

AD-A284 224



RL-TR-94-134
Final Technical Report
August 1994



1

FUSION OF CORRELATED SENSOR OBSERVATIONS

Kaman Sciences Corporation
Robert Vienneau



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

102 PJ

94-29383



DTIC QUALITY INSPECTED 3

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

94 9 07 1 92

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-94-134 has been reviewed and is approved for publication.

APPROVED:



JAMES H. MICHELS
Project Engineer

FOR THE COMMANDER:



LUKE L. LUCAS, Colonel, USAF
Deputy Director of Surveillance & Photonics

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 (OCTM) Griffiss AFB NY 13441. 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	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden, to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE August 1994		3. REPORT TYPE AND DATES COVERED Final -----
4. TITLE AND SUBTITLE FUSION OF CORRELATED SENSOR OBSERVATIONS			5. FUNDING NUMBERS C - F30602-93-C-0006 PE - 61102F PR - 2304 TA - E8 WU - 02	
6. AUTHOR(S) Robert Vienneau				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Kaman Sciences Corporation 258 Genesee Street Utica NY 13502-4627			8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory (OCTM) 26 Electronic Pky Griffiss AFB NY 13441-4514			10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-94-134	
11. SUPPLEMENTARY NOTES Rome Laboratory Project Engineer: James H. Michels/OCTM/(315) 330-4432				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <p>Kaman Sciences Corporation has designed and implemented a Graphical User Interface-based fourth generation system for analyzing the algorithms developed for the Multichannel Signal Processing Simulation System (MSPSS).</p> <p>The MSPSS is comprised of two major subsystems, a menu-based subsystem and the User Front-end Interface (UFI) based subsystem. The menu-based subsystem interacts with the user by a series of menus and prompts that can be displayed on a line-oriented "glass terminal". It is implemented as a collection of Fortran programs providing the desired signal processing capabilities. Multichannel data is passed between these programs by user-defined files. The menus themselves are Unix c-shells where the bottom-level menu automatically compiles, links, and executes the desired program.</p> <p style="text-align: center;">DTIC QUALITY INSPECTED 3</p>				
14. SUBJECT TERMS Multichannel, Sensor Fusion, Parameter Estimation, Model-Based Detection, Time-Series Analyses			15. NUMBER OF PAGES 106	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED	

TABLE OF CONTENTS

1.	OVERVIEW	3
	1.1 Background	3
	1.1.1 The Multichannel Signal Processing Simulation System	3
	1.1.2 Recent Research	5
	1.2 Objective	6
	1.3 Tasks Performed	6
	1.3.1 Task 1: Sensor Fusion Capabilities	6
	1.3.2 Task 2: Kalman Filtering	6
	1.3.3 Task 3: Extreme Value Method	7
	1.3.4 Task 4: Ozturk Algorithm	7
2.	THE MULTICHANNEL SIGNAL PROCESSING SIMULATION SYSTEM	8
	2.1 Signal Detection as Statistical Hypothesis Testing	8
	2.2 The Signal Detection Algorithm	8
	2.3 Using the MSPSS	10
3.	NEW CAPABILITIES	17
	3.1 Sensor Fusion	17
	3.1.1 Multichannel Synthesis of AR Processes	17
	3.1.1.1 Synthesis of Gaussian Noise	18
	3.1.1.2 Synthesis of Signal and Clutter as AR Processes	19
	3.1.2 The Multichannel AR Model	20
	3.1.2.1 A Distributed Tapped Delay Line Filter	22
	3.1.2.2 Estimation of AR Coefficients	22
	3.1.2.3 Estimation of the Covariance Matrix	23
	3.2 Model Identification	24
	3.2.1 A State Space Model	25
	3.2.1.1 The Innovations Representation of the State Space Model	25
	3.2.1.2 A Basis Transformation of the Innovations Representation	25
	3.2.1.3 A State Space Representation of An Autoregressive Model	26
	3.2.1.4 A State Space Representation of An ARMA Model	27
	3.2.2 Innovations Generation	29
	3.2.3 Estimation of State Space Parameters	30
	3.2.3.1 Step 1: Estimate Correlation Matrices	30
	3.2.3.2 Step 2: Calculate Square Root Matrices	31

3.2.3.3	Step 3: Calculate The Intermediate Matrix A	32
3.2.3.4	Step 4: Perform the SVD of A	32
3.2.3.5	Step 5: Estimate Model Order	32
3.2.3.6	Step 6: Compute Transformation Matrices	34
3.2.3.7	Step 7: Estimate the System Matrix	34
3.2.3.8	Step 8: Estimate Observation Matrix	34
3.2.3.9	Step 9: Estimate Backward Model Observation Matrix	35
3.2.3.10	Step 10: Determine the Remaining System Model Parameters	35
3.3	Extreme Value Method	35
3.3.1	Maximum Likelihood	36
3.3.2	Probability Weighted Moments	37
3.3.3	Ordered Sample Least Squares	38
3.4	Ozturk's Algorithm	38
3.4.1	Splitting Channels	39
3.4.2	Real Components of Complex Data	40
3.4.3	The Quadratic Form q	40
3.4.4	Ozturk's Goodness of Fit Test	41
3.4.5	Ozturk's Algorithm as Parameter Estimation	44
3.4.5.1	Estimating Shape Parameters	44
3.4.5.2	Estimating Location and Scale Parameters	49
4.	IMPLEMENTATION	51
4.1	Sensor Fusion	51
4.1.1	Estimation Program	51
4.1.2	Covariance Matrix Estimation	52
4.1.3	Sensor Fusion Analysis Sequences	52
4.2	Model Identification	56
4.2.1	Conversion of AR Parameters to State Space Parameters	57
4.2.2	State Space Process Synthesis	58
4.2.3	Estimation of State Space Parameters	59
4.2.4	Innovations Filter Program	61
4.2.5	Kalman Filter Analysis Sequences	61
4.3	Extreme Value Theory	63
5.	REFERENCES	65
Appendix A:	Notation	68
Appendix B:	A Single-Channel MiniSystem	74
Appendix C:	Generation of K-Distributed SIRPs	76

Appendix D: One-Pass Algorithms for Calculating Moments of Distributions	77
D.1 Means and Variances	77
D.2 Correlation Coefficients	81
 Appendix E: RANDOM VARIATE GENERATION	 85
E.1 Uniform Variates	85
E.2 Beta Variates	85
E.3 Cauchy Variates	86
E.4 Exponential Variates	86
E.5 Extreme Value Variates	86
E.6 Gamma Variates	86
E.7 Gumbel (Type II) Variates	87
E.8 Johnson Variates	88
E.9 K Distributed Variates	88
E.10 Laplace Variates	88
E.11 Logistic Variates	89
E.12 Lognormal Variates	89
E.13 Normal Variates	89
E.14 Pareto Variates	90
E.15 Weibull Variates	90
 Appendix F: CONFIDENCE REGIONS	 91
F.1 Bivariate Normal Distribution	91
F.2 The Johnson System of Transformations	92
F.2.1 Fitting the Johnson Distribution	93
F.2.2 Confidence Regions	95

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

LIST OF TABLES

Table B-1: Files in the Minisystem	75
--	----

LIST OF FIGURES

Figure 1-1: Invoking a Fortran Program from the Menu-Based System	4
Figure 2-1: System Structure of the Signal Detection Algorithm	9
Figure 2-2: Generating an AR Process with the Menu-Based System	12
Figure 2-3: Generating an AR Process (Continued)	13
Figure 2-4: Plotting a Data File	14
Figure 2-5: The X Windows Display	15
Figure 2-6: The Final Plot	16
Figure 3-1: Tapped Delay Line Filter for the Centralized Approach	21
Figure 3-2: Decentralized Tapped Delay Line Filter and Fusion Center	22
Figure 3-3: Estimation of AR Coefficients	23
Figure 3-4: Estimation of Covariance Matrix	24
Figure 3-5: Innovations Representation of State Space Process Synthesis	26
Figure 3-6: The Innovations Generation Filter	29
Figure 3-7: Ozturk's Algorithm	43
Figure 3-8: Location and Scale Distributions	45
Figure 3-9: Distributions with One Shape Parameter	46
Figure 3-10: Distributions with Two Shape Parameters	47
Figure 3-11: Linear Interpolation for Shape Parameter	48
Figure 4-1: Sensor Fusion AR Estimation	53
Figure 4-2: Sensor Fusion AR Estimation (Continued)	54
Figure 4-3: Estimating the Covariance Matrix	55
Figure 4-4: Sensor Fusion Analysis Sequences	56
Figure 4-5: Converting AR to State Space Parameters	57
Figure 4-6: State Space Synthesis	58
Figure 4-7: State Space Model Identification	60
Figure 4-8: Kalman Filter	62
Figure 4-9: State Space Analysis Sequences	63

Acknowledgements

The Ozturk algorithm was developed by Dr. Aydin Ozturk, a visiting professor at Syracuse University, and coded by Kaman Sciences Corporation. Dr. James Michels, Dr. Jaime Roman, and Ms. Lisa Slaski provided helpful comments on an earlier draft of this report.

1. OVERVIEW

This document is the final report for Fusion of Correlated Sensor Observations, an effort conducted by Kaman Sciences Corporation (KSC) under Rome Laboratory's Broad Agency Announcement (BAA) number 90-04. This effort was performed for

Dr. James H. Michels
Rome Laboratory
RL/OCTM
Griffiss AFB, NY 13441
(315) 330-4432

1.1 Background

Over the last few years, Dr. James Michels of RL/OCTM has been investigating an original approach in signal processing. He has described this approach and results to date in a series of Rome Laboratory Technical Reports (Michels 89, 90a, 90b, 91, 92a, and 92b). This investigation has developed:

- A synthesis procedure for simulating multichannel autoregressive (AR) processes in which intertemporal and interchannel correlations are controlled parametrically.
- Extensions to this synthesis procedure to handle non-Gaussian Spherically Invariant Random Process (SIRP) noise and unconstrained quadrature components.
- Statistical diagnostics for analyzing the expected performance of various multichannel estimating algorithms.
- A multichannel signal detection algorithm based on a generalized likelihood ratio using an innovations approach. Two ratios currently exist, one for Gaussian processes and one for non-Gaussian SIRPs.

1.1.1 The Multichannel Signal Processing Simulation System

Kaman Sciences has designed and implemented a Graphical User Interface-based fourth generation system for analyzing the algorithms developed during this investigation. This Multichannel Signal Processing Simulation System (MSPSS) is described in the Kaman Sciences report *Man Machine Interface Experiment* (Kaman 91a).

The MSPSS is comprised of two major subsystems, a menu-based subsystem and the User Front-end Interface (UFI) based subsystem. The menu-based subsystem interacts with the user by a series of menus and prompts that can be displayed on a line-oriented "glass terminal." It is implemented as a collection of Fortran programs providing the desired signal processing capabilities. Multichannel data is passed between these programs by user-defined files. The menus themselves are unix c-shells where the bottom-level menu automatically compiles, links, and executes the desired program. Figure 1-1 shows an example of the invocation of a Fortran program from the menu-based subsystem.

The structure of the menu-based subsystem provides the user with a great deal of analytical flexibility. The individual Fortran programs provide certain analytical capabilities, but minimal constraints are imposed on the order in which the user can invoke these functions. A number of diagnostic capabilities are provided, including parameter estimation

% mchan

*The user invokes the menu-based
multichannel system from the Unix shell*

Multi-Channel Detection Algorithm

MAIN MENU

Single Channel	Multichannel
M1 -- Process Synthesis	M11 -- Process Synthesis
M2 -- Filtering Methods	M12 -- Filtering Methods
M3 -- Diagnostics	M13 -- Diagnostics

L1 -- List data files L2 -- show the change log
Enter a command or Q to quit: M11 *The user chooses a submenu*

MULTI-CHANNEL (M/C) PROCESS SYNTHESIS MENU

M0 -- Generate Gaussian noise
M1 -- Method 1 (MC1) process synthesis
M2 -- Method 2 (MC2) process synthesis
M3 -- Method 2 process synthesis with unconstrained quadrature components
M4 -- State space process synthesis
M5 -- Apply Levinson-Wiggins-Robinson algorithm
M6 -- Display Multi-Channel (M/C) signal stats
M7 -- Plot M/C data
M8 -- Perform Nuttall-Strand or Vieira-Morf estimation
M9 -- Perform Yule-Walker estimation
M10 -- Estimate AR parameters for each channel
M11 -- Estimate covariance
M12 -- Perform estimation of state space parameters
M13 -- Perform M/C correlation (temporal)
M14 -- Perform M/C correlation (ensemble)
M15 -- Perform M/C correlation (quadrature, temporal)
M16 -- Perform M/C correlation (quadrature, ensemble)
M17 -- Plot ergodic series equation
M18 -- Split a M/C input M24 -- Join inputs
M19 -- Add M/C signals M25 -- Subtract M/C signals
M20 -- Convert data to ASCII file M26*-- Convert ASCII file to data
M21 -- Convert AR to state space
M22 -- Display complex data M27 -- Display coefficient file
M23 -- Catenate M/C signals M28 -- Sum channels
M29 -- Perform SVD
L1 -- List data files L2 -- show the change log

* denotes options that are not yet available

Enter a command or Q to return to the MAIN menu: M4

Linking program mcssl, please stand by ... *A Fortran program is called*

ff77 -O -C -pipe -cg89 sun4/mcssl.o -Lsun4 -lmc -lvec -lmat -lmc

-L/home/marlin/tom/lib/sun4 -lfsl -s -o bin/sun4/mcssl

Synthesizes state space process

Version 1.4

Do you want to set the pseudo random generator seeds (y/n) [N] ? _

The user interacts with the Fortran program

Figure 1-1: Invoking a Fortran Program from the Menu-Based System

for certain models, correlation function estimation, and graphical display of data. Thus the user is provided with a system that supports an exploratory style of analysis.

The UFI-based subsystem is designed for more repetitive analyses characteristic of the determination of detection probabilities for given false alarm probabilities and given algorithms. In such analyses, the user needs to invoke certain signal processing synthesis and analysis functions in a predetermined order with certain parameters. The UFI-based subsystem supports this need by allowing the user to construct an "experiment" specified by various algorithms and parameters. Once an experiment has been completely described, a computer program for performing an experiment is automatically generated, compiled, and executed. Since these are simulation experiments, the generated program can run for quite some time. Typically, a single experiment will be used to analyze the performance of the signal detection algorithm in terms of the false alarm probability and the probability of detection. A detailed example of the use of this subsystem is provided in the *Software User's Manual for the Multichannel Signal Processing Simulation System* (Kaman 92a).

1.1.2 Recent Research

Some research performed in parallel with the development and use of the MSPSS has some interesting implications for Dr. Michels' approach.

RL has been sponsoring work in sensor fusion (Al-Ibrahim 91, Chair 86, and Hoballah 89). In the sensor fusion problem several sensors, perhaps geographically distributed, each process a signal they receive. These sensors send their outputs to a centrally located fusion center which then must decide if a signal is present. The fusion center combines the results of each of the individual sensors to form a decision for the overall system. One can view each channel in a multichannel processing problem as corresponding to the input to a sensor in the sensor fusion problem. Thus, Dr. Michels' algorithm is a promising approach for the sensor fusion problem, but further research is needed to determine quantitative benefits.

RL has been sponsoring some work on Kalman filtering (Roman 93). In Dr. Michels' software detection algorithm, linear filters are applied to the multichannel radar returns before calculating the loglikelihood ratio upon which the signal detection is based. The tapped delay line filters in use when this effort began can be replaced by other filter structures. In particular, a Kalman filter is applicable. Once again additional research is needed to determine the benefits of using the new Kalman filter being sponsored by RL in Dr. Michels' signal detection algorithm.

Signal detection algorithms are traditionally evaluated in terms of the false alarm probability (the percentage of instances in which the algorithm results in a decision that a signal is present when no signal is in fact present) and the probability of detection (the probability that the algorithm will register a signal when in fact a signal is present). In a simulation approach for evaluating a signal detection algorithm, a very large number of samples needs to be used to accurately explore the tails of probability distributions, since the false alarm probability is itself a "tail probability." Recent research has proposed approximating these tails based on a Pareto distribution (Chakravarthi 92 and Rangaswamy 93). This approach promises at least an order of magnitude improvement in throughput in Dr. Michels' studies, but, once again, further research is needed to validate this claim.

Signal, clutter, and noise are often modeled based on certain probability distributions. A powerful algorithm has recently been developed for determining the probability distribution from which a small random sample is drawn (Ozturk 90a, 90b, 90c). The Ozturk algorithm

provides the MSPSS with an important additional diagnostic capability.

To explore the implications of sensor fusion, Kalman filtering, the Pareto distribution approach, and the Ozturk algorithm for Dr. Michels' signal detection algorithm, new capabilities were added to the MSPSS. The result of incorporating these capabilities was an enhanced simulation system for exploring the performance of a sensor fusion approach using an innovations-based technique.

1.2 Objective

The objective of this effort was to incorporate additional functionality into the Multichannel Signal Processing Simulation System (MSPSS) that Kaman Sciences Corporation has developed to support the analysis described in *Multichannel Detection Using the Discrete-Time Model-Based Innovations Approach*, (Michels 91). Additional functions included the following:

- The ability to support analysis of a sensor fusion algorithm
- A multichannel Kalman filter
- The use of extreme value theory to set thresholds accurately with much lower sample sizes
- The Ozturk algorithm for determining a probability distribution from a random sample.

1.3 Tasks Performed

This effort consisted of four major tasks. Some related minor additional tasks were also performed under this effort.

1.3.1 Task 1: Sensor Fusion Capabilities

Under this task, we added two programs to the menu-based subsystem and four analysis sequences to the UFI-based subsystem. One of the menu-based programs estimates the covariance matrix, and the other estimates Autoregressive coefficients for each channel. Two of the UFI-based analysis sequences test the Autoregressive (AR) coefficients and covariance estimation algorithms. The other two sequences analyze sensor fusion detection algorithms; they differ depending upon whether temporally correlated noise (or clutter) is assumed to be present or not.

These new modules, in combination with previously existing modules, provide the user with the capability to analyze single channel processes within each element of a vector process. This capability allows the user to

- Form scalar linear estimates of AR coefficients for each channel process
- Subsequently process each single channel innovations process at a fusion center to remove residual correlations across channels.

1.3.2 Task 2: Kalman Filtering

When this effort began, a software filter and algorithm implementation of a multichannel Kalman filter was being developed for Rome Laboratory under contract F30602-92-C-0081, titled *State-Space Models for Multichannel Detection* (Roman 93). This task implemented in the MSPSS the algorithms developed under that contract. Kaman Sciences began by examining the source code and the reports produced by the RL contract developing the

multichannel Kalman filter. We then developed code conforming to our conventions and interfaces based on the algorithms developed in the RL contract.

These algorithms were first implemented in the menu-based subsystem of the MSPSS. They were then implemented in the UFI-based subsystem. New capabilities consisted of

- The synthesis of state space processes in the menu-based system
- The estimation of state space model parameters
- The implementation of a Kalman filter
- The implementation of a signal detection algorithm based on the state space model and Kalman filter.

1.3.3 Task 3: Extreme Value Method

Under this task, the extreme value theory method for the threshold level selection, described in (Rangaswamy 93), was implemented. This method is included as an alternative to the previous threshold selection method contained in the detection modules of MSPSS software.

Kaman Sciences first developed software implementing the extreme value theory as a module in the diagnostic menu-based software. This only involved adding an additional likelihood function module, since the previous likelihood function module calculates threshold values and detection probabilities as well. Once it was tested, the extreme value module was then modified to run under the User Front-End Interface (UFI). Kaman Sciences wrote additional detection sequences that parallel the previous functionality, but use the extreme value method.

1.3.4 Task 4: Ozturk Algorithm

Ozturk's algorithm, as described in (Ozturk 90a, 90b, 90c), (Shah 93), and (Slaski 93), was implemented under this effort. This implementation resulted in two programs under the diagnostic menu-based system. Ozturk's algorithm is applicable to single-channel real data. The smaller program provides a front-end for converting many realizations of multichannel data into a single realization of single channel real data by calculating a quadratic form.

The second program calculates Ozturk's statistic. It performs both a goodness-of-fit test and estimates a probability distribution from the input data. The distribution of Ozturk's statistic is not known in closed form. Thus, this program includes an extensive Monte-Carlo capability for determining the expected value and confidence regions for the statistic under a wide range of null hypotheses. An important capability of Ozturk's algorithm is the ability to graphically display the results. The algorithm combines a formal statistical test with more intuitive graphical analysis. Our implementation includes an integrated graphical display.

2. THE MULTICHANNEL SIGNAL PROCESSING SIMULATION SYSTEM

The Multichannel Signal Processing Simulation System (MSPSS) was developed by Kaman Sciences to support certain RL/OCTM research. This section briefly overviews the problems that can be analyzed with the MSPSS, the software design, and how new capabilities are added to the system.

2.1 Signal Detection as Statistical Hypothesis Testing

The MSPSS simulates correlated random vectors which characterize several channels of data. The data gathered at any point in time consists of a real, imaginary, or complex vector with a component for each channel. A single realization or trial of the stochastic process generating this data consists of an ordered sequence of vectors.

The signal detection problem can be cast in the form of a problem in statistical hypothesis testing. Let x denote a vector stochastic process representing the radar return, s denote a signal, c denote the clutter, and n denote white noise. Consider deciding between the null hypothesis

- $H_0: x = c + n$

and the alternative

- $H_1: x = s + c + n.$

To accept the alternative hypothesis is to decide that a signal is present.

Neyman-Pearson theory is generally used to evaluate a decision algorithm in statistical hypothesis testing. A test statistic is defined for summarizing the data in a single realization of the stochastic process. If this statistic is in some critical region, then the alternative hypothesis is accepted. Otherwise the null hypothesis is accepted. The probability of accepting the alternative when the null is true is called the significance level; this probability should be as low as possible. The probability of deciding in favor of the alternative when in fact it is true is known as the power of the test. Statistics are designed to maximize the power of the test for a given significance level and sample size. Alternatively, they are designed to minimize the sample size for a given significance level and power.

Because of the physical interpretation of this particular statistical test, different terminology is used in radar theory. The probability of erroneously concluding a signal is present is the false alarm probability. So the false alarm probability is the same as the significance level. The probability of detecting a signal, when there is in fact a signal, is the probability of detection, or the detection probability. Therefore the detection probability and the power of the test are alternate names for the same probability.

2.2 The Signal Detection Algorithm

Figure 2-1 shows the general structure of the signal detection algorithm implemented in the MSPSS. This is a model-based innovations approach to signal detection. The filters are derived from certain models of the signal, clutter, and noise. If the model for the null hypothesis filter accurately describes the sum of clutter and noise, and the input is clutter and noise, then the output of the null hypothesis filter, $\underline{v}_0(n)$, will be uncorrelated both in time and across channels. The outputs are estimates of the innovations for the model underlying the filter. Similarly, if the model for the alternative hypothesis filter accurately describes the sum of the signal, clutter, and noise, the output $\underline{v}_1(n)$ will be an estimate of the innovations for the

alternative hypothesis model, uncorrelated both in time and across channels, when the input to the filter contains signal, clutter, and noise.

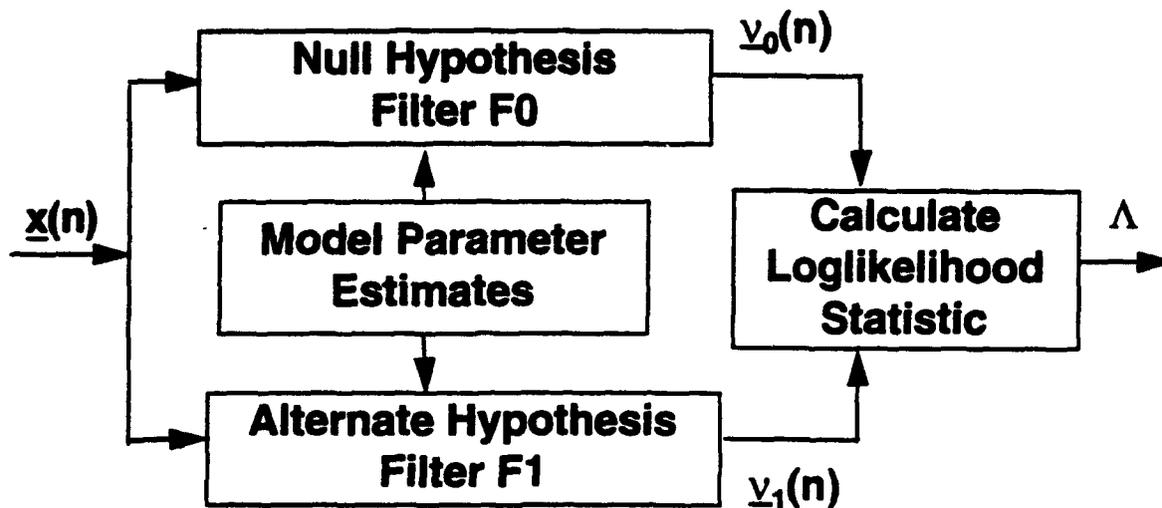


Figure 2-1: System Structure of the Signal Detection Algorithm

The loglikelihood statistic is based on the estimates of the innovations produced by the two filters. If this statistic is above a predefined threshold, one decides a signal is present. Otherwise, the alternative hypothesis is rejected, and one decides no signal is present.

The structure of this algorithm is very general and raises many questions. Different models can be used for the signal, clutter, and noise. Some of these models can be used to approximate other models. For example, if the signal and clutter are Autoregressive (AR) processes and the noise is white, their sum is an Autoregressive Moving Average (ARMA) process. An ARMA process can be approximated by a high order AR process. On the other hand, a State Space model can exactly describe an ARMA process. How does this choice of model impact the performance of the algorithm?

Models may also be chosen on the basis of processing considerations. For example, suppose the sum of the signal, clutter, and noise are approximated by a vector AR process described by Equation 2-1:

$$x(n) = - \sum_{k=1}^p A^H(k)x(n-k) + \epsilon(n). \quad (2-1)$$

where $x(1), x(2), \dots, x(N)$ is the AR process and $\epsilon(n)$ is a random (possibly complex) zero-mean vector with covariance matrix Σ . If the off-diagonal elements of Σ are non-zero, the time samples $x(n)$ are correlated across channels. The matrix AR coefficients $A^H(k)$ lead to

correlation of the process both in time and across channels. The off-diagonal components of the matrix AR coefficients may be non-zero. If they were all zero, Equation 2-1 would be equivalent to Equation 2-2:

$$x_j(n) = - \sum_{k=1}^{P_j} a_j^*(k) x(n-k) + \epsilon_j(n), \quad (2-2)$$

$j = 1, 2, \dots, J$ (J is the number of channels). The model given by Equation 2-2 has the advantage that the corresponding tapped delay line filters F_0 and F_1 can process each channel separately in the signal detection algorithm. This model can be implemented with considerable parallelism, thus gaining an increase in processing speed. But is this increase in processing speed accompanied by a decrease in the detection probability for given false alarm probabilities?

In practice, model parameters will not be known exactly. This raises a host of new questions. For each model, what parameter estimation algorithms are available? How do they perform in various circumstances? For example, Dr. Michels has found that the conditions that increase the accuracy of estimates of the AR weights increase the variability of estimates of the driving noise covariance matrix (Michels 92b). How robust is the signal detection algorithm to errors in parameter estimation and model mismatch? These questions probably do not have one simple answer. Rather, the signal detection performance of any one instantiation of the algorithm shown in Figure 2-1 will vary with process characteristics such as temporal correlation. A full analysis of the model-based innovations approach to signal detection will have to explore the variability in performance with changes in the underlying stochastic processes. Detection probabilities and thresholds will need to be known for given false alarm probabilities and sample sizes in implementing the signal detection algorithm. A complete analysis of the signal detection algorithm will also provide these needed parameters.

The Multichannel Signal Processing Simulation System provides the analyst with tools to answer these questions via Monte-Carlo simulation. Capabilities are provided to synthesize signal, clutter, noise, and their sum as described by a variety of models. The user specifies the number of process realizations to generate, as well as the number of time samples in each realization. Diagnostic capabilities are provided for analyzing the realizations of the generated stochastic processes. The user can calculate means, variances, covariance matrix estimates, correlation functions, model parameter estimates, and Fast Fourier Transformations (FFTs). Graphical capabilities are provided for users operating under a X Windows interface. Functions are provided for manipulating radar returns such as adding them, splitting a multichannel return into several single channel returns, combining single channel returns into multichannel returns, and so on.

Finally, the MSPSS implements the signal detection algorithm described earlier. Tapped delay line and Kalman filters are provided, with user-definable parameters, for removing temporal and cross-channel correlation. Several loglikelihood statistics can be calculated for different models. Thresholds, detection probabilities, and variances in estimates of these parameters can be calculated.

2.3 Using the MSPSS

As was explained in Section 1.1.1, two major subsystems comprise the Multichannel Signal Processing Simulation System (MSPSS), the menu-based subsystem and the UFI-based subsystem. The use of the UFI-based subsystem is extensively described in the *Software User's Manual for the Multichannel Signal Processing Simulation System* (Kaman 92a) and

will not be further examined here.

As shown in Figure 1-1, invoking the menu-based subsystem presents the user with a choice of two sets of three menus. These menus provide the user with a wide range of capabilities. The bottom-level menu initiates execution of a Fortran program. The programs in this system share common conventions for user interaction. Figures 2-2 and 2-3 show an example. The user is prompted for various parameters specifying the synthesis or analysis process. In the example the user describes the synthesis of an Autoregressive process by a "shaping function" approach. The user can receive help messages for prompts that are not self-evident, but a full knowledge of the processing depends on an understanding of the underlying mathematical models.

The menu-based subsystem provides the user with a graphical display capability, as illustrated in Figures 2-4 through 2-6. This capability requires a X Windows interface. The plotting routine first asks the user for the source of the input file and various parameters for the graph (Figure 2-4). An user-friendly windowing interface then appears, permitting the user to manipulate the plot (Figure 2-5). Outputs sent to the screen can also be saved as a Postscript file or printed (Figure 2-6).

