Forecasting Highly-Aggregate Internet Time Series using Wavelet Techniques

BY

Samuel Z. Edwards, LT, USCG

B.S.E.E., U.S. Coast Guard Academy at New London, CT, 2000

THESIS

Submitted to the Faculty of the

Virginia Polytechnic Institute & State University

in partial fulfillment of the requirements for the degree of

Master of Science

In

Electrical Engineering

Dr. Lamine M. Mili, Committee Chair

Dr. Luiz A. Da Silva, Committee Member

Dr. Amy E. Bell, Committee Member

15 May, 2006

Arlington, Virginia

Key Words: Short-range Dependence, Long-range Dependence, Wavelets, Forecast, Time Series Analysis, Fractals, Hurst parameter, Self-Similarity

Forecasting Highly-Aggregate Internet Time Series using Wavelet Techniques

Samuel Z. Edwards, LT, USCG

Abstract

The U.S. Coast Guard maintains a network structure to connect its nation-wide assets. This paper analyzes and models four highly aggregate traces of the traffic to/from the Coast Guard Data Network ship-shore nodes, so that the models may be used to predict future system demand. These internet traces (polled at 5'40" intervals) are shown to adhere to a Gaussian distribution upon detrending, which imposes limits to the exponential distribution of higher time-resolution traces. Wavelet estimation of the Hurst-parameter is shown to outperform estimation by another common method (Sample-Variances). The First Differences method of detrending proved problematic to this analysis and is shown to decorrelate AR(1) processes where $0.65<\phi_1<1.35$ and correlate AR(1) processes with $\phi_1<-0.25$. The Hannan-Rissanen method for estimating ($\hat{\phi}, \hat{\theta}$) is employed to analyze this series and a one-step ahead forecast is generated.

Acknowledgments

I would like to begin by expressing my deep gratitude to my Thesis Advisor, Dr. Mili, for his guidance. My friends advised me to seek a thesis advisor who would build an atmosphere of trust and respect and enjoyment. Dr. Mili's guidance was critical, but his love of learning was especially nourishing. I loved gathering for weekly presentations with his other research students. These all-afternoon sessions gave each of us opportunity to develop our presentation skills and to learn from each other.

Research on the CGDN+ would be impossible without data to analyze. Tom Clark, who is Chief of the Telecommunications Technology Staff at the Coast Guard's Telecommunication and Information Systems Command (TISCOM), provided me with the data and advice that I needed to accomplish this research. Also, he altered the Coast Guard's polling method to provide a constant interval series that can be forecasted and analyzed.

Dr. Amy Bell taught me many of the basic principles of Wavelet Theory, which made the current literature accessible. I am likewise grateful to Dr. DaSilva for discussing my findings throughout this research, and for his practical advise on Network Management.

Many friends and mentors at Virginia Tech, Coast Guard, and at home have provided me the support that's been essential to completing this research. The Coast Guard has provided to me this opportunity for advanced education. I will be physically indebted to the Coast Guard for four years after graduation, but my heart is forever grateful for this opportunity that the Coast Guard and the American taxpayers have provided to me.

My wife, Peggy, has been my constant support and friend in work and education. Life would be incomplete without you, and eternity purposeless without your presence.

Dedicated to Esra Grace Edwards

Table of Contents

ABSTRACT ..... **II**

ACKNOWLEDGMENTS ..... **III**

TABLE OF CONTENTS ..... **V**

LIST OF FIGURES AND TABLES ..... **VI**

1 INTRODUCTION **1**

PRESENTING THE COAST GUARD DATA NETWORK 1
OVERVIEW OF THIS WORK AND ITS CONTRIBUTIONS 4

2 WAVELET TRANSFORMS APPLIED TO TIME SERIES ANALYSIS **7**

FUNCTION SPACES AND WAVELET TRANSFORMS 7
MATHEMATICAL PROPERTIES OF THE WAVELET TRANSFORM 12

3 ASSUMPTIONS OF LINEARITY AND STATIONARITY **19**

GAUSSIANITY EMPLOYED IN HIGHLY-AGGREGATED INTERNET TRACES 19
CGDN+ SERIES EXPERIENCE A STRONG DAILY CYCLE 21
Daily Cycles *25*
Filtering the 24 hour cycle *28*
Beware the First Difference Operation! *30*

4 MODELING LRD DATA **35**

SELF-SIMILARITY IN THE CGDN+ DATA 38

5 ONE-STEP AHEAD FORECASTS OF BYTEOUT **42**

6 CONCLUSIONS AND FUTURE RESEARCH WORK **47**

REFERENCES **49**

APPENDIX A: A DISCUSSION OF THE MOTIVATION FOR FORECASTING **51**

APPENDIX B: COMPARING CGDN+ TO THE BELLCORE TRACES **53**

APPENDIX C: PRIMER ON LINEAR STATIONARY MODELS **60**

FREQUENCIES AND THE Z-DOMAIN 62
GENERAL FORMS OF THE LINEAR STATIONARY MODELS 65
IDENTIFYING THE DEPENDENCIES IN LINEAR STATIONARY MODELS 68

APPENDIX D: MATLAB CODE USED DURING RESEARCH **71**

COMPARISON 71
AUTOCORR 72
QQPLOTS_DETREND 73
ZFUNCTION 74
DAILY_TREND 74
DETREND_DAILY 75
CREATE_SYNTHETIC_TRAFFIC 76
PARTCORR 77
HURST_LOGVAR 78
DISCARD 79
HURST_WAVELET 79
POWER SPECTRAL DENSITY 80
X_TILDA 81
ARMA_TWOSTEP 82

VITA **83**

List of Figures and Tables

FIGURE 1-1: ROUTING SCHEME INVOLVED WITH CGDN+ SHIP/SHORE CONNECTION 2
FIGURE 1-2: SUMMARIES OF NETWORK TRAFFIC TO/FROM CGC FORWARD BETWEEN 1-11 JULY 2005. (A) PACKET ARRIVALS, (B) BYTE ARRIVALS, (C) PACKET TRANSFERS, AND (D) BYTES TRANSFERRED IN EACH POLLING INTERVAL 3
FIGURE 1-3: LAG ONE PLOTS OF ILLUSTRATING (LEFT) BYTE ARRIVALS TO CGC FORWARD, AND (RIGHT) AN UNCORRELATED SERIES. 4
FIGURE 1-4: AUTOCORRELATION FUNCTION OF USCGC FORWARD BYTE ARRIVALS. 5
FIGURE 2-1: NESTED SUBSPACES, V_J, AND THEIR ORTHOGONAL COMPLEMENTS, W_I 7
FIGURE 2-2: TWO BASES FOR THE TWO-DIMENSIONAL COORDINATE PLANE: (LEFT) STANDARD BASIS, AND (RIGHT) HAAR BASIS. 9
FIGURE 2-3: FILTER BANK DETAILING THE DECOMPOSITION OF A SIGNAL BY SUCCESSIVE FILTERING 10
FIGURE 2-4: DISCRETE WAVELET TRANSFORM (DWT) OF BYTEIN SERIES. TRANSFORM CALCULATED USING MATLAB'S WAVELET TOOLBOX. SCALING IS ARRANGED IN DESCENDING ORDER FROM LOWEST TO HIGHEST FREQUENCIES. 14
TABLE 2-5: HURST ESTIMATES OF SUCCESSIVELY SHORTER SEQUENCES AS MEASURED BY WAVELET AND LOG-VARIANCE METHODS. 18
FIGURE 3-1: (LEFT) HISTOGRAM BYTEOUT AFTER DETRENDING APPEARS GAUSSIAN DISTRIBUTED. (LEFT) Q-Q PLOTS SHOW MORE CONCLUSIVE EVIDENCE THAT DETRENDED BYTEOUT IS GAUSSIAN DISTRIBUTED. (LEFT-TOP) ORDERED BYTEOUT BROKEN INTO 100 QUANTILES. (BOTTOM) QUANTILES EXTRACTED FROM 23 SERIES OF BYTEOUT, THEN QUANTILES ARE AVERAGED. 19
FIGURE 3-2: AVERAGE PACKET ARRIVALS OVER 25 DAYS. 21
FIGURE 3-3: AUTO-CORRELATION FUNCTIONS FOR CGC FORWARD DATA SERIES BETWEEN 1-11 JULY 2005. (LEFT) PACKET ARRIVALS VS. BYTE ARRIVALS, (RIGHT) PACKET TRANSFERS VS. BYTE TRANSFERS. 22
FIGURE 3-4: POWER SPECTRAL DENSITIES (PSD) OF BYTEOUT SERIES DEMONSTRATING ITS DAILY PERIODICITY. (A) PSD OF UNWINDOWED BYTEOUT SERIES, (B) PSD OF THE AUTOCORRELATION FUNCTION OF UNWINDOWED BYTEOUT, (C) PSD OF BYTEOUT AFTER HANNING WINDOWED, AND (D) PSD OF AUTOCORRELATION FUNCTION OF BYTEOUT AFTER HANNING WINDOW IS APPLIED. 23
FIGURE 3-5: AVERAGE DAILY TRENDS OF INTERNET DATA EXPERIENCED BY CGC FORWARD: (LEFT) BYTEOUT, (RIGHT) PACKOUT, AND (BOTTOM) PACKIN. 25
FIGURE 3-6: PARTIAL CORRELATION FUNCTIONS FOR DETRENDED SERIES (LEFT TO RIGHT, TOP TO BOTTOM): BYTEOUT, BYTEIN, PACKOUT AND PACKIN. 26
FIGURE 3-7: POLLING INTERVAL LENGTHS OF CGC FORWARD DURING 1-11 JULY 2005 27
TABLE 3-8: FILTER COEFFICIENTS FOR (LEFT) REMOVING AND (RIGHT) RESTORING 24 HOUR TREND. 28
FIGURE 3-9: PARTIAL CORRELATION FUNCTIONS FOR FILTERED SERIES (LEFT TO RIGHT, TOP TO BOTTOM): BYTEOUT, BYTEIN, PACKOUT AND PACKIN. 29
FIGURE 3-10: FIRST DIFFERENCE IN BYTE ARRIVALS FROM CGC FORWARD (1-11 JULY 2005) 30
FIGURE 3-11: DECORRELATING INFLUENCE OF THE DIFFERENCING OPERATION. 31
FIGURE 3-12: PARTIAL CORRELATIONS OF AN AR(1) MODEL (WITH ϕ_1=-0.8) AFTER FIRST DIFFERENCING. RESULTS PLAINLY SHOW THE STRONG CORRELATIONS INTRODUCED TO THE SERIES BY F.D. OPERATION. 32
FIGURE 4-1: HURST VALUE CALCULATED FOR DETRENDED BYTEOUT SERIES BY: (LEFT) LOG-VARIANCE METHOD, AND (RIGHT) WAVELET METHOD. 38
FIGURE 4-2: HURST CALCULATIONS OF SYNTHETIC TRAFFIC SEQUENCE BY (RIGHT) LOG-VARIANCE METHOD AND (LEFT) WAVELET METHOD. 39
TABLE 4-3: H-PARAMETER ESTIMATES FOR CGDN+ DATA. 40
FIGURE 4-4: VARIANCES BY SCALE OF CGC FORWARD USING SEVERAL WAVELET BASES. 41
TABLE 5-1: PARAMETER ESTIMATES OF BYTEOUT 42

FIGURE 5-2: AR(50) REGRESSIONS OF DIFF_BYTEOUT WHERE $\tilde{X}_n$ IS CREATED USING (RIGHT) SAMPLE VARIANCE ESTIMATE, AND (LEFT) WAVELET METHOD. BOTTOM PLOTS ARE THE ERRORS OF EACH MODEL. 44
FIGURE 5-3: POLE-ZERO PLOTS OF ($\hat{\phi}, \hat{\theta}$) AS CALCULATED USING (LEFT) SAMPLE VARIANCE METHOD OR (RIGHT) WAVELET METHOD. 45
FIGURE 5-4: ONE-STEP AHEAD FORECASTS OF DIFF_BYTEOUT WHERE $\tilde{X}_n$ IS CREATED USING (RIGHT) LOG-VARIANCE METHOD, AND (LEFT) WAVELET METHOD. 45
TABLE 5-5: RESULTS OF ONE-STEP AHEAD FORECAST COMPARED USING MEAN-SQUARE ERROR. 46
FIGURE B-1: RELATIVE FREQUENCY OF PACKET SIZES (IN BYTES) CROSSING THE INMARSAT CONNECTION. COLLECTED BETWEEN 9 JUNE AND 16 JUNE 2005 BY NETVCR.CAMSLANT.CGDN.USCG.MIL. 54
FIGURE B-2: HIGH VARIANCE TRAFFIC FROM MANY POSSIBLE SOURCES CAUSES LRD TRAFFIC. 55
FIGURE B-3: SYNTHETIC TRAFFIC SEQUENCE CREATED BY 500 INDEPENDENT SOURCES, $W_M(N)$ 57
FIGURE B-4: COMPARISON OF (LEFT) 'BYTEOUT' SERIES, (CENTER) 'SYNTHETIC' SERIES AND (RIGHT) AN AR(1) SEQUENCE OVER THREE LEVELS OF AGGREGATION EACH. 58
FIGURE B-5: AUTOCORRELATION AND PARTIAL CORRELATION FUNCTIONS OF SYNTHETIC TRAFFIC SERIES. 59
FIGURE C-1: SCHEMATICS OF (A) MOVING AVERAGE (MA) MODEL, (B) AUTOREGRESSIVE (AR) MODEL, AND (C) MIXED AUTOREGRESSIVE-MOVING AVERAGE (ARMA) MODEL. 60
FIGURE C-2: (LEFT) SYNTHETIC AR(1) SERIES (ϕ_1=0.8), AND (RIGHT) AUTOCORRELATION FOR AR(1) SERIES. 62
FIGURE C-3: (LEFT) POLE-ZERO DIAGRAM OF AR(1) PROCESS FOR ϕ_1=0.8, AND (RIGHT) THE CORRESPONDING MAGNITUDE RESPONSE OF THIS AUTOREGRESSIVE FILTER. 64
FIGURE C-4: (LEFT) REALIZATION OF MA(1) PROCESS CREATED WITH θ_1=0.8, AND (RIGHT) POLE-ZERO DIAGRAM FOR THE SAME PROCESS. 67
FIGURE C-5: AUTOCORRELATION AND PARTIAL CORRELATION FUNCTIONS FOR BYTEOUT AND BYTEIN. 70

1 Introduction

Presenting the Coast Guard Data Network

The Coast Guard Data Network (CGDN+)[1] connects all Coast Guard computer resources through numerous connection lines, including T1, T3, modem and satellite links. Through CGDN+, Coast Guard assets (ships and land stations) are connected to each other via the Coast Guard intranet, the Internet, and the SIPRNET (a secret-level Coast Guard/DOD intranet). Coast Guard ships are called cutters.

Packet data from Coast Guard cutters arrives to/from land through a satellite constellation known as INMARSAT. INMARSAT is used for international maritime communications, and provides coverage over most of the earth's seas. Coast Guard cutters currently employ either 64 or 128 kilobit/second (kbps) connections through INMARSAT. This satellite internet connection to Coast Guard cutters is a recent and very welcome development; it allows Coast Guard administrative and logistics work to continue away from the dock and is a tool to ease the regular separation of families while the cutter is underway. The CGDN+ ship-shore connection handles Wide Area Network (WAN) traffic that consists of emails, internet communication and Coast Guard information, making it a heterogeneous batch of traffic.

[1] The Coast Guard Data Network is named 'CGDN+' because it is the latest version of an evolving network.

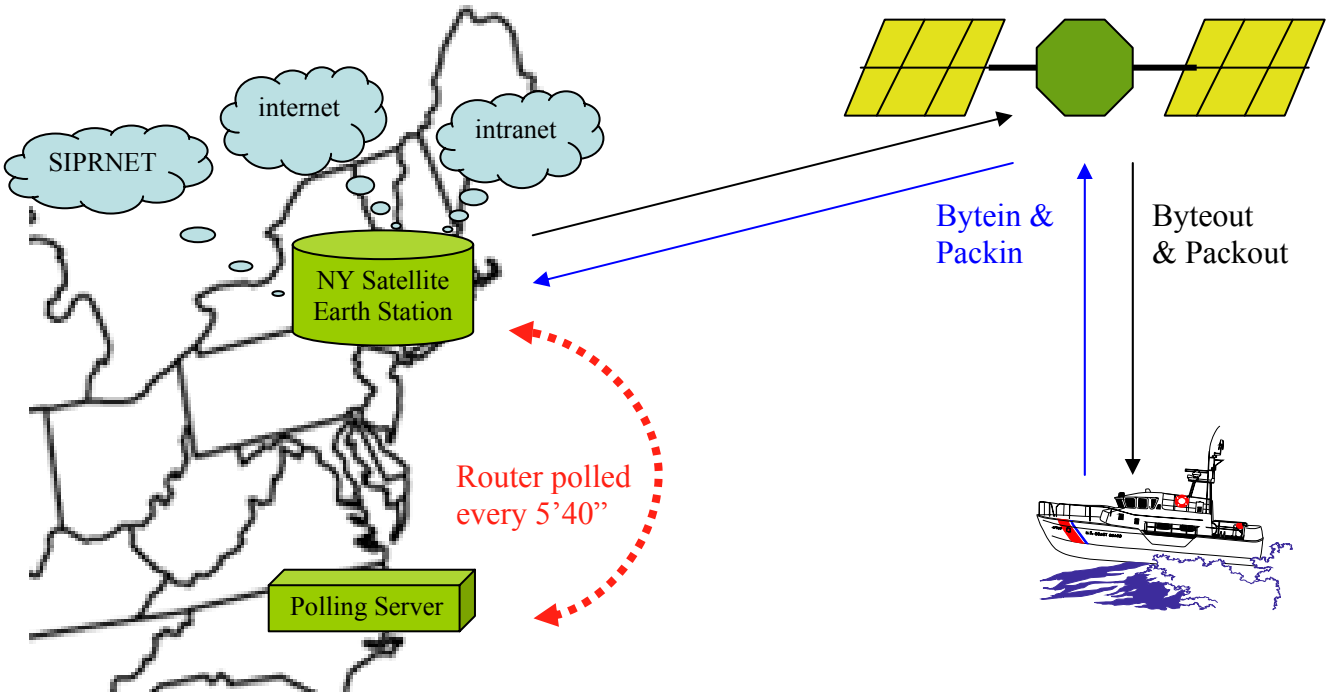


FIGURE 1-1: ROUTING SCHEME INVOLVED WITH CGDN+ SHIP/SHORE CONNECTION

A polling method has been devised to supervise the routers' proper operation. In this polling sequence, an independent server (located at Coast Guard Telecommunication and Information Systems Command at Alexandria, Virginia) requests operational statistics from the routers located at each Satellite Earth Station about once in every five minutes and forty seconds (5'40"). These statistics are an aggregate of the INMARSAT activity with each ship and include the number of packets (and bytes) received from a cutter or transmitted to the same cutter, as well as a few other statistics. The four internet traces analyzed during this paper are known as 'packin', 'packout', 'bytein' and 'byteout'. The direction ('in/out') has reference to the perspective of the router at the Satellite Earth Station, so that byte arrivals are called 'bytein' and represent the bytes transferred from the cutter to the Satellite Earth Station, etc.

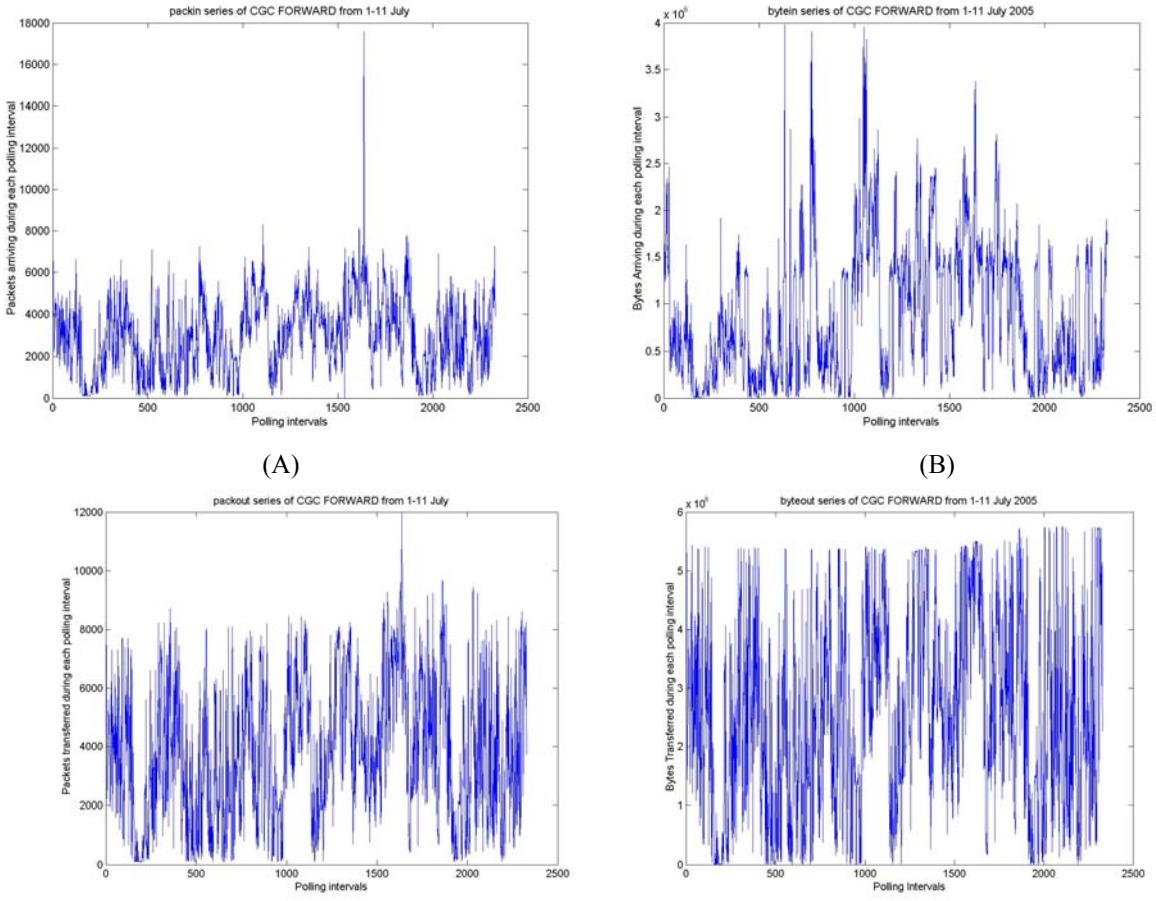


FIGURE 1-2: SUMMARIES OF NETWORK TRAFFIC TO/FROM CGC FORWARD BETWEEN 1-11 JULY 2005. (A) PACKET ARRIVALS, (B) BYTE ARRIVALS, (C) PACKET TRANSFERS, AND (D) BYTES TRANSFERRED IN EACH POLLING INTERVAL.

The threshold apparent in Fig 1-2(D) corresponds with the maximum number of bytes that can be transferred through a 128 kbps connection in 5'40". The congestion in this series suggests it as a candidate for further analysis. Although packet arrivals appear almost equal with packet transfers (compare Figures 1-2 (A) & (C)), many more total bytes are transferred from the Satellite Earth Station during each polling interval than are received. On average, packets transferred contain more than twice the number of bytes as packets arriving. It seems likely that much of the packet traffic arriving to Satellite Earth Station from CGC FORWARD is control traffic (such as ACKnowledgements for packets received).

Overview of this work and its contributions

This research establishes models for understanding and forecasting the CGDN+ data series. The results obtained may prove useful to network administrators of the CGDN+ and similar systems. The first contribution of this paper will be to analyze a highly-aggregate[2] internet series. The current literature prefers to dwell on the high time-resolution series like the Bellcore traces, which are useful for router optimization. [3] Aggregate series, like the CGDN+ series may provide helpful information for more general network administration.

