RL-TR-91-93
Final Technical Report
July 1991

RESEARCH AND DEVELOPMENT FOR DIGITAL VOICE PROCESSING (DVP)

ARCON Corporation

J.D. Tardelli, P.A. LaFollette, C.M. Walter, J. LeBlanc,
P.D. Gatewood, D. Colbert

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-91-93 has been reviewed and is approved for publication.

APPROVED:

LUIGI SPAGNUOLO
Project Engineer

APPROVED:

JOHN K. SCHINDLER
Director of Electromagnetics

FOR THE COMMANDER:

JAMES W. HYDE III
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL(EEV) Hanscom AFB MA 01731-5000. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1991	Final Feb 86 - Jun 88

4. TITLE AND SUBTITLE
RESEARCH AND DEVELOPMENT FOR DIGITAL VOICE PROCESSING (DVP)

5. FUNDING NUMBERS
C - F19628-86-C-0057
PE - 33401F
PR - 7820
TA - 03
WU - 53

6. AUTHOR(S)
J. D. Tardelli, P. A. LaFollette, C. M. Walter, J. LeBlanc, P. D. Gatewood, D. Colbert

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ARCON Corporation
260 Bear Hill Road
Waltham MA 02154

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (EEV)
Hanscom AFB MA 01731-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-91-93

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Luigi Spagnuolo/EEV/(617) 478-4249
NOTE: Rome Laboratory (formerly Rome Air Development Center/RADC)

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This technical report covers a variety of topics and research areas. An update of the Canonical Coordinate (CC) Transformation process for digital speech compression based on non-Euclidean error minimization criteria is given. A method of transforming imperical filter sets to unitary error metrics is introduced and a sample error metric is developed. Research into frequency shift invarient transformation is presented and its utility for developing CC error metrics is discussed. A study into the relationship between CC analysis and Linear Predictive Coding is presented. An improved AP120-B implementation of the CC algorithm is given along with a fully parallel CC implementation on a systolic array processor. The latest developments in processor hardware, operating systems, and program development software at RADC/EEV are documented. Custom additions to the Interactive Laboratory System (ILS) analysis and display package are covered along with its relationship to the Speech Data Base Library used at RADC/EEV. Data Base input/output programs, analysis tools, search/sort programs are all presenced. The status of a communicability testbed system at the Speech Processing Facility is presented.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Digital Voice Processing, Speech Compression, Vocoder Signal Processing, Canonical Coordinate Analysis, Principle Component Analysis, Non-Euclidian Error Minimization, (continued)	148
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	U/L

Block 14 Continued

Error Metric, Speech Processing Facility, Linear Predictive Coding Speech Data Base, Communicability Testing, ILS, Array Processor Programs, Systolic Array Processor, Channel Error Simulation, Channel Delay Simulation, Digital Speech I/O.

CONTENTS

CHAPTER 1 OVERVIEW

1.1 INTRODUCTION 1-1
1.2 OTHER CONTRACT TASKS 1-2
1.2.1 Voice Processor Intelligibility Testing 1-2
1.2.2 System Performance Evaluation 1-2
1.2.3 DoD Digital Voice Processing Consortium Support 1-3
1.3 ACKNOWLEDGMENTS 1-3

CHAPTER 2 CANONICAL COORDINATE BASED DATA COMPRESSION

2.1 CANONICAL COORDINATE BASED VOCODER ALGORITHM DEVELOPMENT EFFORTS 2-2
2.1.1 Structural Characteristics Of Two Canonical Coordinate Techniques 2-3
2.1.2 Updated Version Of The PCC Vocoder Algorithm 2-3
2.2 FREQUENCY SHIFT INVARIANCE / ETA-TRANSFORM STUDY 2-10
2.2.1 Continuous Case 2-11
2.2.2 Discrete Case 2-12
2.2.3 Eta Transform Software 2-19
2.3 STATISTICALLY BASED ERROR METRICS 2-21
2.3.1 Sources Of Data And Covariance Estimation Procedures 2-21
2.3.2 Conclusions 2-26
2.4 APPLICATION OF A MODULAR GENERATING FUNCTION APPROACH TO ERROR METRIC DEFINITION 2-34
2.4.1 Specification of an Error Metric K and Associated Eigenvector/Eigenvalue Matrices V and Γ Using Hermitian Generating Function Procedures 2-35
2.4.2 Special Characteristics Of The Generating Function Based Differential Operator $\delta_G(\Gamma)$ 2-36
2.4.3 Explicit Characterization Of $\delta_G(\Gamma)$ And $\delta_G(\delta_G(\Gamma))$ 2-37
2.4.4 Example Of Construction Of V When G Corresponds To A Family Of Overlapping Rectangular "Box-Car" Filters 2-38
2.5 CANONICAL COORDINATE APPROACH TO INTEROPERABILITY 2-40
2.5.1 Canonical Coordinate Background 2-40

2.5.2 AR/LPC Specification in Terms of a 1-Dimensional Projection of an N-Dimensional Canonical Coordinate Process Based on Triangular D^{-1}, Spectral Whitening $\Lambda=I$ And $J=R^{-1}$ 2-43
2.5.3 Determination Of Predictor Coefficient Vector $\tilde{A}$ And Reflection Coefficient Vector $\tilde{b}$ from Canonical Coordinate Equations for the AR/LPC Situation 2-45
2.5.4 Characterization Of Extended Predictor Coefficient Vector A in Terms of Both Time Domain Matrices J and Frequency Domain Metric K . . . 2-46

CHAPTER 3 ALGORITHM RESEARCH AND IMPLEMENTATION

3.1 CANONICAL COORDINATE ALGORITHM AP-120B IMPLEMENTATIONS 3-1
3.1.1 CC Analysis Improvements 3-2
3.1.2 Intermediate Canonical Coordinate File Structure 3-3
3.1.3 CC Synthesis Improvements 3-4
3.1.4 CC Analysis Operational Instructions 3-5
3.1.5 CC Synthesis Operational Instructions 3-6
3.1.6 Further Improvements 3-7
3.2 CANONICAL COORDINATE ALGORITHM SAP IMPLEMENTATIONS 3-7
3.2.1 Specific Algorithms Used In The SAP Implementation 3-8
3.3 CANONICAL COORDINATE CODING ALGORITHMS 3-12
3.3.1 Program EHISTX 3-13
3.3.2 Program EAHST 3-14
3.3.3 Program CCHIST 3-15
3.4 RADC/EEV LPC-10E IMPLEMENTATIONS 3-16
3.4.1 Differences Between Version 49 And Version 52 3-17
3.4.2 Porting Of LPC10E Code To The PDP-11 3-18

CHAPTER 4 RADC/EEV SPEECH PROCESSING FACILITY COMPUTER SYSTEM

4.1 PDP-11/44 HARDWARE (BLDG. 1120) 4-1
4.1.1 MAP-300 Arithmetic Processor 4-6
4.1.2 AP-120B Array Processor Hardware 4-6
4.1.3 Systolic Array Processor Hardware 4-6
4.2 PDP-11/34 HARDWARE (BLDG. 1120) 4-11
4.2.1 MAP-300 Array Processors 4-14
4.3 PDP-11/34 HARDWARE (BLDG. 1124) 4-14
4.4 PDP 11/44 SYSTEM SOFTWARE (BLDG. 1120) 4-16
4.4.1 PDP 11/44 Sysgen 4-16
4.4.2 Data Transfers 4-17
4.4.3 User Enhancements 4-18
4.5 PDP 11/34 SYSTEM SOFTWARE (BLDG. 1120) 4-18
4.6 PDP 11/34 SYSTEM SOFTWARE (BLDG. 1124) 4-19

4.7 SYSTOLIC ARRAY PROCESSOR PROGRAM DEVELOPMENT SOFTWARE 4-19
4.7.1 SAP Microcode Interpreter - SAI 4-21
4.7.2 SAP Program Compiler And Executive - SAC & SAXM 4-22
4.7.3 SAP Macro Preprocessor - PREP 4-23
4.7.4 Further Modifications 4-25

CHAPTER 5 COMPUTER DATA BASES AND ANALYSIS TOOLS

5.1 INTRODUCTION 5-1
5.2 RADC/EEV COMPUTER DATA BASES 5-2
5.2.1 Speech Data Base 5-2
5.2.2 PC Data Bases 5-7
5.3 SPEECH DATA BASE ANALYSIS AND EVALUATION TOOLS . . 5-7
5.3.1 Speech DB Input/Output 5-7
5.3.2 Speech DB Retrieval And Sorting 5-12
5.3.3 Interactive Laboratory System 5-14
5.3.4 Error Metric DB Display 5-23
5.3.5 Speech DB Utility Routines 5-25
5.4 PC DB TOOLS 5-26

CHAPTER 6 COMMUNICABILITY SYSTEM

6.1 RADC COMMUNICABILITY FACILITY 6-1
6.1.1 PDP-11/34 - MAP-300 Processor 6-3
6.1.2 Other Processors 6-3
6.1.3 Operational Environment Simulations 6-4
6.1.4 System Verification 6-7

REFERENCES R-1

TABLES

Number Title Page

2.1 Example Set of Proportionally Spaced Frequencies 2-13
2.2 AP120B Software for Use in Eta Transforms 2-20
4.1 PDP 11/44 RSX-11M,V3.2 Hardware 4-2
4.2 PDP 11/34 (BLDG. 1120) RSX-11M,V4.0 Hardware . . 4-12
4.3 PDP 11/34 (BLDG. 1124) RSX-11M,V3.2 Hardware . . 4-15
5.1 FILE HEADER BLOCK FORMAT SPEECH DB AND ILS DATA FILES (SEPT 1988) 5-3
5.2 Custom ILS Programs 5-15

FIGURES

Number	Title	Page
2.1	Canonical Coordinate Technique via Correlation Matrix Diagonalization and Intermediate Principal Components	2-4
2.2	Canonical Coordinate Technique via Error Metric Diagonalization and Intermediate Pseudo-Canonical Coordinates	2-5
2.3	Discrete Eta-Transforms of Test Signals	2-15
2.4	Discrete Eta-Transforms with Magnitude Step Omitted	2-16
2.5	Discrete Eta-Transforms with Direct DFT Evaluation (i.e. No Interpolation)	2-17
2.6	Discrete Eta-Transforms with Exact Frequency Ratios	2-18
2.7	Cannonical Coordinate Methods	2-22
2.8	Long Term Spectral Distribution of DRT Wordlist CH307A	2-23
2.9	Long Term Spectral Distribution of DAM Sentence List Speaker CH, File QSET1.DAM	2-23
2.10	Correlation Matrix and Eigenvalue Characteristics White Gaussian Noise, DFT Magnitude, Linear Frequency	2-25
2.11	Correlation Matrix and Eigenvalue Characteristics Speech, DFT, Linear Frequency	2-27
2.12	Correlation Matrix and Eigenvalue Characteristics Speech, DFT, Log Frequency	2-28
2.13	Correlation Matrix and Eigenvalue Characteristics Speech, Magnitude DFT, Linear Frequency	2-29
2.14	Correlation Matrix and Eigenvalue Characteristics Speech, Magnitude DFT, Log Frequency	2-30
2.15	Correlation Matrix and Eigenvalue Characteristics Speech, Log Magnitude DFT, Linear Frequency	2-31
2.16	Correlation Matrix and Eigenvalue Characteristics Speech, Log Magnitude DFT, Log Frequency	2-32
2.17	Correlation Matrix and Eigenvalue Characteristics Speech, Phase DFT, Linear Frequency	2-33
2.18	Digital Box-Car Filter (Real Components)	2-39
2.19	Canonical Coordinate Technique via Error Metric Diagonalization Applied to Spectral Whitening Procedures of the AR/LPC Type	2-41

FIGURES (Continued)

Number	Title	Page
3.1	Pseudo-Canonical Coordinate Compression	3-9
3.2	Block Diagram of Systolic Algorithm	3-9
4.1	PDP 11/44 UNIBUS Structure	4-5
4.2	PDP 11/44 Configuration	4-5
4.3	Systolic Array Processor Modifications	4-10
4.4	PDP-11/34 (BLDG. 1120) UNIBUS Structure	4-11
4.5	PDP-11/34 UNIBUS Structure	4-14
4.6	SAP Program Development Software	4-20
5.1	FLT Filter Characteristics	5-6
5.2	Line Spectral Pair Display	5-20
5.3	Formant Filter Cascade	5-21
5.4	3-D EM Vector Display	5-24
5.5	3-D EM Grayscale Display	5-25
6.1	RADC/EEV Communicability Facility	6-2
6.2	Communicability DRT Profiles	6-8

CHAPTER 1

OVERVIEW

1.1 INTRODUCTION

This is the Final Technical Report describing work performed for the U.S. Air Force Systems Command RADC/EEV Speech Processing Facility at Hanscom AFB, MA. This work was performed under contract number F19628-86-C-0057 during the period 15-February-1986 through 31-July-1988. This effort is a continuation and extension of work performed under previous contracts with RADC/EEV and reported elsewhere [Ref. 1 - 10].

This technical report will cover a number of algorithm research and development topics. An update of research on the use of the Canonical Coordinate (CC) Transformation process for digital speech compression based on non-Euclidean error minimization criteria will be presented. The relationship between CC analysis and Linear Predictive Coding will be covered. A method of transforming empirical filter sets to unitary error metrics will be introduced and a sample error metric will be discussed including test results from an experimental vocoder using this error metric. Development work in the area of quantization and coding of CC parameters will be presented.

The report will also cover algorithm implementations completed during this contract period and work in several support areas. The implementation of a faster version of the CC algorithm operating on the AP-120B array processor will be shown. An implementation of this algorithm on a true systolic processor along with a software development system for this processor will be covered. The latest developments in processor hardware, operating systems and program development software at RADC/EEV will be documented. Additions to the Interactive Laboratory System (ILS) analysis and display package will be covered along with its relationship to the Speech Data Base library used at RADC/EEV. The latest modifications to the communicability system at the Speech Processing Facility will be described.

Numerous software packages are discussed within this report. Source code for all programs developed under this contract is available to authorized users of the RADC/EEV Speech Processing Facility through reference to the virtual disk REPT88 and directory file [200,200]REPT88.DIR. Some software has been segregated by topic on the virtual disks LPCXE (LPC-10E programs), SAP (Systolic Array Processor

program development routines), MAP300 (MAP300 Array Processor routines), IL (ILS standard and custom programs) and the physical disk SPEECH (Speech Data Base). Most PDP-11/44 program tasks can be run from the system device DR: at the special UIC [1,54] (i.e. "$").

1.2 OTHER CONTRACT TASKS

During the course of this contract numerous requirements were met that do not lend themselves to a descriptive section in this technical report. Several of these work areas are covered in the following paragraphs for completeness.

1.2.1 Voice Processor Intelligibility Testing

ARCON provided the training, staff and supervision for the RADC/EEV in-house voice communication systems test and evaluation program. This program utilizes the Diagnostic Rhyme Test (DRT) as a measure of system intelligibility. An average of twice weekly DRT listener sessions were conducted and the data was collected, scored, analyzed, reported on and stored in the DRT Data Base. All software and data for this system was maintained throughout the period of this contract. Evaluation of test data for both in-house research staff and other government users of the RADC/EEV facility was also provided. Numerous in-house DRT tapes were prepared for system evaluation at both the EEV facility and an independent contractor (DYNASTAT Corp., Austin, TX).

Several new analysis tools have been designed and implemented by ARCON. The ability to measure the intelligibility of communication systems corrupted by random burst errors that totally disrupt a transmission channel has been provided by ARCON to RADC/EEV through the development of a modified DRT method and scoring procedure. All software development and special projects having to do with the intelligibility testing of voice communication systems are presented in Reference 11.

Extensive intelligibility testing efforts were conducted during this contract period. They include work for the Air Force HAVE QUICK (HQ) and Defense Communications Agency BIT RATE REDUCTION (BRR) programs. Over 380 3-Speaker DRTs were run, scored and analyzed for several groups in the HQ program. A series of 160 3-Speaker DRTs were run, scored and analyzed for the BRR program. In addition consultation was provided on the preparation of test materials, test series design and the interpretation of results.

1.2.2 System Performance Evaluation

An extensive evaluation of the intelligibility of digitally compressed voice communications over severely degraded communications channels was conducted during this contract period. This effort included channel error and transmission delay simulation along with intelligibility testing.

The error and delay simulation software are discussed in this report. The use of the modified DRT method including the selection of missing words and a method for measuring channel down time are discussed in Reference 11.

1.2.3 DoD Digital Voice Processing Consortium Support

The primary product of this effort was the completed Digital DRT/DAM Test Material Library. This has been an ongoing project aimed at the remastering of the DRT/DAM analog library onto digital video media, the equalization of presentation levels for all speakers and the measurement of speech to noise levels. This project is covered in Reference 11.

Other work in this area included the preparation of DRT and DAM (Diagnostic Acceptability Measure) test material for the evaluation of the NSA LPC-10E algorithm. This work required travel to Maryland. Also, work in this area consisted of the preparation of several DAM test series for submittal to DYNASTAT.

1.3 ACKNOWLEDGMENTS

One of the authors, Mr. Charlton Walter, worked as a private consultant to ARCON Corporation while performing research on Canonical Coordinate Based Data Compression Methods. This work is detailed in Chapter 2 of this report.

The authors wish to acknowledge the cooperation, encouragement and technical support received from Mr. Anton Segota, 1st Lt. Denis Robitaille and Luigi Spagnuolo of RADC/EEV. 1st Lt. Clark Carvalho of EEV was responsible for many of the modifications to the Communicability Test Facility discussed in Chapter 6 of this report.

We would like to dedicate the advancements in our Canonical Coordinate algorithm research to the memory of Mr. Segota who had continually supported this effort until his death in March of 1987.

CHAPTER 2

CANONICAL COORDINATE BASED DATA COMPRESSION

For the past eight years ARCON Corporation has been performing original research for RADC/EEV on Array Processor Speech Compression Methods with the ultimate goal to provide a digital speech compression algorithm that can utilize non-euclidean error minimization criteria and be formulated in a parallel or matrix manner to make use of a new generation of true parallel signal processors. These processors are just now beginning to make their mark in the signal processing field. The ARCON algorithm research covered many areas before isolating a particular decomposition of the canonical coordinate (CC) domain [Refs. 6, 9, 10, 12 - 14]. The CC domain is defined by that space in which the error metric or criteria and the signal correlation matrix are diagonal. This decomposition takes advantage of a pseudo-canonical coordinate parameter and an ordering procedure to meet the needs of an analysis - synthesis bandwidth compression communications system. The current status of this algorithm and its relationship to previous work are presented in Section 2.1.

The algorithm has been successfully implemented in vocoder form on the RADC/EEV FPS AP120-B array processor [Ref. 10]. Programs will accept any hermitian error metric for definition of the error criteria, transform input speech files into the defined CC domain and synthesize output speech files at compression ratios and quantization levels set by the operator. The latest version of this implementation is presented in Chapter 3 of this report.

While the AP120-B processor is capable of high speed and vector processing, it is not a true parallel processor. Because of this the CC algorithm is vectorized when implemented on this processor. The NOSC Systolic Array Processor (SAP) discussed in Chapter 5 has been used by ARCON as a testbed for the fully parallel implementation of the CC algorithm [Ref. 14]. This implementation is presented in Chapter 3 of this report.

Research into the definition of specific error metrics for use by the CC algorithm has covered numerous areas during the current contract period. Statistical approaches have been used to define several spectral moment based error metrics. This work led to the search for a transformation that would provide frequency shift invariance during the generation of a spectral moment metric. This work is discussed in detail in Section 2.2.

Statistically derived error metrics have been generated from speech samples using the transformations of Section 2.2. These error metrics are discussed in Section 2.3.

Empirical methods of developing an error metric from a priori psychacoustic information required the development of complex orthogonality transformations. This work is presented in Section 2.4.

Associated with any speech compression system will generally be some type of distortion measure or error metric, used in optimizing the performance of the system. The effectiveness of interoperability between different speech compression systems can be expected to have a close relationship to the compatibility of the error metrics used in the optimization process. Thus the flexibility of the Canonical Coordinate technique in handling a wide class of different error metrics gives us a mechanism for evaluating the efficiency of interoperability of different systems and for improving this interoperability. The relationship between CC and Linear Predictive Coding (LPC) is discussed in Section 2.5.

2.1 CANONICAL COORDINATE BASED VOCODER ALGORITHM DEVELOPMENT EFFORTS

In this section an attempt will be made to both consolidate and to update original research efforts carried out by ARCON for RADC/EEV in the area of Canonical Coordinate Based Vocoder Algorithm Development over the past eight years.

The objective of this research has been to develop algorithms and procedures that exploit the continuing developments in parallel processor technology in order to create a family of highly flexible and compatible vocoder modules (using both software and hardware) that can be easily reconfigured and reoptimized to more efficiently transmit voice and other types of digital data using a variety of different error criteria and different channel bandwidth constraints.

During the period from 1980 - 1984 a very general Canonical Coordinate (CC) based digital data compression technique was developed and reported on [Ref. 6,9,12]. This technique permitted the use of a wide variety of error metrics in the data compression process and held considerable potential for the use of psychoacoustic information in the DVP optimization procedure. The technique was designed to make the maximum use of evolving parallel array and systolic processor architectures. The computationally intensive nature of this particular CC processing technique employed did, however, present some problems associated with the immediate implementation of the full CC signal analysis/synthesis technique on existing first generation array-oriented processors.

This led, during the period from 1984 - 1986 [Ref. 10,13] to the development of a Pseudo-Canonical Coordinate (PCC) based data compression technique that considerably simplified the computational load when utilizing the present generation of (not highly parallel) array processors.

The early stages in the development of the PCC technique contained several erroneous simplifications that restricted the applicability of this technique when applied to situations that could only be handled by the full CC technique. These erroneous simplifications have been corrected in the PCC technique reported here. Fortunately, the simplifications did not affect applications of the approach that were reported in the cited references, since the full generality of the PCC technique was not used in those applications.

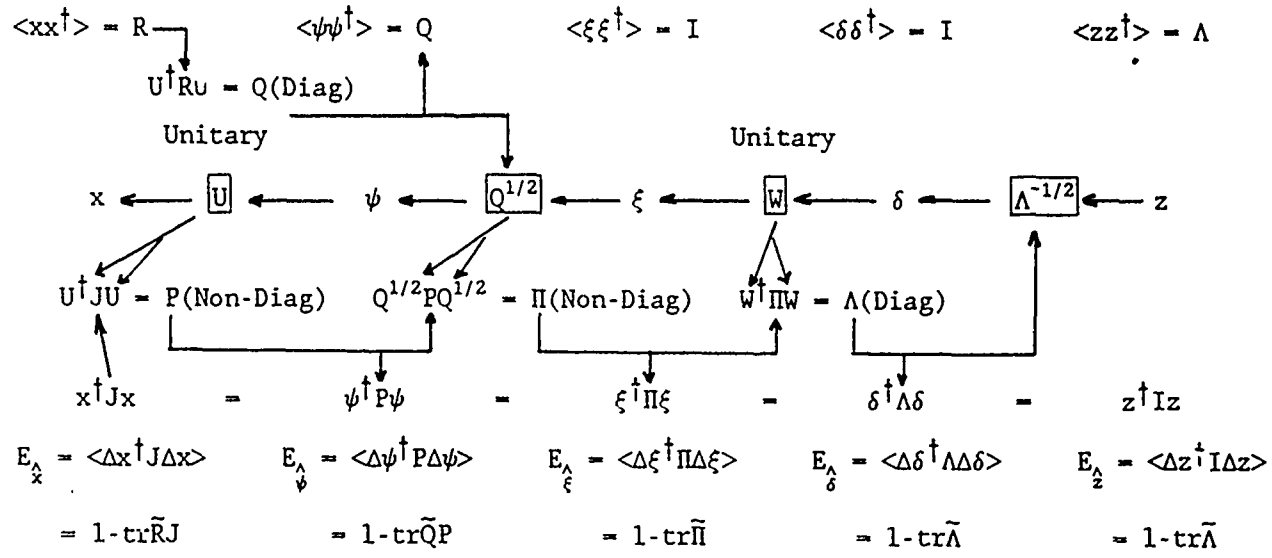
2.1.1 Structural Characteristics Of Two Canonical Coordinate Techniques

The key structural characteristics of the original CC technique is illustrated in Figure 2.1. This is based on an ongoing correlation matrix diagonalization process, involving the use of an intermediate eigenvector/eigenvalue based Principal Component analysis, followed by another eigenvector/eigenvalue type transformation to obtain the full CC representation of the signal data vectors, whose components have been ordered in such a manner that the least error, relative to an appropriate psychoacoustic-based error metric, is incurred in transmitting a truncated representation of the CC data.

Similarly, Figure 2.2 illustrates the key structural characteristics of the updated version of the PCC technique. This technique is based on a single error metric matrix diagonalization application, involving an eigenvector/eigenvalue procedure that defines a transformation to an initial PCC vector β and determines a set of coordinate significance weights, $\{\gamma_n\}$, that provide an initial ordering of the PCC data vector components. This initial pseudo-canonical ordering can then be refined to a full canonical ordering as indicated in the diagram, if necessary. However, as indicated in the detailed description of the PCC technique, a reweighting of γ_n by the factor H_{nn}^2 can be used to obtain a significance weight $\mu_n = \gamma_n H_{nn}^2$ that can be directly applied to ordering the β_n components in an optimum manner relative to the specific error metric.

2.1.2 Updated Version Of The PCC Vocoder Algorithm

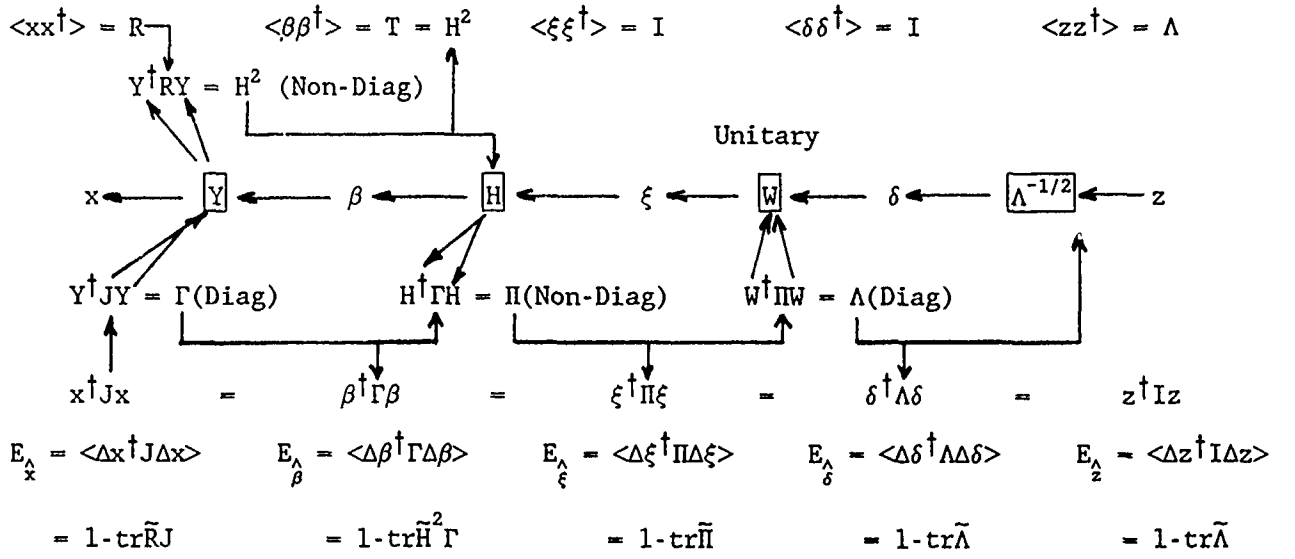
The following description of the PCC technique contains an initial discussion that is essentially the same as that reported in Reference 10, covering the derivation of the pseudo-canonical coordinate β. This is followed with the introduction of another pseudo-canonical coordinate ξ and of its association with the true canonical coordinate δ. Finally, an updated version of signal analysis and reconstruction using the PCC technique is covered.



Domain	x	ψ	ξ	δ	z
Type	Raw Data	Principal Component	Pseudo Canonical	Canonical Coordinate	Canonical Coordinate
Correlation Matrix Characteristic	R Arbitary Hermitian	Q Uncorrelated	I Flat Uncorrelated	I Flat Uncorrelated	Λ Uncorrelated
Error Metric Characteristic	J Arbitrary Hermitian	P Derived Hermitian	Π Derived Hermitian	Λ Real Diagonal	I Real Identity
Error Criteria	$E_{\hat{x}}$ Unordered	$E_{\hat{\psi}}$ Partially Ordered	$E_{\hat{\xi}}$ Partially Ordered	$E_{\hat{\delta}}$ Ordered on Λ	$E_{\hat{z}}$ Ordered on Λ

Figure 2.1

Canonical Coordinate Technique Via Correlation Matrix Diagonalization And Intermediate Principal Components



Domain	x	β	ξ	δ	z
Type	Raw Data	Principal Component	Pseudo Canonical	Canonical Coordinate	Canonical Coordinate
Correlation Matrix Characteristic	R Arbitary Hermitian	T Derived Hermitian	I Flat Uncorrelated	I Flat Uncorrelated	Λ Uncorrelated
Error Metric Characteristic	J Arbitrary Hermitian	Γ Real Diagonal	Π Derived Hermitian	Λ Real Diagonal	I Real Identity
Error Criteria	$E_{\hat{x}}$ Unordered	$E_{\hat{\beta}}$ Partially Ordered	$E_{\hat{\xi}}$ Partially Ordered	$E_{\hat{\delta}}$ Ordered on Λ	$E_{\hat{z}}$ Ordered on Λ

Figure 2.2

Canonical Coordinate Technique Via Error Metric Diagonalization

And Intermediate Pseudo-Canonical Coordinates

Derivation Of The Initial PCC Transformation -

The basic signal analysis and reconstruction process to be utilized in this section is one that treats the original sampled waveform data as a segmented series of consecutive N dimensional data vectors $x(t) = (x_0(t), x_1(t), \ldots, x_{N-1}(t))^T$, indexed on a frame number t, and whose complex valued components, $x_n(t)$, are generally correlated and will be required to satisfy a non-euclidean, quadratic error metric, characterized by an hermitian matrix J in the time domain.

These vectors are first transformed into a frequency domain representation y, through the Fourier transform process Z, as

$$y(t) = Zx(t), \tag{2.1.1}$$

where Z is a unitary DFT matrix having components

$$Z_{mn} = \frac{1}{\sqrt{N}}\exp(\frac{-2\pi imn}{N}).$$

The vectors are then transformed into a succession of special pseudo-canonical and canonical coordinate domains, characterized by the vectors $\beta(t)$, $\delta(t)$ and $z(t)$, that simplifies the reconstruction signal error minimization process, by achieving a more compact representation of the signal x, or its DFT y, in the β, δ and z domains.

Let R be the correlation matrix

$$R = \langle x(t)x^\dagger(t)\rangle ,$$

averaged over a short term ensemble of vectors, $\{x(t)\}$, indexed on the frame number t, where $\langle o\rangle$ represents an averaging process over the ensemble. For the moment, we will permit the probability distribution of x(t) to depend on t. Let S be the "correlation" matrix in the y-domain

$$S = \langle y(t)y^\dagger(t)\rangle .$$

Then from (2.1.1) we have that $\langle yy^\dagger\rangle = Z\langle xx^\dagger\rangle Z^\dagger$, or

$$S = ZRZ^\dagger .$$

Here S is often refered to as a spectral matrix. Note, in particular, that if R is circularly symmetric, then $Z^\dagger$ diagonalizes R, and hence S will be diagonal in this situation.

Now in terms of the above ensemble averaging process, consider a quadratic error criterion, in the x-domain, of the form

$$E_{\hat{x}} = \langle \Delta x^\dagger J \Delta x\rangle = \langle (x(t) - \hat{x}(t))^\dagger J(x(t) - \hat{x}(t))\rangle ,$$

where $\hat{x}(t)$ is an estimate of x(t), and J is the hermitian error metric in the x-domain.

This error can be expressed in terms of an error metric K in the y-domain as

$$E_{\tilde{y}} = \langle \Delta y^{\dagger} K \Delta y \rangle = \langle (y(t) - \hat{y}(t))^{\dagger} K (y(t) - \hat{y}(t)) \rangle ,$$

where $\hat{y}(t) = Z\hat{x}(t)$ and K will be related to J by the equation

$$K = ZJZ^{\dagger} .$$

We now let V be a unitary matrix that transforms K to the diagonal form Γ by means of the eigenvector equation

$$V^{\dagger}KV = \Gamma \quad \text{or} \quad KV = V\Gamma , \text{ where } \Gamma = \text{diag}(\gamma_0, \ldots, \gamma_{N-1}) . \quad (2.1.2)$$

Now let $\beta(t)$ be a vector such that

$$y(t) = V\beta(t) \quad \text{or} \quad \beta(t) = V^{\dagger}y(t) . \quad (2.1.3)$$

Then in terms of the pseudo-canonical coordinate domain β, the quadratic error $E_{\tilde{y}}$ in the y-domain, with metric K, will have the form

$$E_{\tilde{\beta}} = \langle \Delta\beta^{\dagger} \Gamma \Delta\beta \rangle = \langle (\beta(t) - \hat{\beta}(t))^{\dagger} \Gamma (\beta(t) - \hat{\beta}(t)) \rangle ,$$

where Γ ($= V^{\dagger}KV$) is a diagonal error metric.

Let T be the correlation matrix in the β-domain

$$T = \langle \beta(t)\beta^{\dagger}(t) \rangle ; \quad (2.1.4)$$

then from (2.1.3) we have

$$T = V^{\dagger}SV . \quad (2.1.5)$$

Note that T will generally not be diagonal, and hence β will not be a full canonical coordinate, since this requires that both T and Γ be diagonal. However, the diagonality of Γ does provide us with a mechanism for ordering the components of the estimated $\hat{\beta}(t)$ in such a way as to minimize the non-euclidean errors $E_{\tilde{x}}$ or $E_{\tilde{y}}$, where

$$\hat{x}(t) = Z^{\dagger}\hat{y}(t) = Z^{\dagger}V\hat{\beta}(t).$$

Derivation Of Transformation From Pseudo To True CC -

For certain applications an additional pseudo-cannonical coordinate vector $\xi(t)$ and a true canonical coordinate vector $\delta(t)$ need to be defined. Let $\xi(t)$ be such that

$$\beta(t) = H\xi(t) ,$$

where H is an hermitian matrix such that $\xi(t)$ is required to have the identity correlation matrix, i.e. a "flat" spectrum, in the ξ-domain, i.e.

$$<\xi(t)\xi^{\dagger}(t)> = I \ . \qquad (2.1.6)$$

Then from (2.1.4 - 2.1.6) and the condition that H be hermitian (i.e. $H^{\dagger}=H$), we have

$$T = <\beta(t)\beta^{\dagger}(t)> = H<\xi(t)\xi^{\dagger}(t)>H^{\dagger} = HH^{\dagger} = H^2 = V^{\dagger}SV \ . \qquad (2.1.7)$$

From $y(t) = V\beta(t)$ and $V^{\dagger}KV = \Gamma$ we have the inner product equalities

$$y^{\dagger}(t)Ky(t) = \beta^{\dagger}(t)V^{\dagger}KV\beta(t) = \xi^{\dagger}(t)H^{\dagger}\Gamma H\xi(t),$$

i.e.,

$$y^{\dagger}(t)Ky(t) = \beta^{\dagger}(t)\Gamma\beta(t) = \xi^{\dagger}(t)\Pi\xi(t),$$

where Π is the hermitian matrix $\Pi = H^{\dagger}\Gamma H$.

Now let $\delta(t)$ be such that $\delta(t) = W^{\dagger}\xi(t)$, where W is a unitary transformation that diagonalizes Π to Λ via the eigenvector equation

$$W^{\dagger}\Pi W = \Lambda$$

where Λ is real and diagonal since Π is hermitian. Then in the ξ-domain, we have the following error metric, (also inner product), and correlation matrix relationships

$$\delta^{\dagger}(t)\Lambda\delta(t) = \xi^{\dagger}(t)W\Lambda W^{\dagger}\xi(t) = \xi^{\dagger}(t)\Pi\xi(t),$$

and

$$<\delta(t)\delta^{\dagger}(t)> = W^{\dagger}<\xi(t)\xi^{\dagger}(t)>W = W^{\dagger}W = I.$$

Thus $\delta(t)$ is a true canonical coordinate, satisfying the twin requirements of having both a diagonal error metric Λ and a diagonal correlation matrix. Note, in particular, that the correlation "spectrum" in the δ-domain is also "flat".

Signal Analysis And Reconstruction Using The PCC Technique -

From the previous section we see that the pseudo-canonical coordinate β provides the first signal representation domain in which the error metric Γ is guaranteed to be in diagonal form, and hence provides the following procedure for ordering the components of the estimator $\hat{\beta}(t)$ in such a way as to minimize the non-euclidean error $E_{\hat{x}}$, or $E_{\hat{y}}$, where

$$\hat{x}(t) = Z^{\dagger}\hat{y}(t) = Z^{\dagger}V\hat{\beta}(t) \ .$$

In particular, let $\hat{\beta}(t,p)$ be an estimated vector consisting of the

first p components of $\beta(t)$, followed by N-p time invarant estimates for the remaining components that are not transmitted for use in the resynthesis process, i.e.,

$$\hat{\beta}(t,p) = (\beta_0(t), \ \ldots \ , \beta_{p-1}(t), \hat{\beta}_p, \ \ldots \ , \hat{\beta}_{N-1}) \ .$$

Then $\hat{E}_\beta(p) = <\Delta\beta^\dagger \Gamma \Delta\beta>$

$$= \sum_{m,n=0}^{N-1} <\Delta\beta_m^*(t,p)(\mathrm{diag}(\gamma_0, \ \ldots \ , \gamma_{N-1}))\Delta\beta_n(t,p)>$$

$$= \sum_{n=p}^{N-1} \gamma_n <(\beta_n(t)-\hat{\beta}_n)(\beta_n(t)-\hat{\beta}_n)^*>$$

$$= \sum_{n=p}^{N-1} \gamma_n (<\beta_n(t)\beta_n^*(t)> - <\beta_n(t)><\beta_n^*(t)>)$$

and if we normalize $\beta(t)$ so that $<\beta(t)> = 0$, i.e.,

$$<\beta_n(t)> = 0 \text{ for all } n = 0,1, \ \ldots \ ,N-1 \ ,$$

then the above expression reduces to

$$\hat{E}_\beta(p) = \sum_{n=p}^{N-1} \gamma_n H_{nn}^2 = \sum_{n=p}^{N-1} \mu_n \qquad (2.1.8)$$

where from (2.1.7)

$$<\beta_n(t)\beta_n^*(t)> = T_{nn} = H_{nn}^2$$

and where $\mu_n = \gamma_n H_{nn}^2$. We call μ_n the nth component significance measure.

Thus $E_\beta(p)$ will be minimized for each p, provided the coefficients μ_n are ordered so that

$$\mu_0 \geq \mu_1 \geq \ \ldots \ \geq \mu_{N-1} \ .$$

From (2.1.7) we have a convenient procedure for calculating H_{nn}^2 as

$$H_{nn}^2 = v^\dagger(n) S v(n)$$

where v(n) is the nth eigenvector of (2.1.2), i.e. $Kv(n) = \gamma_n v(n)$. Moreover, if S is diagonal, then

$$H_{nn}^2 = \sum_{m=0}^{N-1} |v_m(n)|^2 s_m \ .$$

The specific implementation of the above analysis and reconstruction process can be carried out in two different ways, depending upon whether the ensemble distribution of x(t) is regarded as fixed over all t, or varying with time.

In the first course of action we have available relatively time-independent values for the matrix $H^2 = T = <\beta(t)\beta^\dagger(t)>$ in the β-domain that can be used, in conjunction with a time independent $\Gamma = V^\dagger KV$, to define a set of time independent β-component significance measures $\mu_n = \gamma_n H^2_{nn}$. The time dependent components $\beta_n(t)$ can then be determined from the vector equation $\beta(t) = V^\dagger y(t)$, for each frame $y(t)$, and ordered on the significance measures μ_n for purpose of transmission.

This provides an explicit, time independent procedure for ordering the coordinates in the pseudo-canonical coordinate representation in such a way as to minimize the non-euclidean error for any given number of transmitted components of β. Since the transformation matrix V will be available at both analysis and reconstruction sites, only β-component values, in a specified predetermined order, need be transmitted.

In the second course of action, involving short term averaging and the direct use of the time varying spectra data $y(t)$ as a preliminary step, a different way of implementing the canonical coordinate methodology is desirable. This is particularly true in speech signal data compression and resynthesis applications where long term averaging leads to loss of intelligibility and quality in the resynthesized speech. Here, for each data vector $x(t)$ a DFT vector $y(t) = Zx(t)$ is constructed, using a zerofill procedure. Then, on the assumption that $R(t) = <x(t)x^\dagger(t)>$ is defined by averaging over one circularly shifted frame of data, we have that $R(t)$ is circularly symmetric and hence $S(t) = ZR(t)Z^\dagger = \text{diag}(s_0(t), \ldots, s_{N-1}(t))$ is diagonal, where $s_n(t) = |y_n(t)|^2$. Consequently the component significance measures $\mu_n(t)$, for each frame t, can be expressed as

$$\mu_n(t) = \gamma_n H^2_{nn}(t) = \gamma_n \sum_{m=0}^{N-1} |v_m(n)|^2 s_m(t) ,$$

and used to provide an ordering, via

$$\mu_{q(0)}(t) \geq \mu_{q(1)}(t) \geq \ldots \geq \mu_{q(N-1)}(t),$$

for the order in which the first p components of $\beta(t) = V^\dagger y(t)$,

$$\text{i.e.}(\beta_{q(0)}(t), \beta_{q(1)}(t), \ldots, \beta_{q(p-1)}(t))$$

are to be transmitted, along with the first p values of the permutation vector q, in order to reconstruct the p-th order approximation $\hat{\beta}(t,p)$, then $\hat{y}(t,p)$ and finally $\hat{x}(t,p)$.

