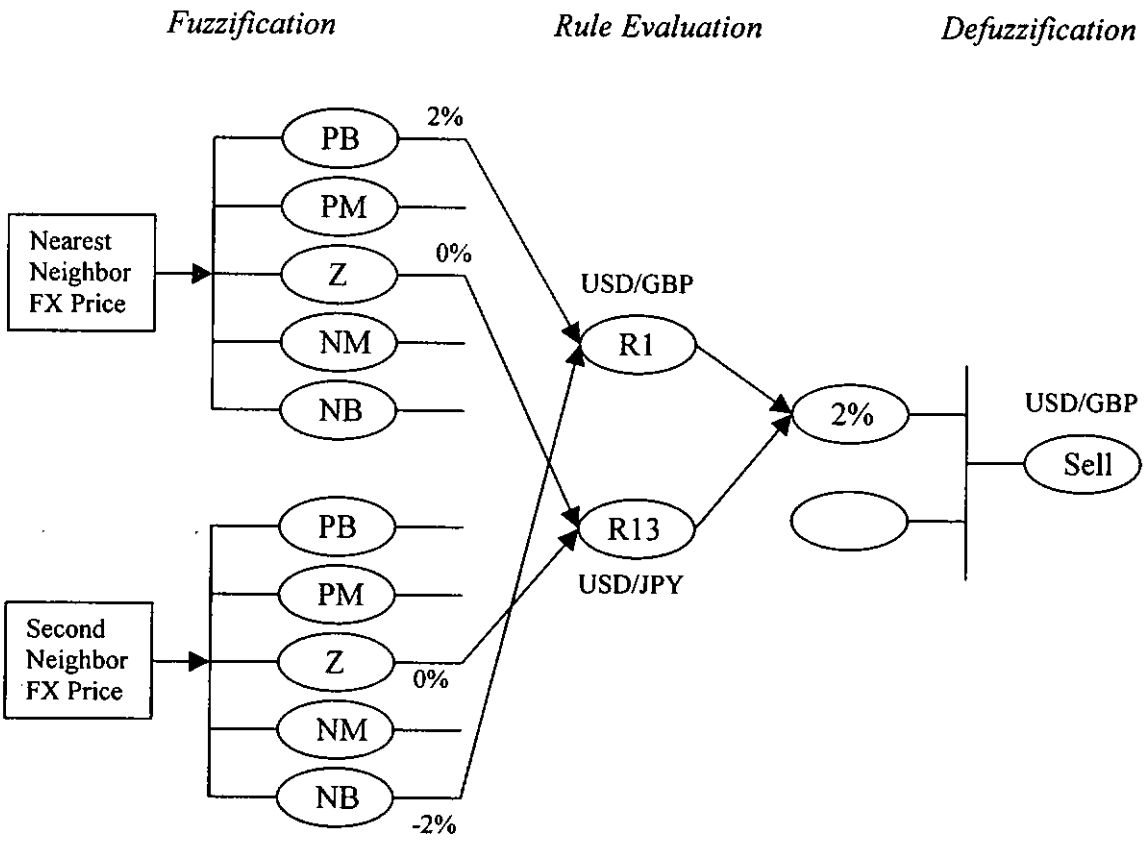
Table 6.3.2

		Second neighbor				
		NB	NM	Z	PM	PB
Nearest neighbor	PB	PB	PM	PM	Z	Z
	PM	PB	PM	Z	Z	NM
	Z	PM	Z	Z	Z	NM
	NM	PM	Z	Z	NM	NB
	NB	Z	Z	NM	NM	NB

6.3.3 Fuzzy Trend Forecasting

A portion of thee complete expert system is illustrated graphically in figure as shown below where the two input variables are nearest neighbor price and second neighbor price and the output variable is the buy / hold / sell recommendation. Fuzzification takes place in the first stage on the left where the fuzzy values are combined and passed to the rules R1 and R25 in parallel. The action parts of the rules are then combined and defuzzified to give the recommended action to buy / hold / sell currency in the FX market.

A Portion of a Fuzzy Expert System



The above diagram shows the complete expert system, but in this dissertation the simple fuzzy system is used. The currencies obey one fuzzy rule only. The defuzzification of each currency separately, not correlation with each other to determine buy / hold / sell.

6.3.4 Fuzzy Decision

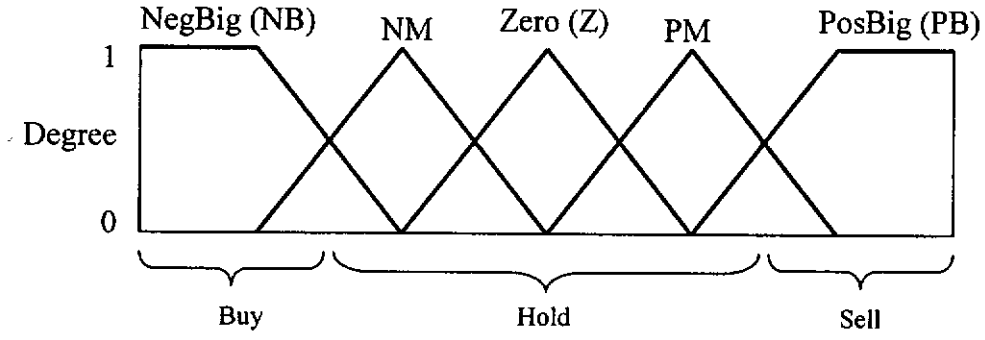
According to fuzzy decision *Table 6.3.2*, the major factor of trend prediction will be affected by PB and NB in second neighbor. The approach for choosing fuzzy decision rules are discussed in *Chapter 6.3.2*. Twenty-five rules are chosen for the training data, but the consequent sets of the fuzzy rules that indicate the trend of FX rates over the second neighbor and expressed linguistically as sell, hold or buy. Actually, nine fuzzy rules are used in my study. Therefore, each currency fuzzy decision table will be simplified as following.

Fuzzy Decision Table for FX rate system

Table 6.3.4

		Second neighbor		
		NB	Z	PB
Nearest Neighbor	PB	PB	PM	Z
	Z	PM	Z	NM
	NB	Z	NM	NB

Membership Function for Output FX rate



The result of fuzzy decision is a three-dimension vector where the component sets are fuzzy rules. The above membership function describes the category that forecasts the trends of the FX rates (buying, selling or holding) over the next period. The final procedure is really a defuzzification procedure using a maximum membership defuzzification scheme. The final results can be represented by three values. NB means the trend is ascending (buy); PB means the trend is descending (sell); otherwise are stationary (hold).

