Spring 1982

NONUNIFORMLY AND RANDOMLY SAMPLED SYSTEMS

GEORGE DIMITRIOS KONTOPIDIS

INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of "sectioning" the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.

University Microfilms International
300 N. Zeeb Road
Ann Arbor, MI 48106

Kontopidis, George Dimitrios

NONUNIFORMLY AND RANDOMLY SAMPLED SYSTEMS

University of New Hampshire PH.D. 1982

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __√__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages __√__
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Other _____

NONUNIFORMLY AND RANDOMLY SAMPLED SYSTEMS

BY

George D. Kontopidis
B.S., National Technical University of Athens, 1977
M.S., University of New Hampshire, 1978

DISSERTATION

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy
in
Engineering
Signal Processing Area

May, 1982

This dissertation has been examined and approved.

Filson H. Glanz

Dissertation director, **Filson H. Glanz**
Associate Professor of
Electrical and Computer Engineering

David E. Limbert

Dissertation director, **David E. Limbert**
Associate Professor of Mechanical Engineering

L Gordon Kraft

L. Gordon Kraft, Assistant Professor of
Electrical and Computer Engineering

John L Pokoski

John L. Pokoski, Professor of
Electrical and Computer Engineering

L D Meeker

L. David Meeker, Professor of Mathematics

April 20, 1982

Date

ACKNOWLEDGEMENTS

I would like to express my deep appreciation and gratitude to my advisors Prof. Filson Glanz and Prof. David Limbert for their encouragement and stimulating motivation for completing this research. Without their continous guidance and creative discussions this work would be impossible. Also, a special thanks goes to Prof. Gordon Kraft for his suggestions and recommendations on my studies and professional activities.

I would also like to thank Prof. John Pokoski and Prof. David Meeker for their interest and contributions in completing this research. I also thank the Electrical Engineering Department, the Mechanical Engineering Department and the Graduate School for their financial support.

A special thanks goes to many colleagues and friends who have encouraged my technical development and sustained me during this educational program.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

Set Notations and Conventions

C	The set of complex numbers
R	The set of real number
N	The set {1,2,3,...}
Z	The set {...,-2,-1,0,1,2,...}
$x \in A$	The element x belongs to the set A
$\cup \cap \subset$	Union, intersection, subset respectively.
$\Longleftrightarrow$	Equivalence relationship

Matrix Notations and Conventions

T	(As a superscript) denotes the transpose of a matrix
$x=(x_1\ x_2\ \ldots\ x_N)$	means that x is a row vector of N elements
$x=(x_1,x_2,\ldots,x_N)$	means that x is a column vector of N elements
$x = \begin{pmatrix} x_1 \\ x_2 \\ .. \\ x_N \end{pmatrix}$	means that x is a column vector of N elements
$A=(a_{ij})$	matrix A consisting of elements a_{ij}.
$(a_{ij}: i=1,N;\ j=1,M)$	matrix A with dimension N by M
$\begin{bmatrix} 11 & 12 & \ldots & 1M \\ 21 & 22 & \ldots & 2M \\ \ldots & \ldots & \ldots & \ldots \\ N1 & N1 & \ldots & NM \end{bmatrix}_a$	has identical meaning with $(a_{ij}: i=1,N;\ j=1,M)$

Probability Theory Notations

$F_X(x)$	Cumulantive probability function of X evaluated at x.

$p_X(x)$	Probability density function (pdf) of X evaluated at x.
$P_X(s)$	Characteristic function of X evaluated at s. Notice that the characteristic function is of X is defined as the Bilateral Laplace transform of the probability distribution of X.
Pr(A)	Probability of the event (or set of values) A.
Ex(t), Ex	Expected value of x evaluated at t.
var{x} $V_x(t)$,V(x;t)	Variance of x, $E(x-Ex)(x-Ex)^T$
cov{x,y}	Covariance of x and y, $E(x-Ex)(y-Ey)^T$
msq{x} $M_x(t)$,M(x;t)	Mean square value of x. That is, msq{x} = Exx^T.
cor{x,y}	Correlation of x with y. That is Exy^T.
R(x;t,s) $R_x(t;s)$	Autocorrelation $Ex(t+s)x^T(t)$
R(x;s) $R_x(s)$	Autocorrelation of a stationary process $Ex(t+s)x^T(t)$.

Miscellaneous Conventions

T, T_n, $\{T_n\}$	Time intervals. T may denote a random variable and T_n can be interpreted as a realization of T. $\{T_n\}$ in general means the sequence of T_n's.
t, t_n, $\{t_n\}$	Time instants. The meaning of t, t_n and $\{t_n\}$ is similar to T, T_n and $\{T_n\}$. In general T refers to time differences and t refers to absolute time.

r,t,u,h	Dummy variables for time used in definitions and integrals.
a,b,c	Dummy integration variables for random variables.
X(s), Lx(t) (Lx)(s)	Bilateral Laplace transform of the function x(t).
$X^*(z)$, Zx(n) (Zx)(z)	Z transform of the sequence x(n) defined by the summation $(Zx)(z) = \sum_{n=0}^{\infty} x(n)z^{-n}$

ABSTRACT

NONUNIFORMLY AND RANDOMLY SAMPLED SYSTEMS

by

GEORGE KONTOPIDIS

University of New Hampshire, May, 1982

Problems with missing data, sampling irregularities and randomly sampled systems are the topics covered by this dissertation.

The spectral analysis of a series of periodically repeated sampling patterns is developed. Parameter estimation of autoregressive moving average models using partial observations and an algorithm to fill in the missing data are proved and demonstrated by simulation programs. Interpolation of missing data using bandlimiting assumptions and discrete Fourier transform techniques is developed. Representation and analysis of randomly sampled linear systems with independent and identically distributed sampling intervals are studied. The mean, and the mean-square behavior of a multiple-input multiple-output randomly sampled system are found. A definition of and results concerning the power spectral density gain are also given.

A complete FORTRAN simulation package is developed and implemented in a microcomputer environment demonstrating the new algorithms.

INTRODUCTION

1. Statement of Purpose
2. Overview of Chapter I
3. Overview of Chapter II
4. Overview of Chapter III
5. Overview of Chapter IV
6. Overview of Simulation Programs
7. State of the Art

Notes and References

INTRODUCTION

This introductory chapter describes the main objectives of this dissertation, overviews the contents of each chapter and relates the present work to the existing literature.

The overview of each chapter consists of the motivation for studying the specific topic, and how this topic is related to nonuniform sampling. It also summarizes the 'uniqueness' of the material in terms of originality of the results and outlines the methodology used.

1. Statement of Purpose

The subject of this dissertation, 'Nonuniformly and Randomly Sampled Systems', is very closely related to the material written in the author's Master's thesis 'Nonuniform Systems' [1]. Both works deal with the analysis of discrete systems that result from irregularly sampled continuous linear systems. As the title indicates the present work extends the material presented in [1].

The areas extended are the frequency domain properties of nonuniformly sampled systems and the study of randomly sampled systems from the analysis viewpoint. In [1] only time domain properties of nonuniformly sampled systems were studied. With the present work, two new dimensions in the study have been added, the frequency behavior of a class of nonuniformly sampled systems and the analysis of random sampling. Randomly sampled, noisy systems is one of the most challenging and interesting areas we deal with.

In order to put the study of nonuniformly and randomly sampled systems into proper context we define the following terms:

a. A Dynamic System is a set of differential equations involving time, t, as an independent variable, an input vector function u(t), and an output vector function y(t). For a given input u(t), the output y(t) satisfies the (given) differential equations. A dynamic system is a proper mathematical model for many physical systems which have the property of causality.

b. A Discrete System is a set of difference equations involving the integer index k as an independent variable, an input vector sequence u(k) and an output vector sequence y(k). For a given input u(k), the output y(k) satisfies the (given) difference equations. A discrete system can be used to model many physical and socio-economic events.

c. A Sampling Sequence is any time sequence consisting of time points selected in a prespecified manner. A sampling sequence can be a deterministic or a random sequence depending upon the law that defines the time points. If there is one-to-one correspondance between an index k and a function $t=t(k)$ the sampling sequence is deterministic. If there is a stochastic mapping from an (experimental) event to the time instants, the sampling sequence is called random.

d. A Nonuniformly Sampled System is the discrete system derived by solving a set of differential equations (called the dynamic system) at the points of a deterministic sampling sequence.

e. A Randomly Sampled System is the discrete system that results from a dynamic system statistically described at the time points of a random sampling sequence. Quite often we use the term Irregularly Sampled System to denote either a nonuniformly or a randomly sampled system.

The purpose of this dissertation is to extend the existing knowledge of the properties and the usefulness of nonuniformly and randomly sampled systems. Both the theoretical and the practical

aspects of irregular sampling are considered in this study.

The material is classified into four chapters. The following two tables illustate how the contents of each chapter are related to the main objective of the dissertation and how 'sampling irregularity' increases from one chapter to the next, respectively.

Table 1: Scope of each chapter

Scope-Objective	Chapter			
1) Frequency domain properties	Ch I		Ch III	
2) Time domain properties		Ch II		
3) Statistical characteristics				Ch IV
4) System design- Identification	Ch I	Ch II		

Table 2: Sampling irregularity of each chapter

Sampling irregularity	Chapter			
1) Periodic-nonuniform	Ch I			
2) Group of missing data		Ch II	Ch III	
3) Nonuniform sampling			Ch III	
4) Random sampling				Ch IV

At the end of each chapter there is a section entitled 'Notes and References'. Each entry in that section is referenced from the corresponding chapter using numbers within square brackets.

The following two cases indicate the necessity of processing irregularly spaced data. The first is the case of observing a physical system and the received information comes irregularly (e.g. meteor trails radar [2], human organism behavior [3]). The second is the case of controlling a physical system, or transmitting data in a nonuniform manner in order to achieve better performance. A more detailed discussion on this subject can be found in chapter I of [1] and the

introductory section of [7].

2. Overview of Chapter I

Chapter I is entitled 'Periodic Nonuniform Sampling' and deals with nonuniformly sampled systems for which the sampling process consists of periodically repeated patterns.

The main motivation for investigating this area comes from the lack of theoretical tools in the existing literature to analyze even simple types of periodic-nonuniformly sampled systems in the frequency domain. As an example we mention burst sampling which is a scheme of collecting data very fast for a small portion of the sampling period. It is expected intuitively (or based on heuristic arguments) that both high and low frequency information about the signal is contained in the collected samples. However, the relationship of the spectrum of the samples to the spectrum of the original signal had not been rigidly determined previously.

One first approach in studying the time domain properties of periodic-nonuniformly sampled systems was published in the paper 'Computer Controlled Systems Using Multiplexed I/O' [4]. In that paper we referred to 'Cyclicly Sampled, Cyclicly Held Systems' which is a special case of a burst sampled system. However, that analysis was limited to time domain and could not be used to predict frequency response characteristics.

Chapter I provides most of the results of our study of periodic-nonuniformly sampled systems concerning to frequency properties. The proof of the first theorems uses the Poisson Sum Formulas in ways very similar to the Nyquist Sampling Theorem. It takes several examples to

illustrate the generality of the main theorems and how they may be applied.

The analysis aspect of periodic-nonuniform sampling is the focal point of chapter I. Of secondary importance is a design procedure for sampling sequences with prespecified frequency characteristics. This design is based on Tchebyscheff polynomials which are commonly used in filter design and optimization. The idea of using weighted samples to achieve certain frequency characteristics comes from linear array antenna design methodology.

Finally, in chapter I we note that the period of the repeated sampling pattern can be less than the duration of the pattern. This scheme can be very useful in implementing very narrow band filters using samples collected at a much lower rate than is required for conventional digital filter designs procedures.

3. Overview of Chapter II

Chapter II is titled 'Partially Sampled ARMA Models' and deals with the identification of autoregressive moving average (ARMA) models based on nonuniformly collected data. With this chapter we try to expand the existing ARMA techniques to nonuniform sampling problems.

The sampling irregularity in this chapter comes from the effect of instrumentation failure; the sampling sequence is originally 'scheduled' to be uniform but the actually collected data are nonuniform due to 'missing' or 'badly' collected samples.

The main difficulty in using an ARMA model to describe a sampled linear system with some sampling irregularities is that the result is a time varying system (that was illustrated in Chapter II of [1]).

However, if the missing data can be 'grouped' over specified regions, or the number of missing samples relative to the total number of samples is small, then this chapter can be used to answer the most important problems: The first is parametric modelling and the second is interpolation of the missing samples.

The key technique used in chapter II is as follows: Consider that a linear system is sampled uniformly but the sampler introduces discrete noise. If we allow the noise level to take infinite values at the time instants of the missing samples, then any identification algorithm will not take into account the missing points. Therefore, if any arbitrary value is used, the final result is independent of that value. This technique is used for both off-line (least squares) and on-line (Kalman filter) identification of the ARMA coefficients. After the coefficients are found, interpolation of the missing data is accomplished by performing a set of linear operations (column permutations and other elementary matrix operations) and then solving another least squares problem.

4. Overview of Chapter III

Chapter III is titled 'Interpolation Using Bandlimiting Assumptions' and deals with bandlimited irregularly sampled signals. The sampling irregularity of this chapter is similar to the previous chapter but the methodology and the criteria used for interpolating the missing points are quite different.

A new assumption is imposed on the description of the original signal; we require that the signal be bandlimited and that the

scheduled sampling is fast enough to guarantee that the spectrum of the samples is also bandlimited.

The motivation for studying the interpolation problem with the above assumptions is based on a discussion with Dr. Griffiths and his paper 'High Resolution Spectral Estimates Obtained Using Data Extrapolation' [5]. In chapter III we generalize his results and we substantially reduce the computational difficulties concerning large matrix inversions.

Additionally, we present the theory behind iterative interpolation techniques which are the disrete time counterpart of Papoulis' work on 'Spectral Analysis and Bandlimited Extrapolation' [6]. Unfortunately, in the discrete case, the convergence of the iterative extrapolation can not be proved; futher discussion on this problem is given in chapter III.

5. Overview of Chapter IV

Chapter IV is entitled 'Randomly Sampled Systems' and deals with nonuniformly sampled systems from the statistical viewpoint.

The motivation in introducing the probabilistic formulation to model nonuniformly sampled systems comes from the complexity of the analysis by deterministic means. In the literature there are several papers discussing the 'simultaneous optimum detection and estimation of signals in noise' [9]. This problem is actually the same as the random sampling problem. We first try to detect which samples are 'good' and if the detection scheme is positive, the sample is incorporated in the estimation algorithm. Of ccurse, the various thresholds and false alarm probabilities are calculated based on statistical information about the

distribution of sampling failures.

One of the most significant contributions in the area of random sampling was made by Kalman's dissertation [7] almost two decades back. At that time state variable models were primitive and the supporting literature was rather poor. Kalman's work was mainly the development of state models to be used in solving closed loop randomly sampled systems. He limited the development to unforced first order systems operating in a closed loop configuration. His ideas in the area of random sampling were almost forgotten for many years because of the limited interest in this particular class of problems.

Our work is mainly an extension (and an update) of part of his work; we do not address the problem of controlling a randomly sampled system as he does, but we focus on statistical modelling and analysis of the randomly sampled system. We started with the paper 'Stochastically Sampled Systems' (unpublished) where we gave some preliminary results for a rather limited class of probability distributions of the sampling intervals. Also at that time, the computation of the various quantities was almost impossible because it involved solution of integral equations. (Later, these integral equations were reduced to ordinary differential equations.)

Continuous improvement of the randomly sampled model and the theory behind it, lead us to more general results. A standard multiple-input, multiple-output state model is used for all the development and only when it is necessary to deal with 'transfer functions' do we limit the results to single-input single-output systems. Also, a compact notation in studying randomly sampled systems is introduced which plays a fundamental role in the subsequent development. New results in the area of propagation of the mean values, propagation of the variances and the power spectral densities of randomly sampled systems are derived.

6. Overview of Simulation Programs

The computer as a tool of research and verification of theory was extensively used during the various phases of this work. (Actually, computer simulations of linear dynamic and discrete systems is one of the author's favorite areas.)

At the end of each chapter there exists either a worked example illustrating the application of the theorems, or a simulation program. A sample run, a brief discussion of the programming results and the source code are also included. Futhermore at the end of the dissertation there is an appendix titled 'Software Support' where we include two very useful software packages. The first consists of FORTRAN IV subroutines performing various matrix operations, and the second is an interactive package using the of functions of the first. Two points are unique about these programs:

(a) They are very easy to use, and

(b) they have been implemented, tested and verified on a microcomputer.

The programs were originally developed using a DEC-10 [8] and then were modified for an 8 bit machine. Benchmarks show that the computational accuracy is not inferior to DEC-10, the memory requirements are much less than the DEC-10 (of course we usually do not care if we 'run out of core' in the DEC-10 until it happens), but the computation time is several times longer.

In using the simulation programs, several theoretical mistakes were detected and corrected. Also, new theoretical horizons were opened and futher research on numerical aspects (i.e. roundoff error, overflow conditions) was done.

7. State of the Art

In this section a very brief outline of the pertinent literature about nonuniform and randomly sampled systems and waveforms is given. The reader is referred to the bibliography at the end of the dissertation for the particular articles and texts that we refer to.

Yen (1956) published one of the first papers referring to irregular sampling. He showed how to reconstruct a bandlimited signal uniformly sampled but with a finite number of points which had 'migrated' from their 'correct' positions. Following Yen's work Linden and Abramson (1960), Helms (1961) and Papoulis (1977) introduce the generalized sampling theorems where the main objective was again signal reconstruction of bandlimited signals using irregularly collected data. The irregularities they dealt with were of the first type in table 2 (periodic-nonuniform). In our development (chapter I) we are mainly concerned with the frequency properties of periodic-nonuniformly sampled signals.

Another group of researchers who have done work in randomly sampled systems consists of Buetler (1966, 1970), Leneman (1966, 1968), and Masry (1978). They discuss the 'alias free' property of randomly sampled systems, the correlation estimates and the power spectral estimates based on randomly collected samples. Lui's (1974) dissertation refers also to spectral properties of the above class of systems. One of the differences of our work with respect to theirs is that we deal with randomly sampled systems in time domain using state models and we do not focus on spectral estimates only. In terms of our techniques we follow Kalman's (1957) approach which is quite different than that of the above researchers. On the other hand Kalman mainly discusses the random samplers as an element of a closed loop system which is different from our objectives. (We are concerned with the open

loop properties in the time and frequency domains.)

Finally, we mention two excellent review papers on sampling written by Jury (1961) and Jerri (1977) discussing the sampling process from the viewpoints, of system theory and communication theory respectively.

Notes and References

[1] Kontopidis G. D., 'Nonuniformly sampled systems,' Master's Thesis, University of New Hampshire, 1978.

[2] Clark Ronald R., 'Meteor Wind Measurements at Durham N.H.,' Journal of Atmospheric Sciences, vol. 32, pp. 1689-1693, Sept 1975.

[3] King Robert, 'Parameter Identification in Strobed Tracer Kinetic Processes,' Int. J. Control, vol. 16, pp. 841-847, 1972.

[4] Kontopidis G., D. Limbert and F. Glanz, 'Computer controlled systems using multiplexed I/O,' IECI'80 Intern. Confer. on Mini and Micro Computer Applications, Philadelphia, March 1980.

[5] Griffiths L. J., 'High Resolution Spectral Estimates Obtained Using Data Extrapolation,' ICASSP 80 Proceed., Denver Colorado, 1980.

[6] Papoulis A., 'A New Algorithm in Spectral Analysis and Bandlimited Extrapolation,' IEEE Trans. Circuit Systems, vol. CAS-22, no. 9, pp. 735-742, Sept 1975.

[7] Kalman R. E., 'Analysis and synthesis of linear systems operating on randomly sampled data,' Ph.D. dissertation, Dept. of Elect. Engin., Columbia University, New York, 1957.

[8] Kontopidis G. D., 'LETSDO: An Interactive Systems and Signals Language,' (unpublished) U.N.H. 1980.

[9] Nahi N. E., 'Optimal recursive estimation with Uncertain Observation,' IEEE Trans. Information Theory, IT-15, pp. 457-462, July 1969.

CHAPTER I

PERIODIC NONUNIFORM SAMPLING

Outline of Chapter I

1. Terminology and Conventions
2. Sampling Using Integration
3. Sampling Using System Observations
4. Weighted Average Burst Sampling
5. Average Burst Sampling
6. Periodic Burst Sampling
7. Filtering by Weighted Average Burst Sampling
8. Designing Equal Ripple Samplers
9. State Space Models of Cyclicly Sampled Systems

Appendix Ia

Appendix Ib

Notes and References I

Outline of Chapter I

The term periodic sampling is used to describe a class of sampling sequences consisting of periodically repeated nonuniform patterns. This chapter consists of 9 sections and deals with the most important cases of periodic sampling.

In section 1 we summarize the definitions and the conventions used in the other sections.

In section 2 the 'sampling using integration' scheme is defined (theorem 1). Section 3 is a generalization of the previous scheme; the 'sampling using system observations' is introduced. An illustrative example applying theorem 1 follows.

In section 4 we define the 'weighted burst sampling' which is related to the averaging of a burst of samples (theorem 3). Section 5 is a generalization of the previous section. Another example follows illustrating theorem 3.

In section 6 'burst sampling' is defined. The theory is formulated with theorem 5, and an example illustrates the usage of that theorem.

Sections 7 and 8 provide the necessary theory to design equal ripple samplers based on Tchebyscheff polynomials. Then, the design procedure is outlined and demonstrated by an example.

Section 9 is of more theoretical importance; it provides a state space view of periodic sampling. It consists of two main theorems (6 and 7) which prove the equivalence of a periodically sampled system to a discrete (time varying) system.

Two appendices at the end of the chapter provide a set of formulas used for the proof of the theorems. A reference list follows with texts and papers related to periodic nonuniform sampling.

The new results claimed in this chapter are

a) the study of the frequency characteristics of periodic nonuniformly sampled systems,

b) the design procedure of samplers to perform digital filtering, and,

c) the formulation of a state model for periodic nonuniformly sampled systems.

1. Terminology and Conventions

Signal x(t)

This is equivalent to saying 'a real function of time'. It is assumed that t can take both positive and negative values.

Samples of x(t)

It is assumed that a period P is implicitly defined; the samples of x(t) consist of the sequence $\{x(nP)\}$ for all $n \in Z$ (Z is the set of all integer numbers). Very often we talk about the Z transform of the samples x(t); we mean the function

$$X^*(z) = \sum_{n=-\infty}^{+\infty} x(nP)\, z^{-n}$$

Sampled Version of x(t)

Similar to the previous definition, a period P is implicitly defined. The sampled version of x(t) with period P is the (continous time) generalized function x'(t) defined by

$$x'(t) = P \sum_{n=-\infty}^{+\infty} x(t)\, \delta(t-nP)$$

Note that the Fourier transform of x'(t) is equal to the Z transform of the samples of x(t) evaluated at z=exp(jwP).

Spectrum of samples

Let x(t) be a signal and $\{t_n\}$ be a sampling sequence (not necessarily uniformly spaced). The spectrum of the samples of x(t) with respect to the sequence $\{t_n\}$ is the Fourier transform of the generalized function

$$x'(t)=C\sum_{n=-\infty}^{\infty} x(t)\cdot\delta(t-t_n) \qquad \text{(i)}$$

where C is a constant (defined by a power normalization procedure).

Note: The spectrum of the samples of x(t) with respect to the sequence t_n can be also defined as the rectangular approximation of the Fourier integral, that is,

$$X'(w)=C\sum_{n=-\infty}^{\infty} x(t_n)\cdot(t_{n+1}-t_n)\exp(-jwt_n)$$

In case of uniform sampling ($t_n=nP$) both definitions reduce to

$X^*(z)$ with $z=\exp(jwP)$.

The development of this chapter is based strictly on definition (i).

Periodic Expansion of a Function

The operators S_T (in time domain) and S_W (in frequency domain) are defined by the equations:

$$S_T f(t)=\sum_{n=-\infty}^{\infty} f(t+nT) \quad \text{(T is an arbitrary time increment)}$$

$$S_W F(w)=\sum_{n=-\infty}^{\infty} F(w+nW) \quad \text{(W is an arbitrary frequency increment)}$$

2. Sampling Using Integration

Integrating samplers are often used to convert analog signals to digital words. They are simple to design, rather accurate and they have excellent noise characteristics. A typical example is the integrating analog to digital converter (ADC) and the dual slope ADC. They are used primarily for instrumentation purposes where the accuracy is more important than speed. The following theorem describes the frequency response of integrating samplers in a more general fashion: the integration period (T) is different than the sampling period (P). Interesting results are derived in case T is greater than P. After the theory has been presented, an example shows the effect of the ratio T/P on the filtering properties of the samplers.

Theorem 1

Let $\{x(t): t \in R\}$ be a signal and $\{y(nP): n \epsilon Z\}$ be a sampling sequence defined by

$$y(nP) = 1/T \int_0^T x(nP+r)dr \qquad \text{for } n=\ldots-2,-1,0,1,2,..$$

that is, the number y(nP) is the integral of the waveform x(t) starting from x(nP) up to x(nP+T) (see figure 1). Also let X(w) be the Fourier transform of x(t) and y'(t) be the sampled version of {y(nP)} defined by

$$y'(t) = P \sum_{n=-\infty}^{\infty} y(nP)\, \delta(t-nP).$$

Then the spectrum of y'(t) is given by:

$$Y'(w) = S_W X(w) f(w) \qquad (W=2\pi/P)$$

with:

$$f(w) = \frac{\sin(wT/2)}{wT/2} \exp(jwT/2)$$

(f(w) is called the sampling gain factor).

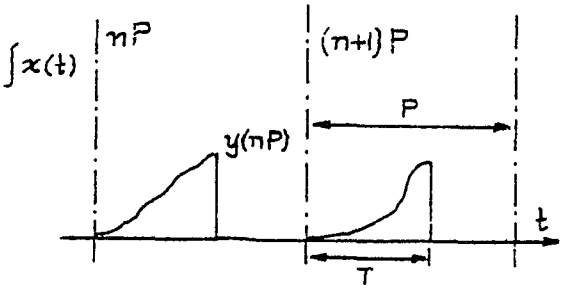


Figure 1: Sampling using integration

Proof

Let's define a new signal z(t) by taking the integral of x(t):

$$z(t) = \int_{-\infty}^{t} x(r)dr$$

Then,

$$y(nP) = 1/T \int_{0}^{T} x(nP+r)dr = 1/T\left(\int_{-\infty}^{nP+T} - \int_{-\infty}^{nP}\right) x(r)dr$$

$$= 1/T[\ z(nP+T) - z(nP)\ ].$$

Also,

$$y'(t) = P\sum_{n=-\infty}^{\infty} y(nP)\delta(t-nP) = P/T \sum_{n=-\infty}^{\infty} \Big(z(nP+T)-z(nP)\Big)\ \delta(t-nP)$$

Consider for simplicity that z(nP) are the samples of z(t) and $z_1(nP) = z(nP+T)$ are the samples of $z_1(t) = z(t+T)$ with spectra $Z(w)$ and $Z_1(w)=Z(w)\exp(jwT)$ respectively. Then, by taking the Fourier transforms of both sides we find:

$$Y'(w) = P/T\left[\sum_{n=-\infty}^{\infty} z_1(nP)\exp(-jnPw) - \sum_{n=-\infty}^{\infty} z(nP)\exp(-jnPw)\right]$$

and by using the Poisson formulas (see Appendix 1) we can find:

$$\sum_{n=-\infty}^{\infty} z(nP)\exp(-jnPw) = 1/P\cdot S_W Z(w)$$

$$\sum_{n=-\infty}^{\infty} z_1(nP)\exp(-jnPw) = 1/P\cdot S_W Z_1(w) = 1/P\cdot S_W Z(w)\exp(jwT).$$

Substituting in Y'(w),

$$Y'(w)=1/T\ S_W Z(w)\cdot\left(-1+\exp(jwT)\ \right).$$

We now define the sampling factor function f(w) by

$$f(w)=\frac{1}{jwT}(-1+\exp(jwT)\ )=\frac{1}{jwT}\exp(jwT/2)\ \left(\exp(jwT/2)-\exp(-jwT/2)\right)$$

and,

$$f(w)\ =\ \frac{\sin(\ wT/2\ )}{wT/2}\ \exp(\ jwT/2\ )$$

Replacing Z(w) by X(w)/jw (using the definition of z(t)) we derive

$$Y'(w)=S_W X(w)f(w)$$

which proves the theorem. An application of this theorem is given later in example 1.

3. Sampling Using System Observations

Theorem 2 is an extension of theorem 1. It describes a more realistic sampler with an arbitrary impulse response. Theorem 2 should be used when a more precise modelling of the sampler is needed.

Theorem 2

Let {x(t): tϵR} be a signal and h(t) be the impulse response of a causal linear, time invariant dynamic system. Also let {y(nP)} be a

sampling sequence defined by the convolution integral

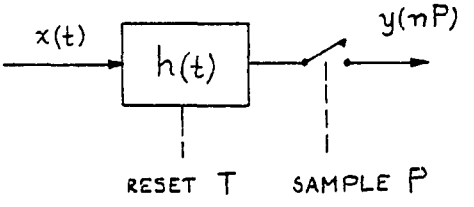
$$y(nP)=\int_0^T h(T-r)x(nP+r)dr \qquad n=\ldots-2,-1,0,1,2,\ldots$$

that is, the number y(nP) is the value of the response at t=nP+T of the system h, when it is excited by an input signal

$$\{x(t):\ nP \underset{=}{<} t \underset{=}{<} nP+T\}$$

and starting with zero initial conditions (see figure 2). Then, the spectrum of y'(t) is given by

$$Y'(w)= S_W X(w)f(w)$$



where,

$$f(w)=\exp(jwT)\ F\{h(t)(\ u(t)-u(t-T)\ )\}$$

Figure 2: Sampling using system observations

Proof

Let's define a new signal z(t) by the convolution integral

$$z(t) = \int_{-\infty}^{t} h(t-r)x(r)dr$$

Then,

$$y(nP)=\int_0^T h(T-r)x(nP+r)dr = \left(\int_{-\infty}^{T} - \int_{-\infty}^{0}\right) h(T-r)x(nP+r)dr$$

$$= (\text{call } c=nP+r) \left(\int_{-\infty}^{nP+T} - \int_{-\infty}^{nP}\right) h(nP+T-c)x(c)dc$$

$$= \int_{-\infty}^{nP+T} h(nP+T-r)x(r)dr - \int_{-\infty}^{nP} h(mP+T-r)x(r)dr$$

Define the 'shifted' impulse response g(t) by

$$g(t)=h(t+T)u(t)$$

and the corresponding Fourier transforms

$$h(t) \longleftrightarrow H(w)$$

$$g(t) \longleftrightarrow G(w).$$

Let also

$$z_h(t) \longleftrightarrow H(w)X(w)$$

$$z_g(t) \longleftrightarrow G(w)X(w).$$

Then, the above expression of y(nP) is equal to

$$y(nP)=z_h(nP+T)-z_g(nP)$$

Here we note that

$z_h(nP+T)$ is the sampled version of $H(w)X(w)\exp(jwT)$, and

$z_g(nP)$ is the sampled version of $G(w)X(w)$

The continuous signal

$$y'(t)=P\sum_{n=-\infty}^{\infty} y(nP)\tilde{\delta}(t-nP) = P\sum_{n=-\infty}^{\infty} [\, z_h(nP+T)-z_g(nP)\,]\cdot\tilde{\delta}(nP)$$

is transformed using the Poisson sum formulas and the above note to:

$$Y'(w) = S_W\{H(w)X(w)\exp(jwT)-G(w)X(w)\}$$

and equivalently,

$$Y'(w) = S_W X(w) f(w)$$

with

$$f(w) = H(w)\exp(jwT) - G(w)$$

Observe that the gain factor f(w) is the Fourier transform of the signal h(t)(u(t)-u(t-T)) shifted by T seconds. This observation conludes the proof of the theorem.

Corollary

Consider the integrating system with inpulse response h(t)=u(t)/T. Then,

$$h(t)(u(t)-u(t-T)) = (u(t)-u(t-T))/T$$

and the Fourier transform of this is

$$\frac{1}{jwT} - \frac{\exp(-jwT)}{jwT}$$

Therefore the gain factor is

$$f(w) = \frac{\exp(jwT)-1}{jwT} = \exp(jwT/2)\,\frac{2\sin(wT/2)}{wT} = \exp(jwT/2)\,\frac{\sin(wT/2)}{wT/2}$$

which agrees with the results of theorem 1

Example 1 (Application of theorem 1)

A signal x(t) is sampled using integration for T seconds with period P=5s. Let W=2π/P and assume that x(t) is bandlimited with

bandwidth B<W/2. We want to find the spectrum of the samples for T=1s, T=5s and T=15s.

Using theorem 1, $Y'(w)=S_W X(w)f(w)$ and because B<W/2, for $w < W/2$,

$$|Y'(w)|=|X(w)||f(w)| \text{ with } |f(w)|=\left|\frac{\sin(wT/2)}{wT/2}\right|$$

The sampling gain factor |f(w)| takes values

1 for w=0

0 for $w_k=2k\pi/T=kW/a$ k=+1, +2,...

where a is the ratio T/P. Also,

$$|f(W/2)|=\frac{|\sin(a\pi/2)|}{|\ a\pi/2\ |}$$

Case 1 (see figure 3a)	Case 2 (see figure 3b)	Case 3 (see figure 3c)
a=1/5	a=1	a=3
|f(w)|=0 every 5W	|f(w)|=0 every W	|f(w)|=0 every W/3
|f(W/2)|=0.984	|f(W/2)|=0.637	|f(W/2)|=0.212

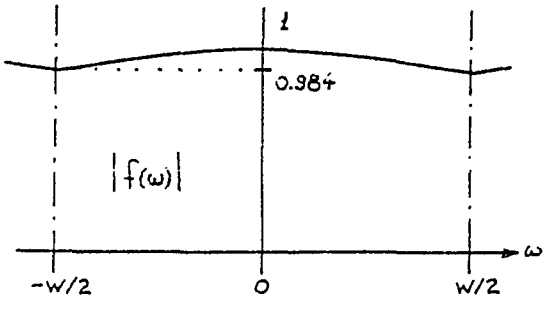


Figure 3a

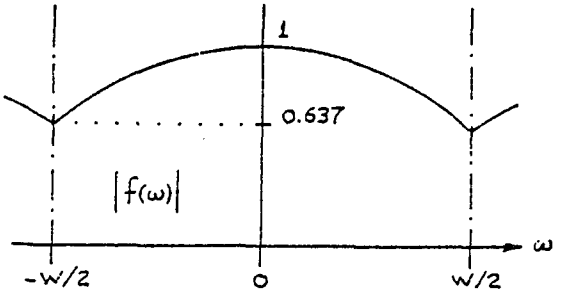


Figure 3b

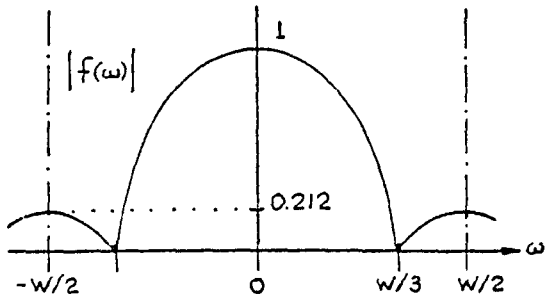


Figure 3c

Figure 3: Sampling using integration (example)

(a) a=1/5

(b) a=1

(c) a=3

Figure 4 illustrates an implementation of the samplers in cases 1 and 2. Figure 5 illustrates the sampler in case 3

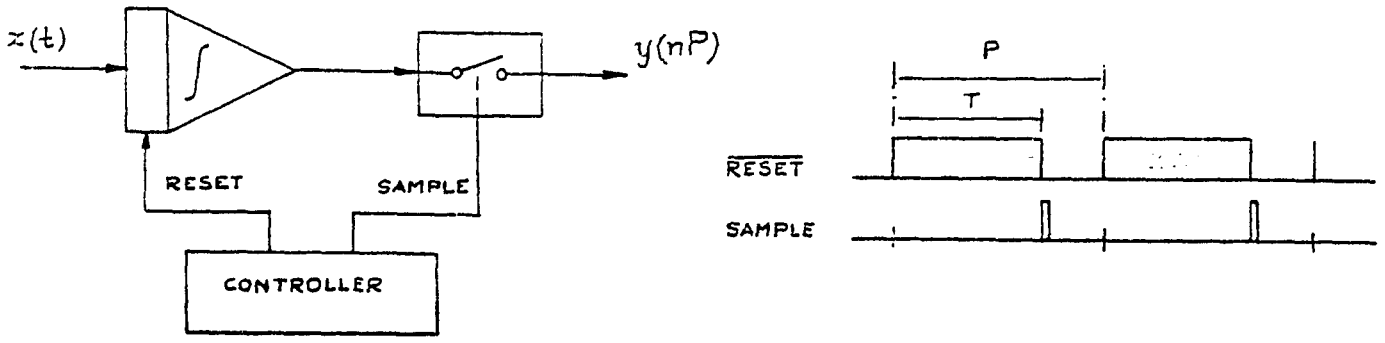


Figure 4: Implementation of integrating samplers

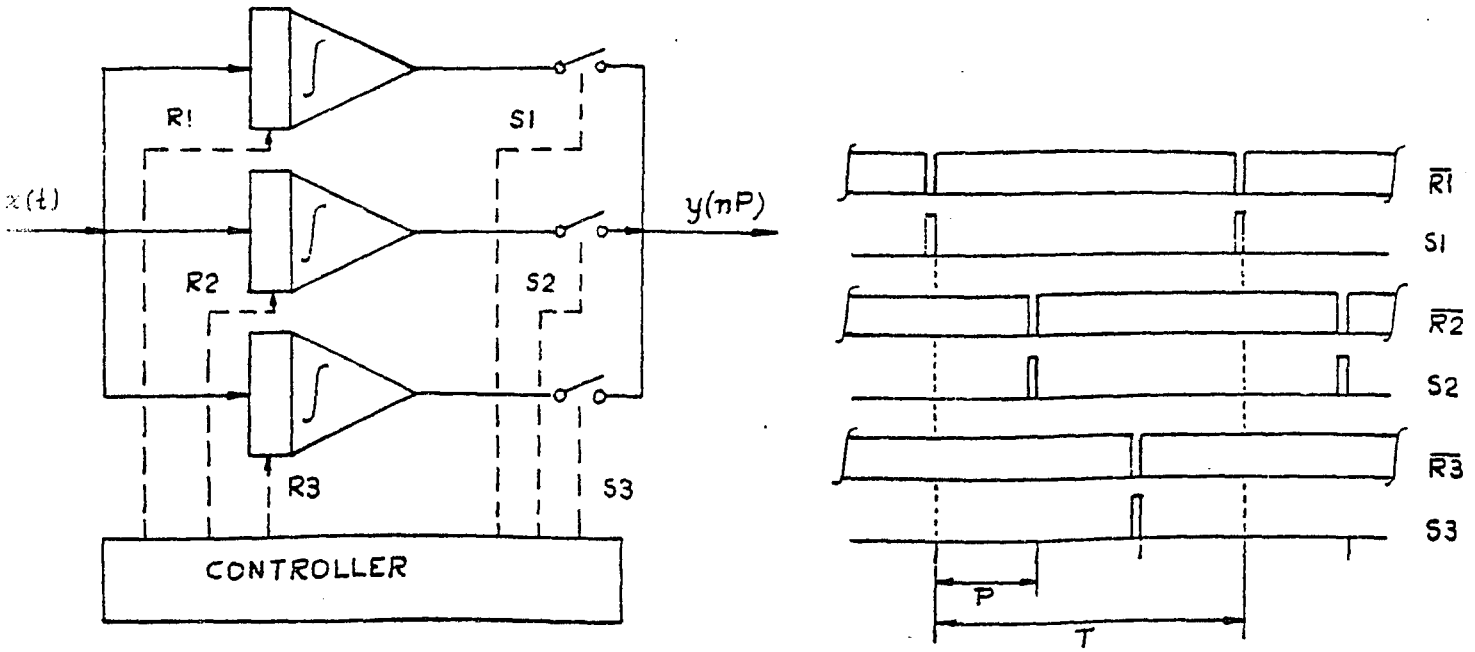


Figure 5: Implementation of overlapping integrating samplers

4. Weighted Average Burst Sampling

The weighted average sample y(nP) is defined as a linear combination of M samples x(nP+T), x(nP+2T),..., x(nP+MT) (see figure 6). For analysis purposes the following constructive formulation of the weighted average burst samples is used:

i. Define the signals $x_k(t)$ by

$$x_k(t)=x(t+kT) \quad \text{for } k=1,2,\ldots,M$$

ii. Define the signal y(t) as the linear combination

$$y(t)=a_1x_1(t)+a_2x_2(t)+\ldots+a_Mx_M(t)$$

where a_i are arbitrary real numbers.

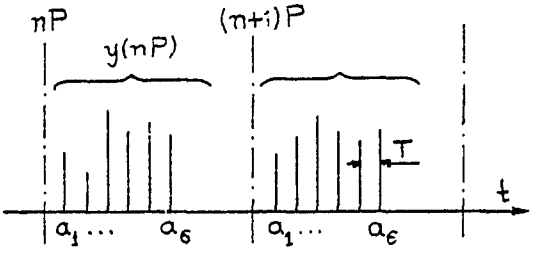


Figure 6: Weighted average burst sampling

Then, the uniform samples of y(t) with period P (considered as elements of a set) are the weighted burst samples of x(t).
Note that MT is not required to be less than P. The case MT>P is illustrated later in example 2.

Theorem 3

The spectrum of the sampled version of y(t) (called the weighted burst sampling spectrum) is

$$Y'(w) = S_W X(w)f(w)$$

where X(w) is the spectrum of x(t), W= 2π/P and the sampling factor function f is given by:

$$f(w) = a_1\exp(jwT) + \ldots + a_M\exp(jwMT)$$

Proof

The sampled version of y(t) is

$$y'(t)=P\sum_{n=-\infty}^{\infty} y(nP)\,\delta(t-nP)$$

with

$$y(nP)=a_1 x(nP+T)+\ldots+a_M x(nP+MT)$$

The spectrum of y'(t) is

$$Y'(w)=P\sum_{n=-\infty}^{\infty} y(nP)\cdot\exp(-jwnP) \qquad \text{(i)}$$

The samples y(nP) have been taken from y(t) with spectrum

$$Y(w) = (a_1\exp(jwT)+\ldots+ a_M\exp(jwMT))\ X(w)$$

or

$$Y(w)=f(w)X(w) \qquad \text{(ii)}$$

Now we apply the Poisson lemma for (i):

$$Y'(w)=\sum_{n=-\infty}^{\infty} Y(w+nW) = S_W Y(w)$$

and by using (ii) the proof is complete.

5. **Average Burst Sampling**

For the weighted burst sampling it was necessary to have equally spaced samples at the beginning of every period P. Theorem 4 does not use this restriction; it deals with samples taken at

$$T_1, T_2, T_3, \ldots, T_M$$

relative to the the beginning of the period.

Theorem 4

Let x(t) be a signal and {y(nP)} be the sequence

$$y(nP)=a_1x(nP+T_1)+\ldots+a_Mx(nP+T_M)$$

where $(a_1,\ldots,a_M)$ and $(T_1,\ldots,T_M)$ are given constants. Then the spectrum of the samples y(nP) is given by

$$Y'(w)=S_WX(w)f(w)$$

where

$$f(w)=a_1\exp(jwT_1)+\ldots+a_M\exp(jwT_M)$$

Proof

Consider the signals

$$x_k(t)=x(t+T_k) \qquad k=1,2,\ldots,M \qquad t\in R.$$

Then the set {y(nP)} is equal to the set

$$\{a_1x_1(nP)+\ldots+a_Mx_M(nP)\}$$

Also,

$$y'(t)=P\sum_{n=-\infty}^{\infty}y(nP)\delta(t-nP), \quad Y'(w)=P\sum_{n=-\infty}^{\infty}y(nP)\exp(-jwP)$$

By considering the pair

$$y(t)=a_1x_1(t)+\ldots+a_Mx_M(t) \quad \longleftrightarrow \quad Y(w)=a_1X_1(w)+\ldots+a_MX_M(w)$$

we can apply Poisson's sum formula to derive

$$P\sum_{n=-\infty}^{\infty} y(nP)\exp(-jnwP) = S_W\left(a_1X_1(w)+\ldots+a_MX_M(w)\right)$$

But

$$X_k(w) = X(w)\exp(jwT_k)$$

by the definition of x_k's. Therefore,

$$Y'(w)=S_W X(w)f(w) \quad \text{with} \quad f(w)=a_1\exp(jwT_1)+\ldots+a_M\exp(jwT_M)$$

which completes the proof.

Corollary 1

Let $T_k=kT$ for $k=1,\ldots,M$. The theorem 4 yields theorem 3.