2.2 FREQUENCY SHIFT INVARIANCE / ETA-TRANSFORM STUDY

In some applications of power spectrum analysis, we are presented with power spectra that differ primarily by a frequency scale factor. This phenomenon appears in the study of acoustic signals produced by a pitch-like excitation, such as voiced speech or aircraft engine noise. If signals contain energy primarily concentrated at the harmonics of a "pitch" frequency, which itself may vary from one power spectrum to

another, then the difference in pitch (which shifts all the harmonics) can mask fundamental similarities between the harmonic structures. In this section we introduce a transform intended to produce a signature that is invariant under pitch changes; we call this transform the Discrete Eta-Transform.

2.2.1 Continuous Case

Before we discuss the discrete η-transform itself, we will define its continuous analog. Let $x(t)$ be a real-valued function whose Fourier transform $y(\omega)$ exists:

$$y(\omega) = \int_{-\infty}^{\infty} e^{-2\pi j\omega t}\, x(t)\, dt, \qquad -\infty < \omega < \infty. \tag{2.2.1}$$

Recall that if $x(t)$ is replaced by a shifted function $x(t+a)$ then $y(\omega)$ is replaced by $e^{-2\pi ja\omega}\, y(\omega)$, so that the magnitude of $y(\omega)$ is unchanged. Define the function $\tilde{y}$ by

$$\tilde{y}(\lambda) = |y(e^{\lambda})|, \qquad -\infty < \lambda < \infty,$$

and define η by

$$\eta(\sigma) = \int_{-\infty}^{\infty} e^{-2\pi j\sigma\lambda}\, \tilde{y}(\lambda)\, d\lambda, \qquad -\infty < \sigma < \infty,$$

or equivalently

$$\eta(\sigma) = \int_{-\infty}^{\infty} e^{-2\pi j\sigma\lambda}\, |y(e^{\lambda})|\, d\lambda. \tag{2.2.2}$$

We call η the eta-transform of x.

NOTE: η is related to the Mellin transform of the function $|y|$. The Mellin transform [Ref. 15] of a function x (of the real variable t) is the function X (of the complex variable z) defined by:

$$X(z) = \int_{0}^{\infty} u^{z-1}\, x(u)\, du,$$

or after the change of variable $u = e^{s}$,

$$X(z) = \int_{-\infty}^{\infty} e^{zs}\, x(e^{s})\, ds,$$

and if we evaluate X at $z=-2\pi j\sigma$ then we obtain

$$X(-2\pi j\sigma) = \int_{-\infty}^{\infty} e^{-2\pi j\sigma s}\, x(e^{s})\, ds. \tag{2.2.3}$$

Comparing equations (2.2.2) and (2.2.3), we see that $\eta(\sigma)$ is the Mellin transform of $|y|$ evaluated on the imaginary axis at $-2\pi j\sigma$.

The key property of the η-transform is that its magnitude is invariant under both time shifts and linear changes in frequency scale. Suppose that two functions x_1 and x_2 have Fourier transforms y_1 and y_2 that are related by

$$|y_1(\omega)| = |y_2(a\omega)| \qquad (2.2.4)$$

where a is a positive constant independent of ω. (Note that (2.2.4) will be satisfied if x_1 and x_2 differ by a time shift and a linear change in frequency scale.) Then the functions $\tilde{y}_1$ and $\tilde{y}_2$ satisfy

$$\tilde{y}_1(\lambda) = \left|y_1(e^{\lambda})\right| = \left|y_2(a\, e^{\lambda})\right| = \left|y_2(\, e^{\lambda + \ln a})\right| = \tilde{y}_2(\, \lambda + \ln a\,);$$

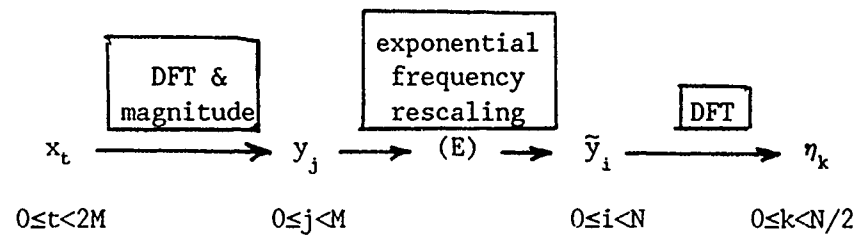
i.e., $\tilde{y}_1$ and $\tilde{y}_2$ differ by a shift of ln a. It follows that their Fourier transforms η_1 and η_2 are related by

$$\eta_1(\sigma) = e^{-2\pi j\sigma \ln a}\, \eta_2(\sigma) \qquad \text{for all } \sigma,$$

and so $\eta_1(\sigma)$ and $\eta_2(\sigma)$ have the same magnitude, for each σ.

2.2.2 Discrete Case

By analogy with the continuous η-transform defined above, the discrete η-transform is formed by taking the magnitudes of a discrete Fourier transform (DFT), discarding the "redundant" half of this vector, performing an exponential distortion of the frequency domain, and imposing another DFT, as shown in the block diagram below:



The "frequency rescaling" operation takes as its input a vector whose components are uniformly spaced in the frequency domain (at frequencies 0, c, 2c, 3c, ...), and produces as its output a vector whose components are proportionally spaced (at frequencies fr^0, fr^1, fr^2, fr^3, ...). Table 2.1 shows an example of a set of proportionally spaced frequencies.

If the three operations (DFT, rescaling, and second DFT) are replaced by their continuous analogs (Continuous Fourier Transform, rescaling, and second Continuous Fourier Transform), we have the η-transform described in the previous section. As shown there, a pitch-like change in the input signal produces no change in the magnitudes of the continuous η-transform, because the rescaling step turns the pitch-like change into a simple linear shift. It can be hoped that the discrete version would share this desirable property, with appropriate constraints on the input signal.

Figure 2.3 shows the discrete η-transforms of five different time series. Each of the five inputs is a time series generated by sampling the sum of two sinusoids (256 samples, sampling rate = 8192 Hz), hamming weighting and filling with 256 trailing zeroes, the lower-frequency sinusoid having a frequency 2/3 that of the higher-frequency sinusoid. Except for the left-hand column, all the graphs show complex data represented as phase and log magnitude.

Table 2.1
Example Set of Proportionally Spaced Frequencies

base frequency = 150.0 Hz
frequency ratio = 1.05
original DFT size = 128 samples
assumed sampling rate = 8 kHz

Index i	Frequency (Hz)	Fractional DFT bin
0	150.0	2.40
1	157.5	2.52
2	165.4	2.65
3	173.6	2.78
4	182.3	2.92
5	191.4	3.06
6	201.0	3.22
7	211.1	3.38
8	221.6	3.55
9	232.7	3.72
10	244.3	3.91
11	256.6	4.10
55	2195.3	35.13
56	2305.1	36.88
57	2420.4	38.73
58	2541.4	40.66
59	2668.4	42.70
60	2801.9	44.83
61	2942.0	47.07
62	3089.1	49.42
63	3243.5	51.90

We should also mention two possible alternatives to the discrete η-transform we have defined:

1. The rescaling operation E could be applied to the DFT outputs themselves, instead of their magnitudes. In this case the results would be affected by (circular) time shifts in the input. Figure 2.4 shows the same test signals as Figure 2.3, but processed without taking magnitudes of the initial DFT. The final DFT shown in the right-hand column is plotted in full, since its last half is not redundant in this case.

2. The initial DFT and rescaling could be replaced by a single step, consisting of a DFT evaluated directly at proportionally spaced frequencies. This renders FFT techniques unusable, but provides a theoretically attractive method of interpolation. Figure 2.5 shows the same set of test signals processed by direct DFT evaluation.

Four difficulties appear when we pass from the continuous to the discrete realm and apply the transform to real-world signals. First, we lose the shift invariance of the Fourier transform magnitude, gaining instead the circular shift invariance of the discrete Fourier transform magnitude. Therefore we can expect the discrete η-transform magnitude to be unchanged by *circular* linear shifts in the vector $\tilde{y}$. But a linear change in frequency scale does not produce a circular shift in $\tilde{y}$; instead, components shifted off the high (or low) end of $\tilde{y}$ re-enter the same end in a mirror image. Therefore the discrete η-transform magnitude will be changed if there are significant nonzero components shifted off either end of the vector $\tilde{y}$.

A second difficulty is the windowing effect of a finite-length data record. When the signal source undergoes a shift in frequency, the data window does not. Thus there will be changes in the η-transform magnitude due to the interplay of signal and window. This applies whether we use a rectangular window or some other window.

A third difficulty (actually another windowing effect) is that only certain frequency shifts actually leave the discrete η-transform magnitude unchanged. To leave $|\eta|$ exactly the same, the frequency factor must be an integer power of the discrete frequency ratio r that is chosen. The second and third rows of Figure 2.6 show discrete eta-transforms of two data records differing by a frequency scale factor of 4/3, the frequency ratio r being chosen as the 12th root of 4/3. (It can be seen from the figure that even in this case, other windowing effects cause the final transform magnitudes to differ.)

Finally, the discrete η-transform is difficult to invert. In fact, the shortcut version using an FFT and two-point linear interpolation is not invertible at all, since it discards some of the information in its input.

Because of these limitations, we expect the discrete η-transform to be of limited utility in our future research.

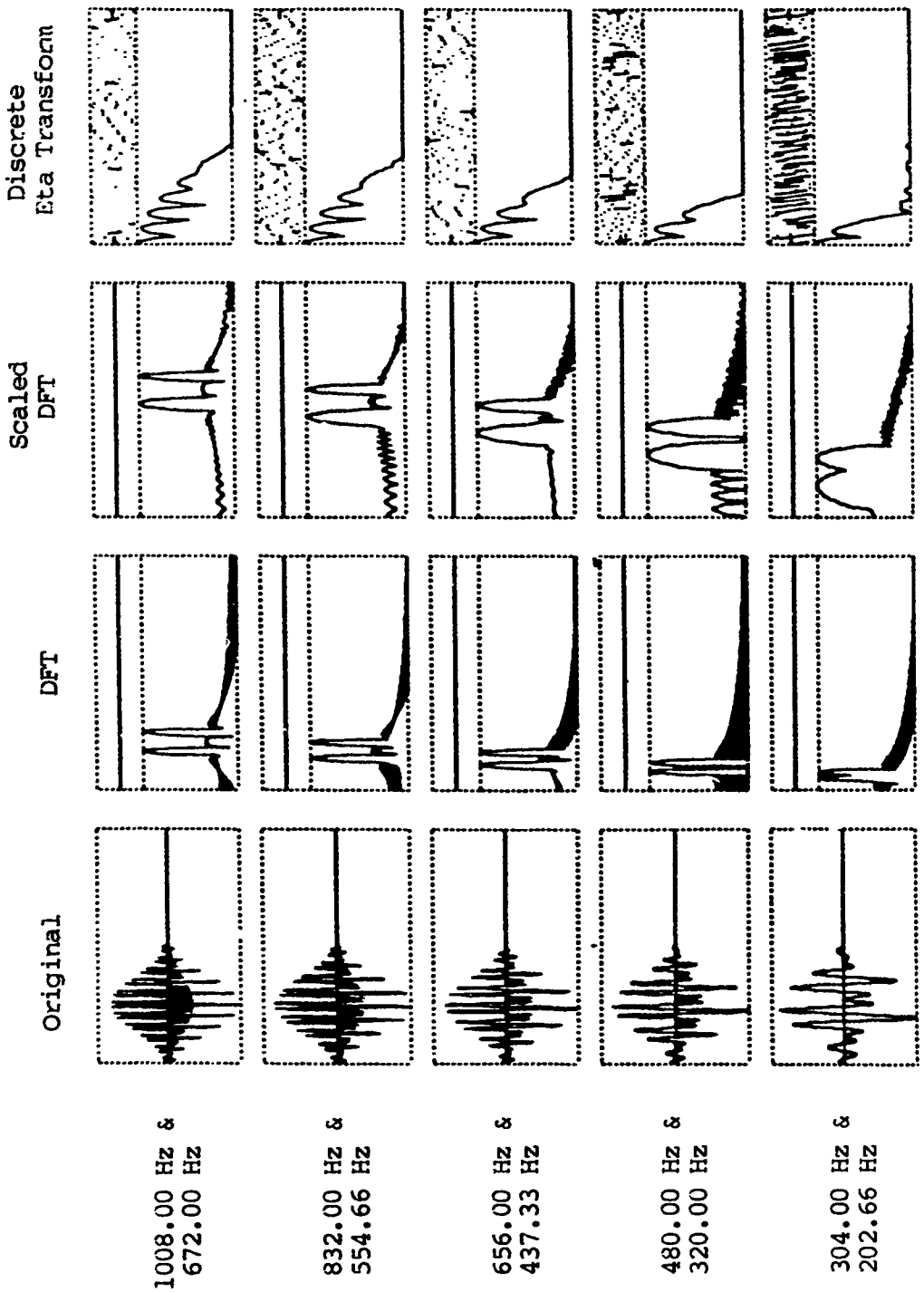


Figure 2.3
Discrete Eta-Transforms of Test Signals

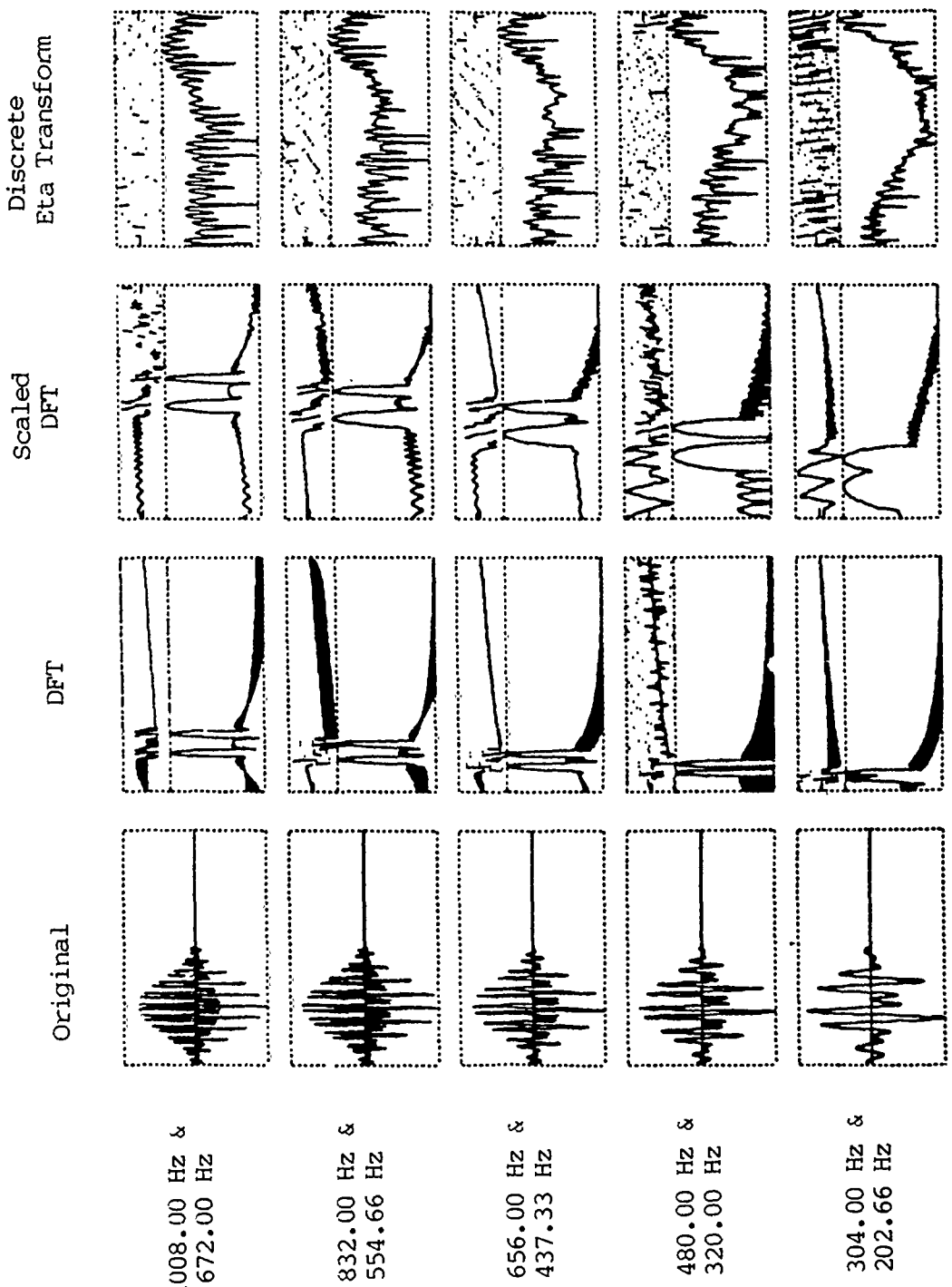


Figure 2.4
Discrete Eta-Transforms with
Magnitude Step Omitted

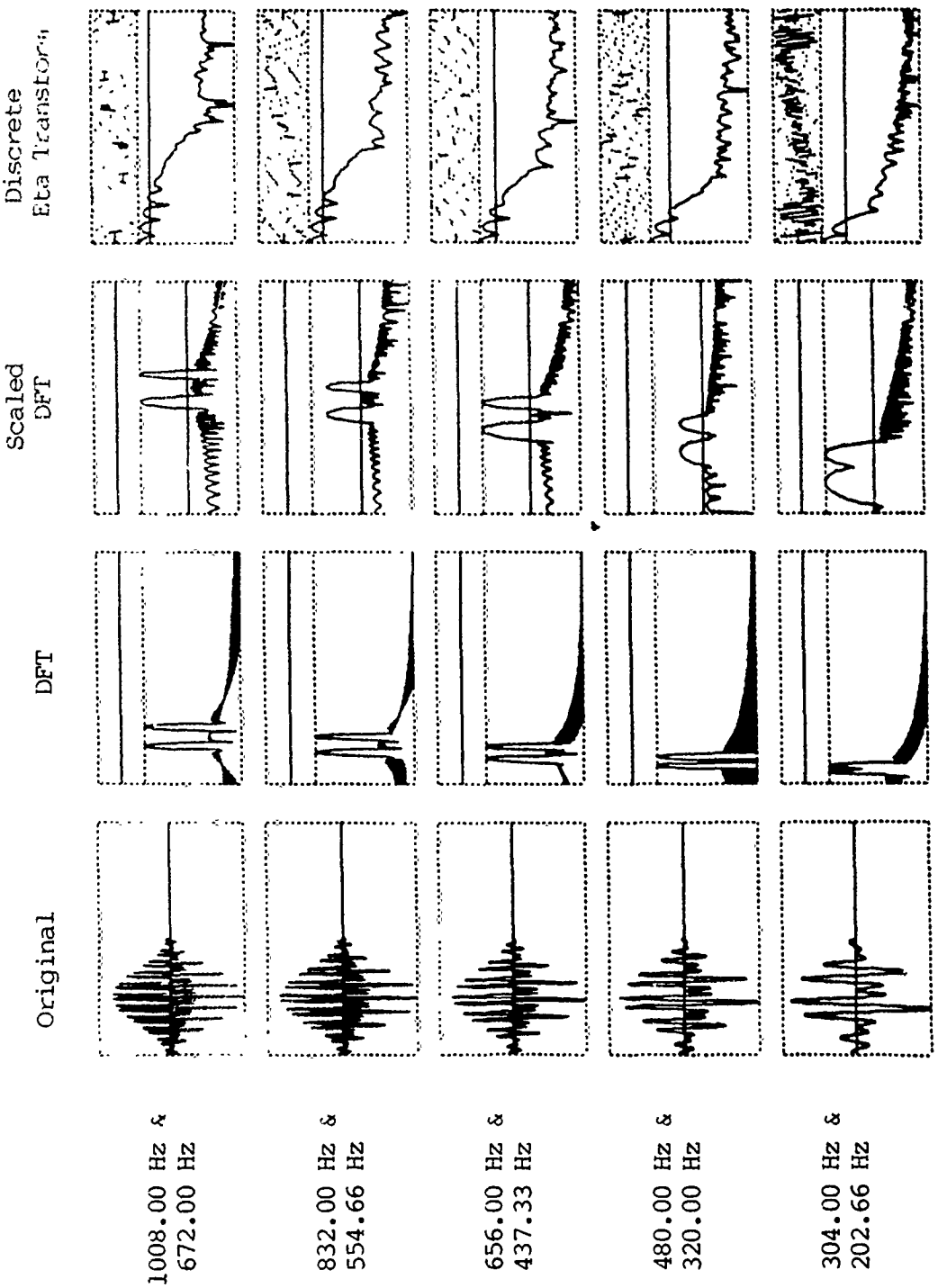


Figure 2.5
Discrete Eta-Transforms with
Direct DFT Evaluation (i.e. No Interpolation)

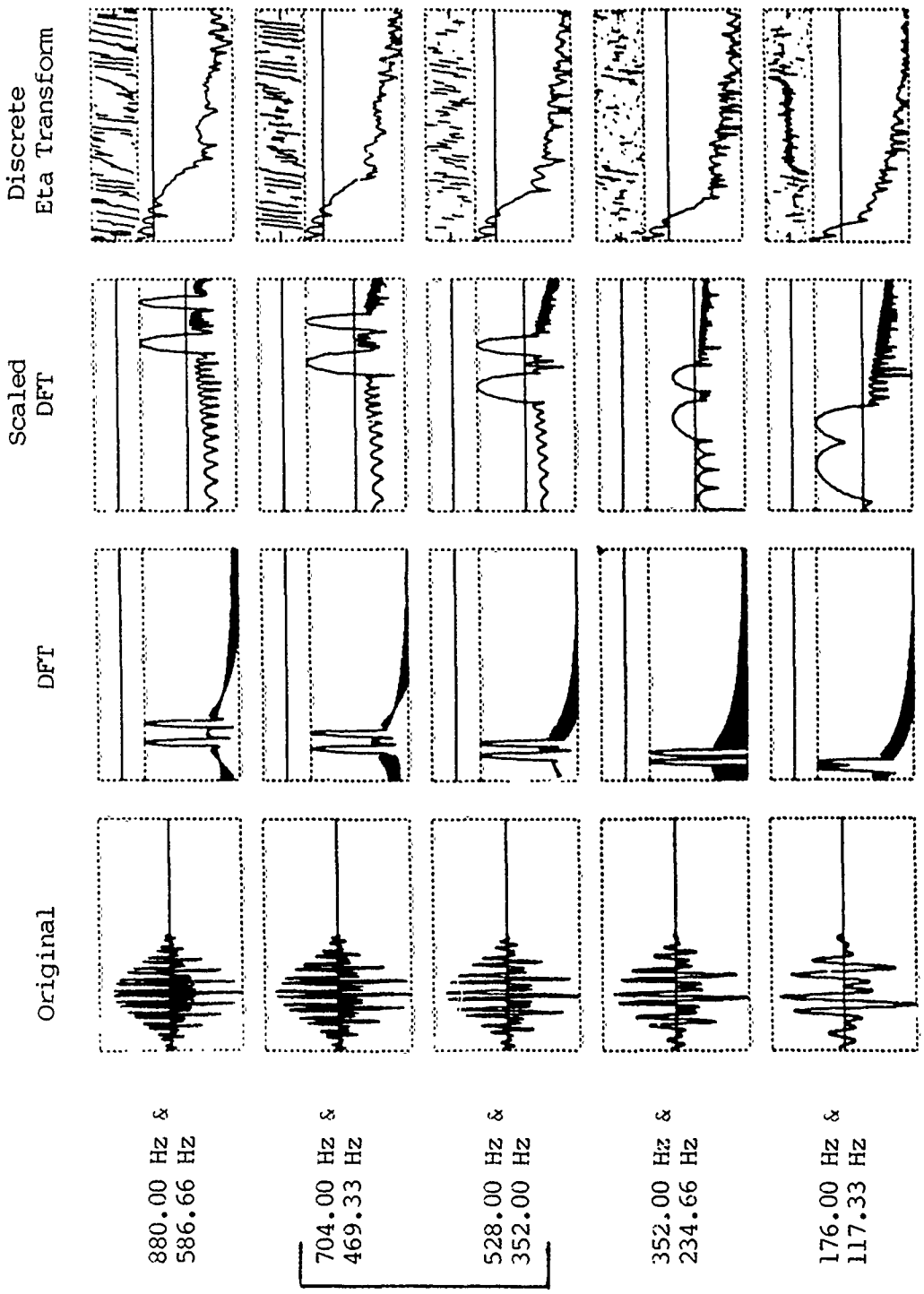


Figure 2.6
Discrete Eta-Transforms with
Exact Frequency Ratios

2.2.3 Eta Transform Software

The following η transform procedures have been written by ARCON for the AP-120B using AP's Vector Function Chainer (VFC) language:

1. MTIFWD -- Eta Transform Interpolation, Forward
2. MTIINV -- Eta Transform Interpolation, Inverse
3. MTDFWD -- Eta Transform Direct, Forward

MTIFWD and MTIINV perform the interpolation step (E in the block diagram) and its inverse. By contrast, MTDFWD replaces both the DFT step and the interpolation step, performing a direct DFT evaluation at proportionately spaced frequencies instead of interpolation. Table 2.2 lists the AP120B subroutines available for forward and inverse interpolation. As the table shows, there are multiple versions of MTIFWD available under different file names, using different methods of interpolation. In order that a program may be written to call them all interchangeably, all versions are called by the name MTIFWD; a particular version is chosen at the task-building stage. An initialization subroutine, MTINIT, creates tables in the AP120B memory for later use by MTIFWD and MTIINV. Programs using MTIFWD or MTIINV must call MTINIT first. Likewise there is an initialization subroutine MTDINI that must be called before using MTDFWD.

MTIFWD takes a vector y of length M, interpreted as the result of a complex DFT, and produces a vector $\tilde{y}$ of length N by "resampling" y at unequally spaced (geometrically related) frequencies

$$f\ r^i, \qquad i = 0, 1, \ldots, N-1,$$

where f and r are constants. The constant f is the "base frequency" in Hz, the lowest frequency used; the constant r is the ratio between successive frequencies. The original DFT y has a frequency resolution $f_s/2M$, where f_s is the sampling frequency. The frequencies represented in $\tilde{y}$ are proportionally spaced, and in general the new frequencies fall between the integer multiples of the DFT resolution $f_s/2M$ that are represented in the vector y. Interpolation is performed to obtain results at the intermediate frequencies.

The base frequency f is converted into a DFT bin number $b = f/(f_s/2M)$, which is not necessarily an integer, and the value of $\tilde{y}_i$ is obtained for each i from 0 to M-1 by computing $j = br^i$, truncating to form [j], the greatest integer not exceeding j, and then interpolating between $y_{[j]}$ and $y_{[j]+1}$.

So far we have not discussed how the interpolation is done, and indeed Table 2.2 shows that more than one interpolation method has been implemented. A simple two-point linear interpolation is one possibility; in this case, the transformation E is itself linear. One advantage of this option is its low computational cost. This option is implemented in one version of MTIFWD. Another possibility is to use a full interpolation formula to reconstruct exactly the DFT evaluated at the fractional frequency, or to evaluate the DFT at fractional frequencies directly from the time-domain signal (as in

MTDFWD). This option is computationally expensive. Still another option is a compromise between computational cost and accuracy: the DFT is converted into polar form and the phases and magnitudes are linearly interpolated independently. This option is implemented in another version of MTIFWD.

MTIINV is something like an inverse to MTIFWD. Strictly speaking, MTIFWD has no inverse--it throws away some of the information contained in the original vector y. But MTIINV attempts to invert the transformation, reversing the scaling transformation $j = br^i$, which when solved for i gives

$$i = \log(j/b) \; / \; \log r \; .$$

Then y_j is reconstructed by interpolating between $\tilde{y}_{[i]}$ and $\tilde{y}_{[i]+1}$.

When this scale change calls for frequencies outside the range represented by values of i between 0 and N-2, a linear extrapolation is performed based on the two frequencies nearest the extreme.

Table 2.2
AP120B Software for Use in Eta Transforms

VFC interpolation routines for forward and inverse discrete eta-transforms, and Fortran subroutines that create the tables needed by the interpolation routines:

MTINPR.FTN--initialization subroutine MTINIT

MTIFWN.VFC--forward VFC subroutine MTIFWD, with two-point interpolation

MTIINN.VFC--inverse VFC subroutine MTIINV, with two-point interpolation

MTIFWP.VFC--forward VFC subroutine MTIFWD, with polar interpolation separately in phase and magnitude

MTDINI.FTN--initialization subroutine MTDINI (for MTDFWD)

MTDFWD.VFC--forward VFC subroutine using direct DFT evaluation at proportionally spaced frequencies

2.3 STATISTICALLY BASED ERROR METRICS

Canonical Coordinate data compression utilizes a simultaneous diagonalization of two hermitian-symmetric matrices: the signal covariance matrix and an error metric matrix. This diagonalization is achieved by one of two methods as summarized in Figure 2.7. The signal vector $\underline{y}$ can be either a directly digitized PCM signal, or can be some vector derived from the original signal.

In Section 2.1 we have given some prominence to the discrete Fourier transform of the original signal, since it may be easier or more intuitive to specify error metrics in the frequency domain. Since the discrete Fourier transform is invertible, a signal estimate can be reconstructed from a DFT estimate at the decoder. Alternatively, we could consider coding the phase and magnitude of the DFT separately. In this case, both phase and magnitude could be independently cast into canonical coordinate systems, each with its own error metric and signal covariance; or the canonical coordinate technique could be applied to the magnitude (or phase) only, leaving the other to be coded by some other method. Other possibilities include coding a nonlinear function (such as the logarithm) of the magnitude of the DFT, or applying logarithmic scaling to the frequency axis as in the Discrete Eta-Transform discussed in Section 2.2. In each case, we have a "signal" vector derived from the original signal vector. The derived vector has its own statistical characteristics different from those of the original signal, and thus its own canonical coordinate transformation.

In order to investigate some of these options, we have studied the covariance matrices implied by some of these signal transformations. We have used actual speech signals and computed the long-term time-averaged statistics of various derived signal vectors (such as DFT magnitude, discrete Eta-transform, or DFT phase), and then obtained the principal component transformations corresponding to these statistics. For simplicity, we have used only Euclidean error metrics.

2.3.1 Sources Of Data And Covariance Estimation Procedures

As our samples of speech, we used two different data sets. In order to control the variability of the data, each data set consisted of utterances by a single speaker, in a quiet environment, using a dynamic microphone. One data set (set "CV") consists of a DRT word list (list CH307A) read by speaker CH; the other data set (set "DA") consists of a DAM sentence list read by speaker CH. The long-term average spectral distribution of the two data sets is shown in Figures 2.8 and 2.9.

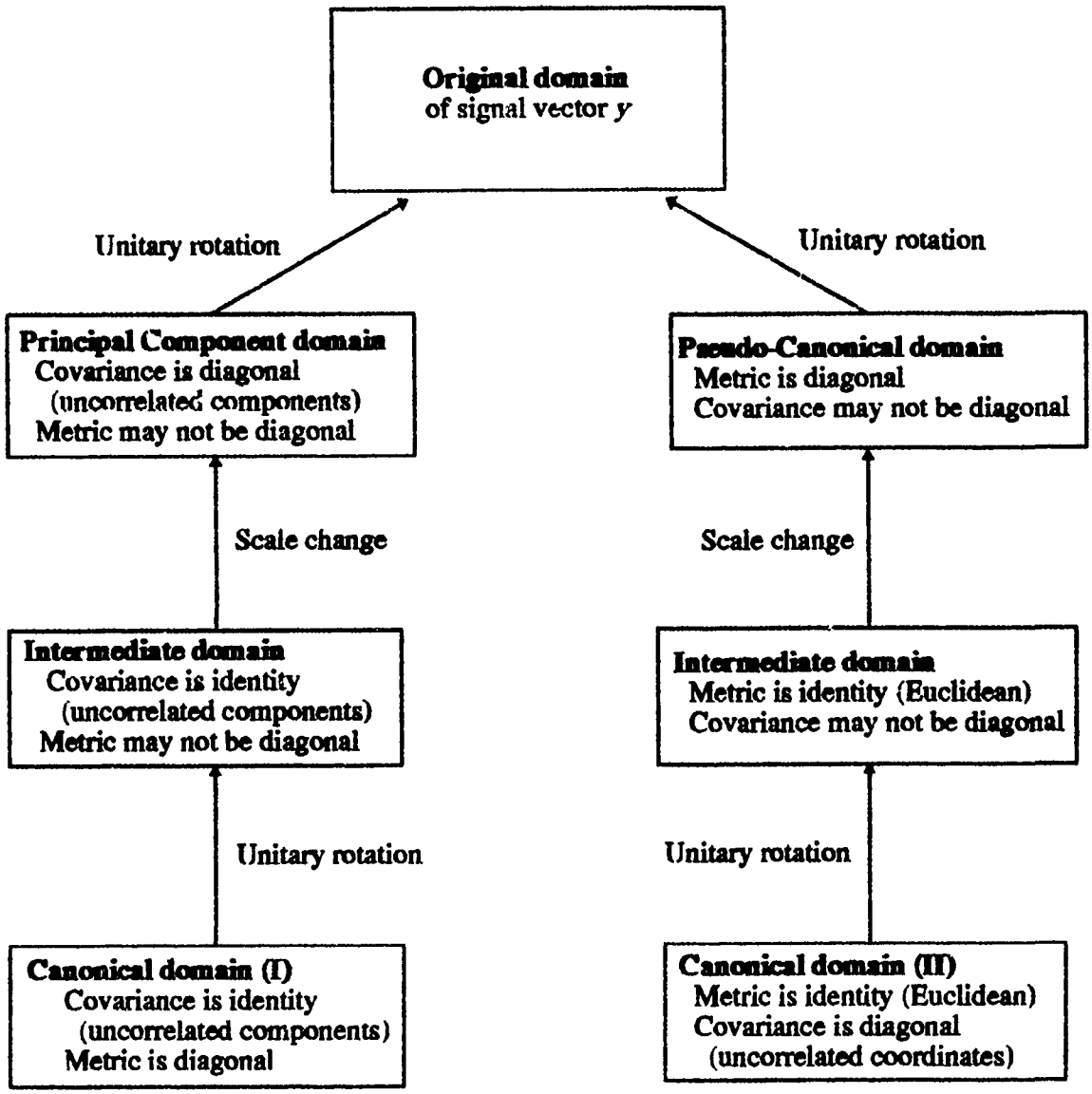


Figure 2.7
Cannonical Coordinate Methods

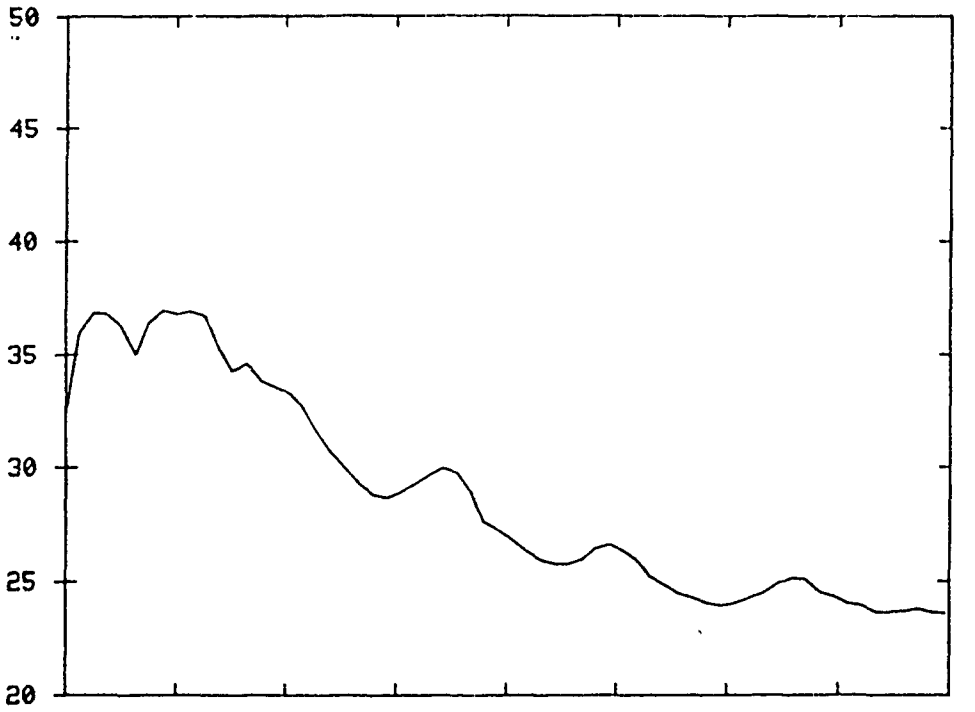


Figure 2.8
Long Term Spectral Distribution of DRT Wordlist CH307A

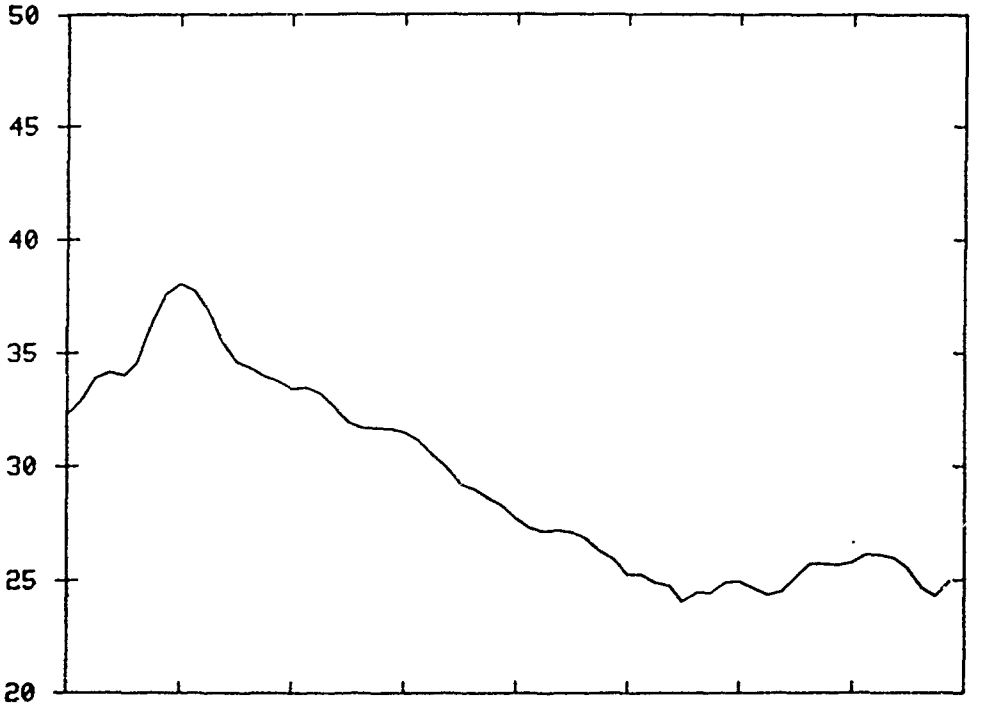


Figure 2.9
Long Term Spectral Distribution of DAM Sentence List
Speaker CH, File QSET1.DAM.

Each data set was analyzed in 64-sample frames. A simple fixed-threshold silence suppression rule was applied. Each non-silent frame was extended by 64 zeroes and a discrete Fourier transform was computed on the result. Of the 128 DFT values, the first 64 were retained. At this point the analyses diverged, depending on whether the derived signal being studied was

1. the raw complex DFT vector,
2. the vector of DFT magnitudes,
3. the vector of logarithms of DFT magnitudes,
4. the raw complex DFT vector with Eta frequency scaling,
5. the vector of DFT magnitudes with Eta frequency scaling, or
6. the vector of logarithms of DFT magnitudes with Eta frequency scaling.

Whichever derived signal was used, it formed a vector of length 64 for each frame of speech. The covariance matrix of this vector was then estimated as the average over time of the various products of the vector components.

For each covariance matrix, we performed an eigensystem solution to diagonalize the matrix, yielding a transformation matrix for principal component compression of the chosen derived signal vector. As a consequence of a relationship analogous to Equation 2.1.8, the mean square error for principal component data compression is equal to the sum of the eigenvalues corresponding to coordinates not transmitted. Therefore the eigenvalues of the covariance matrix provide all the information needed to determine the mean square error at any given level of principal component data compression. Specifically, the mean square error caused by dropping some coordinates is equal to the sum of the eigenvalues corresponding to those coordinates. However, it should be noted that there is no direct comparison between a certain level of mean square error in one type of derived signal vector, such as DFT magnitude, and a certain level of mean square error in a different type of derived signal vector, such as log DFT magnitude with Eta frequency rescaling.