Chapter Seven

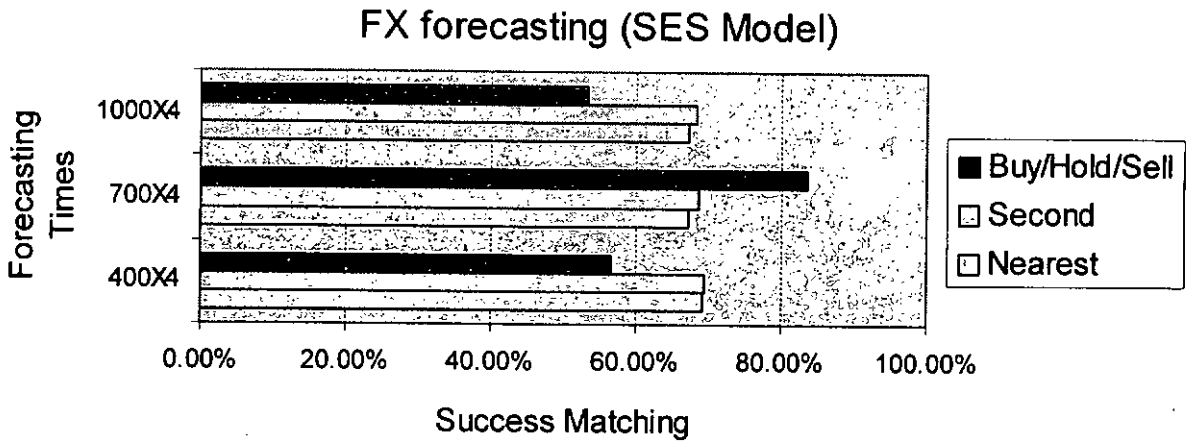
ANALYSIS RESULTS

- Modeling Comparison
- Prediction Results
- Profitability Analysis

7.1 MODELING COMPARISON

7.1.1 Traditional Statistical Model

Since different currencies and different days of the week may have different dynamics, here forecasting using separate modeling which is called **Simple Exponential Smoothing** modeling. By using *equation* (*6.1.1.2*) & (*6.1.1.3*), the nearest neighbor and second neighbor predictions are below 65% success of matching. But the buy/hold/sell decision is unstable and normally the result is approximate to 55%.

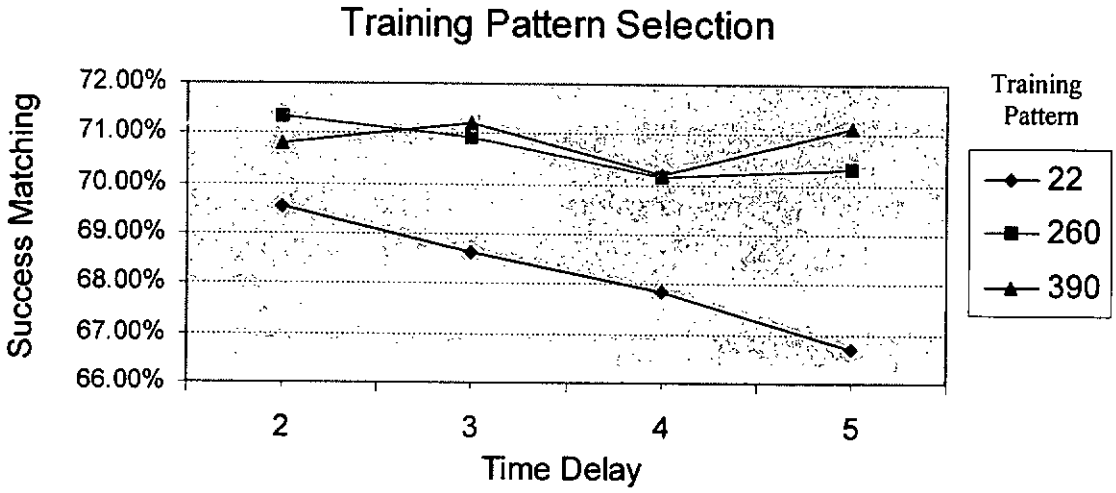


According to the above figure, one of the results is about 80%. This is not a reasonable result because the buy/hold/sell recombination result should be approximate equal to nearest neighbor or second neighbor prediction result. Therefore, this result is omitted.

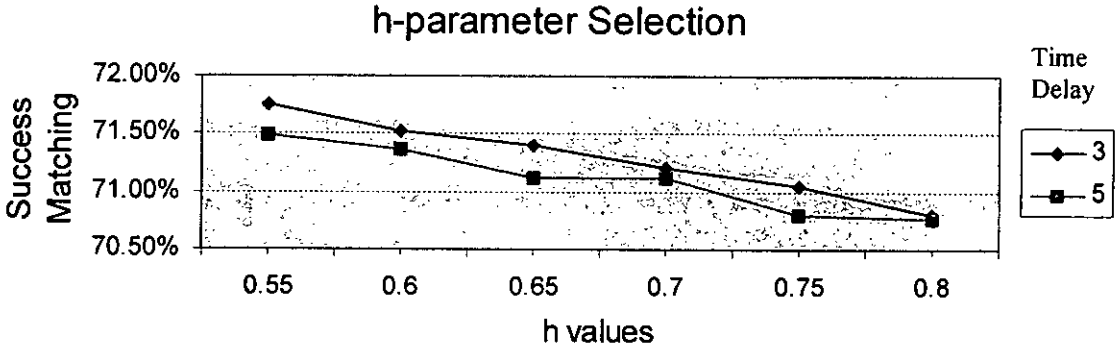
To compare with RNN modeling, a same data set should be used. The comparison data set were the period from 30-6-1993 to 30-6-97, which gives 1000 observations of FX rate. 4 currencies equal to 1000×4 samples, and using this same data to test the RNN modeling. Therefore, 400×4 and 700×4 samplings test the stability of system only, and 1000×4 sampling uses for model comparison. Finally, buy/hold/sell matching is 54% succeed using this SES statistical modeling.

7.1.2 Neural Networks Modeling

This NN modeling is a mixed numerical optimizations of linear and nonlinear. The system structure as shown in *Chapter 6.2*, the parameters are re-tested again. The results are tabulated in *Table 9.4.5 & 9.4.6*, and the figures are illustrated as following.



As shown in above figure, The short term training pattern has lower matching succeed percentage. On the other hand, 260 (yearly) and 390 (1½years) long term of training pattern are very close together. The success matching performance of them is quite stable within the range from 70% to 71.5%, especially the 390 of training pattern. The 3-day and 5-day is the best choice result for the time delay.



The value of *h*-parameter in second neighbor is double of nearest neighbor. The past results of *h*-parameter are approximately equal to 0.7, so for second neighbor prediction, the

h value would be less than 0.7 using Taylor Expansion method. From the *Table 9.4.6*, if h value is less than 0.55, 'Divide by zero' errors will happen during calculation, $h = 0.55$ is negligible because it is very close to the threshold of error. And also the 3-day and 5-day time delay is approximately equal. The final selection of h value is 0.6, and the success matching is 71.5%.

7.1.3 Modeling Selection

There are two main groups of modeling for comparison. They are linear (SES) modeling and nonlinear (RNN) modeling. According to previous studies, final results of two models are as following.

		Neural Network Modeling	SES Statistical Modeling
Total MAE	*Nearest neighbor*	0.55	0.64
	Second neighbor	0.81	0.65
Success matching	*Nearest neighbor*	72.3 %	67.3 %
	Second neighbor	64.8 %	68.3 %
	Buy/Hold/Sell	71.5 %	53.4 %

Obviously, form the above information given that the RNN modeling is better than SES traditional statistical modeling. According to *Table 9.4.4*, SES model is an unstable system. A comparison of the results from *Table 9.4.5 & 9.4.6*, shows that the RNN model is a stable and robustness system.