An outline of the steps required to add capabilities to the MSPSS is provided in (Vienneau 93). Both the menu-based and UFI-based subsystems are described.

The menus are displayed

Enter a command or Q to return to the MAIN menu: m2
Linking program msirp, please start by ...
'bin/sun4/msirp' is up to date.
MSIRP -- Multichannel AR Process Generation with SIRPs
Version 1.7
Do you want to set the pseudo random generator seeds (y/n) [N] ?
Number of channels ? *A Carriage Return obtains help*
Enter the number of channels to be generated in each output signal or
enter a zero to quit the program.
Number of channels ? 1
Number of points to be generated and saved ? 1000
Enter the number of trials to be generated ? 5
Add a signal component (y/n) [Y] ? n *Values in square brackets []*
Add a clutter component (y/n) [Y] ? *are default values*
ENTER PARAMETERS FOR THE CLUTTER.
Order of the AR process ? 3
Do you want the driving noise to be a Gaussian or SIRP process (g/s) [G] ?
Specify Cholesky, L-D-U, or SVD decomposition (c/l/s) [C] ?
Do you want the process to be real, imaginary, or complex (r/i/c) [C] ?
Use Gaussian or Exponential shaped function (g/e) [G] ?
Enter row 1 of the amplitude matrix:
Enter a row ? 1.0
Enter row 1 of the one-lag temporal correlation matrix:
Enter a row ? 0.5
Enter row 1 of the lag matrix:
Enter a row ? 0
Reference doppler frequency ? 0.5
Enter the sample interval ? 0.01
A complex AR(3) process will be generated.
The driving noise will be Gaussian.
Gaussian shaped function.
Amplitude matrix: 1.0000e+00
Temporal correlation: 5.0000e-01
Lag matrix: 0
Reference doppler frequency: 5.0000e-01 Hz.
Sample interval: 1.0000e-02 seconds.
Are these parameters okay (y/n) [Y] ? *Many programs echo user inputs.*

Figure 2-2: Generating an AR Process with the Menu-Based System

```

Correlation lag values:
R(0)=(1.0000e+00,0.0000e+00)

R(1)=(4.9975e-01,1.5705e-02)

R(2)=(6.2377e-02,3.9244e-03)

R(3)=(1.9445e-03,1.8381e-04)

Determinants of principle minors:
(1.0000e+00,0.0000e+00)(7.5000e-01,0.0000e+00)(5.2734e-01,4.3757e-10)(3.6500e-01,6.8065e-10)
Coefficients for the AR process:
a(1)= (-6.5593e-01,-2.0613e-02)

a(2)= (3.2748e-01,2.0603e-02)

a(3)= (-1.2445e-01,-1.1764e-02)

White noise covariance matrix (WNCM): (6.9214e-01,0.0000e+00)
WNCM Cholesky decomp.: (8.3195e-01,0.0000e+00)
Add a white noise component (y/n) [Y] ? N
Total output file name ? r1var The user saves the results in
Title of this dataset ? An AR process a file for later analysis
Clutter output file name ?
No file name entered, is this okay ? y
Output coefficient file name ? r1var.coef
Title of this dataset ? AR Coefficients
Clutter AR coefficients and covariances written to file.
Enter the number of points to be generated before saving data ? 5000
ok?

MULTI-CHANNEL (M/C) PROCESS SYNTHESIS MENU

.
.
.
The user returns to the menus

```

Figure 2-3: Generating an AR Process (Continued)

MULTI-CHANNEL (M/C) DIAGNOSTICS MENU

- D1 -- Display Multi-Channel (M/C) signal stats
- D2 -- Plot M/C data
- D3 -- Perform Nuttall-Strand or Vieira-Morf estimation
- D4 -- Perform Yule-Walker estimation
- D5 -- Estimate AR parameters for each channel
- D6 -- Estimate covariance
- D7 -- Perform estimation of state space parameters
- D8 -- Perform M/C correlation (temporal)
- D9 -- Perform M/C correlation (ensemble)
- D10 -- Perform M/C correlation (quadrature, temporal)
- D11 -- Perform M/C correlation (quadrature, ensemble)
- D12 -- Perform Ozturk's algorithm
- D13 -- Plot ergodic series equation
- D14 -- Generate periodogram
- D15 -- Split a M/C input
- D16 -- Add M/C signals
- D17 -- Convert data to ASCII file
- D18 -- Display complex data
- D19 -- Test modified Bessel function
- D20 -- Calculate SIRP quadratic form
- D21 -- Join inputs
- D22 -- Subtract M/C signals
- D23*-- Convert ASCII file to data
- D24 -- Display coefficient file
- D25 -- Calculate detection probability
- D26 -- Sum channels

L1 -- List data files L2 -- show the change log

* not yet implemented

Enter a command or Q to return to the MAIN menu: d2

Linking program kplt, please stand by ...

'bin/sun4/kplt' is up to date.

KPLT -- convert binary to text file and plot under Openwindows

Version 1.7

Input file name ? r1var

The file to plot

Title: An AR process

First trial to plot ? 1

Number of trials to plot ? 1

X axis title [Time] ?

X axis range ?

Y axis title [Magnitude] ?

Data smoothing ? 0

Complex data conversion: Real, Imaginary, Magnitude, or Angle [R] ? r

converting data ...

Channels written: 1

Trials written: 1

The plot window appears here

ok?

Figure 2-4: Plotting a Data File

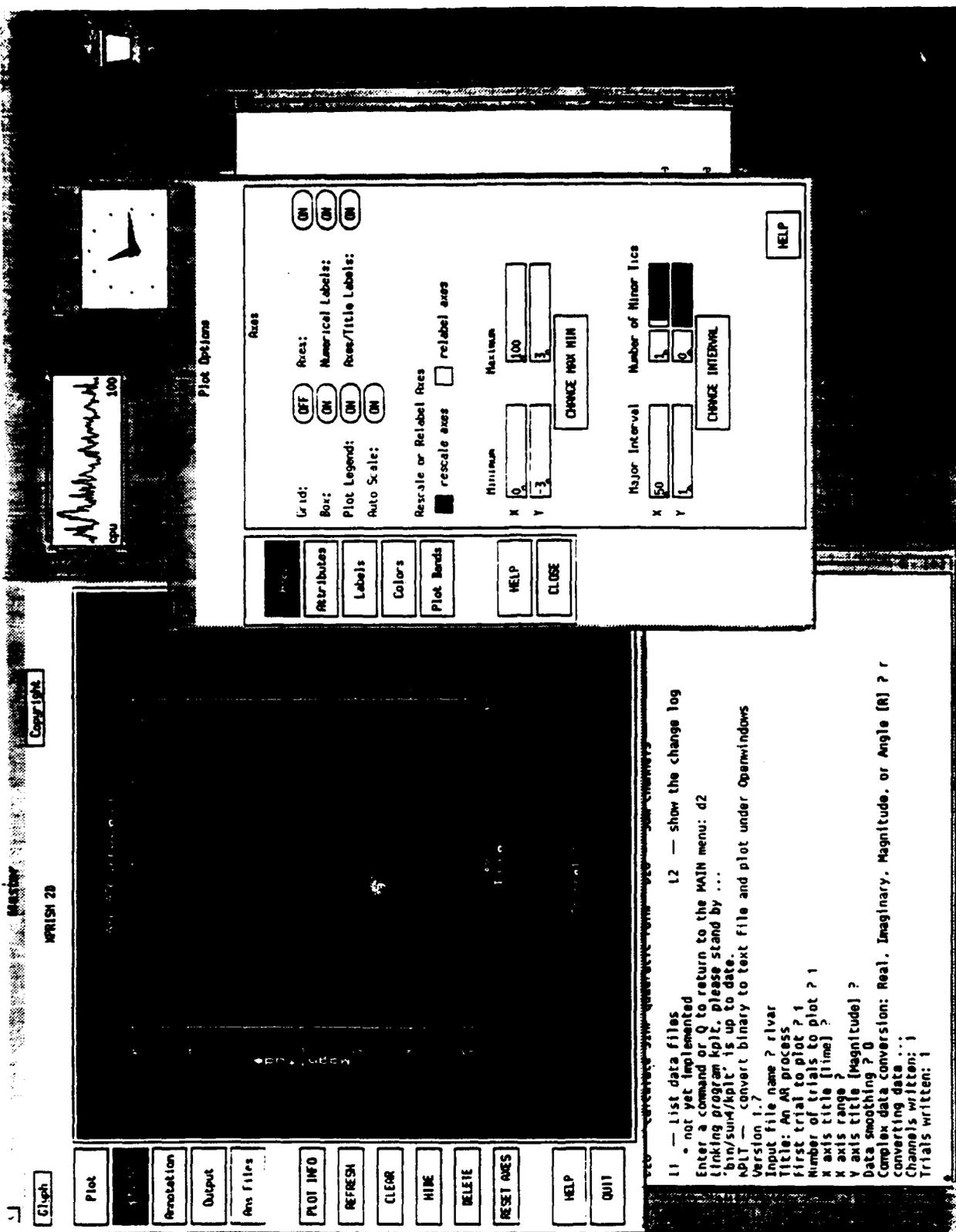
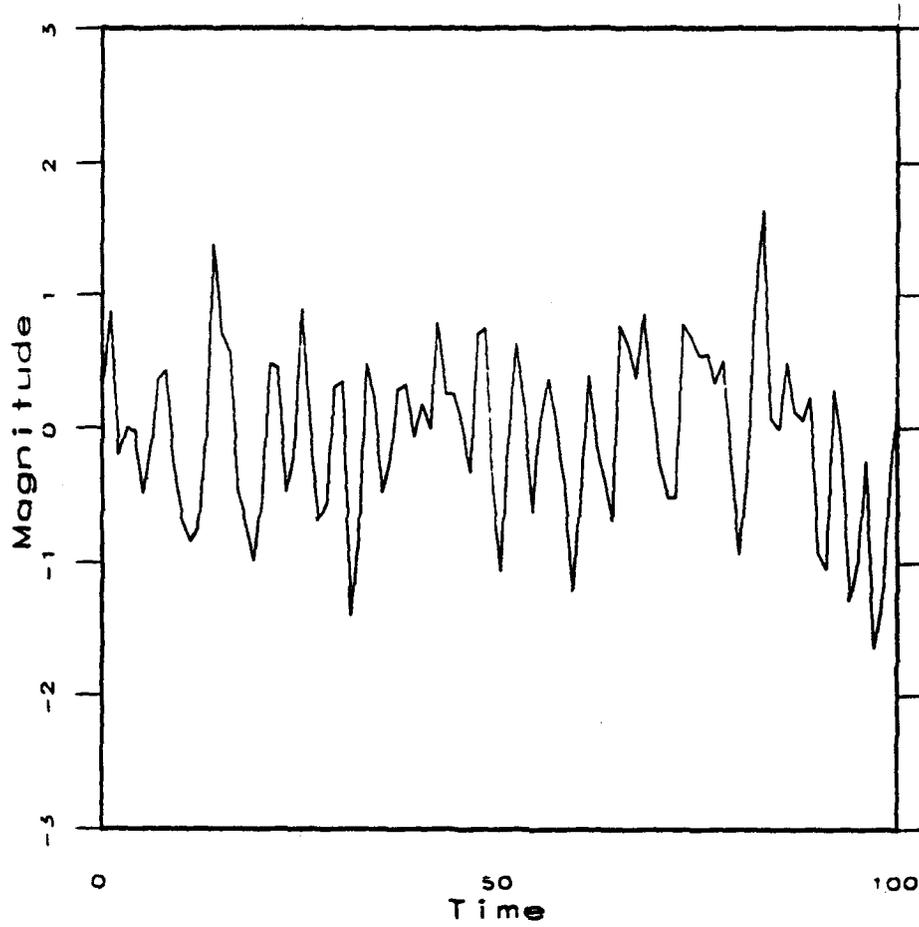


Figure 2-5: The X Windows Display

An AR process



— Channel 1

Figure 2-6: The Final Plot

3. NEW CAPABILITIES

The signal detection algorithm described in Section 2 has been implemented over a series of efforts developing the Multichannel Signal Processing Simulation System. Typically, each effort has implemented a new model or a capability recently developed by RL/OC and their contractors. Under the present effort, new capabilities were added to the MSPSS in the areas of

- Sensor fusion
- State space models and Kalman filtering
- The Extreme Value Method for estimating tail probabilities
- Ozturk's algorithm

New capabilities were first implemented in the menu-based subsystem. When appropriate, they were then implemented in the UFI-based subsystem. This section defines the capabilities implemented under this effort.

3.1 Sensor Fusion

Signal processing algorithms typically process large volumes of data. These demands often stress current computer processor technology. One method for meeting this demand is for concurrent processors to analyze data in parallel. A multichannel system could consist of sensors for each channel. Distributed processing would be performed by each sensor, with subsequent processing being performed at a fusion center.

Previously, the multichannel detection approach consisted of processing the observation data vector using multichannel prediction error filters to obtain a whitened error residual. These processes were obtained by the simultaneous removal of the temporal and cross-channel correlation information. This approach required the use of all the multichannel data at the processing center and is referred to as the centralized approach. In contrast, the method developed here is a decentralized approach which individually processes data on each channel to first form separate error signals on each channel before passing these signals to a central processor for whitening across channels. The performance of this scheme is suboptimal to the centralized approach, but represents applications for which the data is first preprocessed on each channel (or sensor) to reduce the redundant information content before transferal to the centralized processor. In space-time signal processing, this case is referred to as the factored approach in which temporal and spatial processing is performed distinctly. Finally, the computational requirements of the decentralized approach are less than the centralized case.

3.1.1 Multichannel Synthesis of AR Processes

The processes analyzed by the decentralized approach are generated using the existing multichannel MSPSS synthesis procedure. No new process synthesis routines were developed. The radar return is generated as the sum of signal, clutter, and noise. Either signal or clutter may be absent. The noise is synthesized as white noise, while the signal and clutter are AR processes. The sum of white noise and AR processes is an ARMA process, which may be approximated as a higher order AR process.

Specifically, let $x(1), x(2), \dots, x(N)$ be a stochastic process formed by the sum of three processes:

$$x(n) = s(n) + c(n) + w(n), \quad n = 1, 2, \dots, N. \quad (3-1)$$

The sequence of vectors $x(n)$ represent the radar return, $s(n)$ is the signal, $c(n)$ is the clutter, and $w(n)$ is the noise. Each of the subprocesses can be real, imaginary, or complex. The number of elements in each vector $x(n)$, $s(n)$, $c(n)$, and $w(n)$ is the number of channels J .

A single sequence of vectors $x(1), x(2), \dots, x(N)$ is referred to as a *trial* or *realization* of the stochastic process. Since this is a random process, it will vary from trial to trial. Each vector $x(n)$ in the sequence of vectors is referred to as a *time sample*. Again, because of the random nature of the process, the N time samples vary across a single realization of the underlying process with a specified temporal and cross-channel correlation.

3.1.1.1 Synthesis of Gaussian Noise

The noise vector $w(n)$ is synthesized as a zero-mean Gaussian white noise process with covariance matrix Σ_w :

$$\Sigma_w = E [w(n) w^H(n)]. \quad (3-2)$$

The noise process is uncorrelated in time, but may be correlated across channels if the off-diagonal elements of Σ_w are nonzero. By definition, Σ_w is Hermitian. The components of Σ_w have a physical interpretation. The diagonal elements are the variances of the corresponding channels, and the off-diagonal elements are the cross-channel covariances.

The user specifies either the Cholesky decomposition, the LDU decomposition, or the Singular Value Decomposition (SVD) of the covariance matrix. The Cholesky decomposition is given by Equation 3-3:

$$\Sigma_w = C_w C_w^H. \quad (3-3)$$

where C_w is a lower triangular complex matrix. The LDU decomposition is

$$\Sigma_w = L_w D_w L_w^H. \quad (3-4)$$

where L_w is a lower triangular matrix with ones along its diagonal and D_w is a diagonal matrix.

The SVD decomposition of Σ_w is given by Equation 3-5:

$$\Sigma_w = Q_w \Lambda_w Q_w^H. \quad (3-5)$$

Λ_w is a diagonal matrix whose elements are eigenvalues of Σ_w . The columns of Q_w are the corresponding right hand eigenvectors, that is,

$$\Sigma_w (Q_w)_{.j} = (\Lambda_w)_{j,j} (Q_w)_{.j}. \quad (3-6)$$

The rows of Q_w^H are the left hand eigenvectors:

$$(Q_w^H)_i \Sigma_w = (\Lambda_w)_{i,i} (Q_w^H)_i. \quad (3-7)$$

The eigenvectors in Q_w should have a norm of unity. Also, since Σ_w is Hermitian, Q_w^H is the inverse of Q_w .

To generate the white noise vector, the user specifies either C_w , L_w and D_w , or Q_w and Λ_w . For each time sample in a realization of Gaussian white noise, a vector, $v_w(n)$, is generated of statistically independent zero-mean normally distributed random variates. The j th channel has a variance of either unity, if the user specified a Cholesky decomposition; $(D_w)_{j,j}$, if the user specified an LDU decomposition; or $(\Lambda_w)_{j,j}$, if the user specified a SVD. (If the noise is complex, the real and imaginary components of $v_w(n)$ are independently generated

with variances of $\frac{\sigma^2}{2}$, where σ^2 is the desired variance of $v_w(n)$.)

The white noise is then generated as

$$w(n) = T_w v_w(n), \quad (3-8)$$

where T_w is either C_w , L_w , or Q_w .

3.1.1.2 Synthesis of Signal and Clutter as AR Processes

The signal and noise are each synthesized as Autoregressive processes. This section describes the synthesis of the clutter. The signal is synthesized in an identical manner. The AR process for the clutter $c(1), c(2), \dots, c(N)$ is defined by Equation 3-9:

$$c(n) = - \sum_{k=1}^P A_c^H(k) c(n-k) + \varepsilon_c(n), \quad (3-9)$$

where $c(n)$ is a $J \times 1$ vector process and $A_c^H(1), A_c^H(2), \dots, A_c^H(P)$ are the $J \times J$ AR matrix coefficients for an AR process of order P . $\varepsilon_c(n)$ is the driving noise term. It is a zero mean process, uncorrelated across time, with covariance matrix Σ_c defined by Equation 3-10:

$$\Sigma_c = E [c(n)c^H(n)]. \quad (3-10)$$

The driving noise is synthesized as white noise following the procedure described in Section 3.1.1.1 above. The user specifies either the Cholesky, LDU, or Singular Value Decomposition of the covariance matrix Σ_c . The innovations, uncorrelated both in time and across channels, are generated with appropriate channel variances. A matrix from the appropriate decomposition is used to premultiply each time sample from the innovations. This process yields the driving noise time samples $\varepsilon_c(n)$ with the appropriate correlations across channels.

Direct specification of the AR coefficients $A_c^H(k)$ and the covariance matrix Σ_c does not easily permit explicit control of the temporal and cross-channel correlation of the resulting AR process. Consequently, Dr. Michels has developed a "shaping function" approach for controlling AR parameters in terms of physically meaningful parameters (Michels 91).

For the stationary zero-mean stochastic process $c(1), c(2), \dots, c(N)$, a matrix showing the correlations across channels for the k th lag is defined by Equation 3-11:

$$R_c(k) = E [c(n)c^H(n-k)]. \quad (3-11)$$

Note that if the number of channels is J , $R_c(k)$ is a $J \times J$ matrix. Furthermore, it follows from this definition that Equation 3-12 holds:

$$R_c(-k) = R_c^H(k). \quad (3-12)$$

The correlation matrix and the AR parameters are related by the Yule-Walker equations. For example, if there are three lags, $P = 3$, the Yule-Walker equations are given by 3-13:

$$\begin{bmatrix} I & A_c^H(1) & A_c^H(2) & A_c^H(3) \end{bmatrix} \begin{bmatrix} R_c(0) & R_c(1) & R_c(2) & R_c(3) \\ R_c(-1) & R_c(0) & R_c(1) & R_c(2) \\ R_c(-2) & R_c(-1) & R_c(0) & R_c(1) \\ R_c(-3) & R_c(-2) & R_c(-1) & R_c(0) \end{bmatrix} = \begin{bmatrix} \Sigma_c & 0 & 0 & 0 \end{bmatrix} \quad (3-13)$$

Given the correlation matrix R_c , these equations can be solved for the AR coefficients and the driving noise covariance matrix. The MSPSS uses the Levinson-Wiggins-Robinson algorithm to solve the Yule-Walker equations.

The correlation matrix R is specified by the user through the use of parameters describing certain shaping functions from which the correlation matrix is calculated. Shaping functions of a Gaussian and an Exponential form are available. The Gaussian shaping function for the cross-correlation function is given by Equations 3-14 and 3-15 (Michels 90a):

$$R_{c,i,j}(k) = K_{c,i,j} \frac{\lambda_{c,i,j}^{[(k-l_{c,i,j})^2]}}{\lambda_{c,i,j}^{(l_{c,i,j})^2}} e^{2\pi\phi_c T_c k \sqrt{-1}}, \quad i \leq j \quad (3-14)$$

$$R_{c,i,j}(k) = R_{c,j,i}^*(-k), \quad i > j. \quad (3-15)$$

K_c is a constant proportional to the variances on each channel, λ_c is the one-lag temporal correlation parameter, l_c is the lag value at which the function peaks, ϕ_c is the Doppler shift, and T_c is the sampling period. (x^* is the complex conjugate of the complex number x .)

The Exponential shaping function is defined by Equations 3-16 and 3-17:

$$R_{c,i,j}(k) = K_{c,i,j} \frac{\lambda_{c,i,j}^{|k-l_{c,i,j}|}}{\lambda_{c,i,j}^{(l_{c,i,j})}} e^{2\pi\phi_c T_c k \sqrt{-1}}, \quad i \leq j \quad (3-16)$$

$$R_{c,i,j}(k) = R_{c,j,i}^*(-k), \quad i > j. \quad (3-17)$$

3.1.2 The Multichannel AR Model

The synthesis of signal, clutter, and noise has been described in Section 3.1.1. The sum of these processes is, in general, an ARM process. However, an ARMA process can be approximated by a high order AR model. This model may permit correlation between channels. Some of the off-diagonal elements or the AR coefficients may very well be non-zero. A filter based on this model would be a tapped delay line filter with matrix coefficients. Such filters were previously implemented in the MSPSS, but they do not support decentralized sensor fusion analyses.

Let $x(1), x(2), \dots, x(N)$ be a discrete-time complex-valued process. $x(n)$, $n = 1, 2, \dots, N$, is a J element vector where J is the number of channels. The multichannel AR process is described by Equation 3-18:

$$x(n) = - \sum_{k=1}^P A^H(k) x(n-k) + \varepsilon(n), \quad (3-18)$$

where $\varepsilon(n)$ is a $J \times 1$ zero-mean Gaussian white driving noise vector process with covariance matrix Σ . Since the AR coefficients $A^H(k)$ and Σ are matrices, the process exhibits cross-channel correlation.

The driving noise is modeled by one of three decompositions of Σ . The Cholesky decomposition is

$$\Sigma = C C^H, \quad (3-19)$$

where C is a lower triangular matrix. The corresponding model of the driving noise in Equation 3-18 is given by Equation 3-20:

$$\varepsilon(n) = C v(n), \quad (3-20)$$

where $v(n)$ is a $J \times 1$ zero-mean unit variance Gaussian noise vector uncorrelated both in time and across channels.

The LDU decomposition of Σ is given by Equation 3-21:

$$\Sigma = L D L^H \quad (3-21)$$

D is a diagonal matrix, and L is a lower triangular matrix with unity along the principle diagonal. The corresponding model of the driving noise is given by Equation 3-22:

$$\epsilon(n) = L v(n) \quad (3-22)$$

In this case, the covariance matrix of $v(n)$ is D .

The SVD of Σ is given by Equation 3-23:

$$\Sigma = Q \Lambda Q^H \quad (3-23)$$

where Λ is a diagonal matrix whose elements are eigenvalues of Σ . The columns of Q are the corresponding right hand eigenvectors, and the rows of Q^H are the left hand eigenvectors. The eigenvectors in Q have a norm of unity. Also, since Σ is Hermitian, Q^H is the inverse of Q . For the SVD, the model of the driving noise is given by Equation 3-24:

$$\epsilon(n) = Q v(n) \quad (3-24)$$

where $v(n)$ is a zero-mean Gaussian noise process with covariance matrix Λ .

Figure 3-1 shows the prediction error filter used to produce white noise from a signal synthesized by this model. This filter implements the centralized approach. The tapped delay line structure whitens the output, $\epsilon(n)$, across channels. T is either C , L , or Q in the Cholesky, LDU, or Singular Value Decomposition respectively. Thus, the final output of this filter, $v(n)$, is uncorrelated both in time and across channels.

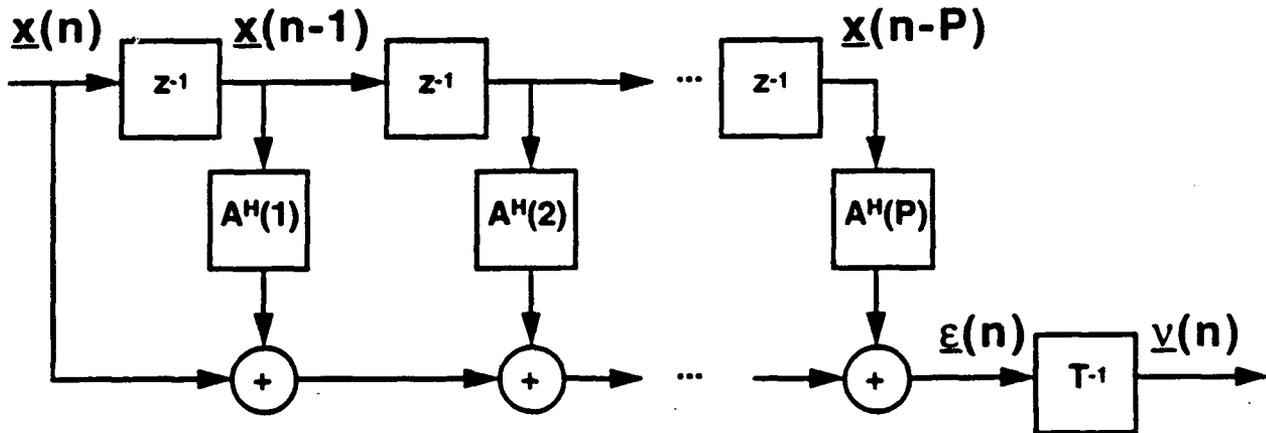


Figure 3-1: Tapped Delay Line Filter for the Centralized Approach

3.1.2.1 A Distributed Tapped Delay Line Filter

In this subsection, we now discuss the form of the prediction error filter used in the decentralized (or factored) signal processing approach. Figure 3-2 shows this filter. We note that this filter lacks the off-diagonal terms in the matrix coefficients of multichannel prediction error filter shown in Figure 3-1 above.

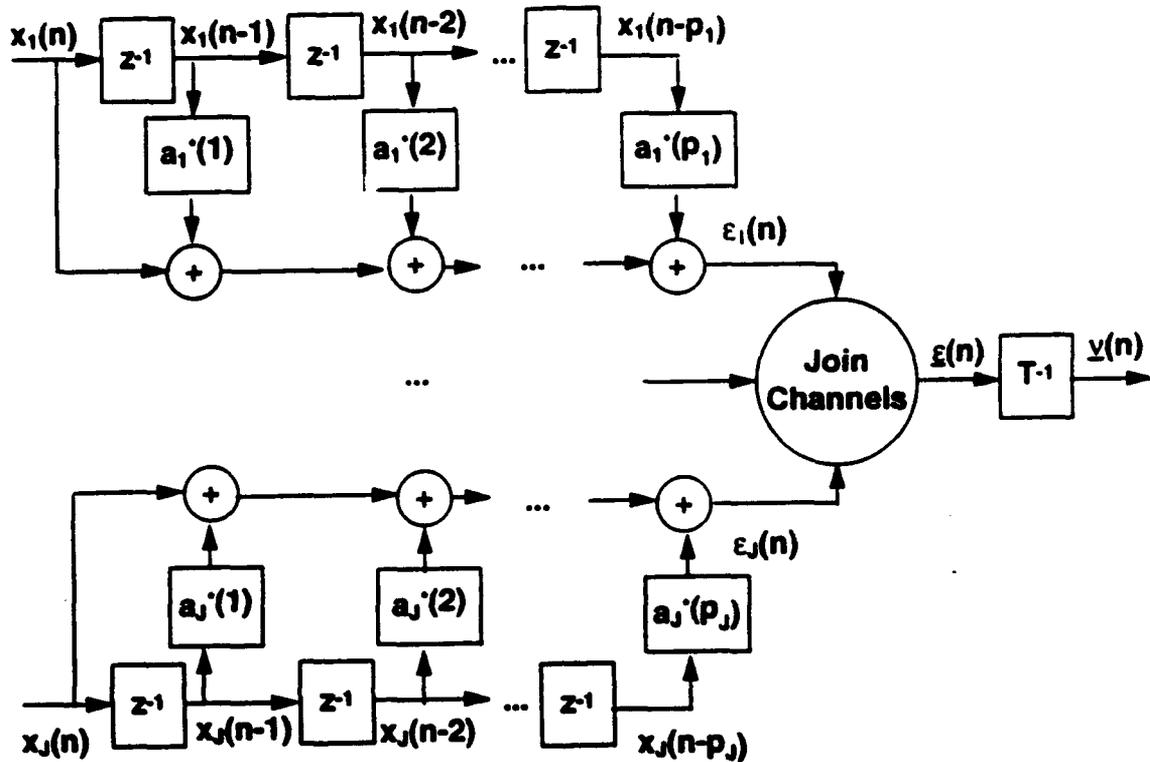


Figure 3-2: Decentralized Tapped Delay Line Filter and Fusion Center

The filter is comprised of tapped delay line filters which first remove temporal correlation in each channel j :

$$\varepsilon_j(n) = x_j(n) + \sum_{k=1}^{p_j} a_j^*(k)x_j(n-k), \quad j = 1, 2, \dots, J. \quad (3-25)$$

The tapped delay line portion of the filters can be distributed among the sensors for each channel. A second phase of the filter, implemented at the sensor fusion site, removes remaining correlation between channels:

$$v(n) = T^{-1} \varepsilon(n), \quad (3-26)$$

where T is either C , L , or Q in the Cholesky, LDU, or Singular Value Decomposition, respectively.