Gaussian Noise Case - In order to provide a simple example of this method, we have computed the covariance matrix and corresponding eigenvalues for the complex DFT vector applied to a synthetic data set consisting of white Gaussian noise. For such a data set, the true covariance matrix is the identity multiplied by a scalar constant, and the eigenvalues are all equal to that constant. Figure 2.10 shows a summary of the covariance metric for the complex DFT vector estimated from white Gaussian noise. The gray scale plot in each figure indicates the magnitudes of elements of the correlation matrix obtained from the estimated covariance matrix. The plot of eigenvalues shows that half the eigenvalues are approximately zero, and the other half are approximately equal to each other. This is not surprising, since the zero-filling operation applied to each frame results in an oversampled DFT. The plot of cumulative eigenvalue sums shows that as we decrease the compression rate from 100% to 50% the expected error decreases from 100% to 0%.

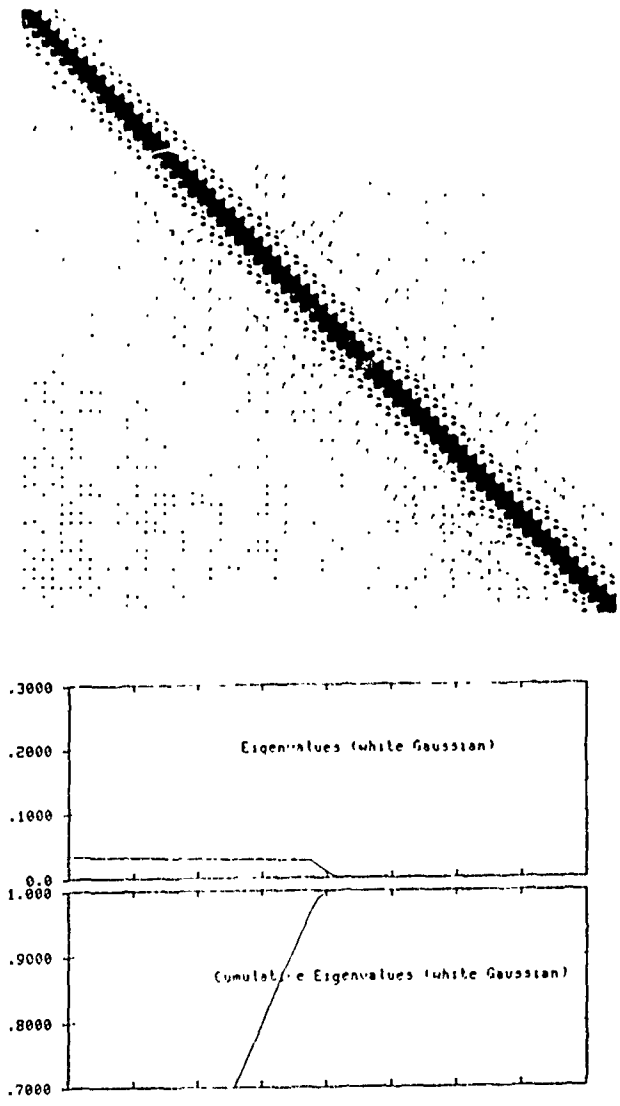


Figure 2.10
Correlation Matrix and Eigenvalue Characteristics
White Gaussian Noise, DFT Magnitude, Linear Frequency

2.3.2 Conclusions

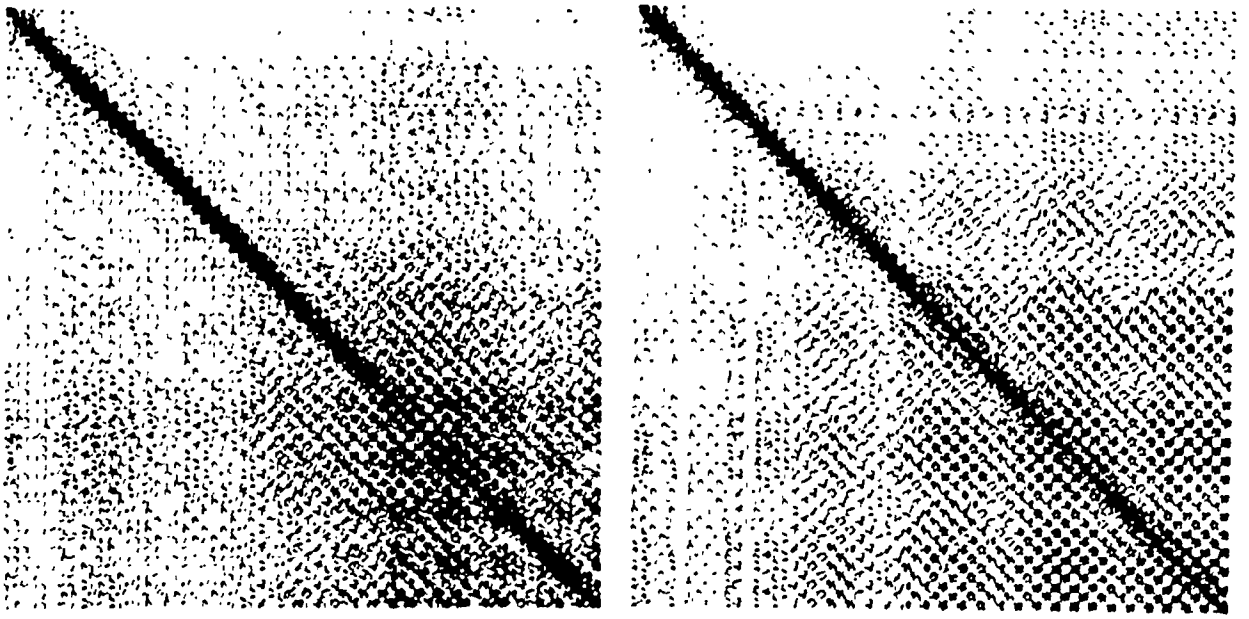
Although the following comparisons show some qualitative distinctions, it should be reiterated that compression efficiency is not directly comparable between any two cases.

Raw DFT - Figures 2.11 and 2.12 summarize statistical characteristics of the complex DFT of speech frames, with and without logarithmic frequency rescaling. The gray scale plot in each figure indicates the magnitudes of elements of the correlation matrix obtained from the estimated covariance matrix. For data set "DA", the cumulative eigenvalue curve is steeper, implying better compression, in the linear-frequency case. On the other hand, for data set "CV", the cumulative eigenvalue curve is steeper in the log-frequency case.

Magnitude Of DFT - Figures 2.13 and 2.14 summarize statistical characteristics of the magnitudes of the DFT of speech frames, with and without logarithmic frequency rescaling. The cumulative eigenvalue curve is steeper, implying better compression, in the log-frequency case.

Logarithm Of Magnitude Of DFT - Figures 2.15 and 2.16 summarize statistical characteristics of the logarithm of magnitudes of the DFT of speech frames, with and without logarithmic frequency rescaling. The cumulative eigenvalue curve is steeper, implying better compression, in the log-frequency case.

Phase Of DFT - Figure 2.17 summarizes statistical characteristics of the phase of the DFT of speech frames. Phases at different frequencies appear to be independent, offering no opportunity for compression by this method.



From DA

From CV

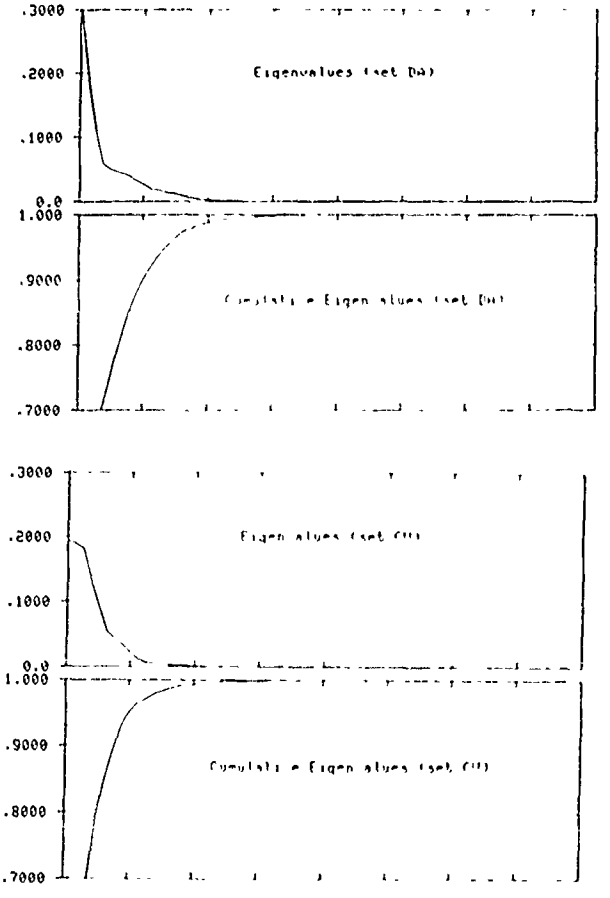
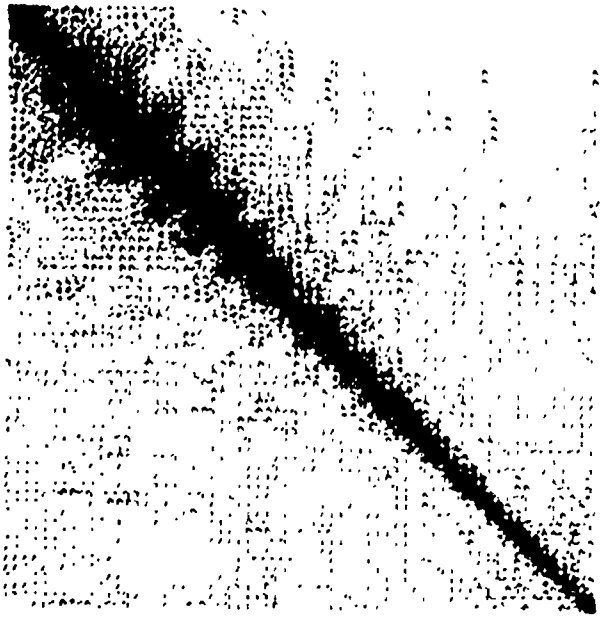
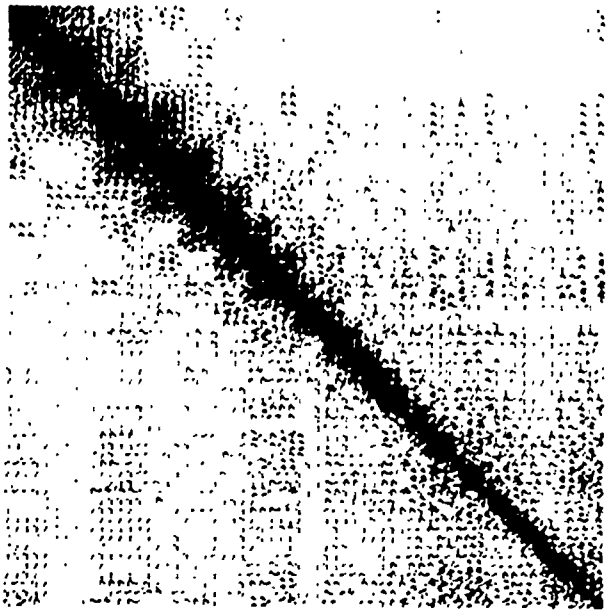


Figure 2.11
Correlation Matrix and Eigenvalue Characteristics
Speech, DFT, Linear Frequency



From DA



From CV

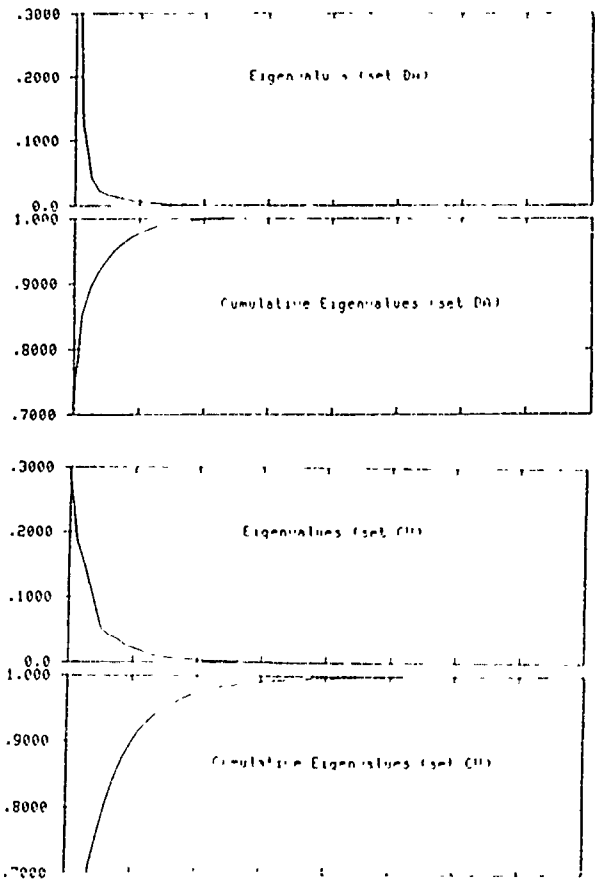
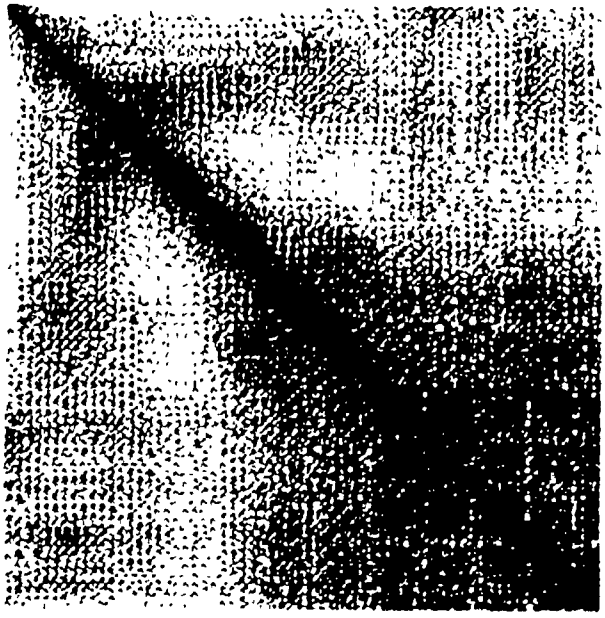
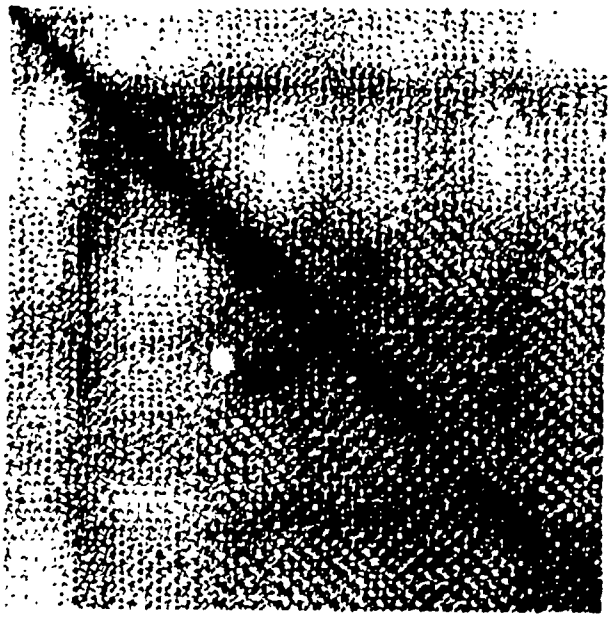


Figure 2.12
Correlation Matrix and Eigenvalue Characteristics
Speech, DFT, Log Frequency



From DA



From CV

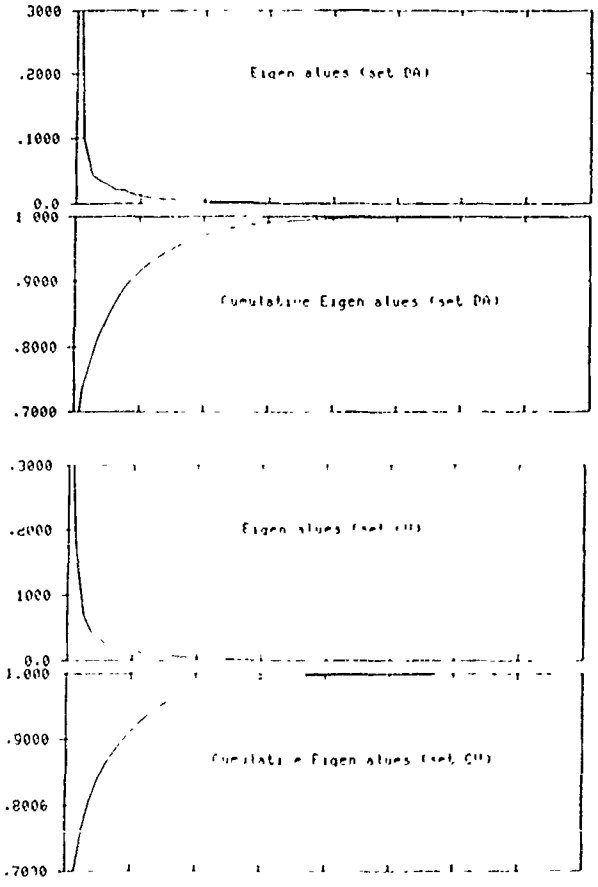
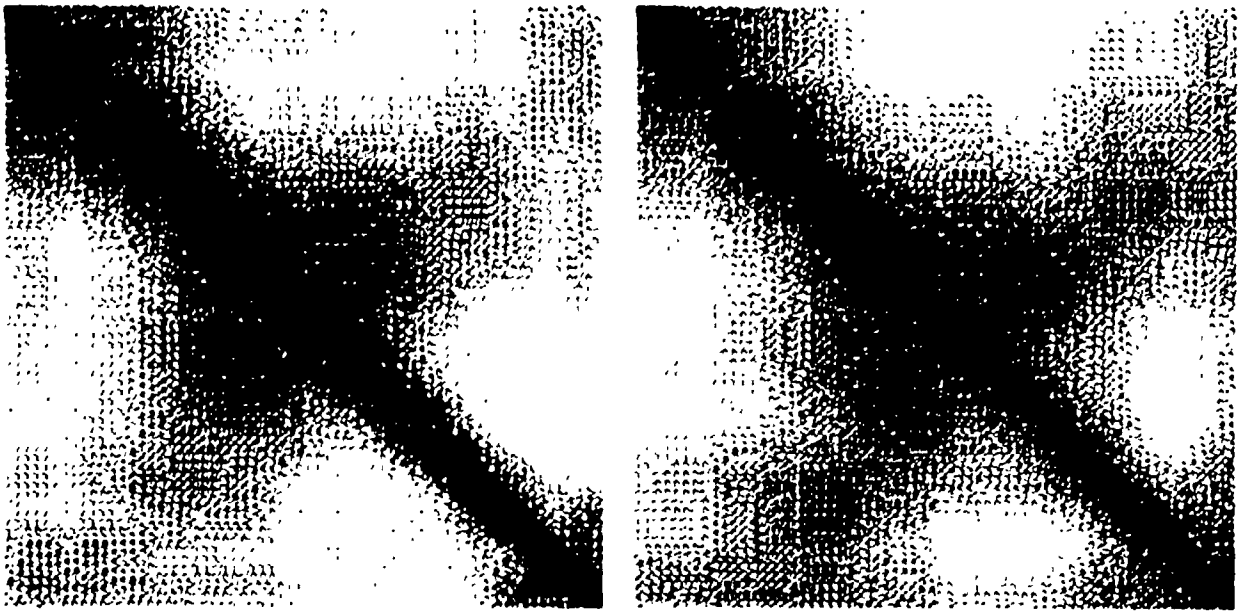


Figure 2.13
Correlation Matrix and Eigenvalue Characteristics
Speech, Magnitude DFT, Linear Frequency



From DA

From CV

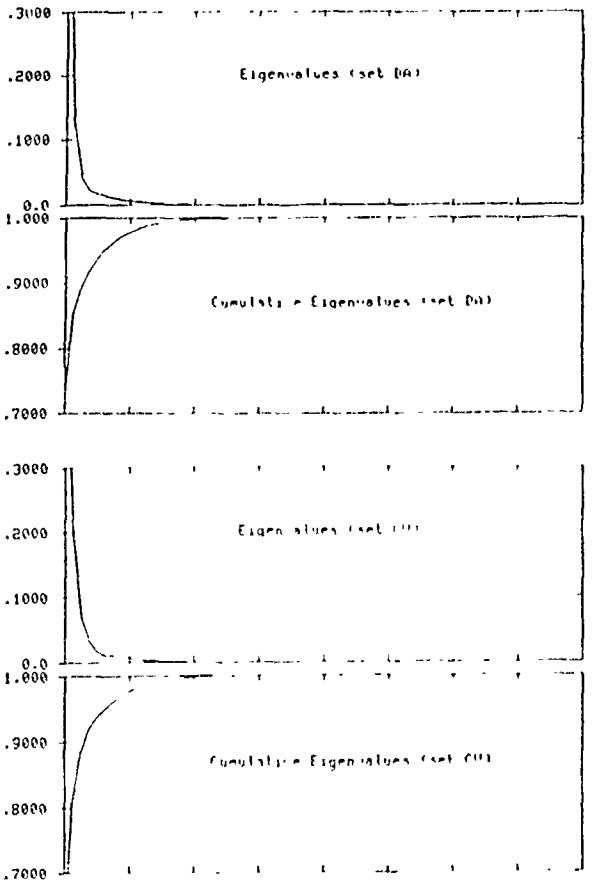
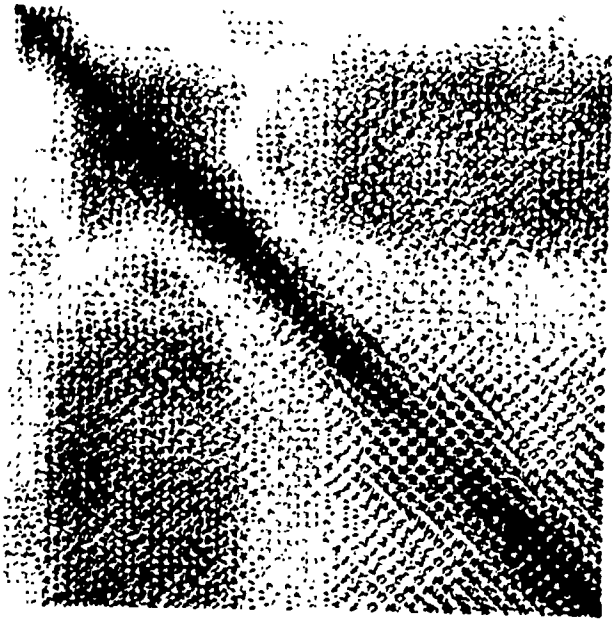


Figure 2.14
Correlation Matrix and Eigenvalue Characteristics
Speech, Magnitude DFT, Log Frequency



From DA

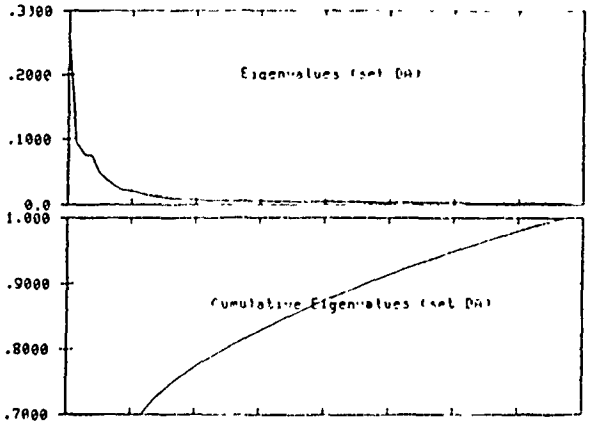
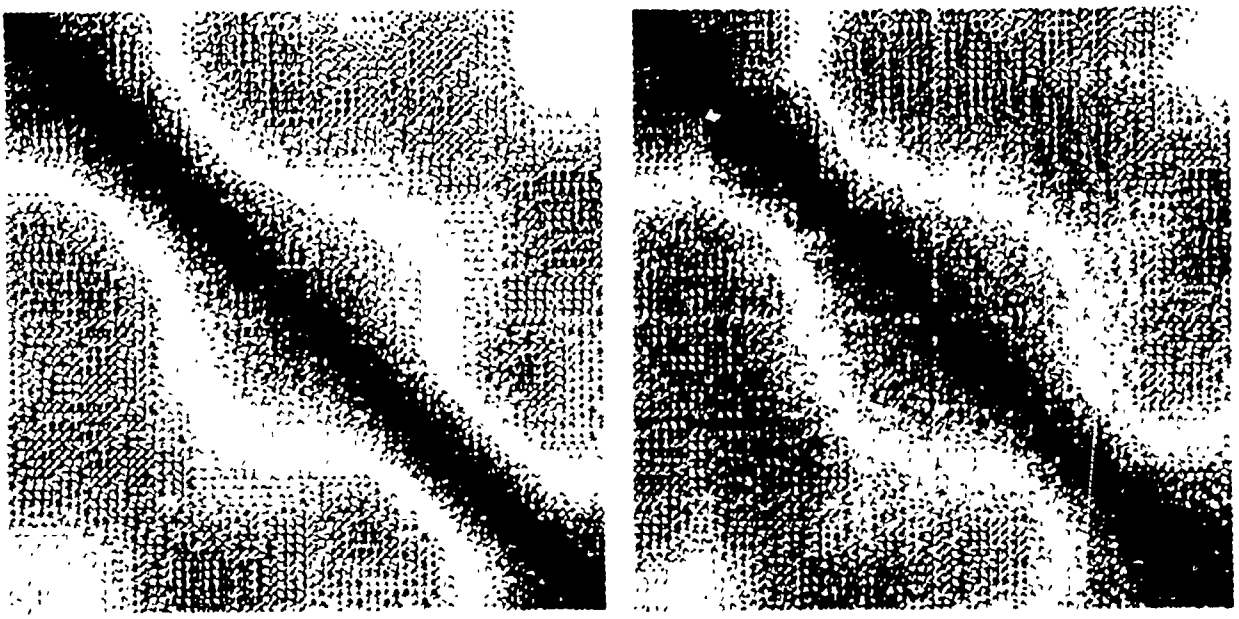


Figure 2.15
Correlation Matrix and Eigenvalue Characteristics
Speech, Log Magnitude DFT, Linear Frequency



From DA

From CV

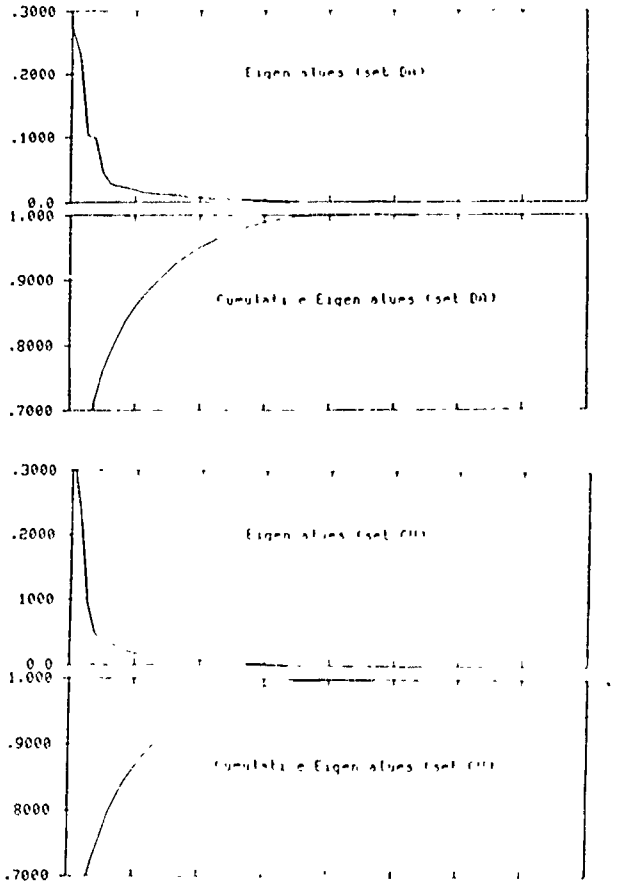
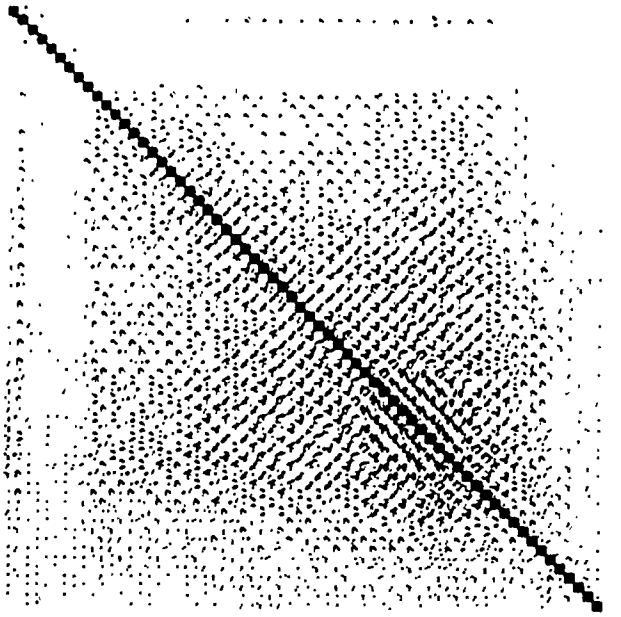


Figure 2.16
Correlation Matrix and Eigenvalue Characteristics
Speech, Log Magnitude DFT, Log Frequency



From DA

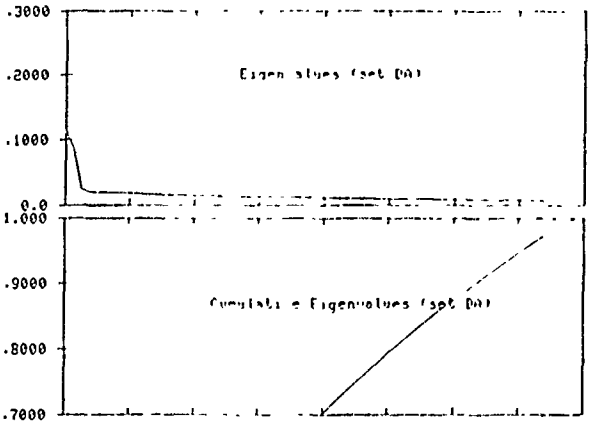


Figure 2.17
Speech, Phase DFT, Linear Frequency

2.4 APPLICATION OF A MODULAR GENERATING FUNCTION APPROACH TO ERROR METRIC DEFINITION

In the process of utilizing the Canonical Coordinate approach, it has become clear that the specification of suitable error metrics for use in the optimization of narrowband voice communication systems is more complicated than was originally anticipated.

The large number of degrees of freedom that are avaiable in directly specifying the error metric K in the frequency domain (or J in the time domain) is such that systematic procedures are needed to accomplish this task. Moreover, it is highly desirable that these procedures be able to make use of as much a priori psychoacoustic information as possible.

The modular generating function (MGF) approach outlined in this section provides a convenient procedure for generating unitary matrices to be considered for the eigenvector based transformation matrix V associated with K (or Y with J).

This is accomplished by using an exponential generating function procedure based on an hermitian matrix G. The structure of the matrix G can be more easily organized (and related to psychoacoustic attributes characterizing the voice spectrum) than that of the more abstract unitary matrix V. This follows since the only mathematical constraint on G is that it be hermitian (i.e. has complex conjugate symmetry) whereas V must satisfy a much more severe set of complex orthogonality constraints.

Moveover, the approach also provides a mechanism for generating K from a series of modular steps, begining with a real diagonal metric Γ, and then adding successively higher order differential variations, based on both G and Γ, that transform Γ into K in a procedure based on

$$K = V\Gamma V^{\dagger} \ .$$

However, in this procedure, the explicit characterization of the sucessive steps can be more easily related to various psychoacoustic attributes of the spectral signal data than in the conventional formutation.

A specific example of a 10 by 10 G matrix with a family of 3 wide overlapping "box-car" type filters is given in Section 2.4.3. In a recent experiment, the MGF procedure utilized a similar family of overlapping "box-car" type filters for the generation of an orthogonal family of transformations operating in the frequency domain. The affect of these transformations was to emphasize formant peaks within a Principal Component based CC speech compression system.

2.4.1 Specification Of An Error Metric K And Associated Eigenvector/Eigenvalue Matrices V And Γ Using Hermitian Generating Function Procedures

Given an hermitian matix K, then there exists a unitary V and a real diagonal matrix Γ satisfying the well known eigenfunction equations,

$$V^{\dagger}KV = \Gamma \quad \text{or} \quad KV = V\Gamma$$

Alternatively, given any unitary matrix V and real diagonal Γ, a hermitian matrix K can be constructed from

$$K = V\Gamma V^{\dagger} .$$

Now this later expression for K can be used to provide considerable insight into the structure of K if we make use of a well known exponential generating function procedure that states that for any unitary V there exists a hermitian G for which $V = \exp\{iG\}$, and conversely.

For convenience in scaling and relating the hermitian matrix G to various physical variables we will introduce a real scalar coefficient k and write the above expansion as

$$V = \exp\{iG/k\} = I + \frac{i}{k}G - \frac{1}{2!k^2}G^2 - \frac{i}{3!k^3}G^3 + \frac{1}{4!k^4}G^4 + \ldots \quad (2.4.1)$$

This expasion is the matrix counterpart of

$$v = e^{ig} = 1 + ig - \frac{1}{2!}g^2 - \frac{i}{3!}g^3 + \frac{1}{4!}g^4 + \ldots$$

relating a real valued variable g to a complex valued v having unit modulus $|v|^2 = 1$.

The importance of the expansion cited in (2.4.1) resides in its ability to characterize the rather complex structure of the unitary matrix V in terms of the much simpler structure of the hermitian matrix G and in the potential for relating G more directly to psychoacoustic attributes involved in the $y = V\beta$ transformation process.

Now expanding K in terms of $V = \exp\{iG/k\}$ yields the power series expansion

$$\begin{aligned} K = V\Gamma V^{\dagger} &= \exp\{iG/k\}\Gamma \exp\{-iG/k\} \\ &= (I + \frac{i}{k}G - \frac{1}{2k^2}G^2 - \frac{i}{6k^3}G^3 + \ldots)\ \Gamma\ (I - \frac{i}{k}G - \frac{1}{2k^2}G^2 + \frac{i}{6k^3}G^3 + \ldots) \end{aligned}$$

To the first order of terms in G, this reduces to

$$K = \Gamma + \frac{i}{k}(G\Gamma - \Gamma G) + O(G^2),$$

and to the 2nd order in G we have

$$K = \Gamma + \frac{i}{k}(G\Gamma - \Gamma G) - \frac{1}{2k^2}(G^2\Gamma - 2G\Gamma G + \Gamma G^2) + O(G^3) .$$

This can also be written, by factoring the 3rd term, as

$$K = \Gamma + \frac{i}{k}(G\Gamma - \Gamma G) - \frac{1}{2k^2}(G(G\Gamma-\Gamma G)-(G\Gamma-\Gamma G)G) + O(G^3). \quad (2.4.2)$$

If we now let $\delta_G(\Gamma)$ be the 1st order variation in Γ, generated by the hermitian operator G, and specified as

$$\boxed{\delta_G(\Gamma) = \frac{i}{k}(G\Gamma - \Gamma G)} , \quad (2.4.3)$$

then K, as expressed in (2.4.2) can be developed in the Taylor-series like expansion (around the diagonal matrix Γ) of the form

$$K = \Gamma + \delta_G(\Gamma) + \frac{i}{2k}(G\delta_G(\Gamma) - \delta_G(\Gamma)G) + O(G^3)$$

or

$$\boxed{K = \Gamma + \delta_G(\Gamma) + \frac{1}{2!}\delta_G(\delta_G(\Gamma))+ O(G^3) = e^{\delta_G}(\Gamma)} . \quad (2.4.4)$$

This series expansion of K in terms of successive differential variations, generated by an hermitian operator G, operating on a diagonal matrix Γ, provides a mechanism for generating K from a concatenation of elementary matrix operations.

2.4.2 Special Characteristics Of The Generating Function Based Differential Operator $\delta_G(\Gamma)$

The first order differential operator $\delta_G(\Gamma)$ was defined in (2.4.3), for hermitian G, real diagonal Γ and real scalar k, and can be written as

$$\delta_G(\Gamma) = \frac{i}{k}(G\Gamma - \Gamma G) = \frac{i}{k}[G, \Gamma] ,$$

in terms of the commutator operation $[G, \Gamma] = G\Gamma - \Gamma G$ of G and Γ.

Note that the commutation notation is applicable to all matrices, whether hermitian or not. However, as used in representing $\delta_G(\Gamma)$, both G and Γ must be hermitian, with Γ further restricted to being diagonal.

The above defined $\delta_G(\Gamma)$ is easily shown to be an hermitian operator, since

$$\delta_G^\dagger(\Gamma) = -\frac{i}{k}((G\Gamma)^\dagger - (\Gamma G)^\dagger) = -\frac{i}{k}(\Gamma G - G\Gamma) = \frac{i}{k}(G\Gamma - \Gamma G) = \delta_G(\Gamma)$$

and is also linear in Γ in the sense that for hermitian Γ and Ω and scalar ω

$$\delta_G(\Gamma + \Omega) = \delta_G(\Gamma) + \delta_G(\Omega) \text{ and } \delta_G(\omega\Gamma) = \omega\delta_G(\Gamma) \ .$$

In particular, however, $\delta_G(\Gamma)$ has the important characteristic of a first order differential operator [i.e. $D(uv) = Du \circ v + u \circ Dv$] in the sense that

$$\delta_G(\Gamma\Omega) = \delta_G(\Gamma)\Omega + \Gamma\delta_G(\Omega) \ .$$

This follows from the Jacobi Identity for Commutators, which states that $[G, \Gamma\Omega] = [G, \Gamma]\Omega + \Gamma[G, \Omega]$.

2.4.3 Explicit Characterization Of $\delta_G(\Gamma)$ And $\delta_G(\delta_G(\Gamma))$

From the explicit characterization of an hermitian G

$$G = \begin{vmatrix} G_{00} & G_{01} & G_{02} & G_{0N} \\ G_{10} & G_{11} & G_{12} & G_{1N} \\ G_{20} & G_{21} & G_{22} & G_{2N} \\ & & & \\ G_{N0} & G_{N1} & G_{N2} & G_{NN} \end{vmatrix} = \begin{vmatrix} G_{00} & G_{01} & G_{02} & G_{0N} \\ G_{10}^* & G_{11} & G_{12} & G_{1N} \\ G_{20}^* & G_{21}^* & G_{22} & G_{2N} \\ & & & \\ G_{N0}^* & G_{N1}^* & G_{N2}^* & G_{NN} \end{vmatrix}$$

and $\Gamma = \text{diag}(\gamma_0, \gamma_1, \ldots \gamma_N)$.

we have $\delta_G(\Gamma) = \frac{i}{k}[G, \Gamma] =$

$$\frac{i}{k}\begin{vmatrix} 0 & -G_{01}(\gamma_0-\gamma_1) & -G_{02}(\gamma_0-\gamma_2) & & -G_{0N}(\gamma_0-\gamma_N) \\ G_{01}^*(\gamma_0-\gamma_1) & 0 & -G_{12}(\gamma_1-\gamma_2) & & -G_{1N}(\gamma_1-\gamma_N) \\ G_{02}^*(\gamma_0-\gamma_2) & G_{12}^*(\gamma_1-\gamma_2) & 0 & & -G_{2N}(\gamma_2-\gamma_N) \\ & & & & \\ G_{0N}^*(\gamma_0-\gamma_N) & G_{1N}^*(\gamma_1-\gamma_N) & & G_{N-1N}^*(\gamma_{N-1}-\gamma_N) & 0 \end{vmatrix}$$

Note that while the commutator itself, $[G, \Gamma]$ is skew hermitian, i.e. $[G,\Gamma]^\dagger = -[G,\Gamma]$, the prefix coefficient i/k makes $\delta_G(\Gamma)$ hermitian.

For the 2nd order differential operator $\delta_G(\delta_G(\Gamma))$ we have the following characterization

$$\delta_G(\delta_G(\Gamma)) = \frac{i}{k}[G,\ \delta_G(\Gamma)] = \frac{i}{k}(G\delta_G(\Gamma) - \delta_G(\Gamma)G)$$

$$= \frac{i}{k}(G\frac{i}{k}(G\Gamma - \Gamma G) - \frac{i}{k}(G\Gamma - \Gamma G)G)$$

or $$\boxed{\delta_G(\delta_G(\Gamma)) = -\frac{1}{k^2}(G^2\Gamma - 2G\Gamma G + \Gamma G^2)} \qquad (2.4.5)$$

2.4.4 Example Of Construction Of V When G Corresponds To A Family Of Overlapping Rectangular "Box-Car" Filters

This example has a generating function G that resembles a sequence of rectangular bandpass (Box-Car) filters in the frequency domain that overlap by 50% and have a uniform width of 3 units each.