Classical statistical techniques for prediction reach their limitations in applications with nonlinearities in the data set. SES forecasting method is only capable of picking up general trends and has difficulty in modeling cycles that are by no means repetitive in amplitude, period or shape. Here showed that even simple RNN learning procedures such as the APIM algorithm far outperform current the 'best practice' in a typical application for FX rate within the framework of the arbitrage pricing model. Their smooth interpolation properties allow RNN models to fit better models to the data and to generalize significantly better.

7.2 PREDICTION RESULTS

7.2.1 Forecasting Strategies

Suppose that on each day t, a decision rule is applied with the intention of achieving profitable trades. The price trend, based on market expectations, determines whether the asset is bought or sold. When the asset is bought, the position initiated in the market is said to be 'long'. When the asset is sold, the position initiated is said to be 'short'. A forecasting technique is assessed as useful and will subsequently be used if it has economic value. In short, the forecast is seen as useful if in dealer terms, it can 'make money'. For achieving this purpose, market participants use price-based forecasts. Therefore, the predictor $Y_{(t)}$ is completely characterized by a mathematical function f of past prices $Y_{(t)} = f\{ X_{(t)}, X_{(t-1)}, X_{(t-2)}, \ldots, X_{(t-389)}\}$. Here I select 390 observations in order to receive enough data samples, which is 390+625+3=1015 samples from 30-6-93 to 30-6-97. This is the same data set with SES statistical model for comparison.

For comparison performance of models, the only crucial feature which is required from the forecasting technique is its ability to accurately predict the direction of the trend in order to generate profitable buy/hold/sell signals. The trading signal is determined by Fuzzy Logic system, which is described in *Chapter 6.3*.

The market environment is a one-agent model, which is introduced in *Chapter 7.3.1*. Agents forecasting economic variables use both internal and external information. Internal information (past prices, spreads and volumes, for example) originates from within the market, while external information (the inflation rate, the interest rate) comes from the economic system surrounding the market. My works only consider one-agent models in which agents analyze internal information in order to discover recurrent patterns in the relevant variables.

7.2.2 Forecasting Results & Analysis

According to my previous studies, the final selection of parameters and the models is shown below. This is a typical result of my works.

Table 7.2.2

Program name:	PREDICT.m
Normalization:	RVD is [x(*t*)–x(*t–1*)]
	Unique interval of APIM is from –1 to +1
Parameters:	Forecasting testing = 625 times
	Training pattern = 390 (1½ years)
	Transaction cost = 1%
	Time delay = 3
	h-parameter = 0.6
Modeling:	Linear APIM optimization
	RNN Taylor Expansion method
Forecasting strategy:	Buy/Hold/Sell trend direction
	Mixed models (linear & RNN)
Performance:	MAE = 0.138 (*nearest neighbor*)
	MAE = 0.203 (*second neighbor*)
	MAPE = 0.449 % (*nearest neighbor*)
	MAPE = 0.725 % (*second neighbor*)
	Success matching = 72.28 % (*nearest neighbor*)
	Success matching = 64.76 % (*second neighbor*)
	Success matching = 71.52 % (*Buy/Hold/Sell*)

7.2.2.1 *Direction Matching*

The evaluation I used here is based on a simple trading strategy, which uses the profitability of the modeling as the performance (Trend direction matching) measure. To determine whether a market is trending, I calculate a 2-period FX rate (*next day / day after*) for each discrete time sample. The total of 4 currency rates matching with an assessment of the basic directional movement of the FX market. Specifically, the high and low rates of the market at the current time are compared with the high and low rates at the previous time. Therefore, to verify the effectiveness of this prediction system, a direction matching of buying and selling of FX was done. Buying and selling according to the prediction system with 1% of transaction cost to make a greater profit than the buying and holding. Finally, the strategy could be clearly evaluated and the forecasting is more than 70% succeed.

7.2.2.2 *Mean Absolute Error* (MAE)

The MAE is the most common criterion. But MAE results depend on the to input data set range, so a same data set should be used for comparison with other prediction system. All the MAE results are the total errors of 4 currency rates for comparison in order to enlarge the results for observation. (*The average of success matching are use in order to determine the performance of the system.*)

7.2.2.3 *Mean Absolute Percentage Error* (MAPE)

The MAPE results in *Chapter 2* normally more than 100% because there is a RVD data set, which consists of +ve or −ve values. Therefore, the post-processing of data should be needed. The formula of $x = (\max_x - \min_x)\frac{x_1 - MAX}{MAX - MIN} + \max_x$ (7.2.2.3)

where x_1 is the scaled variable;
x is the original variable;
$\min_x$,$\max_x$ are the minimum and maximum values of original variable of x;
MIN, *MAX* are the values of the target range.

The overall MAPE for the system is very good. In general, again the RNN fit the true model well; the maximum difference in MAPE is less than one percent. The difference in MAPE was lowest for long term training pattern and low noise.

7.2.2.4 *System Stability*

The neural networks have been known to produce wide variations in their predictive properties. This is to say that small changes in network design, learning times, initial conditions, etc. may produce large changes in network behavior. The target here is to identify intervals of values for these parameters which give statistically stable results, and to demonstrate that these results persist across different training and test sets. So, according to *Chapter 5*, all the selected parameters present the RNN in here is a stable system.

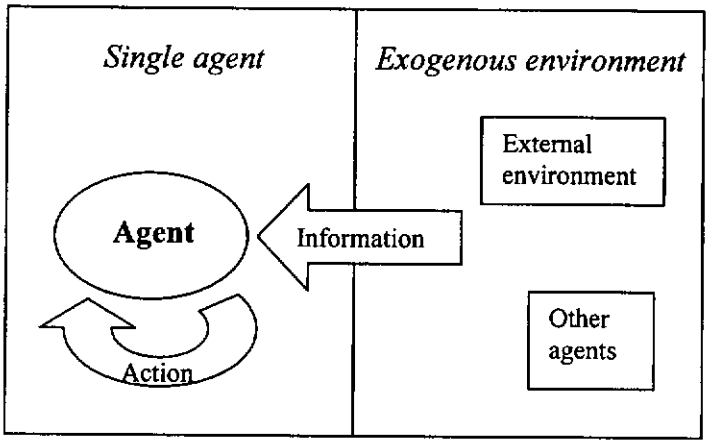
7.3 PROFITABILITY ANALYSIS

This dissertation considers situations where a single agent acts in an exogenous environment by using RNN. The financial trader who may buy or sell FX at a market price that is not affected by the trader's operations. The dissertation layout simple forecasting problems, where the RNN is used for forecasting the values of certain important variables, and more complicated decision problems, where RNN is used as an expert suggesting the action to perform, for example ***buying*** or ***selling***.

7.3.1 One-Agent Model

One-agent model is the first step in complexity. While highly simplified, it represents a good tool for decision-makers because of their adaptability to situations encountered in real markets.

