14:52:36 OCA PAD INITIATION - PROJECT HEADER INFORMATION 04/07/89

Active

Project #: E-21-T18 Cost share #: E-21-327 Rev #: 0
Center # : R6583-T18 Center shr #: F6583-T18 OCA file #: 128
Work type : RES
Contract#: F30602-88-D-0025-0018 Mod #: Document : TO
Prime #: Contract entity: GTRC

Subprojects ? : N
Main project #:

Project unit: EE Unit code: 02.010.118
Project director(s):
PARIS D T EE (404)894-2902

Sponsor/division names: AIR FORCE / GRIFFISS AFB, NY
Sponsor/division codes: 104 / 023

Award period: 890329 to 900128 (performance) 900228 (reports)

Sponsor amount	New this change	Total to date
Contract value	34,000.00	34,000.00
Funded	22,000.00	22,000.00
Cost sharing amount		2,444.00

Does subcontracting plan apply ?: Y

Title: PARALLEL COMPUTER ARCHITECTURES FOR ROBUST PHASED ARRAY RADAR SYSTEMS

PROJECT ADMINISTRATION DATA

OCA contact: Brian J. Lindberg 894-4820

Sponsor technical contact	Sponsor issuing office
ROBERT A. SHORE	GERARD J. BROWN/PKRM (315)330-2308
DEPARTMENT OF THE AIR FORCE ROME AIR DEVELOPMENT CENTER/EEAS GRIFFISS AFB, NY 13441-5700	ROME AIR DEVELOPMENT CENTER DIRECTORATE OF CONTRACTING (PKRM) GRIFFISS AFB, NY 13441-5700

Security class (U,C,S,TS) : U
Defense priority rating : DO-A7
Equipment title vests with: Sponsor
NONE PROPOSED OR ANTICIPATED.

ONR resident rep. is ACO (Y/N): Y
GOVT supplemental sheet
GIT

Administrative comments -
DELIVERY ORDER PARTIALLY FUNDS TASK E-9-7093 (MONTANA STATE UNIVERSITY)
THROUGH 9/30/89.

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 01/28/91

Project No. E-21-T18 Center No. R6583-T18

Project Director JOY E B School/Lab ELEC ENGR

Sponsor AIR FORCE/GRIFFISS AFB, NY

Contract/Grant No. F30602-88-D-0025-0018 Contract Entity GTRC

Prime Contract No. ____

Title PARALLEL COMPUTER ARCHITECTURES FOR ROBUST PHASED ARRAY RADAR SYSTEMS

Effective Completion Date 900831 (Performance) 900930 (Reports)

Closeout Actions Required:	Y/N	Date Submitted
Final Invoice or Copy of Final Invoice	Y	
Final Report of Inventions and/or Subcontracts	Y	
Government Property Inventory & Related Certificate	Y	
Classified Material Certificate	Y	
Release and Assignment	Y	
Other ____	N	

Comments ____

Subproject Under Main Project No. ____

Continues Project No. ____

Distribution Required:

Project Director	Y
Administrative Network Representative	Y
GTRI Accounting/Grants and Contracts	Y
Procurement/Supply Services	Y
Research Property Managment	Y
Research Security Services	Y
Reports Coordinator (OCA)	Y
GTRC	Y
Project File	Y
Other ____	N
____	N

NOTE: Final Patent Questionnaire sent to PDPI.

CONTRACT FUNDS STATUS REPORT (DD FORM 1586)
CONTRACT NUMBER F30602-88-D-0025
QUARTER: MAY-JUN '88

	CURRENT QUARTER FUNDING		$0.00
	CURRENT QUARTER EXPENDITURES		$0.00
	CONTRACT CEILING		$4,200,000.00
	FUNDING TO DATE	-	$0.00
*	PENDING COMMITMENTS	-	$766,000.00
	AVAILABLE FUNDING		$3,434,000.00
	FUNDING TO DATE		$0.00
	YTD EXPENDITURES	-	$0.00
	OUTSTANDING EXPENDITURES		$0.00

*	C-8-2120 WESTINGHOUSE/BEAUDET	$56,000.00
	C-8-2129 RENSSELAER/DAS	$100,000.00
	E-8-7066 UNIV OF PENN/STEINBERG	$100,000.00
	E-8-7124 BOSTON COLLEGE/McFADDEN	$35,000.00
	E-8-7125 BRANDEIS UNIV/HENCHMAN	$23,000.00
	E-8-7126 PENN STATE/CASTLEMAN	$22,000.00
	A-8-1631 UNIV OF PENN/STEINBERG	$100,000.00
	B-8-3617 GA WASHINGTON UNIV/MELTZER	$100,000.00
	B-8-3618 GA WASHINGTON UNIV/BERKOVICH	$100,000.00
	C-8-2492 GA TECH/SMITH	$50,000.00
	A-8-1203 GA TECH/HUGHES	$80,000.00
	TOTAL PENDING	$766,000.00

CONTRACT FUNDS STATUS REPORT (DD FORM 1586)
CONTRACT NUMBER F30602-88-D-0025
QUARTER: JUL-SEPT '88

CURRENT QUARTER FUNDING				$698,034.00
DO #	0001	$56,000		
	0002	$95,141		
	0003	$78,854		
	0004	$230,000		
	0005	$45,561		
	0006	$25,000		
	0007	$20,000		
	0008	$98,374		
	0009	$29,403		
	0010	$19,701		

		$698,034		
CURRENT QUARTER EXPENDITURES				$0.00
CONTRACT CEILING				$4,200,000.00
FUNDING TO DATE			-	$698,034.00
* PENDING COMMITMENTS			-	$426,563.00