Equation (2.4.6)

$$G = (\frac{i}{k})\begin{vmatrix} 1&1&0&0&0&0&0&0&0&1\\ 1&1&1&0&0&0&0&0&0&0\\ 0&1&1&1&0&0&0&0&0&0\\ 0&0&1&1&1&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0&0\\ 0&0&0&0&1&1&1&0&0&0\\ 0&0&0&0&0&1&1&1&0&0\\ 0&0&0&0&0&0&1&1&1&0\\ 0&0&0&0&0&0&0&1&1&1\\ 1&0&0&0&0&0&0&0&1&1 \end{vmatrix} \qquad G^2 = (-\frac{1}{2k^2})\begin{vmatrix} 3&2&1&0&0&0&0&0&1&2\\ 2&3&2&1&0&0&0&0&0&1\\ 1&2&3&2&1&0&0&0&0&0\\ 0&1&2&3&2&1&0&0&0&0\\ 0&0&1&2&3&2&1&0&0&0\\ 0&0&0&1&2&3&2&1&0&0\\ 0&0&0&0&1&2&3&2&1&0\\ 0&0&0&0&0&1&2&3&2&1\\ 1&0&0&0&0&0&1&2&3&2\\ 2&1&0&0&0&0&0&1&2&3 \end{vmatrix}$$

$$G^3 = (-\frac{i}{6k^3})\begin{vmatrix} 7&6&3&1&0&0&0&1&3&6\\ 6&7&6&3&1&0&0&0&1&3\\ 3&6&7&6&3&1&0&0&0&1\\ 1&3&6&7&6&3&1&0&0&0\\ 0&1&3&6&7&6&3&1&0&0\\ 0&0&1&3&6&7&6&3&1&0\\ 0&0&0&1&3&6&7&6&3&1\\ 1&0&0&0&1&3&6&7&6&3\\ 3&1&0&0&0&1&3&6&7&6\\ 6&3&1&0&0&0&1&3&6&7 \end{vmatrix}$$

$$G^4 = (\frac{1}{24k^4})\begin{vmatrix} 19&16&10&4&1&0&1&4&10&16\\ 16&19&16&10&4&1&0&1&4&10\\ 10&16&19&16&10&4&1&0&1&4\\ 4&10&16&19&16&10&4&1&0&1\\ 1&4&10&16&19&16&10&4&1&0\\ 0&1&4&10&16&19&16&10&4&1\\ 1&0&1&4&10&16&19&16&10&4\\ 4&1&0&1&4&10&16&19&16&10\\ 10&4&1&0&1&4&10&16&19&16\\ 16&10&4&1&0&1&4&10&16&19 \end{vmatrix}$$

For the first row of each power of G we have:

n =	0	1	2	3	4	5	6	7	8	9
$G_{0n} = (\frac{i}{k})$	(1	1	0	0	0	0	0	0	0	1)
$G^2_{0n} = (\frac{-1}{2k^2})$	(3	2	1	0	0	0	0	0	1	2)
$G^3_{0n} = (\frac{-i}{6k^3})$	(7	6	3	1	0	0	0	1	3	6)
$G^4_{0n} = (\frac{1}{24k^4})$	(19	16	10	4	1	0	1	4	10	16)

From the expansion of V in terms of G and k to order G^4 we have

$$V = \exp(iG/k) = I + \frac{i}{k}G - \frac{1}{2k^2}G^2 - \frac{i}{6k^3}G^3 + \frac{1}{24k^4}G^4 + O(G^5) \quad (2.4.7)$$

hence the first row of V has, for its first 5 components, the values:

Equation (2.4.8)

n	G^0_{0n}	G^1_{0n}	G^2_{0n}	G^3_{0n}	G^4_{0n}	G^5_{0n}	REAL	IMAG
V_{00} =	1	$+\frac{i}{k}$	$-\frac{3}{2k^2}$	$-\frac{7i}{6k^3}$	$+\frac{19}{24k^4}$	$+O(G^5)$	$=(1-\frac{3}{2k^2}+\frac{19}{24k^4})$	$+\frac{i}{k}(1-\frac{7}{6k^2})+O(G^5)$
V_{01} =		$+\frac{i}{k}$	$-\frac{2}{2k^2}$	$-\frac{6i}{6k^3}$	$+\frac{16}{24k^4}$	$+O(G^5)$	$=(-\frac{1}{k^2}+\frac{2}{3k^4})$	$+\frac{i}{k}(1-\frac{1}{k^2})+O(G^5)$
V_{02} =			$-\frac{1}{2k^2}$	$-\frac{3i}{6k^3}$	$+\frac{10}{24k^4}$	$+O(G^5)$	$=(-\frac{1}{2k^2}+\frac{5}{12k^4})$	$-\frac{i}{2k^3}+O(G^5)$
V_{03} =				$-\frac{i}{6k^3}$	$+\frac{4}{24k^4}$	$+O(G^5)$	$=\frac{1}{6k^4}$	$-\frac{i}{6k^3}+O(G^5)$
V_{04} =					$+\frac{1}{24k^4}$	$+O(G^5)$	$=\frac{1}{24k^4}$	$+0+O(G^5)$

From the circular symmetry exhibited by the particular G, and each of its powers, shown in (2.4.6), it follows from (2.4.7) that V has the same circular symetric property, and therefore can be characterized by a single row (or column) vector of the type illustrated in (2.4.8). Moreover, the complex conjugate matrix $V^\dagger$ will have the same property, and hence from

$$\beta = V^\dagger y, \text{ or } \beta_n = v^\dagger(n)y, \text{ where } v^*_m(n) = V^*_{nm},$$

we can consider $\{v^\dagger(n)\}$ as a family of circularly shifted row vectors of $V^\dagger$, operating as "filters" on y to yield the filtered canonical coordinate β. A typical version of $v^\dagger(n)$, centered on 0, will have the form whose real components are illustrated in Figure 2.18.

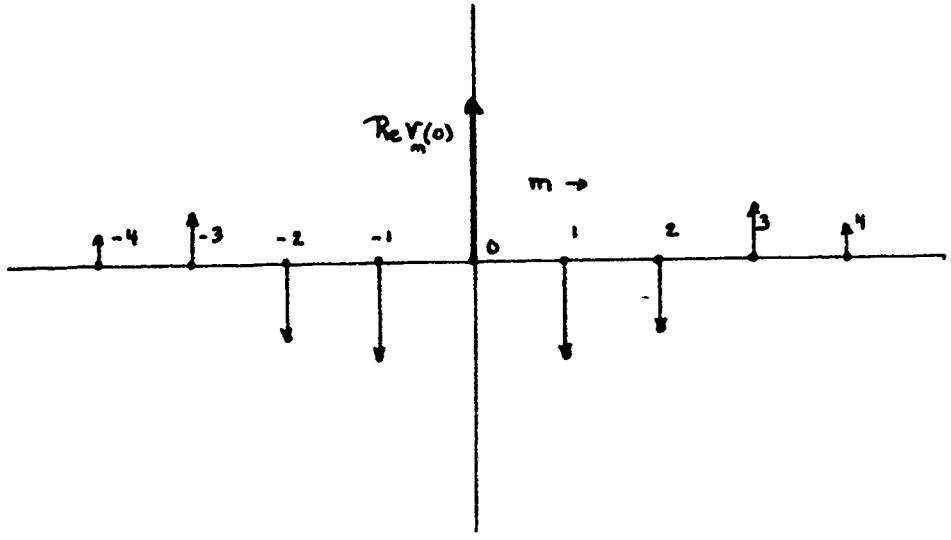


Figure 2.18
Digital Box-Car Filter (Real Components)

2.5 CANONICAL COORDINATE APPROACH TO INTEROPERABILITY

The Canonical Coordinate (CC) approach permits speech compression systems optimization with respect to any arbitrary quadratic error metric, in accordance with a very specific set of signal data transformation construction rules that go from highly cross-associated signal input (x-domain) data representations to completely uncorrelated (ξ, δ, z domain) representations. It is this facility for systematically transforming between a number of signal parameter representations of the same underlying digital voice data that holds the greatest promise for achieving the highest possible degree of interoperability between different speech compression systems.

However, it must be noted that a high degree of interoperability between different speech compression systems is also highly contingent upon the degree of compatability between the error metrics used in the optimization process associated with each class of system. To the extent that there is serious incompatability between the error metrics used, there is likely to be difficulty in achieving a satisfactory degree of interoperability between the different classes of systems.

In the majority of speech compression systems the error metric will be some variant of either the mean-square-error (i.e. L^2) metric or of some other L^p metric. In particular, most LPC-type systems are based on minimizing a mean-square unit prediction error. This translates, in terms of an N-dimensional vector data frame, into an error metric $J=R^{-1}$ where R is an N-dimensional correlation matrix in the time domain, and to an error metric $K=S^{-1}$ in the frequency domain, where S is an N-dimensional spectral matrix. A detailed proof of this process is provided in this section. On the other hand, the error metric associated with Principal Component, or Loeve-Karhunen, type vocoders [Ref. 16.] is well known to be simple euclidean in form, i.e. J = I (hence also K = I).

2.5.1 Canonical Coordinate Background

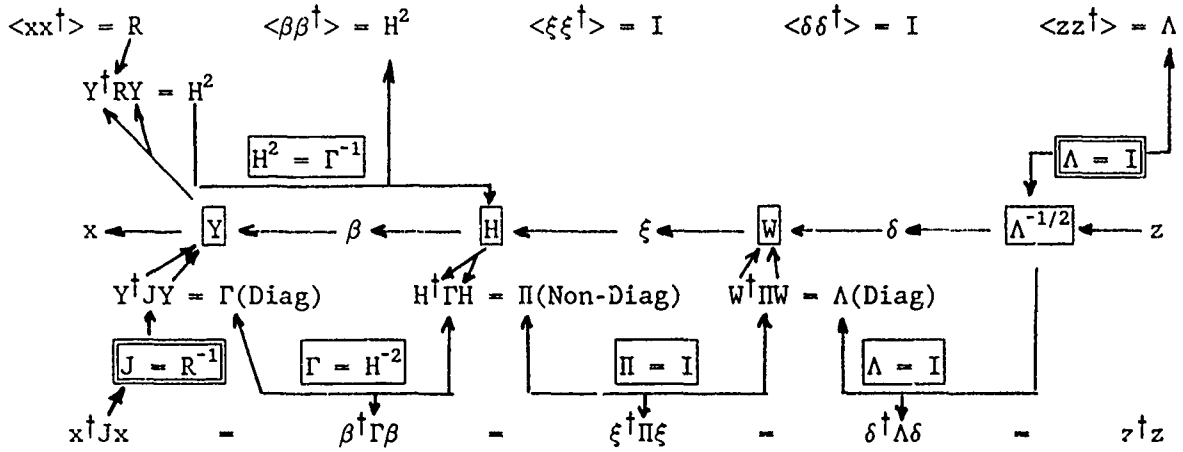
Following the notation in Figure 2.19, let D be a matrix defining a non-orthogonal transformation from a "canonical" coordinate component vector z to the "raw signal vector x, i.e.

$$\boxed{x = Dz = YHW\Omega^{-1/2}z} \qquad (2.5.1)$$

where x satisfies a non-euclidean "orthonormalization" condition, based on a specified hermitian matrix J, of the form $x^{\dagger}Jx = 1$ and z satisfies an euclidean orthonormalization condition $z^{\dagger}z = 1$.

Hence D satisfies a basic complex conjugate matrix equation of the form

$$\boxed{D^{\dagger}JD = I \quad \text{or} \quad J = D^{\dagger-1}D^{-1}} \; . \qquad (2.5.2)$$



- The Spectral Whitening Assumption $\Lambda = I$ is Equivalent to the Error Metric Assumption $J = R^{-1}$.

- The General CC Decoposition, $x = Dz = YHW\Lambda^{-1/2}z$, Reduces to $x = Y\Gamma^{-1/2}Wz$, Where Y and Γ Satisfy the Eigenvector Equation $Y^{\dagger}RY = \Gamma^{-1}$ with Unitary Y and Diagonal Γ, and Where W is an Arbitrary Unitary Matrix.

- Detailed Specification of W Depends Upon Specific AR/LPC Type Structural Constraints on $D = Y\Gamma^{-1/2}W$, such as Triangularity.

Figure 2.19

Canonical Coordinate Technique Via Error Metric Diagonalization

Applied to Spectral Whitening Procedures of the AR/LPC Type

If we approximate x by an estimate $\hat{x}(p)$ by transmitting only the first p components of z and replacing the remaining N - p components of z by preselected estimates $\hat{z}_p, \ldots, \hat{z}_{N-1}$, then it can be shown that minimizing the non-euclidean error E(p), defined as

$$E(p) = \langle \Delta x^t(p)\, J\, \Delta x(p) \rangle = \langle [x - \hat{x}(p)]^t J [x - \hat{x}(p)] \rangle$$

where $\hat{x}(p) = DI_p z + D(I - I_p)\hat{z}$ with $I_p = \text{diag}(1_0, \ldots, 1_{p-1}, 0_p, \ldots, 0_{N-1})$ can be written

$$E(p) = \text{tr}(I - I_p)\Lambda$$

where Λ satisfies a generalized eigenvalue equation of the form

$$\boxed{JRJD = JD\Lambda \quad \text{or} \quad D^tJRJD = \Lambda} \tag{2.5.3}$$

where R is the covariance matrix

$$R = \langle (x - \langle x \rangle)(x - \langle x \rangle)^t \rangle$$

or $R = \langle xx^t \rangle$ when $\langle x \rangle = 0$

and Λ is a diagonal matrix whose diagonal elements are ordered as $\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{N-1}$.

Now for any positive diagonal scaling matrix Ω whatsoever we may define

$$A = \Omega^{-1/2}D^{-1} \quad \text{and} \quad B = A^{-1}$$

and then write

$$\boxed{z = D^{-1}x = \Omega^{-1/2}\, Ax} \tag{2.5.4}$$

where

$$\boxed{D = B\Omega^{1/2}} \ . \tag{2.5.5}$$

Then (2.5.2) can be written $J = A^t\Omega^{-1}A$ and (2.5.3) reduces to

$$\boxed{ARA^t = \Omega\Lambda} \tag{2.5.6}$$

or $RA^t = B(\Omega\Lambda)$ (2.5.7)

or $\boxed{R = B(\Omega\Lambda)B^t}$ (2.5.8)

and, in particular, $R^{-1} = A^t(\Omega\Lambda)^{-1}A$.

It is also important to note, from (2.5.2) and (2.5.3) that

$$\boxed{R = D\Lambda D^t} \tag{2.5.9}$$

and from $R = \langle xx^t \rangle = D\langle zz^t \rangle D^t$, that

$$\boxed{\Lambda = \langle zz^t \rangle} \ . \tag{2.5.10}$$

The diagonal scaling matrix Ω specified in (2.5.5) above is necessary in certain situations, such as that encountered in AR/LPC analysis where the matrices A and B must be both triangular and also scaled so that they have 1's on the main diagonal. In this situation, it is shown in Section 2.5.2, that both $\Lambda = I$ and $J = R^{-1}$, thus leading to the Cholesky-type decomposition $R = B\Omega B^t$ cited in (2.5.18).

If, however, we are not constrained to have D be triangular, and hence neither A nor B, then the decomposition of R specified by (2.5.9) will not be of the Cholesky type and will not in general have a completely recursive solution. In fact, the solution will not be uniquely specified until a constraint of the form specified by (2.5.2), involving an error metric J is specified.

2.5.2 AR/LPC Specification In Terms Of A 1-Dimensional Projection Of An N-Dimensional Canonical Coordinate Process Based On Triangular D^{-1}, Spectral Whitening $\Lambda=I$ And $J=R^{-1}$

The typical AR/LPC process can be considered as the minimization, in the mean-square-error sense, of the difference $x_0 - \hat{x}_0$ between a variable x_0 and a linear estimate

$$\hat{x}_0 = -\sum_{i=1}^{M} a_i x_i = -\tilde{a}^t \tilde{x}$$

of the next M terms in a sequence

$$\tilde{x} = (x_1, \ldots, x_M)^t$$

where $\tilde{a}^t = (a_1, \ldots, a_M)$.

This can be characterized as the minimization of $\langle |x_0 - \hat{x}_0|^2 \rangle$ subject to the constraint that $a_0 = 1$ or, equivalently, as the minimization of

$$\langle |x_0 - \hat{x}_0|^2 \rangle = \langle |x_0 - \tilde{a}^t \tilde{x}|^2 \rangle = \omega_0 \langle |z_0|^2 \rangle \qquad (2.5.11)$$

subject to the constraint that

$$\boxed{\langle |z_0|^2 \rangle = 1} \qquad (2.5.12)$$

where, from (2.5.4), z_0 can be expressed as

$$z_0 = \omega_0^{-1/2}(x_0 - \tilde{a}^t \tilde{x}) = \omega_0^{-1/2} a^t x \ . \qquad (2.5.13)$$

Here x and a are the N = M+1 dimensional vectors

$$x = (x_0, x_1, \ldots, x_M)^t \ ,$$
$$a = (1, a_1, \ldots, a_M)^t \ .$$

Note, in particular, that (2.5.13) can be considered as the projection, P_0 on the 0th vector component of the N dimensional canonical coordinate equation (cf 2.5.4)

$$\boxed{z = D^{-1}x = \Omega^{-1/2}Ax} \tag{2.5.14}$$

i.e. $z_0 = P_0 z = P_0\Omega^{-1/2}Ax = \omega_0^{-1/2}a^{\dagger}x$,

and from (2.5.11) and (2.5.12) that $\omega_0 = \min\langle|x_0 - \hat{x}_0|^2\rangle$.

Here A, for purposes of recursive computational convenience, must be assumed to be upper triangular in form

$$\boxed{A = \begin{bmatrix} 1 & \tilde{a}^{\dagger} \\ 0 & \tilde{A} \end{bmatrix}} \tag{2.5.15}$$

and

$$P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \mathrm{diag}(1_0, 0_1, \ldots, 0_M)$$

In terms of (2.5.10) and (2.5.14) we have

$$\Lambda = \langle zz^{\dagger}\rangle = \Omega^{-1/2}A\langle xx^{\dagger}\rangle A^{\dagger}\Omega^{-1/2} = \Omega^{-1/2}ARA^{\dagger}\Omega^{-1/2}. \tag{2.5.16}$$

The AR/LPC constraint specified by (2.5.12), i.e. $\langle|z_0|^2\rangle = 1$, is simply the projection $P_0\Lambda P_0 = P_0\langle zz^{\dagger}\rangle P_0 = \langle|z_0|^2\rangle = 1$ of the general constraint that $\Lambda = I$.

This says that the general vector-based AR/LPC process carries out an inverse filtering operation $z = D^{-1}x$ $(= \Omega^{-1/2}Ax)$ in which z has a flat uncorrelated spectrum $\langle zz^{\dagger}\rangle = \Lambda = I$.

Hence (2.5.16) reduces to $ARA^{\dagger} = \Omega$.

And consequently the general canonical coordinate Eqs. (2.5.7) and (2.5.8) become, in the AR/LPC context

$$RA^{\dagger} = B\Omega \tag{2.5.17}$$

and $$R = B\Omega B^{\dagger} \tag{2.5.18}$$

where, since A is upper triangular, as specified in (2.5.15), this inverse B must also be upper triangular, i.e.

$$\boxed{B = \begin{bmatrix} 1 & \tilde{b}^{\dagger} \\ 0 & \tilde{B} \end{bmatrix}} \; . \tag{2.5.19}$$

Finally, we note, from (2.5.18), that in the AR/LPC solution

$$R^{-1} = A^{\dagger}\Omega^{-1}A$$

and since, from (2.5.2) and (2.5.5), we also have

$$J = D^{t-1}D^{-1} = (A^t\Omega^{-1/2})(\Omega^{-1/2}A) = A^t\Omega^{-1/2}A,$$

it follows that

$$\boxed{J = R^{-1}}$$

is the metric in the x-coordinate system that characterizes the general vector-based AR/LPC process.

2.5.3 Determination Of Predictor Coefficient Vector $\tilde{a}$ And Reflection Coefficient Vector $\tilde{b}$ from Canonical Coordinate Equations for the AR/LPC Situation

In Section 2.5.2 we saw that the basic Canonical Coordinate equations in the AR/LPC context could be put in the form (2.5.17) of an Nth order matrix equation

$$\boxed{RA^t = B\Omega} \qquad (2.5.20)$$

where both A and R ($= A^{-1}$) are upper triangular matrices having 1's on their main diagonals, specified in (2.5.15) and (2.5.19) as

$$A = \begin{bmatrix} 1 & \tilde{a}^t \\ 0 & \tilde{A} \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & \tilde{b}^t \\ 0 & \tilde{B} \end{bmatrix}.$$

Because of the triangular nature of A and B, equation (2.5.20) has a recursive solution for these matrices and for the diagonal matrix Ω. The following partitioned expansion of the matrix equation yields the well known interrelationships between a, b, ω_0, Ω, $\tilde{A}$, $\tilde{B}$ and R, r_0, $\tilde{r}$, $\tilde{R}$:

$$\begin{bmatrix} r_0 & \tilde{r}^t \\ \tilde{r} & \tilde{R} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \tilde{a} & \tilde{A}^t \end{bmatrix} = \begin{bmatrix} 1 & \tilde{b}^t \\ 0 & \tilde{B} \end{bmatrix}\begin{bmatrix} \omega_0 & 0 \\ 0 & \tilde{\Omega} \end{bmatrix}$$

or

$$\begin{bmatrix} r_0 + \tilde{r}^t\tilde{a} & \tilde{r}^tA^t \\ \tilde{r} + \tilde{R}\tilde{a} & \tilde{R}\tilde{A}^t \end{bmatrix} = \begin{bmatrix} \omega_0 & \tilde{b}^t\tilde{\Omega} \\ 0 & \tilde{B}\tilde{\Omega} \end{bmatrix}.$$

Thus we obtain

1. A scalar relationship characterizing the Mth order AR/LPC error ω_0.

$$\omega_0 = r_0 + \tilde{r}^t\tilde{a}$$

2. A vector relationship characterizing the Mth order prediction coefficient $\tilde{a}$.

$$\boxed{\tilde{R}\tilde{a} = -\tilde{r} \quad \text{or} \quad \tilde{a} = -\tilde{R}^{-1}\tilde{r}} \qquad (2.5.21)$$

3. A vector relationship characterizing the Mth order reflection coefficient $\tilde{b}$.

$$\tilde{A}\tilde{r} = \tilde{\Omega}\tilde{b} \quad \text{or} \quad \tilde{b} = \tilde{\Omega}^{-1}\tilde{A}\tilde{r} \ .$$

4. An Mth order matrix equation for use in the recursive determination of $\tilde{A}$, $\tilde{B}$, and $\tilde{\Omega}$ from Mth order $\tilde{R}$.

$$\tilde{R}\tilde{A}^{t} = \tilde{B}\tilde{\Omega}$$

From the above basic AR/LPC relationships we can also express ω_0 as

$$\omega_0 = r_0 - \tilde{a}^{t}\tilde{R}\tilde{a} = a^{t}Ra$$

or, in terms of $\tilde{b}$ as

$$\boxed{\omega_0 = r_0 - \tilde{b}^{t}\tilde{\Omega}\tilde{b} \quad \text{and} \quad r_0 = b^{t}\Omega b} \qquad (2.5.22)$$

Note also the following relationships between $\tilde{a}$ and $\tilde{b}$

$$\tilde{b} = -\tilde{B}^{t}\tilde{a} \quad \text{and} \quad \tilde{a} = -\tilde{A}^{t}\tilde{b} \ .$$

2.5.4 Characterization Of Extended Predictor Coefficient Vector a in Terms of Both Time Domain Matrices J and Frequency Domain Metric K.

In terms of the extended predictor coefficient vector a, of dimension N = M+1, defined as $a = (1, a_1, \ldots, a_M)^{t} = (1, \tilde{a}^{t})^{t}$, we can rewrite (2.5.21) as

$$\boxed{Ra = \omega_0\rho \quad \text{or} \quad a = \omega_0 R^{-1}\rho} \ . \qquad (2.5.23)$$

where R is an N dimensional correlation matrix and ρ is a "unit impulse" vector

$$\boxed{\rho = (1, 0, \ldots, 0)^{t} \quad \text{and} \quad \rho^{t}\rho = 1}$$

Thus the vector a is ω_0 times the first column of the inverse matrix R^{-1}, or, since $J = R^{-1}$, we have

$$\boxed{a = \omega_0 J\rho} \ . \qquad (2.5.24)$$

Now upon noting that the counterparts of R and J in the time domain are the spectral matrix S and metric K in the frequency domain, given by

$$S = ZRZ^{\dagger} \text{ and } K = ZJZ^{\dagger}$$

it follows, if we let c be the Fourier transform of a, i.e.,

$$\boxed{c = Za \quad \text{or} \quad a = Z^{\dagger}c}$$

that (2.5.23) can be written

$$\boxed{Sc = \omega_0 \sigma \quad \text{or} \quad c = \omega_0 S^{-1}\sigma} \qquad (2.5.25)$$

where σ is the flat spectrum" vector in the frequency domain corresponding to the "unit impulse" vector ρ in the time domain, i.e.,

$$\boxed{\sigma = \frac{1}{\sqrt{N}}(1,\ 1,\ldots,\ 1)^{t} = Z\rho \quad \text{where} \quad \sigma^{\dagger}\sigma = 1}\ .$$

Moreover, the counterparts of (2.5.24) and (2.5.22) are, in the frequency domain, that

$$\boxed{c = \omega_0 K\sigma} \qquad (2.5.26)$$

and $\boxed{\omega_0 = c^{\dagger}Sc}$ (2.5.27)

since, from $J = R^{-1}$, it follows that $K = S^{-1}$.

The particular significance of the expression for c and ω_0 in (2.5.25 - 2.5.27), arises from the fact that S, and hence also K ($=S^{-1}$) are diagonal, when R is circularly symetric, and therefore S^{-1} has the simple characterization

$$\boxed{S^{-1} = \text{diag}(1/s_0,\ 1/s_1,\ldots,\ 1/s_{N-1})} \qquad (2.5.28)$$

CHAPTER 3

ALGORITHM RESEARCH AND IMPLEMENTATION

Algorithms implemented during this contract period have included two versions of the Canonical Coordinate Speech Compression System and two versions of the NSA LPC-10E Linear Predictive Coding Vocoder.

A faster AP-120B implementation of the CC algorithm was developed to aid in Error Metric research and in quantization/coding studies. A fully parallel implementation of the CC algorithm on a Systolic Array Processor was developed to demonstrate the versatility of the algorithm when combined with a truly parallel processing architecture. The algorithm research and development required by this task was reported by ARCON at ICASSP-87 [Ref. 14].

A true vocoder implementation of the CC method requires research into parameter quantization effects and the development of a coding capability for the transmission of CC coefficients between analysis and synthesis processors. Several development tools have been generated during this contract period to aid in this work. Because they are closely related to the CC algorithm implementations, they will be included in this Chapter.

The emergence of the FSVT and STU-III programs during this contract period led to the necessity of an operational version of the LPC-10E algorithm at RADC/EEV. Two versions of this algorithm (V49 and V52) were received from NSA at separate times in PDP VAX Fortran form. They were ported to the RADC/EEV PDP-11 Fortran IV system by ARCON.

3.1 CANONICAL COORDINATE ALGORITHM AP-120B IMPLEMENTATIONS

ARCON has continued to improve the RADC/EEV implementation of the Canonical Coordinate Algorithm on the FPS AP-120B located at the speech lab. This implementation has undergone a significant number of changes and has evolved into two separate routines. Both routines are Fortran programs which utilize AP-120B library function calls optimized for AP program memory use with the Vector Function Chainer (VFC) language. Speech input, CC parameters and synthesized speech output are all stored and/or accessed from Speech Data Base files.

The first routine, CCANL1, implements the analysis portion of the algorithm. It will:

1. input an error metric transformation matrix and eigenvalues
2. loop over a user specified number of input speech data frames,
3. condition the input signal,
4. calculate the pseudo CC parameters beta and the permutation vector based on the CC parameter mu,
5. code and quantize the speech coefficients (magnitudes and phases of the pseudo CC parameters beta, the inverse permutation vector and the signal vector scale factors) and pack them into the output file.

The second routine, CCSYN1, implements the synthesis portion of the algorithm. It will:

1. input an error metric inverse transformation matrix and eigenvalues
2. unpack and decode speech coefficients,
3. further quantize phases if requested,
4. synthesize an estimate of the input signal based on any requested percent of the ordered beta's.

The splitting of the algorithm into these two routines has permitted more efficient testing of speech coefficient compression. The analyst no longer has to re-analyze speech data to study the effects of varying the amount of beta synthesis. Furthermore, the breakup has enabled ARCON to study the effects of quantization and coding of the speech coefficients on the performance of the algorithm.

With this latest implementation, significant improvements in performance have been realized. The original implementation of this algorithm, CCVOC, executed at approximately 25X real time. The throughput of the latest implementation is approximately 5X real time (3X real time for the analysis routine and 2X real time for the synthesis routine). Several changes in the manner in which the analysis and synthesis routines execute are responsible for this performance improvement. The basic idea is to distribute the processing between the AP and the PDP-11 and to have them process data simultaneously. The first step in this process was the separation of the CCVOC algorithm into analysis and synthesis routines joined by a CC parameter file. Speed improvements to the analysis and synthesis routines were then implemented. Descriptions of these improvements are given below along with operational instructions for the analyst.

3.1.1 CC Analysis Improvements

The previous version of CCANL1 read one block of data at a time and converted it to floating point representation. This input buffer was then loaded into the AP-120B one frame (64 words) at a time and analyzed by the AP routines DECOMN and PCCZ. The speech parameters were retrieved from the AP, encoded and packed into the output buffer, and written as necessary to disk. This was repeated until all the input blocks had been processed.

The first change to CCANL1 was to distribute the processing between the PDP-11 and the AP-120B. On the input side the processing required to float and scale the input data is performed by the AP. On the output side the processing required to encode and pack the magnitudes, phases and indices into the output buffer is also performed by the AP. The packing of the signal vector scale factors is still performed by the PDP-11.

The second change was to have the PDP-11 and the AP-120B do more processing simultaneously. This was accomplished by setting up two input and two output buffers in the AP-120B. CCANL1 now reads 8 blocks of input data and transfers it directly to one of two input buffers in the AP. The PDP-11 signals the AP to begin analyzing data in the input buffer. While the AP is analyzing the data in the input buffer, one half the previous AP output buffer (representing 16 frames of speech parameters) containing the encoded and packed magnitudes, phases, and indices and the as yet unpacked signal vector scale factors, is retrieved from the AP. The signal vector scale factors are packed into the PDP-11's output buffer and the buffer is written to disk. Similarly the second half of the AP's output buffer is retrieved, prepared and written to disk. The next 8 blocks of input data are read from disk and transferred to the next AP input buffer. Each input buffer is 2048 words in size and represents 32 frames of input speech. Each output buffer is 4160 words in size (representing 4096 words of encoded and packed magnitudes, phases and indices and 64 words of signal vector scale factors).

The two AP routines DECOMN and PCCZ were combined into one routine, DECPCC, and modified to process 32 frames of data each time it is called from the host program. For each set of 32 frames in the input data, processing in the AP-120B is as follows. The input buffer is first floated and scaled and then on a frame by frame basis the input data is analyzed and the magnitudes, phases and indices are encoded and packed into the appropriate output buffer. The signal vector scale factors for the frame are also moved to the output buffer. After all frames have been analyzed, the first 4096 words of the output buffer are converted to integer and AP processing terminates.

3.1.2 Intermediate Canonical Coordinate File Structure

An intermediate file of Speech Data Base structure is used for the storage and transfer of the results for the CC analysis routine. This file contains a full set of CC parameters without any effort made to compress the information. Due to the fixed point file structure, these parameters must be quantized.

Studies were undertaken to determine adequate quantization levels of the various speech coefficients. From these studies it was determined that magnitudes would be represented by integers in the range of -32767 to 32767 (2 bytes), phases would be represented by integers in the range of -127 to 127 (1 byte) and indices would be represented by integers from 0 to 63 (1 byte). Since storage of the indices required a maximum of 6 bits the signal vector scale factors are packed into the upper two bits of the first 32 indices. As a result, the speech

coefficients for one frame of data (64 words) require 128 words in the output file.

The complete definition of all CC parameters in this file allows for the synthesis routine to be used repetitively with changing degrees of compression. This provides an easy method of comparing compression and coding schemes. As ranges of compression, quantization levels and coding methods are developed they can be incorporated into the analysis routine or operate on the CC parameter file separately.

3.1.3 CC Synthesis Improvements

On the synthesis side, CCSYN1 previously read one block of speech parameters from disk, unpacked and decoded the speech parameters, loaded one frame's parameters into the AP and called the AP routine SYNTH which generated synthesized speech from the speech parameters. The synthesized speech is retrieved from the AP and output to disk as required.

The first change to CCSYN1 distributed the processing between the PDP-11 and the AP-120B. On the input side, the processing required to reconstruct the magnitudes, phases and inverse permutation indices is now performed by the AP-120B. Reconstruction of the signal vector scale factors is still performed by the PDP-11 and the information required to reconstruct them is removed from the input data before it is loaded into the AP-120B. On the output side, the processing required to clip, scale and convert the output data to integer is also performed by the AP.

The second change was to have the PDP-11 and the AP-120B do more processing simultaneously. This was accomplished by setting up two input and two output buffers in the AP-120B. The PDP-11 reads 8 blocks of input data from disk representing 16 frames of speech parameters. Two signal vector scale factors for each frame are reconstructed and the information used to reconstruct them is removed from the input data. The modified input buffer and the signal vector scale factors are loaded into the AP. Another 8 blocks of data is read, processed and loaded into the AP yielding a total of 32 frames of speech parameters per AP input buffer. Two reads are required due to size limitations. The PDP-11 signals the AP to begin synthesizing the data. While the AP is synthesizing the data in the input buffer the PDP-11 retrieves the synthesized data from the previous output buffer, writes it to disk, and then prepares and loads the next input buffer into the AP. Each input buffer is 4160 words long (4096 words of encoded magnitudes, phases and indices and 64 words of signal vector scale factors). Each output buffer is 2048 words.

In the AP-120B, the first 4096 words of the input buffer are converted to floating point. Then on a frame by frame basis the magnitudes, phases and indices are reconstructed and a normalized time domain signal is generated and stored in the output buffer. After all frames have been processed the output buffer is scaled and converted to integer.

3.1.4 CC Analysis Operational Instructions

Program CCANL1 implements the analysis portion of the Canonical Coordinate algorithm. This program requires the analyst to provide an error metric file, a speech data base file to be analyzed and an output file to receive the speech parameters generated by the program. The output file will be twice the number of blocks the analyst chooses to analyze in size.

The analyst may invoke CCANL1 simply by entering the command "RUN $CCANL1". The program performs some initialization and attempts to allocate the AP-120B. If the AP is not available the program waits. Once the AP has been allocated a short informational message is displayed and the analyst is asked for the name of the error metric file. The error metric file is opened and the header and frame size are displayed. After a short wait, caused by the loading of the AP with the error metric, the analyst is asked if the input signal should be preprocessed with pre-emphasis and/or normalization. The analyst is then prompted for the name of the speech data base file that is to be analyzed. The file is opened, its header and the number of analysis blocks available are displayed. The starting block within the input file and the number of blocks to analyze are requested next. Should the analyst attempt to analyze more blocks than possible, the requests for the starting block and the number of blocks are re-issued.

The analyst is next given the option of opening a new or existing file to receive the output and the name of the file is requested. If the file is to be a new file, it is created with sufficient size and opened; otherwise it is opened and the number of blocks in the file and the number required are displayed. At this point, if there is insufficient space in the output file, CCANL1 branches back to the new or existing file option prompt; otherwise the analyst is prompted for the starting block in the output file at which to begin storing speech parameters. A check is performed to ensure that sufficient space exists in the output file from the point at which data is to be stored. If there is insufficient space the program branches back to the new or existing file option prompt.

The analysis of the input file now begins. The program keeps the analyst informed of its progress by displaying a status message after every 256 blocks of data have been analyzed. When the analysis has completed the analyst is prompted for a title that will be inserted into the header block of the output file. The comment field from the input file is displayed next and the analyst has the option of keeping the comments, adding to the comments or deleting the comments and generating comments of his/her own. The output file's header block is displayed and the program terminates.

3.1.5 CC Synthesis Operational Instructions

Program CCSYN1 implements the synthesis portion of the Canonical Coordinate algorithm. This program requires the analyst to provide the error metric file used in the generation of the speech parameter file, the speech parameter file to be synthesized and an output file to receive the synthesized speech generated by the program. The output file will be half the number of blocks the analyst chooses to synthesize in size.

The analyst may invoke CCSYN1 simply by entering the command "RUN $CCSYN1". The program performs some initialization and attempts to allocate the AP-120B to the task. If the AP is not available the program waits. Once the AP has been allocated a short informational message is displayed and the analyst is asked for the name of the error metric file. The error metric file is opened and the header and frame size are displayed. After a short wait, caused by the loading of the AP with the error metric, the analyst is prompted for the name of the speech data base file that is to be synthesized. The file is opened, its header and the number of synthesis blocks available are displayed. The analyst is asked the percent of beta synthesis, the number of bits to use in the phase representation the starting block within the input file and the number of blocks to synthesize are requested next. Should the analyst attempt to synthesize more blocks than possible, the requests for the starting block and the number of blocks are re-issued.

The analyst is next given the option of opening a new or existing file to receive the output and the name of the file is requested. If the file is to be a new file, it is created with sufficient size and opened; otherwise it is opened and the number of blocks in the file and the number required are displayed. At this point, if there is insufficient space in the output file, CCSYN1 branches back to the new or existing file option prompt; otherwise the analyst is prompted for the starting block in the output file at which to begin storing synthesized speech. A check is performed to ensure that sufficient space exists in the output file. If there is insufficient space the program branches back to the new or existing file option prompt.

The synthesis of the input file now begins. The program keeps the analyst informed of its progress by displaying a status message after every 256 blocks of data have been synthesized. When the synthesis has completed the analyst is prompted for a title that will be inserted into the header block of the output file. The comment field from the input file is displayed next and the analyst has the option of keeping the comments, adding to the comments or deleting the comments and generating comments of his/her own. The output file's header block is displayed and the program terminates.

3.1.6 Further Improvements

Although significant improvements over the earlier versions of the CC algorithm implementations have been achieved, additional improvements in performance and ease of operation could be realized.

Presently, CCANL1 works on input data 8 blocks at a time (32 frames), with the resulting speech parameters requiring twice the number of output blocks for a total of 16 blocks. While CCSYN1 works on input data 32 frames at a time (16 blocks). The resulting synthesized speech requires half the number of output blocks for a total of 8 blocks. Due to task size limits the speech parameters are transferred to/from the AP 8 blocks at a time for both analyzer and synthesizer. If the size of the output/input buffers could be increased by two then half as many PDP/AP transfers and disk read/writes would be required, thus improving performance.

Another improvement in performance would come about by restructuring the CC parameter file. Presently as each frame is analyzed the magnitudes, phases and indices are encoded and stored in the output buffer. After retrieval each frame is processed; the magnitudes, phases and indices are decoded and synthesized. A reorganization of the CC parameter file such that all the similar speech parameters are consecutive for blocks of 32 frames would allow the AP to encode/decode speech parameters for the 32 frame blocks of parameters. This improvement would require modification to both the analysis and synthesis routines.

A third synthesis improvement in performance may be realized if the job of reconstructing the signal vector scale factors is off-loaded to the AP. It is unclear at this time if this is feasible but is something that could be looked into.

A fourth synthesis improvement would be to output the synthesized speech directly through the IOP to be recorded. Since the synthesis already operates slightly faster than 2X real time this could most likely be achieved.

An error encountered while attempting to open a file results in an error message being displayed and program termination. This can prove quite annoying. The analyst should be prompted again for the file name instead of terminating the program. Correcting this problem would require modification to the QIO Library routine GFQIO.

3.2 CANONICAL COORDINATE ALGORITHM SAP IMPLEMENTATIONS

In addition to the AP-120B implementations discussed in Section 3.1, ARCON has successfully implemented a non-real-time version of Canonical Coordinate signal analysis and reconstruction on the Systolic Array Processor (SAP). The SAP hardware and support software are described in Chapter 4 of this report. In this section we describe the Canonical Coordinate implementation itself, including the systolic algorithms developed for that implementation.

In Canonical Coordinate signal compression, a complex-valued quadratic error metric with its dimension equal to the analysis window size is used in the time or frequency domain, to define a non-euclidean error criterion for a specific application. An eigensystem solution for the frequency-domain error metric diagonalizes the error metric and provides the transformation operator. This computation takes place outside the analysis/reconstruction process and the transformation operator is available a priori to both the transmitter and receiver.

The analysis process as currently implemented on the SAP is depicted in Figure 3.1 and operates frame by frame. The sampled input speech is zero filled and transformed by a DFT to the frequency domain. At this point the complex signal vector is transformed into a pseudo CC domain using the transformation matrix (the complex conjugate transpose of V) as defined by the error metric eigensystem solution. A further transformation to the true CC domain would require a second eigensystem solution at this stage and would be computationally burdensome. However, this is not necessary since the same effect can be achieved by properly ordering the pseudo-CC elements. P is simply the permutation operator that will rank order the real vector U where U is the pseudo CC domain power spectral density weighted by the diagonalized error metric as defined by the eigenvalue vector. P is then used to order the pseudo CC domain signal vector at which point the vector can be truncated to a length that corresponds to the transmission rate desired.

The transmitted information must include the complex-valued signal elements and their ordering parameters. After decoding and reordering of the transmitted signal parameters, a priori knowledge and/or adaptive procedures can be used to estimate the missing signal elements. For reconstruction the pseudo CC domain signal vector estimate is transformed by the matrix V to the frequency domain, and then returned by an inverse DFT to the time domain. Init. zero-filling requires that the upper half of the time-domain estimate be stripped off at this time.