The general structure of one-agent models



In the figure, a one-agent model considers a single agent acting in an exogenous context. Such a context is represented by an external environment, that provides information to the agent and/or other agents living in this same environment. It is essential that the actions of the agent who is studied in the model do not in any way affect the behavior of the environment or the other agents, but only the resources with which the agent is endowed.

By using RNN, the model captures the regularities of the data belonging to the training set. The possible non-stationarity of the time series must be investigated and taken into account, because it can invalidate the generalizability of the model and its practical usefulness as a guide to trading.

7.3.2 Decision Analysis

In the one-agent model of a trader in FX market considered by fuzzy expert system in *Chapter 6.3.3*, a set of fuzzy rules are generally supported by economic laws or by empirical market rules. For example, suppose that an agent acts in a FX market and forecasts the change in price from today to tomorrow and day after to determine his/her action today. A possible fuzzy rule set for choosing among buy/hold/sell actions is described *in Chapter 6.3.4*. The threshold value depends on transaction costs. The hold decision rule is to avoid an action when expected profit is lower than transaction costs.

But there are some main drawbacks to this fuzzy rules decision approach.

(1) **Fuzzy rules are arbitrary**. The way fuzzy rules process the output of the network is decided by the researcher on the basis of expert advice. But experts themselves often disagree on the interpretation of data patterns and are not unanimous in the decision process, so fuzzy rules generally encapsulate the advice of only one expert.

(2) **It can induce undesirable behavior.** The introduced fuzzy rules are simple and seem sensible, but they can induce complete inactivity if daily changes are always below the transaction cost threshold. A more complex set of rules should avoid this problem and take into account possible profits coming from a change in price over a longer time period.

(3) **Forecast, action and learning are difficult to integrate.** The action depends on the rules rather than on the output of the network, so it may be difficult to relate the profitability of the action to the forecast: a bad action in terms of economic outcome may follow a good forecast and vice versa. In this approach, those rules cannot be subject to learning because, in most cases, there is no way to back-propagate a measure of the profitability of action to the forecast for learning purposes.

One the other hand, simplicity is a merit of this approach. It does not require complex forms of learning. And also, a *Fuzzy Expert System* may be solved some above problems.

7.3.3 Results and Summary

Forecasting 4 FX rates are demonstrated using a RNN with a hybrid APIM and TE algorithm. The network was run on a PC using MATLAB® programs. This is one of the research and application areas which could potentially give good results is RNN learning on multiple currencies simultaneously and trading in one currency only, for example, here is 4 currencies and using one-agent model in my studies.

In *Table 7.2.2* shows the typical performance of FX rates predictions. Forecasting results are measured by the MAE, MAPE and the accuracy of direction matching. Furthermore, this chapter reports the results of a comparison of the performance of RNN modeling and non-parametric forecasting methods, such as shown in *Chapter 2*, applied to data on four FX rates. As expected, there is negligible linear dependence in the data. And the results consistently indicate a *low-complexity chaotic* behavior.

To summarize, in economics the potential of forecasting methods using non-parametric techniques has probably been underestimated. The results show that daily FX rate changes are dependent of past changes. The RNN presents the FX times series in a nonlinear dynamic structure. The evidence in my work supports the findings of [Hs89] that there is considerable evidence of nonlinear structure in FX rates. Therefore, the empirical results of this study suggest that in the case of FX rates there is *low-dimensionality chaos* and the use of linear techniques can produce significantly better results.

In addition, since inter-day price changes in most currencies tend to affect only the second or fourth decimal place (depending upon the currency pair) the sensitivity of the FX price changes is very low and this generally in poor learning. Therefore, the FX price inputs are not a desirable data set. Here, I solve this problem by using RVD preprocessing.

Chapter Eight

DISCUSSIONS

- Market Analysis
- Future Enhancement
- Conclusions

8.1 MARKET ANALYSIS

8.1.1 Previous Works

The statistical properties of the returns on financial assets have been extensively studied in the finance literature. Previous work has explored the properties of deterministic systems that exhibit apparently random dynamic behavior, or chaos. Most relevant to the work presented here, [Le92] have found empirical evidence that the time path with *nonlinear deterministic* models rather than purely *stochastic* behavior. Furthermore [Hs89][Hs91] applied ARCH models to the analysis of the returns of exchange rates and stock returns. So great attention is paid in the econometric literature, especially in applications to financial data, see [Bo92]. The results of these studies show that models with conditional heteroscedasticity are able to capture the statistic properties of the daily changes, and report evidence of nonlinearity in exchange rates.

In a recent paper [Da95] proposed the heterogeneous market hypothesis. This hypothesis regards the FX market as composed of agents with different dealing constraints and thus heterogeneous expectations. Experience and the initial study before show the behavior of FX time series to be a problem which can be addressed by neural network learning methods. Systematic study and application of neural network techniques in forecasting FX time series can give very promising results.

To summarize previous works, the behavior of daily and weekly FX rates is the **high frequency data** (HFD). This conclude that [Dr95], first, FX time series appear to be both asymmetric and Lyapunov exponents. Second, it is apparent that these series do not conform to white noise processes. Third, allowing for serial dependence raises the question as to whether one is confronted with linear or nonlinear processes. Finally, the assumption that high frequency exchange rates can be described by *low-dimension chaotic* systems has clearly been rejected.

8.1.2 Heterogeneous Market Modeling

- The description of the economic system is based on a mathematical modeling of the individual agents in terms of maximization problems subject to various constraints. For example, the behavior of consumers depends on preferences which are stable and independent of the environment, while the choices of firms are derived from maximization of profits.
- In large economies the interactions among agents take place by means of a set of demand and supply functions, with prices being considered as signals which make all the actions compatible. In small economies or in interpersonal interaction, agents may behave in a strategic way. The models make enough simplifying assumptions to allow the researcher to derive strong implications in terms of observable variables, which can then be tested by means of statistical techniques.
- In a heterogeneous market, there is no particular trading strategy that is systematically better than all the others. Excess return can be gained for different trading profiles, so various ways of assessing the risk and return of trading models are needed.
- The most profitable models actively trade when many agents are active in the market and do not trade at other times of the day and on weekends. The heterogeneous market hypothesis attributes the profitability of trading models to the simultaneous presence of heterogeneous agents, whereas the classical efficient market hypothesis relates this profitability to inefficiencies.
- The different geographical components of the FX market have different business hours according to different time zones and, on the assumption of the heterogeneous market hypothesis, different strategies. Therefore, there are disruptions in the market behavior from one geographical component to the next.

8.1.3 Geographical Effect

The FX market is worldwide, but the dealers differ in their geographical locations (time zones), working hours, time horizons, home currencies, access to information, transaction cost, and other institutional constraints.

In some cases, investors choose to take positions in a financial market at different times of day. This is frequently the case with the 24-hour FX market. Most investors initiate their positions during their time zone. Geographical components related to the business hours of different trading centers must be taken into account.