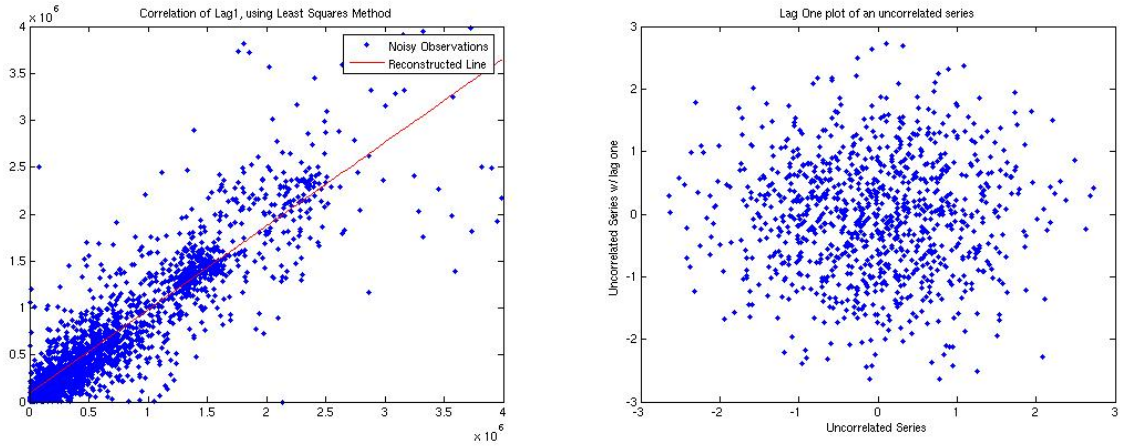


FIGURE 1-3: LAG ONE PLOTS OF ILLUSTRATING (LEFT) BYTE ARRIVALS TO CGC FORWARD, AND (RIGHT) AN UNCORRELATED SERIES.

The CGDN+ traces are realizations of a random process; the process is revealed by the interlag dependencies of its outputs. Figure 1-3 presents strong evidence that significant correlations exist in the CGDN+ traces, which make modeling and forecasting possible. [6], [7] and [19] present models that take advantage of the interlag dependencies existing in time series like the CGDN+. The simplest time series models are called Linear Stationary Models, meaning that they use a linear structure to produce stationary results. This paper shows that a linear

[2] "highly aggregate" refers to the fact that observations in the CGDN+ series summarize all internet activity that occurred during the preceding 5'40".
[3] Appendix B compares the CGDN+ traces with the famous Bellcore traces.

model is appropriate for modeling the CGDN+ series, but that the series are non-stationary and Long-Range Dependent and cannot be reproduced using a stationary model.

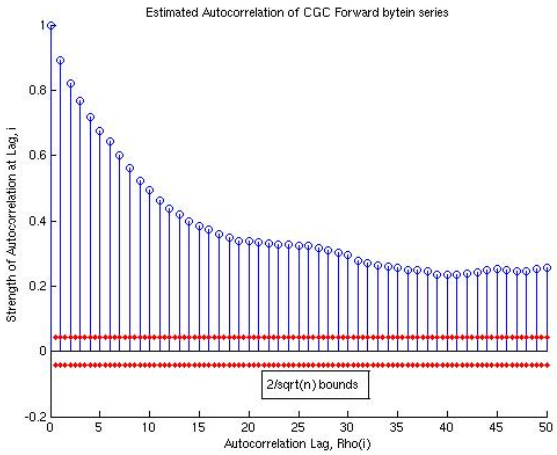


FIGURE 1-4: AUTOCORRELATION FUNCTION OF USCGC FORWARD BYTE ARRIVALS.

Slowly decreasing dependencies, like those in Figure 1-4, are termed Long-Range Dependencies (LRD). They indicate a self-similar system, and the infinite sum of these autocorrelations would be unbounded. The slowly decreasing dependencies in Figure 1-4 imply LRD, but other trends in the byte arrivals must be accounted and removed before that LRD can be quantified. The first step in this work is to appropriately detrend each series before modeling. Three methods of detrending are used on the data, and each produces similar results. Quite notably, the first difference method of detrending (as presented in [6] and [7]) removed all significant correlations from each of the series, as well as the trend. *Detrending by First Differences made the CGDN+ series Short-Range Dependent!* This finding came as a surprise and the second contribution of this work is to present the conditions under which the First Differences operation decorrelates its analyzed series.

Wavelet transforms are especially suited to working with self-similar (and thus LRD) processes since the wavelet basis function is defined so as to be scalable. The wavelet transformation can be plotted to display the time and frequency content of the analyzed signal

since wavelets have finite support. Additionally, each wavelet basis function possesses a number, N, of vanishing moments that remove polynomial trends up to order N-1 upon transformation. Reference [1] takes advantage of these vanishing moments to produce a wavelet method of the Hurst parameter that is unbiased to polynomial and linear trends. This research elaborates by showing that real, LRD series that have one significant partial correlation may be significantly correlated by the Haar wavelet.

The third contribution of this work is to employ the wavelet estimation method (presented in [1]) to produce more accurate forecasts of the CGDN+ traces. Estimates of the Hurst parameter from the wavelet method are compared with the results of the Sample Variances method (presented in [4]). Both estimates are used to transform the LRD byteout series into a Short-Range Dependent series that is evaluated for ($\hat{\phi}, \hat{\theta}$) using the Hannan-Rissanen Algorithm. [7]

Chapter Two begins this discussion with a primer on Wavelet Analysis, providing the tools for improved methods of calculating the Hurst parameter. Chapter Three discusses the assumption of modeling the data with a linear system, analyzes the non-stationarity of the CGDN+ traces and detrends them. Also, the decorrelating influence of First Differences and the conditions for this influence are presented in Chapter Three. Chapter Four presents the ARIMA (p,d,q) and FARIMA(p,d,q) models for modeling non-stationary and LRD series, and quantifies the LRD existing in the series. Chapter Five employs the recursive Hannan-Rissanen Algorithm for modeling and forecasting a detrended byteout series. Chapter Six summarizes this paper's findings and gives recommendations for continued research.

2 Wavelet Transforms applied to Time Series Analysis

Function Spaces and Wavelet Transforms

What do wavelets contribute to Time Series Analysis? The answer begins with a discussion of Functional Analysis. Every signal is a realization of the process that created it. Other signals created by the same process will enjoy similar properties. A function space can be defined as the set of all functions enjoying these similar properties. In Digital Signal Processing, we work with discrete signals, which are either continuous time signals that have been sampled at regular intervals, or (as in the case of this research) a real process that has been measured at regular intervals and thus is only available at these intervals. Any set of functions can be divided into a set of functions enjoying an additional characteristic, and a set of functions that don't enjoy that additional characteristic.

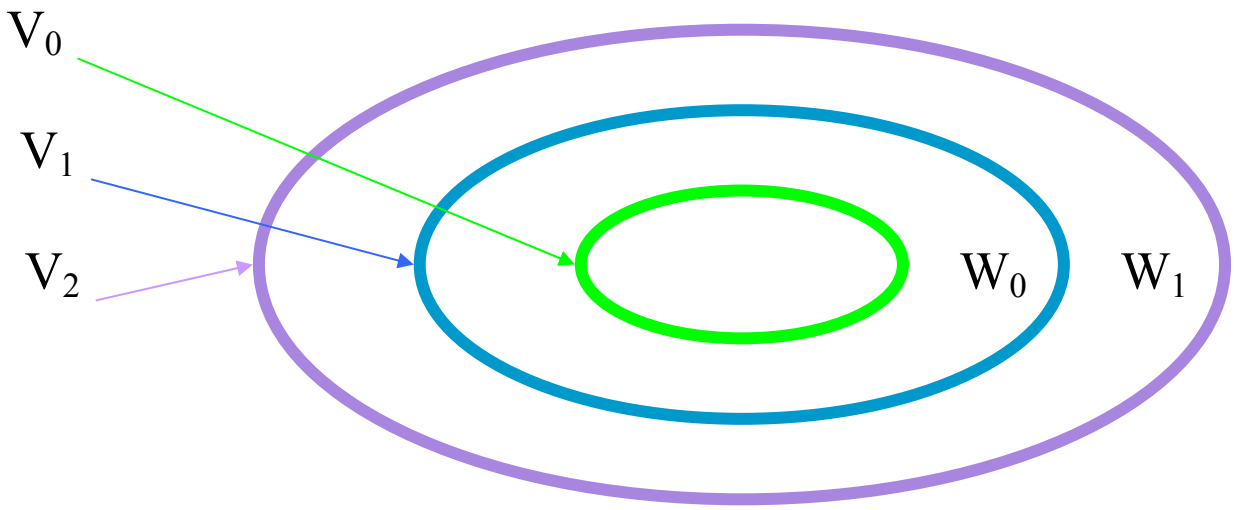


FIGURE 2-1: NESTED SUBSPACES, V_J, AND THEIR ORTHOGONAL COMPLEMENTS, W_1.

Take, for instance, a signal that is sampled at a rate of four times per second. This signal belongs to a function space of all signals that are constant to intervals of 1/4 second, since there is no information about the signal changing between samples. We'll call this function space V_2, and it will be the set of all vectors that remain constant over an interval of 2^{-2} seconds. This

space will certainly contain all the vectors that are sampled at intervals of 1/2 seconds. In fact, these vectors which remain constant over intervals of 2^{-1} seconds will belong to that subset of V_2 which is called V_1. Likewise, the set of all signals constant over periods of one second, called V_0, is a subset of V_1.

When function spaces are nested, as illustrated in Figure 2-1, a class of functions that previously fit in V_j will no longer belong to V_{j-1}, and thus belong to its orthogonal complement, denoted by W_{j-1}. Thus, the "left-overs" are a complementary space to that of the nested subspace. The more general function space, V_j, cannot be fully defined without both the nested subspace, V_{j-1}, and its complement, W_{j-1}.[4] This process of nesting subspaces is called Multi-resolution Analysis. Multi-resolution Analysis involves identifying the key characteristics of a function (in the previous example: constant time intervals) and then matching the function with a set of all functions that enjoy that common characteristic. The common characteristic may be specified further, and with each further specification, the subset of common functions enjoying that characteristic shrinks. Incidentally, it can be said that the CGDN+ series exists in V_{-8}, the nested subset of vectors constant in intervals of at least 256 seconds.

Let a polynomial function evolving over time may be expressed as:

$$f(t) = a_n t^n + a_{n-1} t^{n-1} + \ldots + a_1 t + a_0 \tag{2-1}$$

So that the output, f(t), is related to input, t, by the polynomial having coefficients a_k, where k=1, 2,...,n. The functional relationship between time and signal may be more concisely represented by the n-dimensional vector:

[4] see [23] for a more complete development

$$f = \begin{bmatrix} a_0 \\ a_1 \\ \cdots \\ a_n \end{bmatrix} \tag{2-2}$$

The vector only displays the linear coefficients of the underlying function. Also, a vector may simply consist of the observations in a series, as in the CGDN+ 'packin' series. Vectors that simply present recorded data are commonly termed 'arrays'. Since a vector is another manner of expressing a functional relationship, a 'vector space' may be thought as a synonym of a 'function space'.

Let the basis vector of V_j be orthogonal to the basis vector of W_j by construction, $\sum \overline{w}\overline{v} = \langle \overline{w}, \overline{v} \rangle = 0$. Thus transforming **x** from V_j into V_{j-1} and W_{j-1} can be a lossless transformation. If the bases are likewise orthonormal, $\sum \overline{w}^2 = \sum \overline{v}^2 = 1$, then the transformation will likewise be distortionless. Such is the case with the Haar Basis, that is $\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$ as it appears in $\mathbf{R}^2$.

FIGURE 2-2: TWO BASES FOR THE TWO-DIMENSIONAL COORDINATE PLANE: (LEFT) STANDARD BASIS, AND (RIGHT) HAAR BASIS.

Here we have our introduction to Wavelet Theory. The differencing vector in the Haar basis is the first wavelet vector, and the averaging vector is the first scaling vector. Every wavelet basis consists of two basis vectors: scaling function and wavelet function.

It is also interesting to point out that Figure 2-2(right) may be applied to any discrete time series. With a time series, dimensions are created by delay. Thus, bases in a time series relate present observations with past observations. The total number of dimensions of a time series is determined by its length, N. This is the total possible number of delays that the time series may be subjected to in its analysis. Time series delays are further related to dimensions in Appendix C, "Identifying the Dependencies".

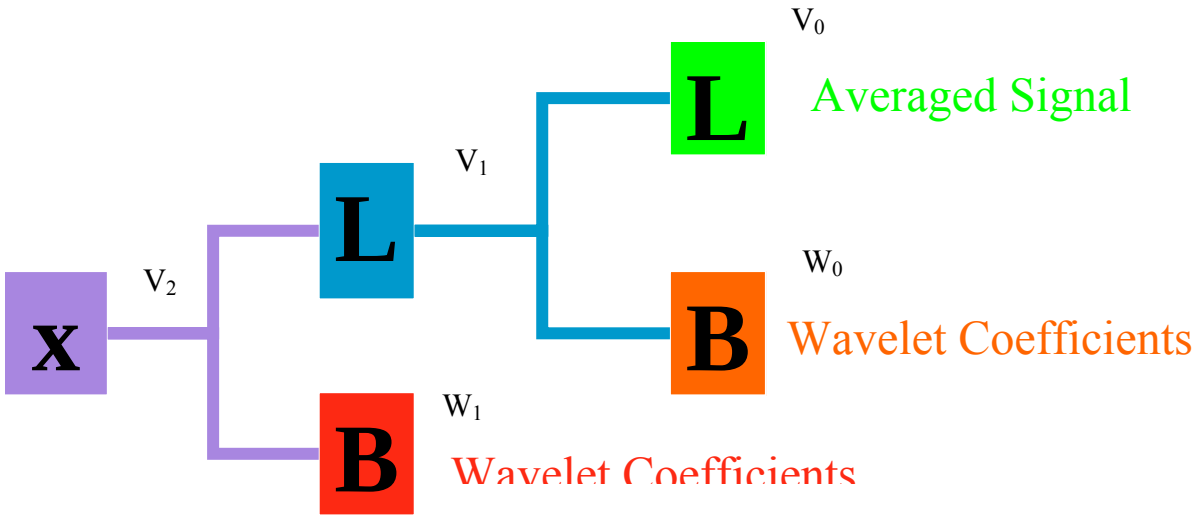


FIGURE 2-3: FILTER BANK DETAILING THE DECOMPOSITION OF A SIGNAL BY SUCCESSIVE FILTERING

The filter bank of Figure 2-3 demonstrates a signal, **x**, being decomposed by a succession of basis vectors. Thus, the signal is averaged by the lowpass filter, and the details are captured by the highpass filter. If the basis vectors used in Figure 2-3 are Haar basis vectors, then the orthogonal wavelet transformation allows that the information captured by the highpass filter is

not redundant to the information captured by the lowpass filter.[5] Thus a partition of vector space V_2 has occurred, and V_2 may be reconstructed by $V_1 \oplus W_1$, where $\oplus$ is the direct sum operator that joins two subsets to a more inclusive superset. Likewise, V may be further decomposed by another bank of lowpass and bandpass filters. If an orthogonal basis has been used to conduct these transformation, then $V_2 = V_0 \oplus W_0 \oplus W_1$. This decomposition can be continued through infinite steps (in theory, but not in practice). In practice, the decomposition will have many levels of wavelet decomposition and a single set of most averaged value, V_{j0}.

[5] The Haar basis vector (like many other wavelet bases) allows for perfect reconstruction as well. More information about the construction of such filter banks is available in [23].

Mathematical Properties of the Wavelet Transform

Mathematically speaking, a wavelet is any function that is:

Condition #1: zero-average

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \tag{2-3}$$

Condition #2: square-integrable to one

$$\int_{-\infty}^{\infty} \psi^2(t)dt = 1 \tag{2-4}$$

Notice that the differencing vector of the Haar basis $\begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$ is a wavelet, meeting both conditions (2-3) and (2-4). On the other hand, a sine wave (the basis of the Fourier Transform) averages to zero, but it is square-integrable to $+\infty$, which disqualifies it as a wavelet transformation. In fact, wavelets must have a finite length (often called 'finite support') to meet (2-4). Wavelet functions may be shifted (translated) in time or scaled as large or small as the following equation shows:

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right). \tag{2-5}$$

This property makes wavelets a powerful transform for representing non-stationary signals. The unshifted and undilated (meaning 'regular sized') wavelet, $\Psi_{0,1}(t)$, is known as the mother wavelet. All other wavelets are shifted and scaled versions of the mother wavelet. A single wavelet pair consists of the scaling function and its mother wavelet. A particular scaling

function can create a mother wavelet (and henceforth all other wavelets in that 'family') through an iterative process.[6]

At each split of the filter bank the signal, f(t) or its approximation, is correlated with both the scaling function on the lowpass channel or the wavelet function on the bandpass channel. The output of the bandpass branch is stored as the wavelet coefficients at that scale, and the output of the lowpass branch is the next averaged version of the input. The correlation is accomplished by an inner product, yielding

$$W_f(u,s) = \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{s}} \psi^* \left(\frac{t-u}{s} \right) dt \quad . \tag{2-6}$$

We have used f(t) and continuous time for generality's sake, although an analogous inner product transformation exists for discrete signals.

The Fourier Transformation is also useful to decode the frequency content of a time-varying signal. The Fourier Transform or the Wavelet Transform (or numerous related transformations) may be used to convert the signal into the frequency domain, but the operation (an inner product) is similar in each transformation. The signal is correlated with the basis function of that transformation (e.g.- a sine wave or a wavelet) to measure its similarity. The output of this correlation is a number representing the signal's content at that frequency and/or scale. A primary advantage in using the Wavelet Transformation instead of the Fourier Transformation is that some measure of time is preserved in the wavelet coefficients. The Fourier Transform cannot preserve time because its basis vectors (sinusoids) have infinite duration. Another significant advantage for using the Discrete Wavelet Transform is that DWT can be performed using an order of 'n' (or 'O(n)') calculations, which is even less than the

[6] see reference [23] for more information about the iterative process of constructing wavelets from their scaling functions.

number of calculations required to conduct the very efficient FFT, which uses an order of $n\log_2 n$ calculations. [16]

Now, to answer the question of how wavelets contribute to Time Series Analysis. The finite support condition of (2-4) allows for viewing the frequency and time characteristics of a series in the same plot through a wavelet transform. Wavelet coefficients are pretty well localized in both time and frequency. Heisenberg's Uncertainty Principle (HUP) assures that time and frequency cannot both be known exactly at the same time [23]. The HUP theorem states that if ||f|| = 1, then the product of $\sigma\hat{\sigma} \geq 1/2$. The function's time spread is defined as $\sigma^2 = \int_{-\infty}^{\infty} t^2 |f(t)|^2 dt$, and its frequency spread is defined as $\hat{\sigma}^2 = \int_{-\infty}^{\infty} \omega^2 \left|\hat{f}(\omega)\right|^2 d\omega$. Thus, increasing the analyzing wavelet's frequency by 2^j results in a larger spread of frequencies, and causes the time interval of wavelet coefficients to decrease by 2^j which is a narrower time spread.

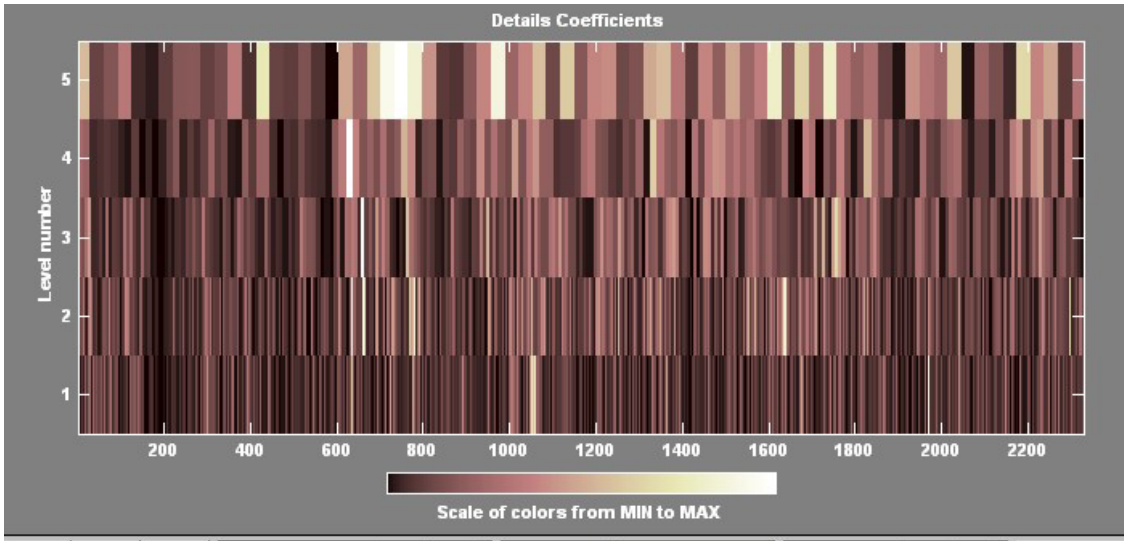


FIGURE 2-4: DISCRETE WAVELET TRANSFORM (DWT) OF BYTEIN SERIES. TRANSFORM CALCULATED USING MATLAB'S WAVELET TOOLBOX. SCALING IS ARRANGED IN DESCENDING ORDER FROM LOWEST TO HIGHEST FREQUENCIES.

A frequency decomposition represents a process by its energy at various frequencies. The Discrete Wavelet Transform (DWT) of bytein similarly displays wavelet coefficients at a

variety of scales. The variance of wavelet coefficients at each scale, u, is referred to as the scale variance $\hat{\sigma}_f^2(u,s)$. Scale Variance is a scale decomposition of a process's energy similar to that performed in a Power Spectral Density.

One use of the scale variance is to provide an unbiased estimate of the variance of a stationary process. The Power Spectral Density (PSD) of a stationary process Y(t) may be integrated as $\int_{-1/2}^{1/2} S_Y(f)df = \sigma_Y^2 \equiv Var\{Y_t\}$ to provide the process's variance. Unfortunately, the sample variance relies upon the sample mean[7], $\tilde{\sigma}_Y^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sigma_Y^2 - Var(\overline{Y})$ making it a biased estimator because $\overline{Y}$ converges in the limit to μ_Y. Since in practice we analyze finite series, therefore the sample mean $\overline{Y} \neq \mu_Y$ does not usually equal true mean, although they will be equal asymptotically, $\lim_{n\to\infty}(\overline{Y}) \to 0$.

On the other hand, scale variances $\hat{\sigma}_f^2(u,s)$ are obtained through an orthogonal (lossless) transformation and have zero mean.

$$E\{w_f(u,s)\} = \sum_{s=0}^{L_u-1} h_{u,s} E\{Y_{t-s}\} = \mu_Y \sum_{s=0}^{L_u-1} h_{u,s} = 0 \qquad (2\text{-}7)$$

where $h_{u,s}$ are the coefficients of the discrete wavelet filter. The average value of the wavelet coefficients at any scale, therefore is 0. The scale variance, then is

$$\hat{\sigma}_f^2(u,s) \equiv Var\{w_f(u,s)\} = E\{w_f^2(u,s)\} - [E\{w_f(u,s)\}]^2 = E\{w_f^2(u,s)\} = \frac{1}{n_j}\sum_{s=0}^{L_u-1} w_f^2(u,s) \qquad (2\text{-}8)$$

Scale variance is an unbiased estimator of a process's variance across each scale. Process variance is then obtained by

[7] see [21] for explicit proof

$$Var\{Y_t\} = \text{var}\{v_f(J_0,s)\} + \sum_{u=1}^{J_0} \hat{\sigma}_f^2(u,s) \tag{2-9}$$

where $v_f(J_0,s)$ is the scaling function of f(t) after J_0 levels of wavelet decomposition. Admittedly, the accuracy of variance is limited by the number of wavelet decompositions are taken. In theory, infinite steps of decomposition will provide a series of scale variances that sum to the true variance. Practical series, however, are finite and the wavelet decomposition is approximated at some point J_0. A thorough treatment of the accuracy and cost associated with approximation at level J_0 can be found in [23].