3.2.1 Specific Algorithms Used In The SAP Implementation

Canonical coordinate processing, summarized in Figure 3.2, is primarily a series of linear transformations on an N-dimensional signal space, where N is the frame size, together with data-dependent sorting operations. In our systolic implementation, we use a linearly connected array of N processors, each responsible for a one-component "slice" of the computation. Most of the steps begin with a vector contained within the systolic array, one component in each processing element, and produce a transformed vector still contained within the array. With this data organization, the steps of CC processing involving component-by-component operations become trivial.

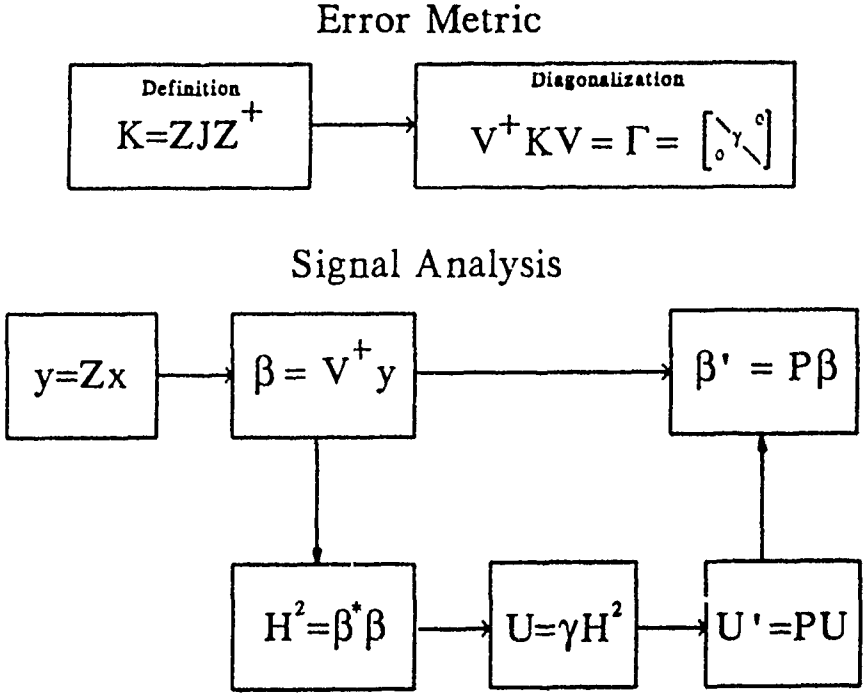


Figure 3.1
Pseudo-Canonical Coordinate Compression

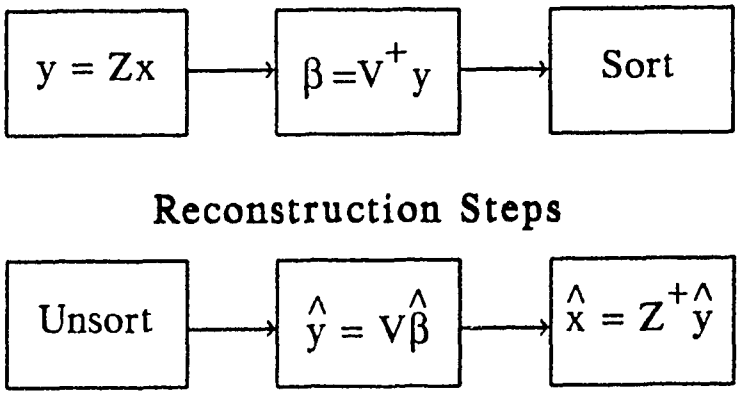


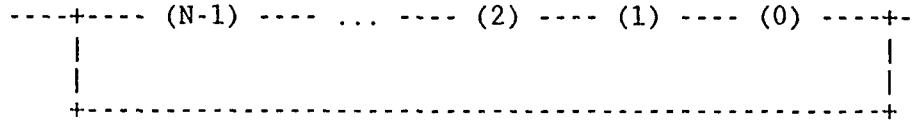
Figure 3.2
Block Diagram of Systolic Algorithm

Discrete Fourier Transform - The first step of analysis, the DFT, shifts the original signal into the array as it computes the transform. For the DFT, the input signal is fed into the array from the left end (entering node N-1 first), node m computing the m-th complex DFT value. After the last sample of the input has entered the array, N zeroes are shifted in. At each systolic step, a node computes a new complex power of the N-th root of unity and multiplies it by the arriving input sample, accumulating the result. The powers of the root of unity are generated recursively by complex multiplication rather than being stored in a table within the processing elements. A similar method is used for the inverse DFT performed as the last step of synthesis, except that the inverse DFT begins with its input already in the array, and simulates the presence of the "redundant" half of the 2N-dimensional complex vector.

Matrix-By-Vector Multiplication - For the matrix multiplication, the vector y is rotated through the array one full cycle, and as the components of y pass through, processor m computes one component of the product. Each processing element contains in its RAM one row of the complex conjugate transpose of V, which in our formulation of CC processing does not change from one frame to another. During the synthesis, the matrix multiplication is computed in the same way, but with one row of V itself stored internally by each processing element.

Sorting Algorithm - Our sorting algorithm is derived from the "odd-even transposition sort" [Ref. 17], modified to work on an array with one-way communication paths. The purpose of the algorithm is to sort the pseudo-canonical coordinates b(i) so that they are ordered by their magnitudes U(i).

This algorithm operates with a circularly connected linear array of N nodes, i.e., one in which there is a data path from node i to node i-1 for every i, and a data path back to node N-1 from node 0:



Each node of the array contains an ordered triple (U(i), b(i), i); in sorting terminology, U(i) is the "key." During each step of the algorithm some of the nodes exchange their triples with those of their neighbors, and at the end the permutation P has been applied to b. Initially node number k has the triple (U(k), b(k), k), for every k. In each systolic step, the processors are paired in a pattern to be described below, and in each pair the following two operations are performed:

a. The left-hand node of each pair sends its triple to the right-hand node.

b. The right-hand node, which began the step with a triple (U(i), b(i), i), has now received another triple (U(j), b(j), j) from the left-hand node. This node compares U(i) and U(j), then keeps the triple whose U is smaller, and sends the other triple to the node on its right. (The node to the "right" of node 0 is node N-1.)

For every step the nodes are paired: node N-1 with node N-2, node N-3 with node N-4, etc., ending with nodes 1 and 0. The algorithm proceeds in N steps. For the first, third, and all other odd-numbered steps, all pairs behave the same. But in each even-numbered step there is one pair of nodes that are "special." In step 2, nodes N-1 and N-2 are the special pair; in step 4 it is nodes N-3 and N-4 that are special; and in general in step 2i the special nodes are N-2i+1 and N-2i. In an even-numbered step, the left-hand node of a "special" pair sends its number and tag to the right-hand processor as usual, but the right-hand node sends its own number and tag on to the right, instead of comparing and choosing the smaller number to send. After step N, the sort will be finished.

The "unsort" operation at the beginning of signal reconstruction (Figure 3.1), which has to place b(n) into node n for n = 0, ..., N-1, is well suited to a systolic array. The permuted coordinates b(P(m)) are rotated through the array together with their original subscripts P(m); each node examines the subscripts as they come by, and when node n sees that the subscript P(m) is equal to its own node number, n, it picks up the coordinate arriving with it, which is b(n). (Each node has its own node number stored in RAM, so that this comparison can be made.)

Speech has been processed with the systolic implementation described here and developed by ARCON. The results are in excellent agreement with those obtained from non-systolic implementations of CC processing for several error metrics, at various compression ratios.

SAP Future Efforts -

This systolic implementation of the CC algorithm provided the design parameters necessary to define a self-contained real-time systolic architecture. For a frame size of 64, each node would require storage for the 64 complex numbers in one row of V (or of its complex conjugate transpose), and could require storage for an additional 64 complex numbers if the DFT coefficients are stored rather than being computed. An individual node would have a repertoire of operations more specialized to the CC application, such as a complex multiply/accumulate operation. The sequencer would consist of read-only memory incorporated into a systolic control module. Buffered A/D and D/A converters would replace the disk input and output used in the SAP implementation.

A practical narrowband CC vocoder would operate on signal vectors of length from 128 to 256 elements. This would require an increase in the number of processor nodes and the memory size at each node. An alternate solution would be to utilize the circular nature of the array for multiple passes of the vector. In this way processor speed can be traded off against the number of processors required. These requirements are well within the range of current technology if a one-chip-per-node approach is taken. Currently such a real-time implementation might be expensive, but in the near future it would become more practical.

3.3 CANONICAL COORDINATE CODING ALGORITHMS

Three programs were developed to assist the analyst in selecting quantization for the speech coefficients. In addition, these programs are useful in assisting the analyst to zero in on the amount of speech coefficient compression to employ in the synthesis program.

The first of these programs, EHISTX, generates energy and silence threshold histograms consisting of 256 bins and stores these consecutively in an output file. After every fifty blocks of the input file have been processed, the energy and silence histograms are retrieved from the AP. The energy histogram is scaled by 256 and its last bin is set equal to 256 and the two histograms are written to the output file. This file can be displayed by ILS routines. Coding of the program is in Fortran and VFC routines which execute on the PDP-11/44 and AP-120B, respectively.

The second program, EASTST, is a combination of CCANL1 and EHISTX except that the magnitudes, phases, and indices are not quantized and no histograms are output. The program computes the RMS for the current frame and if it exceeds the silence threshold it performs the analysis of the frame; otherwise no analysis is done. This program thus outputs speech coefficients for frames of data whose RMS exceeds that of the silence threshold. These speech coefficients are then studied by the last of the three programs, CCHIST. Coding of the program is in Fortran and VFC routines which execute on the PDP-11/44 and AP-120B, respectively.

CCHIST generates histograms of selected speech coefficients. The analyst may generate individual, identical or sets of histograms. These histograms are displayed at the analyst's terminal as they are generated, printed on the line printer when a program terminates normally, and are written to an ILS compatible data file. Coding of the program is in Fortran and VFC routines which execute on the PDP-11/44 and AP-120B, respectively.

3.3.1 Program EHISTX

EHISTX computes energy and silence threshold histograms from raw input speech data. Histograms contain 256 bins each and are output after each 50 blocks of input speech. The energy histogram is scaled by 256 and the last bin is set to 256. The format of the output file is compatible with ILS; therefore, ILS plotting routines may be used to display data.

EHISTX requires the analyst to provide the input file from which energy and silence threshold histograms are to be produced. In addition, the analyst must provide EHISTX with the name of a file which will receive the generated histograms.

The analyst may invoke EHISTX simply by entering the command "RUN $EHISTX". The program performs some initialization and attempts to allocate the AP-120B. If the AP is not available the program waits. Once the AP has been allocated a short informational message is displayed and the analyst is asked if the input signal should be preprocessed with pre-emphasis and/or normalization. Next the analyst is asked to enter values of the decay and knee constants. The value of the decay and knee constants are displayed and the analyst is then prompted for the name of the speech data base file that is to be analyzed. The file is opened, its header and the number of analysis blocks available are displayed. The starting block within the input file and the number of blocks to analyze are requested next. Should the analyst attempt to analyze more blocks than possible, the requests for the starting block and the number of blocks are re-issued.

The analyst is next given the option of opening a new or existing file to receive the output and the name of the file is requested. If the file is to be a new file, it is created with sufficient size and opened; otherwise it is opened and the number of blocks in the file and the number required are displayed. At this point, if there is insufficient space in the output file, EHISTX branches back to the new or existing file option prompt; otherwise the analyst is prompted for the starting block in the output file at which to begin storing speech parameters. A check is performed to ensure that sufficient space exists in the output file from the point at which data is to be stored. If there is insufficient space the program branches back to the new or existing file option prompt.

The analysis of the input file now begins. The program keeps the analyst informed of its progress by displaying a status message after every 50 blocks of data have been analyzed. When the analysis has completed the analyst is prompted for a title that will be inserted into the header block of the output file. The comment field from the input file is displayed next and the analyst has the option of keeping the comments, adding to the comments or deleting the comments and generating comments of his/her own. The output file's header block is displayed and the program terminates.

3.3.2 Program EAHST

Program EAHST is a combination of EHISTX and CCANL1. The purpose of this program is to analyze only those frames of input data where the RMS of the input signal exceeds the silence threshold. Output of this program is processed by CCHIST which generates histograms of analyst-selected criteria.

The analyst is required to supply EAHST with an error metric file, an input speech data base file and output file.

The analyst may invoke EAHST simply by entering the command "RUN $EAHST". The program performs some initialization and attempts to allocate the AP-120B. If the AP is not available the program waits. Once the AP has been allocated a short informational message is displayed and the analyst is requested for the name of the error metric file. The error metric file is opened and the header and frame size are displayed. After a short wait, caused by the loading of the AP with the error metric, the analyst is asked if the input signal should be preprocessed with pre-emphasis and/or normalization. The program then prompts for the values of the decay and knee constants to be used in the computation of silence threshold. After displaying the decay and knee constants the analyst is prompted for the name of the speech data base file that is to be analyzed. The file is opened, its header and the number of analysis blocks available are displayed. The starting block within the input file and the number of blocks to analyze are requested next. Should the analyst attempt to analyze more blocks than possible, the requests for the starting block and the number of blocks are re-issued.

The analyst is next given the option of opening a new or existing file to receive the output and the name of the file is requested. If the file is to be a new file, it is created with sufficient size and opened; otherwise it is opened and the number of blocks in the file and the number required are displayed. At this point, if there is insufficient space in the output file, EAHST branches back to the new or existing file option prompt; otherwise the analyst is prompted for the starting block in the output file at which to begin storing speech parameters. A check is performed to ensure that sufficient space exists in the output file from the point at which data is to be stored. If there is insufficient space the program branches back to the new or existing file option prompt.

The analysis of the input file now begins. The program keeps the analyst informed of its progress by displaying a status message after every 50 blocks of data have been analyzed. When the analysis has completed the analyst is prompted for a title that will be inserted into the header block of the output file. The comment field from the input file is displayed next and the analyst has the option of keeping the comments, adding to the comments or deleting the comments and generating comments of his/her own. The output file's header block is displayed and the program terminates.

3.3.3 Program CCHIST

CCHIST accepts as input a parameter file which has been generated by EAHST. From this file histograms of selected coefficients may be generated. These histograms are displayed at the terminal and written to a print file and an ILS compatible data file as they are generated. The maximum number of histograms that may be produced is 30. The analyst may specify individual, identical or sets of histograms be generated.

When identical or sets of histograms are generated, the histogram parameters are inserted at the start of the comment field in the header block. This information takes a maximum of 154 bytes. Any comments added or kept by the user will be displaced by the length of parameter comments.

The analyst may invoke CCHIST simply by entering the command "RUN $CCHIST". The program performs some initialization and displays a short informational message. The analyst is then prompted for the name of the speech parameter file from which histograms of selected coefficients will be produced. The file is opened, its header and the number of frames available are displayed and the name of the output file is requested. The analyst is then asked whether individual, identical or sets of histograms are desired.

If the analyst selects individual histograms the following processing occurs. The analyst is asked if the generation of a histogram is desired. An answer of no terminates histogram generation. An answer of yes results in the analyst being prompted for the type of coefficient, the starting frame, the number of frames, the number of bins, the minimum and maximum endpoints, the type of histogram and the element to plot. The histogram is generated and displayed at the analyst's terminal and the analyst is again asked if the generation of a histogram is desired. To generate another histogram the analyst must enter the criteria again.

Should the analyst choose to generate identical histograms CCHIST prompts for the number of histograms to generate. Unlike generating an individual histogram, the analyst enters the histogram criteria only once. After the first histogram has been generated and displayed the analyst need only enter the element number for the next histogram. Generating sets of histograms is similar to identical histograms except that criteria for the magnitude, phase, and index histograms are requested. The histograms for each coefficient are generated and displayed and the analyst is prompted for the next element to plot.

When histogram generation concludes the analyst is prompted for a title that will be inserted into the header block of the output file. The comment field from the input file is displayed next. If the analyst selected identical or sets of histograms the comment field has histogram criteria inserted at the beginning. The analyst has the option of keeping the comments, adding to the comments or deleting the comments and generating comments of his/her own. The output file's header block is displayed, the print file is closed and printed and the program terminates.

3.4 RADC/EEV LPC-10E IMPLEMENTATIONS

At the RADC/EEV Speech Laboratory, there exist two implementations, Version 49 and Version 52, of NSA's LPC-10E speech coding algorithm. ARCON was responsible for the implementation of version 49 and its upgrade to version 52. A VAX Fortran-77 version 52 of NSA's LPC-10E was made available to ARCON with the intention of porting this code to the PDP-11/44 at the Speech Processing Facility. Since the source code received from NSA is intended for execution on a VAX system and incompatible for direct use on the PDP-11, ARCON needed to revise this code. The modifications included the ability to access speech data to and from the RADC/EEV Speech Data Base (see Section 5.2). Once the required modifications had been made, the correctness of the PDP-11 implementation was verified by running both versions with identical input and comparing both the listing output and synthesized speech data output from the PDP-11 version with the outputs available from the VAX implementation.

The user interfaces to the two implementations are similar. For the PDP-11 implementation, the following user supplied parameters are required:

1. Starting frame (a frame is 180 samples).
2. Number of frames to process.
3. Whether to perform analysis, synthesis, or both.
4. Listing level (Note: on the PDP-11, listing output goes to LUN 6 which is assigned to TI: at task build time).
5. Input file (Device and UIC default to SP:[200,200]).
6. Output file (Device and UIC default to SP:[200,200]).

The user has the option of performing only analysis, only synthesis, or both during one execution of the task. In the cases of only doing analysis or synthesis, a representation of the bit stream data between a transmitter and receiver is stored in a file of ASCII hex digits with each record in the file holding 54 bits of information for one LPC frame. Thus to do synthesis only, a bit stream data file must already exist. In the case of both analysis and synthesis being performed, no bit stream data files are created. However, a simulation of the encoding and decoding processes during channel transmission is performed between the analysis and synthesis processes. The original source code as received from NSA provides user-selectable channel bit error generation. However the source lines supporting this option were all commented out and we did not restore them in the PDP-11 version. The only other option presented to the user concerns the level of detail provided by the listing output:

Level	Meaning
-1	No data file generated
0	Vary detail level by frame (not implemented on PDP-11)
1	Processing errors and statistics only
2	Coded parameters for each frame
3	Scalar variables and RC's
4	Vectors and matrices
5	Synthesis buffers
6	Analysis buffers

In the NSA VAX version of the LPC-10E code, the user specifies the amount of input data to process in terms of a number of sentences, rather than specifying frame numbers. Code to detect sentences is based upon the counting of consecutive unvoiced frames and comparing the count to a predetermined threshold. The technique seems to work well enough with a file of DAM sentences in a quiet environment. For applications in the Speech Lab, however, a user specified starting frame number and the number of frames of data to process is a more useful control scheme. This scheme is used on other utilities in the Speech Lab such as MAPLP. The user is prompted for these two parameters at the beginning of program execution. Note that for LPC-10E a frame is 180 data points long while Speech Data Base blocks a e always specified in terms of 256 word disk blocks.

3.4.1 Differences Between Version 49 And Version 52

Many of the changes to the source code in going from version 49 to version 52 have more to do with commenting and p· ·gramming practice, as opposed to functional changes in the algorithm. (A complete computer-generated list of differences between Version 49 and Version 52 code is contained in the file LPC52.DIF on the virtual disk LPCXE.)

The functional changes to the algorithm are:

A - An integer square root function was added to replace the standard square root function as found in the Fortran Run-Time library. The new function uses a successive approximation method which can take advantage of a DSP's fast multiply capability for a real-time implementation of the algorithm. This method eliminates prescaling and division common to Newtonian methods.

B - An implementation error in the TBDM subroutine was fixed. The TBDM subroutine calculates Average Magnitude Difference (AMDF) values as required for making estimates of pitch and voicing decisions. The "bug" was in regard to the selection of lag values set one octave up from the initially selected lag with a minimum AMDF value. AMDF values at the upper octave lags are also computed and a final selection as to the AMDF minimum is then made.

C - A signal-to-noise ratio (SNR) value is computed as the running average of the ratio of the running average full-band voiced energy to the running average full-band unvoiced energy. The SNR is used in attempting to optimize the voicing decision in the presence of high noise environments which have been historically problematic for LPC processing of speech. The voicing decision is reduced to a linear discriminant function based upon a set of fixed "voicing parameters" (e.g., zero crossing counts, ratio of AMDF max. and min.). Finer voiced/unvoiced discriminations are statistically possible given knowledge of the current SNR. In this implementation, the SNR is used to select a vector of coefficients used to form the linear discriminant function evaluated once every half frame.

D - A check is no longer performed to monitor for the erroneous "locking up" in the voicing state as has tended to occur within high noise environments.

E - A new normalization subroutine was added for normalizing calculated voicing parameters. These parameters are all represented such that they all fit within 16-bit data structures.

F - The dynamic programming algorithm for pitch tracking was changed such that a search for a pitch winner value at 1/3 and 1/4 of the original best pitch estimate is made. Previously, a search was only made at 1/2 of the original pitch estimate.

G - The pitch smoothing logic during the decoding of transmitted parameters was corrected.

3.4.2 Porting Of LPC10E Code To The PDP-11

Because the Speech Processing Facility does not have a full Fortran-77 implementation, special techniques were necessary to adapt the NSA LPC-10E Version 52 code to run on the RADC/EEV PDP-11/44. This section describes the problems found and the solutions used. With appropriate modifications, the same approach can also be used for any future LPC-10E version obtained from NSA. The source files and demonstration data files were read from a magtape on ARCON's own VAX system and were transferred to RL02 disks for eventual transfer to the RADC PDP-11 system. The source code was successfully compiled and run on the ARCON VAX system.

Several problems remained before this code could be run on the PDP-11 system with its Fortran IV compiler:

1. Conversion of VAX system calls into equivalent calls to the RSX-11M executive. These calls are for functions such as returning the current time and date and, in the case of the VAX, dynamically allocated a Logical Unit Number (LUN) for file I/O. Some of these functions can be simply deleted by doing things like static allocation of LUN's at compile time as is typical with RSX-11M application programs.

2. Conversion of CHARACTER data structures in VAX Fortran into LOGICAL*1 structures in Fortran IV and rewriting any string manipulation sections of the code.

3. Conversion of all WHILE DO constructs into IF and GOTO equivalences.

4. Conversion of file I/O routines into QIOLIB subroutine calls which will allow the LPC10E program to work with Speech Data Base files.

5. Removing all END DO statements and replacing them with appropriate statement labels which must also be included in the corresponding DO statements.

6. Removing all "Include" statements and manually including the material from the include file by using the include function available with the EDT text editor.

A fully functional Fortran-77 compiler and run time object library would have made this project considerably easier. (The Vax Fortran, for which the NSA code was written, is an enhanced version of the Fortran-77 standard.) There is a Fortran-77 compiler on the PDP-11 systems which seems to work without error. However, the appropriate run time object modules for the compiler are not available and therefore there are limitations on the types of operations that can be performed program modules compiled with this compiler. Most importantly, no run-time I/O support is available, as either the required run time modules are not found at link time or the program is terminated at run time with errors within the I/O modules. Thus Fortran-77 modules that perform I/O cannot be run on the PDP-11 system.

A complete conversion of the LPC10E code to Fortran IV is also problematic in that the block structure offered by Fortran-77 would have to be taken apart, hand-coded with GO TO statements, and carefully tested to make sure that the hand-coded conversion is correct. In addition, a way of dealing with 32-bit integers would have to be devised. It should be noted that the code as compiled on the VAX uses a default of 32-bit integers (compared to the 16-bit integers of Fortran IV) and at some places in the code, the use of all of the precision offered by 32-bit integers is explicitly called for.

ARCON's approach to this problem was to sort the LPC-10E modules into those that require the I/O offered by Fortran IV and those that require the 32-bit integer operations and block structure offered by Fortran-77. The modules were appropriately linked together and overlaid as needed. At task build time, the appropriate run-time library is associated with the correct set of object modules. This is ensured by placing all Fortran-77 modules in "Fortran-77 only" overlays for which the Fortran-77 run-time library is explicitly linked. This approach works but care must be taken regarding the default data types for the two compilers (e.g., logicals), which is best dealt with by explicit declaration of data types (e.g., logical*1). All files for this development are located on the virtual disk LPCXE.

The task build command file LPCBLD.CMD utilizes an overlay structure and defines a task name ...LPC. The overlay descriptive language file LPCBLD.ODL is extensive and combines Fortran IV and 77 modules. The task image consists of a root segment made up of Fortran IV modules only and two large memory resident overlay segments sharing common virtual address space. A remapping of the overlay segments occurs once every analysis frame.

As opposed to the dynamic LUN assignment used in the VAX environment a static LUN assignment is made for the PDP-11 task:

LUN #1 = Speech data file input (Default SP:[200,200])
LUN #2 = Speech data file output (Default SP:[200,200])
LUN #3 = Bit stream input (Default SY:current UIC)
LUN #4 = Bit stream output (Default SY:current UIC)
LUN #5 = TI:
LUN #6 = Listing output set to TI: at Task Build.

Note that the LUN for the listing output can be changed by first installing the task and then using the REA (Reassign) command available from MCR. Thus, the listing output can be directed either to the lineprinter or to a disk device in which case a file named LPCDATA.DAT is created. Also of note is the fact that the listing output requires more than 80 columns per line. When the listing output is directed to a VT100, a SET /BUF=TI:140. command and the changing of the VT100's SET-UP parameter concerning 80/132 COLUMNS must be enacted.

The correctness of the PDP-11 implementation was verified by comparing both the listing output and synthesized speech data output from the PDP-11 version with the outputs available from the VAX implementation. Of course, the outputs compared were taken with common inputs to the PDP-11 task and the VAX task: the input speech data used is in a file provided by NSA, DAM9.SPD, which consists of 9 DAM sentences each read by a different speaker. After some errors in the PDP-11 task were located and corrected, the output files matched the VAX output files. Also, a listening comparison of the synthesized data files from the PDP-11 and VAX was done using the MAPLP utility to confirm the integrity of the output files and compatibility with the Speech Data Base. LPC10E analysis of other Speech Data Base files on the PDP-11 system has also been performed successfully.

CHAPTER 4

RADC/EEV SPEECH PROCESSING FACILITY COMPUTER SYSTEM

The primary computing machine used for the generation, maintenance, and updating of data at the RADC/EEV Speech Laboratory is the DEC PDP-11/44 minicomputer. It is the central facility for software development, data storage and data analysis. It serves as the host for all the other processors in the lab. These peripheral processors include:

1. FPS AP-120B array processor
2. MAP-300 arithmetic processor
3. CSP-30 signal processing minicomputer
4. Spectral Dynamics digital signal analyzer system
5. NOSC Systolic Array Processor (SAP)
6. TRS-80 Model-100 microcomputer DRT data entry units (DEU's)

There are two other systems, DEC PDP-11/34s, available at RADC/EEV. The first is located at the RADC/EEV Communicability Test Facility in Building 1120. It hosts two MAP-300 array processors and is primarily used for the communicability testing effort. The second is located at the RADC/EEV Test and Evaluation Facility in Building 1124 and is used extensively for the Digital Tape Mastering and T&E requirements of the DoD Digital Voice Processing Consortium.

The following sections will describe in detail the current status of hardware, system software, and program development software for each of these systems. Changes or additions made to these systems under the current contract will be stressed. These systems have been previously discussed in References 9 and 10.

4.1 PDP-11/44 HARDWARE (BLDG. 1120)

The PDP-11/44 functions as the host--providing data communication and storage facilities, and serving as the program development center--for all other processors available in the Speech Lab. The PDP-11 runs the RSX-11M operating system, a real-time, interrupt-driven, multi-user environment which is well suited to the development of scientific applications, and which is easily modified to accommodate a changing hardware setup.

The PDP 11/44 hardware configuration has undergone some minor changes during the contract period. Table 4.1 shows the current status of the backplane of the PDP 11/44 minicomputer located in the RADC/EEV Speech laboratory. For each Unibus module present in the 11/44 system, Table 4.1 presents the board number(s), a description, memory address(es), interrupt vector location(s), and bus priority. Note that several of the modules listed are not presently interfaced into the system but are available for future use. On the other hand there are devices which are mentioned but are not actually available at the Speech lab (e.g., a TM11 magtape controller); software device drivers were generated for them nevertheless.

Figure 4.1 is a representation of the processors and peripheral devices currently at the speech lab. It illustrates the common communication path which they share--the Unibus. Figure 4.2 puts this hardware in a physical perspective as it illustrates the location of each device in the computer room.

Previously, the storage devices on the PDP-11/44 consisted of two 300 Mbyte disk drives, two RL02 drives (10 Mbytes each), two RL01 drives (5 Mbytes each), and four RK05 drives. The RK05 hardware was expensive to maintain and had been superseded by the RL01/RL02 devices for offline data storage. The system was restructured which led to the removal of the RK05 hardware and an obsolete paper tape punch/reader. This resulted in a reduction of the PDP-11/44's physical size and allowed for more convenient placement of the large CDC drives. The second 300 Mbyte drive was previously used only for backup purposes and as a standby in the event of a failure on the primary drive that was used as the main "system" device. The second drive is now used for the storage of the Speech Data Base.

Table 4.1
PDP 11/44 RSX-11M,V3.2 Hardware

The hardware consists of a PDP 11/44 with 1024K words (16 bits each) of MOS memory with Parity checking or an address space of 0-7640000 (octal) bytes, plus a Unibus addressing space of 8 Kbytes, i.e.17760000-17777777(8).

BA11-AA unit, from right to left

Device Controller		Function	Address	Vector	BR
PDP 11/44 CPU					
-- KD11-Z	M7090	Console interface	17777560	60	4
			17777566	64	-
		Line clock	17777546	100	-
	FP11-F	M7093 Floating		224	-
		Point Unit			
	M7094	Data Path			
	M7095	Control			
	M7097	Cache	17777744-54		
	M7098	Unibus interface			
	M8743	1 Meg. byte of memory			
	M8743	1 Meg. byte of memory			

Table 4.1 (Continued)
PDP 11/44 RSX-11M,V3.2 Hardware

BA11-AA unit, from right to left(Continued)

Device	Controller		Function	Address	Vector	BR
		M9202	Unibus connector			
SA:	DR11-C	M7860	SAP intf. (non-DMA)	17767770	320	4
TT1:	DL11-W	M7856	LA-36 via QUINTRELL	17776520	330	4
			RS232-C, 300 baud	17776524	334	4
TT2:	DL11-A	M7800	YA Tektronix 4015-1	17775610	310	4
			@20 ma, 9600 baud, self-clocked	17775614	314	4
TT3:	DL11-W	M7856	ZENITH P.C. EIA, 2400 BAUD	17775620	340	4
SA:	DR11-W	M8716	SAP intf. (DMA)	17772414	324	5

UNIBUS to BA11-F =========>

BA11-F unit, from front to rear

Device	Controller		Function	Address	Vector	BR
--	DR11-C	M7860	16-BIT Parallel Interface Digital data I/O SD350	767760	410	5
TT4:	DZ11-A	M7819	Speech Peripheral Bench RS232-C, 4800 baud	760010	350	5
TT5:		"	VT-100 @ 9600 baud	"	"	"
TT6:		"	VT-100 @ 9600 baud	"	"	"
TT7:		"	M100, Cmptr Rm, @ 9600 baud	"	"	"
TT10:		"	Modem @ 300 baud	"	"	"
TT11:		"	M100, Sound Rm, @ 9600 baud	"	"	"
TT12:		"	VT-100 @ 9600 baud	"	"	"
TT13:		"	VT-100 @ 9600 baud	"	"	"
RL0:	RL11-AK	M7762	RL02-AK drive	774400	360	5
RL1:		"	RL02-AK drive	"	"	"
RL2:		"	RL01-AK drive	"	"	"
RL3:		"	RL01-AK drive	"	"	"
MP0:	HIC-11		MAP-300 Array Processor	766004	440	7
LP0:	LXY11	M7258	Printronix P300	777514	200	4
DR0:	Xylogics 650		Xylogics RM05 Emulation	776700	254	4
SD0:	SD13209		Modified DR11-C's for	767700	300!	x
SD1:	"		Spectral Dynamics 360	767710	300!	x
SD2:			(looks like four devices to the			
SD3:			RSX-11M operating system)			

UNIBUS to FPS AP-120B =========>

Table 4.1 (Continued)
PDP 11/44 RSX-11M,V3.2 Hardware

BA11-F unit, from front to rear (Continued)

Device	Controller		Function	Address	Vector	BR
	DDV11-C		LSI11 Backplane for:			
	DW11-B	M8217	UNIBUS/QBUS Converter for	-	-	-
IB0:	IBV11-A	M7954	Instrument Bus for SD-350	760150	420/430	x
			IEEE-488 Bus in/out ==========>			
	DW11-B	M9401	QBUS Mirror Image	-	-	-
	TEV11	M9400YB	QBUS Terminator	-	-	-
	DW11-B	M9403	QBUS Connector	-	-	-
			FPS AP-120B			
AP0:	FPS #218		FPS Arith. Proc. AP-120B	776000	170	x
			UNIBUS to BA11-E ==========>			

BA11-E unit, from front to rear

Device	Controller		Function	Address	Vector	BR
OMR	DC11-AB	M7821	Decision Inc. 6510	776500	300!	-
	"	M957,M594	Optical Mark Reader	776504	300!	-
	"	M105				
-	DB11-A	M7248	UNIBUS repeater	-		-
	"	M7213,M783				
	"	M784,M785,M930A				
			UNIBUS to CSP-30 S/N 20 ==========>			
CSP30	1020A		CSP30 to PDP 11/44 link	760020	370/374	-
		M930A	Passive UNIBUS terminator	-	-	-

Special status devices

Controller not connected:

SD-13378	GPIB adapter for SD350-6, talks to IBV11-A	-	- -

Loadable driver for pseudo-devices:

VD:	Virtual Disk driver		-

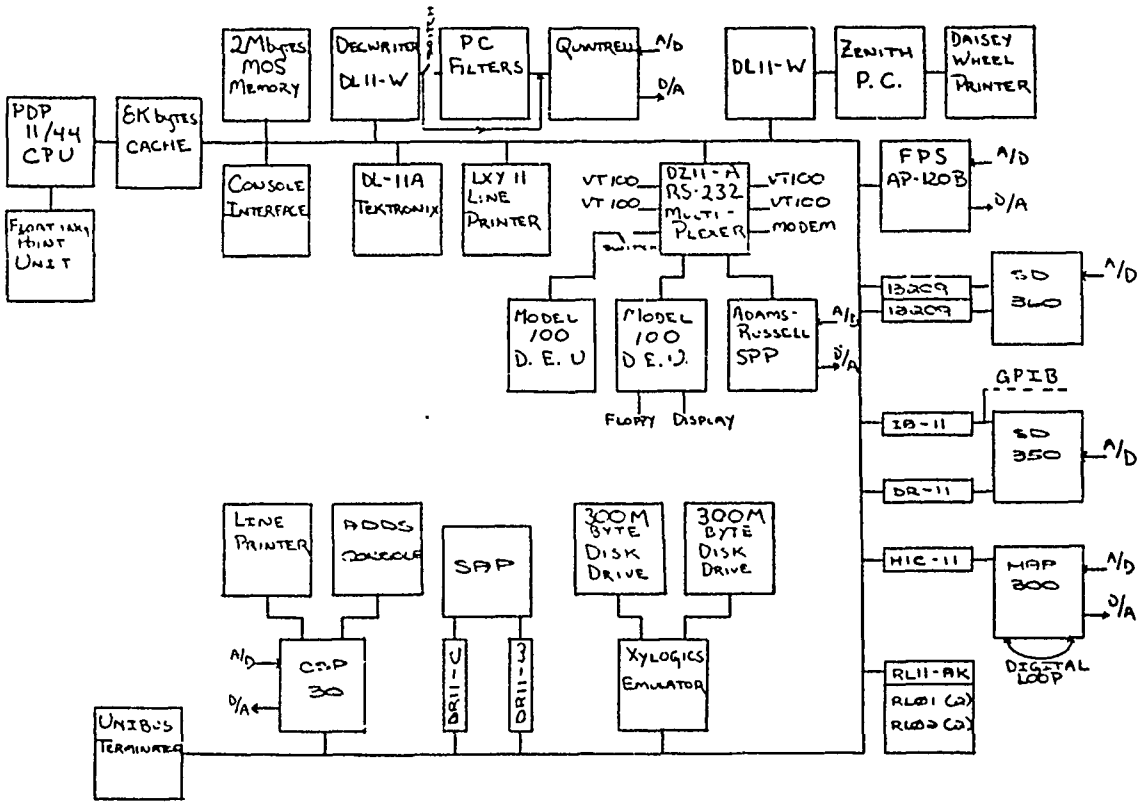


Figure 4.1
PDP 11/44 UNIBUS Structure

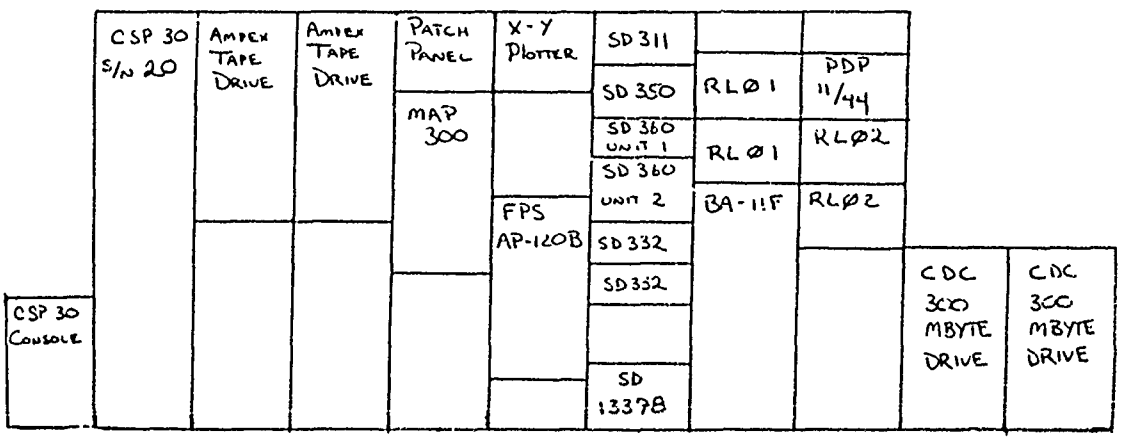


Figure 4.2
PDP 11/44 Configuration

4.1.1 MAP-300 Arithmetic Processor

There is currently one MAP-300 Array Processor interfaced into the PDP-11/44 system at the Speech Laboratory and is used mainly for program development of speech algorithms discussed in Chapter 3. The MAP-300 consists of a Central Signal Processing Unit (CSPU), an Arithmetic Processor (AP), a Host Interface Module (HIM), and two I/O Scrolls (ADAM and AOM). There are three separate Memories, each capable of supporting a maximum of 256 Kbytes of MOS memory or 64 Kbytes of Bipolar Memory. The word size for this machine is 32 bits. Each memory is served by a separate 32-bit bus and memory arbitration logic. It is possible to have concurrent operation of all processors with little memory contention. The MAP-300 on the PDP-11/44 system at RADC/EEV has 128 Kbytes of 500 nanosecond memory on bus 1, 32 Kbytes of 500 nanosecond memory on bus 2, and 20 Kbytes of 200 nanosecond memory on bus 3.

4.1.2 AP-120B Array Processor Hardware

The Floating Point Systems AP-120B array processor at the RADC/EEV facility uses a pipeline architecture. The cycle time of the adder, multiplier and memory is 167 nsec, giving a pipelined multiply/accumulate every 167 nsec. This will allow a 1024 point complex FFT every 4.8 msec. The 120-B uses a 38-bit floating point format, with a 10-bit binary exponent and a 28-bit two's complement mantissa. It has several memories, each with its own dedicated controller: a 4K addressable word program memory with 64-bit words, the 38-bit/word Data memory with two "data-pads" for scratch registers, and a 38-bit/word table memory. There are 16 words of 16-bit integer scratch pad for use in transmitting address data, loop counters and decision logic. The AP-120B was modified to include dual channel programmable D/A - A/D ports for signal data using FPS IOP-16 interface cards and Datel D/A - A/D boards. Software was developed to utilize these data I/O ports for vocoder research.

4.1.3 Systolic Array Processor Hardware

The Systolic Array Processor (SAP) was originally developed by J. J. Symanski of the U. S. Naval Ocean Systems Center (NOSC) [Ref. 18]. This device is a programmable, reconfigurable systolic array testbed created for the purpose of early research on the implementation of signal processing algorithms on a lattice of identical processors operating in parallel on data synchronously flowing through the structure. The SAP consists of 64 processing nodes each capable of bit-serial data communications with neighboring nodes. The data paths for nodes on the lattice structure's boundary can be rerouted under software control.

For control and arithmetic, each node contains a simple microcontroller (Intel 8031) and an arithmetic processing unit (Intel 8231). The microcontroller also synchronizes data transfers and provides command and data I/O for the SAP. One Kbyte of local RAM for data storage is available at each node, and each node maintains a local program store (4 Kbyte EPROM) with the SAP's central control module globally broadcasting codes to select specific functions to be executed within the nodes.

Upon the arrival of the SAP at the RADC Speech Laboratory in 1986, interface hardware was designed and built to connect the SAP with the Speech Laboratory's PDP-11/44 system. The hardware used consisted simply of an interrupt driven DR-11C device and a newly configured cable. A device driver was designed to provide the I/O operations peculiar to the SAP's 16 bit command transfers and 8 bit data transfers. The driver is in the standard RSX format and both the executable code and the data structures are "loadable". The source code for this device driver and data tables, SADRV.MAC and SATAB.MAC, are on the virtual disk SAP at [200,300]. The device logical is SA:.