The FX market is perhaps the only market which is open virtually 24 hours a day, five days a week. The market opens early in New Zealand and Australia, followed by Japan. A little later, several trading centers in the Far East, such as Hong Kong and Singapore, enter the global market. As market closes in Middle East and Europe, closely followed by the United Kingdom. While the UK is still actively trading, North America to close, New Zealand and Australia return to the market and the next 24-hour cycle begins, On Saturday and Sunday very limited trading is reported from the Middle East markets.

On the other hand, with increasingly efficient communication and rate distribution systems available from various rate and market information vendors around the world, the trading network has expanded at a very rapid pace — dependable and consistent spot rates are available in every geographical location with almost equal efficiency. Therefore a FX market does not provide any geographical locational arbitrage such as one sometimes finds in other markets.

8.1.4 FX Market Characteristics

- Trading volumes are very heavy, particularly in four major currency pairs USD/DEM, USD/CHF, USD/GBP, and USD/JPY and also some cross-currency pairs such as DEM/GBP, JPY/GBP, JPY/CHF. Therefore the market is considered deep and very liquid for the purpose of initiating any trade and closing the trade at any time of the day. In general, FX market trades can be completed in few seconds.
- Because of low transactions costs, it is easy to identify good trades during intra-day market movements. Also it is sometimes possible to turn around two or three good trades during the day itself. This is not possible if only open, high, low and closing prices are monitored. As discussed earlier, the FX market never closes and so opening, high, low and closing prices are less meaningful.
- Funding costs for overnight FX trading positions are based on the interest differentials of the two currencies involved in trading. These are normally low unless a high interest rate currency is sold against a low interest rate currency. Therefore, in general, it does not cost much (in terms of funding charges) to hold trading positions in this market.
- One very significant point which relates to volatility in the FX market is sudden and sometimes large movements which are caused immediately after the release of some economic data relating to the US and the other major economies of the world. The exact time of release of these data is precisely known, and thus it may be useful if ways can be studied to incorporate this fact into the neural modeling.
- Another significant reason for the increase in trading volume and deep liquidity in FX market is the rapid liberalization of several economies in the world. As more and more economies open up and their currency is floated on the world market, traders, speculators, investors and currency fund managers see increased opportunity to trade on his market, making the market deep and perhaps efficient.

8.2 FUTURE ENHANCEMENT

8.2.1 Evaluations

Although the RNN approach was successful, there are a number of issues that were raised which could be investigated further.

8.2.1.1 *Real time Prediction*

In this study, all the training and testing of models was carried out on 'dead' data; using such models in a real time system presents a number of additional problems. During the studies, all the models were developed and tested with data extracts on a stand-alone machine. The intra-day trading system currently being developed takes a live data feed and is integrated with other trading floor systems. In addition, spurious prices are removed and the models have to produce their predictions in real time.

8.2.1.2 *Profitability*

A buy/hold/sell action is based on building fuzzy decision rules in order to make the maximum profits. An evaluation of the profitability of the action or of the strategy is the next logical step in the decision process. This evaluation is generally based on one or more indicators measuring the effectiveness of the strategy in economic terms. It is impossible to present general criteria for defining such indicators: they are, in fact, highly problem-dependent.

8.2.1.3 *Optimization Strategy*

With regard to class of optimization techniques, RNN is a local optimization technique: it explores the solution space one point at a time, and is likely to be trapped in local maxima. Genetic algorithms can instead be considered as global optimization algorithms, in the sense that they perform a parallel exploration of the space and are able to escape from local maxima. So APIM may be replaced by GA optimization.

8.2.2 Suggestions

From the results presented elsewhere in the literature and it this dissertation, there is now strong evidence that neural network are suitable for task of FX rates prediction and that they are able to outperform traditional statistical method in this area. But some of further works should be done in order to improve predictability of the system.

(1) Collect more significant indicators such as Interest Rate, Foreign stock index, Bond rating, etc.

(2) The system add one more output for prediction of option price of FX, which can robustly to foresee the FX up or down in order to decide the buy/hold/sell action.

(3) Consider the trading models with different geographical markets, not only the closing FX rates.

(4) To determine the market correlation of exchange rates for fuzzy decision system. A complete fuzzy expert system and fuzzy rules are used in order to improve the profitability.

(5) Further research is ongoing to develop and compare other indicator subset algorithms. An important contribution of this work is to focus the attention of neural network researchers on the need for systematic feature preprocessing methodology for the purpose of improving predictability.

(6) The model needs to further analyze the data and present more test cases to have a robust predictor, such as different neurons in hidden layer.

(7) To improve the prediction strategies, more analysis tests should be needed such as HFD, Correlation, Chaotic dynamics, … etc.

(8) In addition, for actively managing positions and for risk-control purposes, it is a good idea to sample more often than once in a day.

8.3 CONCLUSIONS

This study shows that daily FX rate changes are not independent of past changes. I have used the *Direction Matching* test to search for evidence of nonlinear dynamic structure in the FX rate changes. Furthermore, this dissertation presents the results of a comparison of the performance of a SES forecasting method and the RNN prediction applied to data on the FX rate. The results consistently indicate a *low complexity chaotic* behavior. The reason for this is that the RNN model used the APIM technique, outperforms the forecasting results of the non-parametric method and LSE method, respectively. Forecasting performance are measured by the MAPE, the MAE and the accuracy rate of Direction Matching. The MAE has values of approximately 0.15, MAPE is around 0.5%, and accuracy rates of up to 71.5% were obtained. These results lead to the conclusion that the APIM technique can considerably improve the short-term prediction of FX rate changes.

In my study, RNN is used to predict buy/hold/sell recommendation for *one-agent* investor. The interaction of many factors is important in FX rates prediction. Traditionally, research in this area has used statistical regression for prediction purposes. The above showed that RNN has better modeling capabilities than statistical modeling. However, RNN are still not capable of predicting FX time series well. Although RNN provide a good mathematical fit of the data, this does not necessarily lead to good prediction model. Factors such as selection of predictor variables are vitally important in arriving at a good prediction model. RNNs have several advantages over statistical models. No assumptions regarding the distributions of the data need to be made. RNNs are capable of fitting complex nonlinear models to the data. They are flexible and allow the modeler to use different model configurations such as TE method.

According to Chapter 7.3.2, the decision rule (trading rule) of fuzzy decision system is applied with the intention of achieving profitable trades. The FX price trend, based on market expectations, determines whether the asset is bought or sold. From the comparison of statistical modeling, this is equivalent to the assumption that the past correlation matrix **A,B,C,D** & **E** contains information about the FX market. Establishing theoretical Correlations between trading rules has been considered as an extremely difficult task. However, a complete **fuzzy expert system** allows the joint profitability of a set of trading rules to be tested in order to make maximum profits.

Theoretically, FX rates present to characterize the *high frequency data*. According to the results, the linear modeling was used in next day prediction, and nonlinear in day after prediction. Obviously, the FX rates is a **low-dimensionality chaotic** behavior time series. Furthermore, in this work attempts have been made to improve forecasting results by variations of parameters of the **mixed optimization**. The results suggest, however, that A hybrid system of forecasting (Linear & Nonlinear modeling) is the best modeling for solving this contradiction.

In concluding, the resulting model of the relationship between the FX rates and factors of the capital market contains dependencies in time and multiple connections between several influencing factors (major currencies). Therefore RNNs can give more reliable forecasts for FX rates than the comparable statistical techniques.

