

Deploying hybrid methodology in enhancing large file transfer

Ng Kar Hoong, Poo Kuan Hoong, Wong Chee Onn

Faculty of Information Technology, Multimedia University, Cyberjaya 63100, Selangor, Malaysia

ng.kar.hoong10@mmu.edu.my, khpoo@mmu.edu.my, cowong@mmu.edu.my

Abstract— File transfer is important for interconnected network systems. File synchronization is a process that uses file transfer method to update files between distanced locations. For an organization with regional distanced companies, file synchronization plays an important role in maintaining data concurrency between the companies. Due to the demand of faster file transfer methods, many related research have been done to improve the performance of file transfer. One of the approaches is to identify the network traffic condition and perform file transfer when the network traffic is relatively low. Time series forecasting exhibits the possibility for predicting the network traffic condition in the future events accurately. However, there are a few issues that needed to be resolved in order to utilize forecasting in file transfer. The objective of this paper is to evaluate the total improvement of a proposed file transfer method built on a forecasting module. This paper consists of three parts. Part one is time series model determination. This consists of time series model parameter estimation method by using Box-Jenkins methodology. Part two is to deploy and evaluate the performance by implementing file chunking with forecasting method before initializing file transfer. This is to determine the performance of file chunking algorithm in assisting time series forecasting during file transfer. Part three is similar process in part two with slight changes in the file transfer order. In the end of the experiment, we will conclude the usability of the proposed method based on the experiment conducted for this paper.

Keywords— Time series forecasting, file transfer, file chunking

I. INTRODUCTION

In an environment such as international organizations, fast data transfer plays an important role in keeping the data concurrent. File synchronization method such as RSYNC have been used to increase the performance of servers in updating data. Double checksum of RSYNC also decrease the chance of data corruption when transfer file over the network [1]. Several researches revolve around the file synchronization algorithm. Enhancement by decreasing the overheads due to the folder hierarchy is one of the attempts to improve the method. However, this method requires directory monitoring which might add extra loads to the system machines [2]. Ghobadi et. al. proposed intelligent file chunking method that split large files into smaller files before initialize file transfer with RSYNC. The authors used Artificial Neural Network (ANN) to determine the suitable file size for a specific file size before sending it over the network [5]. However, the

methods proposed do not include the network traffic information. So, by obtaining network traffic condition and sent at the suitable period of time can reduce the file transfer time.

Time series forecasting can provide the future network traffic information to the user and reduce the overall file transfer time [3][4]. Even though we can utilize the forecasted network traffic to predict the period with low traffic, the network itself is not necessary to be low at all time. Problems also arise when initializing large file transfer that utilizing time series forecasting. Large file requires longer time to complete its data transfer, which might cause the forecasted data not suitable to be used [4].

In this paper, we will implement the same statistical time series model approximation method as described by Ng et. al. [4] and compare another approach in model estimation method. File chunking method will be deployed in this experiment to observe the performance of combining both methodologies.

The early hypothesis is that the hybrid methodology can improve the overall file transfer time for a large file.

In the next section, we will cover tools and methodology involved in section II, experiment setup in section III. While the result and analysis will be discussed in section IV and finally conclude the paper in section V.

II. TOOLS AND METHODOLOGY

In this experiment, we use the following tools; Cronos library for time series forecasting application, libpcap script for collecting network traffic data and RSYNC to initialize file transfer.

A. Cronos – Time series analysis

Cronos is an open source graphical time series analysis application. It consists of many tools that allow data manipulation and support several time series forecasting models including Autoregressive Integrated Moving Average (ARIMA) that used in this publication.

ARIMA model is the generalized form of the Autoregressive Moving Average (ARMA). ARIMA model can be explained as a combination of three parts; the Autoregressive (AR) which represents the differenced historical values; the Moving Average (MA) which represents the error component; and the “integrated” part which

represents the non-stationary time series which needs to be differenced.