The DR-11C device has a measured throughput of 75-85 microseconds per transfer. Most all of the possible operations within the SAP can be completed in much less time than 80 microseconds, in fact, many can be completed in less than one microsecond. In order to reduce this SAP idle time during its performance of some tasks, a faster alternative to this hardware/software driver configuration was needed. A DMA version of the DR-11 (i.e., a DR-11W) which provides one word transfers every 2 microseconds was selected. This resulted in a 30 to 40 times decrease in SAP/PDP interaction time.

The DR-11W was attached as a peripheral to the Unibus of the PDP-11/44. The address and vector switches on the board were set for 772410 and 324, respectively. The hex board was inserted into a slot of the peripherals backplane in the CPU box after the removal of the GRANT CONT CARD from the slot. The NPG (non-processor grant) jumper for this slot (pins CA1 to CB1) was removed from the backplane.

WARNING!!

> Should the DR-11W board ever be removed from the system, the NPG jumper must be reinserted (this is true of any DMA device). Also, a Grant Cont. Card would also be inserted into the vacated slot.

A device driver was written for operation with the DR-11W. The driver for the DR-11C provided the foundation for the new driver. The source code files, SADRV.MAC and SATAB.MAC, for this new driver are on the virtual disk SAP at [200,201] along with command files. The major changes involve the control of DMA operations and the restriction of only word (16-bit) transfers with the DR-11W as contrasted with the byte I/O capabilities offered by the DR-11C.

This driver was written for a PDP-11/44 and will not work with any other PDP-11 not supporting extended memory (e.g., PDP 11/34). The extended memory of the PDP-11/44 requires the translation of an 18-bit memory address code placed on the Unibus by the DR-11W during a DMA process into a full 22-bit value to allow access to the full memory space available. Unibus Mapping Registers (UMR) on the PDP-11/44 provide this address translation (See PDP-11 Processor Handbook). Two system routines are available in RSX-11 for allocating and loading UMR's by a task--$STMAP, $MPUBM (See Appendix B of the Guide to Writing RSX-11 Device Drivers for information of using $STMAP and $MPUBM). The initialization of the data structures associated with the new driver in SATAB.MAC is slightly different from those of the DR-11C. An additional 6 word block at the end of the STATUS CONTROL BLOCK is allocated for storing UMR information. The Control Byte in the UNIT CONTROL BLOCK is also initialized to indicate that the device is a DMA device and that data buffers processed by the driver must be word aligned.

A separate cable was constructed for interfacing the SAP to the DR-11W. Both versions of the SA: driver have been retained as well as the cable for the DR-11C. Thus the DR-11C will provide a back-up for the DR-11W.

Problems were encountered after the DMA cable was debugged. Sequences of SAP commands which previously worked with the DR-11C (interrupt driven) interface no longer worked when the commands were sent to the SAP at a much higher transfer rate via the DMA interface. (The peak transfer rates for the DR-11C and DR-11W are approximately 75 microseconds and 3 microseconds, respectively.) The first problem was found with the SAP command CP ("clear processors") which does a hardware reset of all of the microcontrollers making up the Systolic Processing Elements (SPE's). Upon reset, each microcontroller goes through some initialization code in preparation for receiving its first opcode and interrupt signaling a request to perform the function encoded in the opcode. A certain period of time is required after the reception of CP before the SPE's are fully initialized. However, while the SPE's are executing the initialization routine, the SAP Computer Interface (CI) can receive, decode, and execute subsequent commands. It was found that if commands are sent to load the SPE's with an opcode and interrupt immediately following a CP, without first using a delay command (DY 15), the opcodes did not get performed. This was never a problem with the old interrupt driven interface since the SPE's had sufficient time to initialize themselves before the first interrupt occurred. Thus, SAP programmers are advised to always include a DY 15 command after a CP command.

Other, similar "race conditions" were noted. Using the DMA interface, it is necessary for the programmer to insure that delays are included in "FLAG CYCLE" code sequences. These sequences involve the FLAG signal line which is broadcast to all SPE's and which goes through a specific number of cycles with each cycle used to synchronize a specific operation. For example, during an "A-Store Lateral Move" function, four bytes of data are transferred from a SPE's local memory to the local memory of one of its neighbors. Only one byte can be transferred at a time using the MD command. Also, certain operations

must be performed on the data at the "boundary" SPE's such as introducing input data to the array or collecting output data from the array. Thus a means of synchronizing the individual byte transfers among SPE's is required. The clearing and the resetting of the FLAG line defines a FLAG cycle. However, when using DMA, a delay after clearing the FLAG and after resetting the FLAG is needed to give the microcontroller time to perform the software polling of the FLAG line.

Another example of the need for a delay between successive SA commands is the IT WT sequence which will interrupt (IT) the SPE's and thereby initiate a function to be performed within the individual SPE's, whereupon then the SAP will wait (WT) until all SPE's have indicated the completion of the function. While in the wait state, the SAP will not request further commands from the host. Disastrous results can ensue should certain commands be sent to the SAP before all of the SPE's have completed their functions. When the microcontrollers are performing functions requiring interaction with the APU chips, they must operate with a 2 mHz. clock as opposed to the "fast" 8 mHz. clock. While in the slow clock mode of operation, it is possible for the WT command in an IT, WT sequence to be executed before the SPE's can signal themselves busy. Thus the SAP does not go into the appropriate wait state and programs do not operate as intended. Note that this problem never occurred while using the slower, interrupt-driven interface. One solution is to always include a DY 3 command between the IT and WT commands. (The operand of the DY command specifies the length of delay which ranges between 16 and 256 microseconds.) A couple of disadvantages to this solution are:

1. Extra command words must be generated, stored, and transmitted to the SAP. A significant SAP program will have thousands of IT, DY 03, WT sequences, so adding an extra word on each sequence will have a notable effect.

2. The delay can slow down program execution in the sense that the function can be executed in less than the time taken by the delay.

An alternative to the IT, DY 3, WT sequence for executing SPE functions is to use the IW command which combines the interrupt and wait operation into one command. However, even with the older interface, this command did not work as expected while the SPE's were working in the slow clock mode. The hardware for implementing the IW instruction was modified so that it would accurately detect the length of time required for SPE's to perform the requested function even when the slow clock was used. The specific change was to use the 64 kHz., instead of the 0.25 mHz. clock, as the clock input to the quad D flip-flop chip at K32 in the SAP computer interface. Schematics showing the current state are available in the SAP Hardware file at the RADC/EEV Speech Laboratory. Following these hardware changes, correct program performance was validated using the IW instruction for initiating SPE functions.

The SAP is software reconfigurable. For vector operations it can be configured such that data flows in a linear path through all 64 processors by starting at the top left processor (viewing the SAP from the front) and ending at the bottom right processor. This configuration is set up by setting the "A Data Mux" to mode 2. The hardware on the AB module is constructed such that the top left processor receives input to its A register from the "Virtual A-0" register while in mode 2. This means that a new data byte can be pumped into the linear array from the virtual A-0 register while an output byte is taken from the bottom right processor's A register which feeds into the A-7 virtual register.

The Canonical Coordinate algorithm (see Chapter 3) required that another mode of A register data flow be created. In this mode the linear flow through all 64 processors is maintained yet the output of the array (bottom right processor) is fed back into the input of the array (top left processor). This mode of data flow is selected by a value of 6 in the field of the "SA" command which selects the "A Data Mux" input. This enhancement consists of adding inputs to the 8 multiplexers used to select the inputs to the leftmost A registers when the "A data" is shifted right. Figure 4.3 displays the schematics for details of the changes.

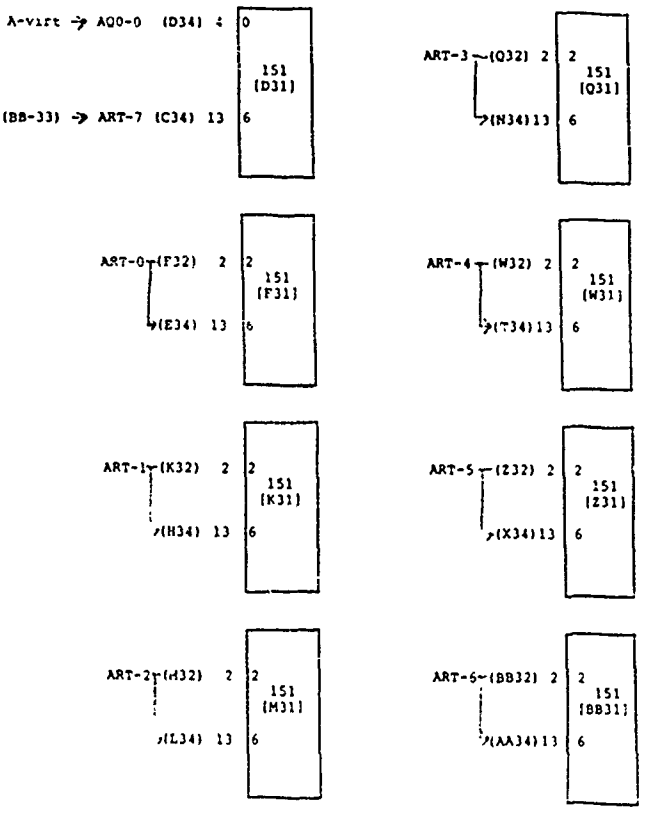


Figure 4.3
Systolic Array Processor Modifications

4.2 PDP-11/34 HARDWARE (BLDG. 1120)

A separate PDP-11 system received at the Speech Lab from DCEC in March 1985 is now operational. An evaluation of the system following the installation performed by DEC Field Service indicates all hardware subsystems are functional except for one of the magtape drives (MM1:). This drive fails diagnostics and fails attempts to write tapes using RSX-11 utilities. The lack of this one subsystem does not impede any of the near-term projects for which the PDP-11/34 is allocated.

This system is utilized primarily for communicability testing. Several real-time vocoder algorithms designed for this system are now operational and provide the basis for the communicability testing.

Figure 4.4 shows the peripherals available on this system and Table 4.2 illustrates the system configuration.

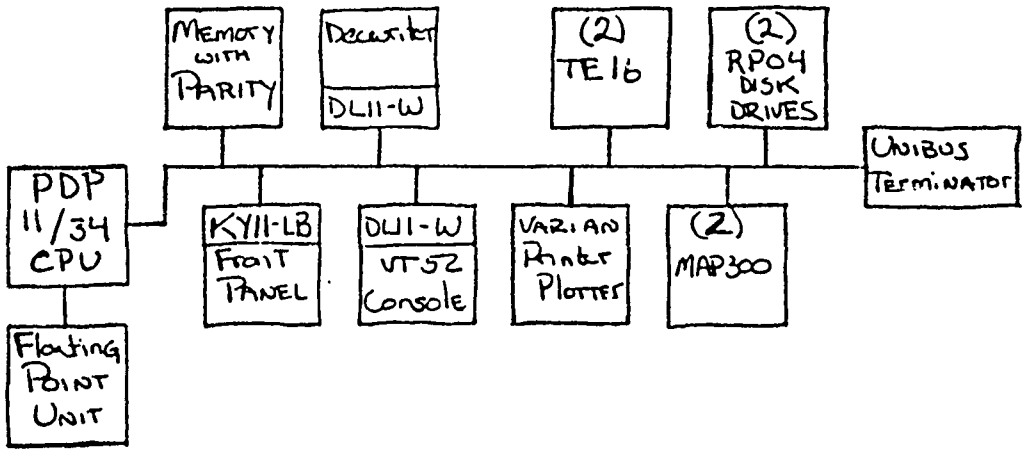


Figure 4.4
PDP-11/34 (BLDG. 1120) UNIBUS Structure

Table 4.2
PDP 11/34 (BLDG. 1120) RSX-11M,V4.0 Hardware

The hardware consists of a PDP 11/34 with 124K words (16 bits each) of MOS memory with Parity checking or an address space of 0-757777 (octal) bytes, plus a UNIBUS addressing space of 20(octal) Kbytes, i.e.760000-777777(8).

BA11-KA unit, from right to left

Device	Controller		Function	Address	Vector	BR
PDP 11/34 CPU						
--	KD11-EA	M7859	Programmers Console Console Switch Register	777570	-	-
--	FP11-A	M8267	Floating Point Unit		-	-
		M8265	Data Path		-	-
		M8266	Control		-	-
		M9312	Boot diags, Console Emulator		-	-
TT0:	DL11-W	M7856	LA-36 @20 ma, 300 baud	777530	300	4
	MS11-JP	M7847	1 Meg. byte of MOS memory (4 16K boards)			
	" "	M7847	1 Meg. byte of MOS memory (4 16K boards)			
		M9202	Unibus connector			
	MS11-JP	M7847	1 Meg. byte of MOS memory (4 16K boards)			
	" "	M7847	1 Meg. byte of MOS memory (4 16K boards)			
	" "	M7847	1 Meg. byte of MOS memory (4 16K boards)			
	" "	M7847	1 Meg. byte of MOS memory (4 16K boards)			
	" "	M7847	1 Meg. byte of MOS memory (4 16K boards)			
	" "	M7847	1 Meg. byte of MOS memory (4 16K boards)			
TT1:	DL11-W	M7856	VT-52 Console	777560	60	4
TT2:	DL11-W	M7856	VT-52 Console		-	-
		M7850	Parity Controller (2 units)		-	-
		M7850			-	-

UNIBUS to BA11-K ==========>

BA11-K unit, from front to rear

Device	Controller		Function	Address	Vector	BR
--	BA11-K		Expansion Box with Backplanes and Power Supply			
--	RH11-C	M7297	RP04 Disk Drive	776700	254	5
	" "	M7296	" " "			
	" "	M7295	" " "			
	" "	M5904				
	" "	M5904				
	" "	M5904				
		M9202	Unibus connector			

Table 4.2 (Continued)
PDP 11/34 (BLDG. 1120) RSX-11M,V4.0 Hardware

BA11-K unit, from right to left (Continued)

Device	Controller		Function	Address	Vector	BR
MT0:	RH11-C	M7297	TE16 800 bpi Magtape Drive	772440	224	5
	" "	M7296				
MT1:	RH11-C	M7295	TE16 800 bpi Magtape Drive			
	" "	M7294				
	" "	M5904				
	" "	M5904				
	"	M5904				
		M9202	Unibus connector			
MP0:		G727	MAP-300 Array Processor			
MP1:		G727	MAP-300 Array Processor			

UNIBUS to BA11-F ==========>

BA11-F unit, from front to rear

Device	Controller		Function	Address	Vector	BR
--	DB11	M7248	Bus Repeater			
		M7212				
		M7212				
		M7212				
		M784	Unibus Receiver			
		M9202	Unibus connector			
	DR11-C	M7860	Interface	767770	420	
	DR11-C	M7860	"	767760	430	
	VARIAN		Status 42 printer/ plotter	777514	200	
	VARIAN		" "			
		M9202	Unibus connector			
	DL11-D	M7800		775610	300	4
	DL11-D	M7800		775620	310	4
	DL11-D	M7800		775630	320	4
	DL11-E	M7800		775640	330	4
		M9202	Unibus connector			
	DL11-D	M7800		775650	340	4
		M9202	Unibus connector			
		M9302	UNIBUS Terminator			

4.2.1 MAP-300 Array Processors

The two additional MAP-300 processors are included with the PDP-11/34 system at the Communicability Test Facility in Building 1120. MAP diagnostic programs and the required MAP loader programs can be found on DB:[6,100].

These two MAPs primary use is in the communicability testing effort and are similar to the MAP-300 on the 11/44 except for one difference. The difference is that the dual MAPs have 96 Kbytes of 500 nanosecond memory on bus 1, 32 Kbytes of 300 nanosecond memory on bus 2, and 16 Kbytes of 170 nanosecond memory on bus 3. Therefore, the dual MAPs have faster memory yet not as much memory as the PDP-11/44 MAP-300.

Each MAP has an audio interface unit that includes handset I/O and anti-alias filters. These interfaces were modified to interact with the Speech Lab's audio system. A cable to connect the "Modem" ports of the two MAPs is available for transferring the digital bit stream from one processor to the other or through a bit error simulation device. Loop-back connectors are also available for these "Modem" ports.

4.3 PDP-11/34 HARDWARE (BLDG. 1124)

The second PDP-11/34 is located in the RADC/EEV Test and Evaluation Facility in Building 1124 and is used primarily for the DoD Digital Voice Processing Consortium effort. This PDP-11/34 replaced a disabled PDP-11/20. It includes two RL02 disk drives permitting the system to run with the full version of RSX11M. The peripherals used with the PDP-11/20 were retained. The devices supported by the operating system are 1) an RL02 disk drive 2) a 9 track magtape 3) two terminals and 4) A/D and D/A converters. Figure 4.5 and Table 4.3 are a representation of the peripheral devices currently available with this system.

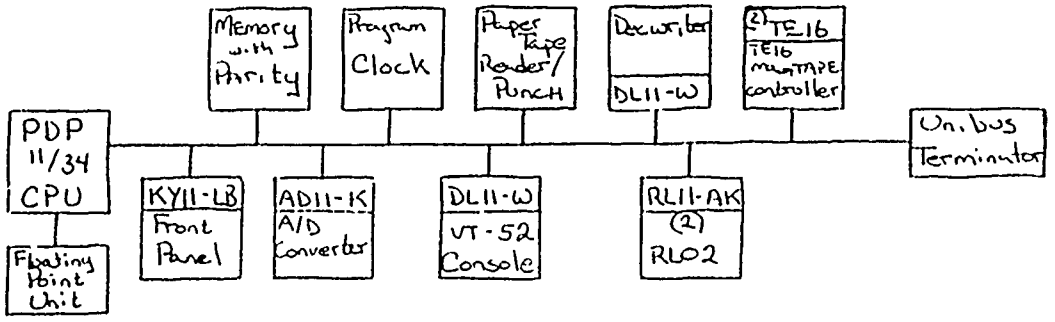


Figure 4.5
PDP-11/34 UNIBUS Structure

Table 4.3
PDP 11/34 (BLDG. 1124) RSX-11M,V3.2 Hardware

The hardware consists of a PDP 11/34 with 124K words (16 bits each) of MOS memory with Parity checking or an address space of 0-757777(8) bytes, plus a UNIBUS addressing space of 20(8) Kbytes, i.e. 760000-777777(8).

Device	Controller		Function	Address	Vector	BR
-	KD11-EA	M8266	CPU, board 2	-	-	-
-	"		CPU, board 1	-	-	-
			with Memory	772300...356	250	-
			Management at	777572...656		
-	FP11	M8267	Floating Point Unit	-	244	-
-	MR11-EA	M9312	Boot diags,Console Emulator	765000	-	-
	"	"	DLn: and	773000	-	-
-	KY11-LB	M7859	Programmers Console with Console Switch Register	777570	-	-
-	MS11-LD	M7891	MOS memory with	0-757777	-	-
-	"	(M7850)	Parity Controller	772100	114	-
TT0:	DL11-W	M7856	VT-52 Console RS232-C	777560	60	4
TT1:	DL11-W	M7856	LA-36	777530	300	4
			@20 ma, 300 baud	776524	334	4
PR0:	PC11	M7810	Paper Tape Reader	777550	70	4
PP0:	"		Paper Tape Punch	777554	74	4
RL0:	RL11-AK	M7762	RL02-AK drive	774400	160	5
RL1:	"		RL02-AK drive	"	"	"
MT0:	TM11		800 bpi Magtape	772520	224	5
MT1:	"		800 bpi Magtape	"	"	"

********** Devices available without Drivers**********

KW11-K	PROG. CLOCK	170404	444	-
AD11-K	A/D CONVERTER	170400	340	-
AA11-K	D/A CONVERTER	170416	360	-

4.4 PDP 11/44 SYSTEM SOFTWARE (BLDG. 1120)

4.4.1 PDP 11/44 Sysgen

A new Sysgen for the PDP 11/44 was carried out due to a depletion of dynamic pool (DP) space. The DP is a segment of physical memory reserved for use by the operating system for maintaining dynamic data structures (e.g., file control blocks, task control blocks) and for buffering data during I/O (.e.g., terminal input characters). A listing of the usages of DP can be found on 5-29 of the RSX-11 Sysgen Manual. The operating system is responsible for managing this resource and allocates blocks of memory in the DP on a "first-fit" basis. The size of the DP memory region is limited by the virtual address space available to the executive task. When the operating system is assembled with VMR following the sysgen process, the DP space is allocated. In the case of the currently used system, the maximum amount of pool space available was requested. It does follow that if the size of the executive built during the sysgen process was reduced, more DP space would be available. Therefore, several steps were taken to alleviate this problem by removing unnecessary requests for DP. The new sysgen decreased the size of the executive and therefore provided for increased DP. The changes in the operating system produced by this new sysgen are:

1. No resident DK:(RK05), PP:, PR:(paper punch), CR:(card reader), or MT:(magtape) device drivers were created. These devices were either removed from the system since the last sysgen or were never included in the actual physical system. It should be noted that if any one of these devices is needed in the future, a loadable device driver can be included into the operating system without going through a complete sysgen.

2. No ANSI magtape support is available.

3. No on-line formatting or diagnostics are available. The only formatting needed on the system is for the 300 Mbyte disks which have always been done off-line with the custom formatter provided by Xylogics.

4. No crash-dump analysis. However the XDT executive debugger will continue to be executed on system crashes.

All custom device drivers (e.g., AP:, SD:, VD:, MP:) were rebuilt for the new executive along with the custom BYE task. The new executive file and system task files were replaced on the 300 Mbyte system disk. The virtual disk used for the sysgen was copied back onto the SYSGEN virtual disk on the old 11/34 system disk pack.

A gain of approximately 800 words of DP was realized following all of these procedures. This should meet the demands placed on the PDP 11/44 system for the near future. The next course of action, should demands exceed the available DP resource, is to generate a version 4.x of the RSX-11M operating system which has provided a means of significantly reducing the virtual address demands of the executive task and consequently provides more DP space.

4.4.2 Data Transfers

Interactions between the PDP-11/44 and certain non-Unibus devices have been provided by ARCON. A program called UPLOAD was written to upload text files from the Zenith PC to the PDP-11. This utility can be used to transfer the DRT data received from Dynastat on MS-DOS formatted 5-1/4" floppy disks. More generally, it will transfer any MS-DOS ASCII files and store the information in FILES-11 format on any of the disks available on the PDP-11 system. The terminal emulation program, ZSTEM, on the PC provides the access to the floppy disk data and directs the upload activity on the PC's serial interface port tied to the PDP-11. Having this capability allows an analyst to transfer data between systems easier.At times it has been necessary to transfer data between the PDP-11/34 and the PDP-11/44 or between the PDP-11/44 and the ARCON VAX/VMS system. A route for the interconnection of the PDP-11/34 to the PDP-11/44 consists of using the RL02 disk drives on the PDP-11/44 and the magtape on the PDP-11/34. The conversion from one medium to the other (RL02 <-> Magtape) is provided by the third PDP-11 at RADC/EEV which has both of these devices. All data can be formatted as Files-11 since all machines involved run under the RSX-11M operating system. As for the interconnection between the PDP-11/44 and the VAX, they both have an RL02 disk drive and the VAX has the software capability of transforming FILES-11 format to fit its systems needs.

Another set of routines were written to provide serial communication between the PDP-11 and another computer. The Fortran programs PC2PDP ("PC to PDP"), PDP2PC ("PDP to PC"), and XMODEM are all XMODEM-protocol file transfer programs. They offer the option of binary transfer, and for ASCII files they allow the user to distinguish between PDP-11 files (such as Fortran source) that rely on implicit CR/LF characters between records, and PDP-11 files (such as Runoff output) that contain all their carriage control characters explicitly. [These programs do NOT perform conversion to or from Fortran column-1 carriage control format.] All source code, command files and a documentation file for these routines are located on the REPT88 virtual disk. The assembly language package TTIN does the PDP-11's input during file transfer, storing received characters in a circular buffer that is currently 512 characters long.

It must be noted that at high speeds (9600 bits per second or so), file transfers from the PC to the PDP-11 may fail if other users are on the PDP-11. But failed transfers will be detected as such, and can be retried later. This problem can be solved by running PC2PDP at a priority of 71 decimal (107 octal) or higher, but then other users will experience a gross degradation of throughput.

4.4.3 User Enhancements

The STARTUP.CMD file has had minor modifications and is now fully documented to ease future modifications. This new file is found on DR:[1,2].

The HELP utility has been used to provide a new user with an understanding of the programs and data bases available on the system. New help files provide information on special login accounts, the contents of virtual disks, and the system backup procedures. All help files are on DR:[1,2].

The help file HELPVDK.HLP will assist users of the PDP-11/44 in the location of source code. It can be accessed simply by entering HELP VDISKS at a terminal. A list of the available virtual disks, how to access them, whether they are backed up, what type they are, and what each of them contains will scroll on the terminal screen.

The help file HELPACN.HLP will list the currently active system, backup, and research and development accounts on the system. It can be accessed by typing HELP ACCOUNTS at a terminal. Specialized research accounts along with utility accounts for system shutdown, backup procedures, device setups and initialization will be listed.

A unified backup procedure was developed to protect the various virtual disks and data bases. The help file HELPBKU.HLP will assist the user in backing up the appropriate disks. A user can obtain this listing by entering HELP BKU at any terminal.

All virtual disks have been indexed through the use of READ.ME files on each disk. These files will be as current as their creation date and should provide the uninitiated with a basic understanding of the UIC and file contents of the disk.

4.5 PDP 11/34 SYSTEM SOFTWARE (BLDG. 1120)

The new PDP-11/34 runs an RSX-11M version 4.0 operating system. One distinctive difference between the version 3.2 RSX-11M operating system and the version 4.0 system, is the ability to use command line interpreters (CLI) other than just MCR. DCL is the other popular CLI available on RSX-11M version 4.0 and many of the login command files on the system assign DCL as the default CLI. The user can go from one CLI to the other with simple terminal commands.

Some of the system software provided with the DCEC system on the PDP-11/34 was explored. It was discovered that a half-duplex terminal driver was generated for this RSX-11M executive. The EDT editor can not be operated in Keypad mode with a half-duplex terminal driver. The Fortran-77 compiler on the system was also tested. An unresolved problem was encountered at Task Build time in that the required object modules called by the Fortran-77 modules could not be located even when the F77.OLB library was explicitly indicated in the TSK command input. Complete documentation for the Fortran-77 compiler has not been located.

Speech algorithms such as LPC-10, CVSD, APCSQ, along with others are readily available for use with the MAPS on the 11/34 system. A command file for executing the speech algorithms on the MAPS can be found at DB:[6,100]MPSPEAK.CMD. These algorithms were tested and execute as expected.

For easy access to these speech algorithms, a user can simply log in on the PDP 11/34 under the account SPEECH/RUN. The user will be prompted for a MAP-300 designator (A or C), shown a list of the algorithms available, and asked to enter the algorithm name the analyst wishes to execute.

4.6 PDP 11/34 SYSTEM SOFTWARE (BLDG. 1124)

Currently, the dedicated application for the PDP-11/34 is the collection of EPL and Speech/Noise ratio data from DRT source tapes [Ref. 9] and a digital bit stream error generation task (BER). The EPL software was revised such that the program no longer needs to be run in the absence of an operating system (i.e. stand alone), but is able to execute as a RSX-11M task. The BER task simulates transmission channel error conditions modeled on magtape and utilizes the PDP-11's D/A capability to output TTL level bit streams [Ref. 9]. This software was also modified to operate under RSX-11M.

4.7 SYSTOLIC ARRAY PROCESSOR PROGRAM DEVELOPMENT SOFTWARE

The SAP is not a stand-alone computing system as it requires streams of externally generated commands and data in order to do meaningful work. The PDP-11/44 serves as a host for the SAP and its Program Development Software. The host must operate both as a control store and sequencer, and as a file server and data formatter. Communications between the host and the SAP is by way of two unidirectional busses with a maximum transfer rate providing a command to the SAP approximately every 3 microseconds. In using a minicomputer as a sequencer, command transfer speed is sacrificed for the flexibility offered by an environment wherein program changes can be easily and quickly made. The minicomputer as file server provides the SAP with access to large segments of speech data stored on disk and also provides storage for data produced during the algorithm's execution.

Software was developed on the minicomputer to provide SAP programming and "execution" capabilities Both an interpreter and a compiler were developed. Command and data transfers with the SAP and host directives are enacted as the interpreter analyzes an input source file. The compiler generates a binary object file encoding the stream of SAP commands and host directives to be actually executed at some later time.

Interacting with the SAP via the interpreter provides a means of quickly debugging small routines; on the other hand, the execution of compiled code provides significant execution speed advantages. A SAP execution or executive program is available which reads a specified object file, scans the commands and host directives encoded in the file, performs the requested host directives, transmits blocks of SAP commands to the SAP, and performs data I/O between the SAP and the host.

The instruction set for the SAP consists of the basic 32 instructions recognized by the SAP's interface and control module (e.g., MD for move data on previously selected inter-nodal path), and eight directives for operations to be performed by the host (e.g., GB for get a block of data from a previously opened file). Symbolic variables, assignment statements, expression evaluation, and looping constructs are all available in the SAP's programming language. The source code input to the interpreter and compiler tends to be very cryptic and redundant. It is difficult to convey the logical steps of an algorithm with just the SAP source code mnemonics. A macro preprocessor was developed in response to this problem. A "library" of commonly used higher level operations can be created by macro definitions and included into a user's program as needed.

The interactions of the various SAP program development tools are shown in Figure 4.6. Their specifics will be addressed in the following sections. Program source code, task build and overlay command files, program instructional files for the various routines resides on the virtual disk SAP along with and a complete set of development memoranda (SAPMEMO.RNO). The program tasks are located on the system disk DR: at UIC [1,54] and are installed.

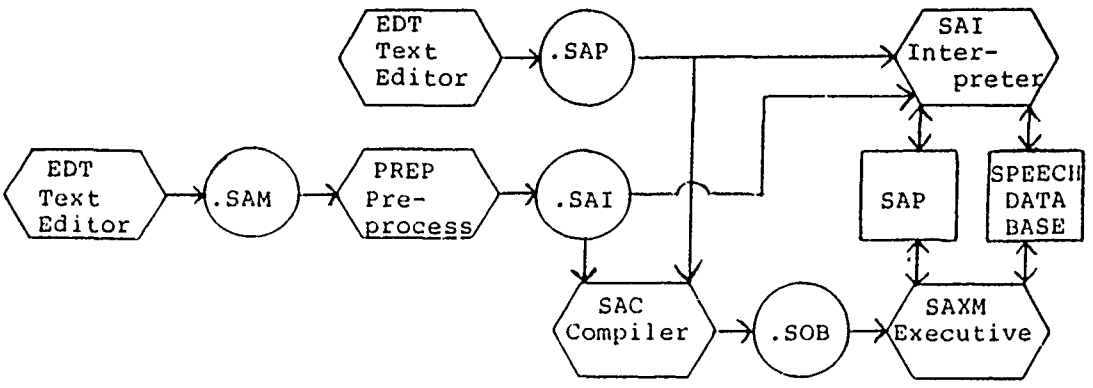


Figure 4.6
SAP Program Development Software

4.7.1 SAP Microcode Interpreter - SAI

A microcode interpreter (SAI) based on NOSC software was developed to provide a means of quickly debugging small routines. The SAP interpreter prompts the user for the input source file of SAP command mnemonics and their operands as well as additional commands specified to perform operations in the host program such as disk I/O and terminal display and then proceeds in converting these commands to their binary counterparts and sending them to the SAP in the requested sequence. The commands currently available besides the basic 32 SAP commands are:

1. FO--opens a disk file specified by name for either input or output.
2. FC--closes a specified open disk file.
3. GB--gets a block of data from specified disk file.
4. PB--puts a block of data into a disk file.
5. CO--converts either PDP integer or floating point data array into a separate array of Intel floating point values (4 byte each), or performs the opposite conversion from Intel format to the PDP format.
6. DO--begins a "DO LOOP" for multiple iterations of the same block of commands
7. EN--delimits end of "DO LOOP".
8. ST--delimits end of source file

The parameters for the commands can be represented as symbols and expressions formed by symbols along with the operators +,-,/,*, and parentheses. Also hexadecimal constants can be specified for these parameters which is useful since much of the original documentation for the SAP instructions from NOSC refer to these parameters in hexadecimal notation. Assignment of values to symbols is accomplished by a standard assignment operation denoted by '='. A summary of the features of SAI are:

1. Use of symbols to identify variables to which integer values can be assigned using an assign statement. (Currently room in symbol table for 20 symbols).
2. Use of expressions for assignment values and for some of the parameters for the SAP commands and the host directives.
3. Use of decimal or hexadecimal constants.
4. Up to 4 levels of DO-LOOP nesting.
5. Two data "channels" between disk files and the 1024 byte RAM buffer in the SAP (the SAP BUFFER). Each channel can be associated with a distinct disk file. Data conversion between PDP-11 integer or floating point format and Intel floating point format is performed in the channels. The "scattering" of data bytes for the Intel floating point values is also performed to allow loading of data into processing elements of the SAP in linear, row/column, or matrix format. An inverse process of "gathering" the data read back from the SAP is also available.
6. Literal string data delimited by double quotation marks are displayed on the terminal at run time.

The syntax for all of the commands and host directives used by the SAI program are documented in the instructional file SAIUSE.DOC. A file SAIDOC.DOC gives software details of the SAI program.

4.7.2 SAP Program Compiler And Executive - SAC & SAXM

As anticipated, the SAI program which must translate each SAP command before sending it to the SAP for execution results in very inefficient use of the array of processors in the sense that the SAP must spend a great deal of time waiting for the PDP-11 to translate and transmit instructions. However the SAI interpreter does provide a means of easily changing a SAP program under development as this involves editing the SAP source file with EDT. Once the SAP algorithm has been debugged and is ready to run using a large stream of data (speech data for instance), then a faster means of feeding instructions and data to the SAP is needed than can be provided by the SAI utility.

Since the execution of compiled code provides significant execution speed advantages, a compiler was written, SAC, which translates the same set of SAP command mnemonics and host directives (e.g., open a data file, read a block from a data file) as does SAI. SAC, however, stores the translated binary information in a disk file instead of sending it to the SAP for execution. Parameters associated with host directives are also placed in the binary file. Differentiation of host directives and SAP commands is facilitated by having all host directives coded as negative values (all information in the file is stored in 16-bit word units) while none of the SAP commands are ever coded as negative values.

The user of SAC is only required to provide a source file of SAP commands in the same format as used by SAI and an output SAP object file. A file naming convention to use might be .SAP extensions for source files and .SOB for object files.

The execution of SAP object files created by SAC is performed by "SAP execution" program SAXM. This program reads a specified .SOB file, scans the commands and directives coded in this data stream, performs the requested host directives, transmits blocks of SAP commands to the SAP, and performs data I/O between the SAP and the host.

Refinements to the SAC and SAXM programs were made as experience was gained by actually trying to program the SAP. The command stream to the SAP tends to be very repetitious with the same 10 to 15 command segments being sent multiple times. Also signal processing algorithms applied to real-time signals as found in speech research also tend to be redundant with each block of input data processed with the exact same set of commands as all the previous blocks. Thus a DO-LOOP construct has been included with the SAP programming tools to provide multiple iterations of the commands and directives found within the body of the DO-LOOP reserved words "DO" and "EN". Actually, two variations of the DO-LOOP constructs have evolved. An invariant DO-LOOP can be specified by the programmer with the "DO/I" command in the source file while a DO-LOOP which contains command parameters which change from one iteration of the loop to the next is specified

simply as "DO". The manner in which these two variations of the DO-LOOP are processed by the SAC and SAXM programs differ significantly and the way in which a programmer uses them will greatly affect the execution speed of the algorithm.

A simple "DO" loop is expanded by the SAC program such that multiple copies of the commands within the loop are included with the .SOB object file. The commands from one iteration of the loop to the next may not exactly match those of other iterations. It is obvious that the length of code generated is a function of the number of instructions within the DO-LOOP and the number of specified iterations of the loop. If the code within the DO-LOOP body is invariant from one iteration to the next, then only one copy of this command stream is needed along with a parameter indicating the number of iterations of the loop requested. A large reduction in the size of the .SOB file is made possible by the use of DO/I when the desired loop is invariant.

Another, perhaps more important advantage is offered by the discrimination of invariant and variant DO-LOOPS. The transfer of SAP commands from the PDP-11 can be substantially increased if it is known that a finite set of instructions is to be sent to the SAP over and over again. Instead of having to read multiple iterations of the same commands and directives from a disk file, the copy of the loop's commands and directives can be buffered into RAM on the first pass through the loop with all further iterations requiring no more disk I/O which is very slow compared to accessing all of the required data in RAM.

In anticipation of the need to write speech processing algorithms which will require several thousand SAP commands (all contained perhaps in one "outer" DO-LOOP) to process one block of speech data, the RAM resources offered by the upper 1 Megabyte of the PDP-11/44's memory have been tapped. The most recent version of SAXM has the capability to map this memory region as needed to store up to 512 K SAP instructions and to transfer these commands from PDP-11 RAM to the SAP interface at almost maximal rates.

4.7.3 SAP Macro Preprocessor - PREP

The SAP source code written for input to the SAI and SAC interpreter and compiler tends to be very cryptic and redundant. It is difficult to convey the logical steps in an algorithm with the symbols used for the SAP compiler. It was decided that a more efficient programming environment could be created by the introduction of a Macro preprocessor in which many instructions making up one logical operation in a SAP program can be represented by a single or very few symbol(s). A "library" of commonly used operations can be created by these Macro definitions and included into a user's program as needed.

The Macro preprocessor was developed and is available for writing source code for the SAP interpreter (SAI) and compiler (SAC). User instructions are available in the file PREP.DOC.

An input file to the preprocessor includes the SAP commands and host directives mnemonics normally provided to SAI and SAC, as well as directives to the preprocessor for declaring macro definitions, specifying include files, and defining conditional macro expansions. The macro preprocessor's directives are:

.MAC --macro definition
```
     syntax:  .MAC macroname {.......macro definition.......}
     example: .MAC APU {SS 03, LA %1, SS 01, MS, SL, IT, WT, FS}
     NOTES: 1) A "%n", where n is a number between 1 and 10,
               denotes an argument reference to be provided
               when the macro is actually called. A leading
               space is required before the "%" character
               unless it is the first character in a line.
            2) Carriage return/linefeeds within the macro
               definition are retained and included in the
               macro's expansion.
            3) Macro definitions cannot be nested within other
               definitions although calls to previously
               defined macros within a definition is permitted.
               Limit the nesting of macro calls to 10 levels.
```

.IF --conditional macro definition
```
     syntax: .MAC macroname {  .if %n = xxx {....conditional
                               definition}     }
     example: .MAC DOIT {SS 03, LA %2, SS 01,MS,
                         .IF %1 = FAST {IT,WT}
                         .IF %1 = SLOW {SL,IT,WT,FS} }
     NOTES: 1) Only "equal to" conditional operations are
               allowed with the left side of the operator always
               being an argument reference and the right side
               being a string constant.
            2) Text within the "{}" following the conditional
               statement are included upon expansion only
               if the supplied argument to the macro call
               matches the string constant. Otherwise
               the material is excluded upon expansion.
```

.INC --include file directive
```
     syntax: .INC filename
     example: .INC MACRODEFS.SAM
     NOTES: 1) The specified file is open for input and the
               preprocessing continues using the source
               code within the include file. Upon reaching
               the end of the include file, the preprocessing
               resumes in the original source file at the
               statement following the .INC directive.
            2) Nesting of .INC directives in include files
               is not permitted.
```

MACRO CALLS

syntax: macroname (arg1,arg2,....,arg10)

example: DOIT (FAST,0CH)

NOTES: 1) Expansion of the previously defined macro begins. If the specified macro has not been previously defined, the macroname and argument list will be copied to the output file.

2) Null arguments can be specified by ",,". Leading spaces in argument strings are not stripped out.

3) Arguments can consist of macronames and argument references.

4.7.4 Further Modifications

In an attempt to meet the programming needs for the Canonical Coordinate (CC) project, the SAXM (execution) and SAI (interpreter) programs include the following capabilities:

1. The source input file for the SAXM or SAI program can be included in the MCR command line used to invoke the task. Thus, indirect command files can be developed for controlling the sequential execution of "SAP programs".
2. Upon the detection of End-of-File (EOF) during a read operation from a data file, all open data files are closed and the task is terminated.
3. Upon the detection of EOF during a write operation to a data file, a subroutine (EXTEND) is called to request an extension of the file by 50 blocks. The write operation is attempted a second time after a successful extension.

The macro preprocessor (PREP) and compiler (SAC) tasks also accept file specifications in the MCR command line invoking them. The specification's syntax is "output file = input file". (Note, RSX-11 tasks must be installed with ...XXX task names in order to access the command line buffer via the GETMCR system directive.)

CHAPTER 5

COMPUTER DATA BASES AND ANALYSIS TOOLS

5.1 INTRODUCTION

The data at RADC/EEV has developed over several years and includes every thing from DRT results to samples of processed speech. It has been organized into several data bases which include:

1. Speech Data Base
2. DRT Raw Data Base
3. DRT Analysis Data Base
4. Raw & Processed Acoustic Noise Data Base
5. Annotated Graphics Package (AGP) Noise Data Base
6. Error Metric Data Base

ARCON was responsible for incorporating two large format disk pack random access devices into the EEV facility. This extensive random access memory allowed for the design of a general speech data base (DB) accessible by all processors. The first device, DR0:, is used as the System default disk and for the storage of most pseudo disks. The second device, DR1:SPEECH, is used for this general speech data base and has the pseudo device name SP:. ARCON has designed, developed and installed this Speech Data Base at the RADC Speech Processing Facility. This DB allows A/D and D/A real time I/O to files from all processors that have this capability. In addition, and of primary importance, raw data files can be downloaded to any processor for analysis. The resulting processed data files can be directly compared with processed files from any other device/algorithm analysis operation within the system. A major advantage in the use of DB file data for comparison is an exact start point in the data frame for any analysis. The header file attached to all DB files contains the information required to trace the processing that has been accomplished.