Chapter Nine

APPENDICES

- Bibliography
- References & Glossary
- Appendix A (Indicators)
- Appendix B (Output Figures)
- Appendix C (Source Codes)

9.1 BIBLIOGRAPHY

[Bi93] BIS (1993)
Central Bank Survey of Foreign Exchange Market Activity in April 1992
Bank for International Settlements (Monetary and Economic Dept.), Basel.

[Bo92] Bollerslev, T., Chou, R.Y. and Kroner, K.F.
ARCH Modeling in Finance: A Review of Theory and Empirical Evidence
Journal of Econometrics, 52, pp.5-59. 1992.

[Da95] Dacorogna, Michel M., Muller, U.A., Jost, C., Pictet, O.V., Olsen,R.B., and Ward J.R.
Heterogeneous Real-time Trading Strategies in the Foreign Exchange Market
European Journal of Finance, Vol. 1, No. 4, pp383-403, 1995.

[Dr95] Drunat, J., Dufrenot, G., Dunis, C., and Mathieu L.
Stochastic or Chaotic Dynamics in High Frequency Exchange Rates?
Working Papers in Financial Economics, Chemical Bank, No. 6, pp1-7, June 1995.

[El90] Elman, J. L.
Finding Structure in Time
Cognitive Science, Vol. 14, pp.179-211. 1990.

[El91] Elman, J. L.
Distributed Representations, Simple Recurrent Networks, and Grammatical Structure
Machine Learning, Vol. 7, pp.195-225. 1991.

[Hs88] Hsieh, David A.
The Statistical Properties of Daily Foreign Exchange Rates
Journal of International Economics, 24, pp129-145. 1988.

[Hs89] Hsieh, David A.
Testing for Nonlinear Dependence in Daily Foreign Exchange Rates
Journal of Business, Vol. 62, pp.339-368. 1989.

[Hs91] Hsieh, David A.
Chaos and Nonlinear Dynamics: Application to Financial Markets
Journal of Finance, 46(6), pp.1839-1877. 1991.

[Ko96] Kohara, K., Fukuhara, Y., and Nakamura, Y.
Selective Presentation Learning for Neural Network Forecasting of Stock Markets
Neural Computing and Applications, Vol. 4, pp.143-148. 1996.

[Le92] LeBaron, B.
Forecast Improvements Using a Volatility Index
Journal of Applied Econometrics, No.7, pp.137-149, 1992.

[Li96] Li, Leong Kwan
Learning Financial Market Behaviours By Neural Networks: A Non-Linear Auto-Regression Approach
Hong Kong Polytechnic University, 1996.

[Pe96] Petridis, V., and Kehagias, A.
A Recurrent Network Implementation of Time Series Classification
Neural Computation, Vol. 8, pp.357-372, 1996.

[Re94] Refenes, A.N., and Azema-Barac, M.
Neural Network Applications in Financial Asset Management
Neural Computing and Applications, Vol. 2, pp.13-29, 1994.

[Se91] Servan-Schreiber, D., A. Cleeremans and McClelland, J. L.
The Representation of Temporal Contingencies in Simple Recurrent Networks
Machine Learning, Vol. 7, pp.166-93, 1991.

[Ti89] Tiao, G.C. and Tsay, R.S.
Model Specification in Multivariate time series
Journal of the Royal Statistical Society, B51, pp.157-213, 1989.

[We90] Werbos, P.J.
Backpropagation through Time: What it Does and How to Do it
IEEE Transcripts on Neural Networks, 78(10), 1990, pp.261-277.

[Wi87] Widrow, B.
Adaline and Madaline: plenary speech
Proc. 1st IEEE International Conf. Neural Networks, San Diego, 1987, pp.143-158.
The IEEE Inc., Piscataway, NJ, 1987.

[Wi90] Widrow, B. and Lehr, M. A.
Thirty years of adaptive neural networks: Perceptron, Madaline and backpropagation
Proc. IEEE, 78, pp.1415-1442, 1990.

[Wi89] Williams, R.J., and Zipser, D.
A Learning Algorithm for Continually Running Fully Recurrent Neural Networks
Neural Computation, Vol. 1(2), pp.270-280. 1989.

9.2 REFERENCE

[1] Deboeck, Guido J.
Trading on the edge — Wiley, 1994

[2] Dunis, Christian
Forecasting Financial Markets — Wiley, 1996

[3] Fausett, L.
Fundamentals of Neural Networks, Architectures, Algorithms & Applications — Prentice Hall, 1994.

[4] Fu, Li Min
Neural Networks in Computer Intelligence, — McGraw-Hill, 1994.

[5] Klir, George J.
Fuzzy Sets and Fuzzy Logic — Prentice Hall, 1995

[6] Peters, Edgar E.
Fractal Market Analysis — Wiley, 1994

[7] Refenes, Abu-Mostafa, Moody & Weigend
Neural Networks in Financial Engineering — Wiley, 1995

[8] Refenes, Apostolos-Paul
Neural Networks in the Capital Markets — John Wiley & Sons, 1995.

[9] Reinhardt, B.M. & Strickland, M.T.
Neural Networks — Springer, 1995.

[10] Skapura, D.M.
Building Neural Networks — Addison Wesley, 1996.