3.1.2.2 Estimation of AR Coefficients

In practice, the filter parameters Σ and $a_j^*(k)$, $k = 1, 2, \dots, p_j$, $j = 1, 2, \dots, J$, will be estimates obtained from the data. These estimates are obtained in two stages. First, the AR coefficients $a_j^*(k)$ are estimated from the j th channel data. Then, the resulting error signals from the J channels are used to estimate the covariance matrix Σ .

Figure 3-3 shows the system structure used in estimating the AR coefficients. The input is many realizations of the modeled process. A set of AR coefficients, $\hat{a}_j^*(1)$, $\hat{a}_j^*(2)$, ..., $\hat{a}_j^*(P_j)$, is estimated for each realization using some suitable single-channel algorithm. The MSPSS currently allows the user to choose between the Yule-Walker and Burg algorithms implemented as described in (Marple 87). The user must specify the order, P_j , for these algorithms; it is not estimated. These estimation algorithms also produce an estimate of the variance, $\hat{\sigma}_j^2$ for each channel. These variance estimates are discarded and not used further.

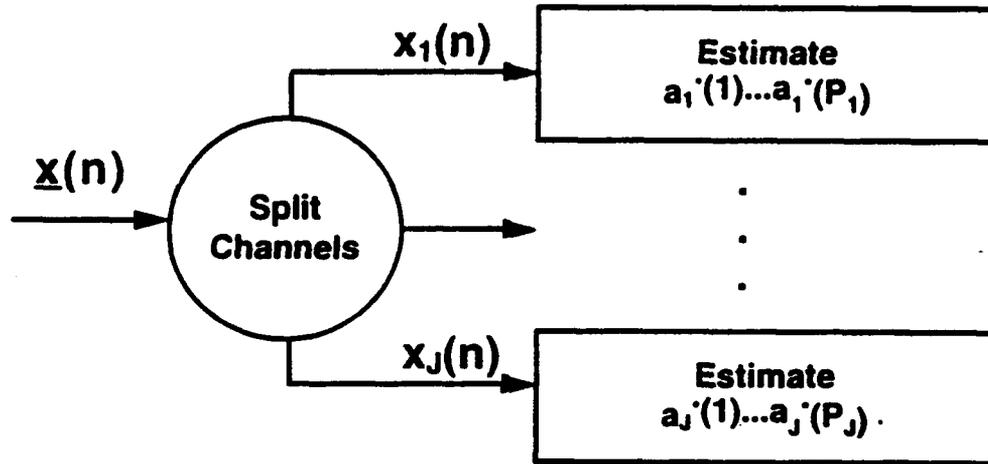


Figure 3-3: Estimation of AR Coefficients

Once a set of estimates of AR coefficients is obtained for each trial, they can be averaged together across trials at the user's discretion. This average is the desired estimate of the AR coefficients to be used in the distributed tapped delay line filter shown in Figure 3-2. The sample variance of these estimates can also be calculated across channels to assess algorithm performance. Appendix D describes a numerically stable one-pass algorithm useful in estimating means and variances.

3.1.2.3 Estimation of the Covariance Matrix

The covariance matrix of the resulting error signals on the J channels is estimated from the output data from the J channels. As can be seen from Figure 3-4, the covariance matrix estimation process relies on the previously determined AR coefficients $\hat{a}_j^*(1)$, $\hat{a}_j^*(2)$, ..., $\hat{a}_j^*(P_j)$. These values can either be actual values, if known, or the estimates obtained by the process described above. The covariance matrix is estimated from the $J \times 1$ vector $\epsilon(n)$ produced by the first phase in the filter.

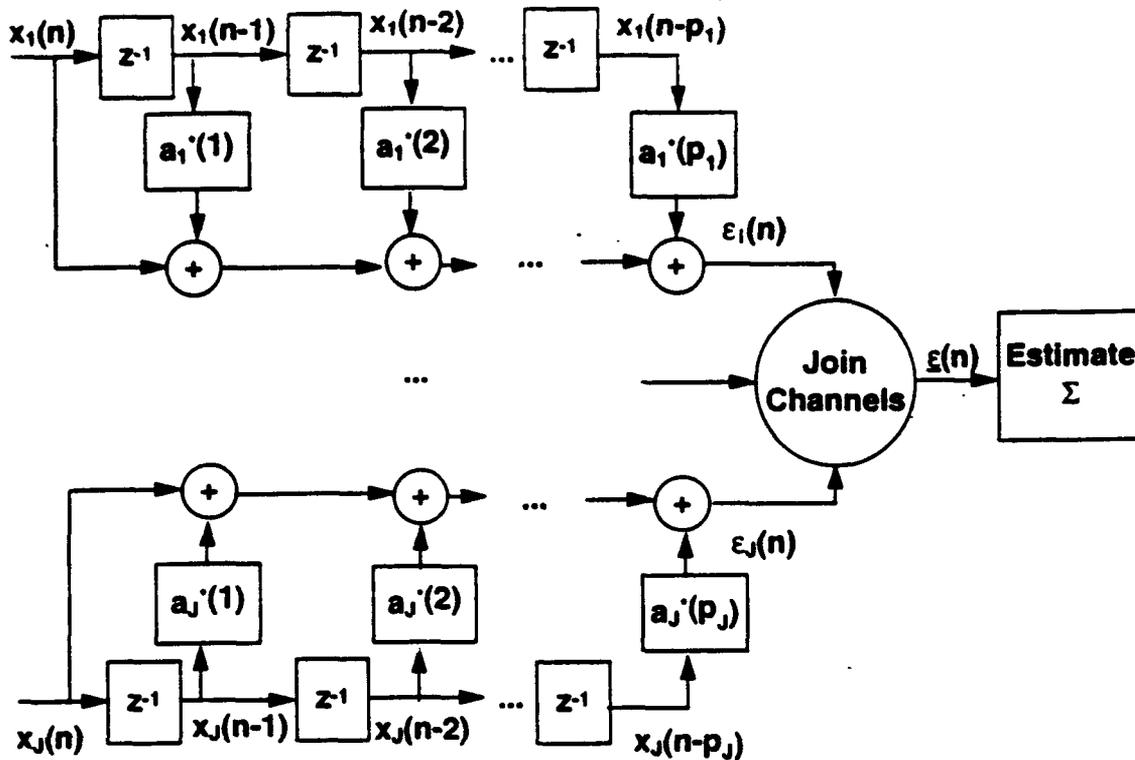


Figure 3-4: Estimation of Covariance Matrix

The covariance matrix is estimated for each trial of $\epsilon(n)$ as follows. First, the channel mean is found across time samples:

$$\bar{\epsilon}_j = \frac{1}{N} \sum_{n=1}^N \epsilon_j(n). \quad (3-27)$$

where N is the number of time samples (the mean should be statistically equal to zero). The variances and covariances are then estimated by Equation 3-28:

$$\hat{\Sigma}_{i,j} = \frac{1}{N} \sum_{n=1}^N [\epsilon_i(n) - \bar{\epsilon}_i][\epsilon_j(n) - \bar{\epsilon}_j]^H. \quad (3-28)$$

The resulting estimates are averaged across trials to obtain the estimate of Σ whose decomposition is used in the filter.

3.2 Model Identification

The sum of noise and clutter, or of noise, clutter, and signal, can be modeled as a state space process. In the model-based signal detection algorithm described in Section 2.2, the corresponding filter is a Kalman filter. This approach has been developed in (Roman 93), and the state space model and Kalman filter were implemented in the MSPSS under this effort. This implementation provided the following new capabilities:

- Conversion of AR model parameters to state space model parameters
- State space process synthesis based on direct user input of model parameters
- Estimation of state space model parameters
- A Kalman filter that converts the radar return process to Gaussian white noise (innovations).

- A signal detection algorithm based on the state space model.

3.2.1 A State Space Model

Under the state space approach, the radar returns are modeled as follows. Let $x(1), x(2), \dots, x(N)$ be a discrete-time complex-valued process representing the radar returns generated as described in Equations 3-29 and 3-30:

$$y(n+1) = F y(n) + G u(n) \quad (3-29)$$

$$x(n) = H^H y(n) + D^H w(n). \quad (3-30)$$

$x(n), u(n),$ and $w(n)$ are J -element vectors, where J is the number of channels. The process $u(1), \dots, u(N)$ denotes the input, while $w(1), \dots, w(N)$ is measurement noise.

$y(n)$ denotes an M -element vector representing the state variables. The state space can be thought of as a memory associated with the process. The state variables evolve in accordance with the first equation above, while the value of the output process at any time is a transformation of the state variables and measurement noise.

3.2.1.1 The Innovations Representation of the State Space Model

To implement an innovations-based signal detection method, filters are constructed such that the outputs of the filters are the innovations under the appropriate hypotheses. Innovations are defined by a (not necessarily Gaussian) white-noise process. Kalman filters are used to produce the innovations.

The state-space model can be transformed into the following innovations representation:

$$\alpha(n+1) = F \alpha(n) + K \varepsilon(n) \quad (3-31)$$

$$x(n) = H^H \alpha(n) + \varepsilon(n). \quad (3-32)$$

The process $\varepsilon(1), \varepsilon(2), \dots, \varepsilon(N)$ defines the innovations, and the process $\alpha(1), \alpha(2), \dots, \alpha(N)$ is the state of the innovations representation. The matrix K is the Kalman filter gain.

Figure 3-5 shows a system block diagram for the state space model. Although the output process and certain matrix parameters are represented by the same symbols as in Equation 3-29 and 3-30, these parameters may reflect a change in basis. The innovations model is "correlation equivalent" to the general state space model.

3.2.1.2 A Basis Transformation of the Innovations Representation

The innovations representation of the state space model is defined only up to a basis transformation of the state vector. Let T be an invertible $M \times M$ matrix representing a basis transformation of the state vector. Define $\beta(n), F_\beta, K_\beta,$ and H_β^H as follows:

$$\beta(n) = T \alpha(n) \quad (3-33)$$

$$F_\beta = T F T^{-1} \quad (3-34)$$

$$K_\beta = T K \quad (3-35)$$

$$H_\beta^H = H^H T^{-1}. \quad (3-36)$$

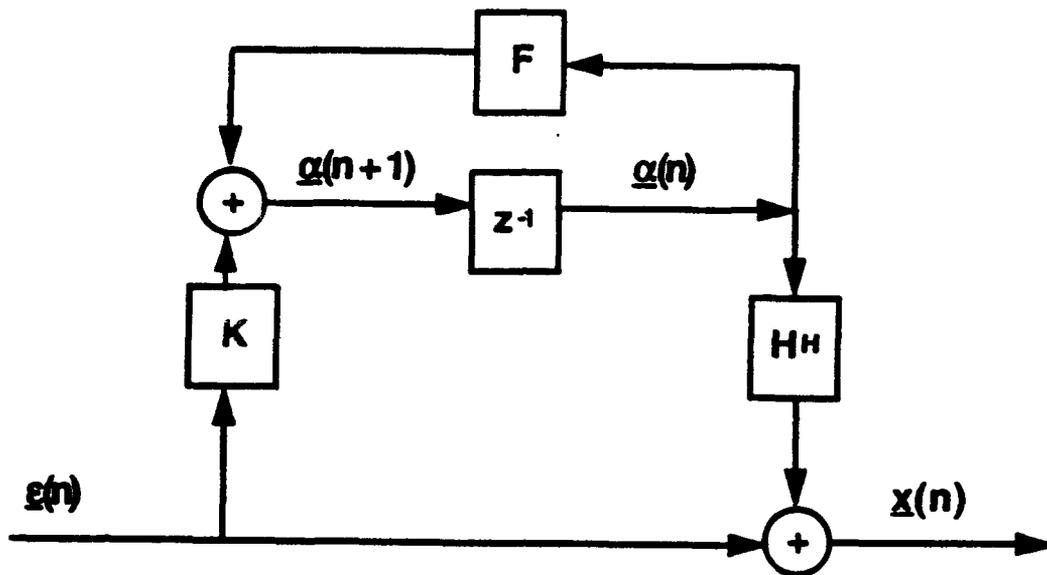


Figure 3-5: Innovations Representation of State Space Process Synthesis

Under this basis transformation, an alternative innovations representation of the state space model is given by Equation 3-37 and 3-38:

$$\beta(n+1) = F_{\beta} \beta(n) + K_{\beta} \varepsilon(n) \quad (3-37)$$

$$x(n) = H_{\beta}^H \beta(n) + \varepsilon(n). \quad (3-38)$$

The innovations and the output process remain unchanged.

Equations 3-37 and 3-38 are of the same form as Equations 3-31 and 3-32. Both sets of equations are innovations representations of the same state space model. This demonstrates that the innovations representation is unique only up to a basis transformation of the state space. A practical implication is a difficulty in validating state space estimation algorithms. The output process may be synthesized with given state space parameters, but estimates of these parameters obtained from the output process can differ from the parameters used in synthesis. These differences are acceptable as long as they represent a basis transformation of the original parameters. Whether this is so or not may be difficult to determine. In any case, the model order and the covariance matrix for the innovations are unique.

3.2.1.3 A State Space Representation of An Autoregressive Model

An Autoregressive (AR) model can be cast in the form of a state-space model. Consider the case of the P th order AR model $x(1), x(2), \dots, x(N)$:

$$x(n) = - \sum_{k=1}^P A^H(k) x(n-k) + \varepsilon(n). \quad (3-39)$$

where $\varepsilon(n)$ is a zero mean driving noise term with covariance matrix Σ .

In an AR model, the current value of the process is a linear combination of the last P values and white noise. In a state space model, the current value is a function of the state variables and white noise. Thus, let the state variables be the past P values of the process:

$$y(n) = \begin{bmatrix} x(n-1) \\ x(n-2) \\ \vdots \\ x(n-P) \end{bmatrix} \quad (3-40)$$

Since $x(n)$ is a J -element vector, there are $M = P \times J$ state variables.

To complete the state space representation of the AR process, let the state-space parameters be defined as follows:

$$F = \begin{bmatrix} -A^H(1) & -A^H(2) & \cdots & -A^H(P-1) & -A^H(P) \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix} \quad (3-41)$$

$$G = \begin{bmatrix} I \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \quad (3-42)$$

$$H^H = [-A^H(1) -A^H(2) \cdots -A^H(P)] \quad (3-43)$$

$$D^H = I \quad (3-44)$$

Furthermore, let the input noise and the measurement noise be equal to the driving noise process in the AR model:

$$w(n) = u(n) = \varepsilon(n). \quad (3-45)$$

Using these definitions of the relevant parameters, the input process, and the measurement noise, the AR process is represented in a state space form, as defined in Equations 3-29 and 3-30. This state space representation is also the innovations representation of the state space model.

3.2.1.4 A State Space Representation of An ARMA Model

An Autoregressive Moving Average (ARMA) model can also be conceptualized as a state-space model. Define the ARMA model to be generated by the sum of AR and white noise processes:

$$x(n) = s(n) + \varepsilon_1(n). \quad (3-46)$$

where

$$s(n) = - \sum_{k=1}^P A^H(k) s(n-k) + \varepsilon_2(n). \quad (3-47)$$

This is a special type of ARMA model, and the state space representation defined below for this model is not valid for all types of ARMA models.

In the state space representation, the state variables are the $M = P \times J$ variables defined by past values of the AR process:

$$y(n) = \begin{bmatrix} s(n-1) \\ s(n-2) \\ \vdots \\ s(n-P) \end{bmatrix} \quad (3-48)$$

The state space parameters are defined as in the AR model:

$$F = \begin{bmatrix} -A^H(1) & -A^H(2) & \cdots & -A^H(P-1) & -A^H(P) \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix} \quad (3-49)$$

$$G = \begin{bmatrix} I \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \quad (3-50)$$

$$H^H = [-A^H(1) -A^H(2) \cdots -A^H(P)] \quad (3-51)$$

$$D^H = I \quad (3-52)$$

The state space representation of this type of ARMA process is distinguished from an AR process by the treatment of the input process and the measurement noise:

$$u(n) = \varepsilon_2(n) \quad (3-53)$$

$$w(n) = \varepsilon_1(n) + \varepsilon_2(n) \quad (3-54)$$

Notice that the resulting state space model is not an innovations representation. Conversion of this model to an innovations representation requires determination of the Kalman gain for this system.

Other special cases of the state space model are also of interest, although not defined here. For example, the multipath model defined in (Michels 91) can be represented by a state space model.

3.2.2 Innovations Generation

Given a state space model of the process $x(n)$, a Kalman filter can be used to generate the innovations sequence. Alternatively, the whitening filter associated with the Kalman filter can be used. The MSPSS uses a steady state approximation to a one-step predictor Kalman filter followed by a linear transformation. This filter is shown in Figure 3-6.

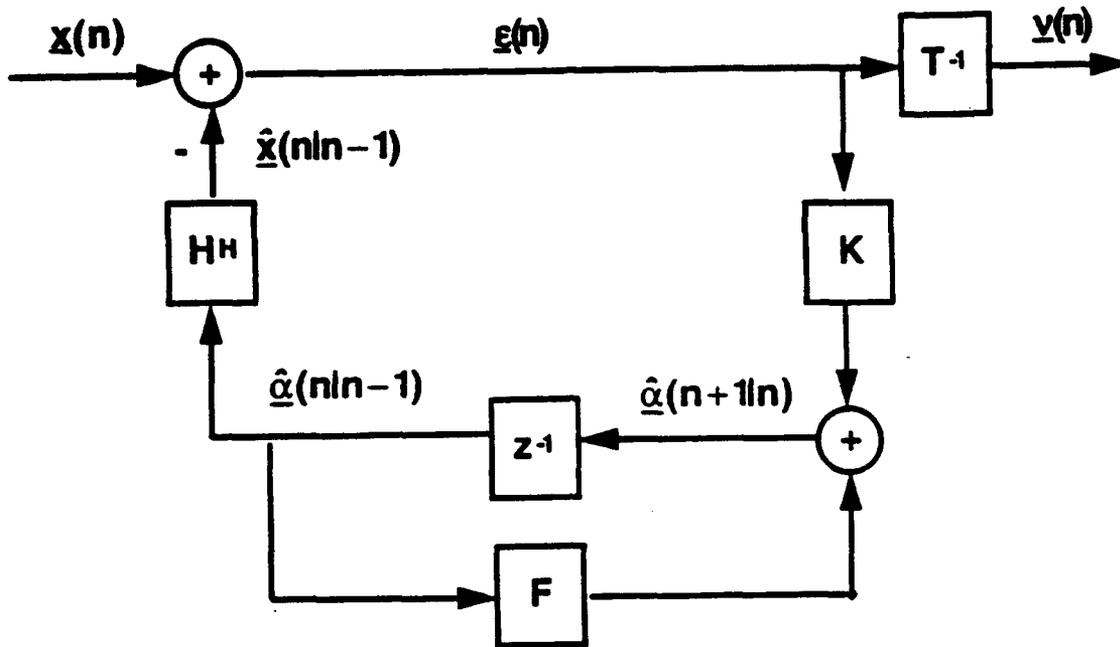


Figure 3-6: The Innovations Generation Filter

Formally, let $x(1), x(2), \dots, x(N)$ be the input process to the filter. The first phase is a Kalman filter, and its output is the process $\epsilon(1), \epsilon(2), \dots, \epsilon(N)$. The second phase is a spatial filter which removes correlation between channels. The output of this second phase, and therefore the output of the entire filter, is $v(1), v(2), \dots, v(N)$.

The output of the first phase at each time sample is the difference between the input and a minimum variance estimate of the input:

$$\epsilon(n) = x(n) - \hat{x}(n|n-1). \quad (3-55)$$

The estimate of the input is based on an estimate of the state variables in the innovations representation of the state space model:

$$\hat{x}(n|n-1) = H^H \hat{\alpha}(n|n-1). \quad (3-56)$$

The estimate of the state variables is updated in accordance with the system dynamics:

$$\hat{\alpha}(n+1|n) = F \hat{\alpha}(n|n-1) + K \epsilon(n) \quad (3-57)$$

$$\hat{\alpha}(0|-1) = 0. \quad (3-58)$$

where K is the "Kalman gain matrix," and the matrix F embodies the "system dynamics." The

product of the gain matrix and the output, $K \epsilon(n)$, is known as the "correction vector." In many Kalman filter implementations, the gain matrix is updated for each time sample. The fact that K is not updated for this Kalman filter reflects a steady state approximation.

The second phase of the filter is based on a decomposition of the covariance matrix Σ of the driving noise process in the innovations representation of the state space model. Let T be either C , L , or Q in the Cholesky, LDU, or Singular Value Decomposition of Σ , respectively. The output of the second phase is calculated by decorrelating the output of the first phase across channels:

$$v(n) = T^{-1} \epsilon(n). \quad (3-59)$$

When a SVD of Σ is used, T^{-1} is equal to T^H . Hence, the inverse of Q is calculated as its Hermetian transpose.

If the filter parameters embody the state space model from which the input is generated, the output of this Kalman filter will be a reasonable estimate of the innovations process. In practice, the true filter parameters are unavailable, and estimates are used for the matrices K , F , H^H , and Σ .

3.2.3 Estimation of State Space Parameters

The MSPSS can be used to synthesize AR and ARMA processes, as well as a few more complex processes (e.g. for multipath). As shown in Section 3.2.1, AR and ARMA processes can be represented exactly in a state space model. This section defines the Canonical Correlations Algorithm, which is used to estimate state space model parameters in the MSPSS.

Let $x(1), x(2), \dots, x(N)$ be a single realization of a stochastic process generated by a state space model. The Canonical Correlations Algorithm uses data on N_R such realizations to generate an estimate of the state space model order M and the matrices F , K , and H^H which appear in the innovations representation of the state space model (Section 3.2.1.1). The covariance, Σ , of the innovations is also computed.

3.2.3.1 Step 1: Estimate Correlation Matrices

The algorithm is based on estimates of correlation matrices, not the raw data. These matrices are easily calculated. Let Λ_l represent the correlations both within a channel and between channels at lag l :

$$\Lambda_l = E [x(n)x^H(n-l)]. \quad (3-60)$$

Since $x(n)$ is a J -element column vector and its Hermitian transpose is a J -element row vector, Λ_l is an array with J rows and J columns. Λ_l is a Hermitian matrix only for the zeroth lag. The Hermitian transpose of correlation matrices for positive lags provides a convenient method for calculating correlations for negative lags:

$$\Lambda_{-l} = \Lambda_l^H. \quad (3-61)$$

Given data on N_R realizations of a stochastic process, the correlation matrices can be estimated by averaging the sample correlations over time and across all trials. Two methods exist, one producing biased estimates of the correlation matrices and the other producing unbiased estimates. For nonnegative l , the biased estimate is generated as

$$\hat{\Lambda}_i = \frac{1}{N_R} \sum_{i=1}^{N_R} \frac{1}{N} \sum_{k=i+1}^N x^i(k) [x^i(k-l)]^H, \quad (3-62)$$

where $x^i(n)$ is the n th time sample of the vector $x(n)$ from the i th realization of the stochastic process. The unbiased estimate is generated as

$$\hat{\Lambda}_i = \frac{1}{N_R} \sum_{i=1}^{N_R} \frac{1}{N-l} \sum_{k=i+1}^N x^i(k) [x^i(k-l)]^H. \quad (3-63)$$

The correlation matrices $\hat{\Lambda}_0, \hat{\Lambda}_1, \dots, \hat{\Lambda}_{L-1}$ are used to estimate the past and future block correlation matrices. The estimate of the $JL \times JL$ past correlation matrix, in block form, is

$$\hat{R}_{P:L,L} = \begin{bmatrix} \hat{\Lambda}_0 & \hat{\Lambda}_1 & \cdots & \hat{\Lambda}_{L-1} \\ \hat{\Lambda}_{-1} & \hat{\Lambda}_0 & \cdots & \hat{\Lambda}_{L-2} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\Lambda}_{1-L} & \hat{\Lambda}_{2-L} & \cdots & \hat{\Lambda}_0 \end{bmatrix} = E \begin{bmatrix} x(n-1) \\ x(n-2) \\ \vdots \\ x(n-L) \end{bmatrix} [x^H(n-1) \ x^H(n-2) \ \dots \ x^H(n-L)] \quad (3-64)$$

The estimate of the $JL \times JL$ future block correlation matrix is

$$\hat{R}_{F:L,L} = \begin{bmatrix} \hat{\Lambda}_0 & \hat{\Lambda}_{-1} & \cdots & \hat{\Lambda}_{1-L} \\ \hat{\Lambda}_1 & \hat{\Lambda}_0 & \cdots & \hat{\Lambda}_{2-L} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\Lambda}_{L-1} & \hat{\Lambda}_{L-2} & \cdots & \hat{\Lambda}_0 \end{bmatrix} = E \begin{bmatrix} x(n) \\ x(n+1) \\ \vdots \\ x(n+L-1) \end{bmatrix} [x^H(n) \ x^H(n+1) \ \dots \ x^H(n+L-1)] \quad (3-65)$$

Notice that the future block correlation matrix, $\hat{R}_{F:L,L}$, is the block transpose of the past block correlation matrix, $\hat{R}_{P:L,L}$. Furthermore, both the past and future block correlation matrices are Hermitian.

The constant L , which defines the number of block rows and columns in the past and future block correlation matrices, is a user input. It should be between one and 20, and chosen such that the user believes JL to be greater than M , the number of state variables.

3.2.3.2 Step 2: Calculate Square Root Matrices

Square root matrices are calculated for the past and future block correlation matrices. The square root is obtained using the Singular Value Decomposition (SVD) of the block correlation matrices. The SVDs of the past and future block correlation matrices are defined by Equations 3-66 and 3-67:

$$\hat{R}_{P:L,L} = U_P S_P U_P^H. \quad (3-66)$$

$$\hat{R}_{F:L,L} = U_F S_F U_F^H. \quad (3-67)$$

The matrices S_P and S_F are diagonal matrices composed of the eigenvalues of the block correlation matrices arranged in decreasing order of magnitude. The eigenvalues of the matrices in Equations 3-64 and 3-65 are positive. The square roots of matrices S_P and S_F can be calculated on an element-by-element basis along the principal diagonal. Since the block correlation matrices are Hermitian, the inverses of U_P and U_F are their Hermitian transposes.

Therefore, the following hold:

$$\hat{R}_{P:L,L} = (U_P S_P^{1/2} U_P^H)(U_P S_P^{1/2} U_P^H). \quad (3-68)$$

$$\hat{R}_{F:L,L} = (U_F S_F^{1/2} U_F^H)(U_F S_F^{1/2} U_F^H). \quad (3-69)$$

Thus, the square roots of the block correlation matrices can be calculated as shown in Equations 3-70 and 3-71:

$$\hat{R}_{P:L,L}^{1/2} = U_P S_P^{1/2} U_P^H. \quad (3-70)$$

$$\hat{R}_{F:L,L}^{1/2} = U_F S_F^{1/2} U_F^H. \quad (3-71)$$

3.2.3.3 Step 3: Calculate The Intermediate Matrix A

An estimate of the $JL \times JL$ stochastic block Hankel matrix is found as follows:

$$H_{L,L} = \begin{bmatrix} \hat{\Lambda}_1 & \hat{\Lambda}_2 & \cdots & \hat{\Lambda}_L \\ \hat{\Lambda}_2 & \hat{\Lambda}_3 & \cdots & \hat{\Lambda}_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\Lambda}_L & \hat{\Lambda}_{L+1} & \cdots & \hat{\Lambda}_{2L-1} \end{bmatrix} = E \begin{bmatrix} x(n) \\ x(n+1) \\ \vdots \\ x(n+L-1) \end{bmatrix} [x^H(n-1) \ x^H(n-2) \ \cdots \ x^H(n-L)] \quad (3-72)$$

The intermediate $JL \times JL$ matrix A is defined as follows:

$$A = \hat{R}_{F:L,L}^{-1/2} H_{L,L} \hat{R}_{P:L,L}^{-1/2} = (U_F S_F^{-1/2} U_F^H) H_{L,L} (U_P S_P^{-1/2} U_P^H). \quad (3-73)$$

3.2.3.4 Step 4: Perform the SVD of A

An SVD of A is performed next. The SVD is defined as:

$$A = U_A S_A V_A^H. \quad (3-74)$$

where S_A is a diagonal matrix with the singular values of A along the principal diagonal in order of decreasing magnitude. All the singular values are positive real numbers between zero and unity. Let $\rho_1, \rho_2, \dots, \rho_{JL}$ denote the (ordered) singular values. These singular values are the correlations between the normalized past and the normalized future, and are referred to as the canonical correlations.

3.2.3.5 Step 5: Estimate Model Order

The model order (the number of state variables), M , can be determined from the canonical correlations. In theory, exactly M canonical correlations should be nonzero, but due to numerical and statistical estimation errors some may be nonzero. Methods for determining model order are based on identifying the number of canonical correlations that are meaningfully different from zero.

A number of methods exist for determining the model order. These methods are based on:

- A. The canonical correlations
- B. The normalized running sum of the canonical correlations

- C. The square of the canonical correlations
- D. The normalized running sum of the squares of the canonical correlations
- E. Log parameters
- F. Normalized mutual information parameters

The user can circumvent the whole process of calculating the model order by entering the order directly.

Method A.

The model order M can be determined as the largest value such that for all $k > M$, the canonical correlation ρ_k is less than or equal to some user-specified (small) value between zero and unity.

Method B.

The normalized running sum of the canonical correlations is defined to be:

$$NRS(k) = \frac{\sum_{i=1}^k |\rho_i|}{\sum_{i=1}^{JL} |\rho_i|} \quad (3-75)$$

In theory, all values of ρ_i should be nonnegative, but numerical errors may lead to small negative values. The model order can be selected as the smallest value M such that for all $k \geq M$, $NRS(k)$ is greater than or equal to some user-specified (large) value between zero and unity.

Method C.