AVAILABLE FUNDING				$3,075,403.00
FUNDING TO DATE				$698,034.00
YTD EXPENDITURES			-	$0.00

OUTSTANDING EXPENDITURES				$698,034.00

* DO # 0001	INCREMENTAL FUNDING	$90,729.00
0002	INCREMENTAL FUNDING	$66,680.00
0003	INCREMENTAL FUNDING	$54,154.00
0004	INCREMENTAL FUNDING	$20,000.00
C-8-2400	STATE UNIV OF NY/FAM	$95,000.00
C-8-2402	RENSSELAER/SAULNER	$100,000.00

	TOTAL PENDING	$426,563.00

CONTRACT FUNDS STATUS REPORT (DD FORM 1586)
CONTRACT NUMBER F30602-88-D-0025
QUARTER: OCT-DEC '88

CURRENT QUARTER FUNDING			$120,834.00
DO # 0004	$66,680		
0006	$54,154		

	$120,834		

CURRENT QUARTER EXPENDITURES		$28,740.82
CONTRACT CEILING		$4,200,000.00
FUNDING TO DATE	-	$818,868.00
* PENDING COMMITMENTS	-	$784,729.00

AVAILABLE FUNDING		$2,596,403.00
FUNDING TO DATE		$818,868.00
YTD EXPENDITURES	-	$28,740.82

OUTSTANDING EXPENDITURES		$790,127.18

*	DO # 0001	INCREMENTAL FUNDING	$90,729.00
	0007	INCREMENTAL FUNDING	$20,000.00
	C-8-2400	STATE UNIV OF NY/FAM	$95,000.00
	C-8-2402	RENSSELAER/SAULNER	$100,000.00
	B-9-3592	UNIV OF CA/DAVIS/LEVITT	$60,000.00
	N-9-5514	SOHAR INC./HECHT	$50,000.00
	C-9-2015	NCS/O'NEAL	$100,000.00
	A-9-1120	HITEC, INC./KAZAKOS	$75,000.00
	E-9-7057	UNIV OF TX/ARLINGTON/FUNG	$40,000.00
	E-9-7093	MONTANA STATE/JOHNSON	$34,000.00
	S-9-7552	ALFRED UNIV/SYNDER	$20,000.00
	C-9-2404	STANFORD UNIV/WIDROW	$100,000.00

		TOTAL PENDING	$784,729.00

CONTRACT FUNDS STATUS REPORT (DD FORM 1586)
CONTRACT NUMBER F30602-88-D-0025
QUARTER: JAN-MAR '89

CURRENT QUARTER FUNDING			$574,457.00
DO # 0001	$90,729		
0011	$75,000		
0012	$75,000		
0013	$59,989		
0014	$49,989		
0015	$70,000		
0016	$43,750		
0017	$30,000		
0018	$22,000		
0019	$38,000		
0020	$20,000		

	$574,457		

CURRENT QUARTER EXPENDITURES		$86,324.15
CONTRACT CEILING		$4,200,000.00
FUNDING TO DATE	-	$1,393,325.00
* PENDING COMMITMENTS	-	$594,651.00

AVAILABLE FUNDING		$2,212,024.00
FUNDING TO DATE		$1,393,325.00
YTD EXPENDITURES	-	$115,064.97

OUTSTANDING EXPENDITURES		$1,278,260.03

* DO # 0007	INCREMENTAL FUNDING	$20,000.00
0011	INCREMENTAL FUNDING	$19,568.00
0012	INCREMENTAL FUNDING	$24,700.00
0015	INCREMENTAL FUNDING	$29,783.00
0016	INCREMENTAL FUNDING	$31,250.00
0017	INCREMENTAL FUNDING	$10,000.00
0018	INCREMENTAL FUNDING	$12,000.00
0019	INCREMENTAL FUNDING	$12,000.00
C-8-2404	STANFORD UNIV/WIDROW	$100,000.00
N-9-5732	GRIFFIN	$25,000.00
A-9-1476	BOWDOIN COLLEGE/CHONACKY	$20,350.00
E-9-7110	UNIV OF LOWELL/SALES	$50,000.00
S-9-7559	UNIV OF MICHIGAN/ROBINSON	$20,000.00
B-9-3621	SRI/LUNT	$20,000.00
N-9-5308	KAMAN SCIENCES	$100,000.00
E-9-7119	DARTMOUTH COLLEGE/CRANE	$100,000.00

	TOTAL PENDING	$594,651.00

CONTRACT FUNDS STATUS REPORT (DD FORM 1586)
CONTRACT NUMBER F30602-88-D-0025
QUARTER: APR-JUN '89

CURRENT QUARTER FUNDING			$160,350.00
DO # 0021	$25,000		
0022	$45,000		
0023	$20,350		
0024	$50,000		
0025	$20,000		

	$160,350		

CURRENT QUARTER EXPENDITURES		$318,963.82

	CONTRACT CEILING		$4,200,000.00
	FUNDING TO DATE	-	$1,553,675.00
*	PENDING COMMITMENTS	-	$718,994.00

	AVAILABLE FUNDING		$1,927,331.00

FUNDING TO DATE		$1,553,675.00
YTD EXPENDITURES	-	$434,028.79

OUTSTANDING EXPENDITURES		$1,119,646.21

*	DO # 0007	INCREMENTAL FUNDING	$20,000.00
	0011	INCREMENTAL FUNDING	$19,568.00
	0012	INCREMENTAL FUNDING	$24,700.00
	0015	INCREMENTAL FUNDING	$29,783.00
	0016	INCREMENTAL FUNDING	$31,250.00
	0017	INCREMENTAL FUNDING	$10,000.00
	0018	INCREMENTAL FUNDING	$12,000.00
	0019	INCREMENTAL FUNDING	$12,000.00
	0022	INCREMENTAL FUNDING	$54,693.00
	B-9-3621	SRI/LUNT	$20,000.00
	N-9-5308	KAMAN SCIENCES	$100,000.00
	E-9-7119	DARTMOUTH COLLEGE/CRANE	$100,000.00
	N-9-5740	CHRISTIANSON	$15,000.00
	N-9-5317	UNIV OF CO/NORGARD	$50,000.00
	S-9-7625	UNIV OF CA/DAVIS/KOWELL	$20,000.00
	N-9-5314	KAMAN SCIENCES	$100,000.00
	N-9-5315	KAMAN SCIENCES	$100,000.00