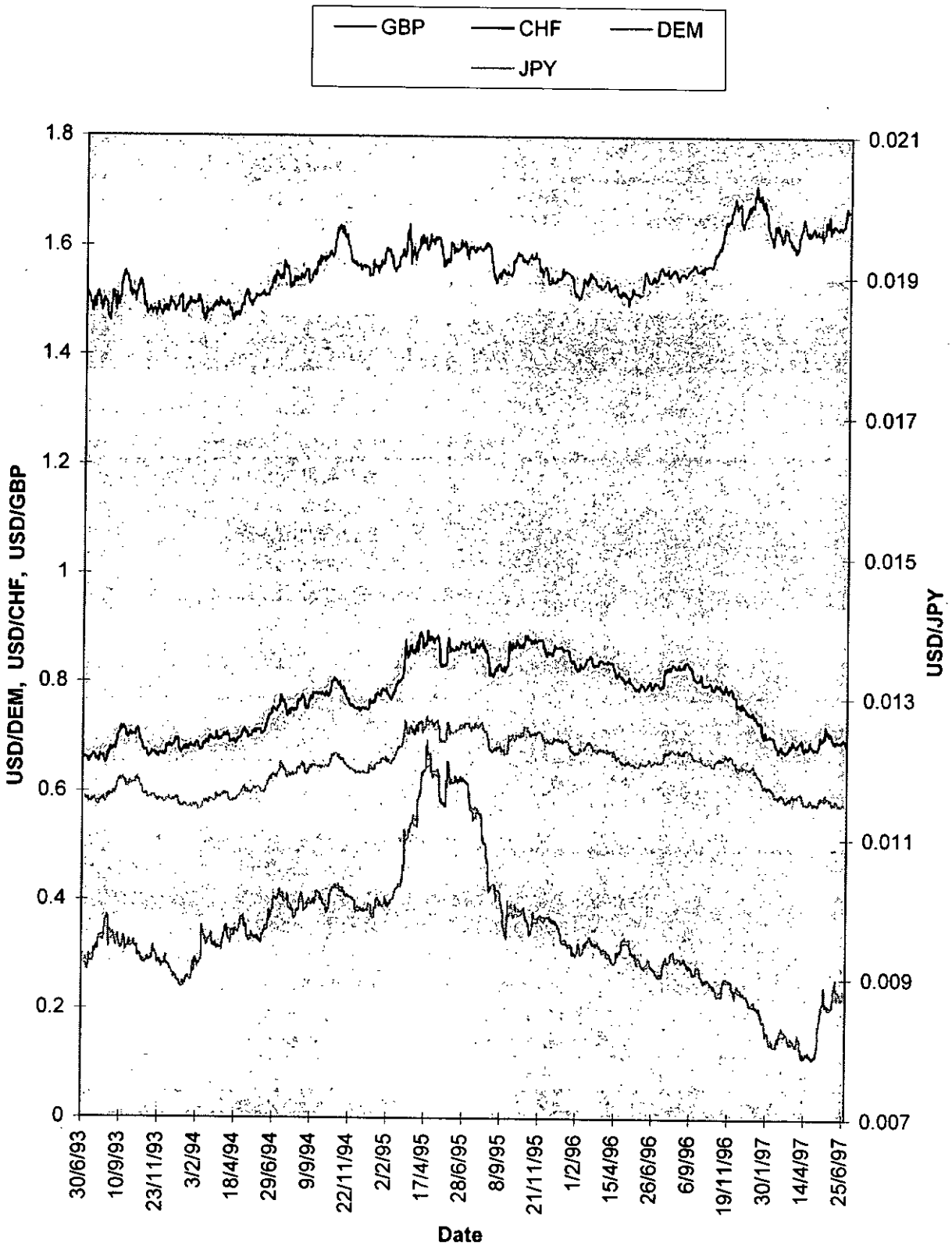
9.3 GLOSSARY

APIM	Adaptive Pseudo-Inverse Matrix Optimization
ARCH	Auto-regressive Conditional Heteroscedasticity
CHF	Swiss Franc
DEM	Deutsche Mark
DI	Double Interval method
FX	Foreign Exchange
HFD	High Frequency Data
GA	Genetic Algorithms
GARCH	Generalized Auto-regressive Conditional Heteroscedasticity
GBP	United Kingdom Pound
JPY	Japan Yen
LMA	Long Moving Average
LMS	Least Mean Square
LSE	Least Square Error Optimization
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Square Error
RNN	Recurrent Neural Network
RVD	Relative Variable Difference
SES	Simple Exponential Smoothing
SMA	Short Moving Average
SRN	Simple Recurrent Network
TE	Taylor Expansion method
USD	United States Dollar

9.4 APPENDICES

Appendix A

Foreign Exchange Rate



Appendix B

Table 9.4.1 **Alpha Selection of SES Model**

Parameters: Forecasting testing = 1000 times

Formula: $\hat{X}(t+1) = \alpha X(t) + (1-\alpha) X(t-1)$; $0 \le \alpha \le 1$

Alpha	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Raw	.00772	.00733	.00697	.00662	.00631	.00604	.00581	.00564	.00555	.00552	.00555
Relative Difference	.00805	.00769	.00743	.00726	.00718	.00721	.00736	.00759	.00793	.00834	.00881

Table 9.4.2 **Period Selection of MA Model**

Parameters: Forecasting testing = 1000 times

Formula: $\hat{X}_{t+1} = \frac{1}{n}\sum_{i=1}^{n} X_{t-i+1}$

Periods (*n*)	2	10	20	30	40	50
Raw	0.005814	0.01033	0.01313	0.01522	0.01689	0.01866
Relative Difference	0.006996	0.005858	0.005618	0.005506	0.005458	0.005474

Table 9.4.3 **Short / Long Term Selection**

Program name: AM_DELAY.m
Parameters: Time delay = 5 days (weekly)
h-parameter = 0.7
Forecasting testing = 100×4 times
Preprocessing: RVD is $[x(t)-x(t-1)]/SD$
Unique interval of APIM is –1 to +1
Formula: *Chapter 5.4.1*

Length of Data Set	20	60	120	180	240	280	320	400
MAE	0.7073	0.5066	0.4926	0.4771	0.4771	0.4704	0.4689	0.4789
Success Matching	243	260	264	276	276	274	274	267

Table 9.4.4 **SES Forecasting Model**

Program name: PREDICT4.m
Parameters: Alpha = 0.5 for nearest neighbor FX rate
Alpha = 0.3 for second neighbor FX rate
Preprocessing: RVD is $[x(t)-x(t-1)]/SD$
Unique interval from –1 to +1
Formula: *Equation* (*6.1.2.2.2*) for nearest neighbor
Equation (*6.1.2.2.3*) for second neighbor

Forecasting Testing	400×4	700×4	1000×4	Average
Total MAE of Nearest Neighbor	0.5308	0.6413	0.6383	0.6035
Total MAE of Second Neighbor	0.5387	0.6564	0.6485	0.6145
Success Matching of Nearest Neighbor (%)	69.3	67.1	67.3	67.90
Success Matching of Second Neighbor (%)	69.4	68.6	68.3	68.77
Success Buy/Hold/Sell Prediction (%)	56.8	83.5	53.4	64.56

Table 9.4.5 **Buy / Sell / Hold** (mixed model)

Program name: PREDICT.m
Parameters: *h*-parameter = 0.7
Transaction cost = 1%
Forecasting testing = 1000 times for Training pattern = 22
Forecasting testing = 750 times for Training pattern = 260
Forecasting testing = 625 times for Training pattern = 390
Preprocessing: RVD is [x(*t*)–x(*t*–*1*)]
Unique interval of APIM is –1 to +1
Formula: *Chapter 5.4.1* for nearest neighbor
Equation (*5.2.2.2*) for second neighbor

	Time Delay	2	3	4	5	Average
Direction matching	Training Pattern = 22	69.53 %	68.63 %	67.85 %	66.70 %	68.2 %
	Training Pattern = 260	71.33 %	70.93 %	70.17 %	70.30 %	70.7 %
	Training Pattern = 390	70.80 %	71.20 %	70.20 %	71.12 %	70.8 %

Table 9.4.6 **Buy / Sell / Hold** (mixed model)

Program name: PREDICT.m
Parameters: Forecasting testing = 625 times
Training pattern = 390 (1½ years)
Transaction cost = 1%
Preprocessing: RVD is [x(*t*)–x(*t*–*1*)]
Unique interval of APIM is –1 to +1
Formula: *Chapter 5.4.1* for nearest neighbor
Equation (*5.2.2.2*) for second neighbor

Direction matching (Up / Down)		Training pattern				Average
		260		390		
	Time delay	3	5	3	5	
h parameter	0.50	***71.80 %***	***70.83 %***	***72.08 %***	***71.80 %***	***71.6 %***
	0.55	71.60 %	70.83 %	71.76 %	71.48 %	71.4 %
	0.60	71.37 %	70.70 %	71.52 %	71.36 %	71.2 %
	0.65	71.23 %	70.50 %	71.40 %	71.12 %	71.1 %
	0.70	70.93 %	70.30 %	71.20 %	71.12 %	70.9 %
	0.75	70.93 %	70.13 %	71.04 %	70.80 %	70.7 %
	0.80	70.87 %	69.97 %	70.80 %	70.76 %	70.6 %

NB: ***Italic*** means 'Divide by zero' errors are happened during calculation.