A wavelet-based method of estimating the Hurst parameter, H, is presented in [1]. The wavelet method is performed by conducting a time average of squared wavelet coefficients at each scale.

$$\hat{\sigma}_x^2(u,s) \equiv \frac{1}{n_j} \sum_{s=0}^{L_u-1} w_x^2(u,s) \tag{2-10}$$

Since wavelet coefficients are a zero-mean process of x(n), the time average of squared wavelet coefficients is the variance of wavelet coefficients at that scale. This 'scale-variance', $\hat{\sigma}_x^2(u,s)$ is the measure of energy that lies within a given bandwidth around the center frequency of each scale in x(n). When each of these scale-variances are plotted (as Figure 3-16 (right) will reveal), the slope, β, is found to be a robust estimate of the process's H parameter

$$H = \tfrac{1}{2}(\beta + 1) \tag{2-11}$$

With this wavelet estimator, LRD series will display a positive spectral slope. But, if the series's wavelet variances were flat, the series is uncorrelated, or if they decrease, then the series is SRD.

Reference [1] includes a thorough treatment of the wavelet-based estimator, including a detailed comparison with another standard H-estimator, the D-Whittle estimator. This paper

shows that the wavelet estimator is resistant to polynomial and sinusoidal trends when a decomposition wavelet with sufficient vanishing moments is chosen. Wavelet bases have a number of vanishing moments, N,

$$\int t^k \psi_0(t) dt \equiv 0 , \quad \forall \text{ k=0,1,...,N-1} \tag{2-12}$$

So, polynomial trends t^k in the analyzed signal (up to order N-1) are not transformed with the signal into the wavelet domain. The Haar wavelet is the first in a series of Daubechies wavelets, numbered in accordance with the number N vanishing moments that the basis wavelet can ignore. [1] also explains that using a wavelet basis with more vanishing moments means increasing the wavelet's coefficients. So, N=H+1 is suggested as a good balance, since more coefficients can lead to inaccuracy due to border effects, and fewer coefficients results in an estimate with higher variance. Finally, [1] indicates that the wavelet method is an efficient estimator of $\hat{H}$, with a variance equal to the Cramer-Rao lower bound.

This research has also found the wavelet method able to accurately measure H with shorter batches of samples than other standard H-estimates. Byteout was aggregated successively and then measured for H. The log-variance method produces H-estimates with unreasonable variance when the number of samples is reduced to 100 samples. However, the wavelet method is able to produce consistently LRD results with data samples of down to 50 samples (see Table 2-5).

Sequence Title	H (by wavelet method)	H (by sample var. method)
Byteout (full sequence)	0.975	0.836
Byteout(1:500)	0.927	0.867
Byteout(1:200)	0.808	0.803
Byteout(1:100)	0.800	Inf
Byteout(1:50)	0.659	Inf
Byteout(1:20)	1.007	Inf
Byteout(1:10)	0.850	Inf
Byteout(1:5)	-1.805	Inf

TABLE 2-5: HURST ESTIMATES OF SUCCESSIVELY SHORTER SEQUENCES AS MEASURED BY WAVELET AND LOG-VARIANCE METHODS.

3 Assumptions of Linearity and Stationarity

Gaussianity employed in highly-aggregated internet traces

Choosing the most appropriate model depends upon the assumptions that may be made regarding the CGDN+ series. This chapter will demonstrate that the CGDN+ series can be modeled with a linear system, but that these series are not stationary. The series will then be detrended to allow for comparison between Hurst parameter estimators in the next chapter, as well as to simplify the modeling conducted in Chapter Five.

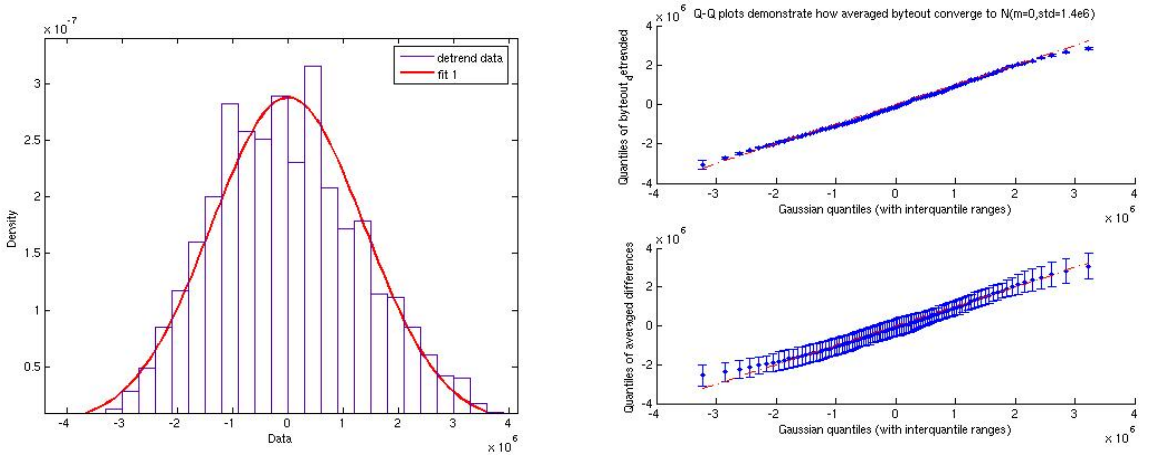


FIGURE 3-1: (LEFT) HISTOGRAM BYTEOUT AFTER DETRENDING APPEARS GAUSSIAN DISTRIBUTED. (LEFT) Q-Q PLOTS[8] SHOW MORE CONCLUSIVE EVIDENCE THAT DETRENDED BYTEOUT IS GAUSSIAN DISTRIBUTED. (LEFT-TOP) ORDERED BYTEOUT BROKEN INTO 100 QUANTILES. (BOTTOM) QUANTILES EXTRACTED FROM 23 SERIES OF BYTEOUT, THEN QUANTILES ARE AVERAGED.

Finding gaussianity in the differenced CGDN+ series simplifies analysis of these highly aggregated internet time series. This thought runs counter to current research, where assumptions of gaussianity are relaxed whenever possible. Yet, the detrended byteout series is manifestly Gaussian (as shown in Figure 3-1). Similar tests showed that the detrended bytein, packout and packin series also converge to a Gaussian distribution. According to R. E. Kalman [10], "assuming independent gaussian primary random sources, if the observed random signal is also gaussian, we may assume that the dynamic system between the observer and the primary

[8] Q-Q plots are created by comparing quantiles of the series under testing against quantiles of a standard distribution (e.g.-Gaussian distribution).

source is linear." This means that since the detrended CGDN+ series appear Gaussian distributed, it is reasonable to use a linear system in modeling these series.

A linear model is more easily manipulated by engineering tools, and linear system theory is much more complete than nonlinear systems theory. Also, the Gaussian distribution is stable and its domain of attraction includes many commonly found distributions, including the uniform and the exponential. Linear stationary models, as well as the ARIMA (p,d,q) and FARIMA(p,d,q) are linear models [4], [6], and [7]. Following Kalman's reasoning, it is reasonable to expect that a linear model, fueled by Gaussian inputs could recreate the CGDN+ series.

Internet series with higher time resolution are not generally Gaussian distributed. According to [2], "...Weibullian or log-normal behavior is more common than Gaussian [in internet traces], unless the data has already been highly aggregated or if scales above a few seconds are examined." Similarly, Leland and Wilson published finding that "LAN traffic is extremely burst across time domains spanning six orders of magnitude" [11]. *Finding Gaussianity in the aggregate traces of CGDN+ is not a trivial finding*. The CGDN+ (and more general internet traces) converge to the Gaussian distribution according to the Central Limit Theorem. Therefore, the exponent of high-resolution traces cannot be so large as to create a thick tailed distribution that would diverge from the Gaussian upon aggregation.

Discovering Gaussianity in highly aggregated internet series can work to the network administrator's advantage, by providing a strong foundation for modeling through linear systems and Gaussian inputs. Whereas the distributions of highly aggregate internet series are only mentioned in [2] and [11], one of the significant contributions of this work is to point out the simplifications made available by finding that the CGDN+ series are Gaussian distributed.

CGDN+ series experience a strong daily cycle

Perhaps the experienced reader suspected that the CGDN+ data was nonstationary by viewing Figure 1-2. The averaged signal in Figure 3-2 presents another evidence of non-stationarity. As mentioned in Chapter 2, the wavelet decomposition separates high frequencies from lower frequencies (or an averaged signal). The wavelet decomposition likewise allows for perfect reconstruction of the original sequence. If perfect reconstruction is not desired, thresholding the wavelet coefficients before synthesis provides a wavelet method of denoising the signal. Figure 3-2 shows a 25-day packet arrival series whose wavelet coefficients were thresholded; roughly 25 peaks evidence the daily cycle found in much of the CGDN+ ship-shore traffic. Daily cycles are frequently found in internet series.

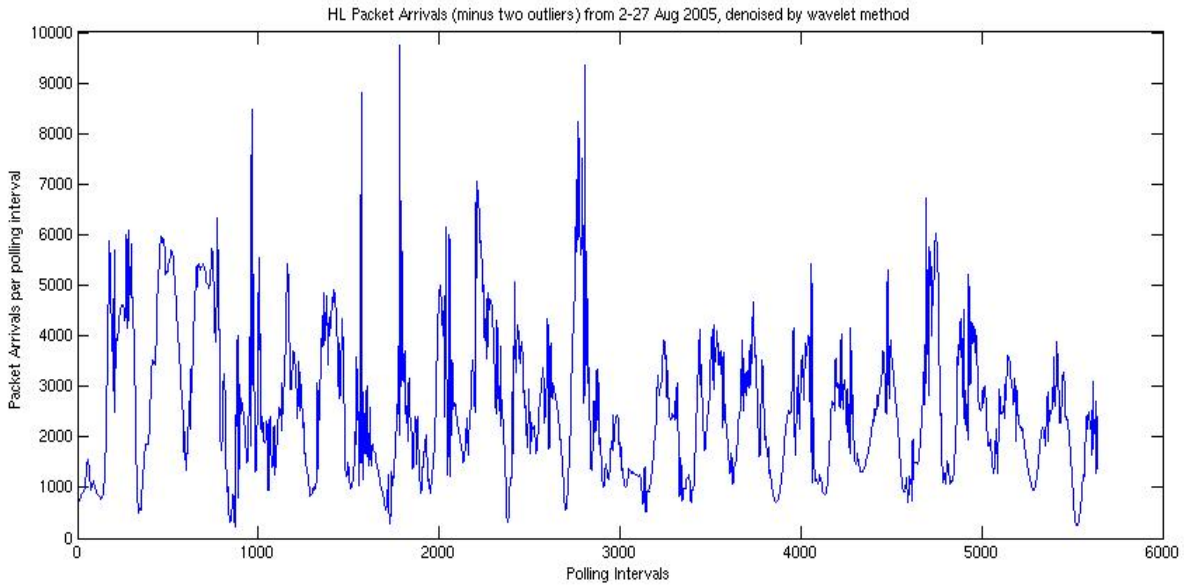


FIGURE 3-2: AVERAGE PACKET ARRIVALS OVER 25 DAYS.

The reader may notice that two outliers are subtracted from the packet arrivals series before Figure 3-2 is calculated. Outliers are a serious consideration in analyzing the CGDN+ data, where mistakes are sometimes created by the data collection process. Additionally, long polling intervals occur on occasion, giving a disproportionately high packet/byte statistic for that interval. The two outliers subtracted from the sequence in Figure 3-2 are identified visually,

since they are so large as to visually diminish all other observations. The four series analyzed in this research did not have similarly self-evident outliers, and were not subjected to robust or classical methods of outlier detection prior to analysis.

Figure 3-2 gives a more precise estimate of the periodicity in each CGDN+ series. The series byteout, packin and packout are approximately 24 hours periodic. Periodicity of the bytein series is much less pronounced, and seems to be out of sync with packin until after about lag 1000. Perhaps bytein is composed primarily of control data. Control data (such as acknowledgements for packets received) are a more consistent form of internet traffic. Since the network will constantly check that the cutter is still online, some of the control data will not be dependent upon the cutter's workday. Byte arrivals seem to indicate traffic volume, while packet arrivals give an idea of the changing traffic frequency. When more packets are sent, then more acknowledgments (which are byte-wise small packets) will be received. Thus packin expresses much weaker 24 hour periodicity than do the other series.

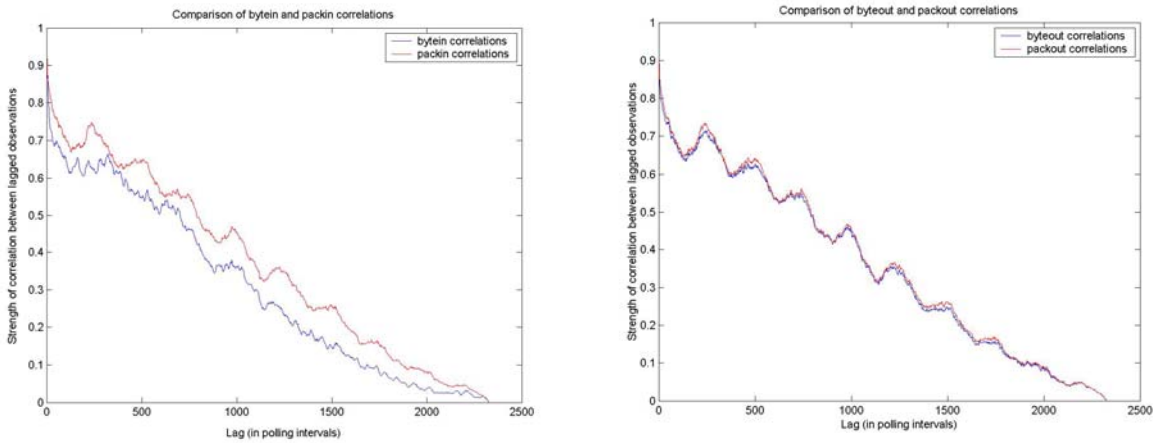


FIGURE 3-3: AUTO-CORRELATION FUNCTIONS FOR CGC FORWARD DATA SERIES BETWEEN 1-11 JULY 2005. (LEFT) PACKET ARRIVALS VS. BYTE ARRIVALS, (RIGHT) PACKET TRANSFERS VS. BYTE TRANSFERS.

Figure 3-4 applies the Fourier Transform to further quantify the estimate of byteout periodicity. A Hanning Window (employed in Figure 3-4 (C) and (D)) reduces sidelobe interference before calculating the series's Magnitude Response. Further frequency domain

smoothing is accomplished by processing the Fourier Transform of a sequence's autocorrelation function (in Figure 3-4 (B) & (D)). This method takes advantage of the Wiener-Khinchine relation that relates Spectral Density to the Fourier Transform of the Autocorrelation function.[9]

$$S_Y(\omega) = \int_{-\infty}^{\infty} R_Y(\tau) e^{-j\omega\tau} d\tau = \Im\{R_Y(\tau)\} \tag{3-1}$$

Smoothing is attained by this method because an estimated autocorrelation of byteout is used. Therefore, only the most significant frequencies of byteout are transformed in Figure 3-4 (B).

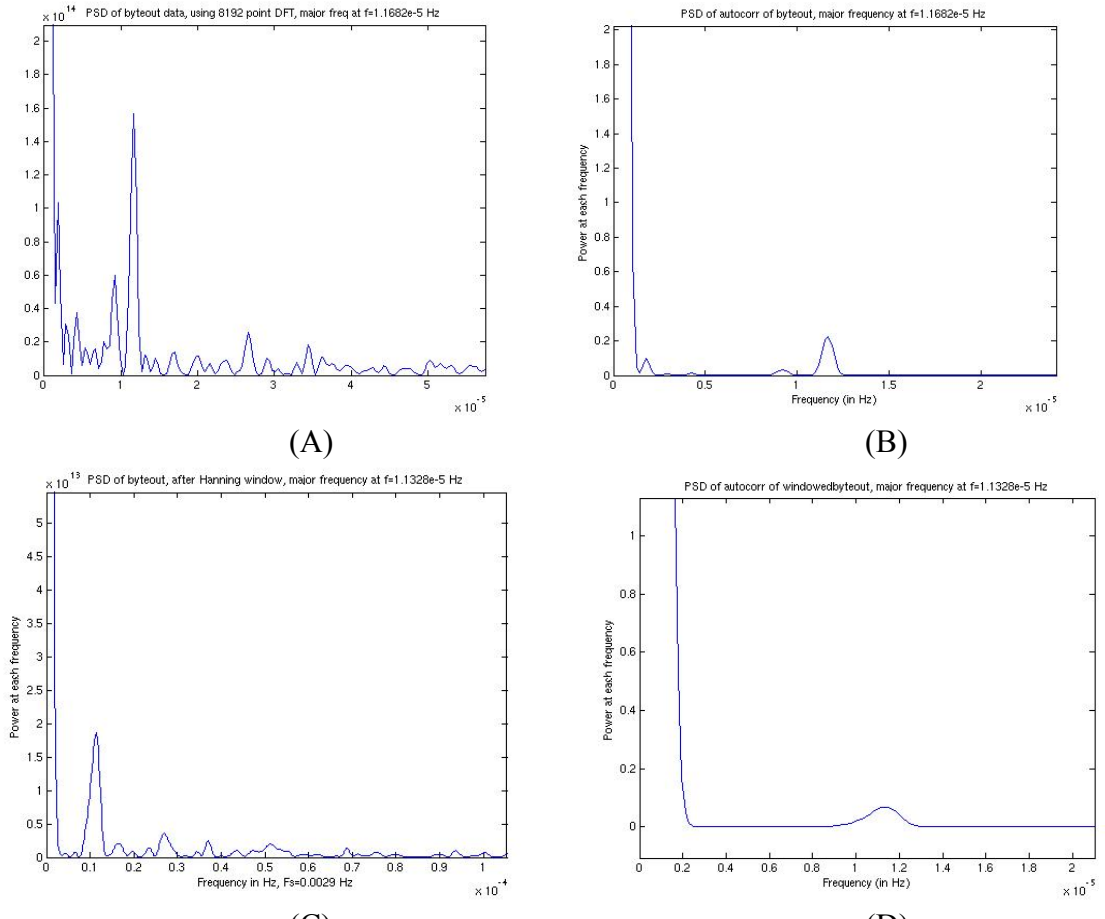


FIGURE 3-4: POWER SPECTRAL DENSITIES (PSD) OF BYTEOUT SERIES DEMONSTRATING ITS DAILY PERIODICITY. (A) PSD OF UNWINDOWED BYTEOUT SERIES, (B) PSD OF THE AUTOCORRELATION FUNCTION OF UNWINDOWED BYTEOUT, (C) PSD OF BYTEOUT AFTER HANNING WINDOWED, AND (D) PSD OF AUTOCORRELATION FUNCTION OF BYTEOUT AFTER HANNING WINDOW IS APPLIED.

[9] The relation between autocorrelation function and sample spectrum is also proved in [6].

Figure 3-4 (D) shows that the 24-hour cycle of byteout is truly the most powerful trend in this series. Similar power spectral densities were calculated for packin and packout series. The PSD of byteout and PSD of windowed byteout both have strongest frequencies at 1.1682e-5 Hz, which corresponds with a trend period of T=23.778 hrs. Interestingly, the Hanning window has smoothed the frequency domain, but also widened the main lobe. This increased frequency spread results in a strongest frequency peak of 1.1328e-5 Hz (or T=24.5213 hrs). Each of these most significant peaks is very close to the daily frequency (f_{24hr} =1.1574e-5 Hz). Bytein has its most significant peak at the daily frequency, but this peak is 1/15 the power of that observed in the other series.

Since the Coast Guard data series has a strong 24-hour periodicity, it must be detrended before a linear stationary model can be used. Any trend in the data, like periodicity, artificially inflates any calculations of variance, making model identification much more challenging. Also, any trend in the data acts itself as a long-range dependency and artificially inflates any dependency calculation. Although [1] demonstrates that the Wavelet Method is resistant to periodicity and polynomial trends, the Sample Variance method (to be presented in the next chapter) is not so robust. Therefore, detrending must be accomplished before a fair comparison can be accomplished. Three methods for detrending will be discussed below.

Daily Cycles

Figure 3-5 displays daily cycles of byteout, packout and packin by averaging 3 relatively long, stable sequences of CGDN+ statistics. An average cycle in byteout for CGC FORWARD has its peak at about 0100 GMT[10], after which byteout steadily falls to a minimum at about 0930 GMT. Shortly after reaching the minimum, byteout quickly climbs toward its 1500 GMT level, which it maintains or exceeds through the rest of the evening. The packout and packin daily

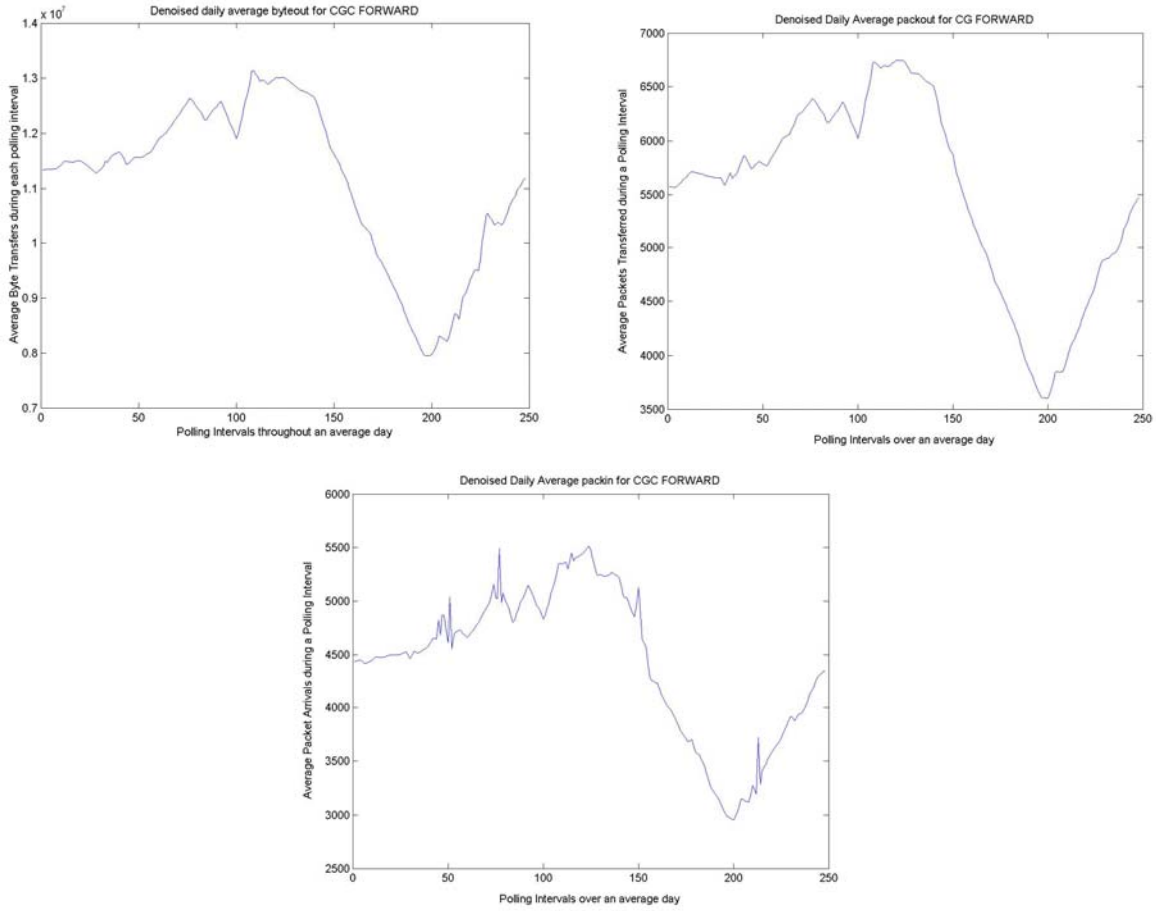


FIGURE 3-5: AVERAGE DAILY TRENDS OF INTERNET DATA EXPERIENCED BY CGC FORWARD: (LEFT) BYTEOUT, (RIGHT) PACKOUT, AND (BOTTOM) PACKIN.