		TOTAL PENDING	$718,994.00

E-21-T18

CONTRACT FUNDS STATUS REPORT (DD FORM 1586)
CONTRACT NUMBER F30602-88-D-0025
QUARTER: JUL-SEP '89

CURRENT QUARTER FUNDING		$476,000.00
DO # 0017	$10,000	
0026	$15,000	
0027	$20,000	
0028	$50,000	
0029	$40,000	
0030	$30,000	
0031	$20,000	
0032	$66,000	
0033	$70,000	
0034	$85,000	
0035	$70,000	
	$476,000	

CURRENT QUARTER EXPENDITURES	$415,422.69
CONTRACT CEILING	$4,200,000.00
FUNDING TO DATE	- $2,029,675.00
* PENDING COMMITMENTS	- $253,994.00
AVAILABLE FUNDING	$1,916,331.00
FUNDING TO DATE	$2,029,675.00
YTD EXPENDITURES	- $849,451.48
OUTSTANDING EXPENDITURES	$1,180,223.52

* DO # 0007	INCREMENTAL FUNDING	$20,000.00
0011	INCREMENTAL FUNDING	$19,568.00
0012	INCREMENTAL FUNDING	$24,700.00
0015	INCREMENTAL FUNDING	$29,783.00
0016	INCREMENTAL FUNDING	$31,250.00
0018	INCREMENTAL FUNDING	$12,000.00
0019	INCREMENTAL FUNDING	$12,000.00
0022	INCREMENTAL FUNDING	$54,693.00
N-0-5703	UNIV OF SOUTHERN FLA/WILSON	$50,000.00
	TOTAL PENDING	$253,994.00

CONTRACT FUNDS STATUS REPORT (DD FORM 1586)
CONTRACT NUMBER F30602-88-D-0025
QUARTER: OCT-DEC '89

CURRENT QUARTER FUNDING				$292,994.00
DO #	0001	$9,000	C-8-2129	
	0011	$19,568	C-8-2400	
	0012	$24,700	C-8-2402	
	0015	$29,783	C-9-2015	
	0016	$31,250	A-9-1120	
	0018	$12,000	E-9-7093	
	0019	$62,000	C-9-2109	
	0022	$54,693	C-9-2404	
	0028	$50,000	N-9-5308	

		$292,994		

CURRENT QUARTER EXPENDITURES	$286,691.16

CONTRACT CEILING	$4,200,000.00
FUNDING TO DATE	- $2,322,669.00
* PENDING COMMITMENTS	- $595,000.00

AVAILABLE FUNDING	$1,282,331.00

FUNDING TO DATE	$2,322,669.00
YTD EXPENDITURES	- $1,136,142.64

OUTSTANDING EXPENDITURES	$1,186,526.36

* DO # 0007	S-8-7592	INCREMENTAL FUNDING	$20,000.00
0029	E-9-7119	INCREMENTAL FUNDING	$60,000.00
0030	N-9-5317	INCREMENTAL FUNDING	$20,000.00
0034	N-9-5314	INCREMENTAL FUNDING	$15,000.00
0016	N-9-5315	INCREMENTAL FUNDING	$30,000.00
N-0-5703	UNIV OF SOUTHERN FLA/WILSON		$50,000.00
A-0-1102	UNIV OF CA/SMOOT, BARBER, GT		$100,000.00
P-0-6011	NCSU/VANDERLUGT		$100,000.00
C-0-2456	NEW JERSEY INST/BAR-NESS		$100,000.00
P-0-6014	STEVENS INST/ZMUDA		$100,000.00

	TOTAL PENDING		$595,000.00

WAITING FOR PROPOSALS:	P-0-6018	UAH/CAULFIELD
	P-0-6021	GT/SUMNERS
	P-0-6022	CORNELL UNIV/TANG
	B-0-3353	ROCHESTER INST/LASKY

CONTRACT FUNDS STATUS REPORT (DD FORM 1586)
CONTRACT NUMBER F30602-88-D-0025
QUARTER: JAN-MAR '90

CURRENT QUARTER FUNDING				$114,301.00
DO # 0007	$9,000	S-8-7592		
0029	$19,568	E-9-7119		
0030	$24,700	N-9-5317		
0036	$29,783	P-0-6014		
0037	$31,250	P-0-6011		
	$114,301			

CURRENT QUARTER EXPENDITURES		$376,743.62
CONTRACT CEILING		$4,200,000.00
FUNDING TO DATE	-	$2,436,970.00
* PENDING COMMITMENTS	-	$532,800.00
AVAILABLE FUNDING		$1,230,230.00
FUNDING TO DATE		$2,436,970.00
YTD EXPENDITURES	-	$1,512,886.26
OUTSTANDING EXPENDITURES		$924,083.74