Appendix C

Source Code

File name: **PREDICT.m**

```
% Sell/Hold/Buy Prediction
% Date:     30-12-97

answer='          Predict Buy/Hold/Sell FX'

clear
h=0.6
hold=0.01         % Transaction cost
len=390           % data set for training
fore=625          % days of forecasting
var=3             % numbers of variables
period=1;         % periods of revise the variables
pos=1;            % Unique interval (MAX)
neg=-1;           % Unique interval (MIN)

load yspm.dat
raw=yspm';
clear yspm;

cur=size(raw,1);          % numbers of currency to forecast
alldata=size(raw,2);      % numbers of data of days
MAE=zeros(2,4);           % Mean Absolute Errors
MAPE=zeros(2,4);          % Mean Absolute Percentage Errors
updown=zeros(2,4);        % predict up/down
buysell=0;

alldata=alldata-1;
for j=1:alldata
  rawx(:,j)=raw(:,j+1)-raw(:,j);
end

for i=1:cur
  nu(i)=min(rawx(i,:));
  de(i)=max(rawx(i,:))-min(rawx(i,:));
end

for i=1:cur
  for j=1:alldata
    dt(i,j)=2*pos*(rawx(i,j)-nu(i))/de(i)+neg;;
  end
end

clear rawx;


for k=1:fore      % Call another dataset

  optdr=raw(:,alldata-fore+k);
  predr=raw(:,alldata-fore+k-1);
  addraw=raw(:,alldata-fore+k-2);
  optd=dt(:,alldata-fore+k);
  pred=dt(:,alldata-fore+k-1);
```

```
for j=1:len+6
  xd(:,len+7-j)=dt(:,j+alldata-len-fore-8+k);
end


if rem(k-1,period)==0 % Revise all variables

  for j=1:len
    yx1(:,j)=xd(:,j);
    x11(:,j)=xd(:,j+1);
    x12(:,j)=xd(:,j+2);
    x13(:,j)=xd(:,j+3);
    x14(:,j)=xd(:,j+4);
    x15(:,j)=xd(:,j+5);
    yx2(:,j)=atanh(xd(:,j+1)+(xd(:,j)-xd(:,j+1))/h);
  end

  a1=yx1*x11'*inv(x11*x11');

  for j=1:len
    yy1(:,j)=xd(:,j)-a1*xd(:,j+1);
  end

  b1=yy1*x12'*inv(x12*x12');

  for j=1:len
    yy1(:,j)=xd(:,j)-a1*xd(:,j+1)-b1*xd(:,j+2);
  end

  c1=yy1*x13'*inv(x13*x13');

  for j=1:len
    yy1(:,j)=xd(:,j)-a1*xd(:,j+1)-b1*xd(:,j+2)-c1*xd(:,j+3);
  end

  d1=yy1*x14'*inv(x14*x14');

  for j=1:len
    yy1(:,j)=xd(:,j)-a1*xd(:,j+1)-b1*xd(:,j+2)-c1*xd(:,j+3)-d1*xd(:,j+4);
  end

  e1=yy1*x15'*inv(x15*x15');

end % Revise all variables



if var==5
  y1=a1*xd(:,1)+b1*xd(:,2)+c1*xd(:,3)+d1*xd(:,4)+e1*xd(:,5);
  madd1=a1*y1+b1*xd(:,1)+c1*xd(:,2)+d1*xd(:,3)+e1*xd(:,4);
end

if var==4
  y1=a1*xd(:,1)+b1*xd(:,2)+c1*xd(:,3)+d1*xd(:,4);
  madd1=a1*y1+b1*xd(:,1)+c1*xd(:,2)+d1*xd(:,3);
end

if var==3
  y1=a1*xd(:,1)+b1*xd(:,2)+c1*xd(:,3);
  madd1=a1*y1+b1*xd(:,1)+c1*xd(:,2);
end
```

```
  if var==2
    y1=a1*xd(:,1)+b1*xd(:,2);
    madd1=a1*y1+b1*xd(:,1);
  end

  if var==1
    y1=a1*xd(:,1);
    madd1=a1*y1;
  end

  pre1(:,k)=y1;
  pre2(:,k)=xd(:,1)+2*h*(-y1+tanh(madd1));
  p1=pred(:)-pre1(:,k);
  p2=optd(:)-pre2(:,k);
  MAE=MAE+sqrt([diag(p1*p1')';diag(p2*p2')']);

  for i=1:cur
    pp1=(pre1(i,k)-neg)*de(i)/(2*pos)+nu(i);
    pp2=(pre2(i,k)-neg)*de(i)/(2*pos)+nu(i);
    ppp1=predr(i)-(pp1+addraw(i));
    ppp2=optdr(i)-(pp2+predr(i));
    if ppp1<0, MAPE(1,i)=MAPE(1,i)-ppp1/predr(i); else
               MAPE(1,i)=MAPE(1,i)+ppp1/predr(i); end;
    if ppp2<0, MAPE(2,i)=MAPE(2,i)-ppp2/optdr(i); else
               MAPE(2,i)=MAPE(2,i)+ppp2/optdr(i); end;
  end

  for i=1:cur
    if (pre1(i,k)>=0 & pred(i)>=0) | (pre1(i,k)<0 & pred(i)<0),
      updown(1,i)=updown(1,i)+1; end
    if (pre2(i,k)>=0 & optd(i)>=0) | (pre2(i,k)<0 & optd(i)<0),
      updown(2,i)=updown(2,i)+1; end
    if pre1(i,k)<-hold & pre2(i,k)>hold, pbhs=1; else pbhs=2; end
    if pre1(i,k)>hold & pre2(i,k)<-hold, pbhs=3; else pbhs=2; end

    % buy=1 hold=2 sell=3
    if pred(i)<-hold & optd(i)>hold, bhs=1; else bhs=2; end
    if pred(i)>hold & optd(i)<-hold, bhs=3; else bhs=2; end
    if bhs==pbhs, buysell=buysell+1; end
  end

end   % Call another dataset

MAE=(1/fore)*MAE

MAE_total=zeros(2,1);
for i=1:cur,  MAE_total=MAE_total+MAE(:,i);  end
MAE_average=MAE_total/fore

MAPE_percent=MAPE*100/fore
MAPE_average=zeros(2,1);
for i=1:cur,  MAPE_average=MAPE_average+MAPE_percent(:,i);  end
MAPE_average

updown_percent=updown/fore

total=zeros(2,1);
for i=1:cur,  total=total+updown(:,i);  end
updown_average_percent=100*total/(4*fore)

buysell_percent=100*buysell/(4*fore)
```