Corollary 2

Let $a_k = 1/M$ for $k=1,\ldots,M$. Then the spectrum of the samples is

$$Y'(w) = S_W X(w)f(w)$$

with

$$f(w)= \frac{\sin(wMT/2)}{M\sin(wT/2)} \exp(jw \frac{M+1}{2} T)$$

Proof

$$f(w)=(\exp(jwT)+\ldots+\exp(jMwT))/M =(z^1+ \ldots + z^M)/M$$

(where $z=\exp(jwT)$)

$$=\frac{1}{M}(z^{M+1}-z)/(z-1) = \frac{1}{M}z^{(M+1)/2} \frac{z^{M/2} - z^{-M/2}}{z^{1/2} -z^{-1/2}}$$

from which f(w) can be readily derived.

Example 2 (Application of theorem 3)

A signal x(t) is sampled using the weighted average burst sampling technique. The sampling period is P=5s and the burst period is T=1s. We assume that x(t) is bandlimited with bandwidth B<W/2 (W=2π/P) and we want to find the spectrum of the samples in the cases M=3, 5 and 15, using weights $a_i=1/M$

Using theorem 3, $Y'(w)=S_W Y(w)f(w)$ with

$$f(w)=(\exp(jwT)+\ldots+\exp(jwMT))/M$$

$$=\exp(j\frac{M+1}{2}wT)\frac{\sin(MwT/2)}{M\sin(wT/2)}$$

The sampling gain factor $|f(w)|$ takes values 1 for w=0 and 0 for $w=2k\pi/MT=kW/a$ (k=1,2,...) where a is the ratio MT/P. Also, $|f(w)|$ is periodic with period $2\pi/T$.

Case 1 (see figure 7a)

a=3/5

$|f(w)|=0$ every $5W/3$

$$|f(W/2)|=\left|\frac{\sin(3\pi/10)}{3\sin(\pi/10)}\right|=0.873$$

Case 2 (see figure 7b)

a=1

$|f(w)|=0$ every W

$$|f(W/2)|=\left|\frac{\sin(5\pi/10)}{5\sin(\pi/10)}\right|=0.647$$

Case 3 (see figure 7c)

a=3

$|f(w)|=0$ every $W/3$

$$|f(W/2)|=\left|\frac{\sin(15\pi/10)}{15\sin(\pi/10)}\right|=0.216$$

Figure 8 illustrates an implementation of the samplers in cases 1 and 2. In case 3, the sampler can be implemented using three feedback summers (similar to the ones in figure 8), or by cascading 15 delays (figure 9).

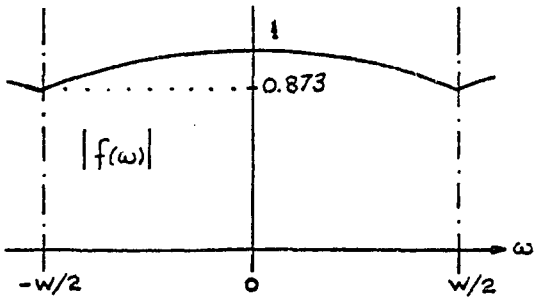


Figure 7a

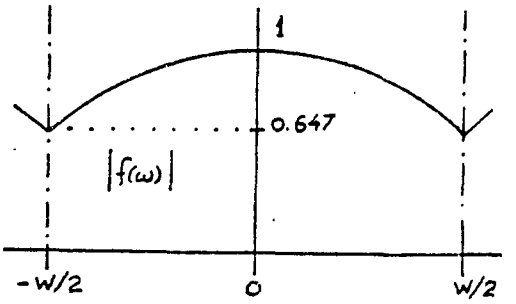


Figure 7b

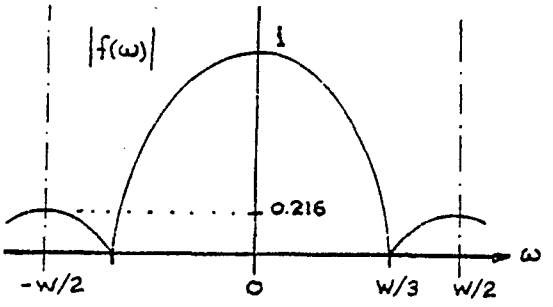


Figure 7c

Figure 7: Weighted average burst sampling example

(a) a=3/5

(b) a=1

(c) a=3

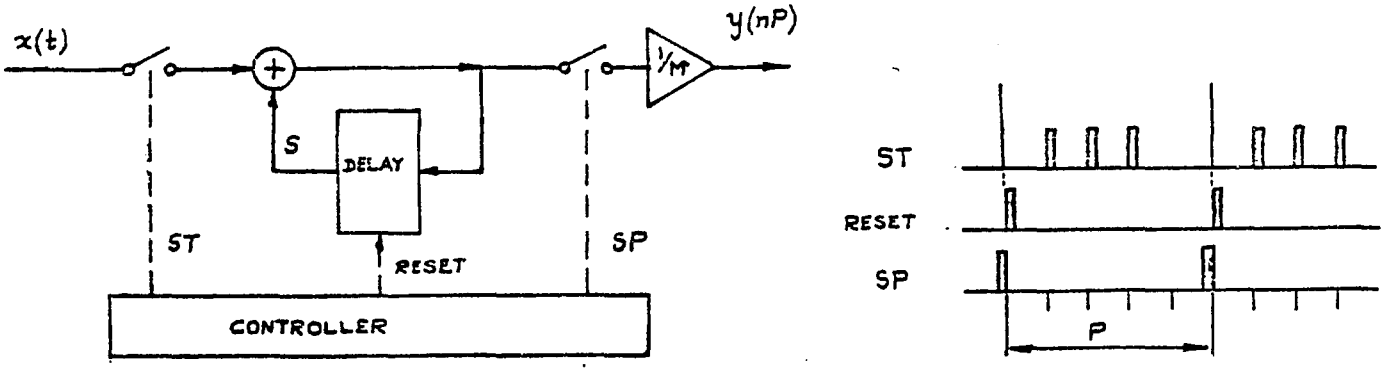


Figure 8: Implementation of weighted burst samplers

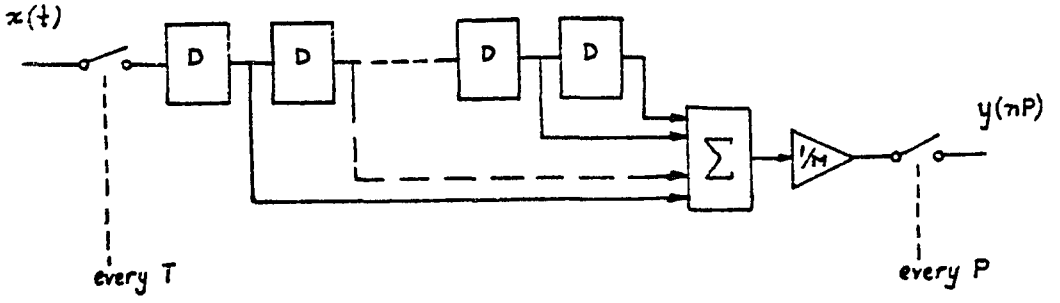


Figure 9: Implementation of overlapping burst samplers

6. Periodic Burst Sampling

Theorem 5 deals with the spectrum of a 'burst of samples' taken every P time units. More particularly, we deal with a uniform burst of samples, where within the burst, the samples are taken every T time units (see figure 10). The sampling sequence looks like

$$\ldots\ x(nP+T),\ x(nP+2T),\ \ldots\ x(nP+MT),\ x(\ (n+1)P+T\ ),\ldots$$

Note that MT is not necessarily less than or equal to P. The proof is still true for MT>P. This is illustrated later in example 3.

Theorem 5

The spectrum of the samples $\{x(nP+mT):\ n\in Z,\ m=1,2,\ldots,M\}$ is given by the formula

$$Y'(w) = \sum_{n=-\infty}^{\infty} X(w+nW)\ f(n;W,T)$$

where

$$f(n;W,T) = \exp\left(jn\frac{M+1}{2}WT\right)\ \frac{\sin(nWMT/2)}{\sin(nWT/2)}$$

and $W=2\pi/P$

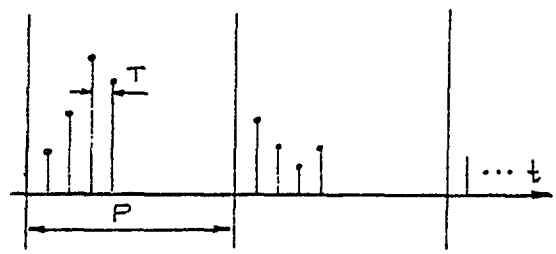


Figure 10: Periodic burst sampling

Proof

Let y'(t) be the sampled version of a function of the samples of x(t). Then,

$$y'(t) = P \sum_{n=-\infty}^{\infty} \sum_{m=1}^{M} x(nP+mT)\delta(t-nP-mT)$$

The Fourier transform of y'(t) is

$$Y'(w)= \sum_{m=1}^{M} \left(P\sum_{n=-\infty}^{\infty} x_m(nP)\exp(-jwnP)\right)\exp(-jwmT)$$

where $x_m(t)$ is defined to be $x(t+mT)$.

Now, we apply Poisson Sum formulas for the summation with respect to n. This yields

$$Y'(w)= \sum_{m=1}^{M} \left(S_W X_m(w)\right)\exp(-jwmT)$$

But

$$X_m(w) = X(w)\exp(jwmT)$$

from the definition of $x_m(t)$. Then, expanding the S_W operator,

$$Y'(w) = \sum_{m=1}^{M} \left(\sum_{n=-\infty}^{\infty} X(w+nW)\exp(\ j(w-nW)mT\ )\right)\exp(-jwmT)$$

But the product of the two exponentials equals exp(-jnWmT) (which does not depend on w). Interchanging the order of the summations,

$$Y'(w) = \sum_{n=-\infty}^{\infty} X(w+nW)\ f(n;W,T)$$

where

$$f(n;W,T)=\exp(-jnWT)+\exp(-jnW2T)+\ldots+\exp(-jnWMT)$$

Call z=exp(-jnWT). Then,

$$f(n;W,T)=z+z^2+\ldots+z^M = \frac{z^{M+1}-z}{z-1} = \exp(jn\frac{M+1}{2}WT)\ \frac{\sin(nWMT/2)}{\sin(nWT/2)}$$

which completes the proof of the theorem.

Corollary 1

Let P=T and M=1. Then WT=2π. Also,

$$\exp(jn(M+1)WT/2) = (-1)^{n(M+1)} = 1$$

and

$$\lim_{WMT \to 2\pi} f(n;W,T) = \lim_{WMT \to 2\pi} \frac{\sin(nWMT/2)}{\sin(nWT/2)} = 1$$

which gives,

$$Y'(w) = \sum_{n=-\infty}^{\infty} X(w+nW)$$

This last statement is the spectrum of the uniform samples (every P).

Corollary 2

Let MT=P and M be any arbitrary integer number. We will prove that theorem 5 results in the sampling theorem:

$$Y'(w) = M \sum_{n=-\infty}^{\infty} X(w+2n\pi/T)$$

To prove that, it suffices to prove that

$$f(n;W,T) = \exp(-jnWT) + \exp(-jnW2T) + \ldots + \exp(-jnWMT)$$

is equal to M for n=0, M, 2M, ... and is equal to 0 for the remaining integer values of n. Because MT=P, WT=2π/M. Also, for n=kM,

$$\exp(jnWT) = \exp(-jkM\ 2\pi/M) = 1.$$

This proves the first assertion. To prove the second, we use:

$$f(n;W,T) = z(1+z+\ldots+z^{M-1}) = z(z^{M} - 1)/(z-1)$$

where $z=\exp(-j2\pi n/M)=1$ if $n=kM$. But $z^M=\exp(-j2\pi)=1$. This completes the proof.

Example 3 (Application of theorem 5)

A burst of M samples every T=1s is collected from a signal x(t). The sampling period is P=5s. The signal x(t) is bandlimited with bandwidth $B<W/2$ ($W=2\pi/P$). We want to describe the spectrum of the samples for M=3,4,5 and 15

Using the results of theorem 5,

$$Y'(w)=M\sum_{-\infty}^{\infty}X(w+nW)f(n;W,T)$$

with

$$|f(n;W,T)|=\left|\frac{\sin(nWMT/2)}{M\sin(nWT/2)}\right|$$

for n=0, $|f(n;W,T)|=1$

n=1, $|f(n;W,T)|=\left|\frac{\sin(M\pi/5)}{M\sin(\pi/5)}\right|$

n=k, $|f(n;W,T)|=\left|\frac{\sin(kM\pi/5)}{M\sin(k\pi/5)}\right|$

Also, $f(n;W,T)$ is the same as $f(n+5l;W,T)$ for all integers l. In this example it is important to note that the sampling gain factor remains constant over the bands $(-kW/2,kW/2)$ for all k.

Case 1 (see figure 11a)	Case 2 (see figure 11b)
M=3	M=4
$\|f(0;W,T)\|=1.000$	$\|f(0;W,T)\|=1.000$
$\|f(1;W,T)\|=0.539$	$\|f(1;W,T)\|=0.250$
$\|f(2;W,T)\|=0.392$	$\|f(2;W,T)\|=0.250$
$\|f(3;W,T)\|=0.392$	$\|f(3;W,T)\|=0.250$
$\|f(4;W,T)\|=0.539$	$\|f(4;W,T)\|=0.250$
$\|f(5;W,T)\|=1.000$	$\|f(5;W,T)\|=1.000$

Case 3 (see figure 11c)

M=5,15,...,5i,...

|f(0;W,T)|=1

|f(i;W,T)|=0 for i=1,2,...

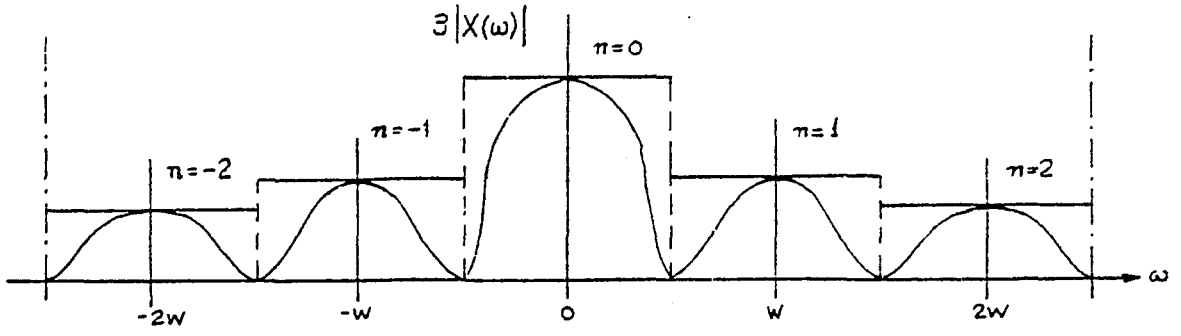


Figure 11a

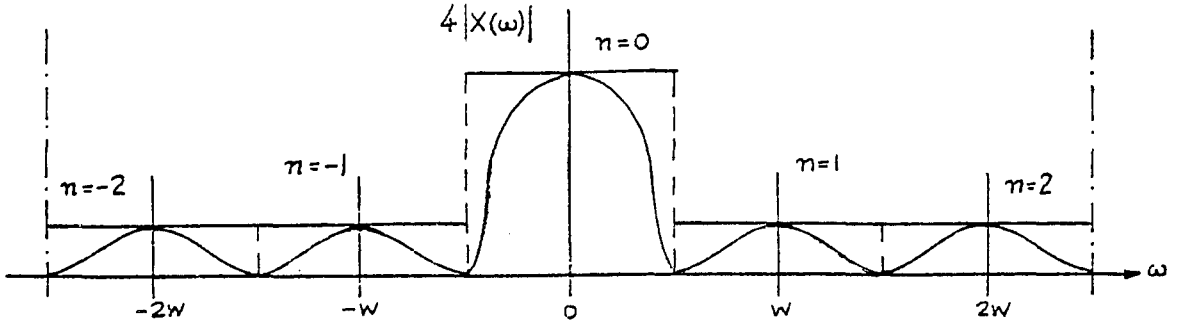


Figure 11b

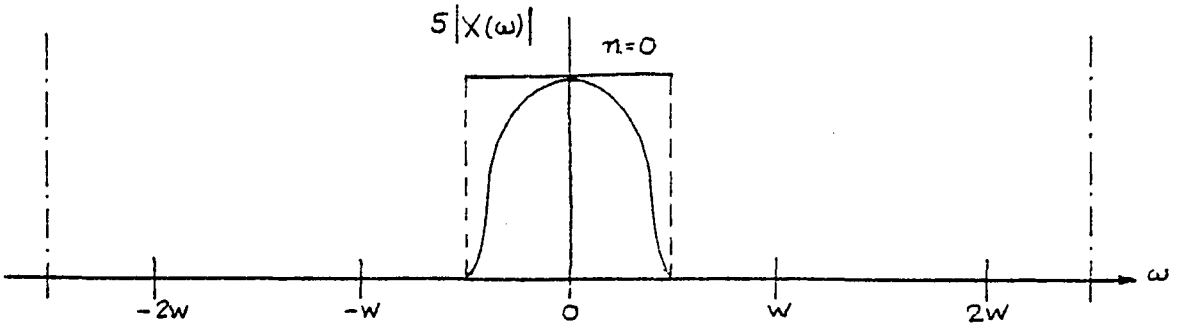


Figure 11c

Figure 11: Periodic Burst Sampling (example)

(a) M=3, (b) M=4, (c) M=5i i=1,2,3...

7. Filtering by Weighted Average Burst Sampling

Theorem 3 indicates that the spectrum of the weighted burst samples depends on the frequency characteristics of the sampling factor f(w). Because f(w) is a function of the weights a_i, the shape of the spectrum of Y'(w) can be modified by choosing the weights properly. The magnitude of the sampling factor f(w) is calculated below.

$$|f(w)| = |z(a_1+a_2z+\ldots+a_Mz)^{M-1}| \qquad z=\exp(jwT)$$

$$= c|z-z_1||z-z_2|| \ \ldots \ |z-z_{M-1}|$$

where z_i is the i-th root of the equation $a_1+a_2z+\ldots+a_Mz^{M-1}=0$

If the w_i's corresponding to the z_i's are placed on the unit circle, (see figure 12) the value of f(w) depends on the distances of w from $w_1, w_2, \ldots w_{M-1}$.

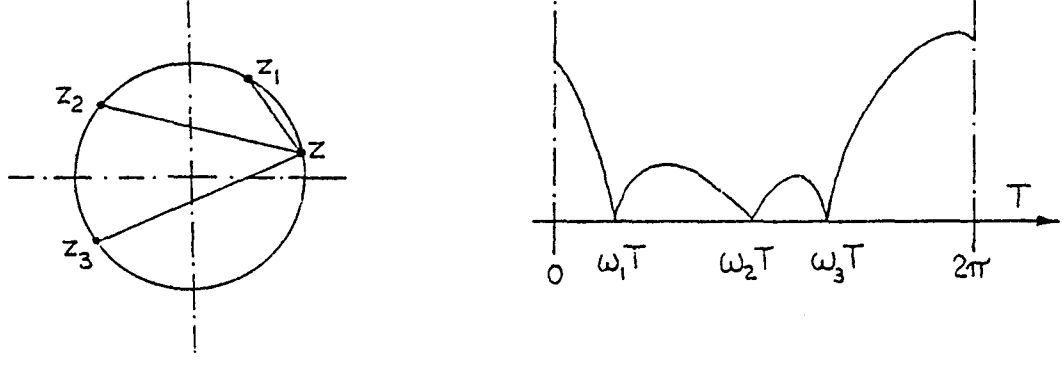


Figure 12: Calculation of the magnitude of the sampling factor

In the following we study the various possibilities for f(w).

Uniform f(w)

In this case, f(w) has equal weights for all the samples. Then,

$$f(w) = \frac{1}{M}(z^1+\ldots+z^M) = \frac{z}{M}\,\frac{z^M-1}{z-1}$$

and

$$|f(w)| = \left|\frac{\sin(wMT/2)}{M\sin(wT/2)}\right|$$

Figures 13 and 14 (graph 3) illustrate the uniform gain factor for M=8 and 20 respectively. For comparison purposes, the function

$$\left|\frac{\sin(wMT/2)}{wMT/2}\right|$$

(graph 4) is also drawn.

Binomial f(w)

In this case, f(w) has a multiple zero at z=-1+j0. Then,

$$f(w) = z\left[\frac{z+1}{2}\right]^{M-1} = 2^{-(M-1)} \sum_{m=1}^{M} \begin{bmatrix} M-1 \\ m \end{bmatrix} z^m = \left[\begin{bmatrix} M-1 \\ 1 \end{bmatrix} z + \ldots + \begin{bmatrix} M-1 \\ M-1 \end{bmatrix} z^M\right] \frac{1}{2^{M-1}}$$

and

$$|f(w)| = \left|\frac{1+\exp(jwT)}{2}\right|^{M-1} = |\cos(wT/2)|^{M-1}$$

Figures 13 and 14 (graph 1) illustrate the binomial gain factor; note that this function does not have any side lobes but the width of main lobe is more than all the other gain functions.

Triangular f(w)

For M equal to an odd number, the triangular sampling factor function is defined by

$$f(w) = z\left(1+z+z^2+\ldots+z^{(M-1)/2}\right)^2 \left[\frac{2}{M+1}\right]^2$$

and $$|f(w)| = \left| \frac{\sin(\frac{M+1}{2} \frac{wT}{2})}{\frac{M+1}{2} \sin(\frac{wT}{2})} \right|^2$$

This gain function is also illustrated in figures 13 and 14 (graph 2).

Equal ripple f(w)

The equal ripple sampling function has equal side lobes around the main lobe. One possible realization is based on Tchebyscheff polynomials. The theory and the selection of the coefficients a_i of the sampling factor function follows. Finally, an example illustrates the calculation steps to obtain a_i's

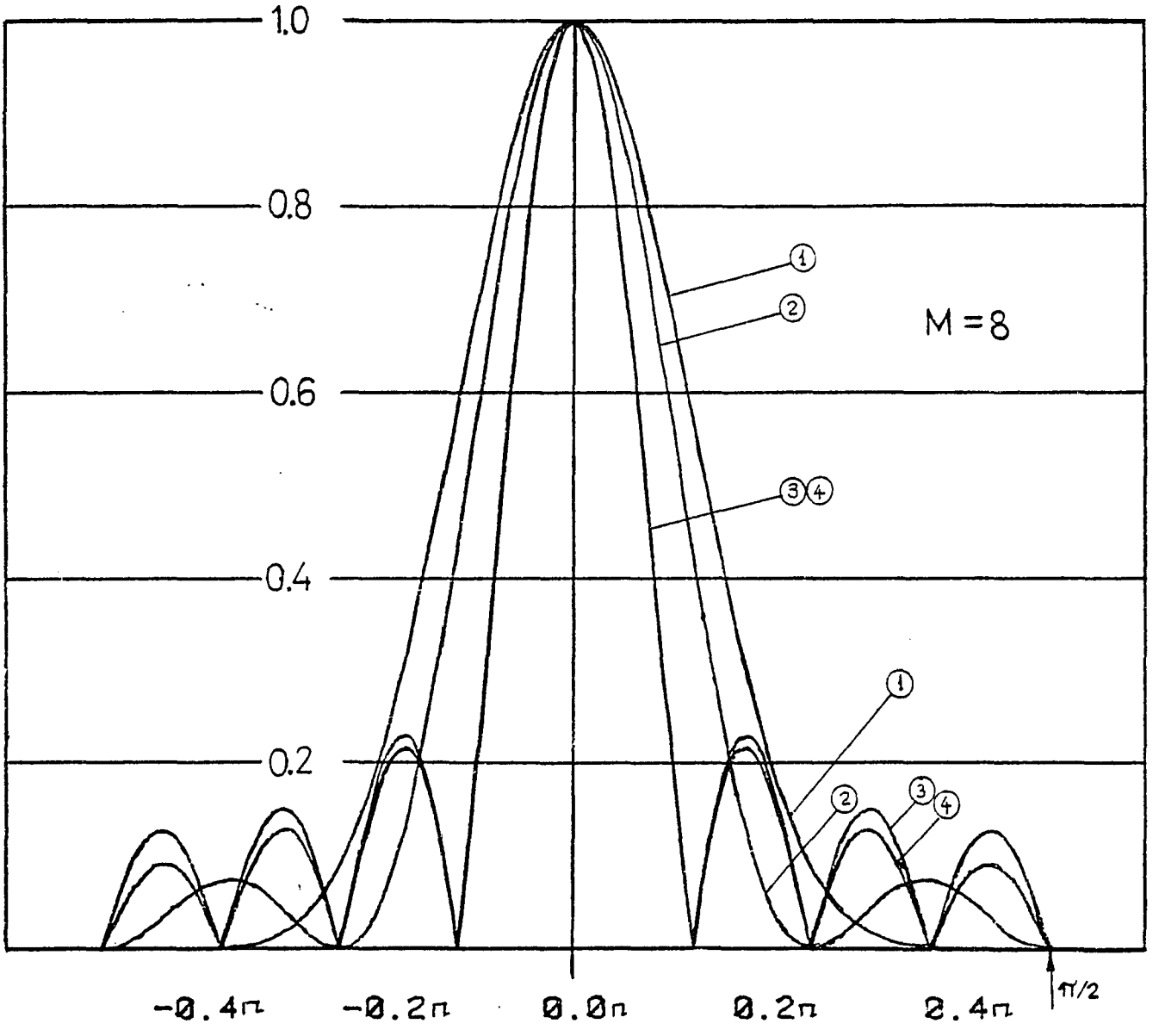


Figure 13: Comparison of sampling gain functions for M=8

(1) $|\cos(x)|^{M-1}$ Binomial gain factor

(2) $\left|\frac{\sin(Mx/2)}{Mx/2}\right|^2$ Triangular gain factor

(3) $\left|\frac{\sin(Mx)}{M\sin(x)}\right|$ Uniform gain factor

(4) $\left|\frac{\sin(Mx)}{Mx}\right|$ Continuous gain (for comparison)

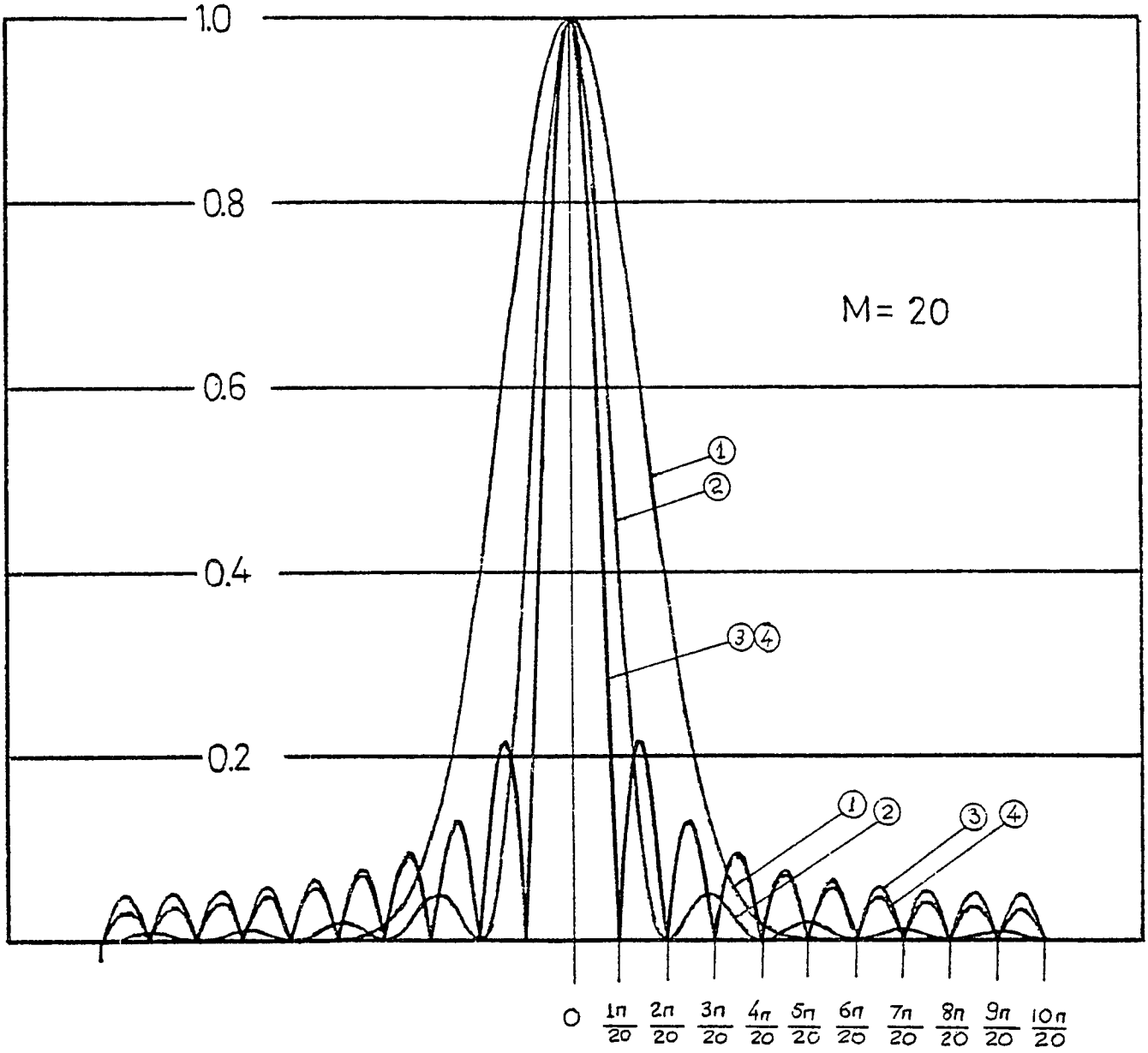


Figure 14: Comparison of sampling gain functions for M=20

(1) $|\cos(x)|^{M-1}$ Binomial gain factor

(2) $\left|\frac{\sin(Mx/2)}{Mx/2}\right|^2$ Triangular gain factor

(3) $\left|\frac{\sin(Mx)}{M\sin(x)}\right|$ Uniform gain factor

(4) $\left|\frac{\sin(Mx)}{Mx}\right|$ Continuous gain (for comparison)

8. Designing Equal Ripple Samplers

Preliminaries

Let $T_{M-1}(x)$ be the (M-1)th order Tchebyscheff polynomial with M even. Consider also the mapping

$$y=\arccos(x/x_o) \iff x=x_o\cos(y)$$

where x_o is an arbitrary constant. Then $T_{M-1}(x(y))$ has maxima and minima at

$$y_k=k\pi \qquad k=0,1,2,...$$

with values

$$T_{M-1}(x(y_k))=T_{M-1}(\pm x_o)=\pm R$$

(see figure 15). Therefore there exist a one-to-one correspondence between x_o and R. Given an R, x_o can be found by solving the equation (see appendix Ib)

$$\cosh((M-1)\operatorname{arcosh}(x_o))=R$$

$$x_o=1/2\left[(R+\sqrt{R^2+1})^{1/(M-1)}+(R-\sqrt{R^2-1})^{1/(M-1)}\right]$$

Futhermore, $T_{M-1}(x)$ has M-1 roots given by (see appendix Ib)

$$x_k=\cos\frac{2k-1}{2(M-1)}\pi \qquad k=1,2,...$$

so, $T_{M-1}(x(y))$ becomes 0 when $y=y_k$ with

$$y_k=\arccos(x_k/x_o) \qquad k=1,2,...$$

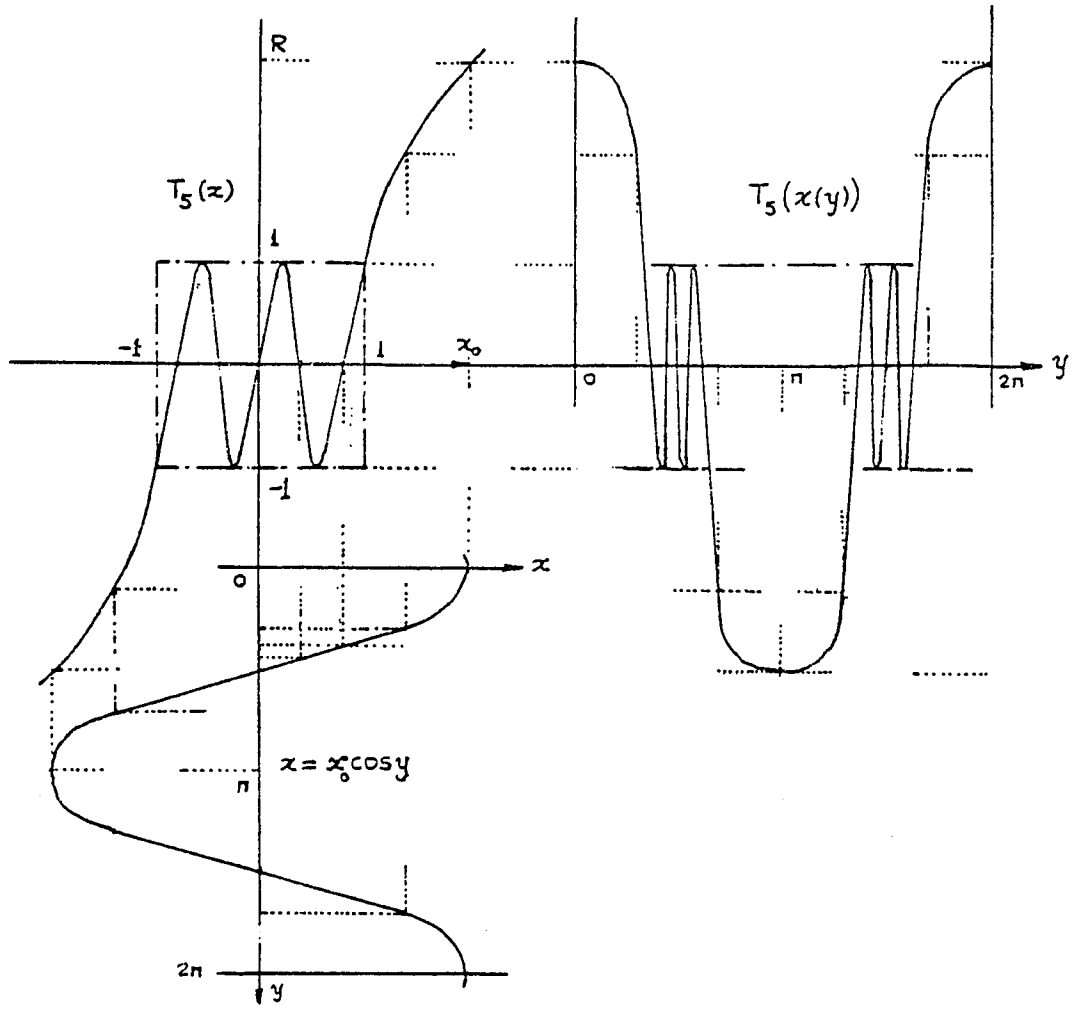


Figure 15: Using the mapping $x=x_o \cos y$ with the Tchebyscheff polynomial $T_5(x)$

The following two lemmas are used for the design of the equal ripple sampling factor.

Lemma 1

The sampling factor

$$f(w)=a_1 z+a_2 z^2+\ldots+a_M z^M \quad \text{with } z=\exp(jwT) \text{ (or } \exp(-jnWT) \text{)}$$

under the assumptions that

(1) M =even integer

(2) $a_k=a_{M-k+1}$ for k=1,2,...

(2) $a_k=a_{M-k+1}$ for k=1,2,...

can be transformed to

(3) $f(w)=2z^{(M-1)/2}(a_1\cos(M-1)y+a_2\cos(M-3)y+\ldots+a_{M/2}\cos y)$

where y=wT/2 (or nWT/2)

Proof

$$f(w)=z(a_1+a_2z^{1}+\ldots+a_Mz^{M-1}).$$

Consider the sums

$$s_k=a_kz^{k-1}+a_{M-k+1}z^{M-k} \quad \text{for } k=1,2,\ldots,M/2$$

Then,

$$f(w)=z(s_1+s_2+\ldots+s_{M/2})$$

But because of the second assumption,

$$s_k=a_k(z^{k-1}+z^{M-k})=a_kz^{m}(z^{k-1-m}+z^{M-k-m})$$

for all m. Now require k-1-m=-(M-k-m). This gives

$$m=(M-1)/2 \quad \text{and} \quad k-1-m=-(M-2k+1)/2$$

Therefore,

$$s_k=a_kz^{(M-1)/2}(z^{k-1-m}+z^{-(k-1-m)})=2a_kz^{(M-1)/2}\cos(M-2k+1)y$$

Summing up all s_k's we prove (3).

Lemma 2

Let f(w) be the sampling factor given in lemma 1, equation (3). Then the mapping

$$y=\arccos(x/x_o) \iff x=x_o\cos y$$

(for any x_o) gives

$$f(w)=2z^{(M-1)/2}(a_1T_{M-1}(x/x_o)+a_2T_{M-2}(x/x_o)+\ldots+a_{M/2}T_1(x/x_o))$$

Proof

Using the definition of the Tchebyscheff polynomials and the result of lemma 1, the above expression is readily obtained.

Summary of the design procedure

The objective here is to find the coefficients $a_1, a_2, \ldots a_{M/2}$ of a sampling factor f(w) of even number of terms and with

$$a_k = a_{M-k+1} \quad \text{for } k=1,2,\ldots,M/2$$

which has the properties:

(1) All side lobes have the same maximum

(2) The main lobes are R times the side lobes.

The coefficients a_k are found by the following procedure

(a) Give R, find x using

$$x_o = 1/2\ (R+\sqrt{R^2+1})^{1/(M-1)} + (R-\sqrt{R^2-1})^{1/(M-1)}$$

(b) Require that

$$T_{M-1}(x) = a_1T_{M-1}(x/x_o)+\ldots+a_{M/2}T_1(x/x_o)$$

and using the expansions of appendix Ib determine the a_i's

Then, $f(w) = T_{M-1}(x(y))$ with $y=wT/2$ (or $nWT/2$)

The following example illustrates the usage of the above procedure in designing an equal ripple sampling factor based on Tchebuscheff polynomials.

Example 4 (Applying the design procedure)

A signal x(t) is sampled using averaged weighted burst sampling with M=8, T=4s and P=5s. It is assumed that x(t) is bandlimited with bandwidth less than W/2 (W=2π/P). Select the weighting coefficients $a_1, a_2 \ldots$ for obtaining equal ripple lobes with R=50.

The spectrum is

$$Y'(w)=S_W X(w) f(w)$$

$$f(w)=a_1 z+\ldots+a_8 z^8 \qquad z=\exp(jwT)$$

Let $a_1=a_8$, $a_2=a_7$, $a_3=a_6$ and $a_4=a_5$. Then, using lemma 1,

$$f(w)=2z^{7/2}(a_1\cos 7y+a_2\cos 5y+a_3\cos 3y+a_4\cos y)$$

with y=wT/2.

Following the design procedure,

$$x_o=1/2\ (50+\sqrt{2501})^{1/7}+(50-\sqrt{2499})^{1/7})=1.2243$$

$$T_7(x)=a_1 T_7(x/x_o)+a_2 T_5(x/x_o)+a_3 T_3(x/x_o)+a_4 T_1(x/x_o)$$

and using the formulas from appendix Ib,

$$64x^7-112x^5+52x^3-7x = a_1 64(x/x_o)^7+(-112a_1+16a_2)(x/x_o)^5+$$

$$+(56a_1-20a_2+4a_3)(x/x_o)^3+(-7a_1+5a_2-3a_3+a_4)(x/x_o)$$

Solving the system

$$64a_1 = 64x_o^7$$

$$-112a_1+16a_2 = -112x_o^5$$

$$56a_1-20a_2+4a_3 = 56x_o^3$$

$$-7a_1+5a_2-3a_3+a_4 = -7x_o$$

we find

$$a_1=4.123,\ a_2=9.6,\ a_3=16,\ a_4=20$$

The following table uses the formulas

$$x_k=\cos(2k+1)\pi/14$$

$$y_k=\arccos(x_k/x_o)$$

$$w_k=2y_k/T$$

$$x_o=1.2243$$

to calculate the zeros of $T_7(x(y))$

Table 1: Calculation of the zeros of $T_7(x(y))$

k	x_k	y_k ( )	w_k ( )
0	0.975	37	18.5
1	0.782	50	25.0
2	0.434	69	34.5
3	0.000	90	45.0
4	-0.434	111	55.5
5	-0.782	130	65.0
6	-0.975	143	71.5

Figure 16 shows a the sampling gain factor for several values of R. Note that as the ratio R increases, so does the width of the main lobe. Because MT>P, a special implementation of the sampler is required. The timing is illustrated with figure 17.

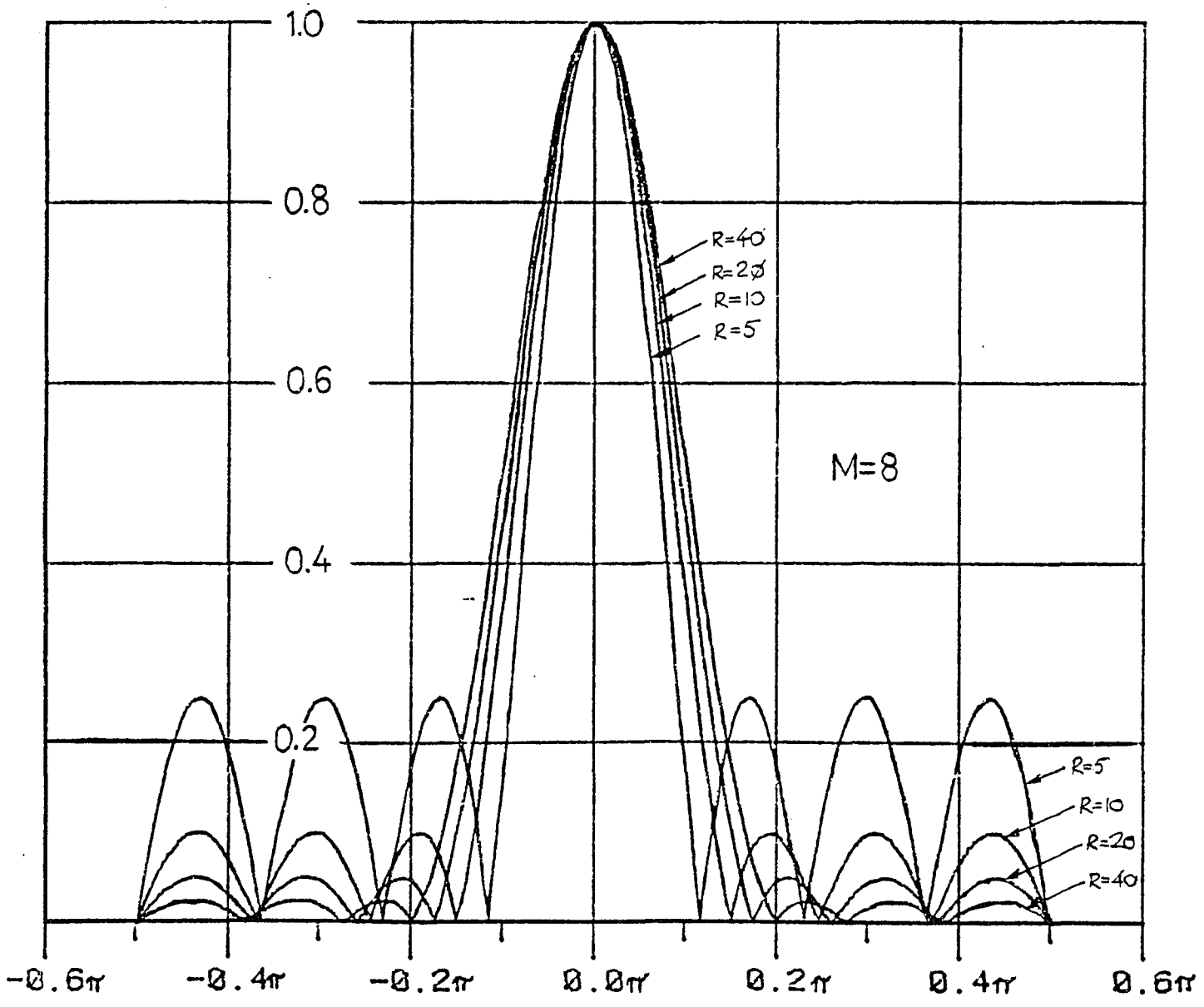


Figure 16: Tchebyscheff sampling gain factors

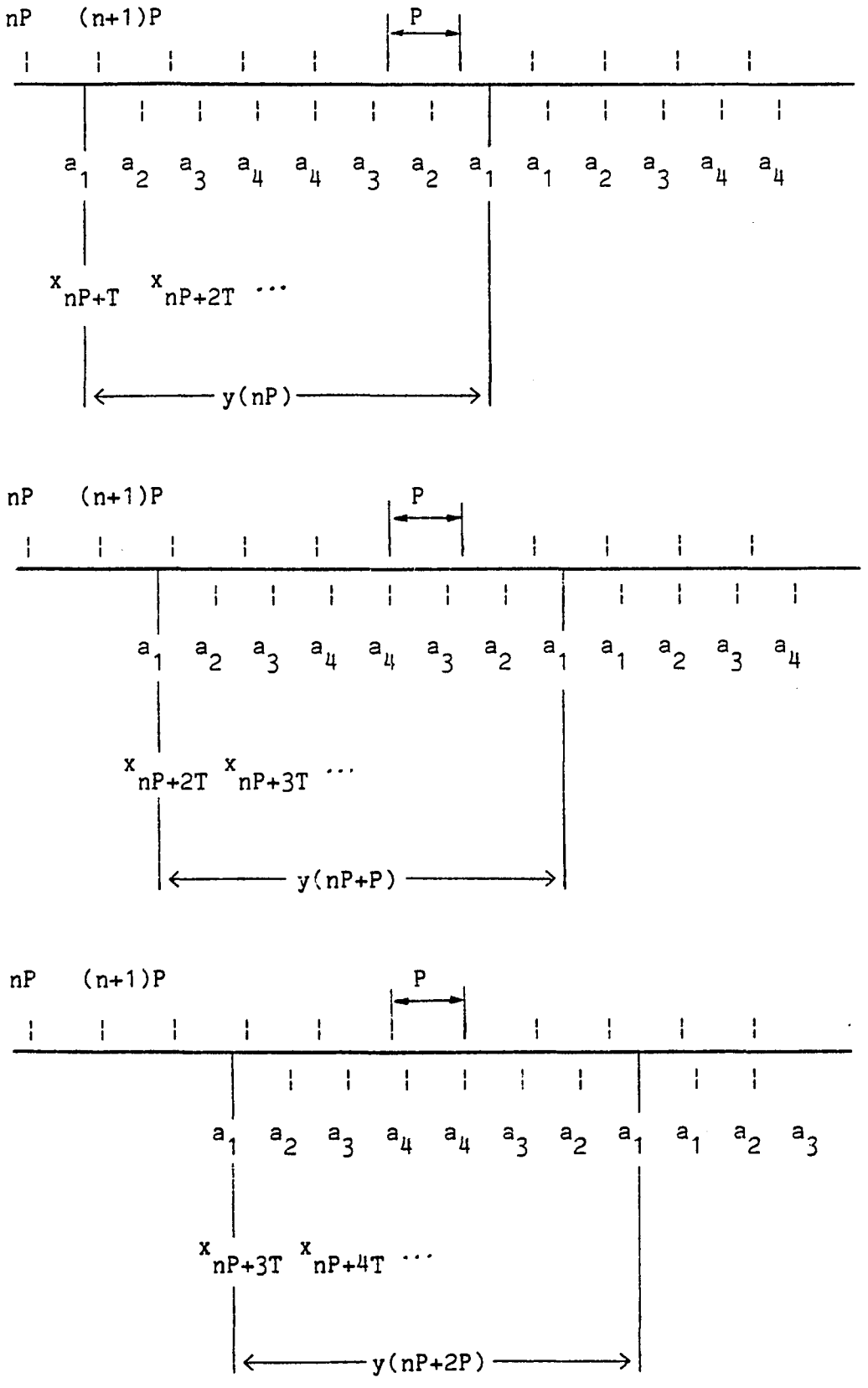


Figure 17: Timing of overlapping Tchebyscheff sampling

9. State Space Models of Cyclicly Sampled Systems

This section provides a 'system theoretic' view of the sampling schemes discussed in the previous sections. Here we use state space techniques to model a class of periodically sampled systems. The section consists of two theorems: Theorem 6 proves that a periodically sampled system is equivalent to a time invariant state equation with time varying output equation. Theorem 7 proves that a periodically sampled system is equivalent to an augmented time invariant state model.