A basic non-seasonal ARIMA model can be written as “ARIMA (p,d,q)”, where the parameters “p” is the number of autoregressive terms, “d” is the number of the difference terms, and “q” is the number of the moving average terms. In this experiment, the model used contained no difference term. The general equation of ARIMA (p,d,q) model is as follows [3][4]:

$$Y_t = \theta_1 y_{t-1} + \theta_2 y_{t-2} \dots + \theta_p y_{t-p} + \varphi_t - \varphi_1 \alpha_{t-1} \dots - \varphi_q \alpha_{t-q} \quad (1)$$

where,

y represents the time series
 Y_t represents the forecast of time series for the time t .
 $y_{t-1} \dots y_{t-p}$ represent the AR terms.
 $\alpha_{t-1} \dots \alpha_{t-q}$ represents the MA terms
 $\theta_1 \dots \theta_p$ represents the coefficients.
 $\varphi_1 \dots \varphi_q$ represents the coefficients

B. Libpcap

Libpcap is an open source network packet capture library that is written in C. Libpcap provides some basic filtering mechanism that can serve our purpose of data collection efficiently. Several network analysis application such as wireshark[6] and tcpdump [5] use this library as their base. This tool is useful in many network related fields such as intrusion detection, learning network protocols, troubleshooting network problems, network bandwidth monitoring and etc.

In this research, libpcap is set to capture TCP and UDP packets only and save to an external file every time interval. As an example, the time interval is set to every 30 seconds. We captured three hour of network data to get 360 points of data. The first two hours serve as training data for the time series model, and the remaining one hour as testing data to compare the accuracy of the forecasted data [4].

C. File chunking

File chunking method is used in two phases of the experiments. File chunking is implemented by using the Linux command “split” and then merged back by using the command “cat”.

The logic behind the file chunking module is assisted by using simple Neural-Network (NN)[10] which was trained beforehand.

D. Methodology

Figure 1 shows the process flow of forecasting in this experiment. The forecast target is the network traffic condition that is represented by the network packets. The first stage in Figure 1, **Collect network traffic data**, a libpcap script is used in capturing the number of packets in the every 30 seconds interval.

In the second stage, **Transformation** is applied to normalize the data by using log return equation as written in (2).

$$R(t) = \ln(NP_t) - \ln(NP_{t-1}) \quad (2)$$

where,

NP_t = number of packets captured at time t

$R(t)$ = log return value at time t

In the third stage, all the normalized time series data (log return) are fitted into the ARIMA model. This will proceed to the final stage where by using the data, we can forecast five observations ahead of the historical data. This process flow was also used in the publication [4]. Figure 2 describes the terms step and step size used in the experiment.

III. EXPERIMENT SETUP

In this section, we conduct three phases of experiments.

In phase I, we perform evaluation of forecasting method which includes forecasting model parameters determination. This is guided by the Box-Jenkins methodology [8][9] and cross-checked by using log likelihood [4]. Autocorrelation function (ACF) and Partial Autocorrelation function (PACF) are used in determining the value of p and q in the model ARIMA (p,d,q) while log likelihood task is to compare the models obtained from the ACF and PACF.

While in phase II, we used the method in Phase I with file chunking method, and initialize file transfer for a file of 100 Megabyte (MB) size. Finally in phase III, we repeat the processes in phase I and II with larger file size.