Data base management software has been developed by ARCON to locate and sort DB files based on the header information. Speech Data Base files can contain either fixed point or floating point data. The speech DB has been matched to commercial ILS software installed by ARCON. The ILS package can be used for the analysis and display of fixed point Speech Data Base files. An ARCON Speech Data Plot Package can handle fixed or floated files and allows numerous options for file comparisons.

The DRT data consists of a collection of unscored DRT system files (XXXX.ZFY's) representing individual listener responses and the DRT Analysis Data Base. An extensive set of comparative analysis routines has developed over several years along with utility routines for management of and searching through these data bases. Current status of the data base structures, data collection test scoring and utility software can be found in the recent ARCON report "RADC/EEV Diagnostic Rhyme Test System Improvements" [Ref. 11].

Acoustic background noise data has been collected in two ways. Raw noise data has been digitized at set sampling rates and is available for a large number of environments in files of the Speech DB format. The Spectral Dynamics SD360 dual channel processor has the ability to collect and analyze data under computer control with AGP software operating on the PDP-11/44 Noise data collected in this manner is available in the AGP data base which has been discussed in detail in Reference 9.

Canonical coordinate research has resulted in a large collection of error metric files. These files are used by the CC analysis and/or synthesis routines for processing speech. They are stored in floating point Speech DB format but have a size and structure uniquely their own. A 3-D display version of these files makes up a subset of this data base.

The previous DB descriptions are an indicator of the many types of data used in speech/noise research. Tools for the management, searching, sorting, display, and analysis of these DB's provide for the full utilization of the data.

5.2 RADC/EEV COMPUTER DATA BASES

5.2.1 Speech Data Base

ARCON Corporation has designed, developed and implemented a speech data base for use at the RADC/EEV Speech Processing Facility. This data base, along with accompanying software, enables users to store and manipulate data from any of the various processors at the Speech Lab. Details on the development of the Speech DB can be found in References 9 and 10.

The Speech Data Base is designed such that all data is stored in files of a standard format. These files are arranged on a "public" (i.e., available to all system users) disk, and can be accessed through calls from user-written programs to a standard subroutine library (QIOLIB). Data is stored in direct access, block I/O files, in either 16-bit integer or 32-bit floating point formats. Block I/O, in which 512-byte segments of data are read or written at one time, is the fastest and most efficient method of file access. It is well suited to our applications, in which disk I/O can be the rate-limiting factor in real-time processing.

All speech data base files are created and manipulated by the standard routines from QIOLIB. Each file has a one-block (512 bytes) header record which defines its content. This header record enables the users to label data files with such information as: whether the data is raw (i.e., straight A/D, no processing) or processed; type of internal format; whether it is real or complex; source processor; processing algorithm used; creation date; title; and any comments. Table 5.1 shows the format of these header records. This header format has been slightly altered to be congruent with the data file format of ILS (see the accompanying section on ILS).The 16-bit integer format is the same as the ILS "sampled data file" format; however, there is no provision in ILS for a floating point sample data file.

The Speech Data Base resides on one of the CDC 300 Mbyte disk drives and is assigned the global pseudo-device name SP: which is declared a "public" device when the operating system is brought up. The QIOLIB routines which open and/or create new files use SP:[200,200] as the default device and directory unless the user specifies otherwise. Therefore, all files referred to by name only are assumed to be located on the Speech Data Base disk. However,if for testing purposes the user wishes to locate files on other volumes or at other UIC's, simply specifying the volume explicitly will cause the file to be accessed there.

The Speech data base format has been used for the organization of data other than speech. The DB can be considered to contain subsets of information. The major subset is speech material and is referred to as "the Speech DB". The remaining subsets are the Digital Acoustic Noise DB and the Error Metric DB.

Table 5.1
FILE HEADER BLOCK FORMAT SPEECH DB AND ILS DATA FILES (SEPT 1988)

Variable types: A - ASCII, F - Fl. Pt., I - Integer, R - RAD50
Lengths given in words. '*' indicates speech data base variable.

NAME	START	LENGTH	TYPE	CONTENTS
N (IFRAME*)	1	1	I	NUMBER OF POINTS PER ANAL. WINDOW
M	2	1	I	NUMBER OF AUTOREGRESSIVE COEFFS.
ICON	3	1	I	PREEMPHASIS CONSTANT (0-100)
NSHFT	4	1	I	SHIFT INTERVAL PER DATA FRAME
IHAM	5	1	I	HAMMING WINDOW ('Y' OR 'N')
NSPBK (NBLKS*)	6	1	I	# BLOCKS IN FILE (NOT INCL HDR)
NP	7	1	I	NUMBER OF RESONANCE PEAKS
ISTAN	8	1	I	STARTING FRAME FOR ANALYSIS
NAN	9	1	I	NUMBER OF FRAMES ANALYZED
NFR	10	1	I	# OF VAR. SIZED FRAMES ANALYZED (PAN)
MU	11	1	I	# OF AUTOREGRESSIVE COEFFS. (PAN)
NT	12	1	I	DOWN-SAMPLING FACTOR

Table 5.1 (Continued)
FILE HEADER BLOCK FORMAT SPEECH DB AND ILS DATA FILES (SEPT 1988)

NAME	START	LENGTH	TYPE	CONTENTS
IFLD(1)	13	1	A	FIELD 1 - 2 ALPHABETICS
IFLD(2)	14	1	A	FIELD 2 - 2 ALPHABETICS
IFLD(3)	15	1	A	FIELD 3 - 2 ALPHABETICS
IFLD(4)	16	1	A	FIELD 4 - 2 ALPHABETICS
NSC	17	1	I	STARTING SECTOR FOR ANALYSIS
IAFIX	18	1	I	FLAG FOR AUTOREGRESSIVE COEFF. (PAN)
IDK	19	1	I	DISK # OF DATA FILE ANALYZED
NFL	20	1	I	FILE # " " " "
ID (1)	23	1	A	IDENTIFICATION - 2 ALPHABETICS
ID (2)	24	1	A	IDENTIFICATION - 2 ALPHABETICS
ID (3)	25	1	A	IDENTIFICATION - 2 ALPHABETICS
ID (4)	26	1	A	IDENTIFICATION - 2 ALPHABETICS
ID (5)	27	1	I	IDENTIFICATION - NUMERIC
NASC	28	1	I	NEXT AVAILABLE SECTOR (TTL)
NAPT	29	1	I	NEXT AVAILABLE POINT (TTL)
NZERO	30	1	I	NUMBER OF ZEROS (TTL)
FLAG	31	1	I	=< 1111 IF SECONDARY FILE INITIALIZED (TTL)
ITYPE*	32	1	I	DATA TYPE: RAW = 0, PROCESSED =1
IFRMAT*	33	1	I	DATA FORMAT: FLT PT =0, INTGR =1
IRLCX*	34	1	I	DATA FORMAT: REAL =0, CMPLX =1
IDEV*	35	1	I	SOURCE PROCESSOR (INTGR CODES)
ENGPW*	36	2	F	ENERGY/POWER
SGNOI*	38	2	F	SIGNAL/NOISE RATIO
RAWFIL*	40	5		SOURCE RAW FILE NAME - FORMAT:
IRFLNM(3)*			R	3 WDS. - RAD50 FILE NAME
IRFLEX*			R	1 WD. - RAD50 EXTENSION
IRFLVR*			I	1 WD. - INTEGER VERSION #
IALG*	45	1	I	COMPRESS. ALGORITHM (INTGR CODES)
ISNCOM*	46	1	I	SYNTHESIZED = 0, COMPRESSED = 1
IWINDW*	47	1	I	WINDOW FUNCTS. USED (INTGR CODES)
IHFLT*	48	1	I	FILTER - HI LIMIT
ILFLT*	49	1	I	FILTER - LO LIMIT
BLANK	50	7	-	UNUSED
ICHAN	58	1	I	STARTING A/D CHANNEL
NCHAN	59	1	I	NUMBER OF CHANNELS
MULAW	60	1	I	SET TO 50 IF 8-BIT LOG QUANT. (REC)
IPWR*	61	1	I	SAMPLING FREQ. =
IFRQ*	62	1	I	IFRQ * (10**IPWR)
FLAG	63	1	I	ILS FILE TYPE -32000 = ILS SAMPLED DATA FILE -30000 = ILS RECORD DATA -29000 = ILS ANALYSIS DATA -31000 = FLOATING POINT AND NOT ILS
XXXXX	64	1	I	= 32149 INITIALIZED ILS SAMPLE DATA FILE
NSIZ1*	65	1	I	NUMBER OF SAMPLES =
NSIZ2*	66	1	I	NSIZ2 + (NSIZ1*32767)
MAXCUR*	67	1	I	CURRENT MAXIMUM AMPLITUDE
MAXORG*	68	1	I	ORIGINAL MAXIMUM AMPLITUDE
DATE*	100	5	A	CREATE DATE - (DD-MMM-YY)
TITLE*	105	10	A	DESCRIPTIVE TITLE - 20 CHARS MAX
COMMNT*	115	142	A	COMMENTS - 282 CHARACTERS MAX

Digital Acoustic Noise Data Base -

This DB contains digitized samples of noise recorded from various operational platforms under numerous configurations. The files can be raw 8 kHz. bandwidth signals, filtered versions, LPC processed versions, processor residuals, etc. The one unifying factor is that all members of the DB can be traced back to a raw noise sample in the DB. Its structure is similar to the Speech Data Base in that it contains a one-block header record to indicate the format and origin of its content. These noise files reside on the device SP: under the directory UIC [200,230] and are used in conjunction with the speech files for the study of different effects of noise on speech.

This DB contains various versions of the acoustic noise data. These include various types of raw input data, compressed versions of the data and synthesized data. A brief description of these file types and their identifying file extensions are given in the following list.

1. Input Data
    1. .NOI - Raw Data Files Sampled at 16 kHz.
    2. .FLT - Downsampled and filtered .NOI files (8 kHz.)
    3. .BBN - Special extended noise data files

2. Compressed Data
    1. .ANA - ILS LPC Parameter files
    2. .RES - ILS LPC Residual files

3. Synthesized Data
    1. .LPC - LPC-10E V52 Synthesized data
    2. .ILS - ILS LPC Synthesized data
    3. .CCX - Canonical Coordinate Synthesized data

All of the Speech DB I/O routines are usable with signal files in the Acoustic Noise DB. Special care has to be taken by the analyst when he adds new raw data files (.NOI) to this DB. All noise samples within the data base are assumed to have been sampled at 16 kHz. with 8 kHz. anti-alias filtering being provided by the RADC/EEV Precision Filter Set. This facilitates comparisons between noise platforms. The 8 kHz. bandwidth was chosen to be meaningful for speech research and still be able to identify important high frequency audio components of the noise. A downsampled version of the .NOI data has been generated for use with analysis tools that expect a 4 kHz. bandwidth. These files have been generated using the ILS routines FLT with an anti-alias filter designed with the ILS filter routine EFI. The command strings used and a plot of the filter characteristics are given in Figure 5.1.

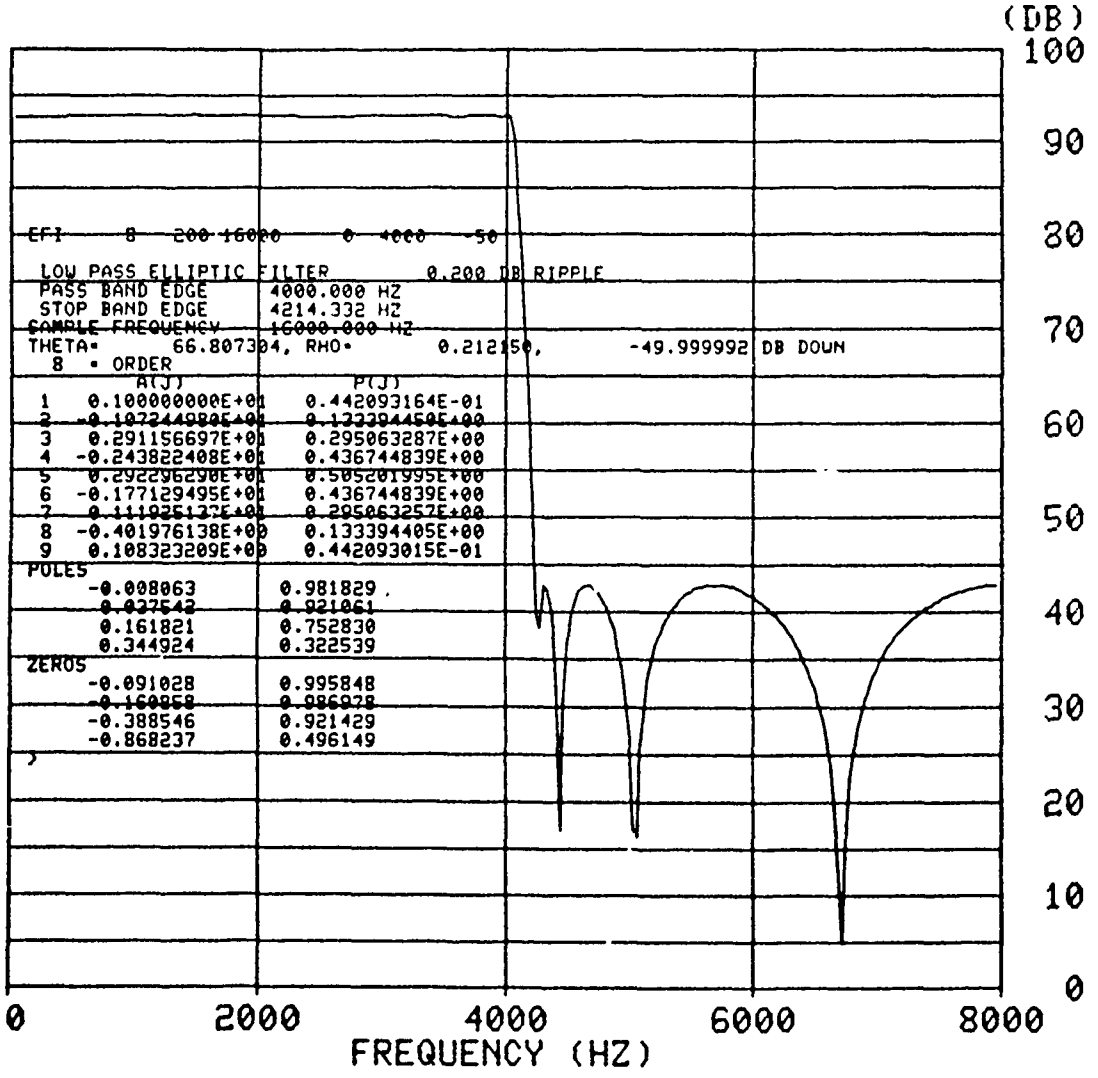


Figure 5.1
FLT Filter Characteristics

Error Metric Data Base -

This DB is used by the Canonical Coordinate research efforts of Chapter 2. These are floating point files with header information and are of the Speech DB format. This data base is recognized by the file extension .ERM and is segregated to the SP: device at UIC [200,200]. The error metric files are application specific and are used for the CC analysis and synthesis of speech. Implementations of various statistical, analytical and empirical algorithms have been used for the generation of the Error Metrics. Eigensystem solution algorithms and Unitary Matrix series expansion methods are included to generate the transformation matrices and eigenvalues required by the Canonical Coordinate methods. Details on the development of these Error Metrics are given in Chapter 2 of this report.

5.2.2 PC Data Bases

There also exist at the Speech Laboratory two other data bases which are worth mentioning. These DBs are accessed through Zenith PC clones. The first is a Technical Information Data Base. This DB contains information about all the technical and final reports located at the Speech Laboratory. The utility program PCFILE 3 provides the capabilities of finding, adding, deleting, sorting, and listing the records of the requested information the user needs.

The second DB is referred to as an Archive Data Base and it is a library data base that will contain information of all audio tapes in the test and evaluation laboratory. The utility program DBASE III provides the user with the capability to add, modify, locate, sort, and list the records available at the RADC/EEV Speech Processing Facility. Because of the large amount of data to enter, this data base is still being added to. The tapes included in this data base are digital DRT/DAM Masters, analog DRT/DAM Masters, and all processed tapes submitted by various organizations.

5.3 SPEECH DATA BASE ANALYSIS AND EVALUATION TOOLS

Software routines provide a set of general functions to facilitate the use of the Speech Data Base. These functions are data collection or generation; DB I/O; DB sorting and retrieval; data processing and analysis; data display; DB maintenance utilities. The Interactive Laboratory System (ILS) software package [Ref. 19] is integral in the support of the Speech DB and its subsets, the Noise DB and the Error Metric DB. Most of the data analysis and processing functions have been implemented as custom ILS programs. The following sections will discuss the development or modification of programs that provide these functions for the Speech DB. These programs can also be used with the Digital Acoustic Noise DB since it is a subset of the Speech DB. Previous program developments can be found in References 9 and 10.

5.3.1 Speech DB Input/Output

ARCON has developed I/O systems on the two array processors, the AP-120B and the MAP-300, available at the RADC/EEV Speech Processing Facility. These systems provide a way of testing speech algorithms written for the processors, and a way of transferring speech from or to the Speech Data Base.

The MAP-300 has two routines MAPIN and MAPLP. The MAPIN program can be used to input audio data to the Speech DB of as a loop-back I/O test of the MAP-300. The MAPLP program is the primary Speech DB output method at EEV. It has numerous options and will be discussed in detail.

The programs APIN and APOUT are used with the AP-120B for Speech DB input and output. These programs run on the PDP-11 and make use of the interface between the AP-120B and a modified IOP-16 scroll board. The IOP provides four 12 bit wide channels each of input and output for the AP-120B. A separate routine APLOOP is used for loop-back I/O testing of the AP-120B.

MAP-300 Input/Output Software -

ARCON designed and implemented a MAP-300 Speech Data Base routine called MAPIN that performs a real-time analog to digital conversion of a speech signal on one channel of the ADAM board. This data is either stored on the Speech DB disk in standard speech DB format or, if in loop back mode, is sent to the AOM board on the MAP-300 system for digital to analog conversion. If in data storage mode, this routine also sets up the header and allows for the analyst to add or modify any of the information within the header before it is written to the speech file. This program has not been modified during this contract period and was completely described in Reference 9.

ARCON developed another MAP-300 speech data base routine called MAPLP which performs the reverse of the operation done by MAPIN. A speech file is read from the Speech Data Base, transmitted to the MAP-300, and sent out to the AOM board, all in real-time. MAPLP not only performs D/A conversion of Speech DB files but also has two added features that make it more useful for the Speech Lab Facility's purposes. The first feature is the ability to control two channels of programmable precision filters. The second feature allows for multiple sets of output blocks from Speech DB files to be defined for output. Within each of those multiple sets, the analyst has the capability of selecting the starting block number, the number of blocks to be converted, and the number of times the set is to be repeated, as well as the option to select a scaling factor or reset the sample rate. This routine has been extensively modified during the current contract period and will be discussed below.

Separate ILS versions of both MAPIN and MAPLP have been developed for the input and output of audio data to the Speech DB from within the ILS package. These routines will be discussed in a later section.

Program MAPLP - The Speech Data Base output program MAPLP may be executed with the "RUN $MAPLP" command. Since it utilizes the MAP-300, this processor must have been previously initialized and must be allocated to the user. The most convenient method of initializing the MAP-300 is with a special login account RADC/MAPUP. Upon execution of MAPLP the program pauses to remind the user that the MAP must be allocated. An opportunity to do this is provided through the use of PAUSE and RESUME commands.

MAPLP asks the analyst if program control of the Precision Filter Set is wanted. Under this option two of the Precision Filter Set channels may be controlled by MAPLP. A channel number is requested and the current cutoff frequency and gain state is given. New cutoff frequency and gain values are requested. An input of zero or a carriage return (CR) defaults to the previous value. After both filters are input the routine lists external actions that must be taken and waits for a CR before continuing.

The name of the Speech DB of interest is requested and its header information is displayed if requested. The default device and UIC for this file is SP:[200,200], other devices and UICs can be accessed if given explicitly. MAPLP allows for up to ten individual segments of the file to be output sequentially. The routine asks for the number of segments or "SETs" that the analyst wants.

For each SET the analyst inputs the start block number, number of blocks to output and the number of times to repeat this SET. For each SET the analyst may modify the output sampling rate from that defined for the DB file and may provide a scaling factor to reduce the signal by shifts of from zero to four bits. If control of the Precision Filter Set was originally chosen, then the characteristics of the original channels can be modified for each output SET. The combination of multiple output segments, controllable ampling rate, scaling and dual channel filter characteristics provides the analyst with a varied experimental capability.

After all SETs are defined, MAPLP waits for a CR and then outputs the data as defined. The output can be terminated at any time with a CTRL Z. For single SETs the default for the number of repeats is -1 or continuous and a CTRL Z is required to terminate the output. On termination MAPLP allows the user to repeat the previous group, define new sets for the same file, define a new file and new sets or exit the program.

AP-120B Input/Output Software -

Three programs utilizing the AP-120B for input, output and loopback of speech are available to the analyst. Two of these programs, APIN and APOUT, have been modified to improve their performance. The size of buffers in both the AP-120B and the PDP-11 have been increased from 1 block to 8 blocks. In addition, double buffering has been implemented in both programs in an effort to allow the PDP-11 to keep up with the AP. This modification allows the AP to instruct the appropriate IOP-16 to load or dump one buffer while the PDP-11 is transferring data from or to the other buffer. These programs provide an alternative to their MAP-300 counterparts in the event the MAP is not available to the analyst.

The third program, APLOOP, is essentially a combination of APIN and APOUT. Since the purpose of this program is simply to input and then output data the PDP's only responsibility is to interface with the analyst. The AP controls the input and output of data.

Program APIN - The program APIN inputs speech data through the IOP to the AP-120B, and transfers it to the Speech DB. The real-time data collection of APIN was limited by the AP-120B memory size. All collected data was stored in the AP-120B before transfer to the Speech DB. This was due to limitations in the addressing of the IOP-16 Scroll board from the AP-120B. ARCON solved this problem by using a double buffering method which involves the filling of one buffer while the second buffer is being output to the PDP-11 for storage on the Speech DB. Thus the size of the speech file is only limited by the amount of available disk space.

Program APIN enables the analyst to input data from an A/D converter into a speech data base file. This is accomplished by using the AP's IOP connected to an A/D converter to transfer data from the A/D converter into the AP. The data stored in the AP is retrieved by the PDP-11/44 and written to disk. The amount of data which could be input was limited by the amount of data memory in the AP which is 64K words (256 disk blocks).

APIN was modified such that the limiting factor is now the available amount of contiguous space that can be allocated to a speech data base file. This was accomplished by setting up two buffers in the AP's data memory and setting up the program such that the PDP transferred one buffer from the AP to disk while the IOP was loading the other buffer. These buffers were made as large as possible, their size being limited by the PDP-11/44's task size which cannot exceed 32767 words. These modifications cut down on the number of transfers which were necessary between the PDP and the AP and the PDP and disk and allows the PDP to keep up with the AP. This is necessary if no data dropouts are to occur. Coding of the program is in Fortran and VFC routines which execute on the PDP-11/44 and AP-120B, respectively.

The analyst may invoke APIN simply by entering the command "RUN $APIN". A heading identifying the program is displayed and the AP is assigned to the task. If the AP is not available the program waits. The analyst is then requested to enter the sampling frequency, the number of seconds of data to collect, the channel number and the name of the file which is to receive the input data. The output file is created and opened and the program displays the number of data blocks and the number of speech samples that will be collected. The program waits for the analyst to enter a CTRL Z before it begins collecting data. When the program is finished collecting data the header is displayed and the analyst is prompted for the high and low filter limits, a title and comments to be inserted into the header. The header is displayed and the analyst is asked if any changes are desired. If so, the program branches back to the high filter limit prompt. When the analyst is satisfied with the header the output file is closed and the analyst may input additional speech data if desired.

Program APOUT - The program APOUT outputs selected blocks of data from a Speech DB file to the PDP-11. The data is then transferred to one of two buffers in the AP-120B. When the IOP-16 is ready the AP-120B outputs the data through the IOP-16 to a D/A converter. The selected blocks are output continuously until halted by the user. APOUT also allows the user the option of continuing with data from another speech file. There is no limit to the number of blocks that can be output.

Program APOUT enables the analyst to output repeatedly a selected amount of a speech data base file through one of the AP's IOPs which is connected to a D/A converter. The program suffered from several logic errors which made its operation unpredictable. In the process of outputting data the program would occasionally hang.

After correcting the logic errors, modifications similar to those made to APIN were implemented. Again the limiting factor in the buffer size was the PDP-11/44's task size. These modifications have made APOUT more resistant to delays that may be caused by the PDP's servicing of other users. Coding of the program is in Fortran and VFC routines which execute on the PDP-11/44 and AP-120B, respectively.

The analyst may invoke APOUT simply by entering the command "RUN $APOUT". A heading identifying the program is displayed and the AP is assigned to the task. If the AP is not available the program waits. The analyst is prompted for the name of the input file and the file is opened. If desired the analyst may display the header. The sampling frequency of the file is displayed and the analyst is given the option of changing it. The output channel is then requested and the number of blocks available in the file is displayed. The analyst is requested for the starting block and the number of blocks to output and the output of the data begins. The data will be continuously output until the analyst halts the output with a CTRL Z. The analyst is then given the option of outputting another file.

Program APLOOP - The third program in this group, APLOOP, implements the loopback mode utilizing the AP-120B. This program inputs data to the AP-120B's data memory through the IOP connected to an A/D converter and outputs the data through the AP's second IOP which is connected to a D/A converter. Double buffering has been employed in the program allowing one buffer to be output while the second buffer is input. Coding of the program is in Fortran and VFC routines which execute on the PDP-11/44 and AP-120B, respectively.

While testing this program periodic noise was detected in the output of the data. Extensive testing has been conducted in an effort to locate the source of this noise. Although to date it has not been isolated what has been learned is that the period of the noise is related to the frequency at which the data is sampled, it is not related to the size or location in memory of the buffers. The noise is induced on the signal before it is stored in the AP's data memory and it is not due to data dropouts caused by switching of buffers.

In an effort to eliminate this noise an attempt was made at using interrupts to control the timing of commands to the IOPs. That is, commands to the IOPs instructed the IOPs to issue interrupts when the transfer of data was complete. These interrupts were decoded and handled by an interrupt handler which issued new commands to the appropriate IOP. From the outset problems were encountered with IOP-2. This IOP appeared to issue interrupts constantly even when commanded not to do so. It is unclear whether this problem is related to the noise encountered by APLOOP but further examination could prove useful.

APLOOP inputs speech data from the A/D converter through an IOP-16 to the AP-120B and outputs the data through a second IOP-16 to a D/A converter.

This program suffers from unwanted noise being imposed on the speech signal. The noise is periodic in nature but varies in amplitude. Through testing it has been found that its period is dependent on the sampling frequency. If the sampling frequency is halved the period doubles. Additional tests have been conducted to determine if the noise is related to buffer transition. These tests consisted of increasing the size of the buffer. If the noise were related to buffer transition then a change in the period would be noticed. No such change occurred. A final test was conducted to determine the point at which the noise was being introduced to the signal. The AP-120B instructed the IOP to output a set of buffers loaded with zeroes while IOP-1 loaded the original buffers. Both sets of buffers were captured, written to disk and displayed by ILS. It was determined that the noise is introduced to the signal before it is stored in the AP.

The analyst may invoke APLOOP simply by entering the command "RUN $APLOOP". A heading identifying the program is displayed and the AP is assigned to the task. If the AP is not available the program waits. The analyst is prompted for the sampling frequency and channel. Data is collected and after a short time is output. The analyst may terminate the loop back program by pressing the RETURN key.

5.3.2 Speech DB Retrieval And Sorting

The PDP system utility routines PIP and SRD are often used to search for Speech DB files with a descriptive name or extension. The DB has grown to include such a large variety and number of files that these utilities are no longer capable of performing the location task. A program FINSP has been developed that has exparded retrieval and sorting capabilities. All source code and command files can be found in the virtual disk REPT88. The task is available on DR:[1,54].

FINSP begins by asking the user what device to search on, in what UIC, and the extension of the files to search. It is important to note that the user can use an asterisk in the UIC and/or extension response to indicate he wishes to search all UIC's on that device and/or all files for the specified material. Once the user has entered this information the routine displays it and gives the user a chance to correct it before the program proceeds.

When the information is correct, FINSP creates a file, 'NAME.CMD', which contains a PIP command line generated from the user's responses. FINSP then spawns MCR, executes the command file, and forms another file, 'NAME.DOC', containing a listing of the file names and the directories where they are located. After the spawning process is completed, the user begins another series of questions and answers in order to obtain the actual information the user wishes to locate.

The first question asked concerns how many fields of the speech file header the user wants to search. A maximum number of fields to search has been set to five. The routine then displays a list of the fields and requests the user to enter the field numbers. An error message will appear if the user attempts to enter the same field number more than once. A list of the fields is as follows:

1 = DATA TYPE
2 = SAMPLE TYPE
3 = SOURCE PROCESSOR
4 = SOURCE OF RAW DATA FILE
5 = COMPRESSION ALGORITHM
6 = SYNTHESIZED / COMPRESSED DATA
7 = FILE CREATION DATE
8 = TITLE
9 = COMMENTS
10 = SAMPLING FREQUENCY
11 = INTERNAL FORMAT
12 = WINDOW

The next set of questions that the user confronts involves more details on each of the fields chosen. For example, if field #1, DATA TYPE, was chosen, the user would then be presented with these three options:

0 = Raw data
1 = Processed data
2 = Raw or Processed data

Each field has different options within them and it is suggested that the user pay close attention to them for the best possible results. The comment field has its own special way of locating key words or phrases. The user can enter expressions consisting of the key words/phrases, parentheses, and boolean operators, .AND. or .OR., such as (XXXX .OR. YYYY), where a search for the XXXX substring is carried out first then a second search is made for the YYYY substring. More than one of these structures can be given within a search expression for multiple layers of ordering.

Once all the data the user wishes to find has been selected, the user is asked what type of output he wishes, either a list of file names or the actual header contents, and if to spool this output to the line printer or the terminal screen. If the user chooses to spool to the line printer, a file named 'NAME.LST' is generated and saved at the end of execution of FINSP.

After all these topics are answered, FINSP starts the actual process of opening the first file in the 'NAME.DOC' file, searching its header for the user-requested information, if found, writing the desired output to the user- chosen device, and closing the file. This process continues until all file names in the 'NAME.DOC' file have been exhausted. It should be noted, however, there are certain files that FINSP will not search. Any WD9999.;1 files and files with the extensions .CMD, .FTN, .LST, .OBJ, .TSK, .DIR, and .DOC will be skipped over when the searching takes place.

5.3.3 Interactive Laboratory System

The Interactive Laboratory System (ILS Version 3.0) is a modular software package for computer use in research involving sampled data and signal processing. It has been programmed to operate in an interactive, multi-user mode. The ILS package distributed by Signal Technology, Inc. consists of about 90 main programs and about 250 Fortran subroutines [Ref. 19]. There are seven assembly language subroutines provided for the primitive disk I/O operations.

To make the ILS software more accessible to RADC/EEV researchers, the second 300 Mbyte disk drive is partitioned into individual file directories dedicated to ILS and speech data base files. Upon login to a researcher's ILS account, he is assigned to his particular ILS UIC. The user has the ability to access ILS data files on other devices and UIC's by way of user-specified directory pathnames created with the ILS task TBL. This utility manipulates a table of pathnames in the user's ILS common file. The actual ILS task files reside on the public virtual disk IL:. Not all of the ILS modules can be installed at one time due to the limited "pool space" memory allocated to the operating system for the use of such things as maintaining task control blocks.

ARCON modifications to the ILS package have been reported in previous reports [Ref. 9,10] and have included the following improvements. A menu of ILS programs is available through the command file ILS.CMD that allows access to the ILS routines without memorizing all the routine mnemonics. The limited ILS numeric file name structure was expanded to include meaningful names and extensions. A dynamic program serving monitor DYNSRV was developed to handle the problem of limited installed routines imposed by "pool space".

Because of the modularity of the ILS system, any program module may be modified without affecting the other modules. This feature also permits the replacement or addition of program modules on disk providing they are properly designed to be compatible with the ILS conventions. Within each ILS program another level of modularity is seen in which all programs are composed of subroutine and function calls to standardized segments of code residing in a well designed library. Thus the ILS system is very amenable to custom alterations of signal processing and analysis software.

A large number of custom ILS routines have been developed and implemented by ARCON for the analysis of speech and noise data in support of the various research tasks under the current contract. These routines have included the implementation of several enhancements to standard LPC processing as suggested by G. Kang & S. Everett of the Naval Research Laboratory [Refs. 20 and 21], the line spectral pair analysis method and a set of pole/zero filter routines. A list of custom ILS modules developed for the RADC/EEV system is provided in Table 5.2.

Table 5.2
Custom ILS Programs

DSP - Sample Data File Display routine with a voicing "ONSET" detection option
FFC - Formant Frequency Resonance Cascade generator with a pole/zero plot
GCD - Glottal Closure Detection Function
HCR - File Header Creation
HDS - File Header Display
HMD - File Header Modification
KEA - Kang/Everett LPC Analysis
KES - Kang/Everett LPC Synthesis
LPM - D/A Output Parameter generator
LSN - D/A Output LISTEN
LSP - Line Spectral Pair Calculation and plot
MFS - Modify Formant Structure for LPC Analysis Files
REC - A/D Input RECORD
RMS - Root Mean Square Function
SNR - Speech to Noise Ratio and Equivalent Peak Level (EPL) Measure

All ILS files are located on the virtual disk ILS. This disk is mounted public at system startup and has the pseudo device name IL:. The various file types are segregated into different UICs as follows.

1. [100,300] - Object Code and Object Libraries
2. [100,302] - Source Code (.FTN) Files
3. [100,304] - Overlay Descriptor (.ODL) Files
4. [100,305] - Compile and Taskbuild (.BLD) Files
5. [100,306] - Executable Tasks (.TSK)
6. [100,307] - ILS Help (.HHH) Files

As a sample of the flexibility of the ILS system several of these programs will be discussed.

MAP300/ILS Listen [LSN] -

An ILS utility has been developed which provides A/D services previously available with the MAPLP task. The program is 'LSN' which enables one to 'listen' to sampled data files while working within the ILS environment. Just as with MAPLP, the MAP-300's digital-to-analog capabilities are used for the real-time analog reconstruction of sampled signals and therefore the user is responsible for initializing and allocating the MAP before using LSN. Several parameters can be specified by the user for controlling the segments of the sampled data file to reconstruct, gain, sampling frequency, and the cut-off frequencies of reconstruction filters. These parameters can either be specified in an interactive manner during the execution of LSN, or they may be stored in a separate file using another new ILS program, LPM - Listen Parameter Modify, and extracted by the LSN program.

The alphabetic options for LSN are:

'S' - use the secondary file for data input (default = primary file)
'N' - output the next consecutive sequence of data with the same number of frames as on the previous invocation of LSN
'A' - read signal segment definitions and filter parameters from file pointed to by the 'PRIMARY-B' filename (set with FIL B)
'M' - allow user to interactively define multiple signal segments for sequential output
'F' - allow user to interactively define filter characteristics for 2 channels of the Precision Filter Bank

The numerical command parameters passed to LSN are:

N1 - starting frame (size of frames controlled by CTX task)
N2 - number of frames to output
N3 - number of repetitions of same segment (-1 results in "infinite" repetitions...or until cntl-Z is pressed)
N4 - scale factor where sampled data is multiplied by 2**(-N4)
N5 - sampling frequency used on reconstruction

The default values for the numerical parameters are:

N1 - N1SAV value from common file which normally will be set by previously executed ILS task (e.g., CUR)
N2 - N2SAV value from common file
N3 - infinite repetitions (-1)
N4 - scale factor of 1.0 (i.e., N4=0)
N5 - sampling frequency specified in header of sampled data file

The LSN task makes use of object modules from the ILSLB, SNPLB, and QIOLB libraries. Note that a special version of the ILSLB module FILIO is used for purposes of controlling the allocation of LUNS used by the ILS modules. This new module circumvents problems encountered when attempting to use SNAPLB modules and ILSLB modules which attempt to use the same LUNS for different purposes.

Disk I/O for the sampled data is done at the QIO level (i.e., block I/O) as previously used with MAPLP. However, since the segments to be output are now specified in terms of a variable frame size, a deblocking scheme is required. An efficient, low-overhead scheme was devised such that the code required for setting up I/O parameters from one disk read to the next is minimized and consists of modulo 256 additions and comparisons. The MAP-300 code itself is just as found in MAPLP although the data buffers are no longer constrained to be of lengths which are integer multiples of 256 samples.

The LPM ("listen parameter modify") utility allows the user to create/modify a parameter file to be used with the LSN A option. The parameters to be specified for each output segment are:

1. starting frame
2. number of frames
3. number of repetitions
4. sampling frequency
5. scale factor
6. filter control parameters for two channels

The parameter file is in the form of an unformatted, direct-access file with the PRIMARY-B filename stored within the common file serving as a pointer to the file. The modification of an existing parameter file pointed to by the PRIMARY-B filename can be carried out by invoking LPM with the 'O' option ("old") while the default is to create a new parameter file. A limit of 10 signal segment definitions per file is in effect.

MAP300/ILS Record [REC] -

The A/D conversion program MAPIN was revised and is now available in the form of an ILS task and is referred to as REC ("record"). This task uses the ADAM module on the MAP-300 system for performing real-time A/D conversions of sampled data files resident on disk. Like all ILS tasks, data files to be accessed by REC are "pointed to" using the FIL task. Alphabetic and numerical arguments are supplied by the user either within the MCR command line invoking REC or following a prompt at run time. The possible alphabetic arguments are:

L - Operate in loopback mode in which the digitized signal is fed immediately back to the MAP's AOM module for D/A conversion. No data files are created in loopback mode. If no "L" is provided, the default mode of operation is one in which the signal samples are written to a disk file.
S - Write signal sample data to the file currently specified as the secondary file within the ILS common file. The default is to write to the primary file.

The possible numerical arguments are:

N1 - Sampling frequency in Hertz
N2 - Sampling duration in seconds
N3 - A/D channel on ADAM (0 or 1); default is 0

The operation of the MAP itself during the data collection is identical to that of MAPIN. One added feature, however, is the monitoring for A/D saturation during the course of sampling a signal. The saturation detection is implemented in hardware on the ADAM board. At the completion of the sampling phase, the ADAM register #4 is read which provides an OVF bit indicating if either the most positive or most negative quantization value had been generated by the A/D converter. If the OVF bit is set, a warning message is provided to the user indicating that A/D saturation had occurred at least for one sample during the whole process. It should be noted here that an ILS task, STE, is available which will scan a data file and will list frames in which samples are found having values greater than 2048 or less than -2048. However, samples generated from the ADAM are left justified (use of upper 12 bits or 16 bit words) and will exceed the range checked by STE even if no saturation has taken place.

The REC task is overlayed and makes use of modules from the SNPLB, QIOLB, and ILSLB libraries. Note that a slightly changed version of the ILS module with file primitives, FILIO, is used for this task build. The new module is MYFILIO with changes made in regards to the allocation of LUNS for ILS file I/O which conflicted with those used for I/O with the MAP.

Speech/Noise Ratio And EPL Measurement Task [SNR] -

An extension to the ILS software involves a task which provides the ability to measure the Equivalent Peak Level of speech segments as well as the Speech-to-Noise ratio [Refs. 9,22,23]. The algorithm to provide these measurements already existed in the form of a PDP-11 assembly language program requiring an analog input. ARCON revised and implemented an ILS version of this program, called SNR, which provides EPL and SNR measurements on a user-specified number of frames of a sample data file stored on the Speech DB.

The speech data file to be analyzed is first specified using the FIL task. Either a primary or secondary file can be analyzed. The three numerical arguments for the SNR command are starting frame, number of frames, and the LUN for displaying results (defaults to KBU). The S/N ratio and EPL estimates generated are displayed on the specified device and they are written into the file's header in the fields allocated for "Signal-to-Noise ratio" and for "Current Max. Amplitude". The EPL is recorded as an integer and is in dBm units. The S/N ratio is stored as a floating point value and is in dB units.

There are a few notes regarding the internal operations of the task:

1. The S/N ratio measurement algorithm is based upon 20 millisecond frames of data. The frames specified by the user for analysis are in terms of the samples/frame set with the CTX task. Internally, these frame parameters are converted to 20 millisecond frame equivalences.

2. A minimum of 3 seconds of input time data is required to even get results from the program and it is strongly recommended that at least 15 seconds of data be used for analysis. Previous work with the algorithm has been based upon 40 second data samples.

3. In response to the inconsistency of the Speech Data Base standard regarding the storage of raw time data in an integer format (left justified vs. right justified 12-bit values), an automatic scaling mechanism is provided when data representing values greater than could be found in a right justified 12-bit variable is found. The scaling consists of dividing all data by 16.

4. Automatic scaling of frame energy values is also included for "centering" within the histogram the hump associated with the distribution of noise frames. This technique was previously developed on the assembly language implementation.