[10] Greenwich Mean Time, subtract Zone difference to calculate Local Mean Time. For example, if the cutter is operating primarily between 67.5W and 82.5 W, then 0100 GMT would equate to 2000 LMT.

cycles closely match the shape of byteout. Figure 3-5 daily trends were averaged from over a week of data during early July, late July and early November. Each of these weekly averages has been decomposed by a Daubechies 3 (db3) wavelet over 6 levels and denoised.

It is possible to subtract the average daily trends from byteout and the other CGDN+ series, similar to analyzing seasonal data[11]. Figure 3-6 shows the partial correlations that remain in the detrended series after its daily trend is subtracted. These partial correlations were obtained by the Yule-Walker method, mentioned in Appendix C, "Identifying the Dependencies in Linear Stationary Models".

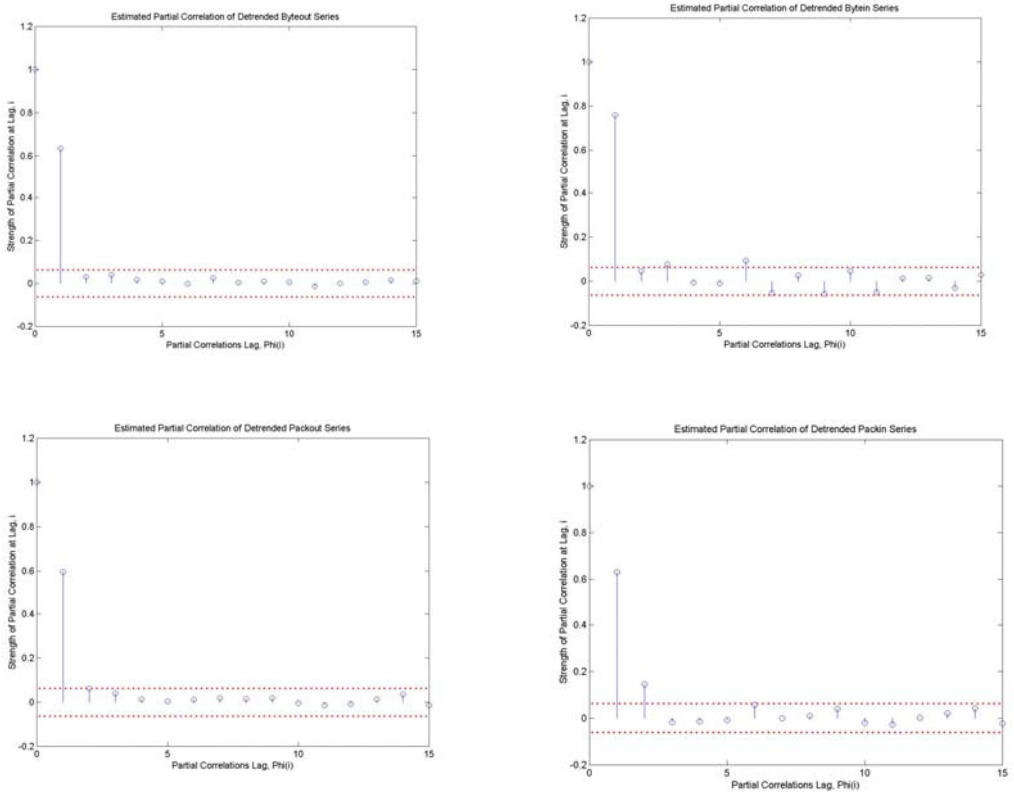


FIGURE 3-6: PARTIAL CORRELATION FUNCTIONS FOR DETRENDED SERIES (LEFT TO RIGHT, TOP TO BOTTOM): BYTEOUT, BYTEIN, PACKOUT AND PACKIN.

It should be pointed out that the spread of polling interval lengths (see Figure 3-7) and very different operating tempos over vastly different weeks makes subtracting the average daily trend

[11] For a thorough discussion of forecasting seasonal data, see [6] and [7].

seem an imprecise method of detrending. However, it will be shown that the partial correlations of each series, whether detrended using daily cycles or another method, are equivalent.

It is interesting to point out that the polling method for collecting CGDN+ statistics was altered to make polling intervals more consistent before modeling was attempted. Formerly, plotting the time intervals in a histogram revealed that CGDN+ polling intervals were exponentially distributed ($\bar{x}$ = 8 minutes) with some outlying intervals of hundreds of hours! Altering the polling method has improved interval consistency, yet Figure 3-7 reveals large polling intervals that are left in the data to be averaged out by very short intervals. Polling intervals of extreme[12] size have been used as natural breaks in the data, causing that the longest CGDN+ series are less than two weeks in duration, although several months of data are available. There are over 2300 polling intervals for CGC FORWARD during 1-11 July 2005 that are Gaussian distributed, N($\bar{x}$=5'42", σ=6.48"), which is relatively stable and long for the CGDN+ traces available.

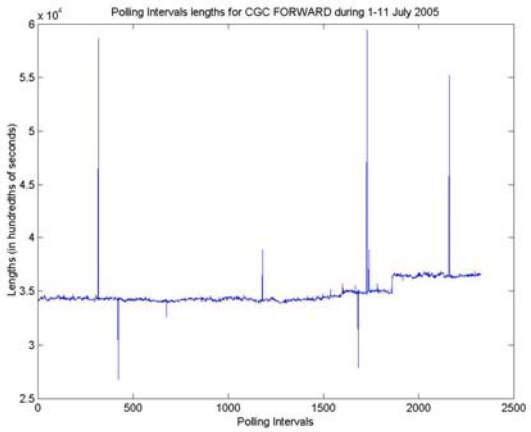


FIGURE 3-7: POLLING INTERVAL LENGTHS OF CGC FORWARD DURING 1-11 JULY 2005

[12] Extreme size here denotes polling intervals so large as to dwarf the rest of the polling intervals.

Filtering the 24 hour cycle

Since the DFT reveals true sinusoidal periods present in the data, a notch filter has been chosen as the second method of detrending. The elliptical filter (in Table 3-8) is developed using MATLAB's Filter Design and Analysis Tool (FDATool). As FDATool will not accept small sampling frequencies, $f_s<1$; so $f_s=1000$ Hz is used to design the filter. Specification frequencies are adjusted accordingly. Coefficients for this sixth order elliptical filter are as they would be for $f_s=1.1574\text{x}10^{-5}$ Hz

Elliptical Notch Filter		Chebyshev II bandpass filter	
Numerator, A(z)	Denominator, B(z)	Numerator, A(z)	Denominator, B(z)
0.9782	1.0000	1.688e-5	1.000
-5.8674	-5.9542	-6.751e-5	-5.994
14.6656	14.7734	8.437e-5	14.97
-19.5529	-19.5522	2.587e-19	-19.95
14.6656	14.5574	-8.437e-5	14.96
-5.8674	-5.7813	6.751e-5	-5.980
0.9782	0.9567	-1.688e-5	0.996

TABLE 3-8: FILTER COEFFICIENTS FOR (LEFT) REMOVING AND (RIGHT) RESTORING 24 HOUR TREND.

Figure 3-9 shows the partial correlation functions of each filtered series. They are remarkably similar to those calculated after detrending the original series (see Fig. 3-5). The Chebyshev II bandpass filter for reconstruction has not been calculated to ensure perfect reconstruction of the original sequence. The forecasted and detrended byteout series are compared in Chapter 5 with the detrended byteout series, and reconstruction of the original byteout series is unnecessary for this analysis.

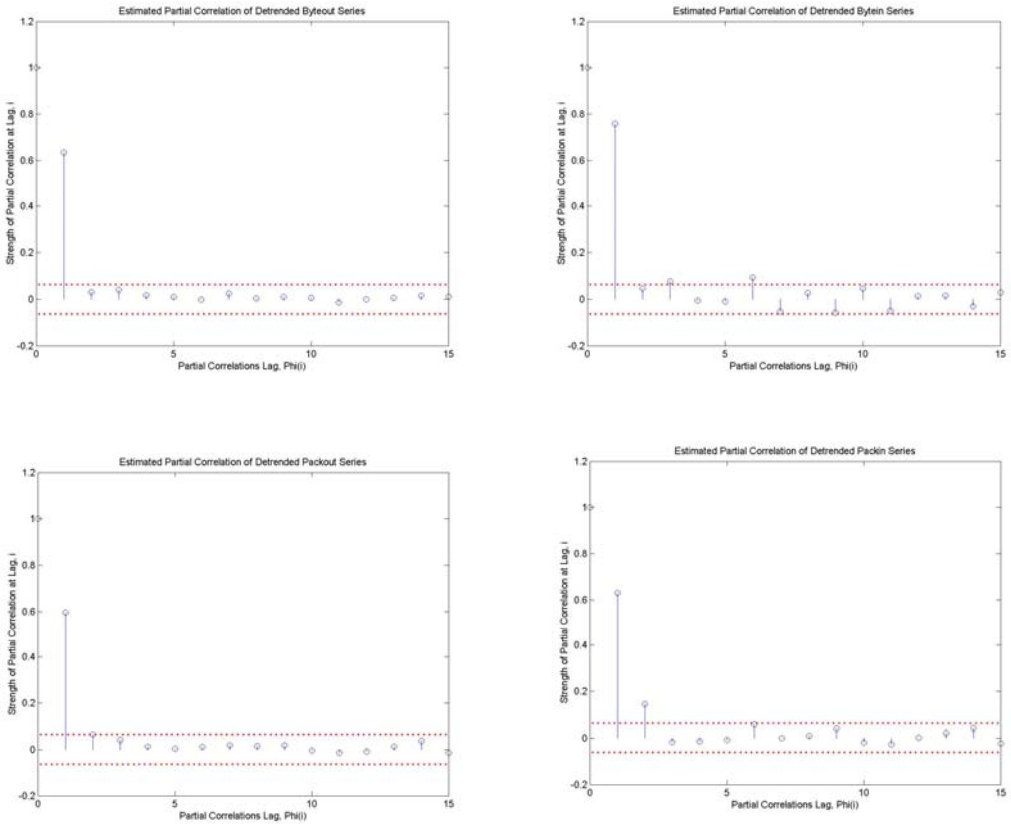


FIGURE 3-9: PARTIAL CORRELATION FUNCTIONS FOR FILTERED SERIES (LEFT TO RIGHT, TOP TO BOTTOM): BYTEOUT, BYTEIN, PACKOUT AND PACKIN.

Chapter 4 will show that detrended byteout is a LRD sequence. Yet Figures 3-6 & 3-9 show that detrended byteout has one significant partial correlation (at lag one). Since an AR(1) model cannot produce a LRD sequence, this result remains unexplained.

Beware the First Difference Operation!

$$Y_n = X_n - X_{n-1} \tag{3-2}$$

The first difference operation involves measuring the change between the process at one time instant and the next time instant. So, although it is unreasonable to predict negative byte arrivals in a given polling interval, it is not unreasonable to predict a relative decrease in the 'bytein' series (see Figure 3-10). Simplicity is the main strength of the First Diffferences method. First differences may be recreated to form the original 'bytein' series using this simple summation:

$$Z_{i+1} = Z_i + Y_i \tag{3-3}$$

where Z_1=0.

The Linear Stationary Models insist upon first and second order statistics that are not functions of time. Figure 3-10 shows that the 'first differences' of bytein appear to have a constant mean, but variance still changes with time. Estimated variance between polling intervals 160-200 is 4.38×10^9, while variance between polling intervals 1000-1150 is 2.99×10^{11}. Therefore, first differences of bytein are not stationary to the second order.

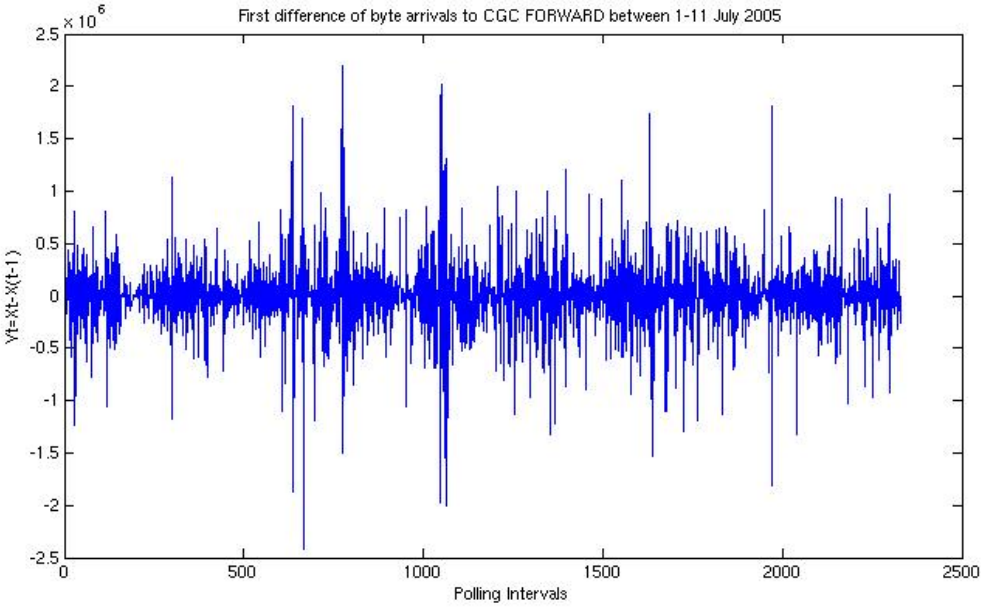


FIGURE 3-10: FIRST DIFFERENCE IN BYTE ARRIVALS FROM CGC FORWARD (1-11 JULY 2005)

Nonstationary variance is not the only problem with the First Difference series, but Figure 3-11 shows evidence that the first difference operation exhibits a decorrelating influence upon the byte arrivals series. This sequence was originally LRD (see Figure 1-3 for example) with $H = 0.87$. After using the first differences operator, the process has become SRD, with $H = 0.135$. This finding caused me quite a bit of confusion while conducting this research. For a time, I was convinced that the highly-aggregate CGDN+ traces were SRD after decorrelation. *Only after decorrelating by other methods (as above) did I learn that the first differences method of detrending in fact decorrelated the bytein series!* Other time series analysts should also be aware of the effect that the first differencing operation has upon time series dependencies, lest they also confuse their committee members!

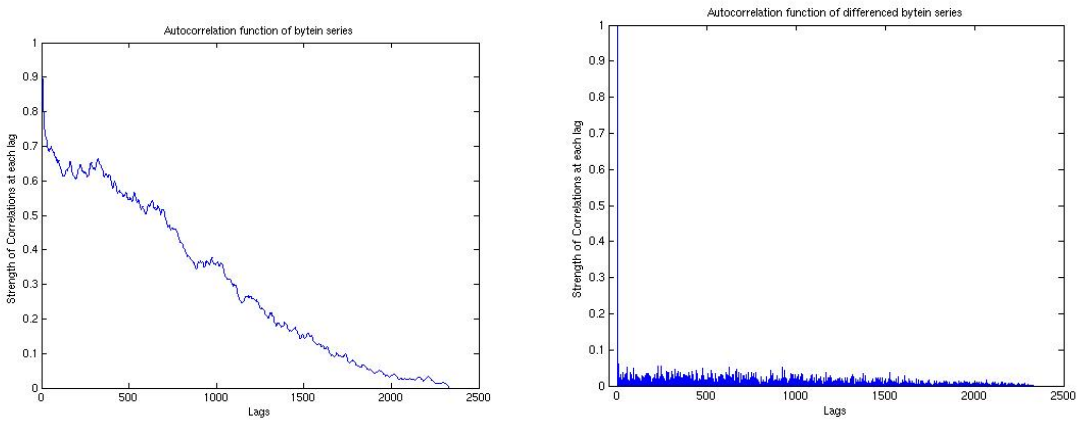


FIGURE 3-11: DECORRELATING INFLUENCE OF THE DIFFERENCING OPERATION.

A direct proof follows to demonstrate that the first difference operation significantly decorrelates an AR(1) process. Z_t is a zero-mean, stationary process with covariance function, γ_k. From Equation (1-1),

$$\begin{aligned} Y_{t-k}Y_t &= Y_{t-k}Z_t - Y_{t-k}Z_{t-1} \\ Y_{t-k}Y_t &= (Z_{t-k} - Z_{t-k-1})Z_t - (Z_{t-k} - Z_{t-k-1})Z_{t-1} \\ Y_{t-k}Y_t &= Z_{t-k}Z_t - Z_{t-k-1}Z_t - Z_{t-k}Z_{t-1} + Z_{t-k-1}Z_{t-1} \end{aligned} \tag{3-4}$$

Now, the covariance function of the first difference is found by applying the expectation operator, E[ ], to these zero-mean processes.

$$\begin{aligned}\gamma_k^* &= E[Y_{t-k}Y_t] = E[Z_{t-k}Z_t] - E[Z_{t-k-1}Z_t] - E[Z_{t-k}Z_{t-1}] + E[Z_{t-k-1}Z_{t-1}] \\ \gamma_k^* &= \gamma_k - \gamma_{k+1} - \gamma_{k-1} + \gamma_k \\ \gamma_k^* &= 2\gamma_k - \gamma_{k+1} - \gamma_{k-1}\end{aligned} \tag{3-5}$$

As explained in Appendix C, "Identifying the Dependencies in Linear Stationary Models", a covariance function may be expressed as a sum of its delays. In the case of a AR(1) model, $\gamma_k = \phi_1\gamma_{k-1}$. Therefore,

$$\begin{aligned}\gamma_k^* &= 2\phi_1\gamma_{k-1} - \phi_1^2\gamma_{k-1} - \gamma_{k-1} \\ \gamma_k^* &= (2\phi_1 - \phi_1^2 - 1)\gamma_{k-1}\end{aligned} \tag{3-6}$$

In the case of an AR(1) model with $0.65<\phi_1<1.35$, the first difference has a strong decorrelating influence upon the analyzed series. If $\phi_1=0.8$, then $\gamma_k^* = -0.04\gamma_{k-1} \approx 0$. On the other hand, the first difference of an AR(1) process with $\phi_1=-0.8$ results in a more strongly correlated series (see Figure 3-12).

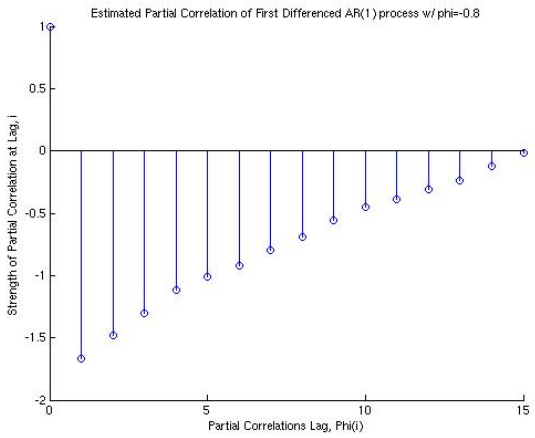


FIGURE 3-12: PARTIAL CORRELATIONS OF AN AR(1) MODEL (WITH ϕ_1=-0.8) AFTER FIRST DIFFERENCING. RESULTS PLAINLY SHOW THE STRONG CORRELATIONS INTRODUCED TO THE SERIES BY F.D. OPERATION.

Please realize that these results are specific to a range of AR(1) models parameters. The first difference equation does not necessarily have a similar decorrelating influence upon higher order

AR(p) or ARMA(p,q) models. The results are important to this study since the first partial correlation of each CGDN+ series is between $0.65<\phi_1<0.8$ (see either Figure 3-6 or 3-9).

The first difference operation (3-2) can be rewritten as

$$Y_n = \begin{bmatrix} 1 \\ -1 \end{bmatrix} X_n \tag{3-7}$$

making it a scaled (or rather unscaled) version of the Haar wavelet. The First Difference operation $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ also acts as an MA(1) filter with $\theta_1=1$, see Equations (B-3) and (B-10). Therefore, both the first-difference equation and Haar wavelet are high-pass filters.[13] So, it can be expected that the Haar Wavelet transform will similarly have a decorrelating influence upon the analyzed bytein series. More interestingly, the Haar Wavelet transform will also

[15] and [21] assert that wavelet functions exhibit a decorrelating influence upon the series they transform. Although the finding is not used in this work, [15] uses this finding to present a simple method of modeling LRD series by transforming uncorrelated data from the wavelet domain into the time domain. [15] demonstrates the decorrelating influence of the wavelet transform using LRD sequences generated by a FARIMA(p,d,q) model, but Figure 3-11 provides evidence that wavelet transforms indeed decorrelate real internet traces. Therefore, data analysts should use the first difference operation with caution, especially when the result will be measured for correlations.

Differencing the series by a given period does not seem to have the same decorrelating influence upon the data. Calculating Equations (3-4) & (3-5) using instead $Y_n = X_n - X_{n-T}$ yields a very different result; the analyzed signal is not decorrelated. The period, T, can be identified by measuring the distance between peaks on the original series' autocorrelation functions (see

[13] See Appendix B for further explanation of MA(q) filters and low/high-pass filters.

Figure 3-3). Again, the partial correlation results for each 'differenced by period' series are practically identical to those obtained by the other two detrending methods, for which reason the results are not repeated here (see Figure 3-6 or 3-9).

4 Modeling LRD data

Internet traffic has been very popular in academic literature primarily because of the long-range dependencies (LRD) they exhibit. Figure 1-3 shows an example of these LRD, where the dependencies shown in the autocorrelation function of bytein decay hyperbolically. As explained earlier, the CGDN+ appears LRD until trends are removed. In the case of Figure 1-3, old observations separated by a very long time (50 p.i. is almost 5 hours) still have a significant effect upon the current observation.

Long and Short-range dependencies are quantified using the Hurst parameter, H. This number is named for the hydrologist who researched annual floodings of the Nile River, and helped to develop some of the mathematics to deal with LRD series[14]. 'H' is a real number index of self-similarity, yet $0<H<1$ is sufficient for most practical purposes [4]. Series whose dependencies diminish quickly have Hurst parameters, $0<H<0.5$, and are known as Short-range Dependent (SRD). Those series wherein the present observation is strongly related to far distant lags in the series are LRD, and have Hurst parameters, $0.5<H<1$. An uncorrelated random series (e.g.- as calculated in MATLAB with 'randn.m') will have a Hurst parameter of $H=0.5$.

Self-similar processes were introduced by Kolmogorov and then reintroduced to statisticians by Benoit Mandelbrot, who progressed the theory toward an understanding of fractals (or fractional dimensional objects). A geometrically self-similar object has parts that resemble the shape of the whole when magnified[15]. Fractals can exhibit exact (or approximate)

[14] See [4] for more information about H. E. Hurst.
[15] [20] includes a good discussion of self-similarity with visual examples in the first chapter.

self-similarity, meaning that subsections of a picture resemble the whole (exactly or approximately).[16]

Reference [2] defines self-similarity as demonstrated in a stochastic process, Y_t, with continuous time paramenter t. "Y_t is called self-similar with self-similarity parameter H, if for any positive stretching factor c, the rescaled process with time scale ct, $c^{-H}Y_{ct}$ is equal in distribution to the original process Y_t." Thus, self-similar processes exhibit statistical self-similarity, which means that scaled versions of the process are *equal in distribution* to each other.