The sampling sequences we deal with are of the form (see figure 18):

$$\{y(nP+T_m) \text{ with } n=0,1,\ldots \text{ and } m=1,2,\ldots,M\} \qquad (i)$$

This is a sequence of patterns consisting of M irregularly spaced samples. For simplicity we assume that T_m is less than P.

We also assume that the above sequence is sampled from a linear, time invariant system with state equations:

$$dx/dt=Fx \quad \text{with } x(0)=b$$
$$y(t) =cx$$

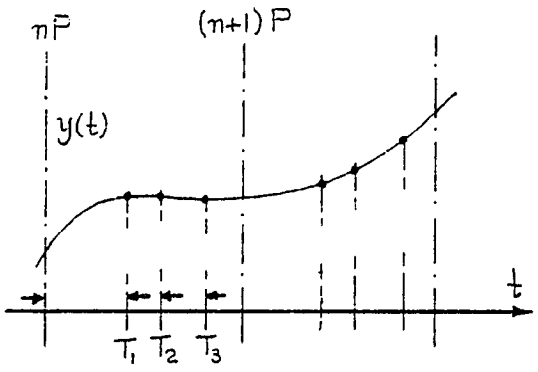


Figure 18: Periodically sampled state model

where c is a row vector, F is a square matrix and b is a column vector. Note that the spectrum of y(t) is given by:

$$Y(w)=c(jwI-F)^{-1}b$$

Definition

For a given pair (n,m) (and a fixed constant M) define the mapping

$$k \longleftrightarrow (n,m)$$

by the equations:

$$k(n,m)=nM+(m-1) \qquad (1)$$

$$n(k)=\text{int}(k/M) \qquad (2)$$

$$m(k)=(k \bmod M)+1 \qquad (3)$$

where int(.) means the largest integer which is smaller than the argument (floor of the argument) and mod is the modulo function (k mod M means the remainder of the division k/M).

After the k=k(n,m) mapping is defined, we can create a time sequence

$$t_k=nP+T_m$$

and a corresponding time series

$$y_k=y(t_k)$$

which consists of the same points as the set (i).

Theorem 6

The time series y_k can be considered as an output of a linear, time varying discrete system of the form:

$$w(k+1)=Sw(k)$$

$$y_k=h(k)w(k) \qquad w(0)=A(h)b$$

i.e. we can use a discrete model where the states are propagated with a

constant matrix S, while the output matrix (row vector h(k)) becomes time varying.

Proof

Let us assume the (n,m) are given, and k=k(n,m) is defined according to (1). Define,

$$h=P/M \tag{4}$$

$$w(k)=w(k(n,m))=x(nP+mh) \tag{5}$$

$$A(r)=\exp(Fr) \tag{6}$$

Then,

$$w(k)=A(mh)x(nP) \tag{7}$$

First we prove that w(k+1) is equal to a constant matrix times w(k). Consider the integer k+1; depending on m,

$$k+1 \text{ corresponds to } \begin{cases} (n+1,0) & \text{if } m=M \\ (n,m+1) & \text{if } m\neq M \end{cases}$$

Therefore, if m=M,

$$\begin{aligned} w(k+1)&=A(h)x((n+1)P) && \text{by applying (7)} \\ &=A(h)A(P)x(nP) && \text{using the state equations and (6)} \\ &=A(h)A(Mh)x(nP) && \text{by using (4)} \\ &=A(h)A(mh)x(nP) && \text{because m=M} \\ &=A(h)w(k) && \text{by applying (7)} \end{aligned}$$

In the case that m≠M,

$$w(k+1)=A((m+1)h)x(nP) \qquad \text{by applying (7)}$$

$=A(h)A(mh)x(nP)$ property of the transition matrix

$=A(h)w(k)$ using (7) again

So, in both cases $w(k+1)=A(h)w(k)$. This proves the first part of the theorem. The second part is proved similarly:

$y(t_k) = cx(nP+T_m)$

$=cA(T_m)x(nP)$ using the state model

$=cA(T_m)A^{-1}(mh)w(k)$

$=cA(T_m-mh)w(k)$

$=h(m)w(k)$ for $h(m)=h(m(k))=cA(T_m-mh)$

(The third equality was derived by using (7) and the fact that A(.) is always nonsingular so,

$$x(nP)=A^{-1}(mh)w(k)$$

which is substituted in place of x(nP).)

This completes the proof of the theorem 1.

Corollary 1

Let $T_k=m(k)h$ (uniform sampling). Then, the model of $y(t_k)$ becomes

$$w(k+1)=A(h)w(k)$$

$$y(t_k)=cw(k)$$

Corollary 2

Let $T_k=qm(k)h$ with $0<q<1$. Then, the model of $y(t_k)$ is

$$w(k+1)=A(h)w(k)$$
$$y(t_k)=cA((q-1)mh)w(k)$$

Theorem 7

Define

$$Y(n)=(y(nP+T_1)\,|\,y(nP+T_2)\,|\,\ldots\,|\,y(nP+T_M))^T$$

Then,

$$x((n+1)P)=A(P)x(nP)$$
$$Y(n)=Hx(nP)$$

where H is an appropriate constant matrix.

Proof

The first equation, $x((n+1)P)=A(P)x(nP)$ is an immediate consequence of the continuous state equation. The second equation comes from the fact

$$\begin{bmatrix} y(nP+T_1) \\ y(nP+T_2) \\ \cdots\cdots \\ y(nP+T_M) \end{bmatrix} = \begin{bmatrix} cA(T_1)x(nP) \\ cA(T_2)x(nP) \\ \cdots\cdots \\ cA(T_M)x(nP) \end{bmatrix} = \begin{bmatrix} cA(T_1) \\ cA(T_2) \\ \cdots \\ cA(T_M) \end{bmatrix} \cdot x(nP)$$

Corollary 1

Define the spectrum of the samples $\{y(t_k)\}$ as the Fourier transform

of the generalized function

$$y'(t)=\sum_{k=o}^{\infty} y(t_k)\delta(t-t_k)$$

$Y'(w)$=Fourier transform of $y'(t)$

Then,

$$Y'(w)=g(w)H(Iz-A)^{-1}b$$

where

$$A=A(P)=\exp(FP)$$

$$g(w)=(\exp(-jwT_1)\ \exp(-jwT_2)\ \ldots\ \exp(-jwT_M))$$

$$z=\exp(jwP)$$

Proof

$$\sum_{n,m}\sum y(nP+T_m)\exp(-jw(nP+T_m)) = \sum_{n=0} g(w)Y(n)\exp(-jwP)$$

But,

$$x((n+1)P)=Ax(nP) \Longrightarrow zX(z)-b=AX(z) \Longrightarrow X(z)=(Iz-A)^{-1}b$$

Then, using the results of theorem 2,

$$Y'(w)=g(w)H(Iz-A)^{-1}b$$

which completes the proof.

Corollary 2

The $Y'(w)$ found above has exactly the same poles as the Fourier transform of the discrete sequence $y(nP)$.

Corollary 3

Assume that $T_i = q_i T_o$ for some integer q_i for i=1,2,...,M and $P = q_o T_o$. Then, Y'(w) is periodic with period T_o.

Proof

Use the expression of g(w) to derive that $g(w) = g(w+2\pi/T_o)$ which means that g(w) is periodic with period $2\pi/T_o$.

Let $a(w) = H(Iz-A)^{-1} \cdot b$; then $a(w) = a(w+2\pi/P)$. Because a(w) is periodic in w with period $2\pi/P$, and P is a multiple of T_o, then a(w) is also periodic in w with period $2\pi/T_o$.

Notes

a) The samples during the intervals [nP, (n+1)P] (for all n) can add zeros through the g(w).

b) $(Iz-A)^{-1}$ has poles associated with the period P i.e. independent of added samples.

c) $(Iz-A)^{-1}$ is periodic in w with period $2\pi/P$; however, g(w) is periodic in w only if the T_i's are all integer multiples of some period T_o. If this happens, g(w) is periodic with period $2\pi/T_o$.

Appendix Ia

The Poisson Sum Formulas

Poisson formulas are of fundamental importance for the sampled systems because they relate Fourier transforms of transient signals (that is, signals with short time records) with their periodic expansions. Poisson's formulas are given by the following assertion:

Assertion

Let f(t) be an L_2 function (of finite energy). We introduce the operator S{.} defined by the summation:

$$S_T f(t) = \sum_{n=-\infty}^{\infty} f(t+nT)$$

where T is an arbitrary real number, called the replication period. The function $S_T f$ is called the periodic expansion of f with period T.

Note that:

$$S_T f(t+aT) = S_T f(t) \qquad \text{for any integer a.}$$

which means that $S_T f$ is a periodic function. The Poisson formulas are:

1. $S_T \delta(t) = 1/T \sum_{n=-\infty}^{\infty} \exp(\, jnWt)$
2. $S_T f(t) = 1/T \sum_{n=-\infty}^{\infty} F(nW) \exp(\, jnWt\,)$
3. $S_W F(w) = T \sum_{n=-\infty}^{\infty} f(nT) \exp(\, -jnTw\,)$

where F(w) is the Fourier transform of f(t) and W is equal to $2\pi/T$.

Proof of 1

Because $S_T \delta$ is a periodic function, we can expand it to a Fourier

series:

$$S_T\delta(t) = \sum_{n=-\infty}^{\infty} a_n \exp(jnWt)$$

where the constants a_n are the Fourier coefficients given by:

$$a_n = 1/T \int_{-T/2}^{+T/2} \delta(t) \exp(-jnwt) dt.$$

But this last integral is equal to 1. Therefore $a_n = 1/T$ and the proof is complete.

Proof of 2

Consider a linear system H(w) with impulse response f(t). Then F(w)=H(w), and the linearity implies:

If the input $\delta(t)$ gives output f(t), then,
the input $S_T\delta(t)$ will give output $S_T f(t)$.

But the output of the system when the input is

$$S_T\delta(t) = 1/T \sum_{n=-\infty}^{\infty} \exp(jnWt)$$

can also be derived as a sum of responses of the exponentials, that is,

$$\text{input} = \exp(jnWt) \implies \text{response} = H(nW)\exp(jnWt)$$

$$\text{input} = 1/T \sum_{n=-\infty}^{\infty} \exp(jnWt) \implies \text{response} = 1/T \sum_{n=-\infty}^{\infty} H(nW)\exp(jnWt)$$

Therefore,

$$S_T f(t) = 1/T \sum_{n=-\infty}^{\infty} H(nW) \exp(jnWt)$$

which is the desired result (considering that F(w)=H(w)).

Proof of 3

The symmetry property:

if $f(t) \longleftrightarrow F(w)$ are Fourier pairs,

then $F(t) \longleftrightarrow 2\pi f(-w)$ are also Fourier pairs,

can be applied to the previous result to find

$$S_T F(t) = 2\pi/T \sum_{n=-\infty}^{\infty} f(-nW) \exp(jnWt)$$

In this equation t is the independent variable and T has to satisfy the equation $T=2\pi/W$. Therefore,

t can be replaced by w, and,

T can be replaced by W.

This gives:

$$S_W F(w) = 2\pi/W \sum_{n=-\infty}^{\infty} f(-nT) \exp(jnTw)$$

Finally, changing the summation index from n to -n and interchanging the summation limits, the desired result (3) is derived.

Appendix Ib

On Tchebyscheff Polynomials

The Tchebyscheff polynomial of order N is defined by

$$T_N(x) = \cos(\, N\arccos(x)\,) \qquad \text{for } -1 \le x \le 1 \qquad (1)$$

and

$$T_N(x) = \cosh(\, N\operatorname{arccosh}(x)\,) \qquad \text{for } |x| > 1 \qquad (2)$$

Expanding the cos() and cosh() functions, the following recursive functional equation is satisfied:

$$T_{N+1}(x) = 2xT_N(x) - T_{N-1}(x) \qquad (3)$$

$$T_0(x) = 1, \quad T_1(x) = x$$

Examples:

$T_0(x)=1$	$1 = T_0(x)$
$T_1(x)=x$	$x = T_1(x)$
$T_2(x)=2x^2-1$	$x^2 = (T_0+T_2)/2$
$T_3(x)=4x^3-3x$	$x^3 = (3T_1+T_3)/4$
$T_4(x)=8x^4-8x^2+1$	$x^4 = (3T_0+4T_2+T_4)/8$
$T_5(x)=16x^5-20x^3+5x^1$	$x^5 = (10T_1+5T_3+T_5)/16$
.....	.....

Tchebyscheff polynomials satisfy the following differential equation

$$(1-x^2)T_N''-xT_N'+N^2T_N=0$$

(where the prime indicates differentiation with respect to x).

The roots of the Tchebyscheff polynomials are

$$\text{(k-th root)} \quad x_k=\cos\frac{(2k-1)\pi}{2N} \qquad k=1,2,\ldots,N \tag{5}$$

and the k-th root is the same as the N-K+1 root.

The following equalities and inequalities are also true:

$$T_N(1) = 1$$

$$T_N(-1)=(-1)^N$$

$$T_{2k}(0)=1$$

$$T_{2k+1}(0)=0$$

$$|T_N(x)|<1 \qquad \text{for } -1\underline{<}x\underline{<}1$$

$$|T_N(x)|\underline{<}2^{N-1}x^N \qquad \text{for } x<-1 \text{ and } x>1$$

$$|T_N(x)|\underline{>}2^N \qquad \text{for } x<-1 \text{ and } x>1$$

$T_{2k}(x)$ is an even function of x

$T_{2k+1}(x)$ is an odd function of x

In addition, the following substitutions are very common:

for $|x|\underline{<}1$ $\quad T_N(x) = \cos(Nw)$ $\quad$ with $x=\cos w$ (6)

$$T_N(x) = (z^N+z^{-N})/2 \qquad \text{with } x=(jz+1/jz)/2 \tag{7}$$

for $|x|\underline{>}1$ $\quad T_N(x) = \cosh(Nw)$ $\quad$ with $x=\cosh w$ (8)

$$T_N(x) = (z^N+z^{-N})/2 \qquad \text{with } x=(z+1/z)/2 \tag{9}$$

Notes and References

[1] Jerri A. J., 'The Shannon sampling theorem - Its various extensions and applications: A Tutorial review,' Proceedings of the IEEE, vol. 65, pp. 1565-1595, Nov. 1977.

[2] Jury E. I., 'Sampling schemes in sampled data control systems,' IRE Trans. on Automatic Control, vol. AC-6, pp. 86-88, Feb. 1961.

[3] Jury E. I. and F. J. Mullin, 'The analysis of sampled-data control systems with a periodically time-varying Sampling rate,' IRE Trans. on Automatic Control, pp. 15-25, May 1959.

[3] Kalman R. E. and J. E. Bertram, 'A unified approach to the theory of sampling systems,' J. Franklin Inst., vol. 267, pp. 405-436, May 1959.

[4] Will P. M., 'Variable Frequency Sampling,' IRE Trans. on Automatic Control, vol. AC-7, p. 126, Oct. 1962.

[5] Yen J. L., 'On nonuniform sampling of bandwidth limited signals,' IRE Trans. Circuit Theory, vol. CT-3, pp. 251-257, 1956.

[6] D' Angelo Henry, Linear Time Varying Systems: Analysis and Synthesis. Boston: Allyn and Bacon Inc., 1970.

[7] King R. W., The Theory of Linear Antennas. Harvard University Press, Cambridge, Mass. 1956.

CHAPTER II

PARTIALLY SAMPLED ARMA MODELS

Outline of Chapter II

1. Notation and Conventions
2. Least Squares Identification
3. Sequential Identification
4. ARMA Model Identification from Partial Observations
5. Filling Missing Data Using ARMA Coefficients
6. Validation of Theorem 1
7. Applying the Results of Theorem 2

Appendix II

Notes and References II

Outline Chapter II

Autoregressive-moving average (ARMA) filters have been widely used to model deterministic events and random processes. The most important property of the ARMA filters is that they can adequately describe the input-output relationship of a uniformly sampled, linear, time invariant dynamic system. In the case of nonuniform sampling, a time invariant system results in a time varying discrete system which can not be analytically studied using ARMA model techniques.

In this chapter we present the results of a study that can be used to model partially sampled dynamic systems. The term partially sampled system is used with the following meaning: sampling of output signals occurs periodically and uniformly but for certain time intervals no samples are taken. We also use the term uniform sampling with missing data points interchangeably. More specifically, the following two problems are addressed:

(1) Given a table of input-output data of a partially sampled system, find the ARMA coefficients corresponding to a uniformly sampled system.
(2) Given the ARMA coefficients of a partially sampled system, find the missing output data points by interpolation.

The chapter is organized into 7 sections and an appendix. Sections 1, 2 and 3 provide the introductory background we use for the next three sections. A list of the vector-matrix notational conventions and the definition of the ARMA(N,M) model is given in section 1. Section 2 outlines the least squares ARMA model identification procedure and section 3 summarizes the sequential identification algorithm based on Kalman filter theory. In section 4 we present the new results on ARMA

model identification based on partial observations and the next section (5) deals with the interpolation of the missing data points. Sections 6 and 7 are devoted to the justification of the previous theory. Simulation programs and results are given to validate the assertions. Finally, in appendix II we include the programs used for the simulations and a list of references.

1. Notation and Conventions

$x=(x_1\ x_2\ \ldots\ x_N)$ means that x is a row vector of N elements

$x=(x_1,x_2,\ldots,x_N)$ means that x is a column vector of N elements

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \cdot\cdot \\ x_N \end{pmatrix}$$ means that x is a column vector of N elements

$A=(a_{ij})$ matrix A consisting of elements a_{ij}.

$(a_{ij}: i=1,N;\ j=1,M)$ matrix A with dimension N by M

$$\begin{pmatrix} 11 & 12 & \ldots & 1M \\ 21 & 22 & \ldots & 2M \\ \cdots & \cdots & \cdots & \cdots \\ N1 & N1 & \ldots & NM \end{pmatrix}_a$$ has identical meaning with $(a_{ij}: i=1,N;\ j=1,M)$

$$\begin{bmatrix} 11' & x & \dots & 1M' \\ y & 22' & \dots & 2M' \\ & & \dots & \\ N1' & w & \dots & NM' \end{bmatrix}_a$$

means that the elements a_{12}, a_{21}, and a_{N2} have been replaced by the scalars x, y, w

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}_y = \begin{bmatrix} y_1 & y_2 \\ y_3 & y_4 \end{bmatrix}$$

In general, we write the index of the variable instead of the indexed variable

Definition of an ARMA(N,M) model

Let {u(k)} be a deterministic sequence of real numbers and {e(k)} be a white random process of zero mean and variance R. An ARMA model of degrees N and M is defined by the difference equation

$$y(k)=a_1y(k-1)+a_2y(k-2)+\dots+a_Ny(k-N)+b_1u(k-1)+\dots+b_Mu(k-M)+e(k)$$

where $M \leq N$ and a_N, b_M are not equal to 0

Taking the Z transform of the above equation and rearranging terms, we find

$$Y(z) = \frac{b_1z^{-1}+\dots+b_Mz^{-M}}{1-a_1z^{-1}-\dots-a_Nz^{-N}}U(z) + \frac{1}{1-a_1z^{-1}-\dots-a_Nz^{-N}}E(z)$$

The first ratio is the transfer function from U(z) to Y(z) and the second ratio is the transfer function from E(z) to Y(z). Now, we introduce the notation

$$f=(a_1,a_2,\ldots,a_N,b_1,\ldots,b_M)$$

$$C(k)=(\ y(k-1)\ \ldots\ y(k-N)\ u(k-1)\ \ldots\ u(k-M)\ )$$

Then, the ARMA(N,M) equation can by written in vector form

$$y(k)=C(k)f+e(k) \qquad \text{(i)}$$

2. Least Squares Identification

The least squares identification problem is formulated in the following manner: Estimate the vector f using the observations ($y_1,y_2 \ldots y_K$) and minimizing the total equation error

$$J(f) = \sum_{k=1}^{K} e^2(k)$$

where,

$$e(k)=y(k)-C(k)f \quad \text{for } k=1,2,\ldots,K$$

Solution

Rewriting equation (i) for k=N,N+1,...,K we form the system of linear equations

$$\begin{bmatrix} y(N+1) \\ y(N+2) \\ \cdots\cdots \\ y(K) \end{bmatrix} = \begin{bmatrix} C(N+1) \\ C(N+2) \\ \cdots\cdots \\ C(K) \end{bmatrix} f + \begin{bmatrix} e(N+1) \\ e(N+2) \\ \cdots\cdots \\ e(K) \end{bmatrix} \qquad \text{(ii)}$$

The coefficient matrix of f (containing the rows C(N+1)...C(K)) is denoted by W and it is partitioned in two blocks A and B as follows

$$W=(A|B)$$

$$A = \begin{pmatrix} N & N-1 & \dots & 1 \\ N+1 & N & \dots & 2 \\ & & & \\ \dots & \dots & \dots & \dots \\ K-1 & K-2 & \dots & K-N \end{pmatrix}_{y} \qquad B = \begin{pmatrix} N & N-1 & \dots & N-M+1 \\ N+1 & N & \dots & N-M+2 \\ & & & \\ \dots & \dots & \dots & \dots \\ K-1 & K-2 & & K-M \end{pmatrix}_{u} \qquad \text{(iii)}$$

Matrix A has dimensions K-N by N and depends only on y(1),...,y(K-1). Matrix B has dimensions K-N by M and depends only on u(N-M+1),...,u(K-1). By defining Y to be the column vector of y's in (ii), and E to be the column vector of e's, we can rewrite (ii) in compact form

$$Y=Wf+E$$

In this equation Y and W are known and f, E are unknowns. Also note that minimization of the sum of squares of the errors e(k) is equivalent to minimizing

$$J(f)=E^{T}E \quad \text{with respect to f.}$$

The solution of the above problem [1] is unique in the case

$$\text{rank}(W|Y) = N+M$$

and it can be found by the formulas

$$f'=W^{+}Y$$

where,

$$W^{+}=(W^{T}W)^{-1}W^{T}$$

resulting in a minimum total error

$$J(f')=Y^{T}(I-WW^{+})Y.$$

Note that if W is a square matrix of full rank (N+M), $WW^{+}=I$ and the total error J(f') is 0. Also, by replacing W^{+} by the generalized Penroze inverse [2], a less restrictive solution f' can be found relaxing from the rank(W|Y)=N+M condition.

3. Sequential Identification

Let (y(1),y(2),...,y(K)) be an observation set of K measurements and f(K) be an estimate of f (least squares estimate or any other estimate) based on the above set. The ARMA model equation (i) can be written in the following state form

$$f(k+1)=If(k) \qquad k=K,K+1,...$$
$$y(k)=C(k)f(k)+e(k)$$

The unity matrix that appears in the first equation has dimensions N+M by N+M. This means that the above system has N+M poles on the unit circle, and because C is a function of the time index k, the system is time varying. The problem is to find least squares estimates f(n) of f based on the next observations K,K+1,...,K+n,...

Solution

The following algorithm is the result of a time varying Kalman filter [3] for the above system:

(1) Get y(n+1) and u(n+1). Formulate

$$C(n+1)=(y(n)\ y(n-1)\ \ldots\ y(n-N+1)\ u(n)\ \ldots\ u(n-M+1))$$

(2) Compute the Kalman gain (vector)

$$L(n+1)= \frac{P(n)}{p}C^{T}(n+1)\left[\frac{1}{q}+C(n+1)\frac{P(n)}{p}C^{T}(n+1)\right]^{-1}$$

(3) Estimate f recursively by

$$f(n+1)=f(n)+L(n+1)\left[y(n+1)-C(n+1)f(n)\right]$$

(4) Update the error variance using

$$P(n+1)=\frac{1}{p}\left[P(n)-L(n+1)C(n+1)P(n)\right]$$

The following notes explain the meaning and the significance of the terms that appear in the above algorithm:

(a) The inverse in step (2) is a scalar quantity. No matrix inversion is required.

(b) Matrix P has dimensions N+M by N+M and corresponds to the variance of the estimation error f(k)-f. Initially, for n=K, P(n) in step (2) can be set equal to eI where e is a large positive number.

(c) The scalars p and q can take the values

p=1 q=1/R for ordinary Kalman filtering (R=var{e})

0<p<1 q=1-p for exponential smoothing Kalman filtering [4]

p=1 q=small positive number for Kalman filtering with unknown measurement noise.

(d) In order for the filter to converge, uniform observability [5] is required. This implies that the Fisher information matrix [6]

$$\sum_{j=K}^{F} C^T(j+1)C(j+1)$$

is positive definite for all the final values F>K.

Using the expression for C(j+1), we find

$$\begin{pmatrix} y(j)y(j) & y(j)y(j-1) & \dots & y(j)u(j-M) \\ y(j-1)y(j) & y(j-1)y(j-1) & \dots & y(j-1)u(j-M) \\ \dots\dots & \dots\dots & & \dots\dots \\ u(j-M)y(j) & u(j-M)y(j-1) & \dots & u(j-M)u(j-M) \end{pmatrix}$$

where the summation limits extend from j=K to F. Observe that when F—>∞ (infinite number of observations) the above summations correspond to the statistical autocorrelations and crosscorrelations of y and u [6]. More specifically, if the system is unforced u(j)=0 and the Fisher matrix is the same as the partial correlations Toepliz matrix [7].

4. ARMA Model Identification from Partial Observations

Problem Formulation

Let $(y_1, y_2, \dots, y_K)$ be a set of uniformly collected observations

from which $(y_i, y_j,\ldots y_k)$ are missing (bad data). Assume also that the driving input sequence $(u_1, u_2,\ldots,u_{K-1})$ is partially known and $(u_i,u_j,\ldots,u_k)$ are missing. The problem is to find the ARMA coefficient vector f in a manner that the total equation error

$$e^2(1)+\ldots+e^2(i-1)+e^2(i+1)+\ldots+e^2(K)$$

(terms i,j,...k are not included)

is minimum.

Theorem 1

The following algorithm finds the optimum vector f (in the least squares sense) for the missing data problem:

(1) Formulate the matrix W=(A|B) and the Y vector according to equations (ii) and (iii). Replace the unknown elements with 0.

(2) Form the sets D(n)={n-N,n-N+1,...,n} for n=i,j,...,k

(3) Form the union U(i,j,...,k)=D(i)uD(j)u...uD(k).

(4) Define a matrix Q with elements e_{nm} where

$e_{nm} \neq 1$ for n=m

=1 if n=m and n does not belong to U(i,j,...,k)

=0 if n=m and n belongs to U(i,j,...,k)

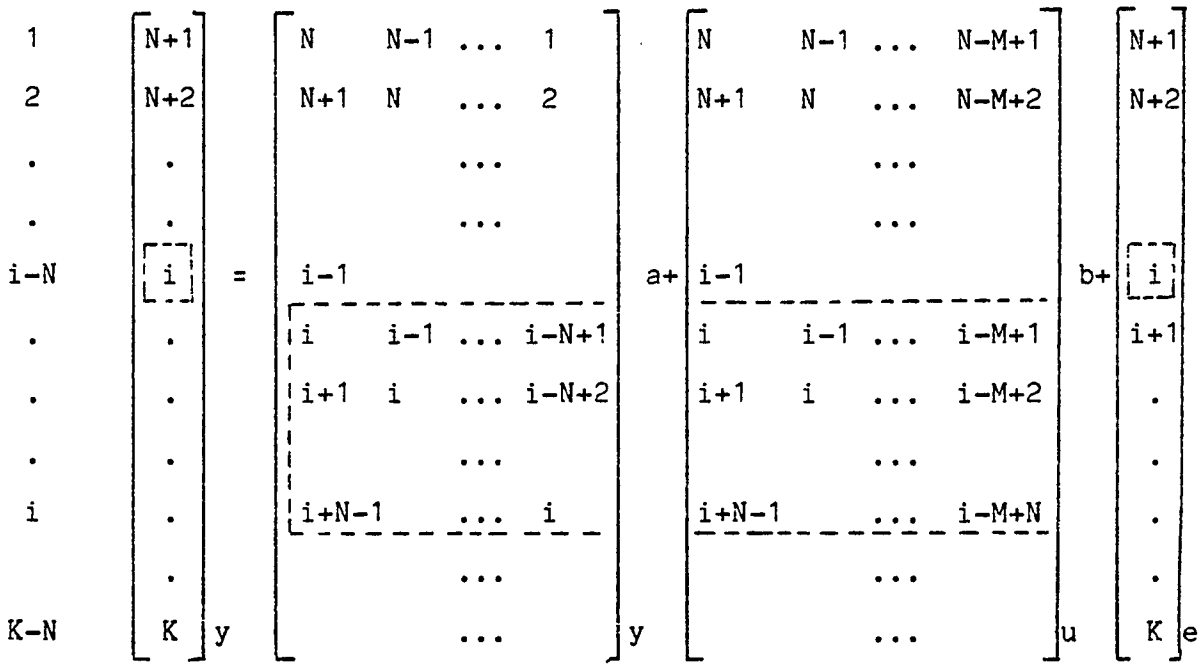
(5) Find the Q-weighted least squares solution of the equation Y=Wf+E, that is

$$f=(W^TQW)^{-1}W^TQY$$

Then, f minimizes the total equation error.

Proof

The proof of theorem 1 is based on the special form of the equation Y=Wf+E. For simplicity we assume that only y(i) and u(i) are missing.



We observe that the missing datum y(i) appears in the (i-N)-th row of the left side and in rows i-N+1,...,i of the right side of matrix A. Also, because N>M, the missing input u(i) appears at most at the (i-N+1,...,i)-th rows of the right side of matrix B. Therefore, by deleting the rows

$$\{i-N, i-N+1, \ldots, i\}$$

the resulting equations do not contain the unknowns y(i) and u(i). Calling Y' and W'=(A'|B') the vector Y and the matrix W=(A|B) after

deleting the above rows, the least squares solution of the equation

$$Y'=W'f+E'$$

with respect to f will gives the ARMA coefficients based on the observations y(1),...,y(i-1),y(i+1),...,y(K). This solution is:

$$f'=(W'^{T}W')^{-1}W'^{T}Y'$$

But this is equivalent to

$$f'=(W^{T}QW)^{-1}W^{T}QY$$

where Q is a diagonal matrix which has elements equal to 1 except at the diagonal positions {i-N,i-N+1,...,i} which are equal to 0. When the observations y(i),y(j),...,y(k) are missing, the above matrix Q can be easily verified to be given by the algorithmic steps (2), (3) and (4). This completes the proof of theorem 1.

Note

If the number of observations are K, and there are no missing observations, the number of equations is K-N. If S sequential observations are missing (say i,i+1,...,i+S-1) then the number of equations reduces to (K-N)-(S+N)=K-S-2N. If S observations which are N indices apart (say i,i+N,i+2N,...,i+(S-1)N) are missing, then the number of equations reduces to (K-N)-S(N+1)=K-S-(S+1)N. In all the above cases K should be sufficient large such that the resulting equations are more than N+M. This does not guarantee the existence of a unique solution but it is a necessary condition for it.

5. Filling Missing Data Using ARMA Coefficients

Theorem 2

The following theorem can be used to interpolate missing data when the ARMA coefficients are known.

Let $(y_1, y_2 \ldots, y_K)$ be a set of observations and $(u_1, u_2 \ldots, u_K)$ be a set of driving inputs to an ARMA(N,M) model. Assume that:

(1) The observations with indices i,j,...,k are missing (bad data).

(2) The ARMA coefficients have been estimated. (Theorem 1 can be used for that purpose)

(3) The equation errors e(k) have been estimated at the known data points.

Then, the vector $Y^{ij.k}$ of the missing data and the error vector $E^{ij.k}$ can be found by solving a least squares problem of the form

$$(\text{known vector}) = (\text{known matrix}) Y^{ij.k} + E^{ij.k}$$

Note that a least squares solution implies that the total error

$$(E^{ij.k})^T (E^{ij.k})$$

is minimized.

The proof of the theorem requires several steps and some more notational conventions. To make the proof easier to follow we give a series of 4 lemmas and then we prove the theorem. The importance of this result is that the missing data can be expressed linearly with respect to known quantities in an algorithmic manner.

Lemma 1

The following equality is true for any scalar s and any N dimensional vector a.

$$\begin{pmatrix} s & 12' & 13' & \dots & 1N' \\ 21' & s & 23' & \dots & 2N' \\ \dots & \dots & \dots & \dots & \dots \\ M1' & M2' & M3' & \dots & s \end{pmatrix}_y \cdot a = \begin{pmatrix} 0 & 12' & 13' & \dots & 1N' \\ 21' & 0 & 23' & \dots & 2N' \\ \dots & \dots & \dots & \dots & \dots \\ M1' & M2' & M3' & \dots & 0 \end{pmatrix}_y \cdot a + sa$$

Lemma 2

The following equality is true for any N dimensional vector a

$$\begin{pmatrix} N & N-1 & \dots & 1 \\ N+1 & N & & 2 \\ \dots & \dots & \dots & \dots \\ i & i-1 & \dots & i-N+1 \\ i+1 & i & & \dots \\ \dots & \dots & \dots & \dots \\ i+N-1 & i+N-2 & \dots & i \\ i+N & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ K-1 & K-2 & & K-N \end{pmatrix}_y \cdot a = \begin{pmatrix} & & & \\ & \text{replace the i-th} & & \\ & \text{elements with a 0's} & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}_y a + \begin{pmatrix} 0 \\ 0 \\ \dots \\ 1 \\ 2 \\ \dots \\ N \\ 0 \\ \dots \\ 0 \end{pmatrix}_a \cdot y_i$$

We now introduce the notation:

A^i = Matrix A after replacing the elements y_i by 0

m_a^i = Vector containing the a-vector at the corresponding rows of A that contain a_i The remaining elements of m_a^i are equal to 0.

Then, the above equation can be written in a more compact form:

$$Aa = A^i a + m_a^i y_i$$

Lemma 3

Let i,j,...,k be indices of elements of matrix A (A has a similar form as in lemma 2). Assume also that

$$N+1 \leq i,j,\ldots,k \leq K-N+1$$

Then, application of the result of lemma 2 gives

$$Aa = A^{ij.k}a + [m_a^i \; m_a^j \ldots m_a^k] \begin{pmatrix} y_i \\ y_j \\ \ldots \\ y_k \end{pmatrix}$$

where $A^{ij.k}$ denotes the matrix A after replacing the elements $y_i, y_j, \ldots, y_k$ by 0.

Now, use the notation

$$M_a^{ij.k} = [m_a^i \; m_a^j \ldots m_a^k]$$

$$Y_{ij.k} = \begin{pmatrix} y_i \\ y_j \\ \ldots \\ y_k \end{pmatrix}$$

to rewrite the above equation. The result is

$$Aa = A^{ij.k}a + M_a^{ij.k} Y_{ij.k}$$

Lemma 4

Let E be a vector with elements $(e_i: i=1,\ldots,N)$. Then,

$$E=E^{i}+n^{i}e_{i}$$

where,

E^{i} = Vector E after replacing the i-th element with a 0

n^{i} = Column of 0's except with a 1 at the i-th position

Extending the above statement, we can derive

$$E=E^{ij.k}+[n^{i}\ n^{j}\ldots n^{k}]\begin{bmatrix} e_{i} \\ e_{j} \\ \ldots \\ e_{k} \end{bmatrix}$$

and by using the notation

$$N^{ij.k}=[n^{i}\ n^{j}\ldots n^{k}]$$

$$E_{ij.k}=\begin{bmatrix} e_{i} \\ e_{j} \\ \ldots \\ e_{k} \end{bmatrix}$$

we have

$$E=E^{ij.k}+N^{ij.k}E_{ij.k}$$

Proof of Theorem 2

The following equation relates the observations with the parameters f

$$Y=Wf+E \quad \text{or} \quad Y=Aa+Bb+E$$

where W, A and B have been defined in section 2. Using lemma 4 we can separate unknown terms from knowns for the Y and E vectors

$$Y = Y^{ij.k} + N^{ij.k} Y_{ij.k}$$

$$E = E^{ij.k} + N^{ij.k} E_{ij.k}$$

Then, using lemma 3, we can separate the unknown y's from A

$$Aa = A^{ij.k} a + M_a^{ij.k} Y_{ij.k}$$

Substituting in Y=Aa+Bb+E we find

$$Y^{ij.k} + N^{ij.k} Y_{ij.k} = A^{ij.k} a + M_a^{ij.k} Y_{ij.k} + Bb + E^{ij.k} + N^{ij.k} E_{ij.k}$$

and equivalently,

$$Y^{ij.k} - A^{ij.k} a - Bb - E^{ij.k} = (M_a^{ij.k} - N^{ij.k}) Y_{ij.k} + N^{ij.k} E_{ij.k} \qquad (v)$$

This equation is of the desired form

(known vector)=(known matrix) (unknown parameters)+(error vector)

To complete the proof of the theorem 2, we need to show that

$$(N^{ij.k} E_{ij.k})^T (N^{ij.k} E_{ij.k}) = e^2(i) + e^2(j) + \ldots + e^2(k)$$

But this is readily obtained from the fact that the square matrix

$$(N^{ij.k})^T (N^{ij.k})$$

contains zero elements except at the i-th, j-th,...,k-th diagonal positions. Therefore, minimization of the error term in (v) is equivalent in minimizing the sum of the squares of the equation error at the missing points i,j,...,k.

6. Validation of Theorem 1

Extensive simulations were used to validate the results of theorem 1. In this section we describe the software written for this purpose and we summarize the experiments performed.

The software consists of two main programs and several subroutines. The first (called ARMASIM), can be used to simulate an ARMA model; the second (called ARMAIDE) can be used to identify an ARMA model from a table of complete or incomplete input-output data. Several disc files are used to hold the intermediate results, so the user can plot input-output relations and trace back the flow of the procedures. The subroutines called by the above programs are

ARMINI	(Initiates the ARMA model)
ARMA	(Finds the output of the model for a given input)
RANDUL	(Random number generator)
GAUSS	(Gaussian random number generator)
LSQ	(High accuracy least squares equation solver)
SQRDC	(Q-R decomposition routine used by LSQ)
SQRST	(Used by LSQ)
VNORM2	(Finds the Eucledian norm of a vector)

The main programs and the above subroutines are given in appendix II. In addition, many routines from EASYPACK are called to perform matrix operations and miscellaneous general purpose functions. (EASYPACK and the interactive program EASY are discussed in more detail in the appendix on software support.)

All the experiments were performed using an input sequence u(i)

i=0...63 which is given in figure 1. As a first step, an ARMA(4,2) model with coefficients

a(1)=-0.11, a(2)=-0.22, a(3)=-0.23, a(4)=-0.10
b(1)= 0.90, b(2)= 0.85

was used to find the output function y(i) (given in figure 2). At this phase, no noise was added to the output y(i). As a second step, the ARMAIDE program was used to identify the coefficients a(.), and b(.) from the u-y table generated before. Then, we assumed that y(25)...y(37) were missing and ARMAIDE was used again to find the coefficients. A part of the printout is given in sample 1 and 2. Figure 2 and samples 3, 4 and 5 show the effect of adding noise to the measurements y. Notice also that in sample 3 and 5 the order of the autoregressive part of the model (N) was assumed to be 5 instead of 4. The results were

a(5)=-0.127E-02 and a(5)=-0.647E-02

respectively which are one order of magnitude less than the other coefficients. In general, the identification was succesful for low noise level measurements; the estimation error for the parameters a and b was almost independent of the amount of missing data.