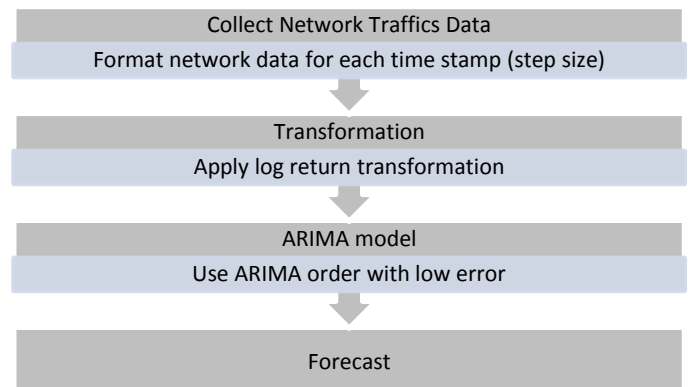


Figure 1. Forecasting process flow

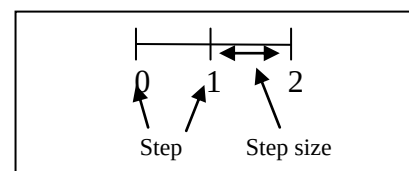


Figure 2. Step and step size

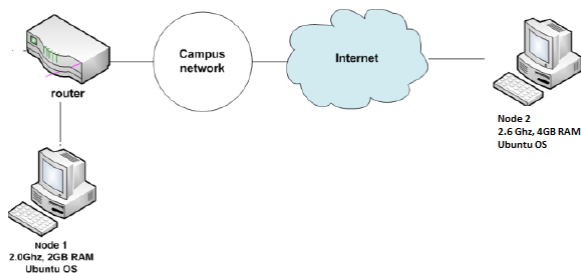


Figure 3. Experimental Setup

TABLE 1. AVERAGE MEAN SQUARED ERROR FOR PHASE I

Step size	Model Order	Average MSE	MSE Step 1
30	ARIMA (2,0,1)	619.9365649	571.581924
60	ARIMA (2,0,0)	1885.303779	2826.232816
90	ARIMA (3,0,3)	1838.354806	6047.412559
120	ARIMA (1,0,1)	2752.319867	2683.973323

A. Phase I: ARIMA model parameter estimation.

The environment setup for the experiment is shown in the Figure 3. The experiment in phase I is conducted in a controlled environment within the campus ground, which is built on Megabit network infrastructure (Node 1). Node 2 is connected at home environment with limited bandwidth (1.2 M).

The network traffic is captured on the machine for total of three hours. Two hours out of the three hours of captured network packets, is used in training of the time series forecasting model. The captured data is formatted to various step sizes to get different variation of input data to the forecasting module. The order of the model is determined from the ACF and PACF graph [7]. Log likelihood can be used in model parameter selection if the ACF and PACF show more than one forecasting model is suitable for the network traffic.

B. Phase II: File transfer

Using the method in phase I, we obtain suitable model order for the network traffic. In this phase, we compare two types of file transfer methods on a 300MB file. One; is assisted by forecasting while two; is assisted by both forecasting and file chunking. The file will be transferred from Node 1 to Node 2 as illustrated in Figure 3.

The file chunking module is guided by trained Neural-Network and triggered by the forecasting module to perform chunking on the targeted file. In training the Neural-Network, input data such as number of packets, processor utilization, bandwidth, file size are included. The task of the Neural-Network is to inform the file chunking module the suitable file size at given input parameters.

Explanation on the steps used in this phase is described in Figure 4.

C. Phase III: Large file scheduling

Repeat the method in phase II, we send larger file size to see the total performance increment in file transfer. The initial expectation for this phase is the proposed algorithm shows reduction in sending large file regarding the overheads that is caused by the file chunking method. Another approach of sending the file will also be tested.

The approach is related to scheduling of the chunks after performing file chunking on a large file.

IV. RESULT AND ANALYSIS

A. Phase I : ARIMA model estimation

The objective of phase one is to estimating time series forecasting model parameter that is suitable for the network traffic in the setup arranged for this research. The justification for the accuracy of forecasting model is determined by the log likelihood function. Suitable parameter can maximized the log likelihood for the appointed step size.

The experiment for this phase consists of test cases with variety of step sizes as in Table 1. The two hours of training data is formatted accordingly to the step sizes, affects the structure of the data itself. For each step size, we forecasted five steps ahead of the historical data by using the method similar to [7].

The suitable model selection for the each step sizes, are justified by the log likelihood value. While for selection of the best model and step size, we choose by using the Mean Squared Error (MSE) as in equation (3).

$$MSE(x) = \frac{1}{n} \sum (Actual_i - Predicted_i)^2 \quad (3)$$

where,

x = step 1, step 2,..., step 5

$i = i^{th}$ observation for step x

n = number of observations for step x

The accuracy of the forecasted steps is measured by using MSE. The average MSE is calculated for all the forecasted steps for comparison purpose. Table 1 also shows the average MSE for each step size which using the unit number of packets. The combination of step size and model with smallest MSE is categorized as the most accurate combination. MSE value for step 1 is shown in table because the step is the point next to the historical data. Forecasting capability regresses due forecasting many steps ahead which make the network traffic inaccurate.

The MSE values in Table 1 are relative large due to the large number of packets in the network. The highest number of packets per step size that we can observe from the data can reach up to 20,000 packets. From Table 1, we can see model ARIMA (2,0,1) for step size 30 seconds obtained the lowest MSE value.

1. Libpcap will collect network data on node 1 and node 2.
2. After two hours,
 - a) Five step ahead will be forecasting using the ARIMA (2,0,1). Libpcap will continuously capture network traffic.
 - b) Initiate file transfer.
 - i. If forecasted network traffic indicates high network traffic, perform file chunking and send
 - ii. Else, send the file as it is
 - c) Wait till the end of file transfer
 - d) Continue to the step a).
3. Compare the forecasted network traffic with the actual network traffic and perform evaluation and analysis of the file transfer.