The squares of the canonical correlations are merely $\rho_1^2, \rho_2^2, \dots, \rho_{JL}^2$. The model order M can be determined as the largest value such that for all $k > M$, the canonical correlation ρ_k^2 is less than or equal to some user-specified (small) value between zero and unity.

Method D.

The normalized running sum of the squares of the canonical correlations is defined to be:

$$NRSS(k) = \frac{\sum_{i=1}^k \rho_i^2}{\sum_{i=1}^{JL} \rho_i^2} \quad (3-76)$$

The model order can be the smallest value M such that for all $k \geq M$, $NRSS(k)$ is greater than or equal to some user-specified (large) value between zero and unity.

Method E.

The log parameters are defined as $-\ln(1 - \rho_1^2), -\ln(1 - \rho_2^2), \dots, -\ln(1 - \rho_{JL}^2)$. Since the canonical correlations are sorted in decreasing order, the log parameters are also in decreasing order. The model order M can be determined as the largest value such that for all $k > M$, the log parameter $-\ln(1 - \rho_k^2)$ is less than or equal to some user-specified (small) value between zero and unity.

Method F.

The normalized mutual information parameters are defined to be:

$$v_k = \frac{\sum_{i=1}^k \ln(1 - \rho_i^2)}{\sum_{i=1}^{JL} \ln(1 - \rho_i^2)} \quad (3-77)$$

The model order can be the smallest value M such that for all $k \geq M$, v_k is greater than or equal to some user-specified (large) value between zero and unity.

Model order determination is an important step in the canonical correlations algorithm, and can be implemented using any of the methods defined above.

3.2.3.6 Step 6: Compute Transformation Matrices

The two $JL \times JL$ "transformation" matrices T_P and T_F are defined by Equations 3-78 and 3-79:

$$T_P = V_A^H \hat{R}_P^{-1/2} \quad (3-78)$$

$$T_F = U_A^H \hat{R}_F^{-1/2} \quad (3-79)$$

3.2.3.7 Step 7: Estimate the System Matrix

The column-shifted $JL \times JL$ block Hankel matrix is defined as follows:

$$\bar{H}_{L,L} = \begin{bmatrix} \hat{\Lambda}_2 & \hat{\Lambda}_3 & \cdots & \hat{\Lambda}_{L+1} \\ \hat{\Lambda}_3 & \hat{\Lambda}_4 & \cdots & \hat{\Lambda}_{L+2} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \hat{\Lambda}_{L+1} & \hat{\Lambda}_{L+2} & \cdots & \hat{\Lambda}_{2L} \end{bmatrix} \quad (3-80)$$

Next the following $JL \times JL$ matrix is calculated:

$$\bar{\Delta} = T_F \bar{H}_{L,L} T_P^H \quad (3-81)$$

Let Z be the square matrix formed from $\bar{\Delta}$ by taking the first M rows and columns, where M is the model order determined using one of the methods specified in Step 5. Aside from numerical errors, all columns of $\bar{\Delta}$ beyond the M th column are zero.

The system matrix F is estimated as:

$$F = S_A^{-1/2} Z S_A^{-1/2} \quad (3-82)$$

where $S_A^{-1/2}$ is the $M \times M$ square matrix in the upper left of S_A obtained using the SVD of A as defined in Step 4. Matrix F is the system matrix in the innovations representation.

3.2.3.8 Step 8: Estimate Observation Matrix

Let Z_H denote the matrix formed from the first J rows and M columns of the matrix $H_{L,L} T_P^H$. Then the Hermitian transpose of the observation matrix H is estimated as follows:

$$H^H = Z_H S_A^{-1/2}. \quad (3-83)$$

This is the observation matrix in the innovations representation.

3.2.3.9 Step 9: Estimate Backward Model Observation Matrix

Let Z_Γ denote the first M rows and J columns of the matrix $T_F H_{L,L}$. Then the backward $M \times J$ observation matrix Γ is estimated as follows:

$$\Gamma = S_A^{-1/2} Z_\Gamma. \quad (3-84)$$

3.2.3.10 Step 10: Determine the Remaining System Model Parameters

The remaining system parameters are estimated as follows:

$$\Pi = S_{A,1} \quad (3-85)$$

$$\Omega = \hat{\Lambda}_0 - H^H \Pi H \quad (3-86)$$

$$K = [\Gamma - F \Pi H] \Omega^{-1}. \quad (3-87)$$

In the innovations representation, Π denotes the (zeroth lag) correlation matrix of the state space vector, Ω denotes the (zeroth lag) correlation matrix of the multichannel innovations process, Λ_0 denotes the (zeroth lag) correlation matrix of the model output process, and K denotes the Kalman gain.

3.3 Extreme Value Method

The determination of thresholds for given false alarm probabilities is one capability of the MSPSS. Thresholds are found by simulating many realizations of the stochastic process corresponding to the null hypothesis, which contains no signal. These realizations are passed through the signal detection algorithm, thereby yielding a list of estimates of the loglikelihood statistic. A threshold is estimated as a value of the statistic such that the probability of exceeding this threshold is the given false alarm probability. Since the false alarm probability is a tail probability, an accurate estimation of thresholds requires the generation of a very large number of realizations of the modeled stochastic process. A full simulation analysis of a signal detection algorithm is thus quite expensive in terms of time.

Prakosh Chakravarthi (92) has developed a method, known as the Extreme Value Theory (EVT), to accurately estimate thresholds based on an order of magnitude less realizations, and this method was implemented in the MSPSS under this effort. The EVT approximates the tail of the distribution of the loglikelihood statistic by a Generalized Pareto distribution. The parameters of the Generalized Pareto distribution can be estimated with fewer samples of the loglikelihood statistic than is needed to directly estimate a threshold. Once the Generalized Pareto distribution is fully known, thresholds can be calculated for any given false alarm probabilities. The programs for estimating thresholds were modified to permit the user to choose this method.

Let $\Lambda^{(1)} \leq \Lambda^{(2)} \leq \dots \leq \Lambda^{(N_R)}$ be ordered sample statistics for the loglikelihood ratio generated under the null hypothesis without a signal present. Let F be the underlying cumulative probability distribution for the (unordered) loglikelihood statistic, and let p denote the desired false alarm probability. The problem treated here is to determine an estimate, $\hat{\eta}$, of the threshold such that

$$F(\eta) = Pr(\Lambda < \eta) = 1 - p. \quad (3-88)$$

where $\hat{\eta}$ is an estimate of η .

The EVT (Rangaswamy 93) is based on approximating the tail of F , where the tail is defined to be the following set of loglikelihood ratios:

$$\{\lambda | \lambda \geq \lambda_0, 1 - F(\lambda_0) = \alpha\}, \quad (3-89)$$

where α is a user-defined (small) probability. (In other words, $Pr(\Lambda \geq \lambda_0) = \alpha$.) The tail of almost any distribution F can be approximated by the Generalized Pareto Distribution. The distribution function, G , for the Generalized Pareto Distribution is given by Equation 3-90:

$$G(\lambda) = 1 - \left[1 + \frac{\gamma\lambda}{\sigma}\right]^{-1/\gamma} \quad (3-90)$$

The tail of F is approximated as follows:

$$\hat{F}(\lambda) = (1 - \alpha) + \alpha G(\lambda - \lambda_0), \lambda > \lambda_0. \quad (3-91)$$

Given estimates of λ_0 , γ , and σ , an estimate of the appropriate threshold can be found for any false alarm probability:

$$\hat{\eta} = \hat{\lambda}_0 + \frac{\hat{\sigma}}{\hat{\gamma}} \left[\left[\frac{\hat{p}}{\alpha} \right]^{-\hat{\gamma}} - 1 \right]. \quad (3-92)$$

(This formula disagrees with Equation 10.51 in (Rangaswamy 93), which is mistaken.)

Once procedures for estimating the tail percentile and the parameters of the Generalized Pareto Distribution are given, the EVT is completely specified. The tail percentile is easily estimated. Let $j = \lfloor \alpha N_R \rfloor$, where $\lfloor \alpha N_R \rfloor$ is the integer part of αN_R . Then, the following equation can be used as an estimate of the tail percentile:

$$\hat{\lambda}_0 = \Lambda^{(N_R - j)}. \quad (3-93)$$

The tail is defined to be those loglikelihood ratios $\Lambda^{(i)}$ such that $\Lambda^{(i)} > \lambda_0$. If $\Lambda^{(N_R - j)}$ is tied with succeeding values, the tail may contain less than j values. Let $Z_{(1)} \leq Z_{(2)} \leq \dots \leq Z_{(m)}$ denote the tail values, that is

$$Z_{(i)} = \Lambda^{(N_R - m + i)} - \lambda_0. \quad (3-94)$$

Three methods are available for estimating the parameters of the Generalized Pareto Distribution:

- Maximum Likelihood
- Probability Weighted Measures
- Ordered Sample Least Squares

3.3.1 Maximum Likelihood

The maximum likelihood method is based on reparameterizing the likelihood function for the tail statistics in terms of σ and τ , where

$$\tau = \gamma/\sigma. \quad (3-95)$$

The derivative of the reparameterized loglikelihood function with respect to σ is

$$\frac{d}{d\sigma} \ln L(\sigma, \tau; z_1, \dots, z_n) = -\frac{m}{\sigma} + \frac{1}{\sigma^2 \tau} \sum_{i=1}^m \ln(1 + \tau z_i). \quad (3-96)$$

(Equation 10.22 in (Rangaswamy 93) contains an error in transcription.) A value that maximizes the loglikelihood ratio can be found by setting this derivative equal to zero. The estimate for σ is then found to be:

$$\hat{\sigma}(\hat{\tau}) = \frac{1}{m \hat{\tau}} \sum_{i=1}^m \ln(1 + \hat{\tau} z_i). \quad (3-97)$$

This expression can be used to express the reparameterized loglikelihood function in terms of τ alone:

$$-\ln L(\tau; z_1, \dots, z_n) = m \ln \hat{\sigma}(\tau) + \left[1 + \frac{1}{\hat{\sigma}(\tau) \tau} \right] \sum_{i=1}^m \ln(1 + \tau z_i). \quad (3-98)$$

An estimate for τ is obtained by minimizing the above equation. The system minimizes this function numerically by using the Nelder-Mead algorithm. The implementation of the Nelder-Mead algorithm in the MSPSS is based on that in (Press 86). This numerical method requires two initial guesses for τ . Our implementation uses two initial guesses based on the Probability Weighted Moments estimates, which can be found in closed form. One is three halves of the Probability Weighted Moments τ , and the other is one half the Probability Weighted Moments τ .

Once $\hat{\tau}$ and $\hat{\sigma}(\hat{\tau})$ are found, γ is estimated by

$$\hat{\gamma} = \hat{\sigma}(\hat{\tau}) \hat{\tau}. \quad (3-99)$$

The above procedure is obtained from (Rangaswamy 93). We did not check that the likelihood function is indeed maximized by these solutions and that the estimates so obtained do not characterize a minimum or a saddle point.

3.3.2 Probability Weighted Moments

Let ε_0 and ε_1 be the following probability weighted moments of the Generalized Pareto Distribution:

$$\varepsilon_0 = E[Z], \quad (3-100)$$

and

$$\varepsilon_1 = E[Z(1 - G(Z))]. \quad (3-101)$$

These parameters are estimated from the data as follows:

$$\hat{\varepsilon}_0 = \frac{1}{m} \sum_{i=1}^m z_i \quad (3-102)$$

$$\hat{\varepsilon}_1 = \frac{1}{m(m-1)} \sum_{i=1}^m (m-i) z_i \quad (3-103)$$

Calculating the relevant integrals yields equations for the probability weighted moments in terms of the parameters of the Generalized Pareto Distribution:

$$\epsilon_0 = \frac{\sigma}{1 - \gamma} \quad (3-104)$$

$$\epsilon_1 = \frac{\sigma}{2(2 - \gamma)} \quad (3-105)$$

These equations can be solved for the Generalized Pareto Distribution parameters. The following estimates of the parameters are thereby obtained:

$$\hat{\sigma} = \frac{2\hat{\epsilon}_0\hat{\epsilon}_1}{\hat{\epsilon}_0 - 2\hat{\epsilon}_1} \quad (3-106)$$

$$\hat{\gamma} = 1 - \frac{\hat{\sigma}}{\hat{\epsilon}_0} = 1 - \frac{2\hat{\epsilon}_1}{\hat{\epsilon}_0 - 2\hat{\epsilon}_1} \quad (3-107)$$

(Equation 10.32 in (Rangaswamy 93) is another, but equivalent formulation.)

3.3.3 Ordered Sample Least Squares

The derivation of the Ordered Sample Least Squares method is given in (Rangaswamy 93). The expected value of each order statistic in the tail can be found as function of γ and σ . The estimators of these parameters are chosen to minimize the sum of the squares of the differences between the observed order statistics and their expected values.

Using this procedure, an analytical formula can be found for σ in terms of γ and the observed data:

$$\hat{\sigma}(\gamma) = \gamma \frac{\sum_{r=1}^m z_r (Q_r(\gamma) - 1)}{\sum_{r=1}^m (Q_r(\gamma) - 1)^2} \quad (3-108)$$

where

$$Q_r(\gamma) = \prod_{i=1}^r \frac{m - i + 1}{m - i + 1 - \gamma} \quad (3-109)$$

The parameter γ is found by a numerical procedure that minimizes the following function:

$$S(\gamma) = \sum_{r=1}^m \left(z_r - \frac{\hat{\sigma}(\gamma)}{\gamma} [Q_r(\gamma) - 1] \right)^2 \quad (3-110)$$

We did not check that a solution exists to this minimization problem. The numerical method requires two initial guesses for γ . Our implementation uses two initial guesses from the Probability Weighted Moments estimate of γ . These guesses are three halves and one half the Probability Weighted Moments γ .

3.4 Ozturk's Algorithm

Dr. Aydin Ozturk has invented a powerful algorithm for determining the distribution of a random sample (Ozturk 90a, 90b, and 90c). It provides a graphical representation of data. The algorithm can be applied in two modes. Under the first mode, it provides a formal statistical hypothesis test of the goodness of fit of a sample. The other mode provides a means of estimating the distribution of the sample. These modes contrast with other goodness-of-fit tests, which usually do not provide an indication of an appropriate distribution if they reject the null hypothesis, where the null hypothesis is that the data come from a specified

distribution. Both modes were implemented under this effort.

Typically, Ozturk's algorithm cannot be applied to the data synthesized by the MSPSS in its raw form. The MSPSS synthesizes multichannel complex data, while Ozturk's algorithm applies to real univariate data. Hence the problem arises concerning how to apply Ozturk's algorithm to complex data. Two possibilities exist. Ozturk's algorithm can be extended to multichannel data (Romeu 90). Or procedures could be created to convert multichannel complex data to univariate real data. The latter approach was adopted for this task.

3.4.1 Splitting Channels

The simplest procedure for analyzing multichannel data is to apply Ozturk's algorithm to each channel separately. The capability to split multichannel data into its component channels already existed at the beginning of this effort, and was not changed here.

In applying Ozturk's goodness-of-fit test to each channel of a multichannel dataset, the user must take into account how the significance level is altered. For example, suppose the data consists of two stochastically independent channels. Consider testing the null hypothesis

- H_0 : $x_1(n)$ is distributed F_1 and $x_2(n)$ is distributed F_2

against the alternative

- H_1 : All other alternatives.

Let α denote the significance level of the overall test:

$$\alpha = Pr(\text{Reject } H_0 | H_0 \text{ true}). \quad (3-111)$$

This statistical test can be considered as a combination of two independent statistical tests. The hypotheses for the two subtests are

- H_0^j : $x_j(n)$ is distributed F_j
- H_1^j : $x_j(n)$ is not distributed F_j ,

where $j = 1, 2$. The significance level for these subtests is α_j :

$$\alpha_j = Pr(\text{Reject } H_0^j | H_0^j \text{ true}). \quad (3-112)$$

A natural decision rule for the overall test is to accept the null hypothesis if both subtests accept their respective nulls. This decision rule implies Equation 3-113 for relating the overall significance level to that of the two subtests:

$$1 - \alpha = Pr((\text{Accept } H_0^1) \text{ and } (\text{Accept } H_0^2) | H_0 \text{ true}) = (1 - \alpha_1)(1 - \alpha_2). \quad (3-113)$$

More generally, if a statistical test is the result of J independent subtests with identical significance levels, the significance level of the subtests, as a function of the overall significance level, is given by Equation 3-114:

$$\alpha_j = 1 - (1 - \alpha)^{1/J}. \quad (3-114)$$

This relationship is not encoded in the MSPSS. Rather, the user must account for it when entering desired significance levels. The user must also account for any correlation between channels and the conversion of complex data to real data.

3.4.2 Real Components of Complex Data

The program implementing Ozturk's algorithm accepts single channel or univariate data created by one of the other programs in the MSPSS. This data can be real or complex. The user is provided four options for converting this data if it is complex. The user can analyze the

- Real part
- Imaginary part
- Magnitude
- Angle.

3.4.3 The Quadratic Form q

A more sophisticated method of converting multichannel complex data to a univariate quantity was also implemented in the MSPSS. This method is based on the theory of Spherically Invariant Random Processes (SIRPs). In particular, a certain quadratic form is a sufficient statistic for SIRPs, and Ozturk's algorithm can be applied to many realizations of that quadratic form. This quadratic form is further defined in this section.

Let $x(1), x(2), \dots, x(N)$ be a multichannel, possibly complex, process. Each time sample $x(n)$ is itself a column vector. If there are J channels, the components of each time sample are as shown in Equation 3-115:

$$x(n) = \begin{bmatrix} x_1(n) \\ x_2(n) \\ \vdots \\ x_J(n) \end{bmatrix} \quad (3-115)$$

The entire stochastic process can be represented as a single vector with JN elements:

$$x = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N) \end{bmatrix} \quad (3-116)$$

The $J \times J$ block correlation matrix summarizes the correlation between channels and among time lags. The correlation matrix for the l th lag is defined as in Equation 3-117:

$$R(l) = E[x(n)x^H(n-l)]. \quad (3-117)$$

The block correlation matrix is defined by Equation 3-118:

$$R = \begin{bmatrix} R(0) & R(1) & \dots & R(N) \\ R(-1) & R(0) & \dots & R(N-1) \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ R(-N) & R(1-N) & \dots & R(0) \end{bmatrix} \quad (3-118)$$

The quadratic form which summarizes a SIRP can now be defined. It is given by Equation 3-119:

$$q = x^H R^{-1} x. \quad (3-119)$$

This quadratic form is always real and positive. Given N_R realizations of a multichannel radar return, Ozturk's algorithm can be applied to the sequence of real scalars q^1, q^2, \dots, q^{N_R} . This data has the form needed to apply Ozturk's algorithm.

The quadratic form is difficult to compute in practice. Radar returns in the MSPSS can contain up to 4 channels and up to 20,000 time samples. Thus, the matrix R can be up to 80,000 x 80,000. Only a special case of the quadratic form was implemented so as to reduce these storage requirements. This special case consists of data uncorrelated across time, but possibly correlated across channels.

For this special case, the block correlation matrix R becomes:

$$R = \begin{bmatrix} \Sigma & 0 & \dots & 0 \\ 0 & \Sigma & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \Sigma \end{bmatrix}. \quad (3-120)$$

where Σ is the $J \times J$ covariance matrix showing cross-channel correlation. The quadratic form is then:

$$q = \sum_{n=1}^N x^H(n) \Sigma^{-1} x(n). \quad (3-121)$$

3.4.4 Ozturk's Goodness of Fit Test

Let $x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(N)}$ be an ordered random sample. These values can be sorted quadratic forms as described above, a single channel of real data, the real part of a single channel of complex data, etc. The application of Ozturk's algorithm as a goodness of fit test allows the user to test whether or not this sample is from a specified distribution. That is, Ozturk's algorithm provides a statistical test for deciding between the null hypothesis

- H_0 : x_n is from a specified probability distribution

and the alternative

- H_1 : otherwise.

The null hypothesis need only be specified as far as the type of the distribution and shape parameters. Location and scale parameters can be any values. Normalization removes any effect from location and scale parameters. The null hypothesis distribution should reflect whatever conversions were used to generate the random sample to which Ozturk's algorithm

is applied. For example, suppose the data being tested is the magnitude of a complex Gaussian process. The null distribution should be Rayleigh (Weibull with a shape parameter of two), since the magnitude of a complex Gaussian random variable is Rayleigh.

Ozturk's statistic Q_N consists of the ordered pair (U_N, V_N) defined by Equations 3-122 through 3-125:

$$U_n = \frac{1}{N} \sum_{i=1}^n |y_{(i)}| \cos \theta_i, \quad (3-122)$$

$$V_n = \frac{1}{N} \sum_{i=1}^n |y_{(i)}| \sin \theta_i, \quad (3-123)$$

$$\theta_i = \pi \Phi(m_{(i)}), \quad (3-124)$$

where $y_{(i)}$ are the standardized values for the ordered sample:

$$y_{(i)} = \frac{x_{(i)} - \bar{x}}{s_x}, \quad (3-125)$$

\bar{x} and s_x are the sample mean and sample standard deviation of the random sample and $m_{(i)}$ is the expected value of the i th order statistic from the reference distribution; that is, the standard Gaussian distribution¹. $\Phi(w)$ is the Cumulative Distribution Function for the reference distribution, as given by Equation 3-126:

$$\Phi(w) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^w e^{-t^2/2} dt. \quad (3-126)$$

Ozturk's algorithm can be visualized as tracing a path out in a two dimensional space (U, V) , as shown in Figure 3-7. The data defines the magnitudes $|y_{(i)}|$ of a sequence of vectors in this space, which are then standardized by the factor $(1/N)$. The sequence of angles between these vectors are defined by the Gaussian reference distribution. The ordered pair given by Ozturk's algorithm is the endpoint of the path formed by linking the vectors together. The null hypothesis defines the expected endpoint of this distribution, as well as the statistical variation about this endpoint and the statistical variation of the path to this endpoint.

The expected order statistics from the Gaussian reference distribution are found by a Monte Carlo estimation algorithm. Let $w_{(1)} \leq w_{(2)} \leq \dots \leq w_{(N)}$ be an ordered sample from a Gaussian distribution with zero mean and unit variance. A user-specified number of realizations of these order statistics are synthesized. Estimates of the expected values of the order statistics are found by averaging over these realizations (see Appendix D).

The user must know the distribution of Q_N under the null hypothesis to decide between the null and alternative hypotheses at a specified significance level. This distribution is not known in closed form. The expected value of the Ozturk statistic is also found by Monte Carlo simulation. Let $z_{(1)} \leq z_{(2)} \leq \dots \leq z_{(N)}$ be an ordered sample from the null hypothesis distribution. Many realizations of this ordered sample are generated. The magnitude of the linked vectors for each realization are given by Equation 3-127:

¹A more general version of Ozturk's algorithm would allow the user to specify the reference distribution as well as the null distribution. Ozturk has shown, however, that the normal distribution empirically works best.

$$|y_{(i)}| = \frac{|z_{(i)} - \bar{z}|}{s_z} \quad (3-127)$$

The sample mean and variance are estimated anew for each realization. The angles between the linked vectors do not change; they are based on the expected values of the ordered sample from the reference distribution, as described above. The expected value of Ozturk's statistic is found by averaging over many realizations of the endpoint of these linked vectors. The average path is shown in Figure 3-7.

Figure 3-7 also shows confidence regions for Ozturk's statistic. These are needed for a formal statistical test. The confidence region should be determined for a level $1 - \alpha$, where α is the significance level of the test. If Ozturk's statistic for the data lies outside the confidence region, the user should reject the null hypothesis that the hypothesized distribution describes the data. In the example, the data is rejected as coming from a Gaussian distribution at the 10% level, but accepted at the 1% level.

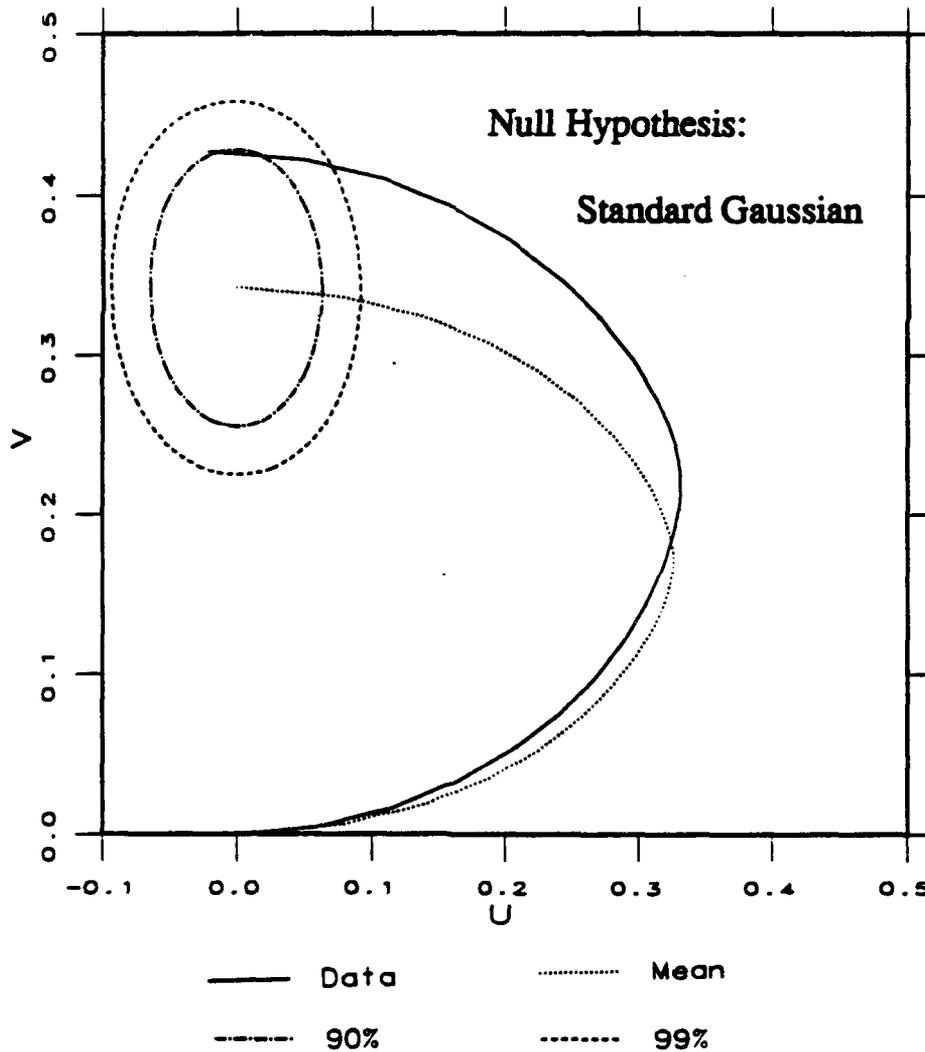


Figure 3-7: Ozturk's Algorithm

Calculation of confidence regions is a fairly complex procedure. Two methods are supplied. One is based on the assumption that the ordered pair (U_N, V_N) is approximately bivariate normally distributed. The other is based on transformations from the Johnson family of distributions. These procedures are explained in more detail in Appendix F.

3.4.5 Ozturk's Algorithm as Parameter Estimation

The Ozturk algorithm traces a path in a two dimensional space whose endpoint is the statistic. Each distribution, when its shape parameters are specified, if any, results in a definite expected endpoint. These endpoints can be compared with Ozturk's statistic as determined from an observed random sample. Under this theory, the distribution that best fits observed data can be determined based on Ozturk's statistic.

The statistic is calculated from the data and compared with the endpoints from various possible distributions. The endpoint closest to the observed statistic is identified, and the distribution associated with this endpoint is selected as the best fit to the data. Figures 3-8, 3-9, and 3-10 show a "distribution approximation chart" needed for this purpose. In addition to the path of linked vectors determined by the data, the expected endpoints are plotted for several distributions. A single point is plotted when the Probability Density Function for the distribution contains no shape parameters, only a location or scale parameter (Figure 3-8). Distributions with a single shape parameter result in a curve, approximated by a finite number of points along the curve (Figure 3-9). The K distribution in this figure is for a real process, while the Gumbel distribution is of Type II. Distributions with two shape parameters, such as the Beta and Johnson SU distributions in Figure 3-10, are shown by a family of curves. Each curve for the Beta distribution, for instance, corresponds to a different value of one shape parameter. The other shape parameter varies along the curve.

The distance between the data and the expected value of the statistic for a given distribution and shape parameters is the Euclidean distance. A more precise method would be based on probabilities. The Euclidean norm is reasonable if most confidence regions are approximately circular. The closest distribution is then chosen as the desired distribution for approximating the observed data.

3.4.5.1 Estimating Shape Parameters

Shape parameters for the selected distribution can be determined from Ozturk's approximation chart. A distribution with one shape parameter will be represented by a single curve in the approximation chart. The shape parameter can be estimated for such a distribution by finding the point along this curve closest to the value of Ozturk's statistic as calculated from the data. In practice, this curve will not be known exactly. Instead, a finite series of points along the curve will be known. Thus, the shape parameter estimate must be approximated.

Suppose (U_1, V_1) and (U_2, V_2) in Figure 3-11 are points along a curve corresponding to a distribution with one shape parameter γ . Let (U, V) be a value of Ozturk's statistic calculated from the data. The distribution curve is approximated by a straight line between (U_1, V_1) and (U_2, V_2) . The problem is to project (U, V) onto the line at point (U_0, V_0) and determine the corresponding shape parameter for this point.