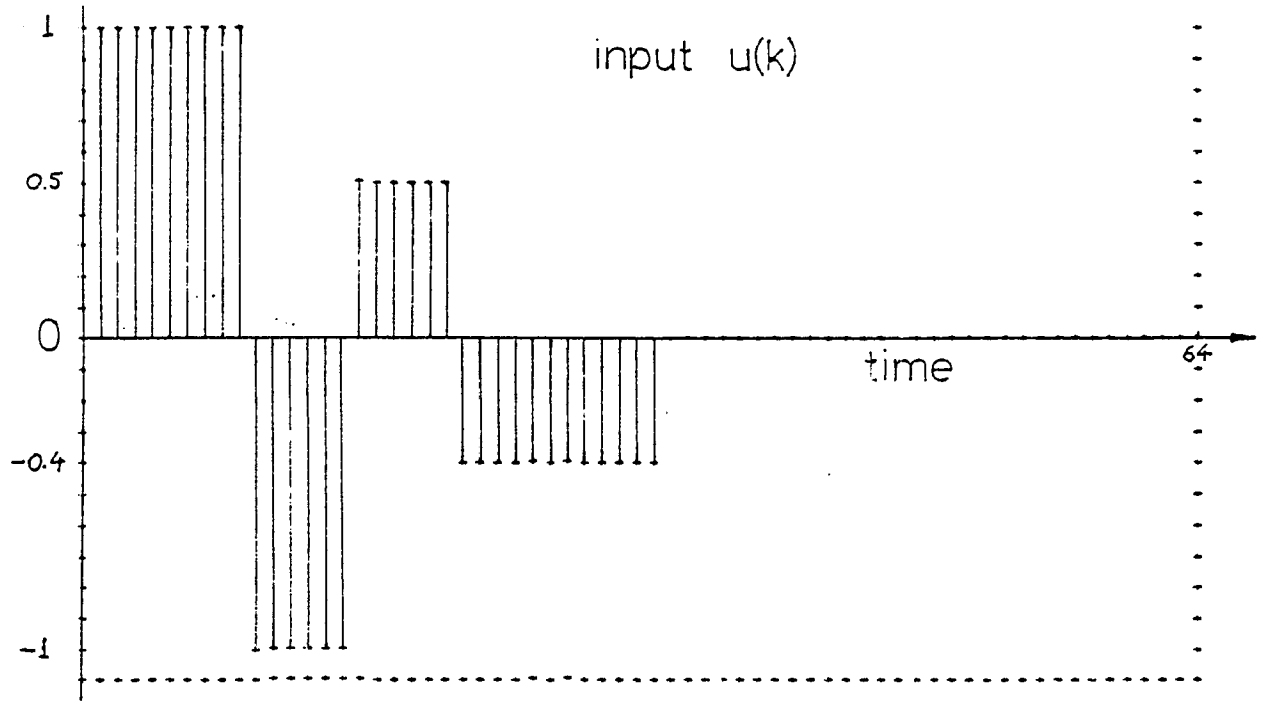


Figure 1: Input sequence

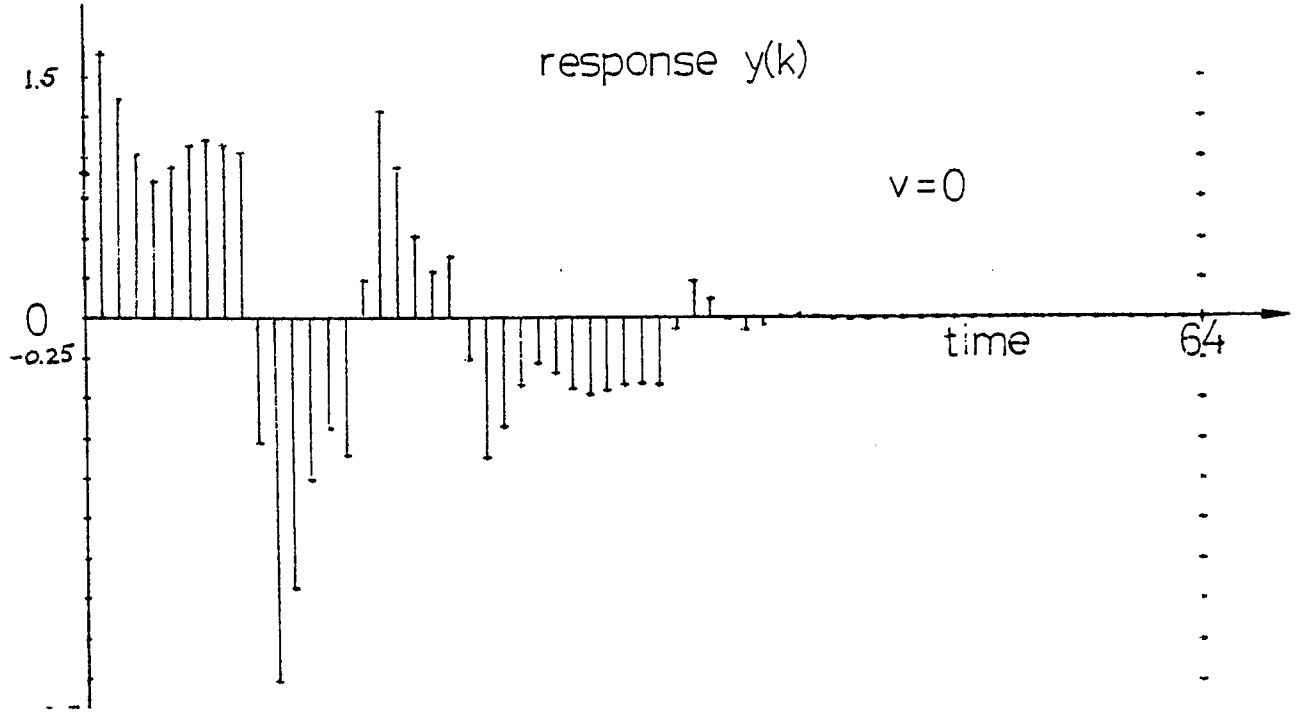


Figure 2: Output sequence (no noise)

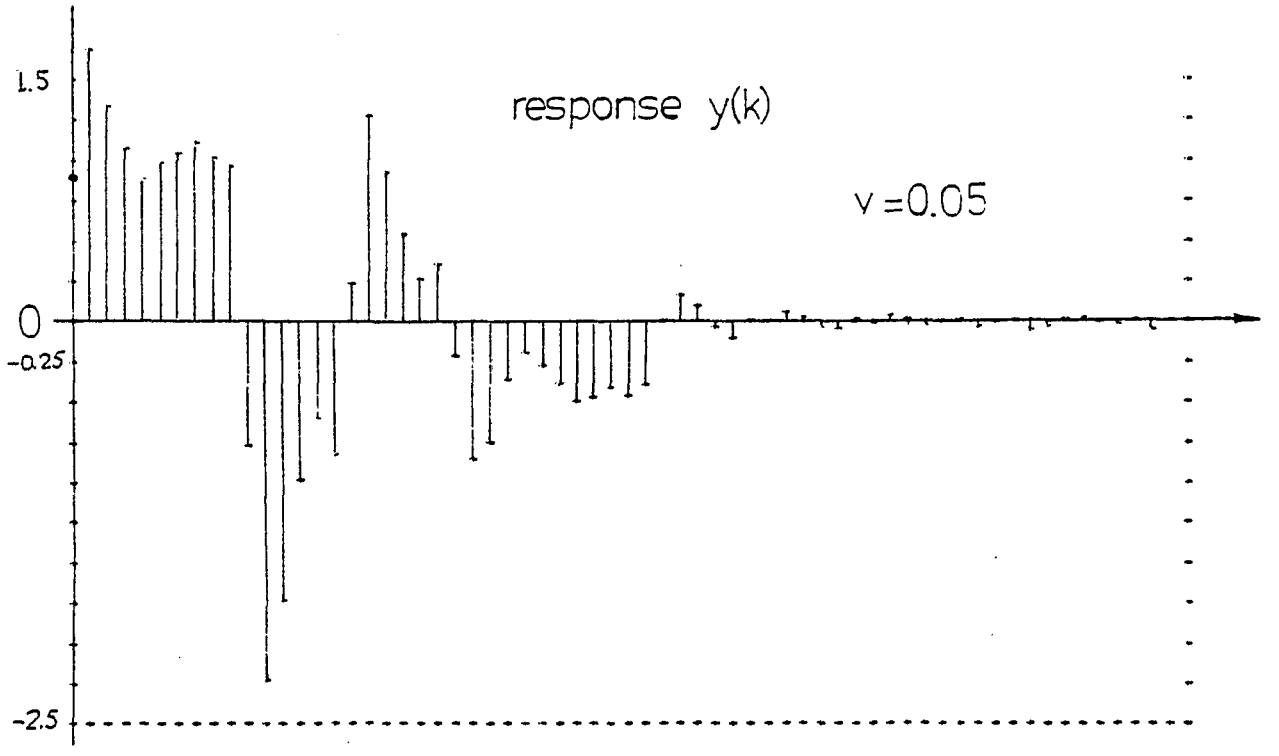


Figure 3: Output sequence (noise added)

```
B>ARMAIDE                          (command to execute the ARMAIDE program)

Enter estimates of the order of the model N,M: 5,2
Enter index limits of missing points 11, 12: 25, 37

Solving a system of 59 equations with   7 unknowns
Please wait...

Results: N, M, estim-a-pars  estim-b-pars
   1:   4.000
   2:   2.000
   3:  -0.110
   4:  -0.220                    Sample 1: Using the ARMAIDE program
   5:  -0.230
   6:  -0.100
   7:   0.900
   8:   0.800

             4=given estimate of N
             2=given estimate of M
The calculated parameters (a and b)
Dimensions:     6 by 1
   1:  -.110
   2:  -.220
   3:  -.230                     Sample 2: No noise, no missing points
   4:  -.100E+00
   5:   .900
   6:   .850

   0.101E-05 = norm2 of the estimation error
```

```
              5=given estimate of N
              2=given estimate of M
The calculated parameters (a and b)
Dimensions:     7 by 1
   1:  -.120
   2:  -.208
   3:  -.247                  Sample 3: Variance of the noise =0.01
   4:  -.943E-01                        Missing points 25...37
   5:  -.127E-02
   6:   .899
   7:   .808

   0.361E-01 = norm2 of the estimation error

              4=given estimate of N
              2=given estimate of M
The calculated parameters (a and b)
Dimensions:     6 by 1
   1:  -.148E-01
   2:  -.260
   3:  -.222                  Sample 4: Variance of the noise = 0.05
   4:  -.866E-01                        No missing points
   5:   .899
   6:   .751

   0.278     = norm2 of the estimation error

              5=given estimate of N
              2=given estimate of M
The calculated parameters (a and b)
```

Dimensions: 7 by 1

1: -.130

2: -.181

3: -.297

4: -.742E-01

5: -.647E-02

6: .900

7: .850

Sample 5: Variance of the noise = 0.05

Missing points 23...34

0.181 = norm2 of the estimation error

7. Applying the Results of Theorem 2

With the following example we illustrate the details of computing the values of missing points using the ARMA coefficients.

Assume (for the moment) that the discrete system

$$y(k) = -0.2y(k-1)-0.1y(k)+5.0u(k)+e(k)$$

is known and it is driven by the input sequence u(i) for i=0 to 15. The following table shows an experimental input sequence u(i) and the associated response y(i). (The table is found using the ARMASIM program.)

u(0)=+1	y(1)= 5.00
u(1)=+1	y(2)= 4.00
u(2)=+1	y(3)= 3.70
u(3)=+1	y(4)= 3.86

u(4)=-1	y(5)=-6.14	
u(5)=-1	y(6)=-4.16	- missing m(1)
u(6)=-1	y(7)=-3.55	- missing m(2)
u(7)=-1	y(8)=-3.87	- missing m(3)
u(8)= 0	y(9)= 1.13	- missing m(4)
u(9)= 0	y(10)= .161	- missing m(5)
u(10)= 0	y(11)=-.145	- missing m(6)
u(11)= 0	y(12)= .0129	
u(12)= 0	y(13)= .0119	
u(13)= 0	y(14)=-.004	
u(14)= 0	y(15)=-.0005	
u(15)= 0	y(16)= .0005	

Now, inverting the assertion, assume that the above table is known but the points y(6) to y(11) are 'missing' (or the collected data are 'bad'). Our objective is to find the missing points y(6)...y(11). As a first step we find the ARMA filter coefficients by using the program ARMAIDE and declaring that data 6 to 11 are missing; the result is,

$$f=\begin{pmatrix}-0.20\\-0.10\\5.00\end{pmatrix}$$

with very small error (the norm-2 of the error is of the order E-6). Next, we apply the steps of theorem 2.
In the case that E is approximately 0, the following equations are true:

(1) Y = Wf+E

(2) Wf=W'f+M'm (Application of lemma 3)

(3) Y =Y'+N'm (Application of lemma 4)

(4) Y'-W'f = (M'-N')m (Equation to be solved from theorem 2)

(The notation has been simplified. Instead of using the superscripts 6,7,...,11 over Y, W, M and N, we simply use a prime.)
For this example, W, W', and M' are

	W(14 by 3)			W'(14 by 3)			(-1)*M'(14 by 6)					
1:	4.00	5.00	1.0	4.00	5.00	1.0	0	0	0	0	0	0
2:	3.70	4.00	1.0	3.70	4.00	1.0	0	0	0	0	0	0
3:	3.86	3.70	-1.0	3.86	3.70	-1.0	0	0	0	0	0	0
4:	-6.14	3.86	-1.0	-6.14	3.86	-1.0	0	0	0	0	0	0
5:	-4.16	-6.14	-1.0	0.00	-6.14	-1.0	.2	0	0	0	0	0
6:	-3.55	-4.16	-1.0	0.00	0.00	-1.0	.1	.2	0	0	0	0
7:	-3.87	-3.55	0.0	0.00	0.00	0.0	0	.1	.2	0	0	0
8:	1.13	-3.87	0.0	0.00	0.00	0.0	0	0	.1	.2	0	0
9:	.161	1.13	0.0	0.00	0.00	0.0	0	0	0	.1	.2	0
10:	-.145	.161	0.0	0.00	0.00	0.0	0	0	0	0	.1	.2
11:	.0129	-.145	0.0	.0129	0.00	0.0	0	0	0	0	0	.1
12:	.0119	.0129	0.0	.0119	.0129	0.0	0	0	0	0	0	0
13:	-.004	.0119	0.0	-.004	.0119	0.0	0	0	0	0	0	0
14:	-.0005	-.004	0.0	-.0005	-.004	0.0	0	0	0	0	0	0

Of course W can not be obtained from the collected data, because y(6)...y(11) are missing; but W' and M' can be found because they do not contain y(6)...y(11). In addition, the vectors Y, Y' and the matrix N' for this example are:

	Y(14)	Y'(14)	N'(14 by 6)					
1:	3.70	3.70	0	0	0	0	0	0
2:	3.86	3.86	0	0	0	0	0	0
3:	-6.14	-6.14	0	0	0	0	0	0
4:	-4.16	0.00	1	0	0	0	0	0
5:	-3.55	0.00	0	1	0	0	0	0
6:	-3.87	0.00	0	0	1	0	0	0
7:	1.13	0.00	0	0	0	1	0	0
8:	.161	0.00	0	0	0	0	1	0
9:	-.145	0.00	0	0	0	0	0	1
10:	.0129	.0129	0	0	0	0	0	0
11:	.0119	.0119	0	0	0	0	0	0
12:	-.004	-.004	0	0	0	0	0	0
13:	-.0005	-.0005	0	0	0	0	0	0
14:	.0005	.0005	0	0	0	0	0	0

Again, Y is not known completely, but Y' and N' can be found from the collected data. After Y', W', M' and N' have been found, we can use (4) to find the vector of missing points m; that is

$$m = (M'-N')^{+}(Y'-W'f)$$

where

$$(M'-N')^{+}=\left[(M'-N')^{T}(M'-N')\right]^{-1}(M'-N')^{T} \quad \text{(pseudo inverse)}$$

All the above calculations were performed using the EASY interactive program (see appendix on software support); it was verified that m(1)...m(6) was identical to the original y(6)...y(11) data points.

Appendix II

```
C
C ------------------------------------------------
C Main Program for Simulating an ARMA Model Response
C ------------------------------------------------
C
C       The flow chart:
C       (1)  Get the coefficients of the model from the user
C       (2)  Let the user specify an input sequence u(i)
C       (3)  Determine the response of the ARMA model
C       (4)  Save the results on the disc for later processing
C
C       a, b, U, Y are in EASYPACK format
C       pars is converted in EASYPACK format
C
C  George Kontopidis, Oct 80, April 81
C
        Dimension a(12),b(12),pars(50), U(258),Y(258),Z(258)
        Common maxwrk,worksp(50)
        maxwrk=50
C
        Call MINP( a, 'Enter auroregressive coefficients $'  )
        Call MINP( b, 'Enter moving average coefficients $'  )
        ST = RGET(    'Enter variance of measurement noise $ ')
C
C...get an input sequense
C    u(0),u(1),...,u(NS-1)
C
        Call MGET( 'ARMAINP.DAT$', U )
        NS = NDIM(U)
Check   NS = IGET(    'Enter simulation steps (less than 256) $')
C
        N = NDIM( a )
        M = NDIM( b )
C
C...initiate the ARMA model and the random generator
C
        Call ARMINI( a(3), b(3), N,M, pars )
        Call RANDUL(30117)
C
C...generate the output sequnce
C    y(1),y(2),...,y(NS)
```

```
C
        Call SETDIM( Y, NS, 1 )
        Do 20 i=1,NS
                i11=i-1
                i22=i+2
                Y(i22)=ARMA( U(i22), pars) + GAUSS( 0., st )
                Write(5,101) i11,U(i22),i,Y(i22)
101     Format(' u(',I3,')=',1G12.2,'  y(',I3,')=',1G12.2)
20      Continue
C
C...save the results on disc
C
        NM2=N+M+2
        Call PACK( pars, NM2, NM2, 1 )
        Call MSTORE( 'ARMAPAR.DAT$', pars )
        Call MSTORE( 'ARMAINP.DAT$', U )
        Call MSTORE( 'ARMAOUT.DAT$', Y )
Check   Call MPRN( Y,'The response is...$')
        END

C
C       ---------------------------------
C       Subroutine ARMINI( a,b,N,M,pars )
C       ---------------------------------
C
C Purpose: Initiates the ARMA model parameters in pars.
C          Sets initial state 0.  The notation used is
C          y(k) = a(1)y(k-1)+...+a(N)y(k-N)+
C                 b(1)u(k-1)+...+b(M)u(k-M)
C
C Usage:   Call ARMINI( a,b,N,M,pars )
C
C a(1:N)   Coefficients of the autoregressive part of the model
C b(1:M)   Coefficients of the moving average part of the model
C pars     (Output) work space of size 2(N+M+1) containing:
C          pars(1)=N, pars(2)=M, pars(3:2+N)=a(i)'s
C          pars(3+N:2+N+M)=b(i)'s, pars(3+N+M:2+2N+2M)= old states.
C
C Required Libraries: EASYPACK
C
C George Kontopidis, Sep 80, March 81
C
```

```
      Subroutine ARMINI( a, b, N, M, pars )
      Dimension a(1), b(1), pars(1)
      Common maxwrk,w(1)
C
      NM=N+M
      pars(1) = N
      pars(2) = M
C
C...L always points to the next free location in pars
C
      L=3
C
C...transfer a(i)'s to pars
C
C     WRITE(5,101) (A(J),J=1,N)
      Call VEQU( pars(L), a(1), N )
      L=L+N
C
C...transfer b(i)'s to pars
C
      Call VEQU( pars(L), b(1), M )
      L=L+M
C
C...initiate the work space to 0 (initial conditions)
C
      Call VZERO( pars(L), NM )
C     L=L+NM
      Return
      END
C
C     ---------------------------
C     Function ARMA( u, pars   )
C     ---------------------------
C
C Purpose:  Following the notation given in ARMINI subroutine,
C           this function generates y(k+1) when called with u(k).
C
C Usage:    yk = ARMA( uk, pars )
C
C  uk       Input real number representing the value u(k)
C  pars     Input-Output real vector of size 2(N+M+1) used as work
```

```
C           area.  Initially, pars must be set by ARMINI.
C
C Required Routines:  EASYPACK
C
C George Kontopidis,  Sept 80, April 81
C
        Real Function ARMA( u, pars )
        Real pars(1)
        Common maxwrk,W(1)
C
        N=pars(1)
        M=pars(2)
C
C ia points the a's in stack
C ib points the b's in stack
C iy points the y(k-1),...,y(k-N)
C iu points the u(k-2),...,u(k-M-1)
C
        ia=3
        ib=ia+N
        iy=ib+M
        iu=iy+N
C
C...push down the stack of input values
C   New state: u(k-1) <new-input>, u(k-2),...u(k-M)
C
        i=iu+M-1
10      If( i.le.iu ) Go to 11
        iold = i-1
        pars( i )=pars( iold )
        i=i-1
        Go to 10
11      pars( iu ) = u
C
C...find the output y(k)
C
        ARMApr   = VDOT( pars(ia), pars(iy), N)
     1           + VDOT( pars(ib), pars(iu), M)
C
C...push down the stack of the output values
C   New state: y(k) <new output>,y(k-1),...,y(k-N+1)
```

```
C
        i=iy+N-1
30      If ( i.LE.iy ) go to 31
        iold = i-1
        pars( i )=pars( iold )
        i=i-1
        Go to 30
31      pars( iy ) = ARMApr
        ARMA=ARMApr
        Return
        END

C       ------------------------------------------
C       ARMA COEFFICIENTS IDENTIFICATION ALGORITHM
C       ------------------------------------------
C
C       Algorithm:  Reads the input sequence  ARMAINP.DAT
C                   Reads the output sequence ARMAOUT.DAT
C                   Assumes that output data 11:12 are missing
C                   Finds the ARMA coefficients using the remaining
C                   Compares these esimates with the true parameters
C
C       George Kontopidis, April 81
C
        Dimension W(65,10), U(67),Y(67),pars(15),B(15),E(67)
        Common maxwrk, worksp(100)
C
C...Read the data
C
        ic=65
        maxwrk=100
        Call MGET('ARMAINP.DAT$',   U )
        Call MGET('ARMAOUT.DAT$',   Y )
        Call MGET('ARMAPAR.DAT$', pars)
Check   Call MPRN( U,'Verify the input vector$')
Check   Call MPRN( Y,'Verify the output vector$')
Check   Call MPRN( pars,'Verify the parameter vector$')
        Call PRN ('Enter estimates of the order of the model N,M $')
        Read( 5,100) N,M
        Call PRN ('Enter index limits of missing points 11,12 $')
        Read( 5,100) 11,12
```

```
100     Format(2I5)
C
C...Define some constants for later
C
        nf    = NDIM(Y)
        nfn   = nf-N
        nm    = N+M
        i11   = l1-N
        i22   = l2
C
C...formulate the W matrix.  Put the y's first
C
        Do 10 i=1,nfn
          Do 10 j=1,N
            it=N+i-j+2
            W(i,j)=Y(it)
10      Continue
C
C...then put the u's
C
        Do 20 i=1,nfn
          Do 20 j=1,M
            it=N+i-j+3
            jN=j+N
            W(i,jN)=U(it)
20      Continue
C
C...Find the second side of the equations
C
        Call VEQU( Y(3),Y(N+3),nfn )
        Call SETDIM( Y,nfn,1 )
C
C...If there exist bad points, put a 0 there
C
        If(l1.LE.0) Go to 301
        If(i11.LT.2 .OR. i22.GT.nfn)  Call PRN('Bad Limits..$')
          Do 310 i=i11,i22
                Do 300 j=1,nm
                  W(i,j)=0.
300             Continue
          Y(i+2)=0.
```

```
310       Continue
301     Continue
C
        Call PACK( W,ic,nfn,nm)
Check   Call MPRN( W, 'Verify the W matrix$')
        Call MSTORE('ARMAW.DAT$',W)
Check   Call MPRN(Y, 'Verify the Y vector$')
        Call MSTORE('ARMAY.DAT$',Y)
C
C...Solve the linear equations
C
        i1 = NDIM(W)
        i2 = MDIM(W)
        Write(5,101) i1,i2
101     Format(' Solving a system of ',I3,' equations by ',
     1        I2,' unknowns')
        Call LSQ( B,E, W,Y )
C
C...Print the solution
C
        Call MPRN( pars, 'N, M, true a-pars, true b-pars$')
        Call IPRN( N,'=given estimate of N$')
        Call IPRN( M,'=given estimate of M$')
        Call MPRN( B,'The calculated parameters (a and b)$')
        xn2= VNORM2( E(3),E(1) )
        Call RPRN( xn2,'= norm2 of the estimation error$')
        End

C
C       ----------------------------
C       Generate a GAUSSIAN deviate
C       ----------------------------
C
C       amean = mean value,   stdev = standard deviation
C
        Real Function GAUSS( amean, stdev )
        Logical i
C
        GAUSS = 0.
        Do 10 i=1,12
10      GAUSS = GAUSS + RANDUL( 0 )
```

```
C
        GAUSS = (GAUSS-6.)*stdev +amean
        RETURN
        END
C
C       ------------------------
C       Random Number Generator
C       ------------------------
C
C Generates uniform random numbers using the Linear Congruential
C Method (Knuth, vol 2).  The general formula is
C
C       x(N+1) := ( a*x(N)+b ) mod M
C
C where a, b and M are 'appropriate constants'.
C
C Usage:        Call RANDUL( large-int )     to seed (initiate)
C               x =  RANDUL( 0 )             to generate a point.
C
C George Kontopidis, Aug 1980
C
        Real Function RANDUL( i )
        Integer xn
        Real    norm
        If( i.EQ.0 )  Go to 10
C
C...seed the generator for my computer
C
        norm = 2.**15-1
        xn   = i
C
C...generate a point
C
10      xn    = X'4205'*xn + 30947
        RANDUL= abs( float(xn)/norm )
        Return
        END
```

```
C       ------------------------------------------------
C       Main Program to Test the Random Generator RANDUL
C       ------------------------------------------------
C
C       George Kontopidis, April 81
C
        Dimension pdf(202)
        irun=1
1       Call IPRN(irun,'  Random Generator pdf Test $')
        npoint = IGET('Number of points $')
        nint   = IGET('Number of intervals (<200) $')
C
C...Initiation of working areas
C
        Call RANDUL( 30201 )
        Call VZERO( pdf, nint )
        xu=1./float(nint)
        fu=1./( float(npoint)*xu )
        Call PRN('Next line: each # indicates 500 RANDUL calls$')
        Call CRLF
C
C...PDF computation
C
        Do 300 i=1,npoint
C
C...print a dot every 500 points
C
                x=RANDUL( 0 )
                If( mod(i,500).EQ.0  ) Call PRN$('#$')
                Do 299 m=1,nint
                        t0=float(M-1)
                        t1=float(M)
                        If ((xu*t0  .LE.  x) .AND. (x .LE. t1*xu))
     1                  pdf(m)=pdf(m)+fu
299             Continue
300     Continue
C
C...perform the chi-square test
C
        xsq=0.
        Do 600 i=1,nint
```

```
600     xsq = xsq+(pdf(i)-1.)*(pdf(i)-1)
        xsq = xsq * float(npoint)/float(nint)
C
C...Print the pdf distribution
C
        Call PACK( pdf, nint, nint, 1 )
        Call RPRN( xsq, '= chi square $')
        Call MPRN( pdf, 'The result distribution is...$')
        Call CRLF
        irun=irun+1
        Go to 1
        END

C               ----------------------------------
C               FINDS THE LEAST SQUARES SOLUTION
C               ----------------------------------
C
C Purpose:  To find the least squares solution of the system
C
C                       X * B = Y + E
C               where  B minimizes  (X*B-Y)'(X*B-Y)
C
C           This subroutine has been tested in finding the minimum
C           norm solution and the least squares solution of 'big'
C           matrices of dimensions up to 128 by 32 using single
C           precision operations only.  Oveflows and underflows
C           are treated properly and partial pivoting is performed.
C           The main reference used was:
C
C               LINPACK user's guide by Dongarra e.l. SIAM 1979
C
C           It has been modified to the EASYPACK format to simplify
C           the usage.  The routine calls the SQRDC and SQRSL
C           routines to perform the Q-R decomposition and the least
C           squares formulation.  The result has been found more
C           accurate than the GINV routine in EASYPACK and LLSQAR
C           routine of the IMSL library.
C
C Parameters:
C   B       (Output) Least squares solution of the system X*B=Y+E
C   E       (Output) Error of the least squares E=X*B-Y
```

```
C   X       (Input ) Coefficient matrix X
C           On return X is DESTROYED (for details see LINPACK p. 9-11
C   Y       (Input ) Right side of equations
C
C       George Kontopidis, April 81
C
        Subroutine LSQ( B,E,X,Y )
        Real X(1),Y(1),B(1),E(1)
        Integer P
        Common maxwrk,iwork(1)
C
C       iwork(1:P) is used as jpvt(1:P) (pivoting indices)
C       work(iw1:iw1+P) = QRAUX
C       work(iw2:iw2+P) = work
C       maximum work space used: 2.5*P real memory storage units
C
        N   = NDIM(X)
        LDX = N
        P   = MDIM(X)
        If (maxwrk.LT.2*P+(P+1)/2) Call PRN('Small wrksp for LSQ$')
        iw1 = P+1
        iw2 = iw1 + 2*P
        Call SETDIM( E, N, 1)
        Call SETDIM( B, P, 1)
C
C...Initialize jpvt so that all columns are free
C
        Do 10 j=1, P
10      iwork(j)=0
C
C...QR decomposition of X
C
        Call SQRDC( X(3),LDX,N,P, iwork(iw1), iwork(1), iwork(iw2),1)
C
C...Solve the least squares problem
C
        Call SQRSL( X(3),LDX,N,P, iwork(iw1), Y(3), E(3),E(3),B(3),
     1              E(3),E(3),110,info)
        If (info.NE.0) Call PRN('Singular Array...$')
C
C...initialize jpvt
```

```
C
        write(5,102) (B(j+2),j=1,P)
102     Format(' ',10G13.6)
        write(5,101) (iwork(j),j=1,P)
101     Format(' ',20I4)
        Do 40 j=1,P
40      iwork(j) = -iwork(j)
C
C...unscramble the solution
C
        Do 70 j=1,P
                If(iwork(j).GT.0) go to 70
                k = -iwork(j)
                iwork(j)=k
50              Continue
                  If(k.EQ.j) Go to 60
                  temp  = B(j+2)
                  B(j+2)= B(k+2)
                  B(k+2)= temp
                  iwork(k)=-iwork(k)
                  k=iwork(k)
                Go to 50
60              Continue
70      Continue
C
C...inverse the sign of E
C
        Call VSCALE( E(3),-1.,E(1))
        Return
        END

C       ----------------------------------------
C       Q-R Factorization of an N by P matrix X
C       ----------------------------------------
C
C  X            Real (LDX:N by P) input.  On output X contains the
C               Q-R decomposition required by SQRSL.
C  jpvt(1:P)    If job=0 jpvt is not used.  Else defines the order
C               of searching for pivoting
C  work(1:P)    If job=0 work is not used.
C  QRAUX(1:P)   (output) Used by SQRSL to recover the Q-R decompo-
```

```
C                sition.
C  job           Pivoting flag; if job=1, pivoting is performed
C
C  Reference:    LINPACK Chapter 9 and appendix C
C
C  George Kontopidis, April 81
C
        Subroutine SQRDC( X,LDX,N,P,QRAUX,jpvt,work,job )
        LogicaL swapj, negj
        Integer LDX,N,P,job
        Integer jpvt(1)
        Integer j, jp, L, Lp1, Lup, maxj, pL, pu
        ReaL    X(LDX,1), QRAUX(1), work(1)
        ReaL    maxnrm, tt
        ReaL    sdot,nrmxL, t
C
        pL=1
        pu=0
        If (job.EQ.0) Go to 60
C
C...Pivoting has been requested.  Rearrange the coLumns according
C...to jpvt
C
          Do 20 j=1,P
                swapj = jpvt(j).GT.0
                negj  = jpvt(j).LT.0
                jpvt(j)= j
                If (negj) jpvt(j) = -j
                If (.NOT.swapj) Go to 10
C replaced...   If(j.NE.pL) Call SSWAP(N,X(1,pL),1,X(1,J),1)
                If(j.NE.pL) Call VSWAP( X(1,pL),X(1,j),N)
                  jpvt(j)  = jpvt(pL)
                  jpvt(pL) = j
                  pL       = pL+1
10              Continue
20        Continue
C
          pu = p
          Do 50 jj=1,P
                j = p-jj+1
                If (jpvt(j).GE.0) Go to 40
```

```
                  jpvt(j)  = -jpvt(j)
                  If (j.EQ.pu)    Go to 30
C replaced...        Call SSWAP(N,X(1,pu),1,X(1,j),1)
                     Call VSWAP(X(1,pu), X(1,j), N)
                     jp       = jpvt(pu)
                     jpvt(pu) = jpvt(j)
                     jpvt(j)  = jp
30                Continue
                  pu = pu -1
40              Continue
50        Continue
60      Continue
C
C...Compute the norms of the free coLumns
C
        If(pu.LT.pL) Go to 80
          Do 70 j=pL,pu
C replaced..QRAUX(j) = SNRM2(N,X(1,j),1)
            QRAUX(j) = VNORM2(X(1,j),N)
            work(j)  = QRAUX(j)
70        Continue
80      Continue
C
C...Perform the HouseholDer reduction of X
C
        Lup = MIN0 (N,P)
        Do 200 L=1,Lup
          If(L.LT.pL.OR.L.GE.pu) Go to 120
C
C...Locate the coLumn of Largest norm and bring it into the pivot
C   position.
C
                maxnrm = 0.0
                maxj   = L
                Do 100 j=L,pu
                  If (QRAUX(j).LE.maxnrm) Go to 90
                        maxnrm = QRAUX(j)
                        maxj   = j
90                Continue
100             Continue
                  If (maxj.EQ.L) Go to 110
```

```
C replaced...        Call SSWAP(N,X(1,L),1,X(1,maxj),1)
                     Call VSWAP( X(1,L),X(1,maxj),N )
                     QRAUX(maxj) = QRAUX(L)
                     work (maxj) = work(L)
                     jp          = jpvt(maxj)
                     jpvt(maxj)  = jpvt(L)
                     jpvt(L)     = jp
110                Continue
120         Continue
            QRAUX(L) = 0.0
            If(L.EQ.N) Go to 190
C
C...Compute the HouseholDer transformation for coLumn L
C
                  Loc1   = N-L+1
C replaced...     nrmxL  = SNRM2( Loc1, X(L,L), 1)
                  nrmxL  = VNORM2( X(L,L),Loc1 )
                  If (nrmxL.EQ.0.0) Go to 180
                    If (X(L,L).NE.0.0) nrmxL = SIGN( nrmxL,X(L,L))
                    Loc1   = N-L+1
                    rLoc   = 1.0/nrmxL
C replaced...       Call SSCAL( Loc1, rLoc1, X(L,L), 1)
                    Call VSCALE( X(L,L), rLoc, Loc1 )
                    X(L,L) = 1.0 +X(L,L)
C
C...AppLy the transformation to the remaining coLumns, updating
C   the norms
C
                    Lp1    = L+1
                    If (p.LT.Lp1) Go to 170
                    Do 160 j=Lp1,p
                          Loc1 = N-L+1
C replaced...             T    = -SDOT(Loc1,X(L,L),1,X(L,j),1)/X(L,L)
                          T    = -VDOT( X(L,L),X(L,j),Loc1) /X(L,L)
C replaced...             Call SAXPY( Loc1, T,X(L,L),1,X(L,j),1)
                          Call VSADD( X(L,j), T, X(L,L), Loc1 )
                          If (j.LT.pL.OR.j.GT.pu) Go to 150
                          If (QRAUX(j).EQ.0.0)    Go to 150
                            tt = 1.0-(ABS(X(L,j))/QRAUX(j))**2
                            tt = AMAX1( tt, 0.0 )
                            t  = tt
```

```
                        tt = 1.0+0.05*tt*(QRAUX(j)/work(j))**2
                        If (tt.EQ.1.0) Go to 130
                              QRAUX(j)=QRAUX(j)*SQRT(t)
                              Go to 140
130                     Continue
                              Loc1    = N-L
C replaced...                 QRAUX(j)=SNRM2(Loc1,X(L+1,j),1)
                              QRAUX(j)=VNORM2( X(L+1,j),Loc1 )
                              work(j) = QRAUX(j)
140                     Continue
C
150                   Continue
160                Continue
170                Continue
C
C...Save the transformation
C
                      QRAUX(L) = X(L,L)
                      X(L,L)   = -nrmxL
180              Continue
190         Continue
200      Continue
         Return
         END

C -------------------------------------------------------------
C Coordinate Transformations, Projections and Least Squares
C -------------------------------------------------------------
C
C  X(LDX:N by P)  output of SQRDC
C  QRAUX(1:P)     auxiliary output from SQRDC
C  Y(1:N)         input real vector
C  job            Specifies what to be done.  Job has the decimal
C                 expansion ABCDE with the following meaning
C                      A .NE. 0  compute  QY=Q*Y
C                      C .NE. 0  compute  B (least squares solution)
C                      D .NE. 0  compute  RSD (lsq residual)
C                      E .NE. 0  compute  XB ( X*B )
C                   B,C,D,E.NE. 0  compute  QTY (Q-transp*Y)
C
C   Refer to LINPACK Appendic C.107
```

```
C
C   George Kontopidis, April 81
C
      Subroutine SQRSL(X,LDX,N,K,QRAUX,Y,QY,QTY,B,RSD,XB,job,info)
      Logical cb,cqy,cqty,cr,cxb
      Integer LDX,N,K,job,info
      Integer i,j,jj,ju,kp1
      Real    SDOT,t,temp
      Real    X(LDX,1),QRAUX(1),Y(1),QY(1),QTY(1),B(1),RSD(1),XB(1)
C
C...set info flag
C
      info = 0
C
C...Determine what is to be computed
C
      cqy  =job/10000          .NE.0
      cqty =mod(job,10000)     .NE.0
      cb   =mod(job,1000 )/100 .NE.0
      cr   =mod(job,100  )/10  .NE.0
      cxb  =mod(job,10   )     .NE.0
      ju   =MIN0(k,N-1)
C
C...special action when N=1
C
      If (ju.NE.0) Go to 40
        If(cqy)  QY(1) =Y(1)
        If(cqty) QTY(1)=Y(1)
        If(cxb)  xb(1) =Y(1)
        If(.NOT.cb) Go to 30
              If(X(1,1).NE.0.0) Go to 10
                info=1
              Go to 20
10            Continue
                B(1)=Y(1)/X(1,1)
20            Continue
30      Continue
        If (cr) rsd(1)=0.0
      Go to 250
40    Continue
C
```

```
C...Set up to compute QY or QTY
C
        If  (cqy)     Call VEQU( QY, Y, N )
C replaced...         Call SCOPY(N,Y,1,QY,1)
        If  (cqty)    Call VEQU( QTY, Y, N )
C replaced...         Call SCOPY(N,Y,1,QTY,1)
        If(.NOT.cqy) Go to 70
C
C...Compute QY
C
          Do 60 jj=1,ju
            j=ju-jj+1
            If (QRAUX(j).EQ.0.0) Go to 50
                temp   = X(j,j)
                X(j,j) = QRAUX(j)
                loc1   = N-J+1
C replaced...   T      =-SDOT(loc1,X(j,j),1,QY(j),1)/X(j,j)
                T      =-VDOT( X(j,j),QY(j),Loc1) /X(j,j)
C replaced...   Call SAXPY(loc1,T,X(j,j),1,QY(j),1)
                Call VSADD( QY(j),T,X(j,j),Loc1 )
                X(j,j) = temp
50          Continue
60        Continue
70      Continue
        If(.NOT.cqty) Go to 100
C
C...Compute tanspose(Q)*Y
C
          Do 90 j=1,ju
            If(QRAUX(j).EQ.0.0) Go to 80
                temp   = X(j,j)
                X(j,j) = QRAUX(j)
                loc1   = N-j+1
C replaced...   T      =-SDOT(loc1,X(j,j),1,QTY(j),1)/X(j,j)
                T      =-VDOT( X(j,j),QTY(j),Loc1) /X(j,j)
C replaced...   Call SAXPY(loc1,t,X(j,j),1,QTY(j),1)
                Call VSADD( QTY(j),t,X(j,j),Loc1)
                X(j,j) = temp
80          Continue
90        Continue
100     Continue
```

```
C
C...Set up to compute B, RSD, or XB.
C
        If(cb)  Call VEQU( B,QTY,k )
C replaced...   Call SCOPY(k,QTY,1,B,1)
        kp1=k+1
        If(cxb) Call VEQU( XB,QTY,k )
C replaced...   Call SCOPY(k,QTY,1,XB,1)
        loc1=N-k
        If(cr.AND.k.LT.N) Call VEQU( RSD(kp1),QTY(kp1),Loc1)
C replaced...             Call SCOPY(loc1,QTY(kp1),1,RSD(kp1),1)
        If(.NOT.cxb.OR.kp1.GT.N) Go to 120
          Do 110 i=kp1,N
            XB(i)=0.0
110       Continue
120     Continue
        If(.NOT.cr) Go to 140
          Do 130 i=1,k
            RSD(i)=0.0
130       Continue
140     Continue
        If(.NOT.cb) Go to 190
C
C...Compute B
C
          Do 170 jj=1,k
            j=k-jj+1
            If(X(j,j).NE.0.0) Go to 150
                info=j
C...            exit
                Go to 180
150         Continue
            B(j)=B(j)/X(j,j)
            If(j.EQ.1) Go to 160
                T = -B(j)
                loc1=j-1
C replaced...   Call SAXPY(loc1,T,X(1,j),1,B,1)
                Call VSADD( B,T,X(1,j),Loc1)
160         Continue
170       Continue
180       Continue
```

```
190     Continue
C
C...Compute RSD or XB as required
C
          Do 230 jj=1,ju
            j=ju-jj+1
            If(QRAUX(j).EQ.0.0) Go to 220
                temp   = X(j,j)
                X(j,j) = QRAUX(j)
                If(.NOT.cr) Go to 200
                  loc1=N-J+1
C replaced...     t=-SDOT(loc1,X(j,j),1,RSD(j),1)/X(j,j)
                  t=-VDOT( X(j,j),RSD(j),Loc1) /X(j,j)
C replaced...     Call SAXPY(loc1,t,X(j,j),1,RSD(j),1)
                  Call VSADD( RSD(j), t, X(j,j), Loc1 )
200             Continue
                If(.NOT.cxb) Go to 210
                  loc1=N-j+1
C replaced...     t=-SDOT(loc1,X(j,j),1,XB(j),1)/X(j,j)
                  t=-VDOT( XB(j),X(j,j),Loc1) /X(j,j)
C replaced...     Call SAXPY(loc1,t,X(j,j),1,XB(j),1)
                  Call VSADD( XB(j), t, X(j,j),Loc1 )
210             Continue
                X(j,j) = temp
220           Continue
230         Continue
240       Continue
250       Continue
          Return
          End

C         ---------------------------------------
C         FINDS THE EUCLIDEAN NORM OF A VECTOR
C         ---------------------------------------
C
C         A common problem in finding the sqrt of the sum of the
C         squares of a vector is the overflows and underflows.  These
C         can be avoided by several techniques discussed in LINPACK
C         Appendix D.  This code is an adaptation for
C
C            Z-80 with Microsoft Fortran,
```

```
C       Smallest real  0.1000...0 * 2 **(-127)
C       Largest real   0.1111...0 * 2 **(+128)
C       Epsilon const  0.0000...1 * 2 **(+000)
C
C       small = 0.1469368  E-38
C       large = 1.7014117  E+38
C       epsil = 0.14693681 E-38
C
C       George Kontopidis, April 81
C
        Real Function VNORM2( SX, N )
        Real SX(1)
        Data cutlo, cuthi / 4.44E-16, 1.304E+19 /
C
C  the DEC-10 uses, 2**(-102) and 2**(127)
C
10      Assign 30 to NEXT
        sum  = 0.0
C
C...The main loop starts here
C
        i    = 1
20      Go to NEXT, (30,50,70,110)
30      If (ABS(sx(i)).GT.cutlo) Go to 85
        Assign 50 to NEXT
        xmax = 0.0
C
C...Phase 1.  The Sum is zero
C
50      If (SX(i).EQ.0.0) Go to 200
        If (ABS(SX(i)).GT.cutlo) Go to 85
C
        Assign 70 to NEXT
        Go to 105
C
100     i = j
        Assign 110 to NEXT
        sum  = (sum/sx(i))/sx(i)
105     xmax = ABS( SX(i) )
        Go to 115
C
```

```
C...Phase 2.  Sum is small.  Scale to avoid underflow
C
70      If( ABS(SX(i)).GT.cutlo) Go to 75
C
C...Common Code for phases 2 and 4.  Scale to avoid overflow
C
110     If( ABS(SX(i)).LE.xmax) Go to 115
        sum  = 1.0 + sum*(xmax/sx(i))**2
        xmax = ABS(SX(i))
        Go to 200
C
115     sum  = sum + (SX(i)/xmax)**2
        Go to 200
C
75      sum  = (sum*xmax) *xmax
85      hitest=cuthi/float(N)
C
C...Phase 3.  The Sum is mid-range.  No scaling
C
        Do 95 j=i,N
        If(ABS(SX(j)).GE.hitest) Go to 100
95         sum = sum+SX(j)**2
        VNORM2 = SQRT( sum )
        Go to 300
C
200     Continue
        i = i+1
        If (i.LE.N) Go to 20
C
C...End of Main Loop
C
        VNORM2 = xmax*SQRT( sum )
300     Continue
        Return
        End
```

Notes and References II

[1] Brogan William. Modern Control Theory. Quantum Publishers, Inc., New York, 1974, Chapter 5.

[2] Dongarra J.J., C.B. Moler, J.R. Bunch, G.W. Stewart. Linpack, User's Guide. Siam, Philadelphia 1979.

[3] Sage Andrew P., James L. Melsa. Estimation Theory with Applications to Communications and Control. McGraw-Hill Book Company, New York, 1971, Chapter 7.

[4] Schweppe Fred C., Uncertain Dynamic Systems. Prentice-Hall Englewood Cliffs, New Jersey, 1973, pp. 151-247.

[5] Astrom K. J., P. Eykhoff, System Identification- A Survey, Automatica, vol. 7., Pergamon Press, 1971, pp. 123-167.

[6] Urlych Tad J., Thomas N. Bishop, Maximum Entropy Spectral Analysis and Autoregressive Decomposition, Reviews of Geophysics and Space Physics, vol. 13, Feb. 1975, pp. 184-199.

[7] Smylie, D. E., G. K. Clarke and T. J. Ulrych. Analysis of Irregularities in the Earth's Rotation. Methods in Computational Physics, vol. 13, pp. 391-430, Academic Press, New York, 1973.