5. Double precision floating point values are used in most of the calculations.

The task was run with input from some of the DRT and DAM files available on the SP: disk as well as a test file created with one of the DRT tapes with helicopter background noise. The S/N and EPL results were within the range expected based upon previous experience with this test material.

Line Spectral Frequencies [LSP] -

An enhancement to the ILS package was the addition of a Line Spectral Pair [LSP] analysis and plotting capability. Line Spectral Pairs [Ref. 24], or Line Spectral Frequencies, provide an alternate parameterization of linear predictive analysis: instead of describing an all-pole filter by giving its prediction coefficients or its reflection coefficients, the all-pole filter is described by giving the poles of two related filters, whose poles are guaranteed to be on the unit circle. These poles, interpreted as frequencies, are called line spectral frequencies, or, since they occur in pairs, line spectral pairs. ARCON has developed software that derives the line spectral frequencies of a linear predictive model and plots these frequencies as a function of time. Figure 5.2 is an example of such a plot, showing a tenth-order analysis for the sentence "Thieves who rob friends deserve jail;" the two curves at the bottom show pitch and energy, for reference purposes.

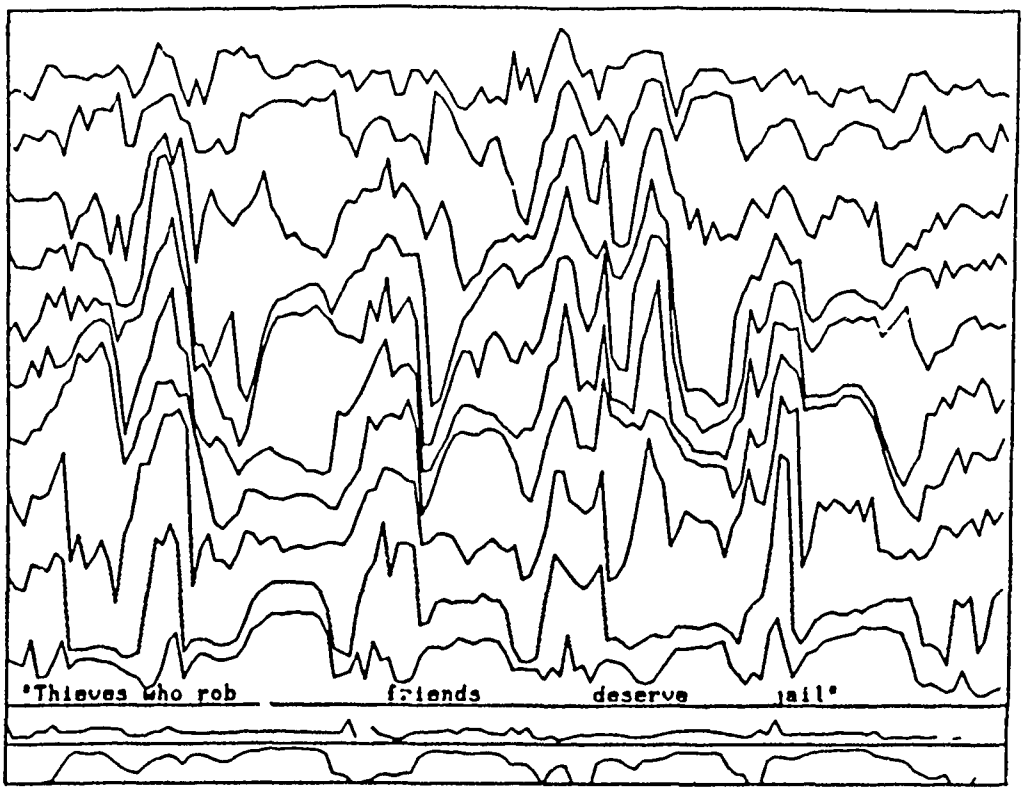


Figure 5.2
Line Spectral Pair Display

Formant Frequency Resonance Cascade [FFC] -

Another addition to the ILS package was the development of a program, called FFC, for the design of a cascade of second order resonance filters to represent formant structures as defined by their order, center frequency, and bandwidth. The routine requests the following input data from the user: the number of formant resonators; the sampling frequency; and the pitch frequency; as well as the center frequency and bandwidth for each of the formant resonators. Once this routine is executed, an output sample data file is generated, the power spectrum of this data is plotted, and the filter parameters are output to the line printer file. This routine has been used for the generation of synthetic vowels. Figure 5.3 illustrates a sample run of the FFC routine.

FORMANT RESONANCE CASCADE

NUMBER OF CASCADED RESONATORS • 5
SAMPLE FREQUENCY • 8000
PITCH PERIOD (8 SAMPLES) • 66
FRAME LENGTH (8 SAMPLES) • 256

THETA	R	A	B	C
0.392699	0.952898	0.147243	1.759248	0.906490
0.981748	0.952898	0.848576	1.057914	0.906490
1.767146	0.946586	2.265182	-0.369388	0.895874
2.387618	0.924465	3.282448	-1.347812	0.854636
0.000000	0.000000	1.000000	-0.450000	-0.475000

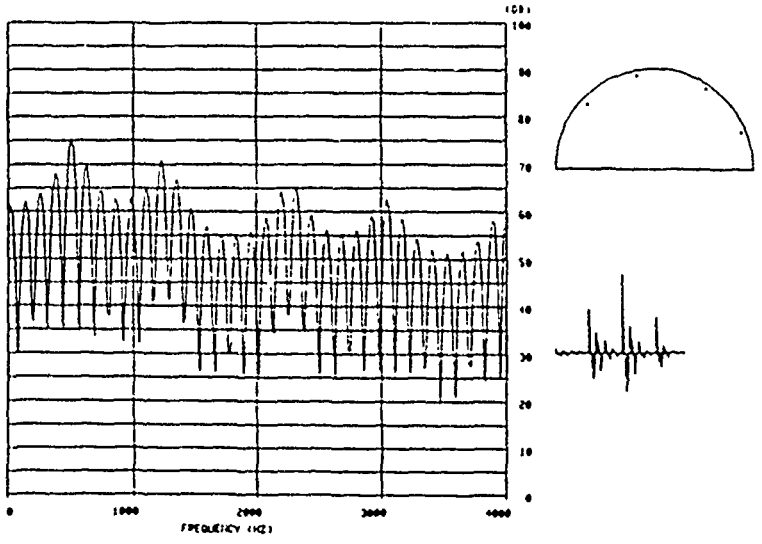


Figure 5.3
Formant Filter Cascade

Modified Formant Frequency Structure [MFS] -

The custom ILS routine MFS was developed to provide the capability of modifying the formant structure of LPC analyzed speech. The routine operates on LPC analysis files that have been generated from sampled data files by any of the ILS LPC programs (i.e. ANA; API; KEA). The analysis file must be assigned as the ILS secondary file and is modified in place by MFS.

The input parameters expected by MFS are the initial frame number and the number of frames to be considered. A "marked" option can be called with an alphabetic parameter M. This option searches the specified range for analysis frames that have been marked or flaged by the routine SGM for special attention.

Each frame or every marked frame, depending on option M, is reported on by listing any measured formants and by giving the formants derived from the roots of the fourier transform of the reflection coefficients. The analyst is then given the opportunity to modify any of these roots by inputting a replacement frequency and bandwidth pair.

After modification of any of the frame's formants, MFS calculates the respective reflection coefficients and replaces the original RC's in the frame with them. In this manner an analyst can shift formant locations and change their bandwidths within an utterance.

The procedure to generate a synthesized output file for the modified utterance requires the use of the ILS Filter routine FLT and the original LPC residual. The residual of the LPC analysis is most easily generated within ILS by the RAN routine. This residual signal is used as the excitation signal for an analysis filter as defined by the MFS modified analysis file.

The MFS program has been used in conjunction with the ANA, RAN, FLT SGM and FFC routines to artificially change the formant structure of vowels within carrier words and phrases to study the least noticeable difference effect or "difference lima" of vowel formant frequencies and bandwidths.

Glottal Closure Detector [GCD] -

The custom ILS routine GCD was developed to provide a glottal closure detector algorithm. GCD requires the user to assign a primary input file and a secondary output file prior to execution. The context of these files must be less than 257.

The user invokes GCD using the following syntax:

GCD N1,N2

where

N1 = frame number to begin on, and
N2 = number of frames to process

N1 and N2 parameters both default to values used in previous routines executed from ILS. If the proper criteria has been met, the routine proceeds to set certain variables with values, such as window sizes, for use later in the algorithm. A list of these variables and their formulas are as follows:

```
MWIND =  0.020  *   FS
LNWND =  0.0030 *   FS
  C   =  FS     /   128
 M1   =  300    /   C
 M2   =  1000   /   C
```

where FS is the sampling frequency of the speech file.

With these variables set, a loop is then initiated which performs such functions as reading frames of data from the speech file, if necessary, saving this frame of data so that another frame can be read in, and writing frames of results to another speech file. Two other subroutines are called from within this loop which accomplish a majority of the computations needed for the GCD algorithm. The first

subroutine, SUMSQ, takes a window, MWIND, of the samples read from the speech file, sums up the squares of these samples, and saves this value in SSQ. The algorithm for this subroutine is as follows:

$$SSQ = \sum_{k=t}^{t+MWIND} s^2(k)$$

Note: t denotes the present position on the signal for the which the computations are being done

The second subroutine, SDFT, takes a smaller window, LNWND, of the same samples and performs a Discrete Fourier Transform of 128 points. It must be noted that if the window size LNWND is less than 128, the remaining points are zeroed so as to look like the subroutine is receiving 128. SDFT returns 64 magnitude-squared samples which are then summed up between M1 and M2 and assigned the variable SSSQ. The algorithms used for those calculations are:

$$D_t(m) = DFT (s')$$

where $s_i' = s_t, s_{t+1}, \ldots, s_{t+n}, \ldots 0,0$ and $i = t, t+127$

$n = LNWIND-1$

and

$$SSSQ = \sum_{k=M1}^{M2} |D_t(k)|^2$$

The final computation involves dividing SSQ by SSSQ, scaling the value by 32767, and storing this final integer as F(t) where it will be later written to the secondary file at the proper time. The formula is as follows:

$$F (t) = SSQ*32767/SSSQ$$

The calculations above are continued until the routine has a reached the final sample point in the number the user requested to process.

5.3.4 Error Metric DB Display

Two methods are available for plotting the magnitudes of the elements of an error metric matrix. In the "waterfall" method each row of the matrix is plotted across the screen with successive rows offset vertically from each other. This kind of a display is available as an option of the ILS DSP command: in effect, a 64 X 64 matrix is treated

as 64 successive 64-sample frames for display purposes. The program DEMDSP (described below) converts a .ERM file to an integer data file that DSP can plot. In reading a plot of this kind (Figure 5.4), it is easy to see where the peaks are within each row, but it is harder to compare the relative heights of different peaks in one row, and virtually impossible to compare the heights of peaks in different rows.

Often it is difficult to interpret an error metric matrix displayed in the "waterfall" form. ARCON has developed software to provide an alternate display format. This program, called ERM3D, uses a raster graphic method to display an error metric matrix, as shown in Figure 5.5. Lighter or darker shades correspond to matrix entries with larger or smaller magnitudes, on a scale with 100 different levels. This gray scale is automatically adjusted to the range of magnitudes present in the matrix. Another option of ERM3D allows display of the magnitudes of the metric's eigenvector matrix.

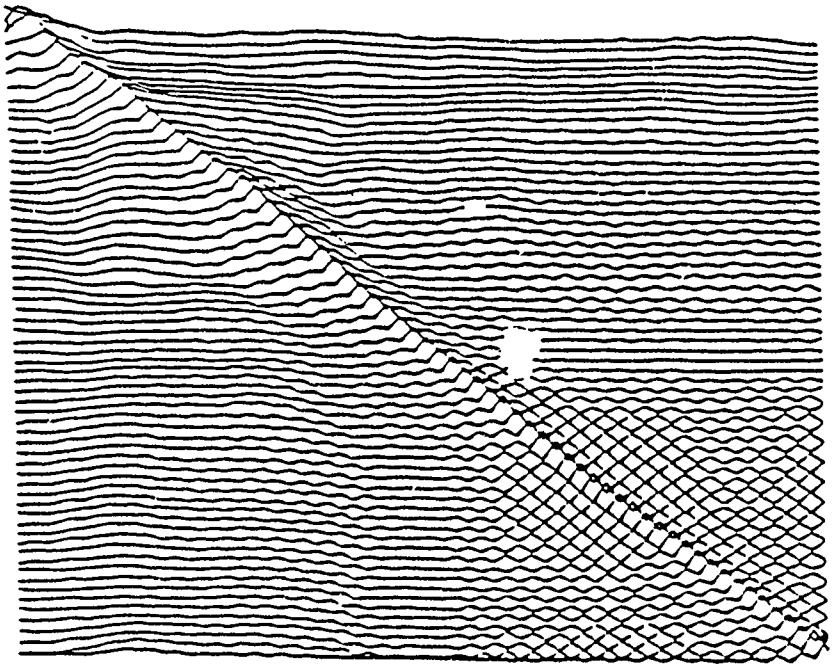


Figure 5.4
3-D EM Vector Display

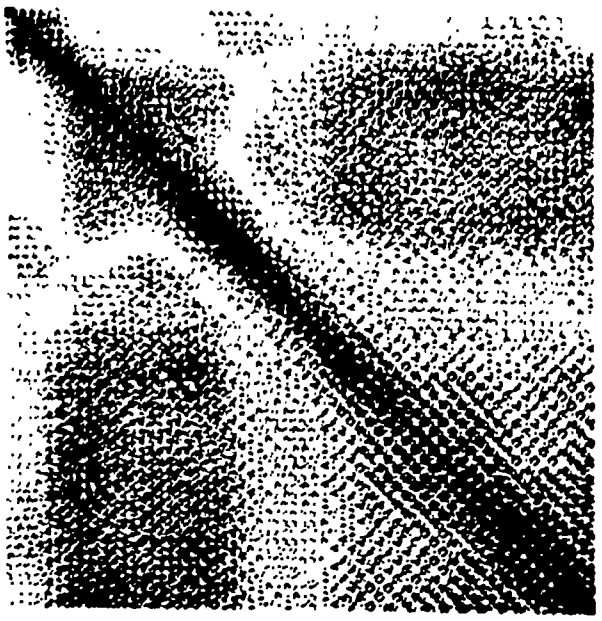


Figure 5.5
3-D EM Grayscale Display

5.3.5 Speech DB Utility Routines

ARCON introduced three separate routines to the ILS package for the generation and maintenance of speech data base header information. The first of these ARCON developed routines is HCR which creates and initializes a speech DB header block for any speech DB file. The second, HDS, displays any speech data base header block the user requests. The last routine, HMD, allows the modification of a speech file header which can be useful if information was incorrectly inserted when created. These routines are of extreme importance because it is through accurate Speech DB header information that all traceback and location routines operate. The originator of DB files holds the responsibility for accurate header information. These routines provide for convenience in fulfilling this responsibility.

Since the Error Metric files in the Speech DB are floating point files of specific form and content they can be converted to fixed point for various analysis purposes. The usual need is for a graphic representation of the metric structure. This can be provided by ILS as discussed above. The DEMDSP routine will open an Error Metric File and pace through its structure asking the analyst which matrices or vectors should be converted and in what form. The resulting file is of ILS form and can be plotted in a number of ways. The most convenient method is with the DSP function using multiple plots. Figure 5.4 demonstrates the results.

5.4 PC DB TOOLS

At the RADC/EEV Speech Processing Facility there exist two other data base utility packages, PCFILE 3 and DBASE III. They are used for the maintenance of the PC Data Bases described earlier in this section and are implemented on Zenith PC clones. PCFILE 3 utilizes the Technical Information DB to provide the ability to add, modify, delete, find, sort, or list any of the technical reports available at the Speech Lab. DBASE III works with the Archive DB and has many of the same qualities as PCFILE. However, DBASE III has one added feature which makes it more useful for the purposes of the Speech Lab. It allows structure modification of the data base even after the data base has been added to. In PCFILE once the user has set up the fields and enters the first record of the data base, the structure can no longer be changed except for creation of a new DB.

CHAPTER 6

COMMUNICABILITY SYSTEM

6.1 RADC COMMUNICABILITY FACILITY

The Communicability Facility at RADC/EEV (Figure 6.1) has been modified extensively and now has full-duplex capability as well as the original half-duplex system. Users communicate in complete freedom in full-duplex, or on a push-to-talk basis in the half duplex mode. Both test stations reside in isolation rooms. Control boxes at both locations are switchable, half to full duplex.

A Carter Engineering headset model CEH157TR with an Electro-Voice M-101/AIC microphone is the current user interface to the system; however, the design of the stations lends itself to changes in headset/microphone arrangements with little difficulty. These headsets plug into in-house designed boxes that amplify the outgoing signal, match impedances, transmit switch information (for half-duplex mode), and provide sidetone (in full-duplex mode).

All audio lines are shielded, balanced cable. In full duplex mode the signal from studio A follows the same path as half duplex, to the control box in the DRT control room. At the box, a printed circuit card labeled "Full Duplex" replaces its counterpart labeled "Half Duplex". This card allows studio A's signal to proceed directly to the computer room, bypassing the relay and amplifier system used in the half duplex mode.

Transformers at the patch panel in the computer room step this signal down to unbalanced and from here a host of signal processing devices may be accessed. Before leaving the computer room, the processed signal is transformed to a balanced state. In full duplex, the signals get routed directly to the opposite station from their origin. Transformers residing at both listening positions step the signal down to unbalanced, low impedance to properly match the headsets.

The test administrator can monitor and record the test proceedings during full duplex operation in either a processed or unprocessed state at the audio equipment racks in the computer room. A microphone is available to communicate in unprocessed speech to each isolation room. The administrator has control over signal levels both pre- and post-processor.

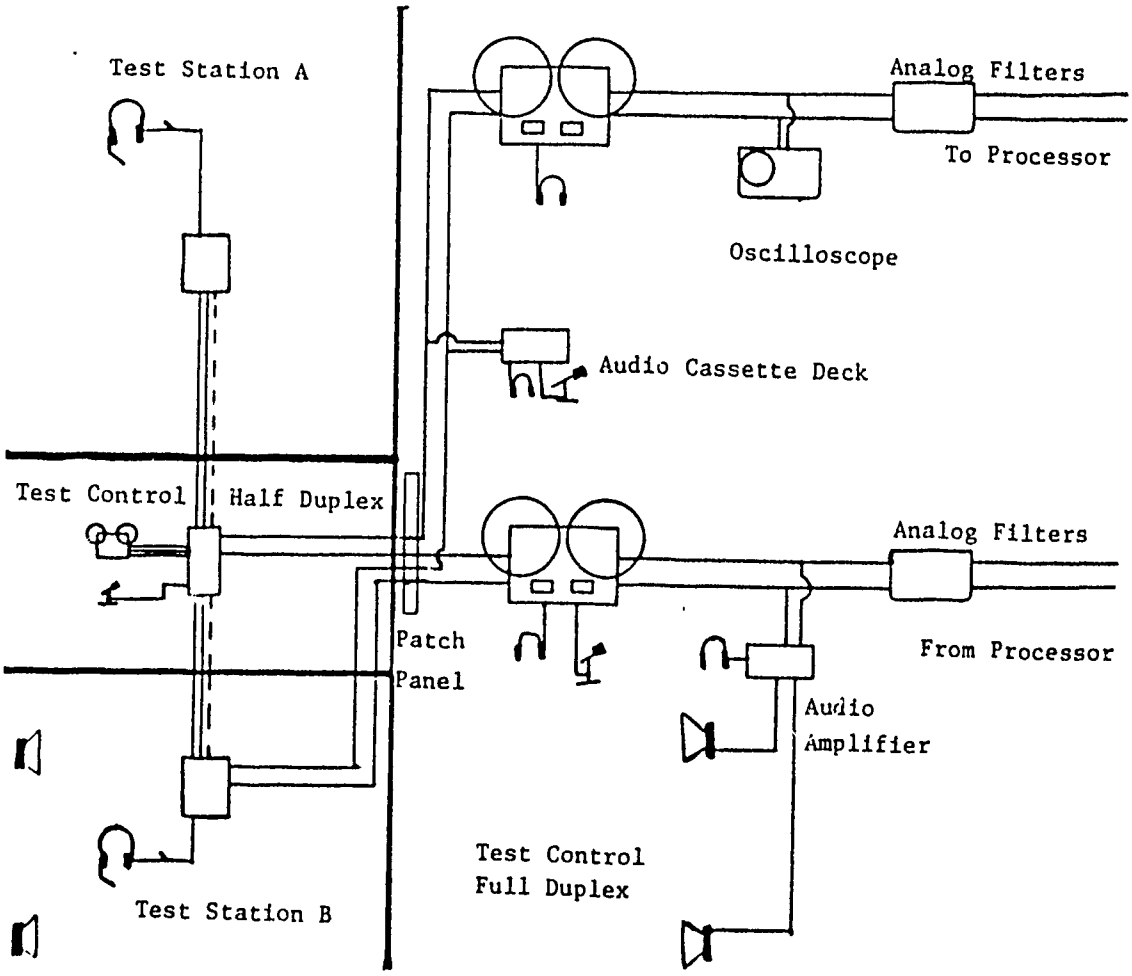


Figure 6.1
RADC/EEV Communicability Facility

This facility can be mated with any number of vocoder systems available at the Speech Laboratory. Acoustic background noise can be provided in one of the test rooms for the simulation of various acoustic environments. A full range of transmission channel effects can modify the transmitted digital bit stream for the simulation of an actual transmission channel. The facility hardware has been tested and verified by ARCON using the DRT.

6.1.1 PDP-11/34 - MAP-300 Processor

Current use of the Communicability Facility most often includes interfacing with a host PDP-11/34 (see chapter 5) and two MAP-300 array processors. These processors are available to run a number of different narrowband digital voice algorithms in real time, full or half duplex. These algorithms include:

1. LPC at 2.4Kbps implemented by BBN
2. APCSQ at 9.6Kbps implemented by NSA and BBN
3. CVSDM, a MITRE implementation of CVSD with variable rate selection
4. APC-NS at 9.6Kbps developed by DCEC with Noise Shaping
5. CVSDND, a University of Notre Dame implementation of CVSD
6. ATC, a 9.6Kbps algorithm developed by GTE
7. PARC, a 9.6Kbps algorithm
8. RELP at 9.6Kbps developed by BBN using residuals of LPC analysis

Most algorithms make available variable bit rates and random error insertion within the software. All algorithms implemented on the MAP300s can be interfaced to the outboard error generators and channel delay simulators.

The MAP-300s are equipped with Speech Processor Interfaces (SPIs) that give access to the analog and digital I/O of the MAPs. ARCON has incorporated transformers to enable the use of balanced cables to reduce line noise. Line input/output jacks and a handset are available. ARCON has made a number of modifications to the SPIs in order to facilitate research efforts at the Speech Lab. The filters, once a permanent feature, now may be switched out of the signal path allowing the use of the more sophisticated outboard filters. The SPIs have also been modified to operate in analog loopback.

6.1.2 Other Processors

The patch panel in the computer room allows the communicability system access to a number of other devices. Any processors with standard audio connectors and line impedances can easily be interfaced. Non-standard processors will require the design of analog and digital interfaces. Algorithms available to the communicability testbed through the patch panel include:

1. Quintrell LPC's & PLPC
2. RADC/EEV's FPS AP-120B algorithms
3. RADC/EEV's MAP-300 algorithms
4. MIT LL 2400 bps LPC (Adams-Russell)
5. MIT LL 1200 bps LPC
6. BBN APC-NS at 9.6 kbps
7. NSA Multiple Rate Processor CVSD, APC, LPC

6.1.3 Operational Environment Simulations

In one of the isolation rooms used for the Communicability Facility acoustic noise backgrounds may be recreated with an audio system consisting of 6 Acoustic Research AR9 tower speakers, two Phase Linear model 700 stereo preamplifiers, 3 Phase Linear model 3500 stereo amplifiers, and 2 Soundcraftsmen stereo 24-band graphic equalizers. Pre-recorded audio tapes (analog or digital) may be played back through this system, or an available white noise generator may be used with one of the graphic equalizers to produce a frequency response curve that approximates background acoustic noise for a variety of environments. Informal measurements suggest the RADC/EEV sound system installed in this isolation room is capable of producing acoustic noise environments approaching an SPL of 110db.

Audio channel problems on either side of a communications link can be simulated at the communicability testbed through the use of interchangeable microphones/headsets, audio noise generators and a wide selection of fixed and programmable filters.

Degraded channel conditions can be simulated with the insertion of error generators and line delays in the digital bit stream. Error generators can be set to produce a given percentage of random bit errors or may be configured to generate block errors that are read off of computer magtape. ARCON has developed software (BER) for use on the CSP-30 high speed signal processor that translates channel models from computer magtape to TTL line level signals that are in turn fed to error generators and output as burst errors. Simulated communication satellite delay can be created on two independent channels with the use of the MAP-300 currently hosted by the PDP-11/44. By sending the digital bit streams and a clock line of one or two processors, the ARCON developed program will give any amount of delay requested by inputting the number of samples of delay desired. Access to the bit streams and clock signals have been facilitated by ARCON designed and implemented patch panels located on the back of all three MAP-300s in order to achieve maximum flexibility in transmission channel simulations.

Bit Error Rate Program [BER] -

A version of BER can be run on the CSP-30 system. This is function 225 on the CSP-30's "disk" and is loaded with the DOS by reference to this number. The procedure for running the program is:

1. Have the CSP-30 file server task running on the PDP-11/44 by logging into the account RADC/CSP which will automatically log itself off after installing the file server.

2. Power-up the CSP-30 with the key switch and also power-up the CRT.

3. Reset the CSP-30 by depressing the Master Reset and I/O Reset Toggle switches on the console. Load the starting address of DOS which should be retained during power-down within the CSP-30's Magnetic core memory. The starting address of DOS is 100000 and is usually retained on the thumbwheel switches. Press the SET ADDRESS toggle. Press RUN and the DOS program should execute.

4. Enter the function code 225 on the CSP-30 terminal.

5. The BER program is loaded and executed as indicated by the prompt to select the desired bit rate.

6. Load the burst error tape on the tape drive nearest the console. Make sure that the tape is at load point and that it is on line.

7. Select the bit rate and indicate the number of files to skip on the tape before beginning output.

8. TTL level error bits will be available on the patch panel in the area labeled CSP-30 and at the BNC connector labeled B-OUT.

The burst error program, BER, is also available as a disk resident, bootable task image on the PDP-11/34 machine in the Building 1124 Speech Lab. The task can be loaded into the machine and executed simply by logging into the RSX-11M system as RADC/BER. The login command file for this account will prompt the user to mount a burst error tape onto the MT0: magtape device and then a booting of the task image will be performed. The RSX-11M operating system code residing in lower memory will be destroyed in this process although the operating system can be easily rebooted from the system disk when needed. The BER program becomes a stand-alone task with complete control of all devices. A prompt at the console will request the user to either advance the magtape to the next burst error file, to back up one file, or to begin reading the burst error information on the file currently positioned at the head of the magtape drive. A line displaying the number of microseconds between each sample output on the D/A will then be shown on the screen. The burst error stream is at that point available from the D/A converter. To stop the program, a control-halt sequence is entered at the PDP's switch console. A control-boot at this point will reboot the RSX-11 operating system. The BER program can be restarted by loading in the start address 1000 (octal) at the console and pressing the start switch. The BER program does not have to be halted before taking the magtape drive off-line and either rewinding the current tape or switching to another burst error tape.

The task is now configured to output samples at a 2400 bits/second rate. This bit rate can be changed by writing in one word at the location 1164 (octal). This word must be the value equal to minus the number of microseconds between each sample. For example, there are 416 microseconds between each sample when operating at 2400 bits/second. Thus, location 1164 has a value of 177140 (octal) which represents -416. After changing this value, the program can be restarted from the console at the start address.

A few changes were made to the original BER source in order to make a bootable version for the PDP-11/34:

1. The first instruction executed is RESET which causes a UNIBUS INIT to be generated which will initialize all device controllers on the bus.

2. Set the programmable clock's interrupt vector to 344 in place of 444 as it was on the old PDP-11/20 system.

3. Set the buffersize for records read from the magtape as 4000(octal) since the newer burst error tapes are recorded with 2048 bytes/record. Some of the older burst error tapes have record sizes other than 2048 bytes. These can be used with the program after changing the BUFSIZ value at location 2440 (octal) to the desired number of bytes per record (currently set to 4000(octal)).

The BER task image was placed on DL:[1,54]. It is booted using the privileged MCR command BOO with the command:

```
BOO DL:[1,54]BER.TSK
```

Note, as soon as this command is executed, the RSX-11 operating system will be overlaid by the BER task.

Copies of the BER source file and task build command file can be found on the Building 1124 PDP 11/34's DL: disk at UIC [200,210] and on the SOFTWARE DEVELOPMENT disk for that system. Copies of the files related to BER can also be found on the PDP-11/44 system on the REPT88 virtual disk.

MAP-300 Modem Delay Program [DELAY] -

Code was written for the I/O Scroll (IOS-2) module of the MAP-300 which performs a delay function on two independent data channels operating at a fixed baud rate of 2400 bps. The "modem" interface circuitry built onto the IOS-2 is used for I/O. The normal receive data and transmit data lines are used for one channel with the second channel created by using two lines normally used for other purposes in normal RS-422 or RS-232 protocol. No bit manipulation is required within the MAP. The incoming half-word (16 bits) encoding the input data is stored into a circular buffer formed on BUS-2 memory. The output data is addressed by a second pointer offset from the input

pointer by the number of samples of delay desired. The transmit timing is controlled by a programmable timing circuit on the IOS-2 which is programmed for the 2400 bps transfer rate. The sampling of incoming data is determined by the external clock signal.

The IOS-2 code implementing the delay is on the MAP300 virtual disk at [200,310]DELAY.MSO. Its binary image must be loaded into the MAP with the MLD loader. However a command file has been created on DR:[1,54]DELAY.CMD which will initialize the MAP and start the DELAY program. The DELAY.FTN program configures the MAP memory, loads the delay integer value (number of sample delays), requests the CSPU to load and bind the DELAY program into the IOS-2's program memory, and loops through this cycle each time the user wants to change the delay value. It should be noted that there is no simple way to have the IOS-2 stop executing once it is started. A complete MAP-STOP, MAP-START sequence is executed each time the user changes delay values. Thus a new loading and binding of the IOS-2 program memory is performed. The time to make this change is well under a second.

6.1.4 System Verification

The communicability testbed has the capability to input and output audio signals at many locations throughout the system. This allows for troubleshooting and verification of the intelligibility of part or all of the system using the DRT. For example, with the use of a switch that effectively locks the signal flow so that station B is always sending and station A is always receiving, an input deck is placed at station B. Pre-recorded test material leaves station B and travels through the communicability system, including any chosen processor, after which it is accessed for recording at the output to station A on the relay box. This allows DRT tapes to be generated that include the entire communicability system's characteristics except for the microphones. Tape decks can also be configured to record conversations taking place in either full or half duplex modes, processed and unprocessed speech.

The plot profile (Figure 6.2) shows high quality scores for the communicability system in loopback as well as interfaced with three standard narrowband digital voice processing algorithms as implemented by the MAP 300s. A comparison of the station A-B signal path and the B-A path are statistically identical. A slightly higher score for the inclusion of the analog loopback of the MAP 300 Speech Processor Interface (the third system on the plot) is a reflection of bandpass filtering taking place at this point in the signal path. Finally, the scores for the three algorithms, LPC-10 at 2.4Kbps, APCSQ at 9.6Kbps, and CVSD at 16Kbps are statistically identical to scores for the processors in a stand-alone configuration. The DRT confirms the high intelligibility of the communicability system.

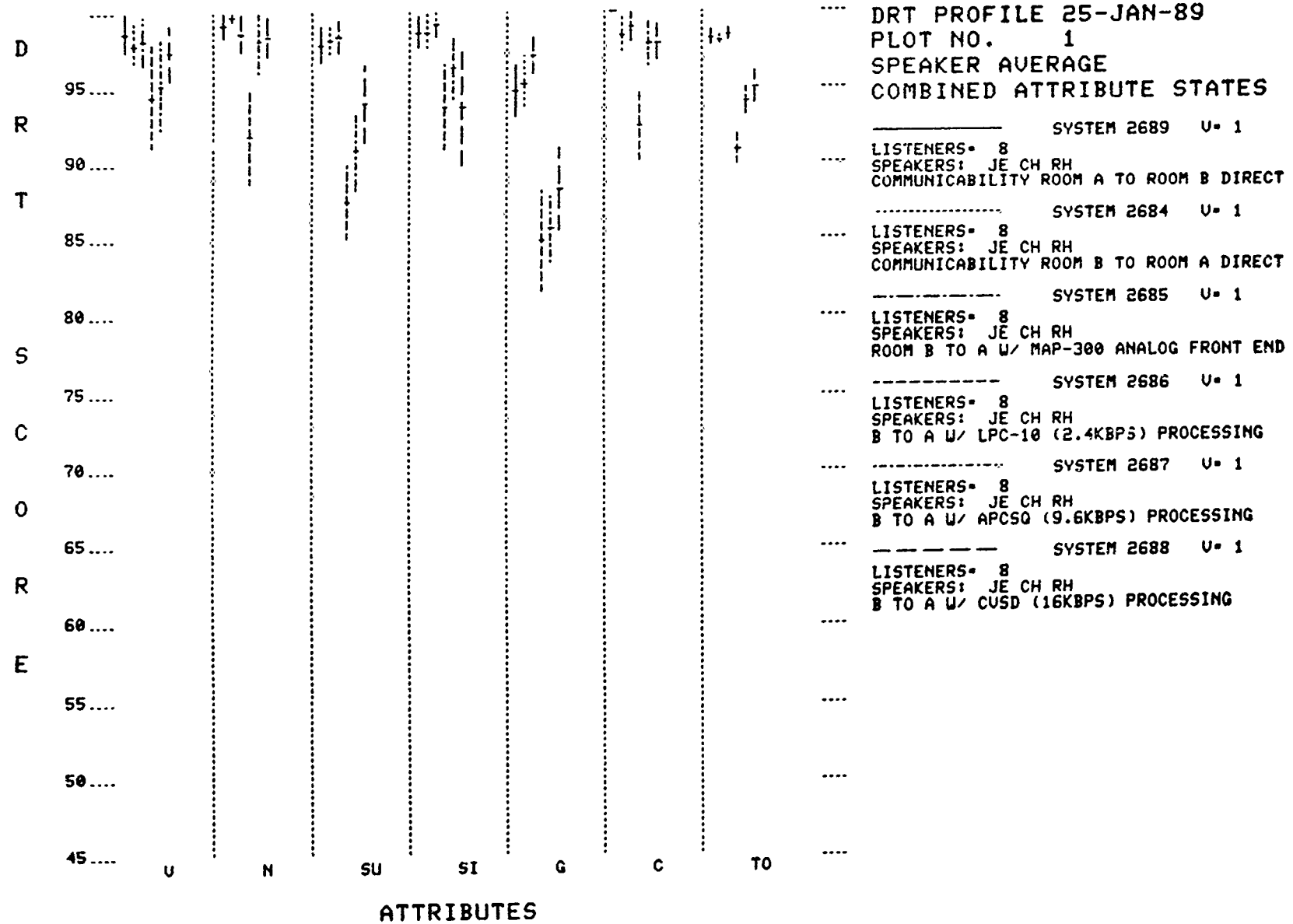


Figure 6.2
Communicability DRT Profiles

REFERENCES

1. S. Woolf, et al., "Interactive Mathematical Modeling", Chapter IX, Speech Processing; RADC-TR-77-86 February 1977; ARCON Corporation, Final Technical Report 1 October 1973 to 1 September 1976.

2. W.H. Rosenfeld and C.B. Hayes, "Speech Processing Software and Computer Systems Development", R77-01W ARCON Corporation, 4 Feb 1977; Final Report 1 September 1975 to 30 September 1976.

3. W. Rosenfeld, "Narrowband Digital Voice Processing (LDVT PLPC/TRIVOC Software Documentation)", ESD TR-76-282, Vol. IV, March 1978.

4. J.D. Tardelli and C.B. Hayes, "Narrowband Digital Voice Processing Software and Computer System Development; Quintrell and PDP-11 Programs and Systems", R78-01W ARCON Corporation, 24 April 1978; Final Report 1 October 1976 to 31 December 1977.

5. J.D. Tardelli and C.B. Hayes, "Mathematical Modeling for Narrowband Digital Voice Processing - A Quintrell Implementation of AFSC LPC-10 and Voice Processing Facility Systems", R80-01W ARCON Corporation, 30 Sept 1980;and Final Report 1 Jan 1978 to 30 April 1980.

6. C.M. Walter and J.D. Tardelli, "Array Processor Oriented Signal Compression Techniques for Optimizing the Performance of Narrowband Digital Transmission Systems", R81-01W ARCON Corporation, 30 Oct 1981;Final Report 1 May 1980 to 30 Sept 1981; and RADC-TR-85-41, March 1985.

7. E. Wachsler and J.D. Tardelli, "Speech Processing Facility System and DRT Software", R82-01W ARCON Corporation, 20 April 1982; Final Report 1 May 1980 to 31 December 1981.

8. J.D. Tardelli and W.F. Bellew, "Diagnostic Rhyme Test Software System", R83-01W ARCON Corporation, 31 January 1983; Interim Report 1 January 1982 to 31 December 1982 and RADC-TR-85-240, December 1985.

9. J.D. Tardelli, C.M. Walter, J.W. Kennedy and J.T. Sims, "Research and Development for Digital Voice Processing", R84-01W ARCON Corporation, 31 July 1984; Final Report 1 January 1982 to 8 May 1984; and RADC-TR-85-46, March 1985.

10. J.D. Tardelli, C.M. Walter, J.T. Sims, P.A. LaFollette and P.D. Gatewood, "Research and Development for Digital Voice Processing (DVP)", R86-01W ARCON Corporation, 30 May 1986; Final Report 1 January 1984 to 14 February 1986; and RADC-TR-86-171, October 1986.

11. J.D. Tardelli, J. LeBlanc and P.D. Gatewood, "RADC/EEV Diagnostic Rhyme Test System Improvements", TR-88-01 ARCON Corporation, 24 June 1988; Interim Report 16 February 1986 to 30 June 1988; and RADC-TR-89-256, November 1989.

12. C.M. Walter and J.D. Tardelli, "Signal Compression Model Interrelationships in the Time, Frequency, Principal Component and Canonical Coordinate Domains", Proc. of 1982 IEEE ICASSP, IEEE Press, New York, 1982, pp. 1367-1370.

13. J.D. Tardelli and C.M. Walter, "A Speech Waveform Analysis and Reconstruction Process Based on Non-Euclidean Error Minimization and Matrix Array Processing Techniques", Proc. of 1986 IEEE ICASSP, IEEE Press, New York, 1986, pp. 1237-1240.

14. P.A. LaFollette, J.T. Sims, and J.D. Tardelli, "A Parallel Implementation of Canonical Coordinate Speech Compression", Proc. of 1987 IEEE ICASSP, IEEE Press, New York, 1987, pp. 2193-2196.

15. Magnus, W., Oberhettinger, F., and Soni, R.P., Formulas and Theorems for the Special Functions of Mathematical Physics, Third Edition, Berlin: Springer-Verlag, 1966.

16. S.A. Zoharian and M. Rosenberg, "Principal-Components Analysis for Low-Redundancy Encoding of Speech Spectra", Journal of the Acoustical Society of America, Vol. 69, pp. 832-845 (March 1981).

17. D.E. Knuth, Sorting and Searching, Reading, MA: Addison-Wesley, 1973, p. 241.

18. J.J. Symanski, "Progress on a Systolic Processor Implementation", Proc. SPIE Tech. Symp. East '82, May 1982.

19. Interactive Laboratory Systems (ILS) Users Guide Version 3.0, Interactive Laboratory Systems (ILS) Programming Guide Version 3.0, 1 October 1980; Signal Technology Inc., Santa Barbara CA 93101

20. G.S. Kang and S.S. Everett, "Improvement of the Narrowband Linear Predictive Coder, Part 1--Analysis Improvements", NRL Report 8645, Naval Research Laboratory, Washington DC, 27 December 1982.

21. G.S. Kang and S.S. Everett, "Improvement of the Narrowband Linear Predictive Coder, Part 2--Synthesis Improvements", NRL Report 8799, Naval Research Laboratory, Washington DC, 11-June-1984.

22. P.T. Brady, "Equivalent Peak Level: A Threshold-Independent Speech-Level Measure", J. Acoust. Soc. Am., Vol. 44, pp. 695-699, 1968.

23. J.T. Sims, "A Speech-To-Noise Ratio Measurement Algorithm", J. Acoust. Soc. Am., Vol. 78, No. 5, pp. 1671-1674, 1985.

24. G.S. Kang and L.J. Fransen, "Low-Bit Rate Speech Encoders Based on Line-Spectrum Frequencies (LSFs)", NRL Report 8857, Naval Research Laboratory, Washington DC, 24 January 1985.

NOTE: Although this report references the following limited documents, no limited information has been extracted.

Item 6. DoD and DoD contractors only; premature dissemination, Mar. 85. Other requests to RADC/EEV Hanscom, AFB MA 07131-5000.

Item 7. DoD and DoD contractors only; software documents, Dec. 85. Other requests to RADC/EEV Hanscom, AFB MA 07131-5000.

Item 9. DoD and DoD contractors only; premature dissemination, Mar. 85. Other requests to RADC/EEV Hanscom, AFB MA 07131-5000.