Ozturk's Statistic

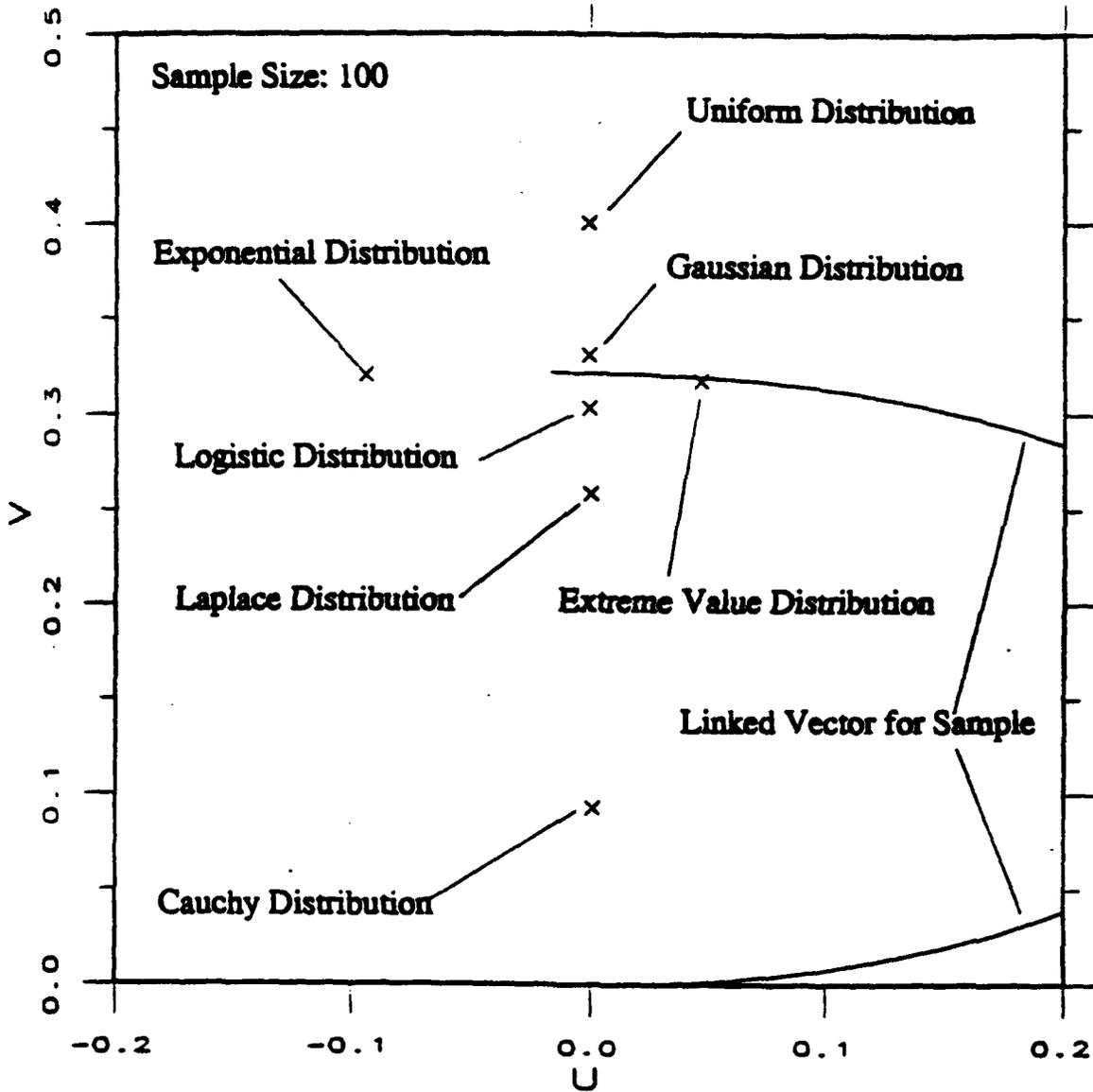


Figure 3-8: Location and Scale Distributions

Ozturk's Statistic

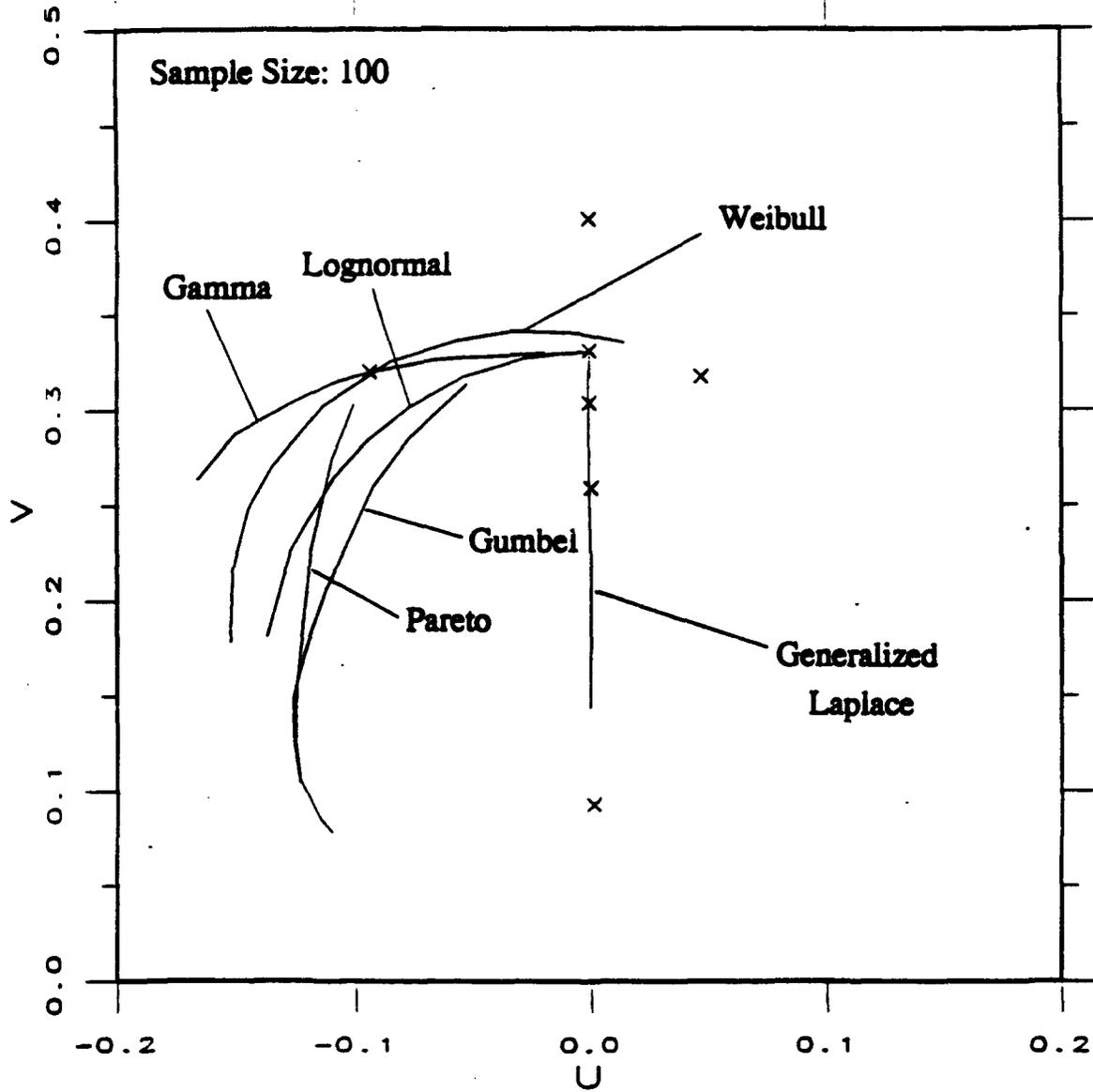


Figure 3-9: Distributions with One Shape Parameter

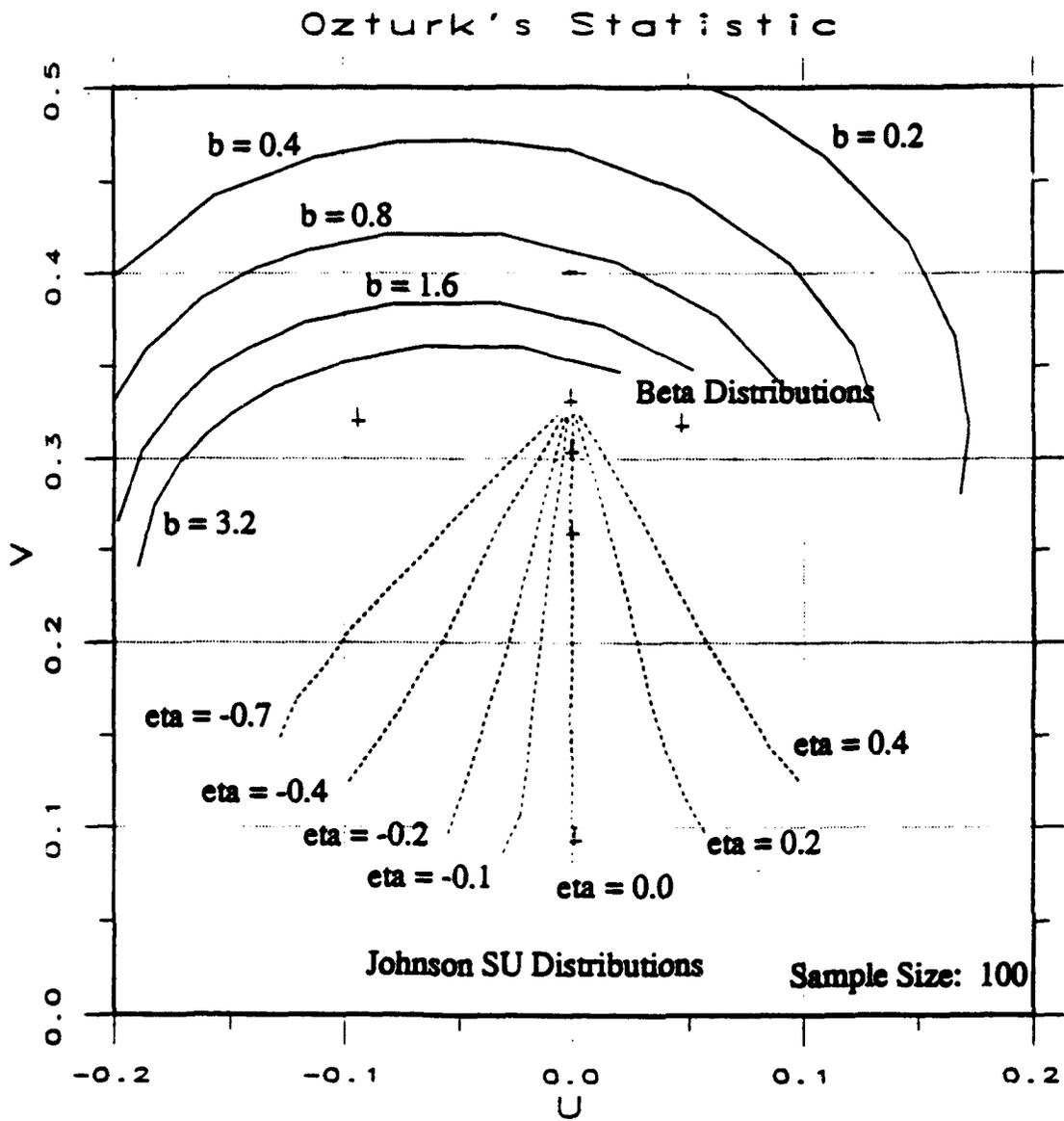


Figure 3-10: Distributions with Two Shape Parameters

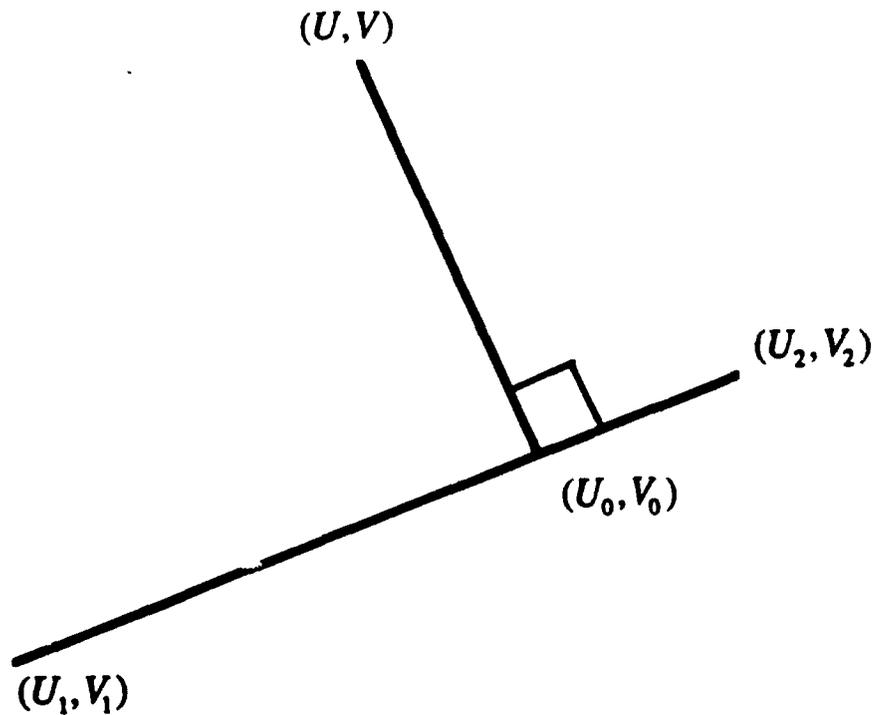


Figure 3-11: Linear Interpolation for Shape Parameter

The slope of the line between (U_1, V_1) and (U_2, V_2) is m :

$$m = \frac{V_2 - V_1}{U_2 - U_1} \quad (3-128)$$

The equations for the two lines whose intersection is (U_0, V_0) are:

$$y = m x + (V_1 - m U_1) \quad (3-129)$$

$$y = \frac{-1}{m} x + (V + \frac{1}{m} U) \quad (3-130)$$

One can substitute (U_0, V_0) into these two equations. The solution of the resulting system of equations is:

$$U_0 = \frac{m^2 U_1 + m (V - V_1) + U}{m^2 + 1} \quad (3-131)$$

$$V_0 = \frac{m^2 V + m (U - U_1) + V_1}{m^2 + 1} \quad (3-132)$$

An estimate of the corresponding shape parameter is then found by linear interpolation between the endpoints:

$$\hat{\gamma} = \frac{U_2 - U_0}{U_2 - U_1} \gamma_1 + \left[1 - \frac{U_2 - U_0}{U_2 - U_1} \right] \gamma_2 \quad (3-133)$$

where γ_1 and γ_2 are the shape parameters corresponding to the endpoints.

Estimating distributions with two shape parameters is more complicated. One method would be to use linear interpolation based on three points around the observed statistic. The method implemented in the MSPSS is simpler. Each curve is treated as a separate distribution. The curve closest to the observed statistic is found. This determines one shape parameter. The other is found by linear interpolation along the selected curve, as above.

3.4.5.2 Estimating Location and Scale Parameters

Ozturk's distribution approximation chart can be used to identify the distribution that most closely matches an observed random sample. It can also be used to determine shape parameters. Estimates of location and scale parameters, however, are not available from the approximation chart. The approach described in (Shah 93) was implemented to estimate location and scale parameters in a manner consistent with Ozturk's algorithm.

Let $x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)}$ be the ordered random sample which is to be used to estimate location and scale parameters. Suppose the underlying Probability Density Function is $f(x, \alpha, \beta)$, where α is the location parameter and β is the scale parameter. Consider the standardized order statistics $S_{(i)}$, $i = 1, 2, \dots, n$:

$$S_{(i)} = \frac{X_{(i)} - \alpha}{\beta} \quad (3-134)$$

Let μ_i be the expected values of these standardized order statistics:

$$\mu_i = E[S_{(i)}] \quad (3-135)$$

In practice, μ_i , $i = 1, 2, \dots, n$ is found by Monte Carlo simulation. The order statistics $S_{(i)}$ are simulated from the given PDF with a location parameter of zero and a scale parameter of unity. The expected value of $X_{(i)}$ is related to the location and scale parameters and the expected values of $S_{(i)}$ as follows:

$$E[X_{(i)}] = \alpha + \mu_i \beta \quad (3-136)$$

Consider the statistics given by Equations 3-137 and 3-138:

$$T_1 = \sum_{i=1}^n \cos(\theta_i) X_{(i)} \quad (3-137)$$

$$T_2 = \sum_{i=1}^n \sin(\theta_i) X_{(i)}, \quad (3-138)$$

where θ_i is as defined in Equation 3-124. These statistics closely resemble Ozturk's statistic. Their expected values are:

$$E[T_1] = a\alpha + b\beta \quad (3-139)$$

$$E[T_2] = c\alpha + d\beta \quad (3-140)$$

where

$$a = \sum_{i=1}^n \text{Cos}(\theta_i) \quad (3-141)$$

$$b = \sum_{i=1}^n \mu_i \text{Cos}(\theta_i) \quad (3-142)$$

$$c = \sum_{i=1}^n \text{Sin}(\theta_i) \quad (3-143)$$

$$d = \sum_{i=1}^n \mu_i \text{Sin}(\theta_i) \quad (3-144)$$

It's fairly apparent by symmetry that a must be equal to zero. Hence estimates of location and scale parameters are:

$$\hat{\beta} = \frac{E[\hat{T}_1]}{b} \quad (3-145)$$

$$\hat{\alpha} = \frac{E[\hat{T}_2] - d \hat{\beta}}{c} \quad (3-146)$$

The expected values of T_1 and T_2 are found by Equations 3-137 and 3-138 from the observed data. As has already been explained μ_i and θ_i are found by Monte-Carlo simulation. With the determination of the location and scale parameters, the Ozturk algorithm for parameter estimation is completely specified.

4. IMPLEMENTATION

Section 3 describes new capabilities implemented under this effort. These capabilities were implemented with several interacting programs under the menu-based subsystem and new analysis sequences under the UFI-based system. Each program under the menu-based system corresponds to a menu choice. Two menu-based programs and four UFI-based analysis sequences were designed to implement the sensor fusion approach. Kalman filtering was implemented with four programs and four analysis sequences. The Extreme Value Theory resulted in the modification of loglikelihood calculation programs and all detection analysis sequences. The Ozturk algorithm was implemented as one program.

4.1 Sensor Fusion

The sensor fusion approach requires the following capabilities:

- Synthesis of multichannel data as ARMA processes where off-diagonal elements of AR coefficient matrices are zero
- Estimation of single channel AR parameters along each channel of multichannel data
- Estimation of the error covariance matrix after tapped delay line structured prediction error filters are applied to each channel of multichannel input
- A sensor fusion filter which applies a tapped delay filter to each channel separately and a second phase at the fusion center for removing interchannel correlation.

Some of these capabilities can be viewed as special cases of functions that existed at the start of this effort. In particular, the previously existing tapped delay line filter has parameters corresponding to an AR model of its input. This filter is functionally-equivalent to the sensor fusion filter when the AR coefficients are diagonal:

$$\hat{A}^H(l) = \begin{bmatrix} \hat{a}_1^*(l) & 0 & 0 & \dots & 0 \\ 0 & \hat{a}_2^*(l) & 0 & & 0 \\ 0 & 0 & \hat{a}_3^*(l) & & . \\ . & . & . & . & . \\ . & . & . & . & 0 \\ 0 & 0 & 0 & \dots & \hat{a}_J^*(l) \end{bmatrix} \quad (4-1)$$

where J is the number of channels. The order of the vector AR process, where each single channel is an AR process, is the maximum of the order of the single channel AR processes.

To take advantage of existing multichannel processing, all files of AR coefficients created by new programs were stored as diagonal matrices. This design decision allowed previous routines for synthesizing ARMA processes, filtering multichannel inputs, displaying AR coefficients, etc. to be available for sensor fusion analysis.

4.1.1 Estimation Program

A menu-based program was written in which AR model parameters are estimated using single channel estimators for each channel in a multichannel process. The input consists of many realizations of a multichannel signal as generated by the multichannel synthesis routines. The output is a set of multichannel AR parameters in which all off-diagonal elements are zero. These parameters, or their average, can be stored in a file suitable for reading by the multichannel tapped delay line filter and other programs in the system. In other

words, the output file is the same format as the output file for the multichannel Yule-Walker estimation program, for example.

On entry, the user is asked for the name of the input file. The user is also asked to input an integer vector in which each element represents the order of the desired AR model for the corresponding channel. The user is asked to choose an algorithm for estimating the AR parameters. Current options consist of the Yule-Walker and Burg algorithms, these being the available algorithms already implemented for single channel AR estimation. If the user wants to use the Yule-Walker algorithm, the user must indicate whether biased or unbiased estimates should be found.

For each realization, the program estimates AR parameters, including the variance, for an AR model for each channel. These estimates are stored in a representation of a multichannel AR process with diagonal matrices for AR coefficients and the covariance matrix.

The method of displaying these estimates for each realization is modeled after the already existing programs for estimating multichannel AR coefficients. The user is given the option of saving the estimates, or their averages over many realizations to a specified file. Variances are calculated and displayed for these averages. No option exists for calculating error variances based on the actual values. Figures 4-1 and 4-2 show an example of user interaction with this program.

4.1.2 Covariance Matrix Estimation

A menu-based program was also written to estimate the covariance matrix. The input is a multichannel signal. This input could be the output of a tapped delay line structured prediction error filter in which the AR coefficients are as determined by the AR estimation program. The matrix used to remove interchannel correlation in the second phase of the filter should be set to identity. By using the menu-based system in this fashion, the structure of the sensor fusion approach is preserved. But the covariance matrix estimation program could be used in other instances as well.

The program estimates the covariance matrix for each realization of the input process, e.g. prediction error filter output. These estimates and their averages across trials are displayed to the user. The estimates are calculated as in Equation 3-27 and 3-28. The user is given the option of saving the average estimates to a file. If the user chooses this option, the user is given the option of storing these covariance estimates with AR parameters read from an already existing file. This further option can produce a coefficient file suitably formatted to serve as the parameters for the sensor fusion filter shown in Figure 3-1. Figure 4-3 shows an example execution of the covariance estimation program.

4.1.3 Sensor Fusion Analysis Sequences

Four new analysis sequences were written to implement sensor fusion in the UFI-based subsystem. Figure 4-4 shows the UFI menu for sensor fusion sequences. The covariance matrix estimation sequence merely generates a user-specified number of trials of white noise and estimates the covariance matrix from those realizations.

The sensor fusion AR parameter estimation sequence is more complicated. A first set of realizations of an AR process are generated. These realizations are used to estimate AR parameters, where each channel is modeled as a single channel AR process. Once the average of these AR parameters is determined, a new set of realizations of the AR process is generated. These realizations are filtered by a tapped delay line. Each channel is filtered

MYWB -- Estimates channel AR parameters by Yule-Walker or Burg algorithm.

Version 1.2

Input file name ? rlv

Title: AR, a = (-0.625, -1.629), (0.25, 0.81)

Output coefficient file name ?

No file name entered, is this okay ? y

Display the estimates on the terminal (y/n) [Y] ?

Do you want to use the Yule-Walker or Burg method (y/b) ?

Enter "y" to use the single-channel Yule-Walker algorithm to estimate AR parameters for each channel. Enter "b" to use the Burg algorithm.

Do you want to use the Yule-Walker or Burg method (y/b) ? b

Enter the order of the AR process for each channel ? 2, 2

Trial 1 size: 20000

Covariance matrix estimate:

(1.0659e+00,0.0000e+00) (0.0000e+00,0.0000e+00)
(0.0000e+00,0.0000e+00) (3.2686e-02,0.0000e+00)

AR coefficient estimates:

A(1)=(-6.2876e-01,4.9529e-03) (0.0000e+00,0.0000e+00)
(0.0000e+00,0.0000e+00) (-1.6330e+00,-3.5726e-03)
A(2)=(2.4834e-01,-9.8383e-04) (0.0000e+00,0.0000e+00)
(0.0000e+00,0.0000e+00) (8.1419e-01,3.7286e-03)

Process Next trial, All trials, or Quit (n/a/q) [N] ? a

Trial 2 size: 20000

Covariance matrix estimate:

(1.0574e+00,0.0000e+00) (0.0000e+00,0.0000e+00)
(0.0000e+00,0.0000e+00) (3.2898e-02,0.0000e+00)

AR coefficient estimates:

A(1)=(-6.2185e-01,4.2076e-03) (0.0000e+00,0.0000e+00)
(0.0000e+00,0.0000e+00) (-1.6302e+00,-1.5750e-04)
A(2)=(2.4799e-01,2.3717e-03) (0.0000e+00,0.0000e+00)
(0.0000e+00,0.0000e+00) (8.0997e-01,5.3056e-04)

Trial 3 size: 20000

Covariance matrix estimate:

(1.0562e+00,0.0000e+00) (0.0000e+00,0.0000e+00)
(0.0000e+00,0.0000e+00) (3.2526e-02,0.0000e+00)

AR coefficient estimates:

A(1)=(-6.3205e-01,9.1982e-03) (0.0000e+00,0.0000e+00)
(0.0000e+00,0.0000e+00) (-1.6264e+00,7.3044e-04)
A(2)=(2.5694e-01,-6.7751e-03) (0.0000e+00,0.0000e+00)
(0.0000e+00,0.0000e+00) (8.0736e-01,6.6720e-04)

Figure 4-1: Sensor Fusion AR Estimation

Trial 4 size: 20000
Covariance matrix estimate:
 (1.0624e+00,0.0000e+00) (0.0000e+00,0.0000e+00)
 (0.0000e+00,0.0000e+00) (3.2659e-02,0.0000e+00)
AR coefficient estimates:
A(1)=(-6.2828×10^{-1} , -3.0305×10^{-3}) (0.0000e+00,0.0000e+00)
 (0.0000e+00,0.0000e+00) (-1.6307×10^0 , 1.4859×10^{-3})
A(2)=(2.5569×10^{-1} , -3.2103×10^{-4}) (0.0000e+00,0.0000e+00)
 (0.0000e+00,0.0000e+00) (8.1168×10^{-1} , -1.2603×10^{-4})
Trial 5 size: 20000
Covariance matrix estimate:
 (1.0618e+00,0.0000e+00) (0.0000e+00,0.0000e+00)
 (0.0000e+00,0.0000e+00) (3.2341e-02,0.0000e+00)
AR coefficient estimates:
A(1)=(-6.2449×10^{-1} , -1.3675×10^{-2}) (0.0000e+00,0.0000e+00)
 (0.0000e+00,0.0000e+00) (-1.6273×10^0 , 8.5958×10^{-4})
A(2)=(2.4287×10^{-1} , 7.6110×10^{-3}) (0.0000e+00,0.0000e+00)
 (0.0000e+00,0.0000e+00) (8.0730×10^{-1} , -2.3590×10^{-3})
Mean values for 5 trials
Mean covariance matrix estimate:
 (1.0607e+00,0.0000e+00) (0.0000e+00,0.0000e+00)
 (0.0000e+00,0.0000e+00) (3.2622e-02,0.0000e+00)
Variance in covariance matrix estimates:
 1.5612e-05 0.0000e+00
 0.0000e+00 4.2464e-08
Mean AR coefficient estimates:
A(1)=(-6.2709×10^{-1} , 3.3064×10^{-4}) (0.0000e+00,0.0000e+00)
 (0.0000e+00,0.0000e+00) (-1.6295×10^0 , -1.3084×10^{-4})
A(2)=(2.5036×10^{-1} , 3.8054×10^{-4}) (0.0000e+00,0.0000e+00)
 (0.0000e+00,0.0000e+00) (8.1010×10^{-1} , 4.8827×10^{-4})
Variance in AR coefficient estimates:
A(1)=9.6361e-05 0.0000e+00
 0.0000e+00 1.1122e-05
A(2)=6.1818e-05 0.0000e+00
 0.0000e+00 1.3396e-05

Figure 4-2: Sensor Fusion AR Estimation (Continued)

```

MVAR -- Estimates multichannel covariance matrix.
Version 1.5
Input file name ? rlv
Title: Complex Gaussian noise, mean = 0, var = (2, 1/2)
Save average covariance estimates to a file (y/n) [Y] ? n
Display the estimates on the terminal (y/n) [Y] ?
Trial 1, Size: 20000 samples.
  Channel 1: Mean=(2.5743e-03,-1.0237e-02)
  Channel 2: Mean=(4.3771e-03,3.3758e-03)
  Covariances: (2.0144e+00,0.0000e+00) (-5.3891e-03,2.2511e-03)
                (-5.3891e-03,-2.2511e-03) (5.0488e-01,0.0000e+00)
Process Next trial, All trials, or Quit (n/a/q) [N] ?
Trial 2, Size: 20000 samples.
  Channel 1: Mean=(-9.7988e-03,2.7998e-03)
  Channel 2: Mean=(6.9658e-03,7.7491e-03)
  Covariances: (1.9965e+00,0.0000e+00) (4.3871e-03,-1.1759e-04)
                (4.3871e-03,1.1759e-04) (5.0255e-01,0.0000e+00)
Process Next trial, All trials, or Quit (n/a/q) [N] ?
Trial 3, Size: 20000 samples.
  Channel 1: Mean=(5.4823e-03,-1.2353e-02)
  Channel 2: Mean=(2.4676e-03,3.3579e-03)
  Covariances: (2.0082e+00,0.0000e+00) (-6.8590e-03,7.9171e-03)
                (-6.8590e-03,-7.9171e-03) (4.9942e-01,0.0000e+00)
Process Next trial, All trials, or Quit (n/a/q) [N] ?
Trial 4, Size: 20000 samples.
  Channel 1: Mean=(2.5110e-03,1.6887e-03)
  Channel 2: Mean=(-3.4951e-03,1.2439e-04)
  Covariances: (2.0102e+00,0.0000e+00) (-3.8123e-04,-3.3313e-03)
                (-3.8123e-04,3.3313e-03) (4.9562e-01,0.0000e+00)
Process Next trial, All trials, or Quit (n/a/q) [N] ?
Trial 5, Size: 20000 samples.
  Channel 1: Mean=(2.6013e-03,-1.1193e-03)
  Channel 2: Mean=(-2.6635e-03,-2.5687e-03)
  Covariances: (2.0029e+00,0.0000e+00) (5.9705e-03,7.5592e-03)
                (5.9705e-03,-7.5592e-03) (5.0278e-01,0.0000e+00)
Process Next trial, All trials, or Quit (n/a/q) [N] ?
Mean values for 5 trials
  Channel 1: Mean=(6.7404e-04,-3.8440e-03) Variance=8.4734e-05
  Channel 2: Mean=(1.5304e-03,2.4077e-03) Variance=3.5430e-05
  Covariances: (2.0065e+00,0.0000e+00) (-4.5434e-04,2.8557e-03)
                (-4.5434e-04,-2.8557e-03) (5.0105e-01,0.0000e+00)
Variances in covariance estimates:
  Variances: 4.7790e-05 5.6330e-05
                5.6330e-05 1.3004e-05

```

Figure 4-3: Estimating the Covariance Matrix

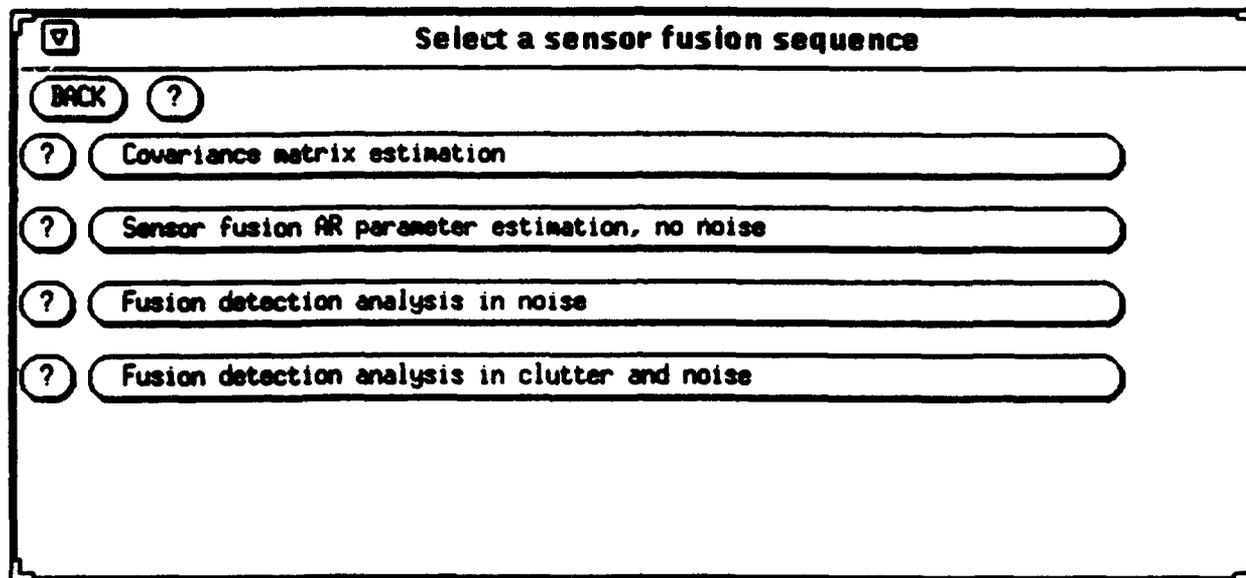


Figure 4-4: Sensor Fusion Analysis Sequences

separately. The resulting multichannel output is then used to estimate the covariance matrix for the signal produced after intertemporal correlation is removed. The final output of this sequence is the average of these covariance matrix estimates.