Figure 4. Procedure for file transfer

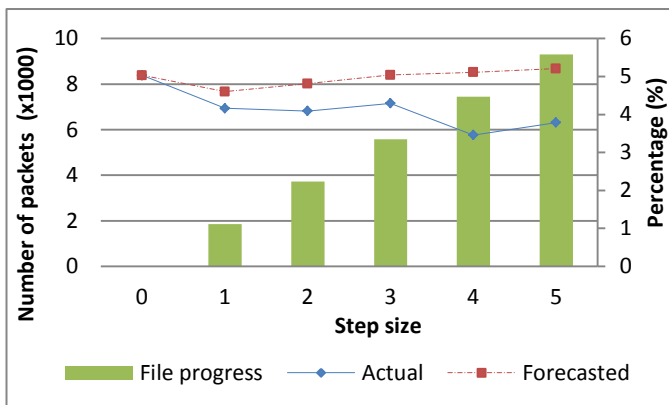


Figure 5. ARIMA (2,0,1) Forecasted low network traffic with the step size of 30 seconds.

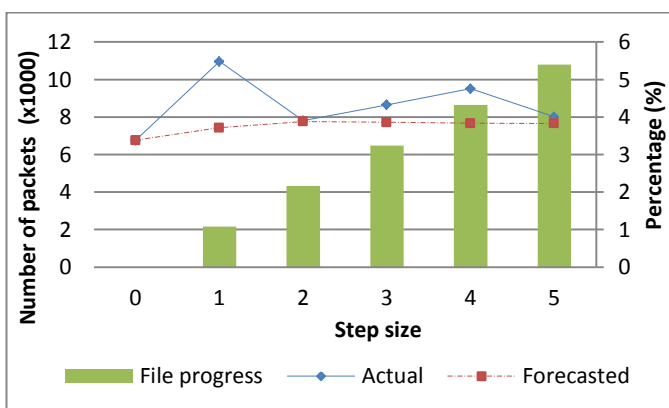


Figure 6. ARIMA (2,0,1) Forecasted high network traffic with the step size of 30 seconds.

B. Phase II : File Transfer

For phase II, we perform file transfers based on two test cases: i) we send file based on the forecasting method only ii) we send file based on the forecasting method and assisted by file chunking. Figure 4 shows the procedure for this phase. Both test cases are using the similar procedure except for the file chunking routine. The target file is an archived (.zip) file of the size of 300MB. The forecasted network traffic is categorized based on the first step (step 1) from the forecasted five steps ahead. Lower number of packets from the step 0 will indicate the next network traffic is low and vice-versa.

In the first test case, we obtain the network traffic information based on the forecasting alone. This test case is mentioned in the publication [4] where we performed file transfer on two occasions: a) when the forecasted network traffic is low b) when the forecasted network traffic is high. The result of the experiment is projected in Figure 5 and 6. Figure 5 forecasted low network traffic at step 1 while Figure 6 forecasted high. The file transfer in Figure 6 shows faster rate at step 1 is due to the system allowing more packets go through the network as shown in the graph. However, the speed went down for Figure 6 and went up for Figure 5 which resulted in a faster file transfer completion as shown in Table 2.

In the second test case, we utilize file chunking in response to the feedback of the forecasting module. File chunking is triggered when the forecasting module forecasted the step 1 is high or congested. The file chunking will separate the large file into smaller files and, then send the first chunk of the separated file to the destination. The result of the experiment for this test case is illustrated in Figure 7 and 8. Even though both graphs show different forecasted results, they show similar speed at the beginning of the file transfer. This is reflected at the file progress at step 1 in Figure 7 and 8. However, the file transfer for Figure 7 completed faster than Figure 8. This is expected from the forecasted network traffic.

Table 2 shows the file transfer time for the test cases mentioned. The table reflects the effect of the forecasting and file chunking towards file transfer time. From the table, we can see for difference in performance for both test cases during the high network traffic. The enhancement from file chunking module is not projected in the experiment. This is due to the overheads generated from the file chunking module and the scheduling method to handle the chunks.