* DO# 0034	N-9-5314	INCREMENTAL FUNDING	$15,000.00
0035	N-9-5315	INCREMENTAL FUNDING	$30,000.00
0037	P-0-6011	INCREMENTAL FUNDING	$10,000.00
N-0-5703	UNIV OF SOUTHERN FLA/WILSON		$50,000.00
A-0-1402	UNIV OF CA/SMOOT, BARBER, GT		$100,000.00
C-0-2456	NEW JERSEY INST/BAR-NESS		$100,000.00
P-0-6021	GT/SUMNERS		$100,000.00
P-0-6022	CORNELL UNIV/TANG		$30,800.00
B-0-3353	ROCHESTER INST/LASKY		$20,000.00
P-0-6018	UAH/CAULFIELD		$77,000.00
	TOTAL PENDING		$532,800.00

WAITING FOR PROPOSALS: P-0-6018 UAH/CAULFIELD
P-0-6021 GT/SUMNERS
P-0-6022 CORNELL UNIV/TANG
B-0-3353 ROCHESTER INST/LASKY

E-21-T18

ROME AIR DEVELOPMENT CENTER
EXPERT SCIENCE AND ENGINEERING PROGRAM
CONTRACT NO. F30602-88-D-0025

R & D STATUS REPORT

PERIOD COVERED: March 29, 1989 - Sept. 30, 1989

TASK NUMBER: E-9-7093

TITLE: Analysis of Parallel Computer Architecture Algorithms for Robust Phased Array Radar Systems

PRINCIPAL INVESTIGATOR: Roy Johnson

INSTITUTION: Dept. of Elect. Engr., Montana State University

OTHER PARTICIPANTS AND TITLES:

Mr. David Virag - graduate student

A. TECHNICAL PROGRESS ACHIEVED ON EFFORT:

1.) Literature has been reviewed

2.) Griffith Algorithm for Pattern Adaption has been programmed.

3.) Effect of Element failures on resulting pattern have been simulated and evaluated quantitatively

4.) Griffith Algorithm has been modified to satisfy a priori constraints imposed by failures and new performance simulated

5.) Report covering progress submitted to Robert Shore - RADC Engineer for evaluation

B. TRAVEL: None

C. PRESENTATIONS AND PUBLICATIONS:

Major Report "An Adaptive Algorithm for Weight Adjustment Due To Random Element Failure," has been forwarded to Robert A. Shore for review.

D. LEVEL OF EFFORT BY EACH CONTRIBUTOR (IN MAN-MONTHS OR MAN-HOURS)

Roy Johnson 75 hours
David Viiay 375 hours

E-21-T-18

MONTANA STATE UNIVERSITY

Electronics Research Laboratory

BOZEMAN, MONTANA

A Reduced Order Generalized Sidelobe Canceller Algorithm For Random Element Failure Compensation

Final Technical Progress Report

by

David Virag and Roy Johnson
Department of Electrical Engineering
Montana State University
Bozeman, MT 59717-0007
(406)-994-4271

Project:

Analysis of Parallel Computer Architecture Algorithms For Robust Phased Array Radar Systems

ERL REPORT 290151-1

Contract: F30602-88-D-0025
Georgia Institute of Technology
Subcontract No.: E-21-T18-S1
Department of Electrical Engineering
Montana State University
A Research Project Sponsored by
United States Department of the Airforce
Rome Air Development Center
Griffiss Air Force Base
New York 13441-5700

A Reduced Order Generalized Sidelobe Canceller Algorithm For Random Element Failure Compensation

D. E. Virag

Montana State University

ABSTRACT

This paper demonstrates a method of reducing an adaptive array algorithm to accommodate for random catastrophic element failures in an equally spaced linear narrow-band adaptive array. The Generalized Sidelobe Canceller (GSC) algorithm is reduced by the number of failed elements in the array. The reduced Generalized Sidelobe Canceller algorithm is shown to satisfy desired constraints when the number of active elements is greater than the total number of constraints. Several examples are presented showing improvement of the modified GSC algorithm over the original algorithm when failures are present. Computational considerations are discussed when the reduced GSC algorithm is used. Several suggestions are given for logical research extension relative to the reduced GSC algorithm.

INTRODUCTION

Adaptive array algorithms have been studied extensively over the past 30 years [1]-[4]. Several techniques have been shown to be effective for narrowband spatial filtering in the presence of noise and interference signals. While the general adaptive array problem is well formulated, little work has been done on array adaptation in the face of catastrophic element failure. Element failure effectively transforms a linear equally-spaced array into an unequally-spaced array. Typical research approaches have been directed at using search techniques to optimize randomly spaced arrays. A methodical process for optimizing the weight vector given the known information of the array structure should be available since a linear array with element failures demonstrates some regularity. The Generalized Sidelobe Canceller, derived by Griffiths *et al.* [5]-[7], provides a vehicle for dealing with these element failures when the failed element positions are known a priori.

OPTIMUM ARRAY PROCESSING

Fig. 1 describes the block diagram of the typical array processing system. Given the general signal environment statistics, determine the optimal set of weights W_o so that the output $y(n)$ is optimal in some sense to the desired input $d(n)$ in the presence of both uncorrelated white noise and jamming signals.

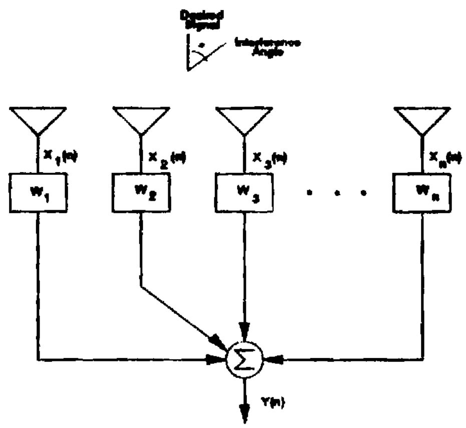

Figure 1.