[8] Jenkins G. M., D. G. Watts. Spectral Analysis and its Applications. Holden-Day, San Francisco, 1969.

[9] Bendat Julius S. Random Data: Analysis and Measurement Procedures. Wiley-Interscience, New York, 1971.

[10] Willsky Alan, Relationships Between Digital Signal Processing and Control and Estimation Theory. Proc. of IEEE, vol. 66, No. 9, Sept. 1978, pp. 996-1017.

[11] Hsia T. C., On Least Squares Algorithms for System Parameter Identification, IEEE Trans. on Automatic Control, vol. AC-22, pp. 104-108.

CHAPTER III

INTERPOLATION
USING BANDLIMITING ASSUMPTIONS

Outline of Chapter III

1. Definition of the Discrete Fourier Transform
2. The Interpolation Theorem
3. Sensitivity of the Estimates
4. Iterative Interpolation Algorithms
5. Simulation Results

Appendix III

Notes and References III

Outline of Chapter III

This chapter deals with a special class of nonuniformly sampled systems. Consider the case of collecting data uniformly but because of external reasons (i.e. instrumentation failure, unpredicted power failure) some of the data are missing or they have been collected but it is known that they are bad. This situation is treated as a special case of a nonuniformly sampled system. We propose an algorithm that can be used (under certain conditions), to interpolate the missing points based on bandlimiting assumptions about the original signal.

One of the most restrictive assumptions we make is that the indices of the missing or bad points are known; this corresponds to the assumption that the time of failure is known. The composite problem dealing with unknown times and incorrect data, is much more complicated and the solution requires a probabilistic formulation. However, in many practical cases one can easily tell or easily detect which measurements are faulty; in this case, the theory given in this chapter can be applied and a very simple algorithm can be used to solve the problem.

A second assumption is also made about the bandwidth of the original signal and the sampling rate. In order to apply our results it is necessary to know that the discrete Fourier transform (DFT) of the time series has nulls at several known frequency constituents. This assumption is true in case of bandlimited signals sampled at a rate which is a multiple of the Nyquist rate. Actually, the case of bandpass signals or any other case at which we know apriori nulls in the frequency domain can be treated with the same theory.

The chapter is organized in 5 sections. In the first we define the DFT transform and we summarize the properties of the DFT matrix (W). In the second we formulate and prove the interpolation theorem; some

computational aspects are also discussed. Section 3 is an error analysis for the interpolation theorem. Section 4 is an extension of a continuous time iterative extrapolation algorithm to discrete time. Finally, in section 5 we include simulation results and a FORTRAN program based on the previous theory. At last, a list of the references appropriate to the subject of this chapter is included.

1. Definition of the Discrete Fourier Transform

Let $x=(x_0, x_1,\ldots x_{N-1})$ be a vector of N elements. We define a new vector $a=(a_0, a_1,\ldots,a_{N-1})$ by the formula

$$a_n = \sum_{m=0}^{N-1} x_m w^{nm} \qquad w=\exp(-j2\pi/N) \qquad n=0,1,\ldots,N-1$$

The vector a is called the Discrete Fourier Transform (DFT) of x. The above equation can be written in the matrix form a=Wx where W is a square matrix (called the DFT matrix) with elements $\{\exp(-j2\pi nm/N)$ $n,m=0,1,\ldots,N-1\}$. A shorthand notation is used for the matrix W by writing the product nm in place of the (n,m)-th element. Using this convention, the matrix equation a=Wx is written explicitly:

$$\begin{bmatrix} 0 \\ 1 \\ 2 \\ \cdots \\ N-1 \end{bmatrix} a = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & (N-1) \\ 0 & 2 & \cdots & 2(N-1) \\ \cdots & & \cdots & \cdots \\ 0 & (N-1) & \cdots & (N-1)(N-1) \end{bmatrix} w \begin{bmatrix} 0 \\ 1 \\ 2 \\ \cdots \\ N-1 \end{bmatrix} x \qquad \text{(i)}$$

Properties of the Matrix W

(1) The (n,m)-th element of W is equal to $\cos(2\pi nm/N)-j\sin(2\pi nm/N)$

(2) The inverse of W is given by

$$W^{-1} = \frac{1}{N} W^{*}$$

where the asterisk denotes the conjugate-transpose of a matrix.
(The proof of (2) is trivial: use the DFT inversion theorem

$$x_m = \frac{1}{N}\sum_{n=0}^{N-1} a_n w^{-nm} \qquad)$$

(3) W is a symmetric matrix (because $\exp(-jnm\pi/N)=\exp(-jmn\pi/N)$)

(4) The sum of the elements of each row or column of W except the first one is 0. This follows from

$$\sum_{m=0}^{N-1} w^{m(k-n)} = \frac{w^{N(k-n)}-1}{w^{k-n}-1} = \begin{cases} 0 & \text{for } k\neq n \\ N & \text{for } k=n \end{cases}$$

(5) The n-th row is the conjugate of the N-n row, and the n-th column is the conjugate of the N-n column (for n=1,2,...,N-1). Proof:

the n-th row has elements w^{nk} $\qquad k=0,\ldots,N-1$

the N-n row has elements $w^{(N-n)k}$

But $\exp[-j2\pi(N-n)k/N]=\exp[j2\pi nk/N]$.

(6) Define the periodic expansion of x by:

$$x^{(S)} = (\underbrace{x_0, x_1, \ldots x_{N-1}}_{\text{group 1}}, \underbrace{x_0, x_1, \ldots, x_{N-1}}_{\text{group 2}}, \ldots, \underbrace{x_0, x_1, \ldots x_{N-1}}_{\text{group S}})$$

Then, the following property is true:

$$a_n = \left(\sum_m w^{nm} x_m^{(S)} \right) \exp(j2\pi r/N)$$

where the summation index m takes values r,r+1,...,r+N-1 for an arbitrary r (within the indices of the augmented x)

(7) Assume that N is an even number (which is the most common case); then there exists a row r_t (and a column c_t) called the mid row (column) with the property

$$r_t = r_t^* \qquad (c_t = c_t^*)$$

This means that the t=N/2 row (column) consists of real numbers. Very often it is convenient to use that row (column) as the 'base' of the indices; for example we write

$$r_{t-2}, r_{t-1}, r_t, r_{t+1}, r_{t+2}$$

in order to refer to the middle 5 rows of W.

(8) If r_t (c_t) is the mid row (column) then,

$$r_{t-n} = r_{t+n}^* \qquad (c_{t-n} = c_{t+n}^*)$$

for every n within in the limits 0 and (N/2)-1.

(9) Define the X matrix in the following manner

$$X=\begin{bmatrix} x_0 & x_1 & \cdots & x_{N-1} \\ x_{N-1} & x_0 & \cdots & x_{N-2} \\ \cdots & & \cdots & \\ x_1 & x_2 & \cdots & x_0 \end{bmatrix}$$

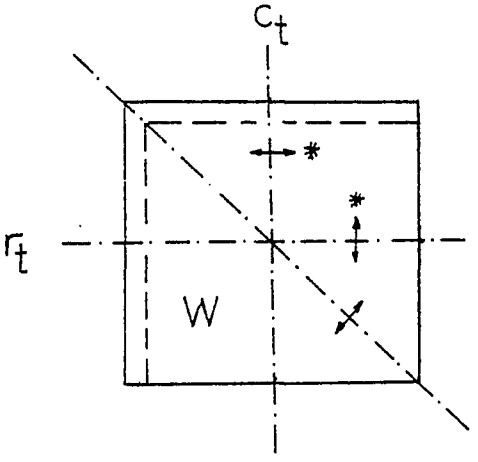
Then, the eigenvalues of X are the elements of the DFT of x and the modal matrix of X is W. Proof: Let v_m be the m-th column of W. Then

$$Xv_m = a_m v_m$$

for m=0,1,...,N-1. This means that a_m and v_m are an eigenvalue-eigenvector pair. Then, $X\,W=W\,\mathrm{diag}(a_0,a_1,\ldots,a_{N-1})$ which completes the proof.

Summary

The following diagram illustrates the symmetry and conjugate symmetry (*) lines within W.



2. The Interpolation Theorem

The interpolation theorem provides a new method of estimating 'missing' or 'bad' points of a time series that is known to be bandlimited. The assumptions, the assertion and the proof of the theorem follow.

Assumptions

(1) Let $x=(x_0,x_1,\ldots,x_{N-1})$ be a vector of N real numbers (time series) and

$$x_M=(x_{i_1},x_{i_2},\ldots x_{i_M})$$

be a subvector of x called the 'missing' (or 'bad') points of x. Let also

$$x_S=(x_{n_1},x_{n_2},\ldots x_{n_S})$$

be the subvector of x which remains after the x_M points have been removed. It is implied that

$$0\leq i_1<i_2<\ldots<i_M\leq N-1$$

$$0\leq n_1<n_2<\ldots<n_S\leq N-1$$

and

$$M+S=N.$$

(2) The total number of points N is assumed to be an even number. A

similar development can be done when N is odd but the first case is more usual in practical problems.

(3) The DFT of the time series x is assumed to have zero elements in the zone t-b,...,t+b (t=N/2). This assumption is satisfied when the continuous time signal from which the time series x was taken is bandlimitted and the sampling rate is several times the Nyquist frequency. (If the continuous time signal is generated from a bandstop -instead of a lowpass- system and the sampling frequency is high enough, then the DFT of x will have zeros within a band t'-b'...t'+b' with $t' \neq N/2$; the interpolation theorem can be easily modified to use this hypothesis instead).

Assertion: The Interpolation Theorem

Under the above assumptions, the vector of missing data x_M can be found by solving a set of linear equations

$$Ax_M + Bx_S = 0$$

where A and B are real matrices and they are defined in the proof of the theorem. The uniqueness or the multiplicity of the solutions depends on the rank of the matrices A and B; for a unique solution a necessary condition (but not sufficient) is

$$2b+1 \geq m$$

Proof

By doing column permutations the DFT equation a=Wx can be written:

$$a=[C_M|C_S]\begin{bmatrix} x_M \\ x_S \end{bmatrix}$$

where

$$C_M=(i_1,i_2,\ldots,i_M) \text{ -columns of } W$$

and

$$C_S=(n_1,n_2,\ldots,n_S) \text{ -columns of } W$$

Because of the third assumption the a vector has 0 elements with indices $t-b,t-b+1,\ldots,t,\ldots t+b-1,t+b$. Then,

$$A_C x_M+B_C x_S=0 \qquad \text{(i)}$$

where

$$A_C=[0|I_{2b+1}|0]C_M$$

$$B_C=[0|I_{2b+1}|0]C_S$$

Both A_C and B_C are complex matrices of 2b+1 rows. Now let

$$A_C=\begin{bmatrix} r_{t-b} \\ \cdots \\ r_t \\ \cdots \\ r_{t+b} \end{bmatrix}$$

be the row partition of A_C. Using the eighth property of W, we find

$$\text{real}(r_{t-i}) = \text{real}(r_{t+i})$$

$$-\text{imag}(r_{t-i}) = \text{imag}(r_{t+i})$$

Therefore, equation (i) can be written as a set of linear equations with real coefficients; that is, A_C is replaced by

$$A = \begin{bmatrix} \text{real}(r_{t-b}) \\ \cdots\cdots \\ \text{real}(r_{t-b+1}) \\ \cdots\cdots \\ r_t \\ \cdots\cdots \\ -\text{imag}(r_{t+b-1}) \\ \cdots\cdots \\ -\text{imag}(r_{t+b}) \end{bmatrix}$$

(r_t is real according to property (7))

In a similar manner, we can define B to be a collection of real and imaginary parts of rows of B_C. This concludes the proof of the theorem.

Computation of A and B

Matrix A is written explicitly below

$$\begin{bmatrix} \cos q(t-b)i_1 & \cos q(t-b)i_2 & \cdots & \cos q(t-b)i_M \\ \cdots & \cdots & \cdots & \cdots \\ \cos q(t-1)i_1 & \cos q(t-1)i_2 & \cdots & \cos q(t-1)i_M \\ e_1 & e_2 & \cdots & e_M \\ -\sin q(t-b)i_1 & -\sin q(t-b)i_2 & \cdots & -\sin q(t-b)i_M \\ \cdots & \cdots & \cdots & \cdots \\ -\sin q(t-1)i_1 & -\sin q(t-1)i_2 & \cdots & -\sin q(t-1)i_M \end{bmatrix}$$

where $e_n=(-1)^{i_n}$ for $n=1,2,\ldots,M$

and $q=2\pi/N$

Matrix B has a similar form; replace i's by n's in the above matrix. Note that the dimensions of A and B are,

A: (number of zero elements in DFT) by (number of missing points)

B: (number of zero elements in DFT) by (number of given points)

Very often, S>>M (which means that the number of given points is greater than the number of missing points), so it is more appropriate to calculate the product $y=Bx_S$ directly.

This calculation is easily performed by the formulas

$$y_{t-i}=\sum_{k=1}^{S} x_{n_k}\cos[q(t-i)n_k] \qquad \text{for } i=1,2,\ldots,t-1$$

$$y_t = \sum_{k=1}^{S} e_k x_{n_k}$$

$$y_{t+i} = -\sum_{k=1}^{S} x_{n_k} \sin[q(t-i)n_k] \qquad \text{for } i=1,2,\ldots,t-1$$

After the y vector has been calculated, any solution of the system of linear equations $Ax_M+y=0$ is an answer to the interpolation problem. The solution depends on the rank of A (2b+1 by m) and the rank of the augmented matrix (A|y). A necessary and sufficient condition for having at least one solution is rank(A|y)=rank(A) [5]. When rank(A|y) $\neq$rank(A), a least squares approximate solution is possible to be found

3. Sensitivity of the Estimates

After x has been found by solving the linear equation Ax+y=0 we would like to know how close the estimate is to the actual data. (For this section we simply write x instead of x_M.)

Let us use x' to denote the numerical solution of Ax+y=0 and x to denote the actual values of the missing data. The reasons that x' is not equal to x are

(a) The assumption (3) is not exactly true; this means that the DFT elements with indices t-b,...,t,...t+b are not exactly 0 and the equation

$$A_C x_M + B_C x_S = 0$$

should be replaced by

$$A_C x_M + B_C x_S = e_C$$

where e_C is a complex vector with small numbers corresponding to the frequencies that x was assumed to be 0.

(b) The matrix A can be numerically ill conditioned and the solution of the equation Ax+y=0 is subject to large errors.

The following theorem gives a lower and an upper bound of the estimate error x'-x. For a given matrix A we can compute a condition number [1] K(A) which indicates the ill-conditioness of A; also, the sensitivity of the solution with respect to the DFT non-zero elements is found.

Theorem

Let |.| be an L(p) norm, x' be the solution of Ax+y=0 and x be the solution of Ax+y=e. Then,

$$\frac{1}{K(A)} \frac{|e|}{|y|} \leq \frac{|x'-x|}{|x'|} \leq K(A) \frac{|e|}{|y|}$$

where $K(A)=|A||A^{-1}|$ is the condition number of A

Proof

Use Ax+y=e to find -A(x'-x)=e, and

$$|e| \leq |A||x'-x| \text{ or } |x'-x| \geq |e|/|A| \qquad \text{(i)}$$

Also, use $x'=-A^{-1}y$ to find

$$|x'| \leq |A^{-1}||y| \qquad \text{(ii)}$$

Taking the inverse of both sides of (ii) and then multiplying by (i), we derive

$$\frac{1}{K(A)}\frac{|e|}{|y|} \leq \frac{|x'-x|}{|x'|}$$

Use $-A(x'-x)=e$, to find $(x'-x)=-A^{-1}e$ and

$$|x'-x| \leq |A^{-1}||e| \qquad \text{(iii)}$$

Also, use $y=-Ax'$ to find

$$|y| \leq |A||x'| \quad \text{or} \quad |x'| \geq |y|/|A| \qquad \text{(iv)}$$

Taking the inverse of both sides of (iv) and then multiplying by (iii), we derive

$$\frac{|x'-x|}{|x'|} \leq K(A)\frac{|e|}{|y|}$$

which proves the right side inequality of the theorem.

Notes

(1) For matrices, the L(1) and L(∞) norms [1] are used because they require the least amount of computations.

(2) In order to apply the interpolation theorem, the number b (number of nulls around the mid point t) must be known. When b is not known and M (number of missing points) is much smaller than N (total number of points) an empirical way of estimating the missing points (x) can be used:

(a) Assume that b is greater than or equal to (M-1)/2; then apply the interpolation theorem to find an estimate of the missing points.

(b) Augment the vector x (of missing points) by including K known points; then, estimate x based on the assumption that b>(K+M-1)/2.

(c) If answers of steps (a) and (b) are similar and the error of the estimates of the K known points is 'small' then the bandlimiting assumption is reasonable (and b is greater than (M-1)/2); otherwise, the signal is not bandlimited and the theorem cannot be applied.

4. Iterative Interpolation Algorithms

This section describes another technique that can be used to solve the interpolation problem using bandlimiting assumptions. This method is the discrete counterpart of the algorithms given in [3, 6]. In these references it is proved that whenever a segment {x(t), a<t<b} is known and x(t) is a bandlimited signal, then an iterative algorithm involving Fourier transforms can be used to determine the values of x(t) at any point t. The conjecture 'if {x(i), n<i<k} is known and x(i) is a bandlimited sequence, then x(i) can be found at any point i' is not true. In the following we discuss this subject in more detail.

Consider the time series x=(x(0),x(1),...,x(N-1)) and assume that

$x_M=(x_{i_1},x_{i_2},\ldots,x_{i_M})$ is the subvector of 'missing' data points

$x_S=(x_{n_1},x_{n_2},\ldots,x_{n_S})$ is the remaining vector of known data points

(the notation conventions are the same as in the Interpolation Theorem). Assume also that the DFT transform of x(n) is 0 at the points

$$t-k_1,\ t-k_2,\ldots,t-k_F \quad t+k_F,\ \ldots,t+k_2,\ t+k_1$$

where t is the mid row index.

Define the diagonal matrices E and U as follows:

$E=\mathrm{diag}(e_i)$ where $e_i=0$ if $i\in\{i_1,i_2,\ldots,i_M\}$

or $e_i=1$ if $i\in\{n_1,n_2,\ldots,n_S\}$

$U=\mathrm{diag}(u_i)$ where $u_i=0$ if $i\in\{t+k_1,\ldots,t+k_F\}$

$u_i=1$ otherwise.

This definition implies that E contains a 1 at the diagonal element corresponding to a given data point and U contains a 1 at the diagonal element corresponding to a non-zero DFT point.

Using these definitions,

$$x_S=Ex \quad \text{and} \quad a=Ua \qquad (a=\mathrm{DFT}\{x\})$$

Theorem

Consider the iterative algorithm

(1) Initial step: $y^0=Ex$, set $n=0$

(2) Loop:

$b^n=Wy^n$ (find the DFT of y^n)

(3) $a^n=Ub^n$ (filter out the zero magnitude frequencies)

(4) $x^n=W^{-1}a^n$ (find the inverse DFT)

(5) $y^{n+1}=x^n+E(x-x^n)$

$=y^n+(I-E)x^n$ (update the estimates y^n at the missing points)

Then, the following are true:

(a) All vectors y^n are equal to x at least at the points

x_S; that is, $y^n_{n_i} = x_{n_i}$ for i=1,2,...,S

(b) All transforms a^n (DFT's of x^n) are 0 at the t-k_i points

(c) The sequences of x^n and a^n do not diverge.

Proof

Use (5) to find that y^n is equal to x at least at the given points.

Use (3) to find that a^n are zero at the t-k_i's frequencies

The above prove (a) and (b). The proof of (c) is not so trivial; first we find the discrete equations for x^n, and a^n.

$$x^{n+1}=W^{-1}a^{n+1}=W^{-1}Ub^{n+1}=W^{-1}UW(x^n+E(x-x^n))=W^{-1}UW\bar{E}\,x^n+W^{-1}UWEx \qquad (i)$$

similarly,

$$a^{n+1}=Ub^{n+1}=UWy^{n+1}=UW\bar{E}x^n+UWEx=UW\bar{E}W^{-1}a^n+UWEx \qquad (ii)$$

where $\bar{E}$ denotes (I-E). This shows that the stability of the discrete equations depends upon the matrices

$$A= W^{-1}UW\bar{E} \quad \text{and} \quad B= UW\bar{E}W^{-1}.$$

To prove that system (i) is bounded (not unstable) we prove that

$$\max_{|x|=1} \frac{x^T A^T A\, x}{x^T x} \leq 1 \qquad (iii)$$

The proof of this is based on the fact $z^T U z \underline{<} z^T z$

$$x^T A^T A x = x^T \bar{E} W^* U (W^{-1})^* W^{-1} U W \bar{E} x \qquad \text{(use the fact } (W^{-1})^* W^{-1} = 1/N \text{)}$$

$$= (1/N)\ (x^T \bar{E} W^*) U (W \bar{E} x) \qquad \text{(use the z-inequality)}$$

$$\underline{<} (1/N) x^T \bar{E} W^* W \bar{E} x \qquad \text{(use the fact } W^* W = N\text{)}$$

$$= x^T \bar{E} x \qquad \text{(use the z-inequality)}$$

$$\underline{<} x^T x$$

Therefore (iii) is true. This also proves that the eigenvalues of A are either inside the unit circle or on the unit circle. Similar proof can be used for the boundness of the discrete system (ii) using matrix B.

5. Simulation Results

The interpolation algorithm has been verified by a simulation program. Also, simulation experiments have been performed using various input time series with missing data. In this section we describe the programs implemented and the observed numerical results.

An outline of the simulation steps follows:

1. A time series with known bandwidth was generated; several points were marked as 'missing' points.
2. The matrix A (see the interpolation theorem assertion) was formulated according to the indices of the missing points.
3. The product $y=Bx_s$ was calculated according to the formulas given in the section 'Computation of A and B'.
4. The linear equations $Ax_s+y=0$ were solved and the results were compared with the original marked missing points.

Step 1 was implemented by using the program EASY (see Appendix on Software Support). We started with a sequence of 64 points

$$x = 0.25(1,1,1,1,0,\ldots,0)$$

then we took the fast Fourier transform (FFT) of x, substitute a 0 in place of the 28th up to 36th point, and then we took the inverse FFT transform. Figure 1 shows the resultant time series and figure 2 shows the corresponding magnitude of the FFT. Note that for all the programs we start the indices of the series from 0 (and not from 1) in order to be consistent with the interpolation theorem and the majority of the literature.

Steps 2, 3 and 4 were implemented by a program (called MISS); the source code is attached. The user can mark as many points as he wishes and consider them as 'missing' points. Then the algorithms are applied and the program prints the estimates of the marked points and the percent error. If the A matrix is not square, a least square solution is found. Also, if A is singular the program prints the dimensions and the rank of A and finds the minimum norm solution of the missing data points.

On the next page there is a sample run of the MISS program. The error of estimates of the marked (missing) points is calculated less than 0.1 for three values of the parameter b (2b+1 are the number of zeros of the DFT).

Sample Run of the Missing Data Interpolation Algorithm

```
B>MISS                                                    (run the program)

Missing data interpolation algorithm

Enter data filename: DAT2937.064                          SAMPLE 1
Enter indices of 'bad' points (terminate with -1)
0, 10, 25, -1
Enter beta (the width of the 0 magnitude DFT): 3
Inverting  7 by  3...
actual-data     estimates       error (%)
0.2438          0.2438          0.2443E-04
0.5635E-02      0.5635E-02      0.7428E-02
-.1853E-02      -.1853E-02      0.1119E-01

Enter data filename: DAT2937.064                          SAMPLE 2
Enter indices of 'bad' points (terminate with -1)
0, 10, 25, -1
Enter beta (the width of the 0 magnitude DFT): 1
Inverting  3 by  3...
actual-data     estimates       error (%)
0.2438          0.2438          0.5315E-03
0.5635E-02      0.5635E-02      0.3151E-01
-.1853E-02      -.1853E-02      0.2468E-01
Condition-1     low-1 bound     upper-1 bound (%)
7.1601          27.7744         1423.986
Condition-inf   low-inf bound   upper-inf bound (%)
7.6537          51.9731         3044.611

Enter data filename: DAT2937.064                          SAMPLE 3
Enter indices of 'bad' points (terminate with -1)
0, 10, 25, 12, 5, 30,-1
Enter beta (the width of the 0 magnitude DFT): 3
Inverting  7 by  5...
actual-data     estimates       error (%)
0.2438          0.2438          0.2443E-04
0.5635E-02      0.5635E-02      0.7428E-02
-.1853E-02      -.1853E-02      0.1119E-01
0.2521          0.2521          0.3771E-02
-.1170          -.1170E-01      0.3932E-01
```

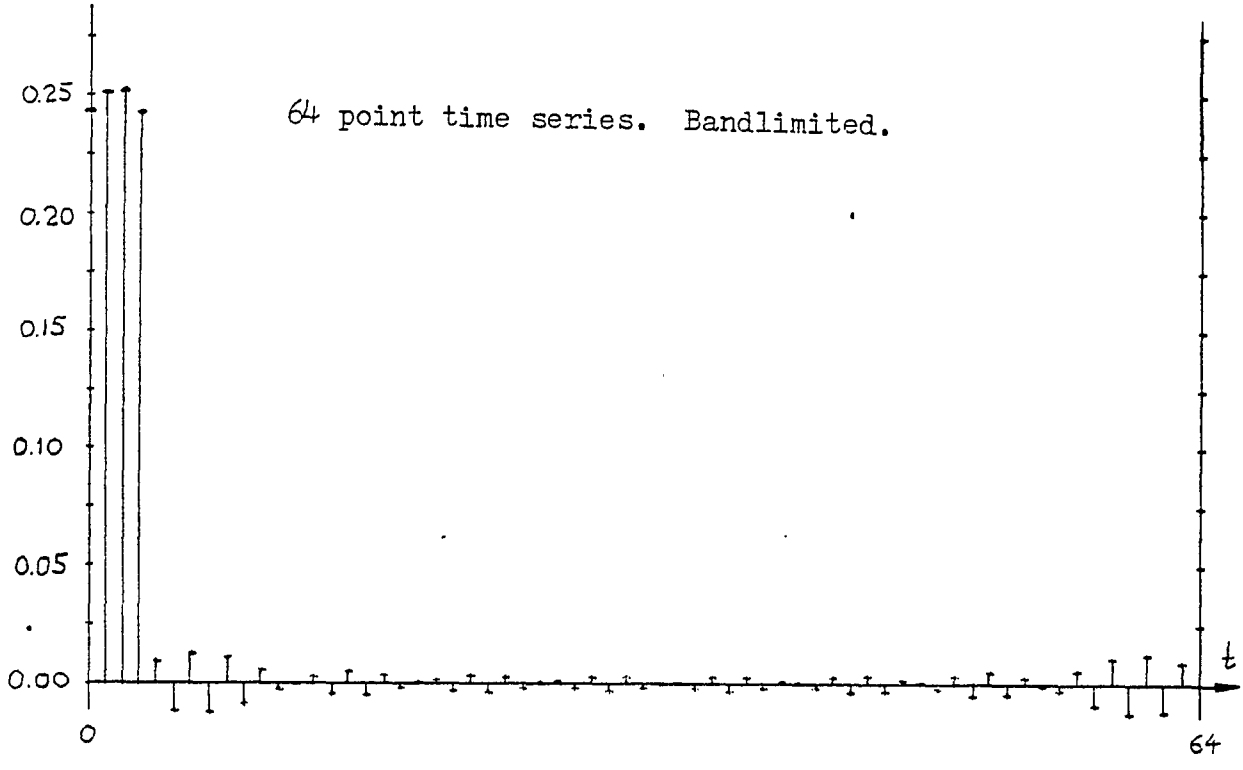


Figure 1 : Bandlimited time series

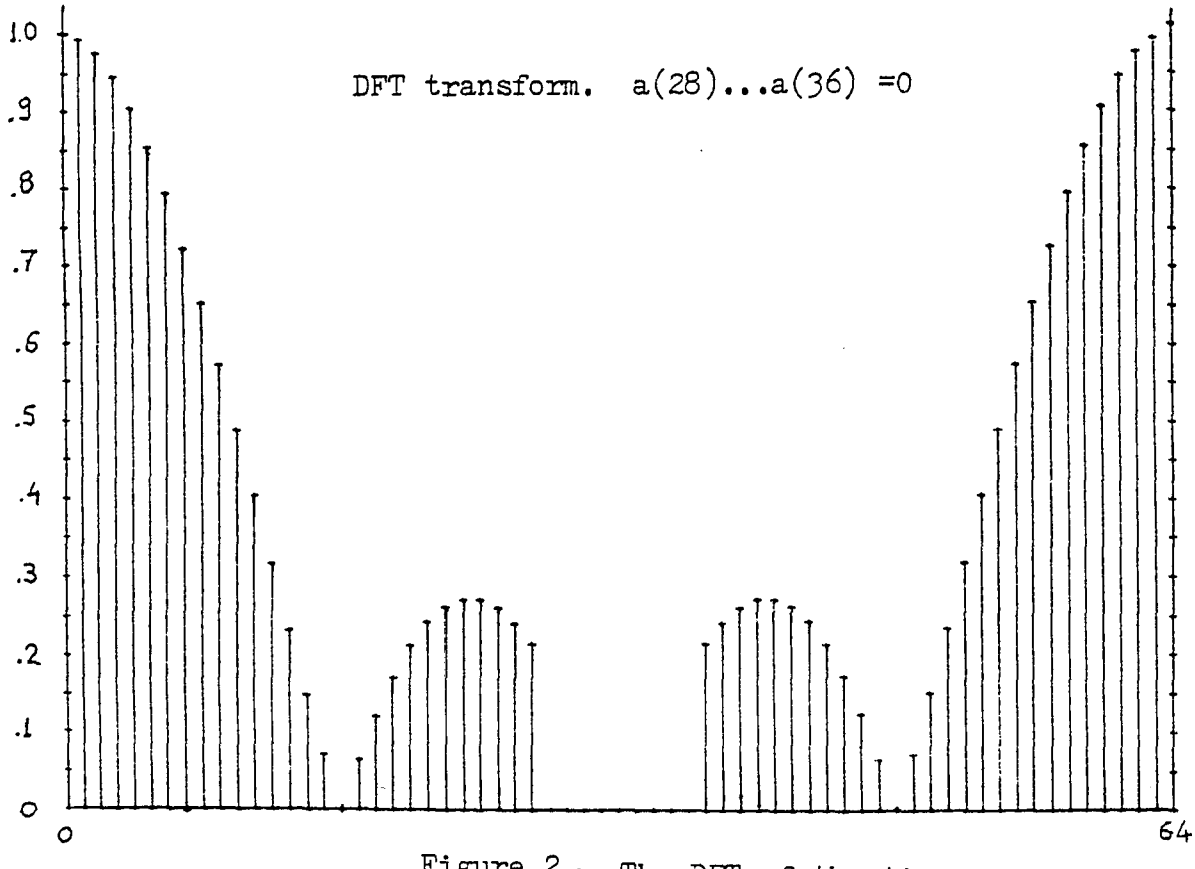


Figure 2 : The DFT of the time series

```
C                  Appendix III
C
C       ******************************************
C       * MISSING DATA INTERPOLATION ALGORITHM    *
C       *  GEORGE KONTOPIDIS, NOV 3 1980  UNH     *
C       ******************************************


C
C DATA   = Data filename containing the bandlimited time series.
C X      = Array with good and bad data of length N
C IBAD   = Indices of the bad data of length LBAD
C IGOOD  = Indices of the good data of length LGOOD
C W22    = Matrix of dimensions (2*BETA+1 BY LBAD) Left side.
C W22inv = Inverse of W22
C Y      = Vector of length 2*BETA+1.  Right side.
C X1     = Result vector with the interpolated points
C

C    Several Statements start with the word 'Check'.  By removing
C    it, additional (testing) information can be printed out.
C    Use the editor command: CCheck$Check<cr>$$ to 'remove it'
C    and use CCheck<cr>$Check$$ to 'put it' back on line.

        DIMENSION X(260),  W22(10,10), Y(260)
        DIMENSION IBAD(30),IGOOD(260), X1(260), W22inv(10,10)
        LOGICAL DATA(12)
        COMMON WORK(260)
C
        Call PRN('Missing data interpolation algorithm$')
1       Call CRLF
        IW=10
C
C...User interaction.
C
        CALL SGET('Enter data filename: $',DATA,12)
        CALL MGET(DATA,X)
Check   CALL MPRN(X,'Time series X$')
        N=NDIM(X)
C
78      CALL PRN('Enter indices of 'bad' points (terminate with -1)$')
        Call CRLF
```

```
        Read(5,200) IBAD
200     Format(30I5)
C
C...I is used as counter. Indices of bad point in IBAD
C
        I=1
5       IF( IBAD(I) ) 10,20,20
20      I=I+1
        GO TO 5
10      LBAD=I-1
C
C...Get beta; check if number of points is even. Correct on error
C
        IBETA=IGET('Enter beta (the width of the 0 magnitude DFT): $')
C
        IF((LBAD/2)*2.NE.LBAD) GO TO 30
        CALL PRN('Number of bad points must be odd for this program$')
        LBAD=LBAD-1
C
C...Find the indices of the good data
C
30      I=1
        N1=N-1
        DO 40 J=0,N1
                DO 41 JJ=1,LBAD
                IF(J.EQ.IBAD(JJ)) GO TO 40
41              CONTINUE
        IGOOD(I)=J
        I=I+1
40      CONTINUE
        LGOOD=I-1
C
C...Print the indices arrays for checking
C
Check   CALL PRN('Verify indices of the bad data$')
Check   WRITE(5,47) (IBAD(I),I=1,LBAD)
Check   CALL PRN('Verify indices of the good data$')
Check   WRITE(5,47) (IGOOD(I),I=1,LGOOD)
Check47 FORMAT(' ',26OI4)
C
C...Pre-computations for indices and limits...
C
```

```
        MID=N/2
        PHI=PI(2.)/FLOAT(N)
        NEQ=2*IBETA+1
        ICNT=IBETA+1
C
C...Formulation of the left side of the equation
C
        IF(IBETA) 53,53,54
54      I=1
        DO 50 I=1,IBETA
        DO 50 J=1,LBAD
                DEX=(MID-IBETA-1+I)*IBAD(J)
                W22(I,J)=COS( PHI*DEX )
                I2=I+ICNT
                W22(I2,J)=-SIN( PHI*DEX )
50      CONTINUE
C
C and the middle row...
C
53      I=1
        DO 60 J=1,LBAD
                IS=IBAD(J)
                IF(2*(IS/2)-IS) 61,62,61
61              W22(ICNT,J)=-1.
                GO TO 60
62              W22(ICNT,J)=1.
60      CONTINUE
C
C...Pack the result matrix and print it for checking purposes
C
        CALL PACK(W22,IW,NEQ,LBAD)
Check   CALL MPRN(W22,'This is the left side matrix$')
C
C...Now find the right side.  Do not form w23 excplicitly
C
        IF(IBETA) 93,93,94
94      I=1
        DO 90 I=1,IBETA
                SUM1=0.0
                SUM2=0.0
                DO 80 J=1,LGOOD
                        IG=IGOOD(J)
```

```
                        DEX=(MID-IBETA-1+I)*IG
                        IG3=IG+3
                        SUM1=SUM1+COS( PHI*DEX )*X(IG3)
                        SUM2=SUM2-SIN( PHI*DEX )*X(IG3)
80              CONTINUE
                Y(I)=(-1.)*SUM1
                I2=I+ICNT
                Y(I2)=(-1.)*SUM2
90      CONTINUE
C
C...and the middle row sum...
C
93      SUM1=0.0
        DO 98 J=1,LGOOD
                IG=IGOOD(J)
                IF(2*(IG/2)-IG )  97,96,97
97              SUM1=SUM1-X(IG+3)
                GO TO 98
96              SUM1=SUM1+X(IG+3)
98      CONTINUE
        Y(ICNT)=(-1.)*SUM1
C
C...Pack it and print it for checking
C
        CALL PACK(Y,IW,NEQ,1)
Check   CALL MPRN(Y,'This is the right side of the equations$')
C
C...Solve the linear equations W22 * X1 = Y
C...If W22 is not square, do least squares
C
        j1=NDIM(W22)
        j2=MDIM(W22)
        Write(5,2010) j1,j2
2010    Format(' Inverting ',I3,' by ',I3,'...')
        CALL GINV(W22inv,W22)
        CALL MUL(X1,W22inv,Y)
C
C...Print the (original data) (estimate) (% error)
C
        CALL PRN('actual-data    estimates      error (%)$')
        DO 57 I=1,LBAD
                I1=3+IBAD(I)
```

```
                I2=2+I
                ELOW=100.*ABS(  ( X(I1)-X1(I2) )/X1(I2)   )
                WRITE(5,900) X(I1),X1(I2),ELOW
57      CONTINUE
C
C...Calculate the condition number and the norms
C...Quit if the W22 matrix is not square
C
        If (j1.NE.j2) Go to 1
        AX1 = XNORM1 ( W22inv )
        AX  = XNORM  ( W22inv )
        BX1 = XNORM1 ( W22    )
        BX  = XNORM  ( W22    )
        VX1 = VNORM1 (Y(3),Y(1))
        VX  = VNORM  (Y(3),Y(1))
        AX =AX*BX
        AX1=AX1*BX1
        ELOW1=100./(AX1*VX1)
        EUP1 =100.*AX1/VX1
        ELOW =100./(AX*VX)
        EUP  =100.*AX/VX
        CALL PRN('Condition-1    low-1 bound    upper-1 bound(%)$')
        WRITE(5,900) AX1,ELOW1,EUP1
        CALL PRN('Condition-inf  low-inf bound  upper-inf bound(%)$')
        WRITE(5,900) AX,ELOW,EUP
900     FORMAT(' ',3G14.7)
        GO TO 1
        END
```

Notes and References III

[1] Ralston A., P. Rabinowitz. A First Course in Numerical Analysis. McGraw-Hill, New York 1978.

[2] Rabiner L. R., B. Gold. Theory and Application of Digital Signal Processing. Prentice Hall, Eglewood Cliffs, New Jersey 1975

[3] Papoulis A. Signal Analysis. McGraw-Hill, New York 1977.

[4] Griffiths L. J., High Resolution Spectral Estimates Obtained Using Data Extrapolation, ICASSP 80 Proceedings, Denver Colorado, 1980.

[5] Brogan William L. Modern Control Theory. Quantum Publishers, Inc. New York, 1974.

[6] Papoulis A., A New Algorithm in Spectral Analysis and Band-limited Extrapolation, IEEE Trans. Circuits Syst., vol. CAS-22, no. 9, pp. 735-742, Sept 1975.

[7] Lacoss R. T., Data Adaptive Spectral Analysis Methods, Geophysics, vol. 86, pp. 661-675, 1971.

CHAPTER IV

RANDOMLY SAMPLED SYSTEMS

Outline of Chapter IV

1. Notation and Conventions
2. Definition of Commonly Used pdf's
3. Examples of Random Sampling Processes
4. Propagation of the Mean State Values
5. Propagation of the Mean Square State Values
6. Defining the Power Spectral Density Gain

Appendix IVa

Appendix IVb

Notes and References IV

Outline of Chapter IV

This chapter discusses randomly sampled systems.

Sections 1 and 2 summarize the notation and basic definitions used in the other sections. Because characteristic functions are extensively used to simplify integral equations, we associate the probability density functions p(x) (lower case p) with the characteristic functions P(jw) (upper case P). The same convention is used for the bilateral Fourier transform pairs.

In section 3 we give several examples of random sampling processes, the corresponding probability densities and the characteristic functions.

Section 4 consists of four theorems (1 to 4) and their proofs. All of them deal with the propagation of the mean values of the states of a randomly sampled system. The theorems give methods of calculating the mean values at the sampling instants (whenever a sample is taken) and at any time instant between (absolute time independent of the sampling process).

Section 5 deals with the propagation of the mean square values of the states in a manner similar to section 4. The problem here is more difficult because of the nonlinearities involved, but usage of the direct products simplify the development.

Section 6 discusses the frequency characteristics of randomly sampled systems. We define the power spectral density gain, and we derive an expression for it involving the system parameters and the sampling characteristics.

Finally, two appendices are given with the required details for several points in the proofs of the theorems. In particular, appendix IVb consists of a collection (and proofs) of the Kronecker (direct) operations which are used extensively in sections 5 and 6. At last,

references used in the study of the subject of this chapter are listed.

The new results claimed in this chapter are:

(a) The propagation of the mean state values of a multi-input multi-output stochastic, randomly sampled system.

(b) The propagation of the mean square values of the above system.

(c) The definitions and the associated theory of the power spectral density gain of a randomly sampled system.

1. Notation and Conventions

T	(As a superscript) denotes the transpose of a matrix
T, T_n, $\{T_n\}$	Time intervals. T may denote a random variable and T_n can be interpreted as a realization of T. $\{T_n\}$ in general means the sequence of T_n's.
t, t_n, $\{t_n\}$	Time instants. The meaning of t, t_n and $\{t_n\}$ is similar to T, T_n and $\{T_n\}$. In general T refers to time differences and t refers to absolute time.
r,t,u,h	Dummy variables for time used in definitions and integrals.
a,b,c	Dummy integration variables for random variables.
X(s), Lx(t), (Lx)(s)	Bilateral Laplace transform of the function x(t).
$X^*(z)$, Zx(n), (Zx)(z)	Z transform of the sequence x(n) defined by the summation $$(Zx)(z) = \sum_{n=0}^{\infty} x(n)z^{-n}$$
$F_X(x)$	Cumulantive probability function of X evaluated at x.

$p_X(x)$	Probability density function (pdf) of X evaluated at x.
$P_X(s)$	Characteristic function of X evaluated at s. Notice that the characteristic function of X is defined as the Bilateral Laplace transform of the probability distribution of X.
Pr(A)	Probability of the event (or set of values) A.
Ex(t), Ex	Expected value of x evaluated at t.
var{x}, $V_x(t)$, V(x;t)	Variance of x, $E(x-Ex)(x-Ex)^T$
cov{x,y}	Covariance of x and y, $E(x-Ex)(y-Ey)^T$
msq{x}, $M_x(t)$, M(x;t)	Mean square value of x. That is, msq{x} = Exx^T.
cor{x,y}	Correlation of x with y. That is Exy^T.
R(x;t,s), $R_x(t;s)$	Autocorrelation $Ex(t+s)x^T(t)$
R(x;s), $R_x(s)$	Autocorrelation of a stationary process $Ex(t+s)x^T(t)$.

2. Definition of Commonly Used pdf's

a. Consider the random variable T of the sampling intervals. The pdf of T's at the sampling intervals T=r is denoted by

$$p_T(r)$$

and has the following meaning:

Pr(the sampling period is inside the interval (r,r+dr))=$p_T(r)dr$

Note that $p_T(r)=0$ for $r<0$ by definition.

b. The characteristic function of $p_T(.)$ is denoted by

$$P_T(s) \quad \text{or} \quad P_T(jw)$$

and it is related to $p_T(.)$ by the equations:

$$P_T(jw)=\int_{-\infty}^{+\infty} p_T(t)\, e^{-jwt}\, dt \qquad p_T(t)=1/2\pi \int_{-\infty}^{+\infty} P_T(jw) e^{jwt}\, dw$$

c. Consider the random variable t_k. The pdf of t_k's which is

is denoted by

$$p_t(r;k),$$

has the meaning:

Pr(the k-th sampling point has value inside the interval (r,r+dr))=

$$p_t(r;k)dr$$

d. The characteristic function of $p_t(r;k)$ is denoted as

$$P_t(s;k)$$

Because the random variable t_k equals the sum of the k independent random variables $T_0, T_1, \ldots T_{k-1}$ and by assuming that all T's have the same distribution, we have the following result:

$$P_t(s,k) = P_T^k(s).$$

3. Examples of Random Sampling Processes

a. Gaussian Sampling

Gaussian sampling is defined by requiring the sampling intervals T to be independent and identically distributed Gaussian random variables with mean T_o and standard deviation a.

It is also assumed that $a<<T_o$ in order that the distribution nearly vanish for negative values of the argument.

$$p_T(T) = (2\pi a^2)^{-1/2} \exp(-(T-T_o)^2 / 2a^2)$$

$$P_T(s) = \exp(-sT_o + s^2 a^2/2)$$

Because of the independence,

$$p_{T_1,T_2,\ldots T_n}(h_1,h_2,\ldots, h_n) = p_T(h_1)\, p_T(h_2) \ldots p_T(h_n)$$

b. Raleigh Sampling

The sampling intervals T are assumed to be independent and identically distributed Raleigh random variables. The probability density function and the characteristic function in this case are:

$$p_T(T) = (T/a^2) \exp(-T^2 / 2a^2) \qquad \text{for } T \geq 0$$

$$p_T(T) = 0 \qquad \text{for } T<0$$

$$P_T(s) = -s\, (2\pi a^2)^{1/2} \exp(a^2 s^2/2)$$

The mean value of the periods T is equal to $a(\pi/2)^{1/2}$ [1,2]

c. Quantized Sampling Intervals

The sampling intervals cannot be arbitrary numbers but can take on only discrete values. That is:

T takes on one value from the set $\{T_1, T_2, \ldots T_q\}$

and

$$Pr(\ T=T_i\ )=P_i \qquad \text{for } i=1,2,\ldots q$$

Also,

$$p_T(T)=P_1\delta(T-T_1)+\ldots+P_q\delta(T-T_q)$$

$$P_T(s)=P_1\exp(\ -T_1 s\ )+\ldots+P_q\exp(\ -T_q s\ )$$

d. Uniformly Distributed Sampling Intervals

The sampling intervals T are idependent and identically distributed RVs with uniform probability density. That is,

$$p_T(t)=\frac{1}{2a}[u(t-(T_o-a))-u(t-(T_o+a))] \qquad T_o>a$$

$$P_T(s) = \exp(-T_o s)\ \sinh(as)/as$$

e. Independent Skip Sampling

Let us consider a sequence $\{h_n\}$ of fixed time points called the

'scheduled' times. Assume also that the following probabilities are given.