TABLE 2. FILE TRANSFER COMPLETION TIME

Network traffic	Forecasting only	Forecasting and file chunking
	File transfer time (seconds)	File transfer time (seconds)
Low	2877 (Fig. 5)	2968 (Fig. 7)
High	3152 (Fig. 6)	3253 (Fig. 8)

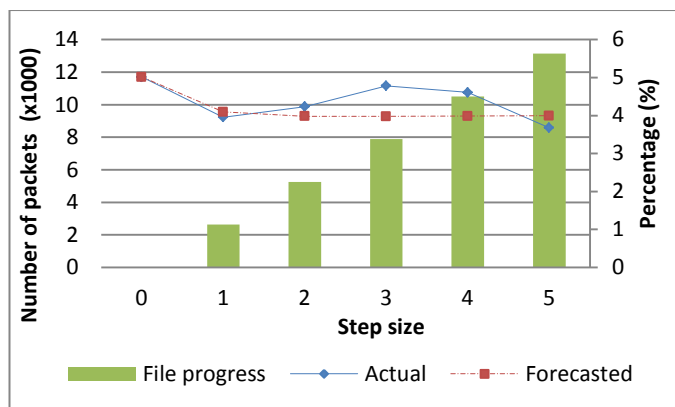


Figure 7. ARIMA (2,0,1) Forecasted low network traffic with the step size of 30 seconds assisted by file chunking

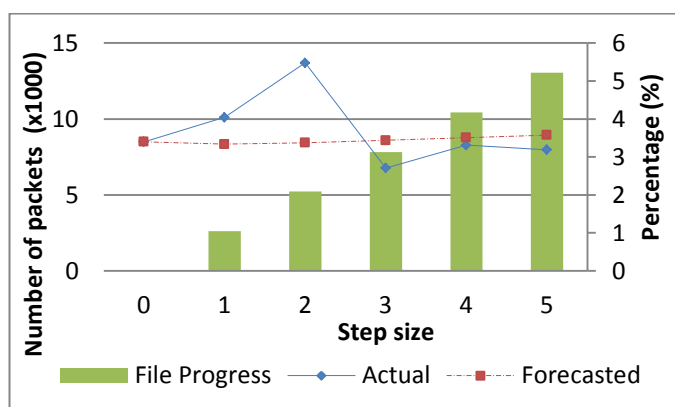


Figure 8. ARIMA (2,0,1) Forecasted high network traffic with the step size of 30 seconds assisted by file chunking

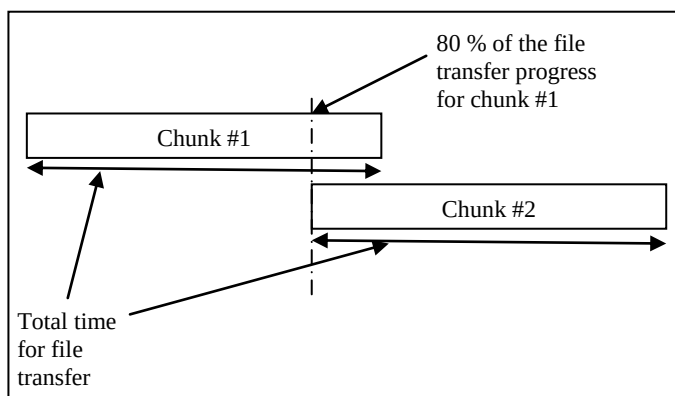


Figure 9. File transfer scheduling for chunks

C. Phase III : Large file scheduling

In this paper, we initiate file transfer with the assist from the forecasting and file chunking module. From our experiments, we determined the suitable time series model is ARIMA (2,0,1) when using network data of step size 30 seconds. However, the performance of the file chunking did not performed as expected in Phase II.

So, we attempt to change the scheduling method of the file transfer and repeat the file transfer for the same file size. The approach is to initiate another file transfer when the initial

data transfer reaching completion. The objective of the method is to reduce the overhead time that caused by the file chunking. This is a method imitating parallel file transfer except that the remaining files are not transferred at the same time with the first file. The process is illustrated at Figure 9.

Assume a large file is separated into two parts or chunks. We initiate a file transfer for the chunk #1. The time for completion for the chunk #1 is 300 seconds which is an average time that can be get from repetitive file transfer of the similar size. When the file transfer reaches 80 percent of the average time which is 240 seconds, we initiate the file transfer for the second chunk (chunk #2).