The actual signal measured at the array is $x(t) = d(t) + n(t) + i(t)$. where $d(t)$ is the actual desired signal, $n(t)$ is the noise present which is assumed to be Gaussian and $i(t)$ is a signal associated with jamming interference. All of the second order statistics are assumed to be stationary. The three input signals are assumed to be uncorrelated with respect to each other, therefore $R_{xx}(t) = R_{dd}(t) + R_{nn}(t) + R_{ii}(t)$, where

$$R_{dd}=\begin{bmatrix} E[d(1)d(1)] & E[d(1)d(2)] & \dots & E[d(1)d(n)] \\ E[d(2)d(1)] & E[d(2)d(2)] & \dots & E[d(2)d(n)] \\ . & . & . & . \\ . & . & . & . \\ E[d(n)d(1)] & \dots & \dots & E[d(n)d(n)] \end{bmatrix}=\begin{bmatrix} \sigma_d^2 e^{j(1-1)} & \sigma_d^2 e^{j(2-1)} & \dots & \sigma_d^2 e^{j(n-1)} \\ \sigma_d^2 e^{j(1-2)} & \sigma_d^2 e^{j(2-2)} & \dots & \sigma_d^2 e^{j(n-2)} \\ . & . & . & . \\ . & . & . & . \\ \sigma_d^2 e^{j(n-1)} & \sigma_d^2 e^{j(n-2)} & \dots & \sigma_d^2 e^{j(n-n)} \end{bmatrix}$$

$$R_{nn}=\begin{bmatrix} E[n(1)n(1)] & E[n(1)n(2)] & \dots & E[n(1)n(n)] \\ E[n(2)n(1)] & E[n(2)n(2)] & \dots & E[n(2)n(n)] \\ . & . & . & . \\ . & . & . & . \\ E[n(n)n(1)] & \dots & \dots & E[n(n)n(n)] \end{bmatrix}=\begin{bmatrix} \sigma_n^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_n^2 & 0 & \dots & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & \dots & 0 & 0 & \sigma_n^2 \end{bmatrix}$$

$$R_{ii}=\begin{bmatrix} E[i(1)i(1)] & E[i(1)i(2)] & \dots & E[i(1)i(n)] \\ E[i(2)i(1)] & E[i(2)i(2)] & \dots & E[i(2)i(n)] \\ . & . & . & . \\ . & . & . & . \\ E[i(n)i(1)] & \dots & \dots & E[i(n)i(n)] \end{bmatrix}=\begin{bmatrix} \sigma_i^2 e^{j(1-1)\alpha} & \sigma_i^2 e^{j(2-1)\alpha} & \dots & \sigma_i^2 e^{j(n-1)\alpha} \\ \sigma_i^2 e^{j(1-2)\alpha} & \sigma_i^2 e^{j(2-2)\alpha} & \dots & \sigma_i^2 e^{j(n-2)\alpha} \\ . & . & . & . \\ . & . & . & . \\ \sigma_i^2 e^{j(n-1)\alpha} & \sigma_i^2 e^{j(n-2)\alpha} & \dots & \sigma_i^2 e^{j(n-n)\alpha} \end{bmatrix}$$

where:

$$\alpha = 2\pi d \sin(\theta)$$

d is the fractional wavelength distance between elements at the desired frequency, and σ_i^2 is the variance of the ith signal. The matrix formulation for R_{dd} assumes the desired signal is broadside to the array while matrix form R_{ii} assumes interference from angle θ measured from broadside to the array.

If minimum output power is the desired optimization condition, we have,

$$\min_W\{E[(XW)^H XW]\} = \min_W\{W^H R_{xx} W\} \tag{1}$$

which has the solution

$$W_{opt} = R_{xx}^{-1} r_{xd} \tag{2}$$

where:

$$r_{xd} = \begin{bmatrix} E[x(1)d(1)] \\ E[x(2)d(2)] \\ \cdot \\ \cdot \\ \cdot \\ E[x(n)d(n)] \end{bmatrix} \tag{3}$$

This solution for the optimal weight vector is effective when the signal to noise ratio (SNR) is low. If the desired signal power is large compared to the interference and noise, this algorithm will penalize the signal by minimizing the overall power, therefore reducing the SNR.

CONSTRAINED ARRAY PROCESSING

The inclusion of constraints in the formulation of the optimization problem leads to more control over desired beam patterns in addition to the ability to null out known interference signals. The constraint equations are formulated as:

$$C^H W = f \tag{4}$$

where:

$C \Leftrightarrow$ Constraint Matrix

$f \Leftrightarrow$ Forcing Vector

If the desired signal is broadside to the array and is required to have 0 dB gain, and L interference signals from L directions are to be nulled out, the C matrix will take the form

$$C = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{j\alpha_1} & \cdots & e^{j\alpha_L} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1 & e^{j(N-1)\alpha_1} & \cdots & e^{j(N-1)\alpha_L} \end{bmatrix} \tag{5}$$

where $\alpha_i = 2\pi d \sin(\theta_i)$ and θ_i is the direction of the *i*th interference signal.

C is an N x L matrix, where N is the number of elements and L is the number of desired constraints.

The response vector f is

$$f = [1\ 0\ 0\ \dots\ 0]^T \tag{6}$$

where the 1 represents 0db gain at broadside and the zeros correspond to 0 gain at the desired null angles.