The Linear Stationary Models, AR(p), MA(q) and ARMA(p,q) discussed in Appendix C, are insufficient to accurately model the nonstationary (let alone LRD) processes.[17] The ARIMA model handles nonstationarity by introducing an integral number of zeros along the unit circle.

$$\phi(B)(1-B)^d X_t = \psi(B)\varepsilon_t \tag{4-1}$$

where B is the backshift operator, $Bz_t=z_{t-1}$. If d=0, then (4-1) becomes the ARMA(p,q) model. When $d \neq 0$, $(1\text{-}B)^d X_t$ may still be replaced by $\widetilde{X}_t$, and is again in the general ARMA(p,q) form. Thus, although the observed process X_t is nonstationary, $\widetilde{X}_t$ may be stationary. [6] refers to homogeneous non-stationary processes as those processes which do not quickly become unbounded. Transforming an ARMA(p,q) process into an ARIMA(p,d,q) process is akin to adding a pole to the unit circle of a given transfer function. If this pole were instead added outside of the unit circle, then the process would quickly become unbounded.

---

[16] See [4] for more discussion, as well as extensive references to Mandelbrots papers. Also, astronomy.swin.edu.au/~pbourke/fractals/selfsimilar/ gives a straightforward and pictoral explanation of the degrees of self-similarity.

[17] Forecasting the detrended byteout series using the Hannan-Rissanen algorithm (see Chapter Five) led to higher MSE ($MSE=1.0303\text{x}10^{12}$) than either of the SRD forecasts presented in this research. AR(50) and ARMA(15,15) models of the detrended byteout series were less capable of modeling the bursty sequence. Plotting the forecast over original series reveals a continual lag in the forecast.

[4] points out that $(1-B)^d$ may be calculated using the Binomial Theorem (from Calculus) and is defined as

$$(1-B)^d = \sum_{k=0}^{d} C_d^k (-1)^k B^k \tag{4-3}$$

with the binomial coefficients

$$C_d^k = \frac{d!}{k!(d-k)!} = \frac{\Gamma(d+1)}{\Gamma(k+1)\Gamma(d-k+1)} \tag{4-4}$$

where $\Gamma(\ )$ is the gamma function. So, although d is limited to integer multiples in the ARIMA(p,d,q), the gamma function allows for any real value of d.

Long-range dependent processes involve fractional dimensions, like fractals. Thus, modeling an LRD process requires non-integer values of d. Since the gamma function can receive non-integer inputs, [4] presents the fractionally-integrated ARIMA, or FARIMA(p,d,q) model, to evaluate LRD sequences. The LRD sequence, X_t, may be transformed into the stationary sequence, $\widetilde{X}_t$, using the operation in (4-3).

$$d=H-1/2 \tag{4-5}$$

LRD processes have a range, $0<d<1/2$, which is then used to calculate the binomial coefficients of Equation (4-4).

Self-Similarity in the CGDN+ data

One of several methods presented in [4] for calculating the Hurst parameter is to find the slope of the Variance of Samples plot. This method involves calculating the progressive mean of groups of 5, 10, 15... 500 samples of the data set under analysis. Each series of means is examined for its variance, and the variances are plotted in a log-log plot similar to Figure 4-1.

$$H = 1 - \frac{\alpha}{2} \tag{4-6}$$

The variance of sampled groups decreases as the sample size increases (slope, $-\alpha=0.6642$). A slowly decreasing mean ($-\alpha<1$) indicates LRD.

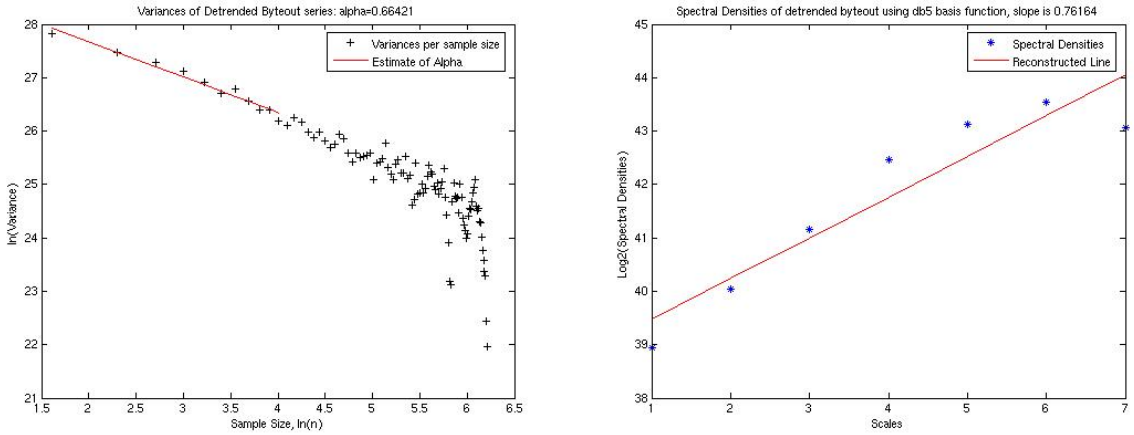


FIGURE 4-1: HURST VALUE CALCULATED FOR DETRENDED BYTEOUT SERIES BY: (LEFT) LOG-VARIANCE METHOD, AND (RIGHT) WAVELET METHOD.

The performance of both Log-Variance and Wavelet methods may be verified by their results for the synthetic traffic sequence generated in Appendix B. This sequence is created to have H=0.9 as described in [24]. Both plots indicate correctly that the synthetic sequence is LRD with a Hurst parameter near 0.9 (see Figure 4-2).

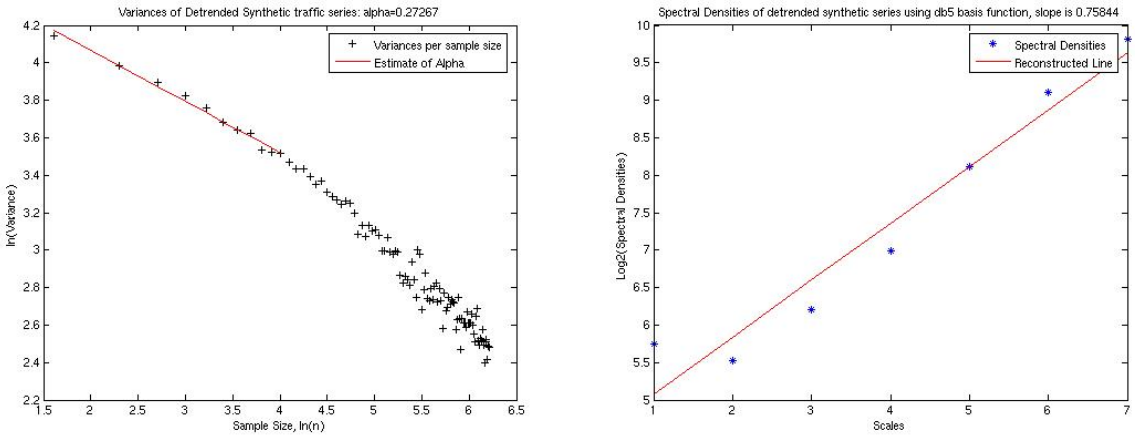


FIGURE 4-2: HURST CALCULATIONS OF SYNTHETIC TRAFFIC SEQUENCE BY (RIGHT) LOG-VARIANCE METHOD AND (LEFT) WAVELET METHOD.

A quick check of the byte transfers from CGC BEAR reveals similar long-range dependencies, with $H \geq 0.8$. This finding suggests that all of ship-shore internet traffic will be LRD if they are detrended in a way that maintains their correlations. With this slight searching into ship-shore dependencies, it is not clear that all ships share a similar value of correlation after detrending. If a generic traffic model is desired for all ships, this will require more research. As far as this research leads, correlations must be tested for each ship and each sequence individually to provide accuracy. Also, the traffic to/from a ship varies so much across months of operation as to suggest that a general model for a single ship may quickly become obsolete.

Findings in Table 4-3 show a high variance in Hurst parameter calculations for particular time series. In many cases, the Wavelet Method predicts Hurst parameter as $H_\beta > H_\alpha + 0.2$! Bytein and packin series calculations for the Hurst parameter appear particularly high. Table 4-3 also demonstrates that the wavelet method of H estimation is indeed resistant to periodic and linear trends found in real data. That is, the H estimates of original and detrended series are almost unchanged for the Wavelet method, but these same estimates from the Sample Variances method change by about 0.1.

	Log-Variance Method, $\hat{H}$	Wavelet Method, $\hat{H}$
Byteout series	.8355	Mean($\hat{H}$)=0.9811 $\sigma_{\hat{H}} = 0.0084$
Differenced byteout	.7183	Mean($\hat{H}$)=0.9658 $\sigma_{\hat{H}} = 0.0132$
Detrended byteout	.7213	Mean($\hat{H}$)=0.9716 $\sigma_{\hat{H}} = 0.0107$
Filtered byteout	.6679	Mean($\hat{H}$)=0.9767 $\sigma_{\hat{H}} = 0.0185$
Bytein series	.8752	Mean($\hat{H}$)=1.1732 $\sigma_{\hat{H}} = 0.0169$
Differenced bytein	.7900	Mean($\hat{H}$)=1.1866 $\sigma_{\hat{H}} = 0.0147$
Detrended bytein	.8117	Mean($\hat{H}$)=1.1749 $\sigma_{\hat{H}} = 0.0167$
Filtered bytein	.8475	Mean($\hat{H}$)=1.1484 $\sigma_{\hat{H}} = 0.0323$
Packout series	.8561	Mean($\hat{H}$)=0.9768 $\sigma_{\hat{H}} = 0.0129$
Differenced packout	.7284	Mean($\hat{H}$)=0.9634 $\sigma_{\hat{H}} = 0.0109$
Detrended packout	.7349	Mean($\hat{H}$)=0.9660 $\sigma_{\hat{H}} = 0.0121$
Filtered packout	.6682	Mean($\hat{H}$)=0.9681 $\sigma_{\hat{H}} = 0.0160$
Packin series	.8692	Mean($\hat{H}$)=1.0412 $\sigma_{\hat{H}} = 0.0150$
Differenced packin	.7421	Mean($\hat{H}$)=1.0370 $\sigma_{\hat{H}} = 0.0102$
Detrended packin	.7605	Mean($\hat{H}$)=1.0335 $\sigma_{\hat{H}} = 0.0119$
Filtered packin	.8349	Mean($\hat{H}$)=1.0255 $\sigma_{\hat{H}} = 0.0212$

TABLE 4-3: H-PARAMETER ESTIMATES FOR CGDN+ DATA.

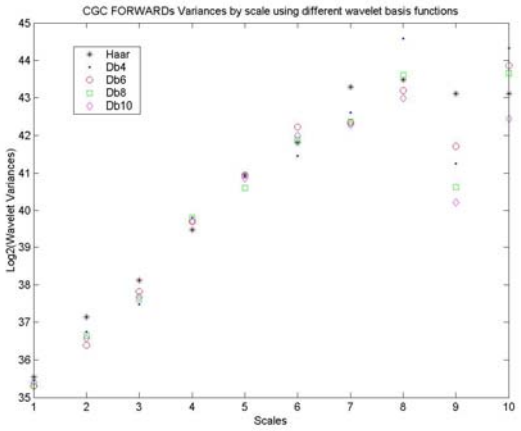


FIGURE 4-4: VARIANCES BY SCALE OF CGC FORWARD USING SEVERAL WAVELET BASES.

[2] mentions that real internet series are sometimes caused by multifractals. This research varied the range of scales analyzed in such a way as to produce the H estimate that would remain consistent with wavelets of increased vanishing moment. Figure 4-4 provides some additional evidence that the scale-variance plot of real signals can have break points. Obviously, the variance at each scale in Figure 4-4 do not all adhere to a linear line. Scale-variance plots of the CGDN+ data do not follow a consistent line for all scales and across all wavelet bases, but subsets of the range of plotted scales do consistently follow a linear plot. This finding seems to indicate the presence of multifractals in the CGDN+ data. A more thorough analysis of this finding is left for future work. Scales 1-6 of byteout and packout seem to follow a linear trend with great consistency. Also, scales 1-6 of bytein and packin adhere closely to a linear trend, but this trend has $H_\beta>1$

5 One-step Ahead Forecasts of Byteout

Forecasts of the detrended byteout series are obtained in this chapter using H-estimates from both the log-variance and wavelet methods. The H parameter estimations from Table 4-3 are translated into d using (4-5). Byteout is detrended using differencing by period and then evaluated for H, then d.

	H	d
Sample Variance method	0.7183	0.2183
Wavelet Method	0.9658	0.4658

TABLE 5-1: PARAMETER ESTIMATES OF BYTEOUT

Several methods exist for calculating the parameters p, q of an ARMA(p,q) process. The Hannan-Rissanen algorithm provides a recursive method for estimating ($\hat{\phi}, \hat{\theta}$), and is chosen in this research for its relative simplicity. Hannan-Rissanen algorithm assumes that the ARMA(p,q) model is invertible and stable. We can employ this algorithm by first reducing detrended byteout series to an SRD sequence, as proposed in [4]. Binomial coefficients are generated using (4-4), and the Long-memory process, X_n multiplied by (4-3) will create the short-memory process, $\tilde{X}_n$. This process is conducted using 'x_tilda.m' from Appendix D. Since $\tilde{X}_n$ is SRD, an ARMA(p,q) model may be used to represent the series.

As discussed in Appendix C, the ARMA model is built from two subordinate models: Auto-regressive (AR(p)) and Moving-average (MA(q)). The first step in Hannan-Rissanen Algorithm is to find a linear regression for the SRD $\tilde{X}_n$ series with a high order AR(p) model, for instance AR(50). The AR(p) assumes that a future output can be determined from past inputs that are individually weighted. Rewriting (2-6) using matrix algebra:

$$Z = Hx + e : \begin{bmatrix} Z_{51} \\ Z_{52} \\ \dots \\ Z_{2030} \end{bmatrix} = \begin{bmatrix} X_{50} & X_{49} & \dots & X_1 \\ X_{51} & X_{50} & \dots & X_2 \\ \vdots & \vdots & \ddots & \vdots \\ X_{2029} & X_{2028} & \dots & X_{1980} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{50} \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_{50} \end{bmatrix} \tag{5-1}$$

where tildas have been dropped for convenience. **Z** models an AR(50) estimate of $\tilde{X}_n$, and is comprised of observations 51-2030 of $\tilde{X}_n$. H is also comprised of sequential values of $\tilde{X}_n$. Only **x** (comprised of ϕs) is unknown. ϕs may be calculated based on Least Squares estimation using the equation:

$$\hat{x} = (H^T H)^{-1} H^T Z \tag{5-2}$$

where Z is $\begin{bmatrix} X_{51} \\ X_{52} \\ \dots \\ X_{2030} \end{bmatrix}$. Once the ϕs have been calculated from (5-2), $\hat{Z} = H\hat{x}$ will obtain the AR(50) estimate to $\tilde{X}_n$. Notice that the AR(p) regression sacrifices the first p values of a sequence, $\{z_1,\dots,z_p\}$ to calculate the output sequence, $\{\hat{z}_{p+1},\dots,\hat{z}_n\}$. Errors between $\tilde{X}_n$ and the AR(50) estimate $\hat{Z}$ are defined as

$$\hat{\varepsilon}_n = X_n - \hat{Z}_n \tag{5-3}$$

These errors are frequently called 'innovations' and will be used in the second step to generate a better (ARMA) model. Notice from Figure 5-2 that these errors are as large as the original series.

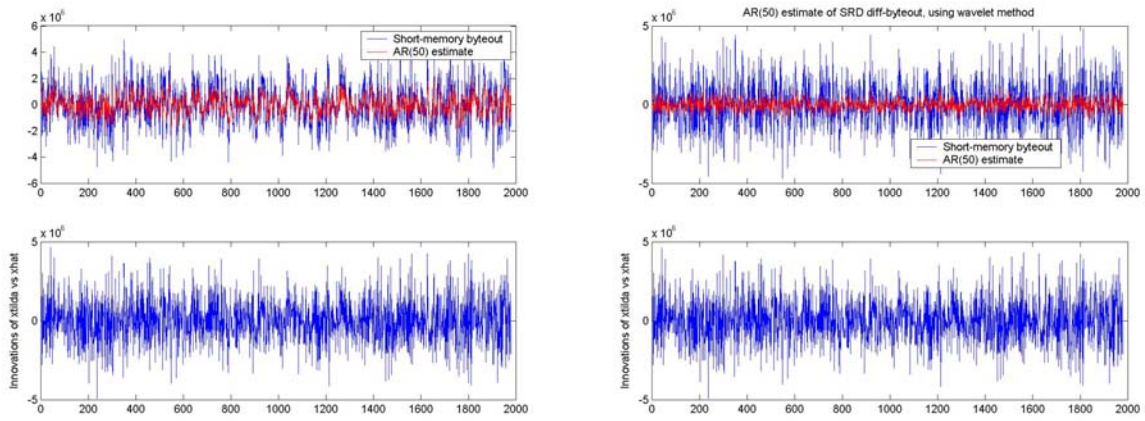


FIGURE 5-2: AR(50) REGRESSIONS OF DIFF_BYTEOUT WHERE $\tilde{X}_n$ IS CREATED USING (RIGHT) SAMPLE VARIANCE ESTIMATE, AND (LEFT) WAVELET METHOD. BOTTOM PLOTS ARE THE ERRORS OF EACH MODEL.

The Moving Average model assumes that the current value is made of weighted inputs, which may consist of the errors $\hat{\varepsilon}_n$ between $\hat{Z}$ and the $\tilde{X}_n$ series. A better estimate $\hat{Z}$ of the original signal is obtained through a direct sum of the range space for $\tilde{X}_n$ and $\hat{\varepsilon}_n$ than would be available through either the AR(p) or the MA(q) model alone.

An ARMA(15,15) model is now created by altering (5-1)

$$Z = H\hat{x} + e : \begin{bmatrix} X_{16} \\ X_{17} \\ \dots \\ X_{1980} \end{bmatrix} = \begin{bmatrix} X_{15} & \dots & X_1 & \hat{\varepsilon}_{15} & \dots & \hat{\varepsilon}_1 \\ X_{16} & \dots & X_2 & \hat{\varepsilon}_{16} & \dots & \hat{\varepsilon}_2 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ X_{1979} & \dots & X_{1965} & \hat{\varepsilon}_{1979} & \dots & \hat{\varepsilon}_{1965} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_{15} \\ \theta_1 \\ \vdots \\ \theta_{15} \end{bmatrix} + \begin{bmatrix} \varepsilon_{16} \\ \varepsilon_{17} \\ \vdots \\ \varepsilon_{1980} \end{bmatrix} \tag{5-4}$$

After writing the sequence in this form, the series of ϕs and θs may be calculated from (5-2). Finally, the one-step ahead forecast of the SRD sequence $\tilde{X}_n$ is created from

$$\hat{Z} = H_{ARMA}\hat{x} \tag{5-5}$$

Figure 5-3 plots the poles and zeros of the transfer functions estimated using (5-4). Notice that poles and zeros are both inside and outside of the unit circle, meaning that the SRD sequence $\hat{Z}_n$

is created using some unstable poles. Figure 5-3 presents sufficient evidence that the Hannan-Risannen algorithm creates an unstable ARMA(p,q) model of $\tilde{X}_n$. The problem of finding an algorithm that creates a stable ARMA(p,q) model is left for future work.

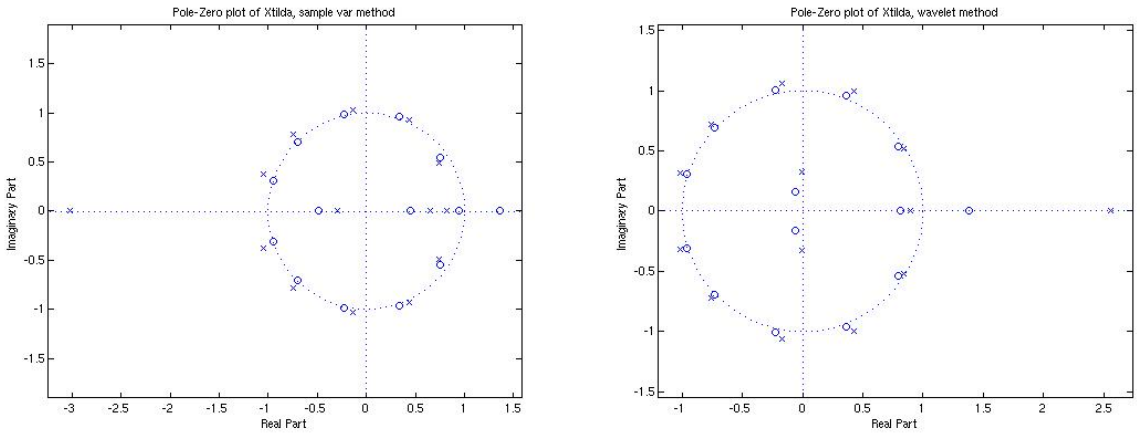


FIGURE 5-3: POLE-ZERO PLOTS OF ($\hat{\phi}, \hat{\theta}$) AS CALCULATED USING (LEFT) SAMPLE VARIANCE METHOD OR (RIGHT) WAVELET METHOD.

Since the ARMA model is a linear model, it cannot produce the spiky data common to internet traffic. Therefore, the prediction sequence $\hat{Z}$ has LRD reinserted using the FARIMA (p,-**d**,q) procedure. Results of this one-step ahead forecast of LRD series X_n are displayed in Figure 5-4.

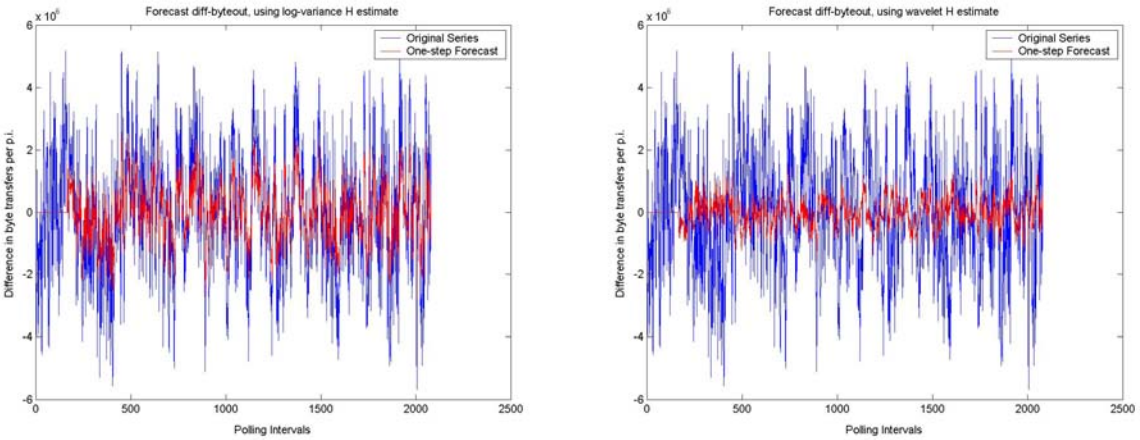


FIGURE 5-4: ONE-STEP AHEAD FORECASTS OF DIFF_BYTEOUT WHERE $\tilde{X}_n$ IS CREATED USING (RIGHT) LOG-VARIANCE METHOD, AND (LEFT) WAVELET METHOD.