The sequence for detection analysis in noise is based on the structure shown in Figure 2-1. The F_0 filter merely decorrelates its inputs across channels; it does not contain a tapped delay structure. The covariance matrix used for this filter is estimated based on a number of realizations of white noise. The F_1 filter has the structure shown in Figure 3-1 and its parameters are estimated as in the sensor fusion AR parameter estimation algorithm. The detection analysis proceeds by first determining the thresholds for given false alarm probabilities. This requires the synthesis and filtering of many realizations of noise. The probability of detection is estimated from many realizations of signal and noise. This whole process is repeated a user-specified number of times so as to be able to estimate means and variances for the thresholds and probabilities of detection.

The sequence for detection analysis in clutter and noise is close in structure. In this case, both filters have the structure shown in Figure 3-1, with corresponding coefficient estimation loops. The output of this sequence is also estimates of thresholds and probabilities of detection for given false alarm probabilities.

4.2 Model Identification

Four programs were added to the menu-based subsystem in support of state space model identification. These programs convert AR model parameters to state space model parameters, synthesize state space output processes, estimate state space model parameters from many realizations of the output process, and implement a Kalman filter. Four sequences were added

to the UFI-based subsystem.

4.2.1 Conversion of AR Parameters to State Space Parameters

The program for converting AR parameters to state space parameters allows the user to either specify a file containing Autoregressive model parameters or enter them directly (Figure 4-5). These parameters are the lagged weights in the AR process and the covariance matrix for the driving noise. The input file, if any, is produced by one of the other programs in the system such as the AR process synthesis or Yule-Walker estimation programs.

As described in Section 3.2.1.3, an AR process can be modeled as an innovations representation of a state space model. The program calculates the innovations model parameters F , K , and H^H as described there. The covariance matrix for the innovations process is unchanged. The state space parameters are output to the user and to a user-specified file. The output file is formatted such that it can be read by the state space process synthesis or Kalman filter programs.

```
Converts AR parameters to state space parameters.
Version 1.3
Read AR parameters coefficients from a file (y/n) [Y] ?
Input file name ? rivar
Title: Actual AR coefficients
AR coefficients:
A(1)=(-6.2500e-01,0.0000e+00) (0.0000e+00,0.0000e+00)
      (0.0000e+00,0.0000e+00) (-1.6290e+00,0.0000e+00)
A(2)=(2.5000e-01,0.0000e+00) (0.0000e+00,0.0000e+00)
      (0.0000e+00,0.0000e+00) (8.1000e-01,0.0000e+00)
Driving noise covariance matrix:
      (1.054688e+00,0.000000e+00) (0.000000e+00,0.000000e+00)
      (0.000000e+00,0.000000e+00) (3.267052e-02,0.000000e+00)

Are these parameters okay (y/n) [Y] ?
Output file name for state space parameter ? rivss
Title of this dataset ? State Space parameters
System Dynamics matrix F:
      (6.250000e-01,0.000000e+00) (0.000000e+00,0.000000e+00) (-2.500000e-01,0.000000e+00) (0.000
      (0.000000e+00,0.000000e+00) (1.629000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (-8.099
      (1.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (0.0000
      (0.000000e+00,0.000000e+00) (1.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (0.0000

Hermitian transpose of Observation matrix H:
      (6.250000e-01,0.000000e+00) (0.000000e+00,0.000000e+00) (-2.500000e-01,0.000000e+00) (0.000
      (0.000000e+00,0.000000e+00) (1.629000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (-8.099

Kalman Gain:
      (1.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00)
      (0.000000e+00,0.000000e+00) (1.000000e+00,0.000000e+00)
      (0.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00)
      (0.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00)

Covariance matrix for innovations:
      (1.054688e+00,0.000000e+00) (0.000000e+00,0.000000e+00)
      (0.000000e+00,0.000000e+00) (3.267052e-02,0.000000e+00)
```

Figure 4-5: Converting AR to State Space Parameters

4.2.2 State Space Process Synthesis

The state space process synthesis program (Figure 4-6) asks the user to either enter the model parameters directly or provide them in an input file. These parameters consist of F , K , and H^H in the innovations representation of the state space model. The covariance matrix of the innovations Ω must be specified also. The state space parameters are used directly in process synthesis; to date no "shaping function" approach has been defined or implemented.

```
Synthesizes state space process
Version 1.4
Do you want to set the pseudo random generator seeds (y/n) [N] ?
Enter the number of trials of the signal to be generated ? 5
Read state space parameters from a file (y/n) [Y] ?
Input parameter file name ? rivss
Title: State Space parameters
Decomposition method: Cholesky, LDU, or SVD (c/l/s) [C] ?
Do you want the driving noise to be a Gaussian or SIRP process (g/s) [G] ?
Length of the generated signal in points ? 10000
Enter the number of points to be generated before saving data ? 2000
Driving noise: real only, imaginary only, or complex (r/i/c) ? r
A 2 channel real order 4 state space process will be generated.
System Dynamics matrix F:
(6.250000e-01,0.000000e+00) (0.000000e+00,0.000000e+00) (-2.500000e-01,0.000000e+00) (0.000
(0.000000e+00,0.000000e+00) (1.629000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (-8.099
(1.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (0.0000
(0.000000e+00,0.000000e+00) (1.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (0.0000

Hermitian transpose of Observation matrix H:
(6.250000e-01,0.000000e+00) (0.000000e+00,0.000000e+00) (-2.500000e-01,0.000000e+00) (0.000
(0.000000e+00,0.000000e+00) (1.629000e+00,0.000000e+00) (0.000000e+00,0.000000e+00) (-8.099

Kalman Gain:
(1.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00)
(0.000000e+00,0.000000e+00) (1.000000e+00,0.000000e+00)
(0.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00)
(0.000000e+00,0.000000e+00) (0.000000e+00,0.000000e+00)

Covariance matrix for innovations:
(1.054688e+00,0.000000e+00) (0.000000e+00,0.000000e+00)
(0.000000e+00,0.000000e+00) (3.267052e-02,0.000000e+00)

Cholesky decomposition of covariance matrix:
(1.026980e+00,0.000000e+00) (0.000000e+00,0.000000e+00)
(0.000000e+00,0.000000e+00) (1.807499e-01,0.000000e+00)

The driving noise will be Gaussian.
10000 time samples will be generated, with 2000 time samples priming the process.
Are these parameters okay (y/n) [Y] ?
Output file name ? riv
Title of this dataset ? Synthesized state space process
```

Figure 4-6: State Space Synthesis

The user can specify the covariance matrix directly or in factored form. Factorization options consist of the Cholesky, L-D-U, and SVD. The program does not provide any checks on the factors the user enters. For example, the program does not verify whether the eigenvector matrix provided as an SVD factor is Hermitian. If the user specifies the covariance matrix directly, the user is asked which decomposition should be used in process synthesis.

The user can choose to specify Gaussian or K-distributed Spherically Invariant Random Process (SIRP) driving noise. The user must also specify the random number seed, the number of time samples in a single realization of the output process, the number of realizations (trials) of the process, and the number of time samples to first generate and then discard in "priming" each realization. This last input allows the user to produce steady state output processes by discarding transient behavior.

The program generates the specified realizations of the process and saves them in a user-specified output file. The output file can be read by many other programs in the system, including plotting, statistics, and estimation routines. The innovations are generated as

$$\varepsilon(n) = T z(n), \quad (4-2)$$

where $z(n)$ is either Gaussian or SIRP white noise uncorrelated across channels with (diagonal) covariance matrix D . If the user chooses a Cholesky decomposition, D is the identity matrix and T is the lower diagonal matrix C where

$$\Omega = C C^H. \quad (4-3)$$

If the user chooses a L-D-U decomposition, T is the lower diagonal matrix L with ones along the main diagonal and where

$$\Omega = L D L^H. \quad (4-4)$$

If the user selects the SVD, D is the matrix of eigenvalues Λ and T is Q where

$$\Omega = Q \Lambda Q^H. \quad (4-5)$$

If the user enters the model parameters directly, the user is given the option of specifying a file in which they are saved. This option permits the user to resynthesize a particular case; it is also useful for the innovations filter program.

4.2.3 Estimation of State Space Parameters

In the program (Figure 4-7) for estimating the parameters of the state space model, the user is asked to specify:

- The upper bound, L , on the size of the block state space vector
- Whether biased or unbiased estimates of correlations should be used
- The model order, if the user wants to input it directly
- The method to be used for calculating the model order
- The bound for calculating the model order, if the user wants the model order to be internally calculated
- Whether diagnostic information should be displayed on the user's terminal
- The file containing the process from which the parameters are to be estimated

- The output file in which the parameter estimates are to be stored.

```

SSC - Estimates state space model parameters using the Scientific Studies Corporation algorithm
Version 1.8
Input file name ? rlv
Title: Synthesized state space process
Output file name for parameter estimates ?
No file name entered, is this okay ? y
Upper bound, L, on size of state space block vector ? 4
Calculate biased or unbiased lags (u/b) ? u
MODEL ORDER MENU
0 - Enter the model order directly
1 - Use the canonical correlations to calculate the model order
2 - Use the normalized running sum of the canonical correlations
3 - Use the squares of the canonical correlations
4 - Use the normalized running sum of squares
5 - Use the log parameters
6 - Use the normalized mutual information parameters
Model order calculation method ? 0
Model order ? 2
Display diagnostic information (y/n) ? n
Order: 2
System Dynamics matrix F:
(7.677540e-03,0.000000e+00) (-3.436075e-01,0.000000e+00)
(5.054108e-02,0.000000e+00) (-1.029191e-01,0.000000e+00)

Hermitian transpose of Observation matrix H:
(8.399569e-02,0.000000e+00) (7.290421e-01,0.000000e+00)
(1.476353e-02,0.000000e+00) (-2.566631e-01,0.000000e+00)

Kalman Gain:
(-3.226846e-04,0.000000e+00) (1.359702e+00,0.000000e+00)
(1.319421e-02,0.000000e+00) (1.833456e-01,0.000000e+00)

Covariance matrix for innovations:
(1.160445e+00,0.000000e+00) (1.323281e-01,0.000000e+00)
(1.323281e-01,0.000000e+00) (4.512954e-01,0.000000e+00)

```

Figure 4-7: State Space Model Identification

The program averages the estimates of the lagged correlations over all trials to generate the needed estimates $\hat{\Lambda}_l$. Using this information, the program estimates the state space model parameters for an innovations representation using the algorithm in Section 3.2.3. Specifically, the program generates estimates of the system dynamics matrix F , the Hermitian transpose of the observation matrix H , the Kalman gain K , and the innovations covariance matrix Ω . The model order and these estimates are echoed to the user's terminal. They are also saved to a file in a format readable by the innovations filter program.

If the user indicates diagnostic information should be displayed, the following additional information is displayed:

- The singular values of the past correlation matrix, $\hat{R}_{P:L,L}$
- The singular values of the future correlation matrix, $\hat{R}_{F:L,L}$
- The canonical correlations, their normalized running sum, the squares of the canonical correlations, the normalized running sum of the squares, the log parameters, and the mutual information parameters (Only the series used in calculating the order is printed if the user indicates diagnostic information should not be printed)
- The singular values of the system dynamics matrix F

4.2.4 Innovations Filter Program

In the Kalman filter implementation (Figure 4-8), the user is required to specify the name of a file containing the input. This file contains a number of trials of the process in the customary format for multichannel processes in this system. The program also asks the user for two output files. One file, which is optional, is for storing the estimates of the state variables. The other file contains the decorrelated innovations filter output.

The user is given the option of reading the filter parameters F , K , H^H , and Ω from a file or inputting them directly. The user is asked for two additional parameters. One is the length of the innovations sequence N_1 , which must be less than or equal to the number of time samples in the input process N . The program discards the first $N - N_1$ time samples of the innovations to remove transient behavior.

In order to whiten the innovations vector across channels, the covariance matrix Ω is factored. Factorization options consist of Cholesky, LDU, and Singular Value Decompositions. The values of the resulting channel variances, σ_j^2 , are stored in a user-supplied file for later use in calculating a loglikelihood ratio statistic. Given the user inputs, the program calculates the innovations process by the Kalman filter as described in Figure 3-6. A user-specified number of time samples of the innovations process are discarded. The final output of the filter is decorrelated across channels as above.

4.2.5 Kalman Filter Analysis Sequences

Four new analysis sequences were written to implement the state space model and Kalman filter under the UFI-based subsystem. Figure 4-9 shows the UFI menu for state space sequences. The sequence for estimating state space model parameters for a signal generates a number of realizations of an AR process. These realizations are used to estimate state space model parameters. These estimates are the final output of this analysis sequence.

The sequence for estimating state space model parameters for the sum of signal and noise is very similar. In this case, however, the estimates are based on the sum of an AR process and white noise.

The two detection analysis sequences closely resemble one another. They implement the signal detection algorithm illustrated in Figure 2-1. Both filters F_0 and F_1 are Kalman filters. The parameters of each of these filters are estimated before determining thresholds, given false alarm probabilities, or probabilities of detection, given threshold. Like all other detection analysis sequences in the UFI-based subsystem, this sequence encloses the whole process in an outer loop. Thus, the final output includes means and variances for thresholds and probabilities of detection.

```

MKAF -- Calculate the innovations process using the Kalman filter
Version 1.4
Input file name ? rlv
Title: Synthesized state space process
File for decorrelated innovations output ? rlv1
Title of this dataset ? Innovations
File for output of the state variables, if desired ?
No file name entered, is this okay ? y
File for output of variances of the decorrelated innovations ?
No file name entered, is this okay ? y
Enter the index of the first output value ? 9971
Do you want a Cholesky, LDU, or SVD (c/l/s) covariance decomposition [C] ?
Input the Kalman filter parameters directly or from a file (d/f) [D] ? f
Parameter file name (Kalman filter matrices) ? rlv2
Title: Estimated state space parameters
A 2 channel order 2 state space process will be filtered.
System Dynamics matrix F:
(7.677540e-03,0.000000e+00) (-3.436075e-01,0.000000e+00)
(5.054108e-02,0.000000e+00) (-1.029191e-01,0.000000e+00)

Hermitian transpose of Observation matrix H:
(8.399569e-02,0.000000e+00) (7.290421e-01,0.000000e+00)
(1.476353e-02,0.000000e+00) (-2.566631e-01,0.000000e+00)

Kalman Gain:
(-3.226846e-04,0.000000e+00) (1.359702e+00,0.000000e+00)
(1.319421e-02,0.000000e+00) (1.833456e-01,0.000000e+00)

Covariance matrix for innovations:
(1.160445e+00,0.000000e+00) (1.323281e-01,0.000000e+00)
(1.323281e-01,0.000000e+00) (4.512954e-01,0.000000e+00)

Cholesky decomposition of covariance matrix:
(1.077240e+00,0.000000e+00) (0.000000e+00,0.000000e+00)
(1.228400e-01,0.000000e+00) (6.604587e-01,0.000000e+00)

Inverse of Cholesky decomposition:
(9.282985e-01,0.000000e+00) (0.000000e+00,0.000000e+00)
(-1.726560e-01,0.000000e+00) (1.514099e+00,0.000000e+00)

The first 9970 time samples will be discarded.
Length of innovations sequence was 30

```

Figure 4-8: Kalman Filter

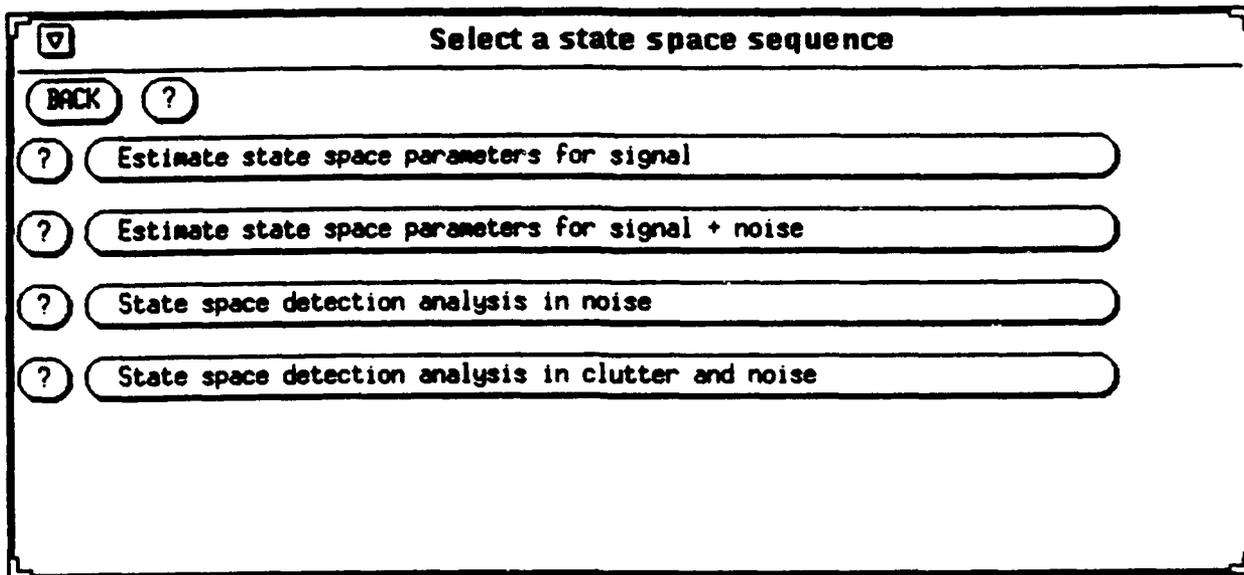


Figure 4-9: State Space Analysis Sequences

4.3 Extreme Value Theory

Implementing the Extreme Value Theory (EVT) did not require writing any new menu-based programs or UFI sequences. Instead the programs implementing loglikelihood ratios were modified, along with the sequences that perform detection analysis with "a priori" estimates. Only two programs calculate loglikelihood ratios in the menu-based system. One implements the SIRP and Gaussian multichannel loglikelihood ratio statistics. The other implements the loglikelihood ratio for the unconstrained quadrature case. The detection analysis sequences were modified to include an option for the EVT.

Three preliminary modifications were made to the two loglikelihood ratio programs in the menu-based system before adding the EVT. When this task began, these programs did not calculate a threshold for a given false alarm probability. Nor did they calculate a detection probability. Rather, in the threshold estimation mode, the user was asked for an index into a sorted list of thresholds. In the detection probability estimation mode, the programs reported the number of loglikelihood values above a given threshold and the total number of loglikelihood values estimated. Finally, the user was originally required to rerun these programs if detection probabilities were desired for different thresholds, even if these estimates were based on the same sample of loglikelihood ratios.

These programs now report a detection probability. Let $X_1 \leq X_2 \leq \dots \leq X_n$ be a sample of order statistics for the relevant loglikelihood ratio. For estimating the detection probability, these estimates should be generated under the alternative hypothesis in which a signal is present. Let η be the user defined threshold. Let j be the smallest index such that for all $k \geq j$, $X_k > \eta$. Consequently, $n - j + 1$ is the number of loglikelihood values strictly greater

than the given threshold. Previously, the programs reported $n - j + 1$ and n to the user. Now, they also report $(n - j + 1)/n$, which is an estimate of the probability of detection.

The programs also now estimate a threshold for a given false alarm probability. The linear interpolation procedure incorporated into the menu-based programs is the same as that previously implemented in the UFI-based system. In this case, let $X_1 \leq X_2 \leq \dots \leq X_n$ be a sample of order statistics for the relevant loglikelihood ratio generated under the null hypothesis without a signal. Let F be the underlying cumulative probability distribution, and let p denote the desired false alarm probability. The program must find an estimate, $\hat{\eta}$ such that

$$F(\eta) = 1 - p. \quad (4-6)$$

where $\hat{\eta}$ is an estimate of η .

Since F is the distribution function for a continuous probability distribution,

$$1 - F(\eta) = Pr(X > \eta) = Pr(X \geq \eta). \quad (4-7)$$

Ignoring ties, the number of sample values greater than or equal to X_{j+1} is $n - j$. Thus, for any $j + 1$, a reasonable estimate of $F(X_{j+1})$ is

$$\hat{F}(X_{j+1}) = 1 - \frac{n - j}{n} = \frac{j}{n}. \quad (4-8)$$

or,

$$j = n \hat{F}(X_{j+1}). \quad (4-9)$$

If $n(1 - p)$ is an integer, X_{j+1} seems a reasonable value for $\hat{\eta}$. In general, this value cannot be assured of being an integer. Therefore, let $n(1 - p) = j + g$ where j is the integer part of $n(1 - p)$ and $0 \leq g < 1$. The estimate of the threshold is

$$\hat{\eta} = (1 - g)X_{j+1} + gX_{j+2}. \quad (4-10)$$

This is the estimate currently used in the MSPSS. The above argument is only a plausibility argument, not a formal derivation. Equation 10.4 in (Rangaswamy 93) gives a slightly different estimate which is asymptotically equivalent.

The two loglikelihood ratio programs now have an additional loop. Previously, if the user desired to calculate thresholds for different indices, or detection probabilities for different thresholds, the user was required to rerun the program. Now, after reporting the desired result, the program asks if the user would like to estimate another threshold or estimate another detection probability, whichever is appropriate. This option is repeated until the user indicates no more estimates are desired. Additional estimates are found based on the previously calculated sample of loglikelihood ratios. Since most of the computational time is expended in these programs in calculating this sample, additional estimates are reported very quickly.

5. REFERENCES

(Al-Ibrahim 91) Mohammad M. Al-Ibrahim and Pramod K. Varshney, *On Distributed Sequential Hypothesis Testing*, RL-TR-91-96, Rome Laboratory, June 1991.

(Chair 86) Z. Chair and P. K. Varshney, "Optimum Data Fusion in Multiple Sensor Detection Systems," *IEEE Transactions on Aerospace and Electronic Systems*, Volume AES-22, pp. 98-101, January 1986.

(Chakravarthi 92) Prakosh R. Chakravarthi, "On Determining the Radar Threshold for Non-Gaussian Processes from Experimental Data," Ph. D. Thesis, Syracuse University, in progress.

(Chan 83) T. F. Chan, G. H. Golub, and R. J. LeVeque, "Algorithms for Computing the Sample Variance: Analysis and Recommendations," *The American Statistician*, Volume 37, Number 3, p. 242-247, August 1983.

(Fishman 73) George S. Fishman, *Concepts and Methods in Discrete Event Digital Simulation*, John Wiley, 1973.

(Harmon 88) M. G. Harmon, T. P. Baker, "An Ada Implementation of Marsaglia's Universal Random Number Generator", *Ada Letters*, Vol VIII, No 2, pp. 110-112, March/April 1988.

(Hoballah 89) I. Y. Hoballah and P. K. Varshney, "Distributed Bayesian Signal Detection," *IEEE Transactions on Information Theory*, Volume IT-35, pp. 995-1001, September 1989.

(Hogg 78) Robert V. Hogg and Allen T. Craig, *Introduction to Mathematical Statistics*, Fourth Edition, Macmillan, 1978.

(Kaman 91a) Kaman Sciences Corporation, *Man Machine Interface Experiment*, January 23, 1991.

(Kaman 91b) Kaman Sciences Corporation, *User Front-End Interface Knowledge Base (UFIKB) Builders' Guide*, February 1991.

(Kaman 92a) Kaman Sciences Corporation, *Software User's Manual for the Multichannel Signal Processing Simulation System*, May 30, 1992.

(Kaman 92b) Kaman Sciences Corporation, *Enhancement of Multichannel Signal Processing Simulation System*, December 3, 1992.

(Marple 87) S. Lawrence Marple Jr., *Digital Spectral Analysis*, Prentice-Hall, Inc., 1987.

(Michels 89) James H. Michels, *A Parametric Detection Approach Using Multichannel Processes*, Rome Air Development Center, RADC-TR-89-306, November 1989.

(Michels 90a) James H. Michels, *Synthesis of Multichannel Autoregressive Random Processes and Ergodicity Considerations*, Rome Air Development Center, RADC-TR-90-211, July 1990.

(Michels 90b) James H. Michels, *Multichannel Linear Prediction and Its Association with Triangular Matrix Decomposition*, Rome Air Development Center, RADC-TR-90-226, November 1990.

(Michels 91) James H. Michels, *Multichannel Detection Using the Discrete-Time Model-Based Innovations Approach*, Rome Laboratory, RL-TR-91-269, August 1991.

(Michels 92a) James H. Michels, *Multichannel Detection of Partially Correlated Signals in Clutter*, Rome Laboratory, RL-TR-92-332, December 1992.

(Michels 92b) James H. Michels, *Considerations of the Error Variances of Time-Averaged Estimators for Correlated Processes*, Rome Laboratory, RL-TR-92-339, December 1992.

(Ozturk 90a) A. Ozturk and E. J. Dudewicz, *A New Statistical Goodness-of-Fit Test Based on Graphical Representation*, Technical Report no. 52, Department of Mathematics, Syracuse University, 1990.

(Ozturk 90b) A. Ozturk, *A General Algorithm for Univariate and Multivariate Goodness-of-Fit Tests Based on Graphical Representation*, Technical Report no. 53, Department of Mathematics, Syracuse University, 1990.

(Ozturk 90c) A. Ozturk, *A New Method for Univariate and Multivariate Distribution Identification*, Syracuse University, 1990.

(Press 86) William H. Press et. al., *Numerical Recipes*, Cambridge University Press, 1986.

(Rangaswamy 91) M. Rangaswamy, D. D. Weiner, and A. Ozturk, "Simulation of Correlated Non-Gaussian Interference for Radar Signal Detection," *Proceedings of Twentyfifth Asilomar Conference on Signals and Computers*, Pacific Grove, CA, 1991.

(Rangaswamy 92) M. Rangaswamy, D. D. Weiner, and J. H. Michels, "Innovations Based Detection Algorithm for Correlated Non-Gaussian Random Process," , 1992.

(Rangaswamy 93) M. Rangaswamy et. al., *Signal Detection in Correlated Gaussian and Non-Gaussian Radar Clutter*, RL-TR-93-79, May 1993.

(Robbins 89) Thomas R. Robbins, *UFP Programmer Manual*, Kaman Sciences Corporation, November, 1989.

(Roman 93) Jaime R. Roman and Dennis W. Davis, *State-Space Models for Multichannel Detection*, RL-TR-93-146, Rome Laboratory, July 1993.

(Romeu 90) Jorge L. Romeu, *Development and Evaluation of a General Procedure for Assessing Multivariate Normality*, CASE Center Technical Report No. 9022, October 1990.

(Romeu 92) Jorge L. Romeu, *Monte Carlo Validation of A Theoretical Model for the Generation of Non-Gaussian Radar Clutter*, Unpublished technical report, 1992.

(Shah 93) Rajiv R. Shah, *A New Technique for Distribution Approximation of Random Data*, Master's Thesis, Syracuse University, December, 1993.

(Slaski 93) Lisa K. Slaski and Murali Rangaswamy, *A New Efficient Algorithm for PDF Approximation* (Draft), September 21, 1993.

(Vienneau 93) Robert Vienneau, David Dekkers, and Sue Eilers, *Generalized Multichannel Signal Detection*, RL-TR-93-154, Rome Laboratory, July 1993.