The constrained optimal weight vector is now that which is closest to the unconstrained vector and satisfies the constraints given. The performance measure to minimize can be given from (2) as

$$\mathrm{Min}_W \{R_{xx} W - r_{xd}\} \tag{7a}$$

subject to:

$$C^H W = f \tag{7b}$$

which has the solution

$$W_{opt} = R_{xx}^{-1} C (C^H R_{xx}^{-1} C)^{-1} f \tag{8}$$

GENERALIZED SIDELOBE CANCELLER

An equivalent representation of the constrained array problem was derived by Griffiths. The Generalized Sidelobe Canceller (GSC) is a transformation of the general constrained optimization problem into a deterministic weight equation and an unconstrained optimal weight equation. This topology is more convenient for implementation due to its reduced number of computations. Fig. 2 shows a block diagram for the basic GSC.

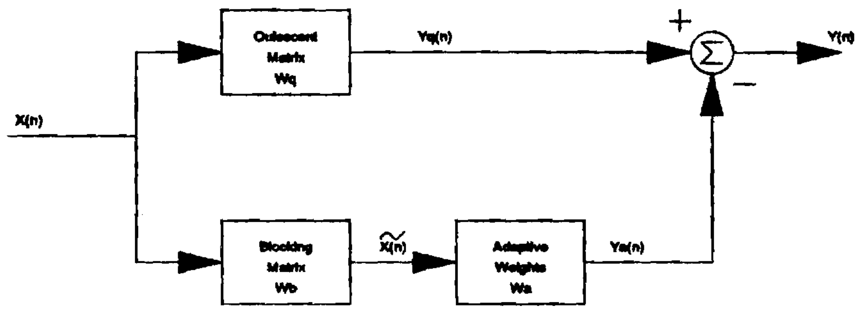

Figure 2.

The parallel paths of the GSC represent the quiescent weights and adaptive weights. The quiescent weights W_q represent the general solution to the adaptive array problem with constraints and will give the desired pattern given only noise on the input elements. The constraints include array pattern, array sidelobe specifications, and spatial nulling. The lower path contains the data dependant adaptive weights, W_a, which are statistically determined for a mean squared error (MSE) approximation (neglecting constraints) to the upper quiescent weights.

The weight vector from the general constrained array problem can be decomposed into two orthogonal spaces which span the range and null space of C [8]. Any desired weight vector can be represented as the sum of the two orthogonal vectors, one vector spanning the range space of C, the other spanning the null space of C. The weight vector in the null space of the constraint matrix will have no effect on the desired constraints and can be optimized accordingly.

Let $W = W_q - V$ where W_q is in the range of C and V is in the null space of C. From Fig. 2, $V = W_b W_a$, also $W_q = C(C^H C)^{-1} f$ to satisfy the constraint equations. Therefore, to minimize total output power, we get

$$\mathrm{Min}_{W_a}\{(W_q - W_b W_a) R_{xx} (W_q - W_b W_a)^H\} \tag{9}$$

The optimal W_a can be solved using classical optimization techniques. Define the performance measure J as

$$J = [W_q - W_b W_a]^H R_{xx} [W_q - W_b W_a] \tag{10a}$$

$$= W_q^H R_{xx} W_q - W_q^H R_{xx} W_b W_a - W_a^H W_b^H R_{xx} W_q + W_a^H W_b^H R_{xx} W_b W_a \tag{10b}$$

The second and third terms of (10b) are scalars and can be combined since the transpose of a scalar is unaffected. Taking the gradient of J with respect to W_a gives

$$\nabla_{w_a} J = 0 = -2W_q^H R_{xx} W_b + 2W_b^H R_{xx} W_b W_a \tag{11}$$

Solving (11) for W_a we find the optimal weight vector as

$$W_a^{opt} = (W_b^H R_{xx} W_b)^{-1} W_b^H R_{xx} W_q \tag{12}$$

An equivalent expression for the optimal weight vector is

$$W_a^{opt} = R_{\xi\xi}^{-1} P_{\xi y_q} \tag{13}$$

(13) can easily be obtained from (12) since $R_{\xi\xi} = W_b^H R_{xx} W_b$ and $P_{\xi_q} = W_b^H R_{xx} W_q$.

While the quiescent weight vector $W_q = C(C^H C)^{-1} f$ is the solution for the general deterministic constraint problem, it is not necessarily the best due to large sidelobes close to the main lobe. If a weighting vector such as a Chebyshev pattern is desired [9], the quiescent weights must be modified.

Consider the desired Chebyshev weight vector W_{cheb}. Clearly W_{cheb} will not satisfy the constraint equation (4). A weight vector is required which satisfies the constraints of (4) and is close to the desired vector W_{cheb}. This is mathematically equivalent to

$$\mathrm{Min}_{\overline{W_q}}\left\{(\overline{W_q}-W_{cheb})^H(\overline{W_q}-W_{cheb})\right\} \tag{14a}$$

Subject to:

$$C^H\overline{W_q}=f \tag{14b}$$

where $\overline{W_q}$ is the modified weight vector to be determined. The performance measure J can be formed as

$$J=[\overline{W_q}-W_{cheb}]^H[\overline{W_q}-W_{cheb}]+\lambda(f-C^H\overline{W_q}) \tag{15a}$$

$$=\overline{W_q}^H\overline{W_q}-\overline{W_q}^HW_{cheb}-W_{cheb}^H\overline{W_q}+W_{cheb}^HW_{cheb}+\lambda f-\lambda C^H\overline{W_q} \tag{15b}$$

The gradient of J with respect to $\overline{W_q}$ is given by

$$\nabla_{\overline{W_q}}J=0=2\overline{W_q}-2W_{cheb}-(\lambda C^H)^H \tag{16}$$