Pr(of taking a sample at the scheduled time h_n)=p

Pr(of not taking a sample at time h_n)=q=1-p

Based on the above sequence, we can define a random sequence $\{t_n\}$ according to the following algorithm:

step 0: start with n=1, k=1

step 1: If a sample is taken, set $t_k=h_n$ and increment n and k

Else, increment n

step 2: Go to step 1 and repeat the loop.

We now simplify the case by considering $h_n=n\Delta$ i.e. uniform scheduled times. Then,

Pr(the m-th interval (period) T_m is equal to $k\Delta$)=

Pr(the n-th scheduled sample was taken AND

the n+1, n+2,...,n+k-1 scheduled samples were not taken)

$= pq^{k-1}$ (because of the independence)

The following figure illustrates the relation between the scheduled and sampling times

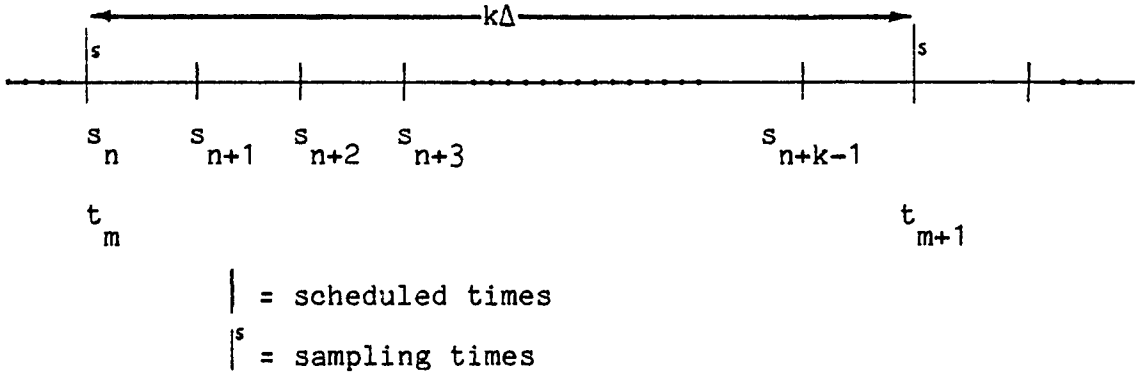


Figure 1: Independent skip sampling

The pdf and the characteristic function of T are:

$$p_T(T) = \sum_{k=1}^{\infty} p\, q^{k-1} \delta(T-k\Delta)$$

$$P_T(s) = pe^{-s\Delta} / (1-qe^{-s\Delta})$$

f. Poisson Sampling

Let us assume that m is the expected number of sampling points per unit time interval. Assume also that

$$\Pr(\text{ of taking n samples during dt }) = mdt$$

Then, integration of the above differential equation [1] gives,

$$\Pr(\text{ of taking n samples in interval of length T }) = \frac{(mT)^n e^{-mT}}{n}$$

Also,

$$p_T(T_k) = m\exp(-mT_k) \qquad \text{i.e. } \Pr(T_k<T<T_k+dh) = m\exp(-mT_k)dh$$

$$P_T(s) = m/(s+m).$$

4. Propagation of the Mean State Values

This section is devoted to the statistical description of the state values of a system. More specifically, we deal with the mean value of the states first at the sampling instances and second at any time instant. The study is logically separated into four parts (theorems 1 to 4). Theorem 1 uses a simple state model to find a way to compute the mean values of the states at any sample instant. Because of the randomness of the sample points, knowledge of the mean at the sample points does not provide much 'useful' information. We propose theorem 2 which finds the same result but at any time instant. Finally, theorems 3 and 4 use a different state model, less restrictive than the one used in theorems 1 and 2, to answer the same questions:

(a) What is the expected value of the states whenever a sample is taken?

(b) What is the expected value of the states at any time t?

Theorem 1

Consider the state equations:

$$x(t_{k+1}) = A(T_k)x_k + B_u(T_k)u_k + B_w(T_k)w_k$$

with,

$A(.)$, $B_u(.)$, $B_w(.)$ given functions

T_k random with given pdf $p_T(T;k)$

$x(t_o)$ a given vector

$\{w_k\}$ is a white random process of zero mean and independent of T_k

$\{u_k\}$ is a deterministic sequence of numbers with known Z transform.

Then

$$(1) \quad Ex_{k+1} = EA \cdot Ex_k + EB_u \cdot u_k$$

$$(2) \quad EX^*(z) = (Iz - EA)^{-1} EB_u \cdot U^*(z)$$

Proof

Taking the expectation of the state equations we find:

$$Ex_{k+1} = EA(T_k)x_k + EB_u(T_k)u_k + EB_w(T_k)w_k$$

Because $A(T_k)$ depends only on T_k and x_k depends only on $T_0, T_1, \ldots, T_{k-1}$ the independence of the T's gives:

$$Ex_{k+1} = EA \cdot Ex_k + EB_u \cdot u_k + EB_w \cdot Ew_k$$

But the {w} process has zero mean; therefore, (1) is true. The second part of the theorem can be readily derived by taking the Z transform of the first part.

Remark 1

The phrase 'taking the expectation of the state equations' requires a more precise definition. The state equation can be seen as a transformation of three random variables (namely $x(t_k)$, T_k, w_k) to a new random variable (namelly $x(t_{k+1})$).

A more rigorous derivation and a more precise notation are given in notes 1 and 2 of the appendix IVa.

Remark 2

The following property is true:

$$Z(\ Ex_k\ ) = E(\ Zx_k\ )$$

The proof is based on the linearity of the operators E and Z. It is assumed that the Z transform of the sequence and the expectation of the values x exist and they are finite. This property has to be used to prove part 2 of theorem 1.

Theorem 2

Consider the discrete state model of theorem 1 with the additional equation:

$$x(t_k + r) = A(r)x(t_k) + B_u(r)u_k + B_w(r)w_k \qquad r \geq 0$$

Then

The Laplace transform of the expected value of x (as a function of t), is given by

$$EX(s) = (A_F(s)\ [\ zI - EA\ ]^{-1}\ EB_u + B_{uF}(s)) \cdot U^*(z)$$

evaluated at $z = 1/\ P_T(s)$ and using the notation

$$A_F(s) = L\{\ A(t)\ [\ 1-F_T(t)\ ]\ \}$$

$$B_{uF}(s) = L\{\ B_u(t)[\ 1-F_T(t)\ ]\ \}$$

F_T is the cumulative distribution of T

Proof

The proof given here is based on the conditional expectation of x. In the appendix IVa (note 3) there is a detailed description of all the 'obvious' steps used below. We start from:

$$Ex(t) = E_{t=t_k+r} \; E\{ x(t_k+r) \text{ GIVEN } r \}$$

But,

$$Ex(t_k+r) = EA(r)x(t_k) + EB_u(r)u_k + EB_w(r)w(t_k)$$

$$= A(r)\, Ex_k + B_u(r)u_k$$

Substituting into the first equation we find,

$$Ex(t) = E_{t=t_k+r} \; A(r)\, Ex_k + B_u(r)u_k$$

$$= \int_{\text{all } r} \sum_{k=0} (A(r)\, Ex_k + B_u(r)u_k)\; p_t(t-r;k)\cdot(1-F_T(r))\cdot dr$$

Then,

$$EX(s) = \sum_{k=0}^{\infty} A_F(s)P_t(s;k)\cdot Ex_k + B_{uF}(s)\cdot P_t(s;k)u_k$$

By considering that,

$$P_t(s;k) = P_T^k(s)$$

and

$$\sum_{k=0}^{\infty} P_T^k(s)\cdot Ex_k = EX^*(z) \quad \text{evaluated at } z = P_T^{-1}(s),$$

we derive,

$$EX(s) = A_F(s)\cdot EX^*(z) + B_{uF}(s)\cdot U^*(z) \qquad \text{at } z=P_T^{-1}(s)$$

Finally, application of theorem 1 gives the desired result

$$EX(s) = A_F(s)\cdot(Iz - EA)^{-1}EB_u U^*(z) + B_{uF}U^*(z)$$

and the proof is complete.

Algorithm for computing EX(s)

1. Find $A_F(s)$ and $B_{uF}(s)$ (as in theorem 2)
2. Find EA and EB by using

$$EA = \int_{\text{all } r} A(r)p_T(r)\,dr$$

3. Find the matrix-functions

$$W(s) = (\, I/P_T(s) - EA \,)^{-1}$$

$$H(s) = A_F(s)W(s)\,EB_u + B_{uF}(s)$$

4. Evaluate $(Zu)(z)$ at $z = 1/P(s)$
5. Calculate

$$EX(s) = H(s)\cdot U^*(1/P_T(s))$$

Theorem 3

Consider the state model

$$x(t_{k+1}) = A(T_k)x(t_k) + B_u(T_k)u(t_k) + B_w(T_k)w_k$$

with

$A(.)$, $B_u(.)$, $B_w(.)$ given functions

T_k random with given pdf $p_T(r;k)$

x_0 a given vector

$\{w_k\}$ is a white random process of zero mean and independent of $\{T_k\}$

$\{u(t_n)\}$ is a sequence of numbers, samples of $u(t)$ at the random points $\{t_n\}$. It is assumed that $u(t)$ is obtained from the inverse Laplace transform of a given $U(s)$. Note that the sequence $\{u(t_k)\}$ is not known because t_k is a random variable.

Then

$$(1)\quad Ex_{k+1} = EA\cdot Ex_k + EB_u\cdot Eu_k$$

where,

$$Eu_k = 1/2\pi \int_{-\infty}^{+\infty} U(jw)\, P_T^k(-jw)\, dw$$

$$(2)\quad (ZEx)(z) = (zI - EA)^{-1} EB_u (ZEu)(z)$$

where,

$$(ZEu)(z) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \frac{U(jw)}{1 - z^{-1} P_T(-jw)}\, dw$$

Proof

Because the variables $A(T_k)$, $B_u(T_k)$ and $B_w(T_k)$ are independent of $x(t_k)$, $u(t_k)$ and w_k respectively, proceeding in the same manner as in theorem 1, we derive:

$$Ex_{k+1} = EA\cdot Ex_k + EB_u \cdot Eu(t_k) \qquad (i)$$

Using

$$u(t) = 1/2\pi \int_{-\infty}^{+\infty} U(jw)\ \exp(jwt)\ dw$$

and the definition

$$Eu(t_k) = \int_{\text{all } r} u(r)\ p_t(r;k)\ dr$$

we derive

$$Eu(t_k) = 1/2\pi \cdot \iint_{\text{all } r,w} U(jw)\ p_t(r;k)\ \exp(jwr)\ drdw$$

But

$$\int_{\text{all } r} p_t(r;k) \cdot \exp(jwr)\ dr = P_t(-jw;k) = P_T^k(-jw)$$

Therefore,

$$Eu(t_k) = 1/2\pi \int_{\text{all } w} U(jw)\ P_T^k(-jw)\ dw$$

This proves assertion (1). The second assertion of the theorem is proved by taking the Z transform of both sides of (i) and using:

$$Z\{E\{\ u(t_k)\ \}\} = 1/2\pi \int_{\text{all } w} U(jw)\ Z\{P_T^k(-jw)\ \} \cdot dw$$

But,

$$Za^k = 1/(\ 1 - z^{-1}a\ )$$

So,

$$(ZEu)(z) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \frac{U(jw)}{1 - z^{-1} P_T(-jw)} \cdot dw$$

Theorem 4

Consider the discrete state model of theorem 3 and the additional equation

$$x(t_k+r) = A(r)x(t_k)+B_u(r)u(t_k)+B_w(r)w_k \qquad r \geq 0$$

Then, the Laplace transform of the expected value of x at any time point t is given by

$$EX(s) = \left(A_F(s)\left(zI - EA \right)^{-1} EB_u + B_{uF} \right)\cdot(ZEu)(z)$$

where,

$$(ZEu)(z) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \frac{U(jw)}{1 - z^{-1}P_T(-jw)} \cdot dw$$

and $z = 1/P_T(jw)$

Proof

$$Ex(t) = E_{t=t_k+r}\ E\{x(t_k+r) \text{ GIVEN } r\}$$

$$Ex(t_k+r) = A(r)\cdot Ex_k + B_u(r)\cdot Eu_k$$

where Eu_k is defined in theorem 3. Then, from note 3 of appendix IVa

$$Ex(t) = \int_{\text{all } r} \sum_{k=0}^{\infty} (A(r)Ex_k + B_u(r)Eu_k)\cdot(1-F_T(r))\ p_t(t-r;k)\cdot dr$$

and by taking the Laplace transform,

$$EX(s) = \sum_{k=0}^{\infty} (A_F(s)P_T^k(s)\cdot Ex_k + B_{uF}(s)P_T^k(s)Eu_k)$$

Then,

$$EX(s) = A_F(s)\cdot EX^*(z) + B_{uF}(s)\cdot(ZEu)(z)$$

with

$z=1/P_T(s)$ and $(ZEu)(z)$ as given in theorem 3.

5. Propagation of the mean square state values

This section is a continuation of the study of the statistical properties of the system state values. More specifically, we deal with the mean square matrix of the state values. Theorem 5 tells us how the mean square matrix propagates from one sample point to the next. Theorem 6 finds the mean square matrix at any time point t. For both theorems, results of section 4 (Propagation of the mean state values) are used. Also, the lexicographic multiplication (described in the appendix IVb) is extensively used. The reader must be familiar with the notation used and the complexity involved. As an example, we explain the computation of two expessions:

$Z[M_k]$ can be calculated by:

(a) Considering the sequence of matrices M_i i=1,2,...

(b) Converting the M_i sequence to the vector $[M_i]$ sequence where the [.] means the lexicographic column.

(c) Finding the Z-transform of that sequence.

Remark

The following are true:

(a) $\langle EX \rangle$ is NOT equal to $E\langle X \rangle$

(b) $(Z\langle X \rangle)(z)$ is NOT equal to $\langle (ZX)(z) \rangle$

(c) [EX] is equal to E[X]

(d) Z[EX] is equal to E[ZX]

Theorem 5

Consider the state model:

$$x(t_{k+1}) = A(T_k)x(t_k) + B(T_k)w_k$$

$$y(t_k) = Cx(t_k) + v_k$$

with,

A(.), B (.) given matrix functions

T_k random with given pdf equal to $p_T(r,k)$

$x(t_o)$ a given vector

$\{w_k\}$ a white random process of zero mean and independent of T_k with variance Q_k.

$\{v_k\}$ a white random process of zero mean and independent of T_k with variance R_k.

Then

1. The mean square values of x at the sampling points satisfy the equation

$$[M_{k+1}] = E\langle A\rangle [M_k] + E\langle B\rangle [Q_k]$$

where:

M_k = msq{ $x(t_k)$ }=matrix of $Ex(t_k)x^T(t_k)$

[X] = lexicographic column of the matrix X

$\langle X\rangle$ = Kronecker product X * X

2. The mean square value of the output y at the sample points satisfies the equation

$$msq\{y(t_k)\} = C \cdot M_k \cdot C^T + R_k$$

3. The Z transform of the mean square of the output is given by

$$(Zmsq\{y\})(z) = C \cdot M^*(z) \cdot C^T + R^*(z)$$

where, $M^*(z)$ satisfies the equation

$$[M^*(z)] = (zI - E\langle A \rangle)^{-1} E\langle B \rangle [Q^*(z)]$$

Proof

Using the definition of the mean square of x we can easily derive

$$M_{k+1} = Ex(t_{k+1})x^T(t_{k+1})$$

$$= EA(T_k)x(t_k)x^T(t_k)A^T(T_k) + EB(T_k)w_k w_k^T B^T(T_k) +$$

$$+EA(T_k)x(t_k)w_k^T B^T(T_k) + EB(T_k)w_k x^T(t_k)A^T(T_k)$$

Because $x(t_k)$ does not depend on w_k (it depends only on the previous values of w), and because $Ew_k = 0$, the last two terms are zero. In order to separate the expectations with respect to T and x, we use the lexicographic multiplication for the remaining two terms of the equation above:

$$[M_{k+1}] = E\{ \langle A \rangle [x(t_k)x^T(t_k)] \} + E\{ \langle B \rangle [w_k w_k^T] \}$$

In this equation, the $\langle A \rangle$ and $\langle B \rangle$ depend on the random variable T_k which is independent of x and w. Therefore,

$$[M_{k+1}] = E\langle A \rangle [M_k] + E\langle B \rangle [Q_k]$$

which proves part (1). At this point it is worth noting that the

lexicographic column of the mean square values of x satisfies a first order recursive equation very similar to the state equations.
Part (2) is trivially proved by taking the expectation

$$Ey(t_k)y^T(t_k) = C \cdot Ex(t_k)x^T(t_k) \cdot C^T + Ev_k v_k^T$$

The third part is proved easily by taking the Z transform of (1),

$$Z[M_{k+1}] = E\langle A\rangle \cdot Z[M_k] + E\langle B\rangle \cdot Z[Q_k]$$

$$z \cdot [M^*(z)] = E\langle A\rangle \cdot [M^*(z)] + E\langle B\rangle \cdot [Q^*(z)]$$

from which $[M^*(z)]$ can be found.

Theorem 6

Consider the discrete state model of theorem 5 with the additional equation:

$$x(t_k + r) = A(r)x(t_k) + B(r)w_k$$

and use the notation:

$\langle X\rangle_F$ = Kronecker product of X'(t) by X'(t) with

$$X'(t) = X(t)\,(1 - F(t))^{1/2}$$

Then

The Laplace transform of the (lexicographic ordered) mean square value of x (as a function of t), is given by:

$$[LM](s) = \left[L\{\langle A\rangle_F\}(zI - E\langle A\rangle)^{-1} E\langle B\rangle + L\{\langle B\rangle_F\} \right][Q^*(z)]$$

evaluated at $z=1/P_T(s)$

Proof

We evaluate the expression:

$$msq\{x(t)\} = E_{t=t_k+r} \; E\{x(t_k+r)x^T(t_k+r) \text{ GIVEN } r\}$$

Here, the second expectation equals:

$$Ex(t_k+r)x^T(t_k+r) = A(r)M_kA^T(r) + B(r)Q_kB^T(r)$$

where $M_k=msq(x_k)$ as it was determined by theorem (5).

Now we take the expectation with respect to $t=t_k+r$

$$E_{t=t_k+r} A(r)M_kA^T(r)+B(r)Q_kB^T(r) =$$

$$= \int_0^t dr \sum_{k=0}^{\infty} (1-F_T(r)) \; p_t(t-r;k) \; \left[A(r)M_kA^T(r)+B(r)Q_kB^T(r)\right]$$

Let's call the above expectation M(t). Then, the Laplace transform of M(t) can be used to replace the convolution integral with a product of terms. That is:

$$L\{M(t)\} = (LM)(s) =$$

$$= \sum_{k=0}^{\infty} L\{p_t(t;k)\} \; L\{(1-F_T(t))\left(A(t)M_kA^T(t) +B(t)Q_kB^T(t)\right)\}$$

Using the lexicographic multiplication we have,

$$[LM(s)] = \sum_{k=0}^{\infty} L\{p_t(t;k)\} \; L\{ \langle A\rangle_F[M_k] + \langle B\rangle_F[Q_k] \}$$

The infinite summation that appears above can be computed by using

$$Lp_t(t;k) = P_T^k(s)$$

and the property of the Z transform:

$$\sum_{k=0}^{\infty} (\text{function of } k)\, P_T^k(s) = (\text{Z-transform of the function evaluated at } z=1/P_T(s))$$

Then,

$$[LM(s)] = L\{ \langle A\rangle_F [ZM_k] + \langle B\rangle_F [ZQ_k] \}$$

evaluated at $z = 1/P_T(s)$

and,

$$[LM](s) = L\{\langle A\rangle_F\}\cdot[M^*(z)] + L\{\langle B\rangle_F\}\cdot[Q^*(z)]$$

Using the result (3) of theorem 5 we can compute the $M^*(z)$ in terms of $Q^*(z)$ and complete the proof of the theorem.

6. Defining the Power Spectral Density Gain

The power spectral density gain of a randomly sampled system is defined by taking the Fourier transform of the autocorrelation function of the output of the system when the input is a white noise process of unit intensity. The development has two parts; first we present a lemma which shows the steps in defining the power spectral density of a nonrandom linear system, and second, we extend the definitions to random systems. At the end, an example is given to indicate the procedure for calculating the bandwidth of a first order randomly sampled system.

Lemma

Consider the following multiple-input multiple-output linear system,

$\dot{x} = Fx+Gw$, $\quad Ew(t)w^T(r) = Q\,\delta(t-r)$

$y = Cx$ $\quad Ew(t)=0$

with F stable.

Let M(t) be the variance of x(t); because the system is assumed to be stable and time invariant, M(t) reaches a steady state matrix, denoted by M.

Also, let A(t) be the transition matrix exp(Ft).

Then

1. The correlation function R(t) of the output is given by

$R(t) = CA(t)MC^T$ $\quad$ for t>0

$R(t) = CMC^T$ $\quad$ for t=0 $\quad$ (R(t) is continous at 0)

$R(t) = CMA^T(-t)C^T$ $\quad$ for t<0

2. The power spectral gain matrix of the system,

$H(jw)H^*(jw)$ $\quad$ (* means conjugate and transpose)

can be found by replacing Q by I and taking the Fourier transform of R(t)

Proof

Because Ex(t)=0, the variance and the autocorrelation of x(t) have the same value. The correlation of the output y(t) is calculated below for lag r>0.

$$R(r) = Ey(t+r)y^T(t) = C\,Ex(t+r)x^T(t)\,C^T =$$

$$= CA(r)\,Ex(t)x^T(t)\,C^T = CA(r)M(t)C^T$$

For $r<0$, we can write

$$R(r) = E\ y(t)y^T(t-r)$$

and following similar steps, we can find,

$$R(r) = CM(t)A^T(-r)C^T$$

which proves the first part of the lemma. The second part can be proved by taking the Fourier transform of $R(t)$. To simplify the notation we use

$F^+\{g(t)\}$ for the Fourier transform of the positive values of t

$F^-\{g(t)\}$ for the Fourier transform of the negative values of t

Then,

$$F\{R(t)\} = F^+\{R(t)\} + F^-\{R(t)\} = F^+\{CA(t)MC^T\}+F^-\{CMA^T(-t)C^T\}$$

$$= C(jwI-F)^{-1}MC^T + CM(-jwI-F^T)^{-1}C^T$$

$$= C(jwI-F)^{-1}\{ M(-jwI-F^T)+(jwI-F)M\} (-jwI-F^T)^{-1}C$$

$$= C(jwI-F)^{-1}\{ -MF^T-FM \} (-jwI-F^T)^{-1}C^T$$

To continue, we make use of the fact that $M(t)$ satisfies the first order linear matrix differential equation:

$$\dot{M}(t) = FM(t)+M(t)F^T+GQG^T$$

which, in the case of a stable system at the steady state, gives

$$0 = FM+MF^T+GQG^T.$$

Solving for the last term and replacing Q by I (as part 2 requires), we can substitute in the previous expression of $F\{R(t)\}$ to find,

$$F\{R(t)\} = C(jwI-F)^{-1}GG^T(-jwI-F^T)^{-1}C^T$$

$$= H(jw)H^{*}(jw)$$

which completes the proof of the lemma.

Power Gain of Randomly Sampled Systems

Consider the randomly sampled system:

$$x(t_{k+1})=A(T_k)x(t_k)+B(T_k)w_k$$

$$y(t_k)=Cx(t_k)+v_k$$

where T_k is random with given pdf, $var(w_k)=Q_k$ and $var(v_k)=R_k$.

Then, according to theorem 1,

$$msq\{y(t_k)\}=CM_kC+R_k$$

where the matrix M_k satisfies the first order difference equation:

$$[M_{k+1}] = E\langle A\rangle\cdot[M_k] + E\langle B\rangle\cdot[Q_k]$$

Now, let's make the following assumptions:

(a) $E\langle A\rangle$ has eigenvalues inside the unit circle

(b) Q_k and R_k are not functions of k.

Under these assumptions, the linear system which propagates $[M_k]$ is stable and for some k (large enough) it reaches a steady state. Then,

$$[M_{ss}]=E\langle A\rangle[M_{ss}]+E\langle B\rangle[Q]$$

$$[M_{ss}] = (I -E\langle A\rangle)^{-1}E\langle B\rangle [Q].$$

In this manner, the steady state variance of the states at the sampling

points can be calculated. Let's now define:

$$R(r;t_k)=Ey(t_k+r)y^T(t_k)$$

namely, R(r;.) is the correlation of the output function at lag r. Then, for all k, and a fixed r>0,

$$R(r;t_k)=CA(r)\ E\{x(t_k)x^T(t_k)\}\ C^T+R$$

and when k is large $R(r;t_k)$ does not depend on t_k, and equals:

$$R(r) = C\ A(r)\ M_{ss}C^T+R$$

or: $$[R(r)]= (\ CA(r)*C^T)\ (I-E<A>)^{-1}E<B>\ [Q] + [R]$$

which is the lexicographic form of the previous one.
In order to define the power spectral gain function we make the additional assumptions:

(c) R=0 (no observation noise)

(d) w and y are scalar functions.

(e) Q=1, i.e. the input noise is of unity intensity.

(f) The matrix A(r) is the transition matrix of a continous time system, that is, there exist a matrix F such that A(r)=exp(Fr).

Then, the correlation function (scalar) of the output is:

$$R(t) =\left(CA(t)*C^T\right)(I-E<A>)^{-1}E<B> \quad \text{for } t>0,$$

and (following similar steps we can find that):

$$R(t) =\left(C*A^T(-t)C^T\right)(I-E<A>)^{-1}E<B> \quad \text{for } t<0.$$

Notice that the only place that t (lag) appears is in the transition matrix A(t). The power spectral gain function will be found by taking

the Fourier transform of R(t):

$$S(w) = (FR)(jw) = \left(C(jw-F)^{-1}*C^{T} + C^{*}(-jw-F^{T})^{-1}C^{T}\right)(I-E\langle A\rangle)^{-1}E\langle B\rangle$$

This function describes the frequency characteristics of a randomly sampled system; notice that S(w) depends only on the expected values of <A> and <B> and not on higher order statistics.

An Example

Lets try to compute the power spectrum gain of a randomly sampled system of first order, which is the sampled version of the continuous system:

$$dx/dt = f\cdot x + g\cdot w.$$
$$y = c\cdot x$$

The randomly sampled version is:

$$x(t_k+r) = a(r)\,x(t_k)+b(r)w_k$$

where a(r)=exp(fr), b(r)=(1-a(r))g/f. Assume also that the sampling process is such that:

$E\langle a(t)\rangle$ (which is the same as $Ea^{2}(t)$) = a^{2}

$E\langle b(t)\rangle = b^{2}$

We apply first the above formula that gives S(w). The various 'pieces' are calculated below:

$$(I-E\langle A\rangle)^{-1}E\langle B\rangle = b^{2}/(1-a^{2})$$

$$C(jw-F)^{-1}*C^{T} = c^{2}/(jw-f)$$

$$C^{*}(-jw-F^{T})^{-1}C^{T} = c^{2}/(-jw-f)$$

$$C^*(-jw-F^T)^{-1}C^T = c^2/(-jw-f)$$

and combining them we derive:

$$S(w) = \frac{2f\,a^2b^2c^2}{w^2+f^2}$$

The 3-db point of this function occurs at the frequency w=-f (minus because dx/dt=fx+gw has to be a stable system, so f is a negative number). Notice that the bandwidth of the first order randomly sampled system HAPPENS to be the same as the continuous system. In general the power spectral gain function S(w) has MORE poles than the order of the original system, and the bandwidth of S(w) is different than the original continuous system.

Appendix IVa

Details in Calculations of Expectations

Note 1 (Interpretation of $Ex_{n+1}=AEx_n$); scalar case

Let $\{x_n\}$ be a (discrete time) stochastic process, and denote by $x(i;n)$ the value of the i-th realization of the process $\{x_n\}$ at time t_n. For a given time index n, the set

$$\{x(i;n)\ i=1,2,..\}$$

is considered as the image of a random variable with probability density function $p(x;n)$. That is

$$\Pr(\{i: \text{at time } t_n \text{ the value } x(i;n) \text{ belongs to } (a,a+da)\}) =$$

$$= p(a;n)da.$$

Now, let us assume that the process $\{x_n\}$ has additionally the property

$$x(i;n+1)=Ax(i;n) \quad \text{for all i and n, for A>0.} \qquad (1)$$

In words, for a given realization i of the process the values at any time instant are A times the value of the process one time instant before. Because of this additional property, $p(x;n)$ must satisfy the equation

$$p(x;n+1)= \frac{1}{A}p(x/A;n) \qquad (2)$$

This is derived by considering the transformation (1) of the random variables $x(i;n+1)$ and $x(i;n)$ [3]. It is claimed that:

$$Ex(i;n+1)= A\ Ex(i;n) \qquad (3)$$

where

$$Ex(i;n) = \int_{\text{all } a} ap(a;n)da.$$

The proof of (3) is based on (2); starting from the left side, we find the right side:

$$\int xp(x;n+1)dx = \int x/A \cdot p(x/A;n)da = \int A\cdot(x/A)\cdot p(x/A;n)\cdot d(x/A) =$$

$$= A\int ap(a;n)da$$

Note 2 (Detailed derivation of $Ex_{n+1} = EA\cdot Ex_n$)

With this note we explain in detail the meaning of the equation

$$x_{n+1} = A(T_n)\cdot x_n \qquad (1)$$

where the T_n are random. Then we prove that the expectation of x_n satisfies a similar recursive equation.

First we define the process $\{x_n\}$ and then we find the expectation Ex at a fixed point n.

Let T be a random variable with probability density function

$$p_T(h) \text{ for } h > 0$$

By performing an experiment that produces independent values of T we can generate a sequence of numbers such as

$$T_0, T_1, T_2, \ldots, T_n, \ldots$$

Because of the independence, the composite event:

$$V=\{ T_0\in(h_0,h_0+dh_0) \text{ AND } T_1\in(h_1,h_1+dh_1)\ldots\text{AND } T_n\in(h_n,h_n+dh_n)\}$$

has elementary probability

$$Pr(V)= p_T(h_0)p_T(h_1)\ldots p_T(h_n)\ dh_0 dh_1 \ldots\ dh_n$$

The sequence $\{T_n\}$ is associated with the sequence $\{x_n\}$, generated the recursive relation

$$x_{i+1}=A(T_i)x_i$$

x_0 is given

$A(.)$ is a given function.

Assume now that the numbers $T_0,\ldots,T_n$ have been generated and the resulting sequence of x's is $x_1,\ldots,x_{n+1}$. Then

$$x_{n+1}=A(T_n)A(T_{n-1})\ldots A(T_0)x_0$$

and by the fundamental theorem of expectation [1]

$$Ex_{n+1}=\left(\int_{\text{all h's}} A(h_n)\ldots A(h_0)p_T(h_n)\ldots p_T(h_0)dh_n\ldots dh_0\right)\cdot x_0$$

But the integral

$$\int_{\text{all } h_i} A(h_i)p_T(h_i)dh_i$$

has been defined as EA(T). Therefore,

$$Ex_{n+1}= EA\cdot Ex_n$$

This concludes the proof. The expectation of x_n may be interpreted

in two manners:

(a) Ex_n is the mean value of x_n when $T_0, T_1 \ldots T_{n-1}$ vary

(b) Ex_n is the mean value of x's (in ensemble sense) for given n

The first interpretation is obvious from the above note and the usage of the fundamental theorem of expectation. The second interpretation is based on the definition of the E operator. A related formalism is

If $x=f(T)$ and the pdf of T is $p_T(h)$, then

(a) $Ex = \int f(h)p_T(h)dh$ (over all possible values of h)

(b) $Ex = \int hp_x(h)dh$ (over all possible values of h)

Note 3 (Detailed derivation of Ex(t))

In this note we find the expectation of x at any time instant t. First we define rigorously the meaning of x(t). In Note 2, the sequence $\{T_n\}$ was defined; here, we associate the partial sums

$$t_n = \sum_{i=0}^{n-1} T_i \qquad n=1,2,\ldots$$

with $\{T_n\}$

The above definition creates a new sequence, namely

$$t_1, t_2, \ldots, t_n \ldots$$

For completeness, we define $t_0 = 0$ as the 'time origin'.

Now the following assumptions are made:

(a) t is a (fixed) positive number

(b) t_k is the largest t_n which is not less than t with $0 \leq n \leq k$

(c) h is the difference $t-t_k$ (greater than or equal to 0)

(d) $x_k(t)$ is given by

$$x_k(t) = A(h)x_k$$

where A(.) is a given deterministic function.

The above assumptions are illustrated by the following figure

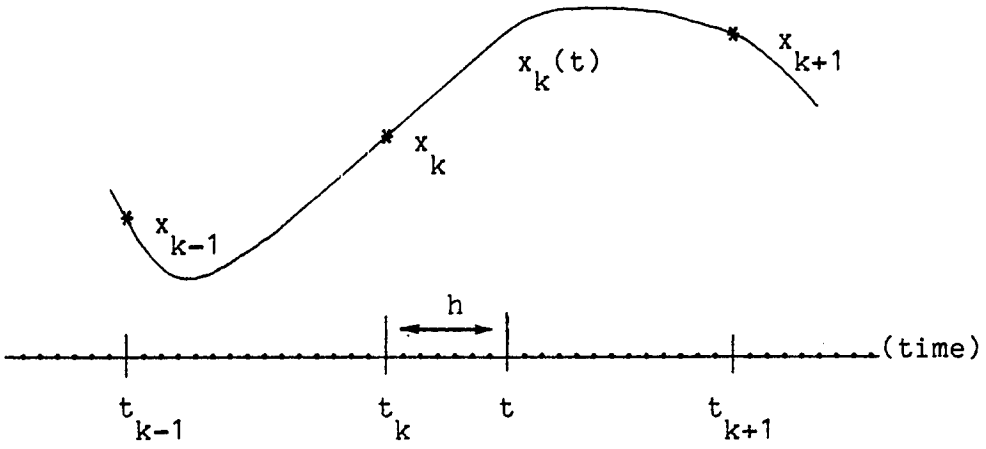


Figure 2: Derivation of Ex(t)

Notice that assumption (a) declares t to be a fixed number, assumption (b) declares t_k as a random variable and assumption (c) means h is a random variable. In the following, we will calculate the expectation of $x_k(t)$. By definition,

$$E\{x_k(t)\} = \int_b b \, dPr(\text{ the number } x_k(t) \text{ lies in the interval } (b,b+db)$$

$$= \int_b b \, dPr(\ A(h)x_k \text{ belongs to } (b,b+db)\ )$$

over all possible values of $x_k(t)$.

Because k is not fixed, the following relation is true:

$$Pr(\ A(h)x_k \text{ is in I}) = Pr(A(h)x_0 \text{ is in I}) + Pr(A(h)x_1 \text{ is in I}) + \ldots$$

for any interval I since the various events are mutually exclusive. Then the previous equation yields

$$E\{x_k(t)\} = \int_b \sum_{n=0}^{\infty} b\,dPr(\ A(h)x_n \text{ is in } (b,b+db) \text{ with } t=t_n+h \text{ AND}$$

$$t_n \text{ is the maximum } t_i \text{ which is less than } t)$$

Next, we apply Bayes rule for the conditional expectations, namely,

$$Pr(A) = \text{sum over all } h \text{ of } Pr(A|h)Pr(h)$$

and find

$$E\{x_k(t)\} = \int \Big(\sum_{n=0}^{\infty} \cdot\ b \int dPr(\ A(h)x_n \text{ is in } (b,b+db) \text{ GIVEN } h=a\ ) \cdot$$

$$\text{over } b \quad \text{over } a$$

$$\cdot\, dPr(\ h \text{ is a number in the interval } (a,a+da) \text{ and } t=t_n+h) \Big)$$

Because b appears only in the first probability, the following integration is necessary

$$\int b\,dPr(\ A(h)x_n \text{ is in } (b,b+db) \text{ GIVEN that } h=a) \ =$$

over all b

$$\int b\,dPr(\ x_n \text{ is in } (A^{-1}(a)b, A^{-1}(a)b + d(\ A^{-1}(a)b)\ ) \ =$$

over all b

$$A(a) \int r\ dPr(\ x_n \text{ is in } (r,r+dr)\ ) = A(a)Ex_n$$

over all r

Now we may substitute this result into the original expectation to find

$$E\{x_k(t)\} = \sum_{n=0}^{\infty} \int A(a) \cdot Ex_n \cdot dPr(h \text{ is in } (a,a+da) \text{ and } t=t_n+h \text{ })$$

over all a

The probability that appears above is easily calculated by

$$dPr(\text{ h belongs to } (a,a+da) \text{ and } t=t_n+h \text{ }) =$$

$$Pr(\text{ having no other } t_i \text{ in the interval } (t_n,t \text{ }) \text{ }) \text{ times}$$

$$dPr(\text{ } t_n \text{ belongs to the interval } (t-a-da,\ t-a) \text{ })$$

But,

$$Pr(\text{ having no other } t_i \text{ in the interval } (t_n,\ t) \text{ }) =$$

$$Pr(\text{ no T has value less than } a=t-t_n \text{ }) =$$

$$1 - Pr(\text{ all T have values less than a }) =$$

$$1 - F_T(a)$$

And,

$$Pr(\text{ } t_n \text{ belongs to } (t-a-da,\ t-a) \text{ }) = p_t(t-a;n)da$$

Substituting the last two expessions into the expectation of x, we find

$$Ex_k(t) = \int \sum_{n=0}^{\infty} A(a)\ Ex_n \cdot (1-F_T(a)) \cdot p_t(t-a;n)\ da$$

over all a

The calculation of the integral can be simplified by using the convolution theorem for the functions

$$A_F(t) = A(t)(\ 1-F_T(t)\ )$$

$$p_t(t;n)$$

Then,

$$L\{ \int_{\text{over all } a} A_F(a)\ p_t(t-a;n)\ da\ \} = (LA_F)(s)\ P_t(s;n)$$

where $P_t(s;n)$ is the Laplace transform of $p_t(t;n)$. Define,

$$EX(s) = L\{ E\{ x_n(t) \} \}$$

then,

$$EX(s) = \sum_{n=0}^{\infty} (LA_F)(s)\ P_t(s;n)\ Ex_n$$

Calculation of the infinite summation is accomplished by using the fact

$$P_t(s;n) = P_T^n(s)$$

and the definition of the Z transform of Ex_n,

$$EX^*(z) = Z\{ E\{ x_n \} \} = \sum_{n=0}^{\infty} Ex_n z^{-n}$$

Then,

$$EX(s) = LA_F(s) \cdot EX^*(z) \quad \text{evaluated at } z=1/P_T(s)$$

Appendix IVb

On Kronecker Operations [4]

The Kronecker product (or direct product or lexicographic multiplication) is defined in proposition 1. The defining equation

$$S = AQB^T \quad \Longleftrightarrow \quad [S] = (A*B)\ [Q]$$

is of fundamental importance because it converts matrix equations to

simple linear vector equations. The symbols [Q], A*B, and <A> are also defined in proposition 1. The reduced lexicographic multiplication is discussed in proposition 2 and then several properties of the Kronecker products are given. Following that the Kronecker sum A#B (or direct sum) is defined and the property

$$\exp(A\#B) = \exp(A) * \exp(B)$$

is pointed out. The eigenvalues of the direct products and sums, which play an significant role in determining the stability of variance equations, are calculated. Finally, two applications illustrate the usefulness of the direct operations.

Proposition 1 (Lexicographic Multiplication)

Let A, Q, B be three matrices of dimensions N by M, M by L and P by L respectively, and S (N by P) be the product

$$S = A\ Q\ B^{T}.$$

We also introduce the notation:

a) [Q] to mean a column of ML elements q_{ij} in the order

$$q_{11}, q_{12}, \cdots q_{21}, q_{22}, \cdots, q_{ML}.$$

That is, q_{ij} is before q_{kl} if the number (i-1)L+j is less than (k-1)L+l. Notice that this happens if i is less than or equal to k and j is less than l.

b) [Q]' to mean [Q^{T}] (vector of ML elements taken column by column).

c) A*B to mean a matrix of dimensions NP by ML consisting of blocks

$$a_{ij}B \qquad i=1,2,\ldots,N \quad j=1,2,\ldots,M.$$

(A*B is called the Kronecker product or the direct product of A

and B.)

d) <A> to mean A*A (Notice that A need not be a square matrix).

Then:

1. [S] = (A*B) [Q]
2. [S]'= (B*A) [Q]'
3. If $S=AQA^T$ then [S]=<A>[Q].

Proof:

To prove (1) we use the Einstein summation notation for the product

$$S = A\ Q\ B^T,$$

$$s_{ij}=a_{im}(\ q_{ml}b_{jl})$$

where l and m are dummy summation indices. Then,

$$s_{ij}=a_{im}[b_{j1}\ b_{j2}\ \dots\ b_{jL}]\begin{bmatrix} q_{m1} \\ q_{m2} \\ \dots \\ q_{mL} \end{bmatrix}$$

$$s_{ij}=a_{i1}[b_{j1}\dots b_{jL}]\begin{bmatrix} q_{11} \\ \dots \\ q_{1L} \end{bmatrix} + a_{i2}[b_{j1}\dots b_{jL}]\begin{bmatrix} q_{21} \\ \dots \\ q_{2L} \end{bmatrix} + \dots$$

$$s_{ij}=[\ a_{i1}[b_{j1}\ldots b_{jL}]\ \ a_{i2}[b_{j1}\ldots b_{jL}]\ \ldots\ ]\begin{bmatrix} q_{11} \\ \ldots \\ q_{1L} \\ q_{21} \\ \ldots \\ q_{nL} \\ \ldots \end{bmatrix}$$

So,

$$s_{11}=\left[a_{11}(b_{11}\ldots b_{1L})\,\middle|\,a_{12}(b_{11}\ldots b_{1L})\ \ldots\,\middle|\,a_{1M}(b_{11}\ldots b_{1L})\right]\cdot[Q]$$

$$s_{12}=\left[a_{11}(b_{21}\ldots b_{2L})\,\middle|\,a_{12}(b_{21}\ldots b_{2L})\ \ldots\,\middle|\,a_{1M}(b_{21}\ldots b_{2L})\right]\cdot[Q]$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$s_{1P}=\left[a_{11}(b_{P1}\ldots b_{PL})\,\middle|\,a_{12}(b_{P1}\ldots b_{PL})\ \ldots\,\middle|\,a_{1M}(b_{P1}\ldots b_{PL})\right]\cdot[Q]$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

Therefore,

[S] = matrix of blocks $a_{ij}B$ times [Q].

This also means that the ij block of the matrix A*B is the block

$$a_{ij}B$$

which proves part (1). Part (2) is proved using part (1) as follows:

$S=AQB^T$ then, $S^T=B\cdot Q^T A^T$ and $[S^T]=(B*A)[Q^T]$.

Using the definition $[S]'=[S^T]$, the second assertion is proved. Part (3) is a trivial case of part (1) using the notation (d)

Proposition 2 (The Reduced Lexicographic Multiplication)

Let Q be a symmetric M by M matrix, A be an N by M matrix, and

$$S = A \cdot Q \cdot A^{T}$$

Let us also use the notation:

a) $[Q]_r$ to mean a column of M(M+1)/2 elements q_{ij} in the order

$$q_{11}\ q_{12} \cdots q_{1M}\ q_{22} \cdots q_{2M} \cdots q_{MM}$$

That is, $[Q]_r$ can be generated from [Q] by deleting the elements with i>j.

b) $\langle A \rangle_r$ to mean the matrix <A> with the following modificatios:

- Associate the rows of <A> with the elements of the [S] vector. The indices of the [S] vector are 11, 12,..., 21, 22,...,ij,..; then delete the rows of <A> corresponding to indices ij with i>j.)
- Associate the columns of <A> with the elements of the [Q] vector. Then, replace the mn columns of <A> with m<n by the sum of the mn and nm columns; then delete the nm columns.