The zeroed portion of each forecast in Figure 5-4 has been sacrificed in the two-step process in order to find ϕs and θs that could represent diff_byteout. It appears in Figure 5-4 that the Wavelet forecast isn't as reactive as the Sample Variance forecast. These two forecasts are compared using the Mean Square Error of each non-zero estimate.

$$MSE = \frac{1}{n}\sum_{n}(X - \hat{X})^2 \tag{5-6}$$

The results, displayed in Table 5-5, indicate that the Wavelet method has produced a Hurst-estimate that contributed to a significantly more accurate forecast of the diff_byteout series.

Method of H-estimation	**d**	**MSE**
Log-Variance	0.2183	3.5868e9
Wavelet	0.4658	1.2485e8

TABLE 5-5: RESULTS OF ONE-STEP AHEAD FORECAST COMPARED USING MEAN-SQUARE ERROR.

The square root of MSE may be compared with byteout to judge the forecast error of either method. The forecast error is admittedly large, and any other modeling algorithm should seek to improve upon this forecast error while creating a stable ARMA(p,q) model.

6 Conclusions and Future Research Work

This paper is the first written to model the traffic crossing over the CGDN+ ship-shore connection, so it seems appropriate to begin with few practical contributions of this work. First, cutters demand more information than they produce, so less bandwidth is needed in the connection leading from cutter to the Satellite Earth Station than for the reciprocal connection. Second, results of this paper indicate that a strong daily cycle exists, especially in the byteout, packout and packin series of CGDN+ data. Peak usage of the CGDN+ connection occurs about 0100, followed by minimal usage occurring at about 0930. These practical lessons about the ship-shore connection can be used to more optimally administer the connection. Also, router optimization will be possible once high time-resolution traces of the connection are available. Such traces may be collected locally at a cutter to avoid burdening the CGDN+ with too many polling queries.

In addition to practical insights to the CGDN+ ship-shore connection, this research has made the following contributions to time series analysis.

Wavelet H-estimator. This paper has found that the wavelet method of H estimation proposed in [1] is resistant to periodic and linear trends found in the analysis of real data. The wavelet estimator was shown capable of detecting LRD over shorter data series than the Sample Variance estimator. The wavelet-based H-estimate outperformed the Sample Variance H-estimate in an otherwise similar set of forecasts.

Decorrelating influence of First Differences. First difference is a common operation for detrending (mentioned in [6] and [7]); however, it must be used with caution when the results will be used in time series analysis. Specifically, this research has found that first differences decorrelates AR(1) processes where $0.65<\phi_1<1.35$ and correlates AR(1) processes with $\phi_1<-0.25$.

Highly-aggregate time series. This research may prove helpful to network administrators of military and commercial networks that are polled at large time intervals (>1 second). When highly aggregated, the CGDN+ maintains LRD and even adheres closely to a Gaussian distribution. These findings lead to a straightforward analysis, and network administrators may benefit from one-hour ahead forecasts that are possible with the techniques herein presented.

Future research concerning the CGDN+ ship-shore connection should seek to build a generally applicable traffic model for cutter traffic. Realistic H estimates of bytein/packin will be necessary to obtain this general model and may be obtained by exploring the presence of multifractals in the series or determining the effect of correlated wavelet coefficients observed in bytein's spectral analysis. Also, the possibility exists that Var($\hat{H}$) grows as H$\rightarrow$1, which would be more generally applicable if proven. Also, continuing research would include reconciling the findings that byteout and packout are LRD, but have only one significant partial correlation found by the Yule-Walker algorithm.

Wavelet methods have proven effective for calculating robust estimates of the Hurst parameter of self-similarity for use in classical methods of time series modeling (e.g.- ARIMA(p,d,q)). It is also possible that wavelets transforms will provide robust estimates of p,q as used in an ARMA(p,q) model. In any case, other modeling algorithms should be explored to improve the forecast error displayed in Table 5-5, and since the Hannan-Rissanen algorithm produces an ARMA(p,q) model that has an unstable pole-zero plot (see Figure 5-3).

References

[1] Abry, Patrice and Darryl Veitch. "Wavelet Analysis of Long-Range-Dependent Traffic." IEEE Transactions on Information Theory 44 (January 1998): 2-15.

[2] Abry, Patrice, Richard Baraniuk, Patrick Flandrin, Rudolf Riedi, and Darryl Veitch. "Multiscale Nature of Network Traffic." IEEE Signal Processing Magazine (May 2002): 28-46.

[3] Anton, Howard. Calculus with Analytic Geometry: Brief Edition. 4th ed. New York: John Wiley & Sons, Inc, 1992.

[4] Beran, Jan. Statistics for Long Memory Processes. New York: Chapman & Hall, 1994.

[5] Bernstein, Peter L. Against The Gods: The Remarkable Story of Risk. New York: John Wiley & Sons, Inc, 1996.

[6] Box, George E. P., Gwilym M. Jenkins and Gregory C. Reinsel. Time Series Analysis: Forecasting and Control. 3rd ed. Upper Saddle River, New Jersey: Prentice-Hall International, Inc, 1994.

[7] Brockwell, Peter J., and Richard A. Davis. Introduction to Time Series and Forecasting. 2nd ed. New York: Springer, 2002.

[8] Hubbard, John H. "Chaos." A First Course in Differential Equations. 5th ed. Comp. Dennis G. Zill. Boston: PWS Publishing Company, 1993. 207-10.

[9] Jackson, Leland B. Signals, Systems, and Transforms. Reading, Massachusetts: Addison-Wesley Publishing Company, 1991.

[10] Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems." Transactions of the ASME-Journal of Basic Engineering 82 (Series D, 1960): 35-45.

[11] Leland, W.E., and Wilson, D.V. "High time-resolution measurement and analysis of LAN traffic: implications for LAN interconnection." Proceedings of the IEEE INFOCOM'91, Bal Harbor, FL (1991): 1360-1366.

[12] Leland, Will E., Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. "On the Self-Similar Nature of Ethernet Traffic (Extended Version)." ACM/ IEEE Transactions on Networking, Vol. 2, pp. 1-15, 1994.

[13] Leon-Garcia, Alberto. Probability and Random Processes for Electrical Engineering. 2nd ed. India: Pearson Education, Inc, 2004.

[14] Lyons, Richard G. Understanding Digital Signal Processing. 2nd ed. Upper Saddle River, New Jersey: Prentice Hall, 2004.

[15] Ma, Sheng, and Chuanyi Ji. "Modeling Heterogeneous Network Traffic in Wavelet Domain," IEEE/ACM Trans. Networking. 9 (October 2001): 634-649.

[16] Mallat, Stéphane. Wavelet Tour of Signal Processing. San Diego: Academic Press, 1998.

[17] Maybeck, Peter S., Stochastic Models, Estimation, and Control. Vol. 1, Academic Press, Inc, 1979.

[18] "Model." Random House Webster's College Dictionary. United States of America: McGraw-Hill Ed. 1991.

[19] Newton, Joseph H., TIMESLAB: A Time Series Analysis Laboratory. Pacific Grove, CA: Advanced Books and Software, 1996.

[20] Park, Kihong, and Walter Willinger, eds. Self-Similar Network Traffic and Performance Evaluation. New York: John Wiley & Sons, Inc, 2000.

[21] Percival, Donald B., and Andrew T. Walden. Wavelet Methods for Time Series Analysis. Cambridge, United Kingdom: Cambridge Press, 2000.

[22] Smith, Steven W. The Scientist and Engineer's Guide to Digital Signal Processing. San Diego, California : California Technical Publishing, 1997.

[23] Strang, Gilbert, and Truong Nguyen. Wavelets and Filter Banks. Rev. ed. Wellesley, MA: Wellesley-Cambridge Press, 1997

[24] Taqqu, Murad S., Walter Willinger and Robert Sherman. "Proof of a Fundamental Result in Self-Similar Traffic Modeling." ACM SIGCOMM Computer Communication Review, Vol. 2, 1997: 5-23.

[25] Wang, Xin, Xiuming Shan. "A Wavelet-Based Method to Predict Internet Traffic." IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions, Vol. 1, 2002: 690-694.

[26] Zill, Dennis G. A First Course in Differential Equations. 5th ed. PWS Publishing Company: Boston, 1993.

Appendix A: A Discussion of the Motivation for Forecasting

Forecasting finds daily application in our modern life. Many people plan their day or their fortune on weather predictions or stock market forecasts. Modern techniques make forecasts possible, within some limits. These forecasts are based upon past outcomes of a given process and a thorough understanding of the mechanics and trends behind the data points. If the process generating an outcome (or 'observation') can be known thoroughly, then future outcomes can be predicted. Weather forecasts serve as sufficient reminder that every forecast has some degree of uncertainty associated with it.

It may be surprising that gambling was the catalyst for this flood of knowledge that impacts nearly every area of modern human achievement. Yes, the fortunes won and lost in the gambling halls of Renaissance Europe provided ample motivation to understand and wield the uncertain. Gamblers, desiring to improve their odds of winning, sought after both amateur and professional mathematicians to help them lay the foundations of Probability Theory. Revolutionary thinkers like Cardano and Pascal, interpreted games of chance and derived laws that govern uncertain events… within some limit [5].

Some probabilistic methods assume independence between events. Gambling, for example, allows the assumption of independence in many games. This research, however, takes advantage of data dependencies. In a completely random (i.e.- uncorrelated) sequence, you can do little better than to identify the mean, the variance, some other moments of the data, and any existing trend the series may follow. These statistics are critical to identifying the data's distribution, but the process mean makes for a poor forecast. Fortunately, strong correlations are

found in the Coast Guard data series. These correlations can be used to narrow the future somewhat, creating a more reliable forecast[18].

Correctly forecasting a time series of internet traffic can allow engineers to predict the aggregate demand on servers several hours in advance. Quality of Service analysts can use expected demand amongst other considerations to track a network's ability to accept more workload. Also, once the process creating a given stochastic process is understood it can be modeled to allow engineers flexibility in predicting the impact of system changes. This paper endeavors to understand the process creating byte arrivals to Coast Guard Cutters (or ships), and then develop an accurate model for that process. These methods have led to considerably increased understanding of the CGDN+, and this work opens the doors for several more areas of research.

[18] The essence of probability (and forecasting) is knowing what can be known, and the limits of what can't yet be known. Says Kenny Rogers' famous gambler, "You've got to know when to hold 'em; Know when to fold 'em; Know when to walk away; Know when to run…"

Appendix B: Comparing CGDN+ to the Bellcore traces

Reference [2] begins with an excellent (and readable) overview of the complexities found while analyzing internet traffic. To summarize, [2] states that internet traffic is made complex by: geography, offered traffic and time burstiness. CGDN+ seems to take geographic complexity to the next level. The INMARSAT connection adds a degree of latency not present in other networks that are cited in current time series research, like ping times to cutters stationed in the Pacific Ocean that can take up to 3 seconds to turn.

As for its offered traffic, CGDN+ ship-shore traffic consists of all sizes of packets, ranging in size from a few kilobytes to 1400 kilobytes, which is the Maximum Transmission Unit (MTU) given the TCP/IP and 'tunneling' used in the connection. Figure B-1 displays a packet distribution that is bimodal, with most packets about 100 bytes/packet, with a significant number of packets also being close to the MTU. That portion of packets which is about 100 bytes/packet may be control data (e.g.-ACKnowledgements, etc) vital to the proper flow of network communications and relatively small in size. Since the data crossing the 'ship-shore' connection consists of packets of various sizes, it is known as heterogeneous traffic. In this regard, the CGDN+ data seems to be as complex as the Bellcore research (see reference [11]). One key difference is that available bandwidth for the CGDN+ (64/128 kbps) is quite restrictive when compared to Bellcore's recordings (1.544 Mbit/second).

FIGURE B-1: RELATIVE FREQUENCY OF PACKET SIZES (IN BYTES) CROSSING THE INMARSAT CONNECTION. COLLECTED BETWEEN 9 JUNE AND 16 JUNE 2005 BY NETVCR.CAMSLANT.CGDN.USCG.MIL.

Time burstiness refers to the tendency for internet data to have a few highly active periods surrounded by many relatively inactive periods. Chapter Three of this research finds the CGDN+ Gaussian distributed, instead of Weibullian or Log-Normal as are other internet traces. The difference in CGDN+ time burstiness is attributable to the aggregate nature of the series, which finding has been reported elsewhere without as much elaboration [2]. With polling intervals of 5'40", the CGDN+ data cannot be used to improve router performance (the motivation behind quantifying and modifying time burstiness), yet it displays the daily trends typical of real-life internet series (compare with most aggregate form of Bellcore data as displayed in [12]).

The famous Bellcore 'traces'[19] are representative of the internet series analyzed in current research. They consist of high time-resolution recordings of Ethernet traffic gathered at Bellcore's Morris Research and Engineering Center during August 1989 through January 1990. Packets in the Bellcore traces are timestamped with an accuracy of at least 100 μs, as compared with the CGDN+ polling method which only summarizes traffic activity at 5'40" intervals. The

[19] Trace: "The marking made by certain instruments, e.g. a seismograph"; as defined in The New Lexicon Webster's Encyclopedic Dictionary of the English Language, Deluxe Edition; Lexicon Publications, Inc.: New York 1989.

laboratory monitored in the Bellcore research was connected to the outside world through a single T1 (1.544 Mbit/second) line, and was completely unrestricted (not a realistic assumption today) [20].

After making the high time-resolution recordings now known as 'Bellcore traces' some of these researchers published [24] to provide a rigorous mathematical demonstration of the mechanism causing self-similarity (or LRD) in internet traces. They described this mechanism

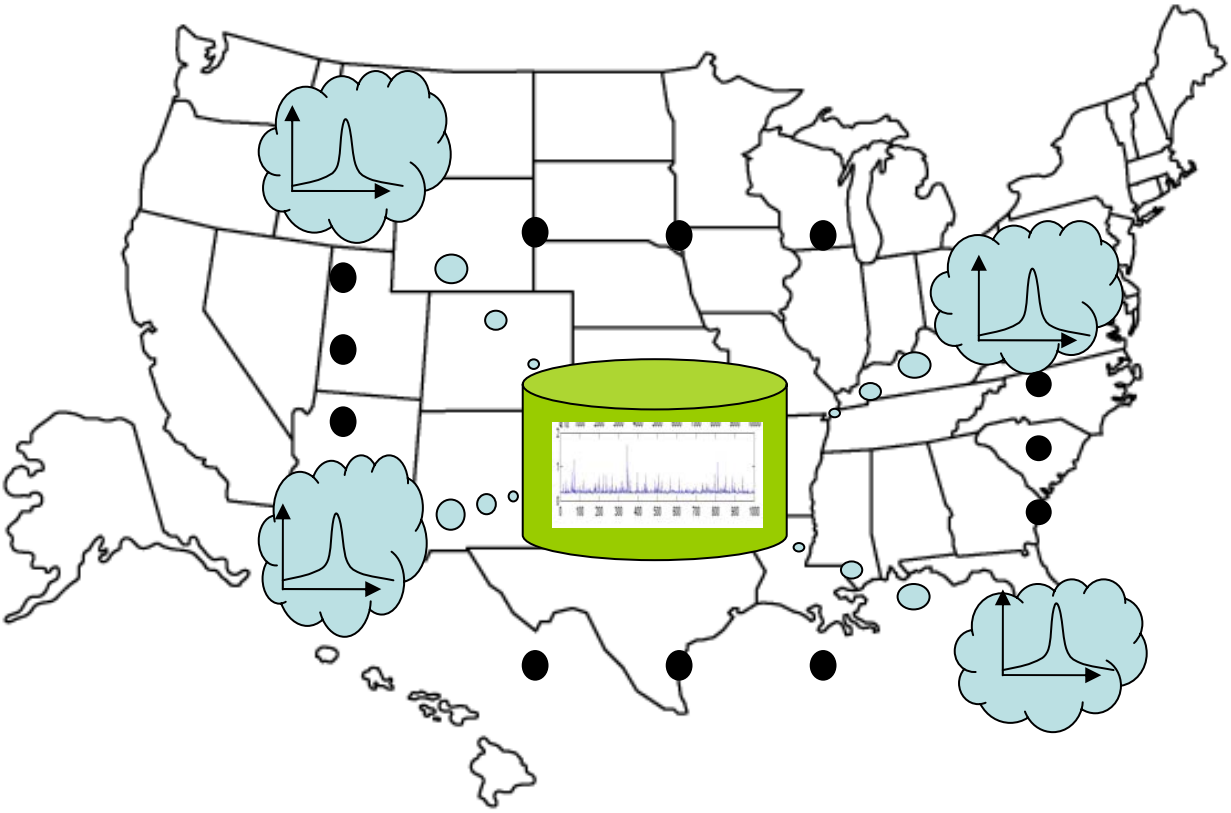


FIGURE B-2: HIGH VARIANCE TRAFFIC FROM MANY POSSIBLE SOURCES CAUSES LRD TRAFFIC.

the Joseph Effect, and proved that it was caused by many independent sources each exhibiting the 'Noah Effect'. Noah Effect hints at the tendency that any one source of internet traffic will send nothing for a very long time, and then suddenly become very active. Combining many

[20] see technical details of this study in [11].

sources that exhibit the Noah Effect causes a self-similar series, or the Joseph Effect, as it is known (see Figure B-2).

The mathematical model proposed in [24] begins by defining a stationary binary time series {W(t), t≥0} known as the reward sequence. W(t)=1 signifies that a single source sends traffic to the receiving server, and W(t)=0 signifies that no traffic is sent during that interval. The length of the ON-periods are independent and identically-distributed (i.i.d.), those of the OFF-periods are i.i.d., and the lengths of the ON- and OFF-periods are independent. The superposition of M of these i.i.d. sources at time t is $\sum_{m=1}^{M} W^{(m)}(t)$. The aggregate cumulative packet counts in the interval $[0,T_t]$ are calculated by

$$W_M^*(Tt) = \int_0^{Tt} \left(\sum_{m=1}^{M} W^{(m)}(u) \right) du \tag{B-1}$$

The statistical behavior of the stochastic process $\{W_M^*(Tt),\ t\geq 0\}$ for large M and T depends on the distributions on the ON- and OFF-periods. The Noah Effect of the ON/OFF-periods of the individual source-destination pairs is an essential ingredient[21]. Fortunately, the distribution of the ON/OFF-periods is controlled by a single variable that is related to the Hurst parameter, H. That variable will be renamed γ in this thesis, but the relationship between H and γ, as presented in [24] remains

$$H = \frac{3-\gamma}{2} \tag{B-2}$$

This formula relates the Noah and Joseph effects, since a Pareto(γ) distribution with $1< \gamma <2$ represents each finite mean and infinite variance internet source. Appendix D includes MATLAB code that implements this method to create Figure B-3. Q-Q plots demonstrate that

[21] [20] includes a helpful explanation of the ON/OFF source method of LRD traffic generation. It also shows the process of adding ON/OFF periods in Figure 1.6.

the synthetic traffic is Gaussian distributed, even though its reward intervals lengths are determined by a Pareto distribution. It must be that the distribution interval lengths of W(t) do not determine necessarily the distribution of the aggregate cumulative series, as displayed in Figure B-3.

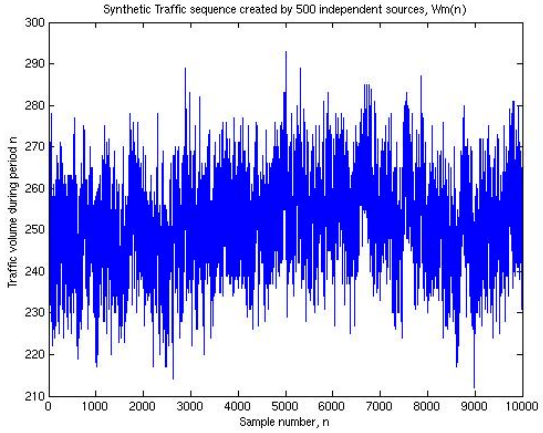


FIGURE B-3: SYNTHETIC TRAFFIC SEQUENCE CREATED BY 500 INDEPENDENT SOURCES, $W_M(N)$

Again, the series in Figure B-3 is not part of the Bellcore traces, but is a synthetic series possessing characteristics similar to the Bellcore traces. Figure B-4 shows a limited aggregation of the 'byteout', 'synthetic', and an AR(1) sequence. This aggregation is limited simply due to: 1) the large memory required compute a longer synthetic sequence, and 2) a maximum of 18,000 observations of byteout available (even in disjoint segments).

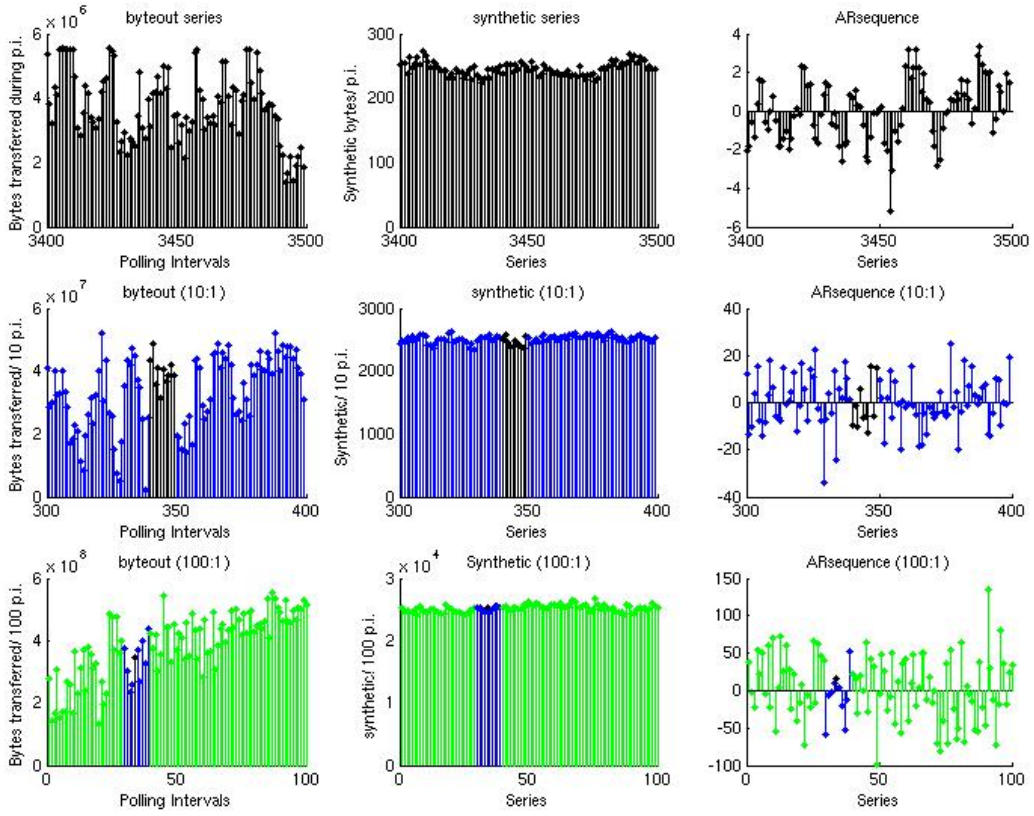


FIGURE B-4: COMPARISON OF (LEFT) 'BYTEOUT' SERIES, (CENTER) 'SYNTHETIC' SERIES AND (RIGHT) AN AR(1) SEQUENCE OVER THREE LEVELS OF AGGREGATION EACH.

Figure B-4 shows that all three series maintain their apparent distributions over three levels of aggregation. Figure B-5 will also reveal the autocorrelations and partial correlations of the synthetically generated 'Bellcore' trace. The autocorrelations in Figure B-5 (left) do not attenuate quickly, indicating an LRD series (compare with Figure 1-4).