Solving (16) for $\overline{W_q}^{opt}$

$$\overline{W_q}^{opt}=W_{cheb}+\frac{1}{2}C\lambda^H \tag{17}$$

Using (4) and (17)

$$\lambda^H=2(C^HC)^{-1}f-2(C^HC)^{-1}C^HW_{cheb} \tag{18}$$

Substituting (18) back into (17) gives the optimal modified weight vector

$$\overline{W_q}^{opt} = (I - C(C^H C)^{-1})W_{cheb} + W_q \tag{19}$$

The modification of W_q alone will not change the overall quiescent pattern until an appropriate change is also made to the constraint matrix. This can be seen by the fact that $\overline{W_q}$ now contains components in the range space of C and in the null space of C. The addition of the component in the null space of C must be included in the constraint equations so that those components will be effectively canceled by the blocking matrix W_b. This modification adds one more column to C so that

$$\overline{C} = [C, W_s] \tag{20}$$

$$\bar{f}^H = \left[f^H, \overline{W_s}^H \overline{W_q}\right] \tag{21}$$

where:

$$W_s = \overline{W_q} - W_q \tag{22}$$

The new constraint matrix $\overline{C}$ is now N x M and $\bar{f}$ is an M vector where M=L+1.

The N x (N-M) blocking matrix W_b eliminates the constraint equations so that an unconstrained optimization can take place. The requirements of the W_b matrix are that:

1.) The columns of W_b are linearly independent

2.) W_b is orthogonal to C

A number of different blocking matrices will be adequate for any given problem. Gram-Schmidt orthogonalization procedures can be used to ensure that the matrices are orthogonal. For simulation purposes, one particular algorithm for determining the W_b matrix is:

$a_{i,j} = 1;$ $\{i = 1, j; \quad j = 1, N-M\}$

$a_{i,j} =$ unknown coefficient; $\{i = j+1, j+N-M; \quad j = 1, N-M\}$

$a_{i,j} = 0;$ otherwise.

This matrix will structurally take the form

$$W_b = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ a_{1,1} & 1 & 0 & \dots & 0 \\ a_{1,2} & a_{2,1} & 1 & \dots & 0 \\ . & a_{2,2} & a_{3,1} & \dots & 0 \\ . & . & a_{3,2} & \dots & 0 \\ a_{1,M} & . & . & \dots & 0 \\ 0 & a_{2,M} & . & \dots & 1 \\ 0 & 0 & a_{3,M} & \dots & a_{N-M,1} \\ . & 0 & 0 & \dots & a_{N-M,2} \\ . & . & 0 & \dots & . \\ . & . & . & \dots & . \\ 0 & 0 & 0 & \dots & a_{N-M,M} \end{bmatrix} \tag{23}$$

where the $a_{i,j}$'s are unknown complex elements to be solved through the linear equation

$$[\overline{C}]^H W_b = 0 \tag{24}$$

.

EXAMPLE

Consider an 8 element array in which 2 pattern null constraints, 0 db gain at broadside, and a Chebyshev pattern are desired. The total number of constraints is 4. The constraint matrix and response vector will take the form

$$\overline{C}^T = \begin{bmatrix} C_{1,1} & C_{2,1} & C_{3,1} & C_{4,1} & C_{5,1} & C_{6,1} & C_{7,1} & C_{8,1} \\ C_{1,2} & C_{2,2} & C_{3,2} & C_{4,2} & C_{5,2} & C_{6,2} & C_{7,2} & C_{8,2} \\ C_{1,3} & C_{2,3} & C_{3,3} & C_{4,3} & C_{5,3} & C_{6,3} & C_{7,3} & C_{8,3} \\ W_{c1} & W_{c2} & W_{c3} & W_{c4} & W_{c5} & W_{c6} & W_{c7} & W_{c8} \end{bmatrix} \quad (25)$$

$$\bar{f} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ f_4 \end{bmatrix} \quad (26)$$

The $C_{i,j}$'s are found from (5) and the W_{ci}'s represent the weight vectors given in (22). The appropriate blocking matrix will take the form

$$W_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a_{1,1} & 1 & 0 & 0 \\ a_{1,2} & a_{2,1} & 1 & 0 \\ a_{1,3} & a_{2,2} & a_{3,1} & 1 \\ a_{1,4} & a_{2,3} & a_{3,2} & a_{4,1} \\ 0 & a_{2,4} & a_{3,3} & a_{4,2} \\ 0 & 0 & a_{3,4} & a_{4,3} \\ 0 & 0 & 0 & a_{4,4} \end{bmatrix} \quad (27)$$

Since W_b will be orthogonal to $\overline{C}$, we can solve (24) and get the resulting matrix equation

$$\begin{bmatrix} C_{1,1} & C_{2,1} & C_{3,1} & C_{4,1} & C_{5,1} & C_{6,1} & C_{7,1} & C_{8,1} \\ C_{1,2} & C_{2,2} & C_{3,2} & C_{4,2} & C_{5,2} & C_{6,2} & C_{7,2} & C_{8,2} \\ C_{1,3} & C_{2,3} & C_{3,3} & C_{4,3} & C_{5,3} & C_{6,3} & C_{7,3} & C_{8,3} \\ W_{c1} & W_{c2} & W_{c3} & W_{c4} & W_{c5} & W_{c6} & W_{c7} & W_{c8} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ a_{1,1} & 1 & 0 & 0 \\ a_{1,2} & a_{2,1} & 1 & 0 \\ a_{1,3} & a_{2,2} & a_{3,1} & 1 \\ a_{1,4} & a_{2,3} & a_{3,2} & a_{4,1} \\ 0 & a_{2,4} & a_{3,3} & a_{4,2} \\ 0 & 0 & a_{3,4} & a_{4,3} \\ 0 & 0 & 0 & a_{4,4} \end{bmatrix} = \overline{0} \qquad (28)$$

Performing the corresponding matrix multiplication and solving for the unknown $a_{1,i}$ coefficients,

$$\begin{bmatrix} C_{2,1} & C_{3,1} & C_{4,1} & C_{5,1} \\ C_{2,2} & C_{3,2} & C_{4,2} & C_{5,2} \\ C_{2,3} & C_{3,3} & C_{4,3} & C_{5,3} \\ W_{c2} & W_{c3} & W_{c4} & W_{c5} \end{bmatrix} \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ a_{4,1} \end{bmatrix} = \begin{bmatrix} -C_{1,1} \\ -C_{1,2} \\ -C_{1,3} \\ -W_{c1} \end{bmatrix} \qquad (29)$$

Identical operations can be done to solve for the remaining $a_{i,j}$ coefficients.