Then:

$$[S]_r = \langle A \rangle_r [Q]_r$$

Proof:

The proof is based on the previous result [S]=<A>[Q]. Because Q is symmetric, S is also symmetric. Therefore,

$$s_{ij} = s_{ji}$$

This means that the ij-th and the ji-th equation of the system [S]=<A>[Q], are the same. So we can ignore one of them, say the one that i>j. This explains why we can delete the ij (with i>j) elements of [S] and the ij (with i>j) rows of <A>. Also, because Q is symmetric

$$q_{ij}=q_{ji}.$$

Following the rule that q_{mn} should be deleted if m>n, the equation below shows how the matrix <A> should be transformed so that the linear system equations [S]=<A>[Q] will be true.

(nm column of <A>)q_{nm}+(mn column of <A>)q_{mn}+(additional clmns)

=(nm column of <A> PLUS mn column of <A>)q_{mn}+(addit. columns)

This proves the proposition. The importance of the reduced lexicographic multiplication is based on the reduction of the number of the multiplications and additions. Many equations (in theorems 5,6, etc) are written in terms of the lexicographic multiplication; some of them satisfy the assumptions of the preceeding proposition, therefore the reduced lexicographic multiplication can be used instead. The reader should substitute the reduced form, when it is possible (despite the fact we do not mention it there).

Properties of the Kronecker Product [4]

The following are true

1. $(A+B)*C = A*C+B*C$ (distributive law)
2. $A*(B*C) = (A*B)*C$ (associative law)
3. $(A*B)(C*D) = (AC)*(BD)$
4. $(A_1*B_1)(A_2*B_2)\ldots(A_n*B_n) = (A_1B_1)*(A_2B_2)*\ldots*(A_nB_n)$
5. $I_n*I_m = I_{n+m}$
6. $(A*B)^{-1} = (A^{-1})*(B^{-1})$
7. $tr(A*B) = tr(A)\ tr(B)$
8. $|A*B| = |A|^n\ |B|^m$

The proofs of most of the above are based on the definition of the

direct product. Justification of (3), (4) and (6) follows.

Consider P and Q satisfying the equation

$$P=(AC)Q(BD)^T$$

then,

$$[P]= (AC)*(BD)\cdot[Q] \qquad \text{(i)}$$

But P can also be written as

$$P=A(CQD^T)B^T$$

$$P=ARB^T \text{ with } R=CQD^T$$

Writing P and R in lexicographic order, the above yields

$$[P]=(A*B)[R] \quad \text{with } [R]=(C*D)[Q]$$

or

$$[P]=(A*B)(C*D)[Q] \qquad \text{(ii)}$$

Because P and Q were arbitrary, (i) and (ii) imply that

$(AC)*(BD)=(A*B)(C*D)$.

Property (4) is proved by using finite induction of (3).

To prove (6) we verify that the product

$$(A*B)\ (A^{-1}*B^{-1})$$

is equal to I by applying properties (3) and (5).

$$(A*B)(\ A^{-1}*B^{-1}) = (A\cdot A^{-1})*(B\ B^{-1})=I_n*I_m=I_{n+m}$$

Kronecker Sums

Let A and B be two square matrices of dimensions n and m respectively.

The sum

$$A*I_n+I_m*B$$

is called the Kronecker sum and it is denoted by the symbol A#B. The following are true:

1. If X(t) and Y(t) are matrix functions of t satisfying the differential equations

$$dX/dt = AX$$

$$dY/dt = BY$$

then: $d(X*Y)/dt = (A\#B)\,(X*Y)$

2. $\exp(A\#B)=\exp(A)*\exp(B)$

The proof of 1. follows

$D(X*Y)=DX*Y+X*DY=AX*Y+X*BY=AX*IY+IX*BY=(A*I)(X*Y)+(I*B)(X*Y)=(A\#B)(X*Y)$

where D=d/dt. Assertion 2. is proved by expanding both sides in Taylor series and by writing A#B explicitly [4].

Eigenvalues of Kronecker Products and Sums

Let A and B be two square matrices of dimension n and m respectively, and assume (for simplicity) that both A and B have simple eigenvalues $a_1,\ldots,a_n$ and $b_1,\ldots,b_m$ respectively. Then

1. A*B has as eigenvalues all possible products $a_i b_j$
2. A#B has as eigenvalues all possible sums a_i+b_j

Proof of 1:

Let u_i and v_j be the eigenvectors corresponding to a_i and b_j.

Then,

$$Au_i = a_i u_i$$

$$Bv_j = b_j v_j$$

so,

$$Au_i v_j^T B^T = a_i b_j u_i v_j^T$$

or

$$(A*B)[u_i v_j^T] = a_i b_j [u_i v_j^T]$$

which verifies that A*B has eigenvalues all possible products $a_i b_j$

Proof of 2:

$$Au_i v_j^T = a_i u_i v_j^T \qquad \text{for all } v_j$$

$$u_i v_j^T B^T = b_i u_i v_j^T \qquad \text{for all } u_i$$

Then,

$$Au_i v_j^T I + Iu_i v_j^T B^T = (a_i + b_j)\, u_i v_j^T$$

$$(A\#B)\,[u_i v_j^T] = (a_i + b_j)[u_i v_j^T]$$

which proves that A#B has eigenvalues all the possible sums $a_i + b_j$.

Applications

1. Solution of the Liapunov equation

$$Q = AQA^T + P$$

Readily,

$$[Q]_r = \langle A \rangle_r [Q]_r + [P]_r$$

therefore,

$$[Q]_r = (I - \langle A \rangle_r)^{-1} [P]_r$$

In order that the inverse exist, the eigenvalues of A must satisfy the property

$a_i a_j$ not equal to 1 for all i,j

Corrolary: If A is a strictly stable matrix (with eigenvalues inside the unit circle), the recursive equation

$$Q_{n+1}=AQ_nA^T+P$$

converges for $n\rightarrow\infty$.

2. Solution of the reduced Riccati equation

$$dQ/dt=FQ+QF^T+U(t)$$

Then,

$$dQ/dt=FQI+IQF^T+U(t)$$

Therefore,

$$d[Q]/dt=(F\#F)[Q]+[U]$$

The transition matrix of the above first order equation is

$$\exp(F\#F)t = \exp(Ft)*\exp(Ft)$$

Conditions for stability can be found by examining the sums of all the possible combinations of the eigenvalues of F.

Example 1

This example is an application of proposition 1 (assertions 1 and 3). It illustrates how to form the Kronecker product of two 3x3 matrices and how to perform the lexicographic multiplication.

$$\text{Let: } A=\begin{pmatrix}0&7&6\\5&1&0\\0&8&3\end{pmatrix} \quad B=\begin{pmatrix}1&6&0\\1&0&4\\5&2&3\end{pmatrix} \quad Q=\begin{pmatrix}3&4&1\\2&1&0\\0&4&5\end{pmatrix}$$

The matrix S is defined by the product AQA^T. This product is

$$Q=\begin{pmatrix}200&134&222\\143&37&142\\136&76&165\end{pmatrix}$$

Following the rules of making the [S], A*B and [Q], we find

$$[S]=\begin{pmatrix}200\\134\\222\\143\\37\\142\\136\\76\\165\end{pmatrix} \quad A*B=\left(\begin{array}{ccc:ccc:ccc}0&0&0&7&42&0&6&36&0\\0&0&0&7&0&28&6&0&24\\0&0&0&35&14&21&30&12&18\\\hdashline 5&30&0&1&6&0&0&0&0\\5&0&20&1&0&4&0&0&0\\25&10&15&5&2&3&0&0&0\\\hdashline 0&0&0&8&48&0&3&18&0\\0&0&0&8&0&32&3&0&12\\0&0&0&40&16&24&15&6&9\end{array}\right) \quad [Q]=\begin{pmatrix}3\\4\\1\\2\\1\\0\\0\\4\\5\end{pmatrix}$$

Using the EASY interactive matrix operations program, we can verify that the equality [S]=(A*B)[Q] is true.

Example 2

This example shows to things: (a) how to form the lexicographic product of rectangular matrices and (b) how to form the reduced lexicographic product. It follows the theory presented in proposition 2 and all the notation used here is consistent with the theoretical proof.

$$\text{Let: } A = \begin{pmatrix} 1 & 2 & 5 \\ 3 & 4 & 5 \end{pmatrix} \text{ (N=2, M=3)} \qquad \text{and} \qquad Q = \begin{pmatrix} 3 & 0 & 7 \\ 0 & 5 & 1 \\ 7 & 1 & 8 \end{pmatrix} \text{ (3x3 symmetric)}$$

The product $S=AQA^T$ can be found in a straight forward manner:

$$S = \begin{pmatrix} 313 & 468 \\ 468 & 695 \end{pmatrix}$$

The lexicographic columns [S] AND [Q] and the product $A*A^T$ can also be found

$$[S] = \begin{pmatrix} 313 \\ 468 \\ 468 \\ 695 \end{pmatrix} \qquad A*A = \left(\begin{array}{ccc|ccc|ccc} 1 & 2 & 5 & 2 & 4 & 10 & 5 & 10 & 25 \\ 3 & 4 & 6 & 6 & 8 & 12 & 15 & 20 & 30 \\ \hline 3 & 6 & 15 & 4 & 8 & 20 & 6 & 12 & 30 \\ 9 & 12 & 18 & 12 & 16 & 24 & 18 & 24 & 36 \end{array}\right) \qquad \begin{array}{ll} s(1,1) & \\ s(1,2) & \\ s(2,1) & \text{deleted} \\ s(2,2) & \end{array}$$

$$[Q]^T = \left(\begin{array}{ccc|ccc|ccc} 3 & 0 & 7 & 0 & 5 & 1 & 7 & 1 & 8 \end{array}\right)$$

$$q_{11}q_{12}q_{13}\ q_{21}q_{22}q_{23}\ q_{31}q_{32}q_{33}$$

(the s(i,j) and q(i,j) are the elements of S and Q respectively). By using EASY again, we can verify that [S]=(A*A)[Q]. By inspecting the A*A matrix we can see some redundancy in the multiplications because of symmetry. This point is exactly what the reduced lexicographic multiplication is about. To form the reduced [S] and [Q] we delete the s(i,j) and q(i,j) elements with i>j. The result is:

$$[Q]_r^T = \begin{pmatrix} 3 & 0 & 7 & 5 & 1 & 8 \\ q_{11} & q_{12} & q_{13} & q_{22} & q_{23} & q_{33} \end{pmatrix}$$

and $$[S]_r = \begin{pmatrix} 313 \\ 468 \\ 695 \end{pmatrix}$$

Since we deleted the s(2,1) element of [S], (3rd row) we also delete the 3rd row of A*A. Since we deleted the q(2,1), q(3,1) and q(3,2) (3rd, 6-th and 7-th column) elements of [Q], we must also delete the 3-rd, 6-th and 7-th columns of A*A, after we add them to the (1,2), (1,3) and (2,3) columns (that is, the 2-nd, 3-rd and 6-th). The result is:

$$A^*A_r = \begin{pmatrix} 1 & 4 & 10 & 4 & 20 & 25 \\ 3 & 10 & 21 & 8 & 32 & 30 \\ 9 & 24 & 36 & 16 & 48 & 36 \end{pmatrix} \quad \begin{matrix} s(1,1) \\ s(1,2) \\ s(2,2) \end{matrix}$$

The result $[S]_r = (A^*A)_r [Q]_r$ can also be verified. Note that the 4x9 A*A matrix has now been reduced to 3x6.

Notes and References IV

[1] Papoulis Athanasios, Probability, Random Variables and Stochastic Processes. New York: McGraw Hill, 1965.

[2] Hwei P. Hsu, Fourier Analysis. New York: Simon and Schuster, 1978.

[3] Wozencraft J.M., Irwin M. Jacobs, Principles of Communication Engineering. New York: John Wiley and Sons, Inc., 1965.

[4] Kalman Ruldolf, Analysis and Synthesis of Randomly Sampled Systems. Ph.D. Dissertation, Columbia University, 1957.

CONCLUSION

This dissertation has studied irregularly sampled systems. Both nonuniformly sampled and randomly sampled systems have been analyzed in time and frequency domain. Particular emphasis has been given to the applicability of the theories in signal processing areas and modern control. We have presented the results following a definition-theorem-proof approach which is a precise and ellegant way of expressing the theory. On the other hand, the various examples and simulations have been written in a simplistic manner in order to show how the theory can be applied to practical problems. Processing nonuniformly sampled data is indeed possible and is not as hopeless as at first it might seem.

In the following, we review the material presented in this work and we point out the main contributions to the area of irregular sampling. Then we include an indication of where future work needs to be done to extend the range of applicability of the results obtained.

Summary

In chapter I we have discussed a class of irregular sampling sequences, namely periodic nonuniform sequences. Periodic nonuniform sampling is interesting because we can modify the frequency domain characteristics of the sampled signal by changing a few sampling parameters. The importance of this work is two fold; first it unifies

a class of irregular sampling under a unique theoretical framework, and second, it provides the analytic and synthetic tools for finding the sampling laws that achieve certain design criteria.

Chapter II has extended the current available ARMA techniques to nonuniformly sampled systems. Both coefficient estimation and missing data interpolation are new results in this area.

Chapter III has presented three new points: An interpolation theorem, sensitivity of the estimates and an iterative interpolation algorithm. The above results are used to fill in (interpolate) missing data of a bandlimited sampled signal. Because it is not necessary to use a particular model for the time series, the techniques discribed can be applied to a wide class of (bandlimited) sequences.

Chapter IV contains several new topics and a variety of extensions to Kalman's work. The propagation of the mean-values and the mean-square values of the outputs of a randomly sampled system are the most important. The usage of direct multiplication to simplify matrix equations is not a new technique but the extensive usage in this particular area has been of great importance. The idea of using the Z-transform in the expressions of the mean and variance simplifies the results significally. Finally, the topic on power gain of randomly sampled systems is a significant result in the area of random sampling because it has been derived from a completely different route than the existing results (by Leneman and Masry) in the literature.

We also consider the sofware support of the dissertation a contribution to the system simulation and signal processing areas. The

fact that all the subroutines have been optimized from the memory requirement viewpoint and they are running under a microcomputer environment is the uniqueness of this work. Using the suggested EASYPACK data type to represent array structures allows the user to concentrate on the problem itself without worrying about the format requirements of a specific language.

Trends and Future Extensions

Irregular sampling is a new area of research and there is much work to be done. The topics with the most iteresting applications are periodic nonuniform sampling and random sampling. Missing data problems are also of great interest and have not been explored deeply enough.

Periodic nonuniform sampling can be used in many practical situations to replace digital filters and achieve better frequency domain characteristics. For that purpose, extensive simulations are needed to verify that the required properties can be obtained. Futhermore, stochastic properties of periodically sampled systems are required to be found; the case of a random process sampled periodically (using nonuniform patterns) has not been studied in this dissertation but is of interest.

Problems associated with missing (or bad data) appear very often in data acquisition systems, and it would be worthwhile to provide more theoretical emphasis to their solution. In this dissertation we give two approches to this class of problems (chapter II and III), but we realize that futher refinements of the algorithms are necessary.

Finally, randomly sampled systems can be practically used only when the synthesis problems (estimation and parameter determination from external characteristics) are solved. Our work was limited to the analysis problem which is only the first step toward the synthesis problems.

APPENDIX

SOFTWARE SUPPORT

Outline of the Appendix

1. EASYPACK Documentation
2. EASYPACK Source Code
3. EASY: Matrix Reverse Polish Calculator
4. EASYPACK Command Summary
5. Matrix Reverse Polish Calculator: Command Summary

Outline of the Appendix

This appendix is devoted in describing the tools used for the various simulations in this dissertation. Several main programs (described in chapter II and III) use the programs described here to perform most of the simulation objectives. Each main program plus the routines of this appendix can run in almost any hardware environment (including machines of 8 to 36 bits); the language used is FORTRAN-IV with some minor I/O extentions.

A common requirement for all the simulations needed is the ability to perform vector oriented operations; this is the main reason of creating the EASYPACK subroutines. EASYPACK is a collection of subroutines in a form of a library file, so that the user can call a specific routine to perform a desired function.

A second program called EASY has been created to supplement the user's tools; EASY is an interactive main program organized in Reverse Polish Notation. It was primarily written to exercise and test the EASYPACK features but it has been extended to include signal processing algorithms commonly needed for simulations. Using EASY, one can solve a (rather large) class of signal processing problems interactively with not much programminng experience. We should also mention the fact that both programs (EASYPACK and EASY) have been developed and operated in a micro computer environment which shows the compactness of the code and the minimal requirements.

The outline of the appendix follows. First we provide the documentation of the EASYPACK routines; there, some examples of using EASYPACK are given. Then we include the source code of the subroutines. In sequence we provide the source code of the EASY main program. To use EASY it is sufficient to know how to run a Hewlett Packard calculator and to remember the EASYPACK features. Finally, a summary of all the EASYPACK and EASY 'commands' is provided for futher reference.

1. EASYPACK Documentation

Introduction

EASYPACK is a set of FORTRAN-IV subroutines and functions to perform matrix and vector operations. The package has been designed with the following specifications:

(1) Each operation is performed by a routine with the minimum number of arguments. This helps the readability of the main program and reduces the number of possible errors.
(2) The subroutines check the dimensions of the arguments; if an error is detected it is reported properly.
(3) The subroutines are optimized in terms of memory requirements. They have been designed for micro- and mini- computer usage. Machines of 8, 16 and 32 bits can easily run the code. The execution speed is sacrificed only for checking purposes.
(4) EASYPACK arrays can be easily passed to existing FORTRAN programs and FORTRAN arrays can be easily converted to EASYPACK format in place (without wasting extra memory).

A new data structure for matrices (EASYPACK format) is used. In FORTRAN an array of dimensions N by M declared with maximum row dimension IA, occupies IA*M real number memory locations. In EASYPACK, an array A of dimensions N by M is stored as a list of (N, M, A(1,1), A(2,1),..., A(N,M)).

Usage

A typical main program using EASYPACK is given below

```
(1)     Dimension A(100), B(100), C(100)
(2)     Common maxwrk, WORK(100)
(2.1)   maxwrk=100
(3)     Call MINP( B, 'Enter matrix B$')
(4)     Call MINP( C, 'Enter matrix C$')
(5)             Call MUL( A, B, C )
(6)             Call GINV( B, B )
(7)     Call MPRN( A, 'This is the product B*C$' )
(8)     Call MPRN( B, 'This is the inverse of B$')
(9)     END
```

Line (2) is necessary to declare some work space needed for EASYPACK
Line (2.1) is used to specify the available work space for the subroutines. If an EASYPACK subroutine needs more work-space than maxwrk, an error message is printed.
Notice the readability of the code, the simplicity of writing it and the 'dimensionless' calling of the subroutines.
The following example shows how to call the FORTRAN routine

```
SUBROUTINE EXAMPLE( ARRAY, IA, N, M, ... )
```

from a main program that uses matrices in EASYPACK format; that is,

```
Real A(100)
......
CALL EXAMPLE( A(3), A(1), A(1), A(2), ... )
```

Note that despite the fact IA, N and M are of type integer, and A is of type real, when EXAMPLE is called the dimensions are passed as the addresses of A(1) and A(2). (Note: assume that FORTRAN stores the integers in two bytes and the reals in 4 bytes. Assume also that ADDR is the address of A(1). Then, N is stored in ADDR and ADDR+1, and M is stored at ADDR+3 and ADDR+4. In this manner, A(1) and A(2) correspond to N and M respectively. All FORTRAN particularities are handled by SETDIM, NDIM, MDIM and NM2DIM. In this manner the portability of the package is established.)
The following example shows how to convert the FORTRAN array A with dimension N, M, and maximum row dimension IA to EASYPACK format

```
CALL PACK( A, IA, N, M )
```

Finally, if A is in EASYPACK format,

`N = NDIM( A )`	gives the row dimension of A
`M = MDIM( A )`	gives the column dimension of A
`s = ELE( A, i, j )`	gives s=A(i,j) and checks if the indices i and j are inside the dimensions of A.
`s = A( ii )`	where ii = 2+i+(NDIM(A)-1)*j, gives the (i,j) element of A with no dimension checking.

Conventions

The following conventions are used for the arguments of the subroutines:

(1) A are result matrices in EASYPACK format
(2) B, C are parameter matrices in EASYPACK format
(3) X is a result vector
(4) Y, Z are parameter vectors
(5) s, p, q, are scalar quantities.
(6) When there are more than two arguments, in place operation is allowed; for example, Call MUL(A, A, C) will perform A:=A*C.
(7) When a routine is called, the arguments have the form result1, result2, ..., parameter1, parameter2,... or, destination1, destination2, ..., source1, source2, ...

EASYPACK Utilities and I/O

```
Call PACK( A, IA,N,M )
        A := packed form of A
        On entry, A is a N by M matrix with max row dimension IA.
        On exit, A is in EASYPACK format

N = NDIM( B ),  M = MDIM( B ),  NM2 = NM2DIM( B )
        Return the number of rows (N), the number of columns (M),
        and the total number of elements (N*M+2) of the matrix B.
        It is assumed that B is in EASYPACK format.

Call SETDIM( A, N, M )
        Redefines A to new dimensions N by M.
        On entry and on exit, A is in EASYPACK format.

s = ELE( B, i, j )
        Returns s:=B(i,j) when B is in EASYPACK format.

Call MINP( A, 'comment$' )
        Inputs the matrix A in EASYPACK format.
        Prints the comment first, then asks for dimensions and
        finally reads the matrix from the terminal.
        Checks for valid dimensions greater then 1.
```

```
Call MPRN( A, 'comment$' )
       Prints the matrix A.  (A in EASYPACK format)
       It prints the comment first, then the dimensions and
       finally the matrix row by row.
       The maximum number of elements per row is 20.

Call ALTER( A, 'comment$' )
       Used to correct a matrix A.  (A in EASYPACK format)
       Prints the comment, then asks for (i,j) indices;
       prints that element and allows alterations.
       If j is not given, 1 is assumed.  Use i=0 to exit.
       No checking if i and j are inside the dimensions of A.

Call MGET( 'filename.ext$', A )
       Reads A from the (binary) filename given.
       A in EASYPACK format
       Calls RDMBIN( 'filename$', A, N, M )

Call MSTORE( 'filename.ext$', A )
       Stores A in the filename given.  Binary form is used.
       A in EASYPACK format
       Calls WRMBIN( 'filename$', A, N, M )

Call RDCBIN( 'filename.ext$', X, N )
       Reads the vector X(1 to N) from the given binary file

Call WRCBIN( 'filename.ext$', X, N )
       Writes the vector X(1 to N) to the given binary file.
```

EASYPACK matrix operations and functions

```
Call UNIT( A, N )
       Defines A to be the unity matrix N by N.
       A in EASYPACK format

Call ZERO( A, N, M )
       Defines A to be the zero matrix N by M.
       A in EASYPACK format

Call RAMP( A, N )
       Defines A to be N by 1 with elements 0, 1, 2, ..., N-1
```

```
        A in EASYPACK format

Call EQU( A, B )
        Equates A to B.
        A and B in EASYPACK format

Call ADD( A, B, C )
        Matrix addition A:=B+C
        A, B, C in EASYPACK format
        Checks if B and C have the same dimensions.

Call SUB( A, B, C )
        Matrix subtraction A:=B-C
        A, B, C in EASYPACK format
        Checks if B and C have the same dimensions.

Call SADD( A, B, s, C )
        Performs the operation  A:=B+sC
        A, B, C in EASYPACK format
        Checks if B and C have the same dimensions.

Call SCALE( A, s, B )
        Scales B by s; that is A:=sB
        A, B  in EASYPACK format

Call TRN( A, B )
        Finds the transpose of B, that is A:=(B transpose).
        A, B in EASYPACK format

Call MUL( A, B, C )
        Finds the matrix product A:=B*C
        Checks if B and C are comformable.
        A, B, C in EASYPACK format
        A common block of size A is needed.

Call MULT( A, B, C )
        Performs the multiplication A:=B * (transpose of C)
        Checks if B and C-transpose are comformable.
        A, B, C in EASYPACK format
        A common block of size A is needed

Call QUA( A, B, C )
        Finds the matrix A:= (transpose of B)*C*B.
```

Checks if B-transpose, C and B are comformable.
A, B, C in EASYPACK format
A common block of size A is needed.

Call GINV(A, B)

A := generalized inverse of B
Common: the size of B.
In place inversion is allowed, i.e. Call GINV(A, A)
A, B in EASYPACK format.
No dimension checking.
B can be N by M with N>M but N<30. Least squares solution.
If B is singular, the routine prints the rank of B and the pseudo inverse is calculated.

Call PART(A, B, N1,N2,M1,M2)

Creates matrix A by extracting the N1...N2 rows and M1...M2 columns of B.
In place operations are NOT allowed.
A, B in EASYPACK format
No dimensional checking takes place.

Call AUGM(A, N, M, B)

Augments matrix A by inserting matrix B. The upper left element of A where B is substituted has indices N, M.
In place augmentation is NOT allowed.
A, B in EASYPACK format
Checks if the dimensions of B 'fit' in A.

s = XNORM1(B)

Finds the 1-norm of a square matrix B; that is,
s is the maximum sum of the absolute values of the elements of the columns of B.
B in EASYPACK format

s = XNORM(B)

Finds the infinity norm of a square matrix B; that is,
s is the maximum sum of the absolute values of the elements of the rows of B.
Note that XNORM(B)=XNORM1(B-transpose).
B in EASYPACK format

EASYPACK Vector Operations and Functions

```
Call VEQU( X, Y, N )
```
Equates X and Y; X:=Y (1 to N).

```
Call VSWAP( X, Y, N )
```
Swaps the X and Y vectors, that is X:=Y and Y:=X.

```
Call VZERO( X, N )
```
Defines a zero vector of N elements.

```
Call VSCALE( X, s, N )
```
Scales X in place, that is, X:=sX

```
Call VSADD( X, s, Y, N )
```
Performs the vector addition X:=X+sY.

```
s = VDOT( Y, Z, N )
```
Finds the dot product of Y and Z. That is, s:=(Y-transpose)*Z.

```
s = VMIN( Y, N )
```
Finds the minimum element of Y from 1 to N.

```
s = VMAX( Y, N )
```
Finds the maximum element of Y from 1 to N.

```
s = VNORM1( Y, N )
```
Finds the 1-norm of the vector Y; that is the sum of the absolute values of the elements of Y.

```
s = VNORM( Y, N )
```
Finds the infinity norm of the vector Y; that is, s equals to the element of Y with the largest absolute value.

```
Call CFFT( X, Y, N )
```
Performs in place complex FFT transform on the X+jY time series. N must be a power of 2.

```
Call IFFT( X, Y, N )
```
Performs in place the inverse FFT transform. N must be a power of 2.

```
Call MAGN( X, Y, N )
```
Converts the X+jY complex vector of N elements to polar coordinates. The magnitude is passed back in X and the phase is passed in Y (in radian).

```
Call RECT( X, Y, N )
```
Converts the polar coordinates X*exp(jY) to rectangular coordinates X+jY.

EASYPACK scalar operations

```
i = IGET('comment$')
```
Print the comment and wait till an integer is typed

```
r = RGET('comment$')
```
Print the comment and wait till a real is typed.

```
Call SGET('comment$', a, N )
```
Read the array a(1 to N) using A1 format

```
Call PRN( 'comment$' )
```
Print the comment.

```
Call IPRN( i, 'comment$')
```
Print the integer i, then the comment (decimal form).

```
Call RPRN( r, 'comment$')
```
Print the real r, then the comment

```
Call HPRN( i, 'comment$')
```
Print the real i in hexadecimal form.

```
Call GOPEN( i, 'filename$')
```
Open a file. i is the channel number

```
a = DEG(x), a = RAD(x), a = PI(b)
```
DEG converts x (in rad) to degrees
RAD converts x (in degrees) to rad.
PI returns a := b times pi.

2. EASYPACK Source Code

```
C-------------------------------------------------------------C
C       EASYPACK Rev 2.0, 2.1, 2.2                            C
C                                                             C
C       Common block /SCRACT/ of 10 integer places            C
C       is used to save some memory for local variables.      C
C       Rev 2.2 uses maxwrk to test available work space      C
C               corrects the problem with MGET and adds       C
C               routine RECT and VSWAP.                       C
C                                                             C
C       Common block WORK is used as a work space.            C
C                                                             C
C       External Calls: PRN, IPRN, GOPEN                      C
C                                                             C
C       George Kontopidis, March 81                           C
C-------------------------------------------------------------C
C
C---GINV
C
        SUBROUTINE GINV(A,B)
        DIMENSION A(1),B(1)
        COMMON maxwrk,W(1)
        N = NDIM( B )
        M = MDIM( B )
        If( maxwrk.LT.N*M ) Call PRN('Small wrksp for GINV$')
        CALL EQU(W,B)
        CALL GMINV(W(3),N,N,M, A,MR)
        Call PACK( A, N,M,N )
        RETURN
        END
C
C---GMINV
C
        SUBROUTINE GMINV(A,IDIM,NR,NC,U,MR)
C
C Rust, B., Burrus, W.R. and Schneeberger, C., 'A Simple Algorithm
C for Computing the Generalized Inverse of a Matrix', Comm. ACM
C Vol. 9, No. 5, May 1966.
C
        DIMENSION A(1),U(1),S(30)
C
```

```
      IDIM1=IDIM+1
      TOL=1.E-14
      ADV=1.E-24
      MR=NC
      NRM1=NR-1
      TOL1=0.
      JJ=1
C
      DO 10 J=1,NC
            S(J)=VDOT(A(JJ),A(JJ),NR)
            IF(S(J).GT.TOL1) TOL1=S(J)
            JJ=JJ+IDIM
10    CONTINUE
C
      TOL1=ADV*TOL1
      ADV=TOL1
C
      JJ=1
      DO 100 J=1,NC
            FAC=S(J)
            JM1=J-1
            JRM=JJ+NRM1
            JCM=JJ+JM1
            DO 20 I=JJ,JCM
                  U(I)=0.
20          CONTINUE
            U(JCM)=1.0
            IF(J.EQ.1) GO TO 54
            KK=1
            DO 30 K=1,JM1
                  IF(S(K).EQ.1.0) GO TO 30
                  TEMP=-VDOT( A(JJ),A(KK),NR )
                  CALL VSADD(U(JJ),TEMP,U(KK),K)
                  KK=KK+IDIM
30          CONTINUE
            DO 51 L=1,2
                  KK=1
                  DO 50 K=1,JM1
                        IF(S(K).EQ.0.) GO TO 50
                        TEMP=-VDOT(A(JJ),A(KK),NR )
                        CALL VSADD(A(JJ),TEMP,A(KK),NR)
                        CALL VSADD(U(JJ),TEMP,U(KK),K)
```

```
                        KK=KK+IDIM
50                  CONTINUE
51              CONTINUE
C
                TOL1=TOL*FAC+ADV
                FAC=VDOT(A(JJ),A(JJ),NR)
54              IF(FAC.GT.TOL1) GO TO 70
                DO 55 I=JJ,JRM
55              A(I)=0.
                S(J)=0.
C
                KK=1
                        DO 65 K=1,JM1
                                IF(S(K).EQ.0.) GO TO 65
                                TEMP=-VDOT(U(KK),U(JJ),K)
                                CALL VSADD(A(JJ),TEMP,A(KK),NR)
                                KK=KK+IDIM
65                      CONTINUE
                        FAC=VDOT( U(JJ),U(JJ),J )
                        MR=MR-1
                        GO TO 75
70                      S(J)=1.0
                        KK=1
                        DO 72 K=1,JM1
                                IF(S(K).EQ.1.) GO TO 72
                                TEMP=-VDOT( A(JJ),A(KK),NR )
                                CALL VSADD(U(JJ),TEMP,U(KK),K)
                                KK=KK+IDIM
72                      CONTINUE
75                      FAC=1./SQRT(FAC)
C
                        DO 80 I=JJ,JRM
80                      A(I)=A(I)*FAC
C
                        DO 85 I=JJ,JCM
85                      U(I)=U(I)*FAC
                JJ=JJ+IDIM
100     CONTINUE
C
        IF(MR.EQ.NR.OR.MR.EQ.NC) GO TO 120
        WRITE(5,110) NR,NC,MR
110     FORMAT(' ',I3,1HX,I2,8H M:RANK,I2)
```

```
120     NEND=NC*IDIM
C
        JJ=1
        DO 135 J=1,NC
                DO 126 I=1,NR
                        II=I-J
                        S(I)=0.
                        DO 125 KK=JJ,NEND,IDIM
                                IK=II+KK
                                S(I)=S(I)+A(IK)*U(KK)
125                     CONTINUE
126             CONTINUE
                II=J
                DO 130 I=1,NR
                        U(II)=S(I)
                        II=II+IDIM
130             CONTINUE
        JJ=JJ+IDIM1
135     CONTINUE
        RETURN
        END
C
C---PACK
C
        SUBROUTINE PACK(A,IA,N,M)
        DIMENSION A(1)
        COMMON/SCRACT/NM,I,J,I1,I2,NOTH(5)
        NM=N*M
C
C       COMPRESS THEM IN PLACE
C
        DO 10 J=1,M
        DO 10 I=1,N
                I1=I+(J-1)*N
                I2=I+(J-1)*IA
                A( I1 )=A( I2 )
10      CONTINUE
C
C       SHIFT BY TWO
C
        DO 20 J=1,NM
                I=NM+1-J
```

```
                I2=I+2
                A(I2)=A(I)
20      CONTINUE
        CALL SETDIM(A,N,M)
        RETURN
        END
C
C---ALTER
C
        SUBROUTINE ALTER(A,COMMEN)
        DIMENSION A(1)
        COMMON/SCRACT/ I,J,II,NOTH(7)
        CALL PRN( COMMEN )
1       WRITE(5,100)
        READ(5,200) I,J
        IF(I.LE.0) RETURN
        IF(J.EQ.0) J=1
        II=2+I+(J-1)*NDIM(A)
        WRITE(5,300) I,J,A(II)
        READ(5,400) A(II)
        GO TO 1
10      RETURN
C
100     FORMAT(' Enter indices (0 to exit): ')
200     Format(2I6)
300     FORMAT(' Element (',I4,',',I4,') =',G14.7,'  Enter value: ')
400     FORMAT(1G14.7)
        END
C
C---MGET
C
        SUBROUTINE MGET(FILE, A)
        DIMENSION A(1), FILE(1)
        CALL RDMBIN( FILE, A(3), A(1), A(2) )
        RETURN
        END
C
C---MSTORE
C
        SUBROUTINE MSTORE(FILE,A)
        DIMENSION A(1), FILE(1)
        CALL WRMBIN( FILE, A(3), A(1), A(2))
```

```
        RETURN
        END
C
C---UNIT
C
        SUBROUTINE UNIT(A,N)
        DIMENSION A(1)
        COMMON/SCRACT/I,NN,NOTH(8)
        NN=N*N
        CALL SETDIM(A,N,N)
        CALL VZERO(A(3),NN)
        DO 20 I=1,N
                NN=2+I+(I-1)*N
                A(NN)=1.
20      CONTINUE
        RETURN
        END
C
C---ZERO
C
        SUBROUTINE ZERO( A,N,M )
        DIMENSION A(1)
        COMMON/SCRACT/NM,I,NOTH(8)
        CALL SETDIM(A,N,M)
        NM=N*M
        CALL VZERO(A(3),NM)
        RETURN
        END
C
C---RAMP
C
        SUBROUTINE RAMP(A,N)
        DIMENSION A(1)
        COMMON /SCRACT/I,I2,NOTH(8)
        CALL SETDIM(A,N,1)
        DO 10 I=1,N
                I2=I+2
                A(I2)=I-1
10      CONTINUE
        RETURN
        END
C
```

```
C---MPRINT
C
        SUBROUTINE MPRN(A,COMMEN)
        DIMENSION A(1),COMMEN(1)
        COMMON/SCRACT/N,M,I,I1,I2,K,NOTH(4)
        CALL PRN( COMMEN )
        N=NDIM(A)
        M=MDIM(A)
        WRITE(5,100) N,M
100     FORMAT(' Dimensions: ',I6,'  by ',I6)
        DO 500 I=1,N
                I1=2+I
                I2=2+I+(M-1)*N
                WRITE(5,600) I,(A(K),K=I1,I2,N)
500     CONTINUE
        CALL CRLF
        RETURN
600     FORMAT(' ',I4,': ',20G10.3)
        END
C
C---MINPUT
C
        SUBROUTINE MINP(A,COMMEN)
        DIMENSION A(1),COMMEN(1)
        COMMON/SCRACT/ N,M,I1,I2,K,NOTH(5)
        CALL PRN(COMMEN)
1       CALL PRN( 'Type dimensions: $')
        READ(5,200) N,M
200     FORMAT(2I6)
        IF(N*M) 1,1,2
2       CALL PRN('Type now the matrix row by row $')
        Call CRLF
                DO 50 I=1,N
                I1=2+I
                I2=2+I+(M-1)*N
                READ(5,400) (A(K),K=I1,I2,N)
50              CONTINUE
400     FORMAT(20G14.7)
        CALL SETDIM(A,N,M)
        RETURN
        END
C
```

```
C
C---DEG
C
      REAL FUNCTION DEG(X)
      DEG=180.*X/PI(1.)
      RETURN
      END
C
C---RAD
C
      REAL FUNCTION RAD(X)
      RAD=X*PI(1.)/180.
      RETURN
      END
C
C---PI
C
      REAL FUNCTION PI(S)
      DATA PII/3.141592654/
      PI=S*PII
      RETURN
      END
C
C---EQU
C
      SUBROUTINE EQU(A,B)
      DIMENSION A(1),B(1)
      COMMON /SCRACT/NM2,NOTH(9)
      NM2=NM2DIM(B)
      CALL VEQU(A(1),B(1),NM2)
      RETURN
      END
C
C---ADD
C
      SUBROUTINE ADD(A,B,C)
      CALL SADD(A,B,1.,C)
      RETURN
      END
C
C---SUB
C
```

```
        SUBROUTINE SUB(A,B,C)
        CALL SADD(A,B,-1.,C)
        RETURN
        END
C
C----SADD
C
        SUBROUTINE SADD(A,B,S,C)
        DIMENSION A(1),B(1),C(1)
        COMMON/SCRACT/I,J,K,NB,MB,NM2,NOTH(4)
        NB=NDIM(B)
        MB=MDIM(B)
        IF( ( NB-NDIM(C) )*( MB-MDIM(C) ) ) 100,200,100
200     NM2=NB*MB+2
        DO 10 I=3,NM2
10      A(I)=B(I)+S*C(I)
        CALL SETDIM(A,NB,MB)
        RETURN
100     Call PRN('Dim Err SADD$')
        RETURN
        END
C
C----SCALE
C
        SUBROUTINE SCALE(A,S,C)
        DIMENSION A(1),C(1)
        COMMON /SCRACT/M,NM,I,N,NOTH(6)
        N=NDIM(C)
        M=MDIM(C)
        NM=NM2DIM(C)
        DO 10 I=3,NM
10      A(I)=S*C(I)
        CALL SETDIM(A,N,M)
        RETURN
        END
C
C---MULT
C
        SUBROUTINE MULT(A,B,C)
        CALL TRN(A,C)
        CALL MUL(A,B,A)
        RETURN
```

```
        END
C
C---QUA
C
        SUBROUTINE QUA(P,A,Q)
        CALL TRN(P,A)
        CALL MUL(P,Q,P)
        CALL MUL(P,A,Q)
        RETURN
        END
C
C---MUL
C
        SUBROUTINE MUL(A,B,C)
        DIMENSION A(1),B(1),C(1)
        COMMON maxwrk, W(1)
        COMMON /SCRACT/ I1,M1,M2,N2,I,J,K,NOTH(3)
        N1=NDIM(B)
        M1=MDIM(B)
        M2=MDIM(C)
        If (maxwrk.LT.N1*M2) Call PRN('Small wrksp for MUL$')
        IF( M1-NDIM(C) ) 200,100,200
100     DO 20 I=1,N1
        DO 20 K=1,M2
                SUM=0.0
                DO 10 J=1,M1
10              SUM=SUM+ELE(B,I,J)*ELE(C,J,K)
        I1=I+(K-1)*N1
20      W(I1)=SUM
        I1=N1*M2
        CALL SETDIM(A,N1,M2)
        CALL VEQU(A(3),W(1),I1)
        RETURN
200     Call PRN('Dim Err MUL$')
        RETURN
        END
C
C---TRN
C
        SUBROUTINE TRN(A,B)
        DIMENSION A(1),B(1)
        COMMON maxwrk,W(1)
```

```
        COMMON /SCRACT/ N,M,I,J,I1,NOTH(5)
        N=NDIM(B)
        M=MDIM(B)
        If (maxwrk.LT.N*M) Call PRN('Small wrksp for TRN$')
        DO 10 I=1,N
        DO 10 J=1,M
                I1=J+(I-1)*M
                W(I1)=ELE(B,I,J)
10      CONTINUE
        CALL VEQU( A(3),W(1),I1 )
        CALL SETDIM(A,M,N)
        RETURN
        END
C
C---PART
C
        SUBROUTINE PART(S,X,N1,N2,M1,M2)
        DIMENSION X(1),S(1)
        COMMON/SCRACT/I,J,K,NOTH(7)
                K=3
        DO 10 J=M1,M2
        DO 10 I=N1,N2
                S(K)=ELE(X,I,J)
                K=K+1
10      CONTINUE
        I=N2-N1+1
        J=M2-M1+1
        CALL SETDIM(S,I,J)
        RETURN
        END
C
C---AUGM
C
        SUBROUTINE AUGM(S,N,M,X)
        DIMENSION S(1),X(1)
        COMMON /SCRACT/ II1,MX,NX,NS,I1,I,J,J1,NOTH(2)
        NX=NDIM(X)
        MX=MDIM(X)
        NS=NDIM(S)
        IF(N+NX-1-NS) 1,1,2
1       IF(M+MX-1-MDIM(S)) 3,3,2
3               I1=N
```

```
        DO 20 I=1,NX
                J1=M
        DO 10 J=1,MX
                II1=2+I1+(J1-1)*NS
                S(II1)=ELE(X,I,J)
                J1=J1+1
10      CONTINUE
                I1=I1+1
20      CONTINUE
        RETURN
2       Call PRN('Dim Err AUGM$')
        RETURN
        END
C
C---XNORM1---
C
C 1-NORM OF A SQUARE MATRIX
C
        REAL FUNCTION XNORM1(A)
        DIMENSION A(1)
        COMMON/SCRATC/ N,I,J,VSUM,TEMP,J1,NOTH(2)
        N=NDIM(A)
        TEMP=0.0
                DO 5 J=1,N
                J1=3+(J-1)*N
                VSUM=SUMABS(A(J1),1,N)
                IF(VSUM.GE.TEMP) TEMP=VSUM
5               CONTINUE
        XNORM1=TEMP
        RETURN
        END
C
C---XNORM---
C
C INFINITY NORM OF A REAL SQUARE MATRIX
C
        REAL FUNCTION XNORM(A)
        DIMENSION A(1)
        COMMON/SCRATC/I,N,I2,TEMP,VSUM,I1,NOTH(2)
        TEMP=0.0
        N=NDIM(A)
                DO 5 I=1,N
```

```
                I2=I+2
                VSUM=SUMABS(A(I2),N,N)
                IF(VSUM.GE.TEMP) TEMP=VSUM
5               CONTINUE
        XNORM=TEMP
        RETURN
        END
C
C---VEQU
C
        SUBROUTINE VEQU(A,B,N)
        DIMENSION A(1),B(1)
        DO 10 I=1,N
10      A(I)=B(I)
        RETURN
        END
C
C---VZERO
C
        SUBROUTINE VZERO(X,N)
        DIMENSION X(1)
        DO 10 I=1,N
10      X(I)=0.
        RETURN
        END
C
C---VSADD---
C
        SUBROUTINE VSADD(A,C1,B,N)
        DIMENSION A(1),B(1)
        DO 1 I=1,N
1       A(I)=A(I)+C1*B(I)
        RETURN
        END
C
C---VDOT
C
        REAL FUNCTION VDOT(X,Y,N)
        DIMENSION X(1),Y(1)
        VDOT=0.
        DO 10 I=1,N
10      VDOT=VDOT+X(I)*Y(I)
```

```
        RETURN
        END
C
C---VMIN, VMAX
C
        REAL FUNCTION VMIN(X,N)
        DIMENSION X(1)
        VMIN=1.E+30
        DO 100 I=1,N
        VMIN=AMIN1( VMIN, X(I) )
100     CONTINUE
        RETURN
        END
C
        REAL FUNCTION VMAX(X,N)
        DIMENSION X(1)
        VMAX=-1.E+30
        DO 100 I=1,N
        VMAX=AMAX1( VMAX, X(I) )
100     CONTINUE
        RETURN
        END
C
C---VNORM1---
C
C VECTOR NORM-1
C
        REAL FUNCTION VNORM1(A,N)
        VNORM1=SUMABS(A,1,N)
        RETURN
        END
C
C---VNORM
C
C VECTOR NORM INFINITY
C
        REAL FUNCTION VNORM(A,N)
        DIMENSION A(1)
        VNORM=0.
        DO 5 I=1,N
        VNORM=AMAX1( VNORM, ABS(A(I)) )
5       CONTINUE
```

```
        RETURN
        END
C
C---VSWAP
C
        Subroutine VSWAP( X,Y,N )
        Dimension X(1),Y(1)
        Do 10 i=1,N
          temp = X(i)
          X(i) = Y(i)
          Y(i) = temp
10      Continue
        Return
        End
C
C--- IFFT( X,Y,N )
C
        Subroutine IFFT( X,Y,N )
        Dimension X(1),Y(1)
        s1=1./float(N)
        s2=-s1
        Call VSCALE( Y,s2,N )
        Call VSCALE( X,s1,N )
        Call CFFT( X,Y,N )
        s1=-1.
        Call VSCALE( Y,s1,N )
        Return
        END
C
C--- VSCALE( X,s,N )
C
        Subroutine VSCALE( X,s,N )
        Dimension X(1)
        Do 10 i=1,N
10      X(i)=s*X(i)
        Return
        END
C
C--- COMPUTES THE COMPLEX FFT A TIME SERIES
C
        SUBROUTINE CFFT(DATA1, DATA2, N)
C
```

```
      DIMENSION DATQ1(1), DATA2(1)
      PI=4.*ATAN2(1.,1.)
      FN=N
C
C... THIS SECTIONOPUTS TATA IN BIT-REVERSED ORDER
C
      J=1
      DO 80 I=1,N
C
C... AT THIS POINT, I AND J ARE A BIT REVERSED PAIR
C
      IF(I-J) 30,40,40
C
C... EXCHANGE DATA(I) WITH DATA(J) IF I.LT.J
C
30    TEMP1=DATA1(J)
      TEMP2=DATA2(J)
      DATA1(J)=DATA1(I)
      DATA2(J)=DATA2(I)
      DATA1(I)=TEMP1
      DATA2(I)=TEMP2
C
C... IMPLEMENT J=J+1 BIT REVERSED COUNTER
C
40    M=N/2
50    IF (J-M) 70, 70, 60
60    J=J-M
      M=(M+1)/2
      GOTO 50
70    J=J+M
80    CONTINUE
C
C... NOW COMPUTE THE BUTTERFLIES
C
      MMAX=1
90    IF (MMAX-N) 100,130,130
100   ISTEP=2*MMAX
      DO 120 M=1,MMAX
            THETA=PI*FLOAT( (-1)*(M-1))/FLOAT(MMAX)
            W1=COS( THETA )
            W2=SIN( THETA )
C
```

```
                DO 110 I=M,N,ISTEP
                        J=I+MMAX
                        TEMP1=W1*DATA1(J)-W2*DATA2(J)
                        TEMP2=W2*DATA1(J)+W1*DATA2(J)
                        DATA1(J)=DATA1(I)-TEMP1
                        DATA2(J)=DATA2(I)-TEMP2
                        DATA1(I)=DATA1(I)+TEMP1
                        DATA2(I)=DATA2(I)+TEMP2
110             CONTINUE
120     CONTINUE
        MMAX=ISTEP
        GOTO 90
130     RETURN
        END

C
C--- FINDS THE MAGNITUDE AND THE PHASE OF COMPLEX DATA
C
C... ENTER WITH THE DATA IN X, Y ARRAYS OF LENGTH N.  RETURNS
C    WITH THE MAGNITUDE IN X AND PHASE IN Y
C
        SUBROUTINE MAGN( X, Y, N )
        DIMENSION X(1), Y(1)
C
        DO 10 I=1, N
        TEMP=X(I)
C
        X(I)=SQRT( X(I)*X(I) + Y(I)*Y(I) )
        IF(TEMP.NE.0.) Y(I)=ATAN2( Y(I) , TEMP )
C
10      CONTINUE
        RETURN
        END
C
C---RECT---
C
C converts to rectangular coordinates
C
        Subroutine RECT( X,Y,N )
        Dimension X(1),Y(1)
        Do 10 i=1,N
          amag = X(i)
```

```
          X(i) = amag * cos (Y(i))
          Y(i) = amag * sin (Y(i))
10        Continue
          Return
          End
C
C---SUMABS---
C
C FINDS THE SUM OF THE ABSOLUTE VALUES OF A VECTOR
C
          REAL FUNCTION SUMABS(V,INC,L)
          DIMENSION V(1)
          TEMP=0.
          J=1+(L-1)*INC
                  DO 5 I=1,J,INC
                  TEMP=TEMP+ABS(V(I))
5                 CONTINUE
          SUMABS=TEMP
          RETURN
          END
C
C---ELE
C
          REAL FUNCTION ELE(A,I,J)
          DIMENSION A(1)
          IPOS=NDIM(A)
          IF( I.GT.IPOS    ) CALL PRN('ELE err: i out of range$')
          IF( J.GT.MDIM(A) ) CALL PRN('ELE err: j out of range$')
          IPOS=2+I+(J-1)*IPOS
          ELE=A(IPOS)
          RETURN
          END
C
C---NDIM, MDIM, NM2DIM
C
          INTEGER FUNCTION NDIM( IA )
          NDIM=IA
          RETURN
          END

          INTEGER FUNCTION MDIM( IA )
          DIMENSION IA(3)
```

```
        MDIM=IA(3)
        RETURN
        END

        INTEGER FUNCTION NM2DIM( IA )
        DIMENSION IA(3)
        NM2DIM=IA(1)*IA(3)+2
        RETURN
        END
C
C---SETDIM
C
        SUBROUTINE SETDIM(IA,N,M)
        DIMENSION IA(3)
        IA(1)=N
        IA(3)=M
        RETURN
        END
C
C--- READ A COLUMN FROM A BINARY FILE
C
C  THE FILE IS OPENED AND AFTER READING IT IS
C  CLOSED.
C
        SUBROUTINE RDCBIN( FILE,DATA,N )
        CALL RDMBIN(FILE,DATA,N,M)
        RETURN
        END
C
        SUBROUTINE RDMBIN( FILE,DATA,N,M )
        DIMENSION DATA(1),BUFFER(32)
        LOGICAL FILE(1)
C
C OPEN THE FILE AND READ THE DATA
C
        CALL GOPEN(1,FILE)
        IREC=0
        READ(1,REC=1,ERR=2,END=1) BUFFER
        N=BUFFER(1)
        M=BUFFER(2)
        NM=N*M
C
```

```
C READ THE ACTUAL DATA
C
        IREC=2
51      READ(1,REC=IREC,ERR=2,END=1) BUFFER
        DO 56 I=1,32
                I1=(IREC-2)*32+I
                If (i1.GT.NM) Go to 3
                DATA(I1)=BUFFER(I)
56      CONTINUE
C
        IREC=IREC+1
        IF( I1.LT.NM ) GO TO 51
        ENDFILE 1
        RETURN
1       CALL IPRN(IREC,'=(record numb) passes the end of file mark$')
        ENDFILE 1
        RETURN
2       CALL IPRN(IREC,'=(record numb) error in reading binary file$')
3       ENDFILE 1
        Return
        END
C
C---- WRITE BINARY FILE
C
        SUBROUTINE WRCBIN( FILE,DATA,N)
        CALL WRMBIN(FILE,DATA,N,1)
        RETURN
        END

        SUBROUTINE WRMBIN( FILE,DATA,N,M )
        DIMENSION DATA(1),BUFFER(32)
        LOGICAL FILE(1)
C
        CALL GOPEN(2,FILE)
        BUFFER(1)=N
        BUFFER(2)=M
        DO 10 J=3,32
10      BUFFER(J)=0.
        WRITE(2,REC=1,ERR=2) BUFFER
C
        IREC=2
51      DO 56 I=1,32
```

```
                I1=(IREC-2)*32+I
                BUFFER(I)=DATA(I1)
56      CONTINUE
C
        WRITE(2,REC=IREC,ERR=2) BUFFER
        IREC=IREC+1
        IF( I1.LT.N*M) GO TO 51
        ENDFILE 2
        RETURN
2       Call IPRN(IREC,'=error in writting this record number$')
        ENDFILE 2
        Return
        END
C
C--- READ A NUMBER FROM BINARY FILE
C
C       Usage:  A=DSKRD( index, device-number )
C
        FUNCTION DSKRD(I,IDEV)
        DIMENSION BUFFER(32)
        IREC=(I-1)/32 + 1
        IPOS=I-(IREC-1)*32
        IREC=IREC+1
        READ(IDEV,REC=IREC,ERR=2,END=1) BUFFER
        DSKRD=BUFFER(IPOS)
        RETURN
1       CALL IPRN(I,'=passes the end of file mark$')
        RETURN
2       CALL IPRN(I,'=error in reading binary file$')
        Return
        END
```