Figure 10 projecting the forecasted network traffic for the file transfer that utilizing file chunking with the scheduling method shown in Figure 9. The total time taken for the method is 2874 seconds. The method chunk the 300MB file into two chunks. The improvement of the method is not significant, merely 8% to 9%.

This might due to the restriction of the system administrator on the campus network as the number of user increases when reaching busy working hour which also restricting the number of packets as we see in Figure 10.

V. CONCLUSION

In this paper, we initiate file transfer with the assist from the forecasting and file chunking module. From our experiments, we found the suitable time series model is ARIMA (2,0,1) when using on network data of step size 30 seconds. The model mentioned is capable of forecasting network traffic by using short-term training data (2 hours).

From our results, we have shown that file chunking capable to reduce the time for file transfer under certain constrains. File chunking can be enhanced in the future to be more efficient in the future by implement "self-learning" Neural-Network that can re-train itself with the change of environment.

The alternate scheduling method shows some room for improvement where we can try to initiate file transfer of the remaining chunks at earlier period of time and not just at the 80 percent of the average completion time.

The conclusion for this research is that, forecasting methods can be further enhanced by incorporating file chunking method into it. However, more adjustment must be done to fix the existing problems that were shown in this publication.

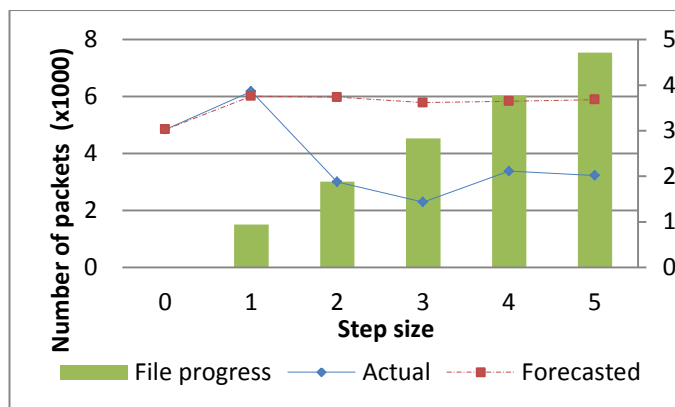


Figure 10. ARIMA (2,0,1) Forecasted high network traffic with the step size of 30 seconds assisted by file chunking using alternative scheduling method.

REFERENCES

- [1] Rsync, <http://rsync.samba.org/> (last accessed 1st November 2011)
- [2] Ghobadi, A. Mahdizadeh, E.H. Yong Lee Kee, Li Kok Wei, Ghods, M.H. , "Pre-processing directory structure for improved RSYNC transfer performance", ICACT 2011 South Korea, Feb 2011, pg. 1043 - 1048 .
- [3] Ian K.T. Tan, Poo Kuan Hoong, Chee Yik Keong, "Towards Forecasting Low Network Traffic for Software Patch Downloads: An ARMA model forecast using CRONOS", Proc. International Conference on Computer and Network Technology (ICCNT 2010), Bangkok, Thailand, 2010, pp. 88 – 92.
- [4] Ng Kar Hoong; Poo Kuan Hoong; Tan, I.K.T.; Muthuvelu, N.; Lee Ching Seng; , "Impact of utilizing forecasted network traffic for data transfers", ICACT 2011 South Korea, Feb 2011, pg. 1199 - 1204 .
- [5] TCPDUMP, http://www.tcpdump.org/tcpdump_man.html (last accessed 1st November 2011).
- [6] Wireshark , <http://www.wireshark.org/> (last accessed 1st November).
- [7] Forecasting , <http://www.duke.edu/~rnau/411home.htm> (last accessed 1st November)
- [8] Brockwell, Peter J. and Davis, Richard A., "Introduction to Time Series and Forecasting (2nd Edition)", Springer-Verlag, 2002, pp. 151- 192.
- [9] Box, George E. P. and Gwilym M. Jenkins, "Time Series Analysis: Forecasting and Control, revised ed.", Holden-Day, Oakland, 1976.
- [10] B. Krose, and P. Van Der Smagt, "An Introduction to Neural Network", pp. 33, 1996.
- [11] Sowell, Fallaw, "Maximum likelihood estimation of stationary univariate fractionally integrated time series models", Journal of Econometrics , July-September 1992, pp. 168-188.