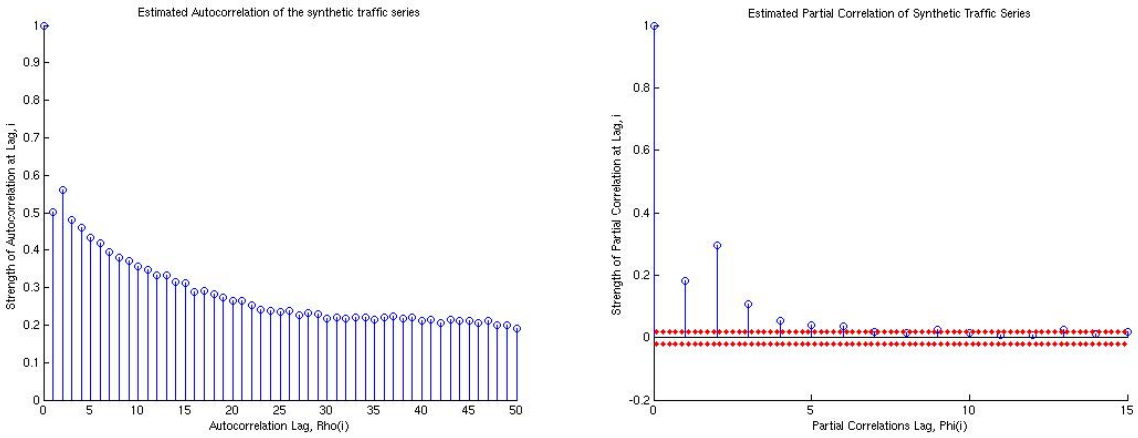


FIGURE B-5: AUTOCORRELATION AND PARTIAL CORRELATION FUNCTIONS OF SYNTHETIC TRAFFIC SERIES.

Appendix C: Primer on Linear Stationary Models

Some of the simplest time series models relate present observations to a linear combination of past and present outputs and inputs. Known as Linear Stationary Models, they are easiest to understand by examining their schematics (see Figure C-1). Gaussian-distributed white noise is input to the model and then processed with delays, gains and sums. The Moving Average (MA) model adds past inputs to the current input to produce an output. This is

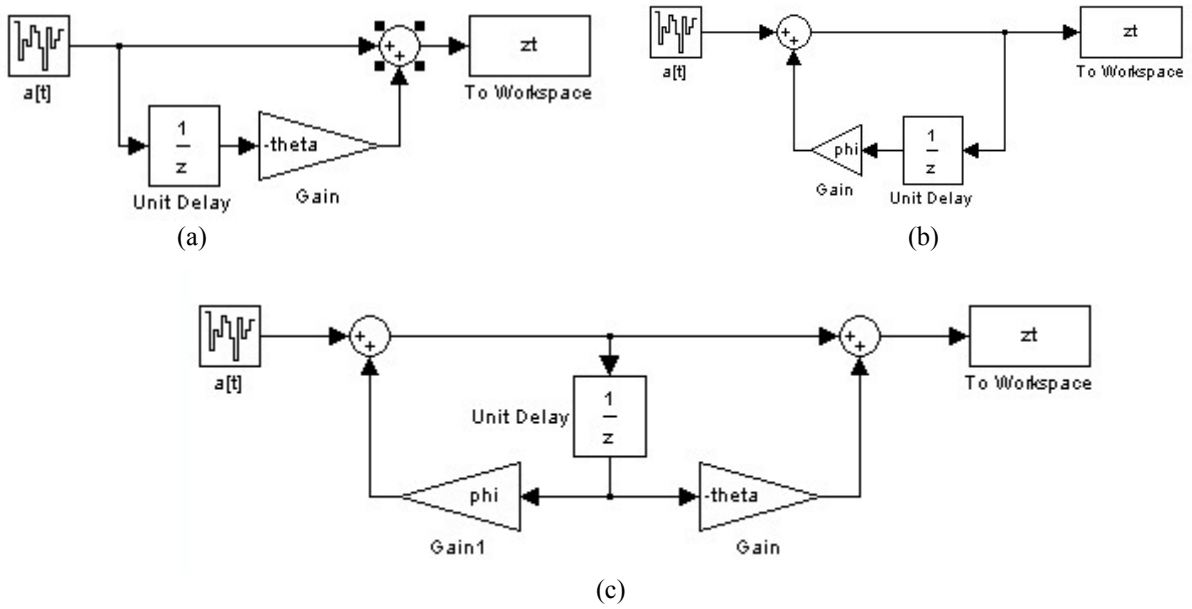


FIGURE C-1: SCHEMATICS OF (A) MOVING AVERAGE (MA) MODEL, (B) AUTOREGRESSIVE (AR) MODEL, AND (C) MIXED AUTOREGRESSIVE-MOVING AVERAGE (ARMA) MODEL.

sometimes called a "feed-forward" loop. Autoregressive (AR) Models add current input to past outputs in what is called a "feedback" loop. Finally, the Mixed Autoregressive-Moving Average (ARMA) Model combines both the feedback and feed-forward loops to produce an output which is the sum of current input, past input and past output. For each of these models, outputs are a linear combination of inputs and outputs, and this is why outputs are correlated with each other.

The schematics for Linear Stationary Models easily translate into mathematical expressions. Box, Jenkins and Reinsel provide a thorough mathematical analysis of Linear Stationary Models in [6]. The AR(1), or first-order Autoregressive Model can be represented as

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + a_t \tag{C-1}$$

where a_t is the input, ϕ_1 is the scaling factor, $\tilde{z}_t$ is the output, and $\tilde{z}_{t-1}$ is the previous output. The MA(1) difference equation is also easily obtained to be

$$\tilde{z}_t = a_t - \theta_1 a_{t-1} \tag{C-2}$$

where the input is added to a scaled version of the previous input to obtain the system's current output. Finally, the ARMA(1,1) model may be expressed mathematically as

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + a_t - \theta_1 a_{t-1} \tag{C-3}$$

These equations of the Linear Stationary Models should be reconcilable with the schematics in Figure C-1. They are known as difference equations because they relate the current output to the difference (and sum) of past outputs and past/current inputs.

The AR(1) can be implemented in a simple MATLAB routine. First, make an array of zeros. MATLAB has a function 'randn.m' that produces normally-distributed random numbers, making the input sequence very simple. A 'for' loop will add the random input to the scaled version of the latest output, and then save this as the current output. Like this:

```
Z=zeros(1,1000);
for i=1:999
Z(i+1)=phi1*Z(i)+randn(1);
end
plot(1:1000,Z)
```

This simple routine allows us to create a synthetic AR(1) series. Synthetic MA(1) and ARMA(1,1) processes may be generated as easily, adapting the middle line of code to implement

the appropriate formula, (C-2) or (C-3). This series may then be analyzed with the autocorrelation function introduced earlier, to show the dependencies between data points.

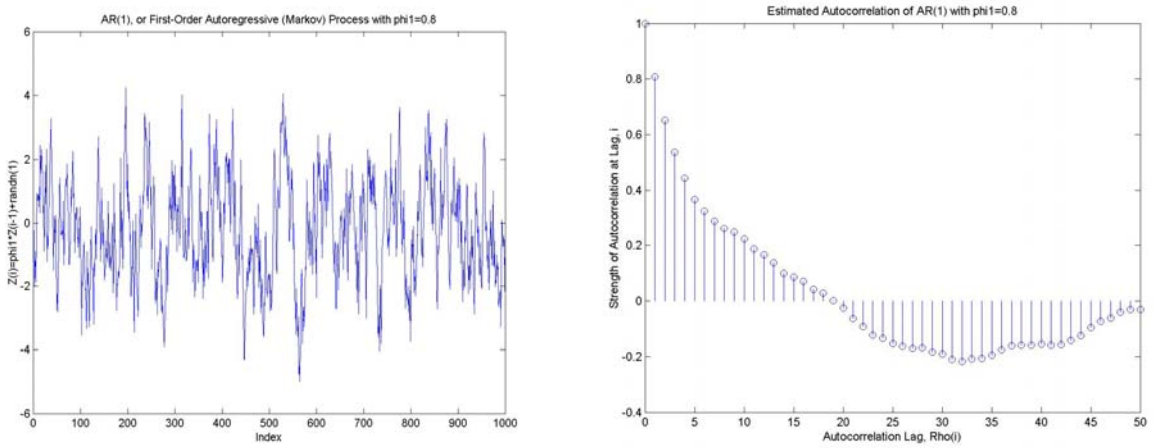


FIGURE C-2: (LEFT) SYNTHETIC AR(1) SERIES (ϕ_1=0.8), AND (RIGHT) AUTOCORRELATION FOR AR(1) SERIES.

The simulation program has allowed us to emphasize the model's assumptions. We have stated that every input is distributed similarly and this is now obvious, since the MATLAB 'for' loop accepts inputs from 'randn.m', a Gaussian-distributed random sequence. Although these inputs were uncorrelated, Figure C-2 (right) shows that the output sequence (from Figure C-2 [left]) is correlated. Next, we will manipulate the mathematical expression (C-1) for the AR(1) process to show that each system passes certain frequencies more easily than others.

Frequencies and the z-domain

We have asserted that the input sequence consists of all possible frequencies. The system, though, will not pass all frequencies equally; instead it will amplify some, while minimizing other frequencies. Since the input is white noise, any dips (zeros) or peaks (poles) in the output's frequency plot are caused by the system. This being the case, it should seem reasonable that the output is a result of the frequencies passed by the system. Box, Jenkins and Reinsel show this through a short proof showing that the autocorrelation function is mathematically related to the spectral density ([6], pp 44-45).

The way to determine which frequencies will be passed through the system is by finding the system's transfer function. Slight algebraic manipulation of (C-1) yields $\tilde{z}_t - \phi_1 \tilde{z}_{t-1} = a_t$, which may then be separated into $\tilde{z}_t(1-\phi_1 z^{-1}) = a_t$ where z^{-1} represents the delay operator [9]. $\tilde{z}_t z^{-1} = \tilde{z}_{t-1}$, $\tilde{z}_t z^{-2} = \tilde{z}_{t-2}$, and so on. There are two reasons why $\tilde{z}_t$ is used, instead of z_t. First, this symbolism is consistent with that chosen in [6]. This section deals with stationary series, which may still have a non-zero mean. Therefore, $\tilde{z}_t = z_t - \mu$. The second reason that we keep $\tilde{z}_t$ is that it should reduce confusion with the time delay operator, z^{-1}.

Finally, the transfer function representation we sought is:

$$H(z) = \frac{\tilde{z}_t}{a_t} = \frac{1}{1-\phi_1 z^{-1}} = \frac{B(z)}{A(z)} \tag{C-4}$$

where B(z) =1 and A(z) =$1-\phi_1 z^{-1}$. B(z) and A(z) are both polynomials of z. But, any z which causes A(z) to equal zero makes H(z) go to infinity. Therefore, in Figure C-3 (left), where ϕ_1=0.8, if z=0.8, then $\frac{0.8}{0.8} = 1$, and 1-1=0, causing the H(z)=∞. In this case, any number whose frequency is close to the root of A(z) will be passed more easily through the system. Figure C-3 (right) is called the system's magnitude response because it displays the magnitude of all frequencies passed by the system.[14],[22]

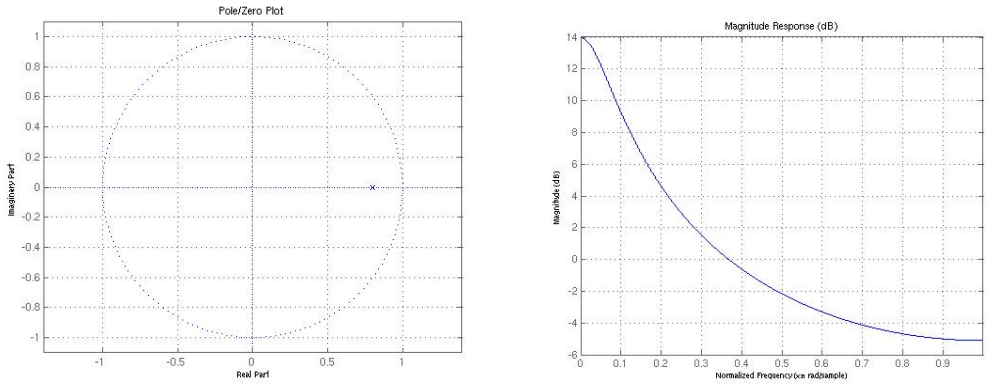


FIGURE C-3: (LEFT) POLE-ZERO DIAGRAM OF AR(1) PROCESS FOR ϕ_1=0.8, AND (RIGHT) THE CORRESPONDING MAGNITUDE RESPONSE OF THIS AUTOREGRESSIVE FILTER.

It may seem strange to equate numbers with frequencies, but here we find reason why complex numbers are important, and have their impact upon daily modern life. The work of Leonhard Euler demonstrated that all numbers have a magnitude and phase associated with them. Real numbers have only two phases possible, either 0° or 180° (corresponding to negative numbers). Complex numbers may have any other phase, including those of the real numbers, since the set of all real numbers is a subset of all complex numbers. [26] Figure C-3 (left) is a representation of the complex plane. This is so because polynomials in general have complex solutions[22]. Equation (C-1) has only one root, which real root falls on the horizontal line (abscissa) bisecting the complex plane. All real numbers are only a small portion of all possible numbers, and may be plotted on the abscissa of the z-plane. Any numbers off of this "real line" will have some imaginary part to them.

Figure C-3 (left) concisely represents the AR(1) process in the complex plane. Any number can be thought of as a vector (anchored at [0,0]) on the complex plane. The dotted or "unit circle" on the plane represents all complex numbers having a magnitude of one (and any phase). Transform the pole-zero diagram to the magnitude response plot by rotating a magnitude

[22] For example, $x^2 + 4 = 0$ has no real solutions. Rather the solutions to this polynomial are x={2i, -2i}. Both of these complex numbers may be plotted on a z-plane.

one (or unit) vector around the unit circle. Imagine an arrow extending from (0,0) on the complex plane to (1,0). This vector represents the lowest frequency (0 radians) of any input to the filter. As the arrow rotates around the "unit circle", still anchored at (0,0), it represents successively higher frequencies passed through the filter. Eventually, the arrow would turn to point to (-1,0), which represents the highest frequency that can be uniquely received by a digital frequency, π radians. All higher frequencies appear as mirror images of this set of frequencies [0, π], due to sampling theory (see [14] and [22]). Thus, while any frequency signal may be sampled and input to a digital system, it will be assigned a phase on the range [0, π]. While representing the 0 radian frequency, the arrow is pointed to (1,0) and must cross the pole. Low frequencies are magnified by this filter. So the AR(1) filter with its root, ϕ_1=0.8, is a low-pass filter since it passes low frequencies. When the arrow is pointed to (-1,0), representing π radians, it is furthest away from the pole on the diagram and the low-pass filter attenuates these high frequencies, as seen in Figure C-3 (right).

General Forms of the Linear Stationary Models

Generally speaking, any polynomial will have a set of complex solutions. Referring back to the schematic in Figure C-1(b), additional delays may be added below the first delay, leading to the general form of the autoregressive process, AR(p). Expressed in mathematical terms, this general autoregressive process

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + \ldots + \phi_p \tilde{z}_{t-p} + a_t \qquad \text{(C-5)}$$

adds a random input plus scalar multiples of several previous outputs to obtain the current output. Performing the same algebraic steps as before, the transfer function may be obtained for this AR(p), as

$$H(z) = \frac{z_t}{a_t} = \frac{1}{1 - \phi_1 z^{-1} - \ldots - \phi_p z^{-p}} = \frac{B(z)}{A(z)} \tag{C-6}$$

The roots of this AR(p) model are represented as X's in the z-plane. Multiple roots allow for more sophisticated systems that can pass bands of frequencies or also 'notch out' (attenuate) bands of frequencies.

General forms of the Moving Average process and Mixed Autoregressive-Moving Average process are also easily imagined. Under the MA(1) schematic of Figure C-1(a), an arbitrary number of delays may be added. This is also true for both the feed-forward and feed-back loops of Figure C-1(c). The mathematical expressions of an MA(q) model or an ARMA(p,q) model are

$$\tilde{z}_t = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ldots - \theta_q a_{t-q} \tag{C-7}$$

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + \ldots + \phi_p \tilde{z}_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ldots - \theta_q a_{t-q} \tag{C-8}$$

Moving Average processes have an all-zero transfer function

$$\frac{\tilde{z}_t}{a_t} = 1 - \sum_{i=1}^{q} \theta_i z^{-i} \tag{C-9}$$

Therefore, its roots are shown by O's instead of X's in the z-plane. MA(1), θ_1=0.8 will be a high-pass filter since a zero replaces the pole of Figure C-3 (left). The MA(q) model will never produce an unstable realization[23], as would the AR(p) when one or more poles are outside of the unit circle. This is because roots of the MA(q) model cause H(z)=0, instead of infinity.

[23] Realization refers to the output of a random process. For example, when a random process is created by inputting a random sequence of numbers into a difference equation, $\tilde{z}_t = a_t - \theta_1 a_{t-1}$, the result will be one realization of a Moving Average Process. But input a different random sequence, and the observed sequence will be different. Yet, despite the differences, the two sequences will retain some key similarities (e.g.- shape of Magnitude Response).

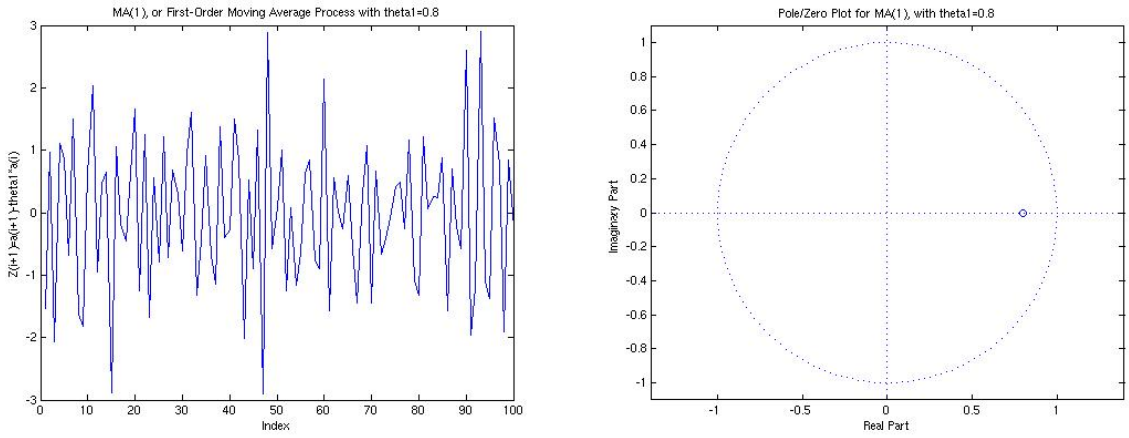


FIGURE C-4: (LEFT) REALIZATION OF MA(1) PROCESS CREATED WITH θ_1=0.8, AND (RIGHT) POLE-ZERO DIAGRAM FOR THE SAME PROCESS.

Finally, the ARMA(p,q) process transfer function combines Equations (C-6) and (C-9)

$$\frac{\tilde{z}_t}{a_t} = \frac{1 - \sum_{i=1}^{q} \theta_i z^{-i}}{1 - \sum_{j=1}^{p} \phi_j z^{-j}} \tag{C-10}$$

Its pole-zero diagram would show both X's and O's on the z-plane. Placing poles and zeros on the complex plane makes more sophisticated filters possible with fewer roots. Many practically occurring processes may be modeled by Linear Stationary models of 2nd order (i.e.- ARMA[2,2]) or less.

The general form of a transfer function, with numerator and denominator polynomials separated into root form is $H(z) = \frac{(z - z_1)(z - z_2)...(z - z_q)}{(z - p_1)(z - p_2)...(z - p_p)}$. The z_i in the numerator are zeros to the transfer function, being numbers that cause H(z) to equal zero. They are plotted as O's on the z-plane. The p_i in the denominator are poles that cause H(z)= ∞. Poles are shown by X's on the z-plane. An AR(p) process has no zeros since B(z)=1.

Identifying the dependencies in Linear Stationary Models

A straight-forward method of finding inter-lag dependencies for the Autoregressive model is the Yule-Walker Equations. Let's repeat the expression for an AR(p) model that was given in (C-5):

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + ... + \phi_p \tilde{z}_{t-p} + a_t$$

Now, by multiplying through by $\tilde{z}_{t-k}$, we will see that covariance functions for the AR(p) model are related to each other by the same relation.

$$\tilde{z}_{t-k}\tilde{z}_t = \phi_1 \tilde{z}_{t-k}\tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-k}\tilde{z}_{t-2} + ... + \phi_p \tilde{z}_{t-k}\tilde{z}_{t-p} + \tilde{z}_{t-k} a_t, \qquad k>0$$

$$\gamma_k = \phi_1 \gamma_{k-1} + \phi_2 \gamma_{k-2} + ... + \phi_p \gamma_{k-p}, \qquad k>0 \qquad \text{(C-11)}$$

Hence, a covariance function of an AR(p) model is related to a linear combination of delayed covariance functions from the same model. Dividing by $\gamma_0 = \sigma_z^2$, shows that the autocorrelation functions are similarly related.

$$\rho_k = \phi_1 \rho_{k-1} + \phi_2 \rho_{k-2} + ... + \phi_p \rho_{k-p}, \quad k>0 \qquad \text{(C-12)}$$

Since autocorrelation functions show the relationship between the present observation and each preceding observation, similarly the lag one correlation will be included in all the previous correlations. Therefore, there is some linear trend to the correlations; the lag one correlation influences the lag two correlation, and the lag two influences the lag three correlation, etc.

Partial correlations reveal exactly how much correlation there is between X_t and X_{t-1}, all other dependencies aside [19]. This method is built upon the recurrence relationship that exists between autocorrelation functions. Since an autocorrelation function is related to a linear combination of time-delayed autocorrelation functions, the linear coefficients, ϕ_k , are called partial correlations. The linear coefficients may be found by solving a set of linear equations.