3. EASY: Matrix Reverse Polish Calculator

```
      C-----------------------------------------C
      C            Rev 1.0, 1.1                 C
      C            G.K March 81                 C
      C-----------------------------------------C

      Dimension X(100),Y(100),Z(100),T(100)
      Dimension R1(100),R2(100)
      Logical file(20)
      Real    table(52)
      Common maxwrk,W(100)
      Common /regX/ X
      Common /regY/ Y
      Common /regZ/ Z
      Common /regT/ T
C
      Data table/
     1  'minp','mprn','alte','mget','msto','map ','rot ','unit',
C        10     20     30     40     50     60     70     80
C
     2  'zero','add ','sub ','scal','xchg','trn ','mul ','tmul',
C        90     100    110    120    130    140    150    160
C
     3  'mult','ginv','cfft','ifft','magn','max ','min ','help',
C        170    180    190    200    210    220    230    240
C
     4  'in  ','x   ','y   ','z   ','t   ','get ','save','1   ',
C        10     20     21     22     23     40     50     80
C
     5  '+   ','-   ','xy  ','xz  ','xt  ',4H    ,'*   ',4H'*  ,
C        100    110    130    131    132   140     150   160
C
     6  4H*'  ,''   ','#   ','0   ','sto1','rec1','push','plot',
C        170    180    120    90     250    260    270    280
C
     7  'sto2','rec2','    ','rect'/
C        290    300    001    310
C
C--- initiation
C
      maxwrk=100
```

```
        CALL PRN('    Matrix Reverse Polish Calulator$')
        Call PRN('  George Kontopidis  Apr 81, Rev 1.1$')
        Call CRLF
        NC=52
        i=5
        Call ZERO( X,i,i )
        Call ZERO( Y,i,i )
        Call ZERO( Z,i,i )
        Call ZERO( T,i,i )
        Call ZERO( W,i,i )
C
C--- main prompting loop
C
1       Call PRN('ready... $')
        Read(5,3000) cmd
3000    Format(A4)
C
C... search for command pointer
C
        Do 2 i=1,NC
2       If(cmd.EQ.table(i)) Go to 3
        Call PRN('Illegal Command$')
        Go to 1
C
C... dispatch
C
3       Goto ( 10, 20, 30, 40, 50, 60, 70, 80,
     1          90,100,110,120,130,140,150,160,
     2         170,180,190,200,210,220,230,240,
     3          10, 20, 21, 22, 23, 40, 50, 80,
     4         100,110,130,131,132,140,150,160,
     5         170,180,120, 90,250,260,270,280,
     6         290,300,  1,310                  ),i
        PAUSE EASY??
C
C                                       --- minp, in
10      Call PUSH
        Call MINP( X,'X matrix$')
        Go to 1
C
C                                       --- mprn, x
20      CALL MPRN( X,'X matrix$')
```

```
      Go to 1
C
C                                     --- y
21    Call MPRN( Y,'Y matrix$')
      Go to 1
C
C                                     --- z
22    Call MPRN( Z,'Z matrix$')
      Go to 1
C
C                                     --- t
23    Call MPRN( T,'T matrix$')
      Go to 1
C
C                                     --- alter
30    Call ALTER( X,'X matrix$')
      Go to 1
C
C                                     --- mget, get
40    Call PUSH
      Call SGET('Enter filename: $',file,20)
      Call MGET( file, X )
      Go to 1
C
C                                     --- mstore, save
50    Call SGET('Enter filename: $',file,20)
      Call MSTORE( file, X )
      Go to 1
C
C                                     --- map
60    nx=NDIM(X)
      mx=MDIM(X)
      ny=NDIM(Y)
      my=MDIM(Y)
      nz=NDIM(Z)
      mz=MDIM(Z)
      nt=NDIM(T)
      mt=MDIM(T)
      nR1=NDIM(R1)
      mR1=MDIM(R1)
      nR2=NDIM(R2)
      mR2=MDIM(R2)
```

```
      Write(5,2000) nx,mx,  ny,my,nz,mz,nt,mt,nr1,mr1,nr2,mr2
2000  Format('  X(',I5,' by ',I5,')'/
     1       '  Y(',I5,' by ',I5,')'/
     2       '  Z(',I5,' by ',I5,')'/
     3       '  T(',I5,' by ',I5,')'/
     1       ' R1(',I5,' by ',I5,')'/
     2       ' R2(',I5,' by ',I5,')'/)
      Go to 1
C
C                                     --- rot
70    CALL EQU(W,T)
      CALL PUSH
      Call EQU(X,W)
      Go to 1
C
C                                     --- unit, 1
80    Call PUSH
      N=IGET('Enter dimension: $')
      Call UNIT( X,N )
      Go to 1
C
C                                     --- zero, 0
90    Call PUSH
      Call PRN('Enter dimensions: $')
      Read(5,1000) N,M
1000  Format(2I6)
      Call ZERO( X,N,M )
      Go to 1
C
C                                     --- add, +
100   Call ADD( X,X,Y )
      Call POP
      Go to 1
C
C                                     --- sub, -
110   Call SUB( X,X,Y )
      Call POP
      Go to 1
C
C                                     --- scale, #
120   s=RGET('Enter scale factor: $')
      Call SCALE( X,s,X )
```

```
        Go to 1
C
C                                       --- xchg, xy
130     Call EQU( W,Y )
        Call EQU( Y,X )
        Call EQU( X,W )
        Go to 1
C
C                                       --- xz
131     Call EQU( W,Z )
        Call EQU( Z,X )
        Call EQU( X,W )
        Go to 1
C
C                                       --- xt
132     Call EQU( W,T )
        Call EQU( T,X )
        Call EQU( X,W )
        Go to 1
C
C                                       --- trn, '
140     Call TRN( X,X )
        Go to 1
C
C                                       --- mul, *
150     Call MUL( X,X,Y )
        Call POP
        Go to 1
C
C                                       --- tmul, '*
160     Call TRN( X,X )
        Call MUL( X,X,Y )
        Call POP
        Go to 1
C
C                                       --- mult, *'
170     Call MULT( X,X,Y )
        Call POP
        Go to 1
C
C                                       --- ginv, '
180     If( NDIM(X).LT.MDIM(X) ) Go to 181
```

```
        Call GINV( X,X )
        Go to 1
181     Call PRN('# of rows of X must be .GE. than # of columns$')
        Go to 1
C
C                                       --- cfft
190     Call CFFT( X(3), Y(3), X(1) )
        Go to 1
C
C                                       --- ifft
200     Call IFFT( X(3), Y(3), X(1) )
        Go to 1
C
C                                       --- magn
210     Call MAGN( X(3),Y(3),X(1) )
        Go to 1
C
C                                       --- vmax
220     nx=NDIM(X)*MDIM(X)
        s=VMAX( X(3), nx )
        Call RPRN(s,'=maximum of X$')
        Go to 1
C
C                                       --- vmin
230     nx=NDIM(X)*MDIM(X)
        s=VMIN( X(3), nx )
        Call RPRN(s,'=minimum of X$')
        Go to 1
C
C                                       --- help
240     Call Gopen(1,'EASY.HLP$')
        i=1
245     Read(1,247,END=241) (W(j),j=1,63)
247     Format(64A1)
        Write(5,242) (W(j),j=1,63)
242     Format(' ',64A1)
        i=i+1
        Go to 245
241     Endfile 1
        Go to 1
C
C                                       --- sto1
```

```
250     Call EQU(R1,X)
        Go to 1
C
C                                       --- rec1
260     Call EQU(X,R1)
        Go to 1
C
C                                       --- push
270     Call PUSH
        Go to 1
C
C                                       --- plot
280     Call prn('Paper limits (left, width ): $')
        Read(5,281) left,ibase
        Call prn('Data  limits (xmin,xmax,dx): $')
        Read(5,282) xmin,xmax,dx
        Call prn('Index limits (str,end,incr): $')
        Read(5,281) i1,i2,i3
        Call DOTARY( X(3), i1,i2,i3, xmin,xmax,dx, left, ibase )
        Go to 1
281     Format(3I6)
282     Format(3G14.7)
C
C                                       --- sto2
290     Call EQU(R1,X)
        Call EQU(R2,Y)
        Go to 1
C
C                                       --- rec2
300     Call EQU(X,R1)
        Call EQU(Y,R2)
        Go to 1
C
C                                       --- rect
310     Call RECT(X(3),Y(3),X(1))
        Go to 1
        END
C
        Subroutine PUSH
        Common /regX/ X(1)
        Common /regY/ Y(1)
        Common /regZ/ Z(1)
```

```
        Common /regT/ T(1)
        Call EQU( T,Z )
        Call EQU( Z,Y )
        Call EQU( Y,X )
        Return
        END
C
        Subroutine POP
        Common /regX/ X(1)
        Common /regY/ Y(1)
        Common /regZ/ Z(1)
        Common /regT/ T(1)
        Call EQU( Y,Z )
        Call EQU( Z,T )
        Return
        END
C
C--- Plots an ARRAY of real numbers
C
C....x(.) i1,i2,i3 (limits) real array
C....xmin,xmax,dx  virtual minimum, maximum, increment
C....left,ibase    paper left margin, paper width in dots (<1180)
C
        Subroutine DOTARY( array,i1,i2,i3,xmin,xmax,dx,left,ibase)
        Dimension array(1),iarray(27),narray(27)
        image(x)=left+INT(float(ibase)*(x-xmin)/(xmax-xmin))
C
C...Initiate printer
C
        NSP   =INT( (xmax-xmin)/dx )
        Call INIDOT
C
C...Draw the scale line
C
                x=xmin
                Do 10 i=1,NSP
                iarray(i)=image(x)
                narray(i)=image(x)
10              x=x+dx
        NSP1=NSP+1
        iarray(NSP1)=image( array(i1) )
        Call DOTS( iarray, NSP1 )
```

```
C
C...Draw the array
C
        i21=i2-1
                Do 20 i=i1,i21,i3
                iarray(1)=left
                iarray(2)= image( array(i+1) )
                Call DOTS( iarray(1),2 )
20              Continue
C
C...print the scale again
C
        Call DOTS( narray(1), NSP )
        Return
        End
C
C--- Image:  convert virtual to paper coordinates
C
C       Integer function image(x)
C       Common xmin,xmax,left,ibase
C
C...Print a series of N dots according to the values
C...of the iarray.  Limits 0:1180
C
        Subroutine DOTS( iarray,N )
        Dimension iarray(1)
C
C...sort
C
        Call SORT(iarray,N)
        If (N-1) 30,30,40
C
C...find the relative distances
C
40              i=N
41              i1=i-1
                iarray(i)=iarray(i)-iarray(i1)
                If(iarray(i).LT.0) pause Dots
                i=i-1
                If( i-1 ) 30,30,41
C
30      continue
```

```
        Do 50 i=1,N
50      Call DOT( iarray(i))
        Call HALFLN
        Return
        End
C
C...Bubble sorting routine for an integer array
C...IN PLACE sorting in ascending order
C
        Subroutine SORT( iarray,N )
        Dimension iarray(1)
C
        If (N-1) 20,20,10
10      N1=N-1
                Do 130 i=1,N1
                j1=i+1
C...FIND max
                        Do 30 j=j1,N
                        iar=iarray(i)
                        If( iarray(j).GE.iar ) Go to 30
                        iarray(i)=iarray(j)
                        iarray(j)=iar
30                      Continue
130             Continue
C
20      Return
        END
```

4. EASYPACK Command Summary

Utilities and I/O

```
        Call PACK( A, IA,N,M )
        N =  NDIM( B )
        M =  MDIM( B )
        NM=  NM2DIM( B )
        Call SETDIM( A, N, M )
        s =  ELE( B, i, j )
        Call MINP( A, 'comment$' )
        Call MPRN( A, 'comment$' )
```

```
Call ALTER( A, 'comment$' )
Call MGET( 'filename.ext$', A )
Call MSTORE( 'filename.ext$', A )
Call RDCBIN( 'filename.ext$', X, N )
Call WRCBIN( 'filename.ext$', X, N )
```

Matrix operations and functions

```
Call UNIT( A, N )
Call ZERO( A, N, M )
Call RAMP( A, N )
Call EQU( A, B )
Call ADD( A, B, C )
Call SUB( A, B, C )
Call SADD( A, B, s, C )
Call SCALE( A, s, B )
Call TRN( A, B )
Call MUL( A, B, C )
Call MULT( A, B, C )
Call QUA( A, B, C )
Call GINV( A, B )
Call PART( A, B, N1,N2,M1,M2 )
Call AUGM( A, N, M, B )
s =  XNORM1( B )
s =  XNORM( B )
```

Vector operations and functions

```
Call VEQU( X, Y, N )
Call VZERO( X, N )
Call VSCALE( X, s, N )
Call VSADD( X, s, Y, N )
s =  VDOT( Y, Z, N )
s =  VMIN( Y, N )
s =  VMAX( Y, N )
s =  VNORM1( Y, N )
s =  VNORM( Y, N )
Call CFFT( X, Y, N )
Call IFFT( X, Y, N )
Call MAGN( X, Y, N )
```

Scalar operations

```
i =  IGET('comment$')
r =  RGET('comment$')
Call SGET('comment$', a, N )
Call PRN( 'comment$' )
Call IPRN( i, 'comment$')
Call RPRN( r, 'comment$')
Call HPRN( i, 'comment$')
Call GOPEN( i, 'filename$')
a =  DEG(x)
a =  RAD(x)
a =  PI(b)
```

Alphabetical order

```
Call  ADD( A, B, C )
Call  ALTER( A, 'comment$' )
Call  AUGM( A, N, M, B )
Call  CFFT( X, Y, N )
 a =  DEG(x)
 s =  ELE( B, i, j )
Call  EQU( A, B )
Call  GINV( A, B )
Call  GOPEN( i, 'filename$')
Call  HPRN( i, 'comment$')
Call  IFFT( X, Y, N )
 i =  IGET('comment$')
Call  IPRN( i, 'comment$')
Call  MAGN( X, Y, N )
 M =  MDIM( B )
Call  MGET( 'filename.ext$', A )
Call  MINP( A, 'comment$' )
Call  MPRN( A, 'comment$' )
Call  MSTORE( 'filename.ext$', A )
Call  MUL( A, B, C )
Call  MULT( A, B, C )
 N =  NDIM( B )
NM =  NM2DIM( B )
Call  PACK( A, IA,N,M )
Call  PART( A, B, N1,N2,M1,M2 )
```

```
 a =  PI(b)
Call  PRN( 'comment$' )
Call  QUA( A, B, C )
 a =  RAD(x)
Call  RAMP( A, N )
Call  RDCBIN( 'filename.ext$', X, N )
 r =  RGET('comment$')
Call  RPRN( r, 'comment$')
Call  SADD( A, B, s, C )
Call  SCALE( A, s, B )
Call  SETDIM( A, N, M )
Call  SGET('comment$', a, N )
Call  SUB( A, B, C )
Call  TRN( A, B )
Call  UNIT( A, N )
Call  VEQU( X, Y, N )
 s =  VDOT( Y, Z, N )
 s =  VMAX( Y, N )
 s =  VMIN( Y, N )
 s =  VNORM1( Y, N )
 s =  VNORM( Y, N )
Call  VSCALE( X, s, N )
Call  VSADD( X, s, Y, N )
Call  VZERO( X, N )
Call  WRCBIN( 'filename.ext$', X, N )
 s =  XNORM1( B )
 s =  XNORM( B )
Call  ZERO( A, N, M )
```

5. MATRIX REVERSE POLISH CALCULATOR: Command Summary

Matrix registers: x (top of stack), y, z, t (bottom)
Matrix buffers: R1, R2

f(x) ⟶x	f(x,y)⟶(x,y)	?? ⟶x push	f(x,y)⟶x pop
alter	cfft	zero 0	add +
scale #	ifft	unit 1	sub -
trn '	magn	minp in	mul *
ginv "	rect	mget get	tmul '*
			mult *'

exchanges	display	utilities
xy xchg	x mprn	mstore save
xz	y	map
xt	z	help
rot	t	
sto1	max	
rec2	min	
sto2	plot	
rec2		
push		

BIBLIOGRAPHY

1. Articles and Dissertations
2. Books and Texts

Bibliography

This bibliography is divided into two sections; the first contains the reviewed articles and dissertations and the second contains the reviewed books and texts in the areas of

random sampling,
maximum entropy,
signal processing, and
estimation-identification.

Both sections have been ordered alphabetically. For convenience, at the end of the text references the Library of Congress Classification Call number (or the Dewey Decimal number for older texts) is usually added inside angle brackets. Single letters inside angle brackets are used for personal filing and they should be ignored.

1. Articles and Dissertations

Astrom K. J. and P. Evkhoff, 'System identification. A survey,' Automatica, vol. 7., pp. 123-162, 1971.

Auslander D. M., Y. Takahashi and M. Tomizuka, 'Direct digital process control: Practice and algorithms for microcomputer application,' Proc. of the IEEE, vol. 66, Feb. 1978.

Bar-Shalom Yaakov, 'Optimal Simultaneous State Estimation and parameter identification in linear discrete time systems,' IEEE Trans. on Automatic Control, vol. AC-17, pp. 308-309, June 1972. <C>

Balakrishnan A. V., 'On the problem of time jitter in sampling,' IRE Trans. on Information Theory, vol. IT-8, pp. 226-236, April 1962.

Burg J. P., 'Maximun entropy spectral analysis,' 37th Meeting of the

Society of Exploration Geophysicists, Oklahoma, 1967. <M>

Burg J. P., 'General principles for estimation of covariance matrices,' Unpublished, Sept. 28, 1973. <M>

Buetler F. J. and O. A. Z. Leneman, 'Random sampling of random processes: stationary point processes,' Inform. and Control, vol. 9, pp. 325-346, 1966.

Buetler F. J., 'Alias free randomly timed sampling of stochastic processes,' IEEE Trans. on Information Theory, vol. IT-16, pp. 147-152, March 1970. <C>

Buetler F. J., 'Error free recovery of signals from irregularly spaced samples,' SIAM, vol. 8, pp. 328-335, July 1966. <C>

Clark Ronald R., 'Meteor Wind Measurements at Durham N.H.,' Journal of Atmospheric Sciences, vol. 32, pp. 1689-1693, Sept 1975. <R>

Conant R.C. and W.R. Ashby, 'Every good regulator of a system must be a model of that system,' International Journal of Systems Science, vol. 1, No. 2, pp. 89-97, 1970. <4>

Carlson Neal A., 'Fast Triangular Formulation of the square root filter,' AIAA Journal, vol. 11, no. 9, pp. 1259-1265, Sept. 1973. <C>

Chow Joseph C., 'On estimation of the moving average parameters,' IEEE Trans. on Automatic Control, vol. AC-17, pp. 268-269, April 1972. <C>

Ciscato D. and L. Mariani, 'On increasing sampling efficiency by adaptive sampling,' IEEE Trans. on Automatic Control, vol. AC-12, pp. 318, June 1967. <R>

Daniel Willie L. and W. E. Bennett, 'Improvement of system response by reducing quantization error through adaptive sampling,' IEEE Trans. Automatic Control, vol. AC-19, pp. 598-599, October 1974. <R>

Dorf R., M. C. Farren, C. A. Phillips, 'Adaptive sampling for sampled data control systems,' IEEE Trans. on Automatic Control, vol. AC-7, pp. 38-47, Jan. 1962.

Galiana F. D. and E. Handschin and A. R. Fiechter, 'Identification of stochastic electric load models from physical data,' IEEE Trans. Automatic Control, vol. AC-19, pp. 887-898, Dec. 1974. <C>

Fougere P. F., 'A solution to the problem of spontaneous line splitting in maximum entropy power spectrum analysis,' Annual AGU Meeting, San Francisco, CA, 1975. <M>

Gerardi F. R., 'Application of Mellin and Hankel Transforms to networks with time varying parameters,' IRE Trans. on Circuit Theory, vol. CT-6, pp. 197-207, 1959. <R>

Harrison S. R., 'Digital Filters,' Ph.D. Dissertation, Purdue Univ., 1970. <P>

Holm S. and J. M. Hovem, 'Estimation of scalar ocean wave spectra by the maximum entropy method,' IEEE Journal of Oceanic Engin., vol. OE-4, pp. 76-83, July 1979. <M>

Hoskins W. D. and D. J. Walton, 'A faster method of computing the square root of a matrix,' Trans. on Automatic Control, AC-23, pp. 494-495, June 1978.

Jerri A. J., 'The Shannon sampling theorem - Its various extensions and applications: A Tutorial review,' Proceedings of the IEEE, vol. 65, pp. 1565-1595, Nov. 1977. <C>

Jury E. I., 'Sampling schemes in sampled data control systems,' IRE Trans. on Automatic Control, vol. AC-6, pp. 86-88, Feb. 1961.

Jury E. I. and F. J. Mullin, 'The analysis of sampled-data control

systems with a periodically time-varying Sampling rate,' IRE Trans. on Automatic Control, pp. 15-25, May 1959. <C>

Kalman R. E., 'A new approach to linear filtering and prediction problems,' Trans. ASME, vol. 82D, pp. 35-45, March 1960.

Kalman R. E., 'Analysis and synthesis of linear systems operating on randomly sampled data,' Ph.D. dissertation, Dept. of Elect. Engin., Columbia University, New York, 1957.

Kalman R. E. and J. E. Bertram, 'A unified approach to the theory of sampling systems,' J. Franklin Inst., vol. 267, pp. 405-436, May 1959.

Kontopidis G. D., 'Nonuniformly sampled systems,' Master's Thesis, University of New Hampshire, 1976.

Kontopidis G. D., 'Models of man-machine tasks based on nonuniformly sampled systems,' 2nd Intern. Confer. on Information Sciences and Systems, Patras Greece, July 1979.

Kontopids G. D., 'Stochastically sampled systems: A state approach,' (unpublished) U.N.H. 1979.

Kontopidis G. D., 'Noniterative algorithms to compute expressions involving the matrix exponential,' (unpublished), U.N.H. 1980.

Kontopidis G., F. Glanz and D. Limbert, 'Models for Human Operators based on nonuniformly sampled systems,' Proc. 15th Annual Conf. on Manual Control, Dayton Ohio, March 1979.

Kontopidis G., D. Limbert and F. Glanz, 'A study of nonuniformly sampled systems,' Intern. Confer. on Cybernetics and Society, Denver Colorado, Oct. 1979.

Kontopidis G., D. Limbert and F. Glanz, 'Computer controlled systems using multiplexed I/O,' IECI'80 Intern. Confer. on Mini- and Micro

Computer Applications, Philadelphia, March 1980.

Kaminski P. G., A. E. Bryson and S. F. Schmidt, 'Discrete square root filtering: A survey of cureent techniques,' IEEE Trans. on Automatic Control, vol. AC-16, pp. 727-735, Dec 1971.

Kwatny H. G., 'A note on stochastic approximations algorithms in system identification,' IEEE Trans. on Automatic Control, vol. AC-17, pp. 571-572, August 1972.

Lacoss R. T., 'Data adaptive spectral analysis methods,' Geophysics, vol. 36, pp. 661-675, August 1971. <M>

Lainiotis Demetrios G., 'Optimal adaptive estimation: Structure and parameter adaptation,' IEEE Trans. on Automatic Control, vol. AC-16, pp. 160-170, April 1971.

Landau I. D., 'A survey of model reference adaptive techniques (Theory and Applications),' Automatica, vol. 10, July 1974.

Lawson F. R., 'Signal processing with sampled correlators,' Ph.D. Dissertation, Purdue, 1970. <P>

Linden D. A., Adramson N. J., 'A generalization of the sampling theorem,' Inform. Control, vol. 3, pp. 26-31, March 1960.

Liu B. and T. P. Stanley, 'Error bounds for jittered sampling,' IEEE Trans. on Automatic Control, vol. AC-10, pp. 449-454, October 1965. <C>

Leneman O. A. Z., 'Random sampling of random processes: mean-square behavior of a first-order closed-loop system,' IEEE Trans. on Automatic Control, pp. 429-432, August 1968. <C>

Leneman O. A. Z., 'A note on the mean square behavior of a first order random sampling system,' IEEE Trans. on Automatic Control, pp. 452-453, Aug 1968. <C>

Leneman O. A. Z., 'On some results in random pulse trains,' IEEE Trans. on Automatic Control, vol. AC-11, p. 331, April 1966.

Leneman O. A. Z. and Lewis J. B., 'Random sampling of random processes: Mean square comparison of various interpolators,' IEEE Trans. on Automatic Control, vol. AC-11, pp. 396-403, July 1966

Lui M. C., 'Spectral estimation of continous parameter processes from randomly spaced observations,' Ph. D. Dissertation, University of California at San Diego, 1974.

Magill D. T., 'Optimal adaptive estimation of sampled stochastic processes,' IEEE Trans. on Automatic Control, vol. AC-10, pp. 434-439, October 1965.

Masry E., 'Alias free sampling: An alternative conceptualization and its applications,' IEEE Trans. on Information Theory, vol IT-24, pp. 317-324, May 1978. <C>

Masry E. and Ming-Chuan C. Lui, 'Discrete time spectral estimation of continuous parameter processes. A new consistent estimate,' IEEE Trans. on Information Theory, vol. IT-22, pp. 298-312, May 1976. <C>

Mehra Raman K., 'Optimal input signals for parameter estimation in dynamic systems. A survey,' IEEE Trans. on Automatic Control, vol. AC-19, pp. 798-809, Dec. 1974.

Mehra Raman K., 'Approaches to adaptive filtering,' IEEE Trans. Automatic Control, vol. AC-17, pp. 693-698, August 1972.

Mitchell J. R. and W. L. McDaniel, 'Adaptive sampling techniques,' IEEE Trans. Automatic Control, vol. AC-19, pp. 200-202, April 1969.

Middleton D. and R.Esposito, 'Simultaneous optimum detection and estimation of signals in noise,' IEEE Trans. Information Theory, vol.

IT-14, pp. 434-444, May 1968.

Morf M., A. Vieira, D. T. Lee and T. Kailath, 'Reqursive multichannel maximum entropy spectral estimation,' IEEE Trans. on Geoscience Electronics, vol. GE-16, pp. 85-94, April 1978. <M>

Nahi N. E., 'Optimal recursive estimation with Uncertain Observation,' IEEE Trans. Information Theory, IT-15, pp. 457-462, July 1969. <C>

Otnes R. K., 'Instability thresholds in digital filters,' Ph. D. Dissertation, University of California, Los Angeles, Jan. 1970.

Palosky P., 'Generating discrete colored noise from discrete white noise,' IEEE Trans. on Automatic Control, vol. AC-11, pp. 148-149, January 1966.

Palmer E., 'Interrupted monitoring of a stochastic process,' Proc. 13-th Annual Conf. on Manual Control, pp. 237-245, June 1977.

Papoulis Athanasios, 'New results in sampling theory,' Hawaii Intern. Conf. on System Sciences, Jan. 1968.

Pearson A. E. 'Finite time interval linear system identification without initial state estimation,' Automatica, vol. 12, pp. 577-587, 1976.

Rothschild D. and A. Jameson, 'Comparison of four numerical algorithms for solving the Liapunov matrix equation,' Intern. Journal of Control, vol. 11, pp. 181-198, 1970.

Shapiro H. S. and R. A. Silverman, 'Alias free sampling of random noise,' SIAM vol. 8, pp. 225-248, June 1960. <C>

Sandoz D. J. and B. H. Swanick, 'A recursive least squares approach to the adaptive control problems,' Intern. Journal of Control, vol. 16, pp. 243-258, 1972.

Scott Paul F., 'Estimation of correlation functions and spectra from randomly spaced data,' Record of 1977 IEEE ICASSP, pp. 70-73, 1977.

Smith Michael J., 'An evaluation of adaptive sampling,' IEEE trans. on Automatic Control, vol. AC-16, pp. 282-284, June 1971.

Smith P. G., 'Numerical solution of the matrix equation AX+XA'+B=0,' IEEE Trans. on Automatic Control, vol. 16, pp. 278-279, Jan. 1971.

Smylie D. E., G. K. Clarke and T. J. Ulrych, 'Analysis of irregularities in the earth's rotation,' Methods in Computational Physics, vol. 13, pp. 391-430, Academic Press, New York, 1973. <M>

Sriyananda H., 'A simple method for the control of divergence in Kalman filter algorithms,' Intern. Journal of Control, vol. 16, pp. 1101-1106, 1972.

Tomovic R. and G. A. Bekey, 'Adaptive sampling based on amplitude sensitivity,' IEEE Trans. on Automatic Control, vol. AC-11, pp. 282-284, April 1966.

Tomovic R. and G. A. Bekey, 'Sensitivity of discrete systems to variation of sampling interval,' IEEE Trans. on Automatic Control, vol. AC-11, pp. 284-287, April 1966.

Ulrych T. J. and T. N. Bishop, 'Maximum entropy spectral analysis and autoregressive decomposition,' Reviews of Geophysics and Space Physics, vol. 13, no. 1, pp. 183-200, Feb. 1975. <M>

Van-Loan Charles F., 'Computing integrals involving the matrix exponential,' IEEE Trans. on Automatic Control, vol. AC-23, pp. 395-404, June 1978.

Van-Ness James E., 'Inverse iteration method for finding eigenvectors,' IEEE Trans. on Automatic Control, vol. AC-14, pp. 63-66, Feb. 1969.

Vaughan David R., 'A negative exponential solution for the matrix Riccati equation,' IEEE Trans. on Automatic Control, vol. AC-14, pp. 72-75, Feb. 1969.

Will P. M., 'Variable Frequency Sampling,' IRE Trans. on Automatic Control, vol. AC-7, p. 126, Oct. 1962.

Yahagi Tahashi, 'A method for adaptive control using a minicomputer,' IEEE Trans. on Industrial Electr. and Control Instrum., vol. IECI-26, Feb. 1979.

Yen J. L., 'On nonuniform sampling of bandwidth limited signals,' IRE Trans. Circuit Theory, vol. CT-3, pp. 251-257, 1956. <C>

Zadeh Lotfi A., 'Frequency analysis of variable networks,' IRE pp. 291-297, March 1950.

Zadeh L. A., 'From circuit to system theory,' Proceedings IRE, vol. 50, pp. 856-865, 1962.

2. Books and Texts

Ahlberg J. H., E. N. Nilson and J. L. Walsh, The theory of Splines and Their Applications., Academic Press, Inc., 1967. <P>

Aoki M, Optimization of Stochastic Systems. Academic Press, New York, 1967. <QA402.3 A58>

Astrom Karl J., Introduction to Stochastic Control Theory. New York: Academic Press, Inc., 1970. <P>

Bellman Richard, Introduction to the Mathematical Theory of Control Processes. vol. I and II, Academic Press Inc., New York, 1967.

Bendat J. S. and A. G. Piersol, Random Data: Analysis and Measurement

Procedures. Wiley-Interscience, New York, 1971.

Box G. E. and G. M. Jenkins, Time Series Analysis Forcasting and Control. Holden-Day, San Francisco, 1970. <QA280 B67>

Brogan William L., Modern Control Theory. Quantum Publishers Inc., New York, 1974. <P>

Cadzow J. A., H. R. Martens, Discrete Time and Computer Control Systems. Prentice Hall, Englewood Cliffs, New Jersey, 1963

D' Angelo Henry, Linear Time Varying Systems: Analysis and Synthesis. Boston: Allyn and Bacon Inc., 1970.

Dongarra J. J., C. B. Moler, J. R. Bunch and G. W. Stewart, LINPACK User's Guide. SIAM, Philadelphia, 1979. <P>

Durling Allen and Donald Childers, Digital Filtering and Signal Processing. West Publishing Company, St. Paul, 1975. <P>

Freeman Herbert, Discrete Time Systems. John Willey Inc., New York, 1964.

Gihman I. I. and A. V. Skorohod, Stochastic Differential Equations. Springer-Verlag, New York, 1972. <QA274 G5513>

Goodwin G. C. and R. L. Payne, Dynamic System Identification: Experiment Design and Data Analysis. Academic Press, New York, 1977. <QA402 G66>

Hsia T. C., System Identification. Massachusetts: Lexington Books, 1977.

Kalman R. E., P. L. Falb and M. A. Arbib, Topics in Mathematical System Theory. McGraw Hill Book Company, New York, 1969.

King R. W., The Theory of Linear Antennas. Harvard University Press, Cambridge, Mass. 1956.

Kuo B. C., Discrete Data Control Systems. Engelwood Cliffs, Prentice Hall, New Jersey, 1970.

Kuo B. C., Digital Control Systems. SRL Publishing Company, Champaign Illinois, 1977. <TJ216 K812>

Kwakernaak H., R. Sivan, Linear Optimal Control Systems. Wiley Interscience, 1972. <QA 402.3 K89>

Lee Robert C. K., Optimal Estimation Identification and Control. Research Monograph No. 28, The MIT Press, Cambridge, Massachusetts, 1964.

LePage W. R., Complex Variables and Laplace Transform for Engineers. McGraw Hill Book Company, New York, 1961. <QA 432 L32>

Luenberger David G., Optimization by Vector Space Methods. John Wiley and Sons, Inc., New York, 1969.

Lusternik L. A. and V. J. Sobolev, Elements of Functional Analysis. Hindustan Publishing Company, India, Delphi 1961. <517.5 L783>

Mehra Raman K. and Dimitri G. Lainiotis, System Identification: Advances and Case Studies. Academic Press, Inc., New York, 1976.

Melsa James L., Stephen K. Jones, Computer Programs for Computational Assistance in the Study of Linear Control Theory. McGraw Hill Book Company, New York, 1973. <QA402.3 M386>

Melsa James L., and Donald G. Schultz, State Function and Linear Control Systems. McGraw Hill Book Company, New York, 1967.

Morrison N., Introduction to Sequential Smoothing and Prediction. McGraw Hill Book Company, New York, 1969.

Nahi N. E., Estimation Theory and Aplication. New York: Wiley, 1969.

Papoulis Athanasios, Probability, Random Variables and Stochastic Processes. McGraw Hill Book Company, New York, 1965. <P>

Papoulis Athanasios, Signal Analysis. McGraw Hill Book Company, New York 1977. <P>

Pressman R. S. and J. E. Williams, Numerical Control and Computer Aided Manufacturing. John Wiley and Sons, N. J. 1977.

Polak E., Computational Methods in Optimization. Academic Press, New York, 1976. <P>

Porter William A., Modern Foundations of System Engineering. McMillan Club, New York, 1971.

Ralston A. and P. Rabinowitz, A First Course in Numerical Analysis. McGraw Hill, New York, 1978. <QA297 R3> <P>

Rabiner Lawrence R., Theory and Application of Digital Signal Processing. Englewood Cliffs, Prentice Hall, New Jersey, 1975. <P>

Reza F. M., An Introduction to Information Theory. McGraw Hill Book Company, New York, 1961. <QA 360 R43>

Robinson Enders A., Multichannel Time Series Analysis with Digital Computer Programs. Holden Day, San Francisco, 1967.

Robinson Enders A. and Sven Treitel, Geophysical Signal Analysis. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1980. <TN269 R55>

Rudin W., Real and Complex Analysis. McGraw Hill Book Company, New York, 1974. <QA300 R82> <P>

Rudin W., Functional Analysis. McGraw Hill Book Company, Newy York,

1973. <QA320 R83> <P>

Sage Andrew P., and James L. Melsa, Estimation Theory with Applications to Communications and Control. McGraw Hill Book Company, New York, 1971. <P>

Sage Andrew P., and James L. Melsa, System Identification. McGraw Hill Book Company, New York, 1971.

Sage Andrew P., and Chelsea C. White, Optimum Systems Control, 2nd Edition, Englewood Cliffs, Prentice Hall, New Jersey, 1977. <P>

Schweppe F. C., Uncertain Dynamic Systems. Prentice Hall, Englewood Cliffs, New Jersey, 1973. <P>

Sokolnikoff I. S. and R. M. Redheffer, Mathematics of Physics and Moderm Engineering. McGraw Hill Book Company, New York, 1958. <530.15 S683>

Solodovnikov V. V., Statistical Dynamics of Linear Automatic Control Systems. D. Van Nostrand Company, Ltd., London, 1965. <TJ213 S5643>

Takahashi Yasundo, M. J. Rabins, D. M. Auslander, Control and Dynamic Systems. Addison-Wiley Publishing Company, New York, 1970.

Wozencraft J. M., and Irwin M. Jacobs, Principles of Communication Engineering. John Willey and Sons, Inc., 1965. <P>

Zadeh L. A. and C. A. Desoer, Linear System Theory. The State Space Approach. McGraw Hill, New York, 1963. <517 Z17>