Appendix A. Notation

A	An intermediate matrix in the Canonical Correlations Matrix.
$A^H(k)$	AR weights; each AR coefficient is a matrix.
$A_c^H(k)$	Matrix AR coefficients for the clutter.
C	The Cholesky decomposition of Σ .
C_w	The Cholesky decomposition of Σ_w .
D	(1) The diagonal matrix in the LDU decomposition of Σ . (2) A matrix in the state space model.
D_w	The diagonal matrix in the LDU decomposition of Σ_w .
F, F_β	The system dynamics matrix in the state space model.
F_0, F_0	(1) The filter corresponding to the null hypothesis. (2) A Cumulative Distribution Function.
F_1, F_1	(1) The filter corresponding to the alternative hypothesis. (2) A Cumulative Distribution Function.
G	A matrix in the state space model.
H, H_β	The observation matrix in the state space model.
$H_{L,L}$	An estimate of the stochastic block Hankel matrix in the Canonical Correlations Algorithm.
$\bar{H}_{L,L}$	The column shifted block Hankel matrix in the Canonical Correlations Algorithm.
H_0	The null hypothesis.
H_1	The alternative hypothesis.
J	The number of channels.
K, K_β	The Kalman gain matrix in the innovations representation of the state space model.
K_c	The amplitude matrix for the clutter.
L	(1) The lower triangular matrix in the LDU decomposition of Σ . (2) The number of lags used in estimating correlation matrices in the Canonical Correlations Algorithm.
L_w	The lower triangular matrix in the LDU decomposition of Σ_w .
M	The dimension of the state vector in the state space model.
M_k	A sample mean for a sample size of k items.
N	The number of time samples in a realization of a stochastic process.
N_R	(1) The number of realizations used in estimating state space parameters in the Canonical Correlations Algorithm. (2) The number of realizations of the loglikelihood statistic in the Extreme Value Theory.

P	The order of an AR process.
Q	The matrix of right hand eigenvectors in the Singular Value Decomposition of Σ .
Q_N	Ozturk's statistic.
Q_w	The matrix of right hand eigenvectors in the Singular Value Decomposition of Σ_w .
R	The block correlation matrix for a radar return.
$R(l)$	The correlation matrix showing cross channel correlations for a radar return at lag l .
$R_c(l)$	The correlation matrix showing cross channel correlations for the clutter at lag l .
$\hat{R}_{F:L,L}$	An estimate of the future block correlation matrix in the Canonical Correlations Algorithm.
$\hat{R}_{P:L,L}$	An estimate of the past block correlation matrix in the Canonical Correlations Algorithm.
S	A random variable used in generating SIRPs.
S^2	The sample variance.
S_A	The diagonal matrix of eigenvalues in the Singular Value Decomposition of the matrix A in the Canonical Correlations Algorithm.
$S_{A,l}$	A square matrix in the Canonical Correlations Algorithm formed by taking the upper M rows and M columns of S_A .
S_F	The diagonal matrix of eigenvalues in the Singular Value Decomposition of the estimate of the future block correlation matrix in the Canonical Correlations Algorithm.
S_P	The diagonal matrix of eigenvalues in the Singular Value Decomposition of the estimate of the past block correlation matrix in the Canonical Correlations Algorithm.
S_k^2	The sample variance for a sample size of k items.
T	(1) Either C , L , or Q in the Cholesky, LDU, or Singular Value Decomposition of Σ . (2) A matrix representing a basis transformation of the state space in the innovations representation of the state space model.
T_1	A statistic used in estimating location and scale parameters in the Ozturk algorithm.
T_2	A statistic used in estimating location and scale parameters in the Ozturk algorithm.
T_F	A transformation matrix in the Canonical Correlations Algorithm.
T_P	A transformation matrix in the Canonical Correlations Algorithm.
T_c	The sampling period for the clutter.
T_w	Either C_w , L_w , or Q_w in the corresponding decomposition of Σ_w .

U_A	The matrix of right hand eigenvectors in the Singular Value Decomposition of the matrix A in the Canonical Correlations Algorithm.
U_N	A component of Ozturk's statistic, the ordered pair (U_N, V_N) .
U_F	The matrix of right hand eigenvectors in the Singular Value Decomposition of the estimate of the future block correlation matrix in the Canonical Correlations Algorithm.
U_P	The matrix of right hand eigenvectors in the Singular Value Decomposition of the estimate of the past block correlation matrix in the Canonical Correlations Algorithm.
V_A^H	The matrix of left hand eigenvectors in the Singular Value Decomposition of the matrix A in the Canonical Correlations Algorithm.
V_N	A component of Ozturk's statistic, the ordered pair (U_N, V_N) .
X^*	The complex conjugate of X .
X^{-1}	The inverse of the matrix X .
X^H	The Hermitian transpose of the matrix X .
Z	The upper square matrix in the Canonical Correlations Algorithm consisting of the appreciably non-zero entries in the matrix $\bar{\Delta}$.
Z^{-1}	A one-element lag in a system block diagram.
Z_H	A matrix in the Canonical Correlations Algorithm formed from the first J rows and M columns of the matrix $H_{L,L} T^{p^H}$.
$Z_{(i)}$	A tail value of the ordered loglikelihood statistics.
Z_T	A matrix in the Canonical Correlations Algorithm formed from the first M rows and J columns of the matrix $T_F H_{L,L}$.
$a_j^*(k)$	A scalar AR weight.
$\hat{a}_j^*(k)$	An estimate of an AR coefficient.
b	A parameter used in generating K-distributed SIRPs.
$c, c(n)$	Multichannel clutter (e.g. a vector AR process).
k	An index over the AR weights.
l_c	The lag value at which the shaping function for the clutter peaks.
m	The number of loglikelihood statistics in the tail.
$m_{(i)}$	In Ozturk's algorithm, the expected value of an order statistic from the standard reference distribution.
n	(1) Noise (e.g. Gaussian white noise correlated across channels, but not in time). (2) An index for time in a vector stochastic process. (3) The sample size.
p	A false alarm probability.
q	A quadratic form that is a sufficient statistic for a Spherically Invariant Random Process (SIRP).

q^i	The quadratic form q as calculated for the i th realization of the stochastic process modeling a SIRP.
$s, s(n)$	A multichannel signal (e.g. a vector AR process).
s_x	A sample standard deviation.
$u(n)$	The input process in the state space model.
$w(n)$	(1) Noise (e.g. Gaussian white noise correlated across channels, but not in time). (2) Measurement noise in the state space model.
$w_{(n)}$	In Ozturk's algorithm, an order statistic from the standard reference distribution.
$x, \underline{x}(n), x(n)$	A multichannel radar return (vector stochastic process).
$x^i(n)$	The n th time sample from the i th realization of the stochastic process $x(n)$.
$\hat{x}(n n-1)$	An estimate of a multichannel radar return.
x_i	An element of a random sample.
$x_{(i)}$	An element of an ordered random sample, an order statistic.
\bar{x}	A sample mean.
$x_j(n)$	A channel in the multichannel radar return $x(n)$.
$y(n)$	The state vector process in the state space model.
$y_{(i)}$	In Ozturk's algorithm, the standardized value of an order statistic.
$z_{(n)}$	An order statistic generated in Ozturk's algorithm by Monte Carlo simulation of the null distribution.
$\bar{\Delta}$	A matrix in the Canonical Correlations Algorithm.
Λ	(1) A loglikelihood statistic. (2) The diagonal matrix of eigenvalues in the Singular Value Decomposition of Σ .
$\Lambda^{(i)}$	The i th order statistic in a sample of loglikelihood statistics.
Λ_l	The correlation matrix at lag l ; notation used in the Canonical Correlations Algorithm.
$\hat{\Lambda}_l$	An estimate of the correlation matrix at lag l .
Λ_w	The diagonal matrix of eigenvalues in the Singular Value Decomposition of Σ_w .
Π	The (zeroth lag) correlation matrix among the state space vectors.
Σ	(1) The covariance matrix for $\varepsilon(n)$. (2) The covariance matrix for $x(n)$.
$\hat{\Sigma}$	An estimate of the covariance matrix.
Σ_c	The covariance matrix for the driving noise process in an AR model of the clutter.
Σ_w	The covariance matrix for the white noise process $w(n)$.
Ω	The (zeroth lag) correlation matrix for the multichannel innovations process in the innovations representation of the state space model.

α	(1) A probability defining the tail in the Extreme Value Theory. (2) The significance level of a statistical test. (3) The location parameter of the distribution to be estimated by Ozturk's algorithm. (4) A parameter used in generating K distributed SIRPs.
α_j	The significance level of a statistical test.
$\alpha(n), \underline{\alpha}(n)$	The state vector in the innovations representation of the state space model.
$\hat{\alpha}(n n-1)$	An estimate of the state space vector in the innovations representation of the state space model.
β	(1) The scale parameter of the distribution to be estimated by Ozturk's algorithm. (2) A parameter of the Gamma probability distribution.
$\beta(n)$	The state vector after a basis transformation of the innovations representation of the state space model.
γ	(1) A parameter in the Generalized Pareto distribution. (2) A confidence level.
$\hat{\gamma}$	An estimate of a parameter in the Generalized Pareto distribution.
ε_0	A probability weighted moment used in the Extreme Value Theory.
$\hat{\varepsilon}_0$	An estimate of ε_0 .
ε_1	A probability weighted moment used in the Extreme Value Theory.
$\hat{\varepsilon}_1$	An estimate of ε_1 .
$\varepsilon(n), \underline{\varepsilon}(n)$	A driving noise term, uncorrelated across time but possibly correlated across channels.
$\bar{\varepsilon}$	The mean over time samples of process $\varepsilon(n)$.
$\varepsilon_c(n)$	The driving noise term for the clutter.
η	A threshold for a given false alarm probability.
$\hat{\eta}$	An estimate of η .
θ_i	The angle with the abscissa of linked vectors in Ozturk's algorithm.
λ	A loglikelihood statistic.
λ_0	The loglikelihood statistic defining the tail in the Extreme Value Theory.
λ_c	The one-lag temporal correlation (matrix) parameter for the clutter.
μ_i	Expected values of certain order statistics in Ozturk's algorithm.
$v(n), \underline{v}(n)$	Innovations, uncorrelated both in time and across channels.
$\underline{v}_0(n)$	The innovations, uncorrelated both in time and across channels, for the null hypothesis model.
$\underline{v}_1(n)$	The innovations, uncorrelated both in time and across channels, for the alternative hypothesis model.
$v_w(n)$	A zero-mean normally distributed random vector, uncorrelated both in time and across channels; used in generating Gaussian white noise $w(n)$.
ρ_j	The jth canonical correlation, sorted in decreasing order, in the Canonical Correlations Algorithm.

σ	A parameter in the Generalized Pareto distribution.
$\hat{\sigma}$	An estimate of σ .
σ^2	A variance.
$\hat{\sigma}_j^2$	An estimate of a channel variance.
τ	A parameter in a reparameterization of the Generalized Pareto distribution.
$\hat{\tau}$	An estimate of τ .
ϕ_c	The Doppler shift for the clutter.
$F(\lambda)$	The cumulative probability distribution function for the loglikelihood statistic under the null distribution.
$\hat{F}(\lambda)$	An approximation of the cumulative distribution function for the loglikelihood statistic under the null distribution.
$G(\lambda)$	The cumulative distribution function for the Generalized Pareto distribution.
$Q_r(\gamma)$	A function used in calculating Ordered Samples Least Squares estimates of the Generalized Pareto distribution.
$f_s(s)$	The probability density function for the random variable S .
$f_v(v)$	The probability density function for the random variable V .
$f_w(w)$	The probability density function for the random variable W .
$\Gamma(x)$	The Gamma function.
$\Phi(w)$	The Cumulative Distribution Function for the reference distribution in Ozturk's algorithm.

Appendix B.

A Single-Channel MiniSystem

During a previous effort (Vienneau 93), a version of the menu-based system was delivered to Dr. Jorge Romeu. This minisystem contained capabilities for synthesizing single-channel processes and saving them to a file. A new version of this minisystem was produced under this effort. This new version of the minisystem is distinguished from the original by the addition of a capability to calculate a quadratic form useful in assessing the performance of SIRP synthesis routines.

The system is delivered on a tar tape containing source code and object files for certain library routines. It is intended to be read on a Sun computer. Object code for the libraries is provided for both the Sun 3 and Sun 4 architectures. Reading the tape creates a subdirectory called "schan" under the user's current directory. This subdirectory contains the files described in Table A-1.

To run this system, there must be an environment variable called ARCH already defined in the user's environment. Currently the Sun 3 and Sun 4 architectures are supported and ARCH must be set to one of these two values. Also, the software requires access to the Unix `f77` and `make` programs.

The system is executed by first changing the working directory to "schan." Then "schan [return]" is entered at the Unix prompt to invoke the Single Channel Process Synthesis Menu. The menu contains the following options:

- Generate K-distributed SIRP
- Generate Gaussian noise
- Generate single channel AR process with SIRP or Gaussian driving noise
- Calculate the SIRP quadratic form
- Calculate histogram data for the SIRP quadratic form
- Convert a file to ASCII in a format suitable for Dr. Romeu's software
- Convert data to ASCII file

The user should select the appropriate menu prompt. The selected programs are automatically compiled if necessary.

Table B-1: Files in the Minisystem

File	Description
README	Describes how to install and run the system
data	An empty subdirectory
source source/cbta.f source/cgwn.f source/data2ascii.f source/qhist.f source/qsirp.f source/scarsirp.f source/sirpm.f source/Makefile source/sun4.make source/sun3.make	A directory of Fortran source code. Converts binary data files to ASCII Generates single channel Gaussian white noise Converts binary files to ASCII in Romeu format Histogram of quadratic form for SIRPs Quadratic form for SIRPs Generates single channel AR processes Generates single channel SIRP noise Compiles and links above programs Called from Makefile Called from Makefile
schan	A c-shell for invoking the system
hlp hlp/cbta.hlp hlp/cgwn.hlp hlp/data2ascii.hlp hlp/scarsirp.hlp hlp/sirpm.hlp	A directory of help files associated with the main programs
sun3 sun3/libfsl.a sun3/libmat.a sun3/libmc.a sun3/libmv.a sun3/libvec.a	Sun 3 object code for various library routines
sun4 sun4/libfsl.a sun4/libmat.a sun4/libmc.a sun4/libmv.a sun4/libvec.a	Sun 4 object code for various library routines
bin bin/sun3 bin/sun4 bin/exe bin/menu1	A directory to contain linked executables and object code A directory to contain compiled Sun 3 object code A directory to contain compiled Sun 4 object code A directory to contain linked executables A c-shell for the main menu

Appendix C. Generation of K-Distributed SIRPs

The MSPSS includes an option for generating K-distributed Spherically Invariant Random Processes (SIRPs). This capability was added during a previous effort (Vienneau 93). The algorithm specified in (Rangaswamy 91) is used to generate a SIRP. This algorithm asks the user to enter the two parameters α and b . As a matter of fact, SIRPs can be generated with knowledge only of α . The parameter b is totally unnecessary. This appendix provides the mathematical foundation for this claim. The approaches described here are not implemented in the MSPSS. This appendix is only provided to document these findings.

K-distributed SIRPs rely on the generation of a random variable S with the following Probability Density Function (PDF):

$$f_s(s) = \frac{2\alpha^\alpha}{\Gamma(\alpha)} s^{2\alpha-1} e^{-\alpha s^2}, \quad s > 0. \quad (C-1)$$

Rangaswamy suggests S can be generated based on the following theorem:

Theorem: Let V be a random variable with the following PDF:

$$f_v(v) = \frac{2b}{\Gamma(\alpha)2^\alpha} (bv)^{2\alpha-1} e^{-b^2v^2/2}, \quad v > 0. \quad (C-2)$$

Let $S = V/a$, where $a^2 = \frac{2\alpha}{b^2}$. Then S is distributed as above with PDF $f_s(s)$.

The random variable V can be generated from $V = \frac{\sqrt{W}}{b}$, where W is from a Chi Squared distribution with 2α degrees of freedom.

Alternately, S can be generated based on either of the following two theorems:

Theorem: Let W be from a Chi Square distribution with 2α degrees of freedom. Let $S = \sqrt{\frac{W}{2\alpha}}$. Then S is distributed as above with PDF $f_s(s)$.

Theorem: Let W be from a Gamma distribution with parameters α and β , where $\beta = 1/\alpha$. That is, W has the following PDF:

$$f_w(w) = \frac{1}{\Gamma(\alpha)\beta^\alpha} w^{\alpha-1} e^{-w/\beta}, \quad x > 0. \quad (C-3)$$

Let $S = \sqrt{W}$ Then S is distributed as above with PDF $f_s(s)$.

Appendix D.

One-Pass Algorithms for Calculating Moments of Distributions

Many programs in the Multichannel Signal Processing Simulation System provide the user with the capability to estimate certain parameters for each of many realizations of a stochastic process. Estimated parameters include Autoregressive coefficients, means, the covariance matrix, and correlation functions. Often the program reports the mean and variance of these estimates calculated over all realizations of the stochastic process.

D.1 Means and Variances

The structure of the MSPSS makes it convenient to calculate these means and variances from one pass over the stochastic process realizations. Since the variance is defined in terms of the mean, calculating the variance from one pass through the data requires some care. The so-called "calculator" formula, found in many statistics textbooks, should not be used:

$$S^2 = \frac{1}{(n-1)} \sum_{i=1}^n (x_i - \bar{x})^2 = \frac{1}{(n-1)} \sum_{i=1}^n x_i^2 - \frac{n}{(n-1)} \bar{x}^2. \quad (D-1)$$

Equation D-1 yields a one-pass algorithm for calculating the variance, but this algorithm is numerically unstable. The two terms in the difference can be of the same order of magnitude, leaving the variance to be determined by round-off error.

The approach for calculating the variance was obtained from (Chen 83) and is based on the two following theorems:

Theorem D-1: Suppose

$$M_k = M_{k-1} + \frac{1}{k} (x_k - M_{k-1}) \quad (D-2)$$

and

$$M_0 = 0. \quad (D-3)$$

Then

$$M_k = \frac{1}{k} \sum_{i=1}^k x_i. \quad (D-4)$$

In other words, M_k , as defined by the above difference equation, is the sample mean of x_1, x_2, \dots, x_k .

Theorem D-2: Let M_k be as above. Suppose

$$(k-1)S_k^2 = (k-2)S_{k-1}^2 + \frac{k}{(k-1)} (x_k - M_k)^2. \quad (D-5)$$

for $k = 2, 3, 4, \dots$ Suppose

$$S_1^2 = 0. \quad (D-6)$$

Then

$$S_k^2 = \frac{1}{(k-1)} \sum_{i=1}^k (x_i - M_k)^2. \quad (\text{D-7})$$

for $k = 2, 3, 4, \dots$. In other words, S_k^2 is the sample variance of x_1, x_2, \dots, x_k .

A computer program based on these theorems would update M_k and S_k^2 from a single pass of the data. M_k must be updated before S_k^2 since M_k , not M_{k-1} , appears in Equation D-5. On completion, the final values of M_k and S_k^2 will be the desired means and variances.

For completeness, proofs of these two theorems are provided below. A discussion of their numerical properties is provided in the referenced paper.

Proof of Theorem D-1 (by induction):

First, show the theorem holds for $k = 1$. By hypothesis,

$$M_1 = M_0 + \frac{1}{1} (x_1 - M_0) \quad (\text{D-8})$$

$$M_1 = 0 + 1(x_1 - 0) = x_1 \quad (\text{D-9})$$

$$M_1 = \frac{1}{1} \sum_{i=1}^1 x_i. \quad (\text{D-10})$$

Second, assume the theorem holds for $k - 1$:

$$M_{k-1} = \frac{1}{k-1} \sum_{i=1}^{k-1} x_i. \quad (\text{D-11})$$

Finally, prove the theorem holds for k :

$$M_k = M_{k-1} + \frac{1}{k} (x_k - M_{k-1}) \quad (\text{D-12})$$

$$M_k = \frac{1}{k} (k M_{k-1} + x_k - M_{k-1}) \quad (\text{D-13})$$

$$M_k = \frac{1}{k} [(k-1)M_{k-1} + x_k] \quad (\text{D-14})$$

$$M_k = \frac{1}{k} \left[(k-1) \frac{1}{(k-1)} \sum_{i=1}^{k-1} x_i + x_k \right] \quad (\text{D-15})$$

$$M_k = \frac{1}{k} \left[\sum_{i=1}^{k-1} x_i + x_k \right] \quad (\text{D-16})$$

$$M_k = \frac{1}{k} \sum_{i=1}^k x_i \quad (\text{D-17})$$

But this is what was to be shown.

Proof of Theorem D-2 (by induction):

First, show the theorem holds for $k = 2$. By hypothesis,

$$(2-1)S_2^2 = (2-2)S_{2-1}^2 + \frac{2}{(2-1)}(x_2 - M_2)^2 \quad (\text{D-18})$$

$$1S_2^2 = 0S_1 + \frac{2}{1}\left[x_2 - \frac{1}{2}(x_1 + x_2)\right]^2 \quad (\text{D-19})$$

$$S_2^2 = 2\left[\frac{x_2 - x_1}{2}\right]^2 \quad (\text{D-20})$$

$$S_2^2 = \left[\frac{x_1 - x_2}{2}\right]^2 + \left[\frac{x_2 - x_1}{2}\right]^2 \quad (\text{D-21})$$

$$S_2^2 = \left[x_1 - \frac{x_1 + x_2}{2}\right]^2 + \left[x_2 - \frac{x_1 + x_2}{2}\right]^2 \quad (\text{D-22})$$

$$S_2^2 = (x_1 - M_2)^2 + (x_2 - M_2)^2 \quad (\text{D-23})$$

$$S_2^2 = \frac{1}{2-1} \sum_{i=1}^2 (x_i - M_2)^2 \quad (\text{D-24})$$

Second, assume the theorem holds for $k - 1$:

$$S_{k-1}^2 = \frac{1}{(k-2)} \sum_{i=1}^{k-1} (x_i - M_{k-1})^2 \quad (\text{D-25})$$

Hence,

$$(k-2)S_{k-1}^2 = \sum_{i=1}^{k-1} (x_i - M_{k-1})^2 \quad (\text{D-26})$$

Finally, prove the theorem holds for k :

$$\sum_{i=1}^k (x_i - M_k)^2 = \sum_{i=1}^{k-1} (x_i - M_k)^2 + (x_k - M_k)^2 \quad (\text{D-27})$$

$$\begin{aligned} \sum_{i=1}^k (x_i - M_k)^2 &= \sum_{i=1}^{k-1} \left[x_i - M_{k-1} - \frac{1}{k}(x_k - M_{k-1})\right]^2 \\ &\quad + \frac{(k-1)}{(k-1)}(x_k - M_k)^2 \end{aligned} \quad (\text{D-28})$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k)^2 &= \sum_{i=1}^{k-1} (x_i - M_{k-1})^2 - \sum_{i=1}^{k-1} \left[\frac{2}{k} (x_k - M_{k-1})(x_i - M_{k-1}) \right] \\
&\quad + \sum_{i=1}^{k-1} \left[\frac{1}{k^2} (x_k - M_{k-1})^2 \right] + \frac{k}{(k-1)} (x_k - M_k)^2 \\
&\quad - \frac{1}{(k-1)} (x_k - M_k)^2
\end{aligned} \tag{D-29}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k)^2 &= (k-2)S_{k-1}^2 - \frac{2}{k} (x_k - M_{k-1}) \sum_{i=1}^{k-1} x_i \\
&\quad + \frac{2(k-1)}{k} (x_k - M_{k-1})M_{k-1} + \frac{(k-1)}{k^2} (x_k - M_{k-1})^2 \\
&\quad + \frac{k}{(k-1)} (x_k - M_k)^2 - \frac{1}{(k-1)} \left[x_k - M_{k-1} - \frac{1}{k} (x_k - M_{k-1}) \right]^2
\end{aligned} \tag{D-30}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k)^2 &= (k-2)S_{k-1}^2 - \frac{2(k-1)}{k} (x_k - M_{k-1}) \frac{1}{(k-1)} \sum_{i=1}^{k-1} x_i \\
&\quad + \frac{2(k-1)}{k} (x_k - M_{k-1})M_{k-1} + \frac{(k-1)}{k^2} (x_k - M_{k-1})^2 \\
&\quad + \frac{k}{(k-1)} (x_k - M_k)^2 - \frac{1}{(k-1)k^2} (k x_k - k M_{k-1} - x_k + M_{k-1})^2
\end{aligned} \tag{D-31}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k)^2 &= (k-2)S_{k-1}^2 - \frac{2(k-1)}{k} (x_k - M_{k-1})M_{k-1} \\
&\quad + \frac{2(k-1)}{k} (x_k - M_{k-1})M_{k-1} + \frac{(k-1)}{k^2} (x_k - M_{k-1})^2 \\
&\quad + \frac{k}{(k-1)} (x_k - M_k)^2 \\
&\quad - \frac{1}{(k-1)k^2} [(k x_k - x_k) - (k M_{k-1} - M_{k-1})]^2
\end{aligned} \tag{D-32}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k)^2 &= (k-2)S_{k-1}^2 + \frac{(k-1)}{k^2} (x_k - M_{k-1})^2 \\
&\quad + \frac{k}{(k-1)} (x_k - M_k)^2 - \frac{(k-1)^2}{(k-1)k^2} (x_k - M_{k-1})^2
\end{aligned} \tag{D-33}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k)^2 &= (k-2)S_{k-1}^2 + \frac{k}{(k-1)} (x_k - M_k)^2 \\
&\quad + \frac{(k-1)}{k^2} (x_k - M_{k-1})^2 - \frac{(k-1)}{k^2} (x_k - M_{k-1})^2
\end{aligned} \tag{D-34}$$

$$\sum_{i=1}^k (x_i - M_k)^2 = (k-2)S_{k-1}^2 + \frac{k}{(k-1)}(x_k - M_k)^2 \quad (\text{D-35})$$

Therefore,

$$(k-1)S_k^2 = (k-2)S_{k-1}^2 + \frac{k}{(k-1)}(x_k - M_k)^2 \quad (\text{D-36})$$

$$(k-1)S_k^2 = \sum_{i=1}^k (x_i - M_k)^2 \quad (\text{D-37})$$

Or

$$S_k^2 = \frac{1}{(k-1)} \sum_{i=1}^k (x_i - M_k)^2 \quad (\text{D-38})$$

which was to be shown.

D.2 Correlation Coefficients

A one-pass algorithm for calculating the correlation coefficient was needed in implementing Ozturk's algorithm. The algorithm implemented relies on the following theorem:

Theorem D-3: Suppose

$$SXY_1 = M_0^X = M_0^Y = 0, \quad (\text{D-39})$$

and

$$M_k^X = M_{k-1}^X + \frac{1}{k}(x_k - M_{k-1}^X), \quad (\text{D-40})$$

$$M_k^Y = M_{k-1}^Y + \frac{1}{k}(y_k - M_{k-1}^Y), \quad (\text{D-41})$$

for $k = 1, 2, 3, \dots$. Suppose

$$SXY_k = SXY_{k-1} + \frac{k}{k-1}(x_k - M_{k-1}^X)(y_k - M_{k-1}^Y), \quad (\text{D-42})$$

for $k = 2, 3, 4, \dots$. Then,

$$SXY_k = \sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y). \quad (\text{D-43})$$

Given two random samples, x_1, x_2, \dots, x_n and y_1, y_2, \dots, y_n , the customary sample estimate of the correlation coefficient is given by Equation D-40:

$$R_n = \frac{SXY_n}{(n-1)S_n^X S_n^Y}. \quad (\text{D-44})$$

A proof of Theorem D-3 follows.

Proof of Theorem D-3 (by induction):

From Theorem D-1, we know

$$M_k^X = \frac{1}{k} \sum_{i=1}^k x_i \quad (\text{D-45})$$

$$M_k^Y = \frac{1}{k} \sum_{i=1}^k y_i \quad (\text{D-46})$$

First, show the theorem holds for $k = 2$. By hypothesis,

$$SXY_2 = SXY_1 + \frac{2}{2-1} (x_2 - M_2^X)(y_2 - M_2^Y) \quad (\text{D-47})$$

$$SXY_2 = SXY_1 + \frac{2}{1} [x_2 - \frac{1}{2}(x_1 + x_2)][y_2 - \frac{1}{2}(y_1 + y_2)] \quad (\text{D-48})$$

$$SXY_2 = [\frac{1}{2}(x_1 + x_2) - x_2][\frac{1}{2}(y_1 + y_2) - y_2] + [x_2 - \frac{1}{2}(x_1 + x_2)][y_2 - \frac{1}{2}(y_1 + y_2)] \quad (\text{D-49})$$

$$SXY_2 = \left[\frac{2x_1 - x_1 + x_2 - 2x_2}{2} \right] \left[\frac{2y_1 - y_1 + y_2 - 2y_2}{2} \right] + (x_2 - M_2^X)(y_2 - M_2^Y) \quad (\text{D-50})$$

$$SXY_2 = \left[x_1 - \frac{x_1 + x_2}{2} \right] \left[y_1 - \frac{y_1 + y_2}{2} \right] + (x_2 - M_2^X)(y_2 - M_2^Y) \quad (\text{D-51})$$

$$SXY_2 = (x_1 - M_2^X)(y_1 - M_2^Y) + (x_2 - M_2^X)(y_2 - M_2^Y) \quad (\text{D-52})$$

$$SXY_2 = \sum_{i=1}^2 (x_i - M_2^X)(y_i - M_2^Y) \quad (\text{D-53})$$

Second, assume the theorem holds for $k - 1$:

$$SXY_{k-1} = \sum_{i=1}^{k-1} (x_i - M_{k-1}^X)(y_i - M_{k-1}^Y) \quad (\text{D-54})$$

Finally, prove the theorem holds for k :

$$\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) = \sum_{i=1}^{k-1} (x_i - M_{k-1}^X)(y_i - M_{k-1}^Y) + (x_k - M_k^X)(y_k - M_k^Y) \quad (\text{D-55})$$

$$\begin{aligned} \sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) &= \sum_{i=1}^{k-1} [x_i - M_{k-1}^X - \frac{1}{k}(x_k - M_{k-1}^X)][y_i - M_{k-1}^Y - \frac{1}{k}(y_k - M_{k-1}^Y)] \\ &\quad + (x_k - M_k^X)(y_k - M_k^Y) \end{aligned} \quad (\text{D-56})$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) &= \sum_{i=1}^{k-1} (x_i - M_{k-1}^X)(y_i - M_{k-1}^Y) \\
&\quad - \frac{1}{k}(x_k - M_{k-1}^X) \sum_{i=1}^{k-1} (y_i - M_{k-1}^Y) \\
&\quad - \frac{1}{k}(y_k - M_{k-1}^Y) \sum_{i=1}^{k-1} (x_i - M_{k-1}^X) \\
&\quad + \frac{(k-1)}{k^2} (x_k - M_{k-1}^X)(y_k - M_{k-1}^Y) \\
&\quad + (x_k - M_k^X)(y_k - M_k^Y)
\end{aligned} \tag{D-57}$$

The inductive hypothesis is used at this step.