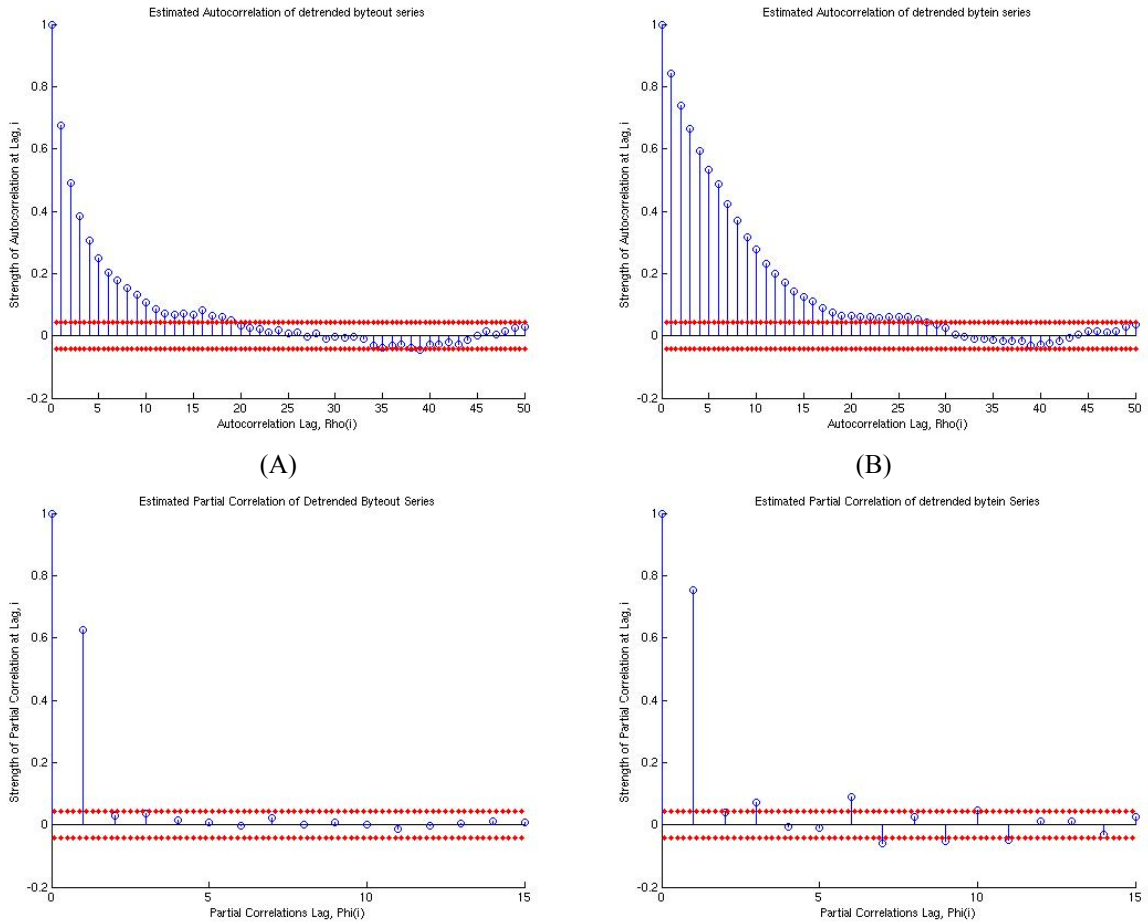
This method, known as the Yule-Walker method, directs the researcher to first calculate an autocorrelation function. Next, build a set of linear equations with k=1, 2, ..., j for increasing rows. By virtue of the normalized and symmetric nature of the autocorrelation function, $\rho_0 = 1$ and $\rho_k = \rho_{-k}$.

$$\begin{aligned}
\rho_1 &= \phi_1 + \phi_2\rho_1 + \ldots + \phi_j\rho_{j-1} \\
\rho_2 &= \phi_1\rho_1 + \phi_2 + \ldots + \phi_j\rho_{j-2} \\
&\ldots \\
\rho_j &= \phi_1\rho_{j-1} + \phi_2\rho_{j-2} + \ldots + \phi_j
\end{aligned} \tag{C-13}$$

The autocorrelation function will be derived by the methods of Chapter 2 from observed data. If the observed data is truly AR(p), and j>p, then $\phi_k > 0$ for k=1,2,...p and $\phi_k \approx 0$ for k=p+1, p+2, ..., j. Partial correlations represent an easy method of identifying which model could recreate a series. Specifically, the partial correlation function will show what order, p, of autoregressive model could represent the signal.

Viewing Figure C-5 (A) & (B), both detrended byteout and bytein series appear to have long correlations. However, partial correlations indicate that detrended byteout (Figure C-5(C)) is adequately described by an AR(1) model. Differenced bytein (Figure C-5(D)) has a long series of partial correlations, reaching beyond AR(15), and requiring a higher model to evaluate. Box, Jenkins and Reinsel establish a boundary of significant correlations as $2\hat{\sigma} = \frac{2}{\sqrt{n}}$, where n is the length of original series. The partial correlations of an uncorrelated series could be expected to have partial correlations within these bounds with 95% confidence. Partial correlations that are significantly more than the confidence boundary and partial correlations that are not bounded more than 5% (as in Figure C-5(D)) provide reason to reject the hypothesis of an uncorrelated

series for AR(k>p). In this case, 95% confidence boundaries are $\frac{1.96}{\sqrt{n}} = \frac{1.96}{\sqrt{2328}} = 0.0415$]. Thus as the series is longer, less false correlation is expected, and the bounds are tightened.



(A) (B)

(C) (D)

FIGURE C-5: AUTOCORRELATION AND PARTIAL CORRELATION FUNCTIONS FOR BYTEOUT AND BYTEIN.

The finding of Figure C-5 (C) is also intriguing, since Chapter 4 shows that byteout is Long-range dependent, and therefore not adequately modeled by an AR(1) model. So, how is it that partial correlations indicate only one significant partial correlation in the detrended byteout series?

Appendix D: MATLAB code used during research

Comparison

```
% Aggregate the data
load segmented
load synthetic_short
load ARsequence

subplot(3,3,1); stem(3400:3499,segmented(3400:3499),'k.')
title('byteout series')
xlabel('Polling Intervals')
ylabel('Bytes transferred during p.i.')

length10=floor(length(segmented)/10);
byteout10=zeros(1,length10);
for i=0:length10-1
   byteout10(i+1)=sum(segmented(i*10+1:i*10+10));
end
subplot(3,3,4); stem(300:339,byteout10(300:339),'b.'); hold on;
stem(340:349,byteout10(340:349),'k.'); stem(350:399,byteout10(350:399),'b.');
title('byteout (10:1)')
xlabel('Polling Intervals')
ylabel('Bytes transferred/ 10 p.i.'); hold off

length100=floor(length(byteout10)/10);
byteout100=zeros(1,length100);
for i=0:length100-1
   byteout100(i+1)=sum(byteout10(i*10+1:i*10+10));
end
subplot(3,3,7); stem(1:29,byteout100(1:29),'g.'); hold on;
stem(30:33,byteout100(30:33),'b.'); stem(34,byteout100(34),'k.');
stem(35:39,byteout100(35:39),'b.'); stem(40:100,byteout100(40:100),'g.');
title('byteout (100:1)')
xlabel('Polling Intervals')
ylabel('Bytes transferred/ 100 p.i.'); hold off;

%  Repeat this method of ‘synthetic’ and the AR(1) sequence, appropriately changing subplots
and axis titles.
```

Autocorr

```
function [rho]=autocorr(series,iter)
% This function creates an estimated autocorrelation for a given set of data. The autocorrelation
function outputs a 'rho' vector not longer than 50 values long, and stem plots 'rho'.  Created by
LT Sam Edwards on 16 July 2005, using the estimation algorithm described by Box and Jenkins
on pg. 30-32 of "Time Series Analysis".
%      [rho]=autocorr(series,iter)

m=mean(series);
ls=length(series);
if nargin==1 & ls<201
   iter=floor(ls/4);
elseif nargin==1 & ls>=201
   iter=50;
elseif iter>50
   iter=50;
end

rho=ones(1,iter+1);
sum=0;  % Creating the c0 (Variance)
for i=1:ls
   sum=sum+(series(i)-m)*(series(i)-m);
end
c0=sum/ls;

for i=1:iter
   sum=0;
   for j=1:ls-i
      sum=sum+(series(j)-m)*(series(j+i)-m);
   end
   sum=sum/ls;
   rho(i+1)=sum/c0;
end
%logrho=log10(rho);
stem(0:iter,rho)
title('Estimated Autocorrelation of byteout series, after Hanning window')
xlabel('Autocorrelation Lag, Rho(i)')
ylabel('Strength of Autocorrelation at Lag, i')
```

Qqplots_detrend

```
% This program performs the Q-Q plot analysis for CGC FORWARD's detrended byteout.
% Created by LT Sam Edwards on 17 March 2006.

clear all; close all              % Load the CGC FORWARD byte arrival data.
load byteout_detrend

m=mean(detrend);                  % Find statistics
std=sqrt(var(detrend));

% Create the Cumulative Gaussian Distribution and fit to 'detrend' statistics
q=0.01:0.01:0.99;
z=Zfunction(q);
z=z.*std+m;

ascend=sort(detrend);             % Fit quantiles of 'bytes' to the Gaussian Distribution
k=1;
for i=0:98
  Z(k)=median(ascend(i*23+1:i*23+23));
  e(k)=.25*(ascend(i*23+23)-ascend(i*23+1));
  k=k+1;
end
subplot(2,1,1)
errorbar(z,Z,e,'.'); hold on;
plot(z,z,'r-.'); hold off;  % errorbar.m plots z vs. Z, so the standard is along y=x line.
title('Q-Q plots demonstrate how averaged byteout converge to N(m=0,std=1.4e6)')
xlabel('Gaussian quantiles (with interquantile ranges)')
ylabel('Quantiles of byteout_detrended')
ascend=[];
Z=[];
e=[];

% Since a linear combination of gaussian distributions converge to a gaussian distribution, I
break the 'detrend' series into 23 sets and average the quantiles of each set, to see if it will
converge to a gaussian distribution.
for i=0:22
   for k=1:99
      Z(k,i+1)=detrend(i*99+k);
   end
end
ascend=sort(Z);
for m=1:99  % find the range of each quantile throughout the 19 sets
   e(m)=.25*(max(ascend(m,:))-min(ascend(m,:)));
end
avg=mean(ascend,2);  % this is the linear combination that should cause
```

```
avg=avg';          % the sets to converge to a gaussian distribution
subplot(2,1,2)
errorbar(z,avg,e, '.'); hold on;
plot(z,z,'r-.') % errorbar.m plots z vs. avg, so the standard is along y=x line.
xlabel('Gaussian quantiles (with interquantile ranges)')
ylabel('Quantiles of averaged differences')

% The plots show convergence toward the y=x line.  Although the detrended sequence was
always close to gaussian distribution, there can be no doubt of its gaussianity upon averaging.
```

Zfunction

```
function z = Zfunction(Q)
% Zfunction(Q) outputs the Z corresponding to a given area under the
% standard normal distribution, Q.  I use this program to determine the
% quantiles of the standard normal distribution.  The output may be adapted
% to a given Normal distribution by multiplying by standard deviation and
% adding the mean.
%   Written and checked by LT Sam Edwards on 8 OCT 2005

z=sqrt(2)*erfinv(2*Q-1);
return
```

Daily_trend

```
% Find the seasonal trend in packin
clear all
load packin
for i=1:248
   d=i:248:2328;
   daily(i)=mean(packin(d));
end

figure
plot(daily)
title('Daily trend of packin during 1-11 July')
xlabel('Polling intervals throughout an average day')
ylabel('Packet Arrivals in an average polling interval')

save daily_packin daily
```

Detrend_daily

```
% Detrend byteout
load daily_avg_packout
load packout
x=packout;
% Subtract daily cycle
trend=[avg_daily_packout avg_daily_packout avg_daily_packout avg_daily_packout...
avg_daily_packout avg_daily_packout avg_daily_packout avg_daily_packout...
avg_daily_packout avg_daily_packout(1:96)];
dpo=x-trend';

% subtract linear trend before 1860
h=1:1860;
H=[h' ones(1860,1)];
x1=inv(H'*H)*H'*dpo(1:1860);
z1=H*x1;  %z1 is the line that will be subtracted before dpo(1860)
y1=dpo(1:1860)-z1;
h=[]; H=[];

% subtract linear trend after 1861
h=1861:2328;
H=[h' ones(468,1)];
x2=inv(H'*H)*H'*dpo(1861:2328);
z2=H*x2;  %z2 is the line that will be subtracted after dpo(1861)
y2=dpo(1861:2328)-z2;

dpo=[y1; y2];
save detrend_packout dpo
figure
plot(dpo)
title('packout series w/o daily cycle or linear trend')
xlabel('Polling Intervals')
ylabel('Packet Transfers per Polling Interval')
R=xcorr(dpo,'coeff');
figure
plot(R(2327:4655))
title('Correlations of Detrended packout series')
xlabel('Lags, i')
ylabel('Correlation at lag i')
```

create_synthetic_traffic

```
% This script creates a sequence of {0,1} with the lengths of Off/On periods determined by a
Pareto distributed random sequence that sums to 10,000. Add 500 of these sequences together to
obtain a self-similar sequence, then aggregate.m will aggregate this sequence by orders of
magnitude.
%
% Method originally presented in "Proof of a Fundamental Result in Self-Similar
% Traffic Modeling" by Murad S Taqqu, et al in ACM SIGCOMM Computer Communication
% Review, Vol. 2, pp.5-23, 1997. Paper includes extensive mathematical proofing.
%
% Created by LT Sam Edwards on 3 March 2006

% **************************OUTSIDE CODE*********************************
% Run the inside code 500 times to aggregate into self-similar data
% This portion of code determines the # of alternating On/Off periods that will
% create this sequence of {0,1}.
s=zeros(10000,1);
for m=1:500

% ***************************INSIDE CODE*********************************
u=rand(10000,1); % start with building a uniformly-random series
w=(1-u).^-.83333; % transform this series into a Pareto-random series
w=floor(w); % make values of the Pareto-series integers, since they will be
lw=0;       % lengths of On/Off periods.
i=1;
while lw<10000
   lw=lw+w(i);  % limit the length of the Pareto series to 10000
   i=i+1;
end

% This portion of code creates the sequence of {0,1}
z=[];
for j=1:2:i
   x=ones(w(j),1);     % make alternating sequences of ones/zeros with their
   y=zeros(w(j+1),1);  % lengths being determined by successive values of the
   z=[z; x; y];        % Pareto-series.
end
z(1)=[];    % Discard the first values, since it'll always be one. Later values won't.
if length(z)>19000  % Sometimes particularly long strings of {0,1} must be dealt
   h=z(1:19000);   % with by cleansing everything above 19000.
   z=[];
   z=h;
   clear h;
end
if length(z)>10000
```

```
   disc=mod(length(z),10000); % keep 10000 values of z
   z=discard(z,disc);
end

% **********************FINISH OUTSIDE CODE*****************************
   s(:,m)=z;
end
synthetic=sum(s,2);
%save test synthetic
```

Partcorr

```
function [phi]=partcorr(rho)
% This function creates an estimated partial correlation function for a given set of
% autocorrelations (rho). The rho can be found using autocorr.m, and the phi will be
% 15 values long, and stem plots 'phi'.  Created by LT Sam Edwards on 28 September 2005,
% using the Yule-Walker equations as described by Box and Jenkins on pg. 64-69 of
% "Time Series Analysis".
%      [phi]=partcorr(rho)

lr=length(rho);
rho(1)=[];
rho(16:lr-1)=[];
P=[1 rho(1:14); rho(1) 1 rho(1:13); rho(2:-1:1) 1 rho(1:12); rho(3:-1:1) 1 rho(1:11);...
       rho(4:-1:1) 1 rho(1:10); rho(5:-1:1) 1 rho(1:9); rho(6:-1:1) 1 rho(1:8);...
       rho(7:-1:1) 1 rho(1:7); rho(8:-1:1) 1 rho(1:6); rho(9:-1:1) 1 rho(1:5);...
       rho(10:-1:1) 1 rho(1:4); rho(11:-1:1) 1 rho(1:3); rho(12:-1:1) 1 rho(1:2);...
       rho(13:-1:1) 1 rho(1); rho(14:-1:1) 1];
rho=rho';
phi=inv(P)*rho;
stem(0:15,[1; phi]); hold on;
title('Estimated Partial Correlation of First Differences Series')
xlabel('Partial Correlations Lag, Phi(i)')
ylabel('Strength of Partial Correlation at Lag, i')

% Add '2*sigma-hat' limits
partcorr_bounds=2*(1/sqrt(1886));
i=0.1:0.2:15;
plot(i,partcorr_bounds,'r.');
plot(i,-partcorr_bounds,'r.');
```

Hurst_logvar

```
% This script will calculate the changing slope of variances for a set of data, which is known as
% the alpha value.  The Hurst parameter may next be calculated as H = 1 - alpha/2.
close all;
clear all;
load byteout_detrend.mat;
HL=detrend;   % Input the data to be analyzed
l=length(HL);
for m=1:100
   disc=mod(l,5*m); % keep only full sets of m values from cutter stats.
   HL1=discard(HL,disc);
   lkept=length(HL1);
   n=floor(lkept/(m*5));
   for j=0:n-1
      z(j+1)=mean(HL1(j*5*m+1:j*5*m+5*m));
   end
   varz(m)=var(z);
   z=[];% reset my temporary variables
end

var_log=log(varz);
x_log=log(5:5:500);
plot(x_log,var_log,'k+')
xlabel('Sample Size, ln(n)');
ylabel('ln(Variance)'); hold on;

% Use Least Squares method to estimate alpha
h=x_log(1:11); % select the number of sample variances which appear linear from
            % the above plot.
H=[h' ones(11,1)];
x=inv(H'*H)*H'*var_log(1:11)';
z1=H*x;
plot(h,z1,'r')
legend('Variances per sample size','Estimate of Alpha')
alpha=-x(1);                % Change Title Name (below)
title(['Variances of Detrended Byteout series: alpha=', num2str(alpha)])
```

discard

```
function [a]=discard (x,m)

% This function dicards the last m values in the array x.
% [a]=discard (x,m)
% Checked and found working on 30 March 2005.

l=length(x);
for i=0:m-1
   x(l-i)=[];
end
a=x;
```

Hurst_wavelet

% This script plots the Spectral Density of a given signal, which may then be used to calculate the Hurst parameter. First, I load the desired sequence into the 1-D Wavelet Analyzer, and analyzed using the a Daubechies wavelet. The wavelet coefficients were then stored in 'db5coefs.mat' or similar. After generating coefficients, I find the variance at each scale and plot these variances. Then use Least Squares Method to plot the slope of changing variance. This is the method described by Abry and Veitch in "Wavelet Analysis of Long-Range Dependent Traffic" IEEE Transactions on Information Theory, vol. 44, No. 1, Jan 1998. The Hurst parameter may next be calculated as H = 1/2*(slope+1).
% The wavelet coefficients are arranged in a concatenated vector, indexed by a 'longs' vector. The first # in longs corresponds with the indices of the averaged signal after 7 levels of decomposition. The next number corresponds to the indices of the level 7 wavelet coefficients, etc. The last number corresponds to the length of the original signal, and is not present in the coeffs vector. This breakdown is well explained at the bottom of "Importing and Exporting Information from the Graphical Interface", from the Help Menu of Wavelet Toolbox.
% Script written by LT Sam Edwards on 21 July 2005.

```
clear;
load byteout_detrend_coefs;
indices=longs;
a7=1:indices(1);
b7=indices(1)+1:sum(indices(1:2));  % Separate indices for each scale's coeffs.
b6=sum(indices(1:2))+1:sum(indices(1:3));
b5=sum(indices(1:3))+1:sum(indices(1:4));
b4=sum(indices(1:4))+1:sum(indices(1:5));
b3=sum(indices(1:5))+1:sum(indices(1:6));
b2=sum(indices(1:6))+1:sum(indices(1:7));
b1=sum(indices(1:7))+1:sum(indices(1:8));
scalevar=zeros(1,7);
scalevar(1)=var(coefs(b1));  % Calculate the variance at each scale
scalevar(2)=var(coefs(b2));
```

```
scalevar(3)=var(coefs(b3));
scalevar(4)=var(coefs(b4));
scalevar(5)=var(coefs(b5));
scalevar(6)=var(coefs(b6));
scalevar(7)=var(coefs(b7));
%stand=zeros(1,10);          % Now, convert to Standard Deviations at each scale
%stand(1)=sqrt(scalevar(1));
%stand(2)=sqrt(scalevar(2));
%stand(3)=sqrt(scalevar(3));
%stand(4)=sqrt(scalevar(4));
%stand(5)=sqrt(scalevar(5));
%stand(6)=sqrt(scalevar(6));
%stand(7)=sqrt(scalevar(7));
%stand(8)=sqrt(scalevar(8));
%stand(9)=sqrt(scalevar(9));
%stand(10)=sqrt(scalevar(10));
logvar=log2(scalevar);
logvar=logvar';
scales=1:7;
scales=scales';
[z1,slope] = LeastSquares(scales(1:7),logvar(1:7));
plot(scales,logvar,'*'); hold on;
plot(scales(1:7),z1,'r-'); hold off;
title(['Spectral Densities of detrended byteout using db5 basis function, slope is ',
num2str(slope(1))])
xlabel('Scales')
ylabel('Log2(Spectral Densities)')
legend('Spectral Densities','Reconstructed Line')
```

Power Spectral Density

The PSD may be calculated by conducting the Fast Fourier Transform (FFT) of the data. The Fast Fourier Transform has a complex result, and the Magnitude Response is then calculated by multiplying the complex vector $\vec{Y}$ by its complex conjugate $\left|\vec{Y}\right| = \sqrt{\vec{Y} \cdot \vec{Y}^*}$. The resulting vector $\left|\vec{Y}\right|$ are the real valued frequencies. These must, then, be arranged along an array of frequencies for a meaningful plot. The MATLAB code is shown below, where only positive frequencies were displayed (negative frequencies being a mirror image of the positive frequencies), and the sampling frequency is $(340 \text{ sec})^{-1}$=0.0029 Hz.

```
Y=fft(byteout,16384);
Pyy=Y.*conj(Y)/16384;
f=.0029*(0:8192)/16384;
plot(f,Pyy(1:8193));
```

x_tilda

% Creating xtilda: X is the aperiodictraining sequence, which has periodicity removed from the training sequence. X multiplied by(1-B)^d to create xtilda. xtilda will be a short-memory process.

```
clear
load diff_byteout
x=diff_bo;
% First find coefficients in the summation
for k = 0:49
Cdk(k+1)=gamma(1.4658)/(gamma(k+1)*gamma(1.4658-k));
series(k+1)=Cdk(k+1)*(-1)^k;
end

% Calculate xtilda during its steady state phase
for i=51:2080
   for j=1:50
      xt(j,i-50)=series(j)*x(i-j+1);
   end
end
steady=sum(xt);
xtilda=steady;

save diff_byteout_sm xtilda;
plot(1:2030,xtilda)
title('Differenced byteout series, long memory removed')
xlabel('Polling Interval')
ylabel('# of Packets Arriving per p.i.')
```

arma_twostep

```
% ARMA(15,15) model for detrended data.  Calculate x_hat, using the AR(50) model
% Glossary:
% xtilda is short-memory byteout
% xhat is AR(50) estimate of short-memory byteout
% xhat2 is ARMA(15,15) estimate of short-memory byteout

clear; load diff_byteout_sm
z = xtilda(51:2030)';
for i=51:2030
   for k=1:50
      H(i-50,k)=xtilda(i-k);
   end
end
phis = inv(H'*H)*H'*z;
xhat=H*phis;
subplot(2,1,1);plot(z); hold on; plot(xhat,'r')
title('AR(50) doesnt sufficiently model byteout-sm')
legend('Short-memory byteout','AR(50) estimate')

% Calculate Innovation
innov = z - xhat;
innov = innov';
hold off; subplot(2,1,2); plot(innov);
ylabel('Innovations of xtilda vs xhat')

% Now form the ARMA(15,15) model; I will use z = xtilda and innov = errors
% first form the new H matrix from z, then the innovations half of H.
for i=16:1980
   for k=1:15
      Hnew(i-15,k)=z(i-k);
      E(i-15,k)=innov(i-k);
   end
end
Harma = [Hnew E];
phis_psis = inv(Harma'*Harma)*Harma'*z(16:1980);
save afterarma phis_psis innov
xhat2=Harma*phis_psis;
save forecast xhat2; figure
subplot(2,1,1); plot(z(16:1980)); hold on; plot(xhat2,'r');hold off;
title('ARMA(15,15) does not produce a better match of byteout-sm')
legend('Short-memory byteout','ARMA(15,15) estimate')
innov2=z(16:1980)-xhat2;
subplot(2,1,2); plot(innov2);
```

Vita

Lieutenant Samuel Edwards graduated from the United States Coast Guard Academy (CGA) in May 2000. His father, Captain Norman Edwards (CGA '68), was a ship captain whose passion for the sea towed Sam's family across the country in support of various Coast Guard assignments. Sam enjoyed his opportunities to see the Coast Guard in action, and eventually applied for his own appointment to the CGA in 1994. His tours of duty since commissioning include: Damage Control Assistant aboard CGC BEAR, homeported in Portsmouth, VA; Project Officer for the Coast Guard's 'Rescue 21' project in Washington, DC; and now Graduate Studies in Electrical Engineering at Virginia Tech. He has been selected to serve as a Communications Engineer for the USCG Research and Development Center in Groton, CT for his follow-on tour. He and his wife, Peggy, currently live in Ledyard, CT. Their first daughter, Esra Grace, arrived as Sam was finishing this thesis. Her name remembers a 'miraculous journey', and graduate studies (this research in particular) have been a miraculous journey for Sam and his family.