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) &= SXY_{k-1} - \frac{1}{k}(x_k - M_{k-1}^X) \left[\sum_{i=1}^{k-1} y_i - (k-1)M_{k-1}^Y \right] \\
&\quad - \frac{1}{k}(y_k - M_{k-1}^Y) \left[\sum_{i=1}^{k-1} x_i - (k-1)M_{k-1}^X \right] \\
&\quad + \frac{(k-1)}{k^2} (x_k - M_{k-1}^X)(y_k - M_{k-1}^Y) \\
&\quad + (x_k - M_k^X)(y_k - M_k^Y)
\end{aligned} \tag{D-58}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) &= SXY_{k-1} - \frac{1}{k}(x_k - M_{k-1}^X) [(k-1)M_{k-1}^Y - (k-1)M_{k-1}^Y] \\
&\quad - \frac{1}{k}(y_k - M_{k-1}^Y) [(k-1)M_{k-1}^X - (k-1)M_{k-1}^X] \\
&\quad + \frac{(k-1)}{k^2} (x_k - M_{k-1}^X)(y_k - M_{k-1}^Y) \\
&\quad + (x_k - M_k^X)(y_k - M_k^Y)
\end{aligned} \tag{D-59}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) &= SXY_{k-1} + \frac{(k-1)}{k^2} (x_k - M_{k-1}^X)(y_k - M_{k-1}^Y) \\
&\quad + (x_k - M_k^X)(y_k - M_k^Y)
\end{aligned} \tag{D-60}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) &= SXY_{k-1} + \frac{(k-1)}{k^2} (x_k - M_{k-1}^X)(y_k - M_{k-1}^Y) \\
&+ \frac{k}{(k-1)} (x_k - M_k^X)(y_k - M_k^Y) \\
&- \frac{1}{(k-1)} (x_k - M_k^X)(y_k - M_k^Y)
\end{aligned} \tag{D-61}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) &= SXY_{k-1} + \frac{k}{(k-1)} (x_k - M_k^X)(y_k - M_k^Y) \\
&+ \frac{(k-1)}{k^2} (x_k - M_{k-1}^X)(y_k - M_{k-1}^Y) \\
&- \frac{1}{(k-1)} [x_k - M_{k-1}^X - \frac{1}{k} (x_k - M_{k-1}^X) \\
&\quad [y_k - M_{k-1}^Y - \frac{1}{k} (y_k - M_{k-1}^Y)]
\end{aligned} \tag{D-62}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) &= SXY_{k-1} + \frac{k}{(k-1)} (x_k - M_k^X)(y_k - M_k^Y) \\
&+ \frac{(k-1)}{k^2} (x_k - M_{k-1}^X)(y_k - M_{k-1}^Y) \\
&- \frac{1}{(k-1)} \left[\frac{(k-1)}{k} x_k - \frac{(k-1)}{k} M_{k-1}^X \right] \\
&\quad \left[\frac{(k-1)}{k} y_k - \frac{(k-1)}{k} M_{k-1}^Y \right]
\end{aligned} \tag{D-63}$$

$$\begin{aligned}
\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) &= SXY_{k-1} + \frac{k}{(k-1)} (x_k - M_k^X)(y_k - M_k^Y) \\
&+ \frac{(k-1)}{k^2} (x_k - M_{k-1}^X)(y_k - M_{k-1}^Y) \\
&- \frac{1}{(k-1)} \frac{(k-1)}{k} [x_k - M_{k-1}^X] \frac{(k-1)}{k} [y_k - M_{k-1}^Y]
\end{aligned} \tag{D-64}$$

Thus,

$$\sum_{i=1}^k (x_i - M_k^X)(y_i - M_k^Y) = SXY_{k-1} + \frac{k}{k-1} (x_k - M_k^X)(y_k - M_k^Y), \tag{D-65}$$

which was to be shown.

Appendix E. RANDOM VARIATE GENERATION

Ozturk's algorithm relies on the simulation of random variates from certain probability distributions. Currently, 15 distributions are implemented in the software implementation of Ozturk's algorithm. This appendix defines these distributions and explains how variates are generated from them.

E.1 Uniform Variates

Variates from all other distributions are generated based on transformations of variates from a uniform distribution. For example, a common technique for generating a variate from a given distribution is to invert the Cumulative Distribution Function (CDF). Then a random variate X can be generated from the desired distribution by Equation E-1:

$$X = F^{-1}(U), \quad (\text{E-1})$$

where U is uniform on $(0, 1)$ and F is the CDF of the desired distribution.

The generation of random variates from a uniform distribution is a problem commonly treated in the literature. The random number generator used in the MSPSS is based on an "universal" random number generator algorithm developed by George Marsaglia of the Supercomputer Computations Research Institute (SCRI) at Florida State University. This algorithm has passed stringent tests for randomness and independence. The generator has an extremely long period, about 2^{144} , and, for the default seed values, it generates exactly the same sequence of 24-bit random numbers in a variety of language and machines. (Harmon 88)

E.2 Beta Variates

A method for generating variates from Beta distributions was already embodied at the beginning of this in the Gamma generator for the MSPSS. This method, which differs from that used by Ozturk, is described here.

The Probability Density Function (PDF) of a Beta distribution is given by Equation E-2:

$$f(x) = \frac{\Gamma(a+b+2)}{\Gamma(a+1)\Gamma(b+1)} x^a (1-x)^b, \quad 0 < x < 1. \quad (\text{E-2})$$

The mean of a Beta distribution is $\frac{a+1}{a+b+2}$, and the variance is $\frac{(a+1)(b+1)}{(a+b+2)^2(a+b+3)}$.

The method used for transforming uniform variates into Beta variates is based on the following theorem:

Theorem: Let U_1 and U_2 be independent identically distributed random variables from a uniform distribution on $(0, 1]$. Let

$$Y_1 = U_1^{1/a}, \quad (\text{E-3})$$

$$Y_2 = U_2^{1/b}, \quad (\text{E-4})$$

If $Y_1 + Y_2$ is less than or equal to unity, set Y to

$$Y = \frac{Y_1}{Y_1 + Y_2}. \quad (\text{E-5})$$

Then Y is from a standard Beta distribution with parameters a and b .

E.3 Cauchy Variates

The PDF of the Cauchy distribution is

$$f(x) = \frac{1}{\pi(1+x^2)}. \quad (\text{E-6})$$

The Cumulative Distribution Function (CDF) is

$$F(x) = \frac{1}{2} + \frac{1}{\pi} \text{Tan}^{-1}(x). \quad (\text{E-7})$$

A variate can be generated from this distribution as

$$X = \text{Tan} \left[\pi \left(U - \frac{1}{2} \right) \right], \quad (\text{E-8})$$

where U is uniform on $(0, 1)$.

E.4 Exponential Variates

The PDF for the Exponential distribution is

$$f(x) = -\frac{1}{\mu} e^{-x/\mu}, \quad x \geq 0. \quad (\text{E-9})$$

The CDF is

$$F(x) = 1 - e^{-x/\mu}, \quad x \geq 0. \quad (\text{E-10})$$

A variate from this distribution can be generated as

$$X = -\mu \text{Ln}(U), \quad (\text{E-11})$$

where U is uniform on $(0, 1]$.

E.5 Extreme Value Variates

The PDF for the Type I Extreme Value distribution for minimum elements, also called the Gumbel distribution, is

$$f(x) = \frac{1}{\sigma} e^{\frac{x}{\sigma} - e^{\frac{x}{\sigma}}}, \quad (\text{E-12})$$

where σ is the scale parameter. The CDF is

$$F(x) = 1 - e^{-e^{\frac{x}{\sigma}}}. \quad (\text{E-13})$$

(Dr. Ozturk code is mistaken in the calculation of the Extreme Value CDF in the routine THETA for calculating the angles between the linked vectors and the abscissa.) A random variate can be generated from this distribution as

$$X = \sigma \text{Ln}[-\text{Ln}(U)], \quad (\text{E-14})$$

where U is uniform on $(0, 1)$.

E.6 Gamma Variates

The capacity to generate Gamma variates already existed in the MSPSS at the beginning of this effort. The algorithm used is taken from (Fishman 73). It tends to create overflow and underflow problems for $\alpha < 0.25$ or $\alpha > 100.0$. Warning messages alert the user to these

problems. The algorithm tends to become much slower as the number of degrees of freedom increases. This algorithm differs from the one embedded in Ozturk's code.

The PDF for a Gamma distribution is

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta}, \quad x > 0. \quad (\text{E-15})$$

where α is the shape parameter and β is the scale parameter. The mean of the Gamma distribution is $\alpha\beta$, and the variance is $\alpha\beta^2$.

The algorithm first generates a random variable Y with a Gamma distribution with a scale parameter equal to unity. Once Y is generated, a variable X is set to βY . X then has the desired shape and scale parameters. This procedure relies on the following theorem:

Theorem: Let Y be a random variable from a Gamma distribution with an arbitrary shape parameter α but a scale parameter equal to one. Let

$$X = \beta Y. \quad (\text{E-16})$$

Then X has a Gamma distribution with shape parameter α and scale parameter β .

The random variate Y is generated by partitioning α into an integer part and a real part. Gamma variates are generated for each of these parts, and their sum is the desired Gamma variate:

Theorem Let V be from a Gamma distribution with a shape parameter $k = \lfloor \alpha \rfloor$ and an unit scale parameter. Let W be from a Gamma distribution with a shape parameter $\alpha - k$ and an unit scale parameter. Let

$$Y = V + W \quad (\text{E-17})$$

Then Y is from a Gamma distribution with shape parameter α and unit scale parameter.

A Gamma-distributed random variate with an integer shape parameter and unit scale parameter is generated as the sum of exponential random variates:

Theorem Let $\Lambda_i, i = 1, 2, \dots, k$ be independent and identically distributed random variables from an exponential distribution with unit means. Let

$$V = \Lambda_1 + \Lambda_2 + \dots + \Lambda_k. \quad (\text{E-18})$$

Then V is Gamma distributed with a shape parameter of k and an unit scale parameter.

A Gamma-distributed random variate with a real shape parameter between zero and one is generated as the product of a Beta-distributed random variate and an exponentially-distributed random variate:

Theorem Let Z be from a Beta distribution with parameters $a = \alpha$ and $b = 1 - \alpha$. Let Λ be exponentially distributed with unit mean. Let

$$W = Z \Lambda. \quad (\text{E-19})$$

Then W is Gamma distributed with shape parameter α and unit scale parameter.

E.7 Gumbel (Type II) Variates

The PDF for a Gumbel (Type II) Extreme Value distribution is

$$f(x) = \frac{\beta}{\alpha} \left[\frac{x}{\alpha} \right]^{-\beta-1} e^{-\left(\frac{x}{\alpha}\right)^{-\beta}}, \quad x > 0. \quad (\text{E-20})$$

where α is the scale parameter and β is the shape parameter. The CDF is

$$F(x) = e^{-\left(\frac{x}{\alpha}\right)^{-\beta}}, \quad x > 0. \quad (\text{E-21})$$

A variate can be generated from this distribution as

$$X = \alpha \left[\frac{-1}{\text{Ln}(U)} \right]^{\frac{1}{\beta}}. \quad (\text{E-22})$$

where U is uniform on (0, 1).

E.8 Johnson Variates

The PDF of a Johnson S_U distribution is

$$f(x) = \frac{\eta}{\sqrt{2\pi\lambda^2[(y-\varepsilon)^2 + \lambda^2]}} \text{Exp} \left[-\frac{1}{2\lambda^2} \left[\gamma - \varepsilon + \eta \text{Ln} \left(\frac{1}{\lambda} [y - \varepsilon + \sqrt{(y - \varepsilon)^2 + \lambda^2}] \right) \right]^2 \right] \quad (\text{E-23})$$

ε is the location parameter, λ is the scale parameter, and γ and η are shape parameters. A random variable from a Johnson S_U distribution can be generated by transforming a standard normally distributed variate:

$$X = \varepsilon + \lambda \sinh \left[\frac{Z - \gamma}{\eta} \right]. \quad (\text{E-24})$$

where Z is standard Gaussian.

E.9 K Distributed Variates

The ability to generate random variables from a K distribution existed in the MSPSS when this project began. Appendix C describes how this capability is implemented. The PDF of a K distribution includes a shape parameter α .

E.10 Laplace Variates

The PDF for the Laplace distribution is

$$f(x) = \frac{1}{2} e^{-|x|}. \quad (\text{E-25})$$

The mean of this distribution is zero, and the variance is two. The CDF is

$$F(x) = \begin{cases} \frac{1}{2} e^x, & x < 0 \\ 1 - \frac{1}{2} e^{-x}, & x \geq 0 \end{cases} \quad (\text{E-26})$$

A variate from the Laplace distribution can be generated by

$$X = \begin{cases} \text{Ln}(2U), & U < \frac{1}{2} \\ -\text{Ln}[2(1-U)], & U \geq \frac{1}{2} \end{cases} \quad (\text{E-27})$$

where U is uniform on (0, 1).

E.11 Logistic Variates

The PDF for the Logistic distribution is

$$f(x) = \frac{e^{-x}}{(1 + e^{-x})^2}. \quad (\text{E-28})$$

The CDF is

$$F(x) = \frac{1}{1 + e^{-x}}. \quad (\text{E-29})$$

A variate can be generated from this distribution as

$$X = -\text{Ln} \left[\frac{1 - U}{U} \right], \quad (\text{E-30})$$

where U is uniform on $(0, 1)$.

E.12 Lognormal Variates

A random variable is from a Lognormal distribution if its natural logarithm is normally distributed. The PDF of a lognormal distribution is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-\frac{1}{2} \left[\frac{\text{Ln}(x) - \mu}{\sigma} \right]^2}, \quad x > 0. \quad (\text{E-31})$$

μ and σ are the mean and standard deviation, respectively, of the normal transformation, not the mean and standard deviation of the lognormal distribution.

Lognormal variates are generated by transforming normal variates:

$$X = e^Z, \quad (\text{E-32})$$

where Z is normally distributed with mean μ and variance σ^2 . (The generation of lognormal variates is not implemented correctly in Dr. Ozturk's code.)

E.13 Normal Variates

The capability to generate normal variates already existed in the MSPSS at the beginning of this effort. The algorithm described here is based on (Press 86). The PDF for the Normal distribution is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x - \mu)^2}{2\sigma^2}}, \quad (\text{E-33})$$

where μ is the mean and σ^2 is the variance.

In generating normal variates, it is sufficient to design an algorithm to generate zero-mean, unit-variance normal variates. Normal variates with arbitrary means and variance can then be generated by means of the following transformation:

$$X = \mu + \sigma Z, \quad (\text{E-34})$$

where Z is normal with mean zero and unit variance. The resulting variate X is normally distributed with mean μ and variance σ^2 .

The algorithm used to generate standard normal variates is as follows. A pair of random variates (V_1, V_2) are generated where the pair is uniformly distributed on the unit circle. Then,

the random variables Z_1 and Z_2 given by

$$Z_1 = V_1 \sqrt{-2 \frac{\text{Ln}(R)}{R}} \quad (\text{E-35})$$

$$Z_2 = V_2 \sqrt{-2 \frac{\text{Ln}(R)}{R}} \quad (\text{E-36})$$

are stochastically independent and normally distributed with mean zero and unit variance, where R is

$$R = V_1^2 + V_2^2. \quad (\text{E-37})$$

E.14 Pareto Variates

The PDF for the Pareto distribution is

$$f(x) = \theta_1^{\theta_2} \theta_2 x^{-(\theta_2+1)}, \quad x \geq \theta_1. \quad (\text{E-38})$$

The mean is $\theta_1 \theta_2 / (\theta_2 - 1)$. The variance is $\theta_1^2 \theta_2 / [(\theta_2 - 1)^2 (\theta_2 - 2)]$. The CDF for this distribution is

$$F(x) = 1 - \left[\frac{\theta_1}{x} \right]^{\theta_2}, \quad x \geq \theta_1. \quad (\text{E-39})$$

A variate from the Pareto distribution can be generated as follows:

$$X = \theta_1 (1/U)^{1/\theta_2}. \quad (\text{E-40})$$

where U is uniform on $(0, 1]$.

E.15 Weibull Variates

The PDF for the Weibull distribution is

$$f(x) = \frac{\beta}{\alpha^\beta} x^{\beta-1} e^{-(x/\alpha)^\beta}, \quad x \geq 0. \quad (\text{E-41})$$

where α is the scale parameter and β is the shape parameter. The CDF is

$$F(x) = 1 - e^{-(x/\alpha)^\beta}, \quad x \geq 0. \quad (\text{E-42})$$

The mean is $\alpha \Gamma(1 + 1/\beta)$ and the variance is $\alpha^2 [\Gamma(1 + 2/\beta) - \Gamma^2(1 + 1/\beta)]$. A Weibull variate can be generated as

$$X = \alpha [-\text{Ln}(U)]^{1/\beta}. \quad (\text{E-43})$$

where U is uniform on $(0, 1]$.

Appendix F. CONFIDENCE REGIONS

Implementing Ozturk's goodness-of-fit test requires the calculation of confidence regions for Ozturk's statistic. This appendix outlines the theory behind these calculations.

F.1 Bivariate Normal Distribution

Consider the problem of determining a confidence ellipse for a pair of random variables from a bivariate normal distribution. Let X_1 and X_2 have the joint Probability Density Function $f(x_1, x_2)$:

$$f(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} e^{\left\{-\frac{1}{2} \frac{1}{(1-\rho^2)} \left[\left(\frac{x_1-\mu_1}{\sigma_1} \right)^2 + 2\rho \left(\frac{x_1-\mu_1}{\sigma_1} \right) \left(\frac{x_2-\mu_2}{\sigma_2} \right) + \left(\frac{x_2-\mu_2}{\sigma_2} \right)^2 \right] \right\}} \quad (\text{F-1})$$

This is the PDF of a bivariate normal distribution where X_i has mean μ_i and variance σ_i^2 , $i = 1, 2$. The correlation coefficient for X_1 and X_2 is ρ . The problem is to determine a confidence region C such that the probability of X_1 and X_2 being in C is 100 $\gamma\%$.

Consider the transformation defined in Equations F-2 and F-3:

$$Z_1 = u_1(X_1, X_2) = \frac{X_1 - \mu_1}{\sigma_1} \quad (\text{F-2})$$

$$Z_2 = u_2(X_1, X_2) = \frac{1}{\sqrt{1-\rho^2}} \left[\left(\frac{X_2 - \mu_2}{\sigma_2} \right) - \rho \left(\frac{X_1 - \mu_1}{\sigma_1} \right) \right] \quad (\text{F-3})$$

The inverse transformation is given by Equations F-4 and F-5:

$$X_1 = w_1(Z_1, Z_2) = \mu_1 + \sigma_1 Z_1 \quad (\text{F-4})$$

$$X_2 = w_2(Z_1, Z_2) = \mu_2 + \sigma_2 \rho Z_1 + \sigma_2 \sqrt{1-\rho^2} Z_2 \quad (\text{F-5})$$

The Jacobian of this transformation is

$$J_w = \begin{vmatrix} \frac{dw_1}{dz_1} & \frac{dw_1}{dz_2} \\ \frac{dw_2}{dz_1} & \frac{dw_2}{dz_2} \end{vmatrix} = \sigma_1 \sigma_2 \sqrt{1-\rho^2}. \quad (\text{F-6})$$

Let $g(z_1, z_2)$ be the joint PDF for Z_1 and Z_2 . From a theorem in analysis relating to a change of variables in integration (Hogg 78), the joint PDF of Z_1 and Z_2 is related to the joint PDF for X_1 and X_2 by:

$$g(z_1, z_2) = |J_w| f(w_1(z_1, z_2), w_2(z_1, z_2)). \quad (\text{F-7})$$

Or

$$g(z_1, z_2) = \frac{1}{2\pi} e^{-\frac{1}{2}(z_1^2 + z_2^2)}. \quad (\text{F-8})$$

So the transformation has converted the original random variables into stochastically independent standard normal variates.

A confidence region in the transformed space is a circle:

$$u(C) = \{(z_1, z_2) \mid z_1^2 + z_2^2 < c^2\}. \quad (\text{F-9})$$

where the radius c is chosen so as to ensure the probability of membership in this circle is the desired value γ . This probability can be found as follows:

$$Pr(z_1^2 + z_2^2 < c^2) = \iint_{d(C)} f(z_1, z_2) dz_1 dz_2 \quad (\text{F-10})$$

$$Pr(z_1^2 + z_2^2 < c^2) = \int_0^{2\pi} \int_0^c \frac{1}{2\pi} e^{-\frac{r^2}{2}} r dr d\theta \quad (\text{F-11})$$

$$Pr(z_1^2 + z_2^2 < c^2) = 1 - e^{-\frac{c^2}{2}}. \quad (\text{F-12})$$

Set this probability to γ , and solve for c :

$$c = \sqrt{-2 \ln(1 - \gamma)}. \quad (\text{F-13})$$

The confidence region for Z_1 and Z_2 can be transformed back into the original space. Thus, a 100 $\gamma\%$ confidence ellipse for X_1 and X_2 is

$$C = \{(x_1, x_2) \mid [u_1(x_1, x_2)]^2 + [u_2(x_1, x_2)]^2 < -2 \ln(1 - \gamma)\}. \quad (\text{F-14})$$

An approximate confidence ellipse is formed by using sample estimates for the means, variances, and correlation coefficients in Equation F-14.

F.2 The Johnson System of Transformations

Ozturk's statistic need not be from a bivariate normal distribution. The problem then arises of approximating a confidence region for other distributions, even if the distribution is unknown. The Johnson system of transformations addresses this problem (Shah 93).

A distribution can be completely characterized by its moments, when they exist. This suggests that any distribution can be closely approximated by a distribution matching the first few moments. The Johnson system of transformations approximates the first four moments of a random variable, known as the mean, variance, skewness, and kurtosis. The Johnson family of distributions is defined by Equation F-15:

$$R = \gamma + \eta f_i(G; \lambda, \epsilon), \quad i = 1, 2, 3, \quad (\text{F-15})$$

where R is from a standard normal distribution. ϵ is a location parameter, λ is a scale parameter, and γ and η are shape parameters.

The Johnson S_U distribution is defined by

$$f_1(G; \lambda, \epsilon) = \sinh^{-1} \left[\frac{G - \epsilon}{\lambda} \right] \quad (\text{F-16})$$

The Johnson S_B distribution is defined by

$$f_2(G; \lambda, \epsilon) = \ln \left[\frac{G - \epsilon}{\lambda + \epsilon - G} \right], \quad \epsilon < G < \epsilon + \lambda. \quad (\text{F-17})$$

Finally, the Johnson S_L distribution is given by

$$f_3(G; \lambda, \epsilon) = \ln \left[\frac{G - \epsilon}{\lambda} \right], \quad G > \epsilon. \quad (\text{F-18})$$

The Johnson S_L distribution can be reparameterized in terms of γ^* , where

$$\gamma^* = \gamma - \eta \ln(\epsilon). \quad (\text{F-19})$$

The transformation for the Johnson S_L distribution becomes

$$R = \gamma^* + \eta \ln(G - \epsilon), \quad G > \epsilon. \quad (\text{F-20})$$

The Johnson S_L divides the skewness-kurtosis space into two parts. The S_U distribution occupies one of the resulting regions, and the S_B distribution occupies the other. One could decide which member of the Johnson family best fits a random sample based on the sample skewness and kurtosis. Estimates of skewness and kurtosis, however, are highly variable for small samples. A more dependable method has been found based on the heaviness of the tails as compared with the center.

F.2.1 Fitting the Johnson Distribution

Let $G_{(1)} \leq G_{(2)} \leq \dots \leq G_{(n)}$ be an observed ordered sample. The problem is to fit G with a Johnson distribution. Consider any given positive number r (0.775449 is used in the Ozturk algorithm). r is used to divide the axis for the normal transformed data into three intervals, each of length $2r$: $(-3r, -r)$, $(-r, r)$, $(r, 3r)$. The Johnson family of transformations maps the observed sample into percentile estimates along the normal axis: $R_{(1)} \leq R_{(2)} \leq \dots \leq R_{(n)}$. Appropriate indices for approximating the endpoints of the given intervals can be found by solving Equation F-21:

$$Pr(R < a) \approx \frac{k_a - 1/2}{n}. \quad (\text{F-21})$$

where

$$a = -3r, -r, r, 3r. \quad (\text{F-22})$$

The solution is

$$k_a = [n Pr(R < a) + 1/2], \quad (\text{F-23})$$

where $[*]$ denotes the nearest integer. Since the Johnson transformations are strictly monotonically increasing, the corresponding percentiles in the untransformed data can be estimated as $g_{k_{-3r}}$, $g_{k_{-r}}$, g_{k_r} , and $g_{k_{3r}}$. Parameters that characterize the tails are given by Equations F-24 through F-26:

$$m = g_{k_{3r}} - g_{k_r}, \quad (\text{F-24})$$

$$l = g_{k_{-r}} - g_{k_{-3r}}, \quad (\text{F-25})$$

$$p = g_{k_r} - g_{k_{-r}}, \quad (\text{F-26})$$

These parameters can be used to decide which member of the Johnson family best characterizes the random sample as follows:

$$\frac{m l}{p^2} > 1 \quad \text{for Johnson } S_U \quad (\text{F-27})$$

$$\frac{m l}{p^2} = 1 \quad \text{for Johnson } S_L \quad (\text{F-28})$$

$$\frac{ml}{p^2} < 1 \text{ for Johnson } S_U \quad (\text{F-29})$$

Since G is a continuous random variable, Equation F-28 will be true with probability zero. Therefore, in practice, one accepts the data is from a Johnson S_U distribution if the left hand side is in a small region around unity.

Parameter estimates for the Johnson S_U distribution are:

$$\eta = \frac{2r}{\cosh^{-1} \left[\frac{1}{2} \left(\frac{m}{p} + \frac{l}{p} \right) \right]} \quad (\text{F-30})$$

$$\gamma = \eta \sinh^{-1} \left[\frac{\frac{l}{p} - \frac{m}{p}}{2 \sqrt{\frac{m}{p} \frac{l}{p} - 1}} \right] \quad (\text{F-31})$$

$$\lambda = \frac{2p \sqrt{\frac{m}{p} \frac{l}{p} - 1}}{\left(\frac{m}{p} + \frac{l}{p} - 2 \right) \sqrt{\frac{m}{p} + \frac{l}{p} + 2}} \quad (\text{F-32})$$

$$\varepsilon = \frac{g_r + g_{-r}}{2} + \frac{p \left(\frac{l}{p} - \frac{m}{p} \right)}{2 \left(\frac{m}{p} + \frac{l}{p} - 2 \right)} \quad (\text{F-33})$$

Parameter estimates for the Johnson S_L distribution are:

$$\eta = \frac{2r}{\ln \left(\frac{m}{p} \right)} \quad (\text{F-34})$$

$$\gamma^* = \eta \ln \left[\frac{\frac{m}{p} - 1}{p \sqrt{\frac{m}{p}}} \right] \quad (\text{F-35})$$

$$\varepsilon = \frac{g_r + g_{-r}}{2} - \frac{p}{2} \frac{\frac{m}{p} + 1}{\frac{m}{p} - 1} \quad (\text{F-36})$$

Parameter estimates for the Johnson S_B distribution are:

$$\eta = \frac{r}{\cosh^{-1} \left[\frac{1}{2} \sqrt{\left(1 + \frac{p}{m} \right) \left(1 + \frac{p}{l} \right)} \right]} \quad (\text{F-37})$$

$$\gamma = \eta \sinh^{-1} \left[\frac{\left(\frac{p}{l} - \frac{p}{m} \right) \sqrt{\left[1 + \frac{p}{m} \right] \left[1 + \frac{p}{l} \right] - 4}}{2 \left[\frac{p}{m} \frac{p}{l} - 1 \right]} \right] \quad (\text{F-38})$$

$$\lambda = \frac{p \sqrt{\left[\left[1 + \frac{p}{m} \right] \left[1 + \frac{p}{l} \right] - 2 \right] - 4}}{\frac{p}{m} \frac{p}{l} - 1} \quad (\text{F-39})$$

$$\varepsilon = \frac{g_r + g_{-r}}{2} - \frac{\lambda}{2} + \frac{p}{2} \frac{\left(\frac{p}{l} - \frac{p}{m} \right)}{\left[\frac{p}{m} \frac{p}{l} - 1 \right]} \quad (\text{F-40})$$

F.2.2 Confidence Regions

A confidence region for Ozturk's statistic can be constructed using the Johnson transformation. The Monte Carlo data for (U_N, V_N) is used to determine the appropriate Johnson transformation. U_N and V_N can be transformed by different members of the Johnson family. The Monte Carlo data is then transformed, and sample means, variances, and the correlation coefficient are calculated in the usual manner from the transformed data. A confidence ellipse is constructed in the transformed Bivariate Normal space. A confidence region in the original space is then formed from inverse Johnson transformations.

***MISSION
OF
ROME LABORATORY***

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

- a. Conducts vigorous research, development and test programs in all applicable technologies;
- b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;
- c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;
- d. Promotes transfer of technology to the private sector;
- e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.