EFFECTS OF ELEMENT FAILURES ON GSC PATTERNS

When one or more of the N array elements fail, the overall pattern deteriorates substantially. Figures 3 illustrates a 32 element array with a Dolph-Chebyshev pattern with pattern nulls constrained at $12°, 14°, 16°$, and $18°$ and an actual interference signal at $22°$ with 20 dB power relative to the desired signal. The sidelobes were specified to be at least 35 dB under the main lobe. Figure 4 demonstrates the same array with a single element failure (element # 3). When several elements fail, the pattern deterioration is catastrophic. The need for weight adaptation in the face of array element failure is very apparent.

A REDUCED ORDER GENRALIZED SIDELOBE CANCELLER

Search techniques are typically used in reconfiguring the elements of an array in which one or more array elements have failed. In general, such techniques can be computationally intensive and are not guaranteed to find the globally optimal solution. A more direct approach to restructuring the array weights is to reduce the order of the active array under the same constraints as the original array. This results in optimal array weights under the given constraints and conditions.

Consider the transformation of the original array into the following system

$$W' = TW \tag{30}$$

where T is an N x N permutation matrix defining the condition of the array system. Ideally, if all elements are active, T will be equal to I_N, the N x N identity matrix. If a single element fails in the kth position, T is defined by

$$T = \begin{bmatrix} 0_{1x(k-1)} & | & 1 & | & 0_{1x(N-k)} \\ - & - & - & - & - \\ I_{(k-1)x(k-1)} & | & 0_{(k-1)xl} & | & 0_{(k-1)x(N-k)} \\ - & - & - & - & - \\ 0_{(N-k)x(k-1)} & | & 0_{(N-k)xl} & | & I_{(N-k)x(k-1)} \end{bmatrix} \tag{31}$$

In general, if r element failures exist, the matrix T is formed by moving each of the r rows corresponding to the failed elements to the top r rows of the T permutation matrix.

The transformed quiescent vector, constraint matrix and blocking matrix are given by

$$W'_q = TW_q \tag{32a}$$

$$C' = TC \tag{32b}$$

$$W'_b = TW_b \tag{32c}$$

The transformed system now contains all weights associated with the r failed elements as the first r rows in the weight vectors which can be partitioned into subarrays representing the failed system elements and the remaining active elements.

$$W'_q = \begin{bmatrix} W'_{qa} \\ - \\ W'_{qb} \end{bmatrix} \tag{33a}$$

$$W'_b = \begin{bmatrix} W'_{ba} \\ - \\ W'_{bb} \end{bmatrix} \tag{33b}$$

$$C' = \begin{bmatrix} C'_1 \\ - \\ C'_2 \end{bmatrix} \tag{33c}$$

The partitions W'_{qa}, W'_{ba}, and C'_1

in (33) correspond to the failed elements and the remaining partitions relate to the active elements.

The constraint equation (4) can now be written in terms of the transformed system as

$$C'^H W' = f \tag{34}$$

$$[C'_1 \mid C'_2]\begin{bmatrix} W'_{qa} \\ - \\ W'_{qb} \end{bmatrix} = f \tag{35}$$

or

$$C'_1 W'_{qa} + C'_2 W'_{qb} = f \tag{36}$$

Because the elements associated with the weight set W'_{qa} are failed, these weights will have no effect on the constraint equation (36) and there is no loss of information if the C'_1 matrix is set to zero. The resulting equation gives

$$C'_2 W'_{qb} = f \tag{37}$$

where the unconstrained estimator solution is given by

$$W'^{opt}_{qb} = C'_2 (C'^H_2 C'_2)^{-1} f \tag{38}$$

If the Chebyshev weight constraints are included in the desired solutions the W'_q vector must be modified as before to minimize the distance from the desired Chebyshev weight vector. The performance minimization is given by

$$\text{Min}_{W'_q}\left\{\left(\begin{bmatrix} W'_{qa} \\ - \\ W'_{qb} \end{bmatrix} - W'_{cheb}\right)^H \left(\begin{bmatrix} W'_{qa} \\ - \\ W'_{qb} \end{bmatrix} - W'_{cheb}\right)\right\} \tag{39a}$$

subject to:

$$C'_2 W'_{qb} = f \tag{39b}$$

where

$$W'_{cheb} = \begin{bmatrix} W'_{ca} \\ - \\ W'_{cb} \end{bmatrix} = TW_{cheb} \tag{40}$$

The expanded performance equation is equivalent to

$$\text{Min}\{(W'_{qa_1} - W'_{ca_1})^2 + (W'_{qa_2} - W'_{ca_2})^2 + \ldots + (W'_{qa_r} - W'_{ca_r})^2 +$$

$$(W'_{qb_1} - W'_{cb_1})^2 + \ldots + (W'_{qb_{N-r}} - W'_{cb_{N-r}})^2\} \tag{41}$$

From the above equation, the minimization clearly takes place when the elements of $W'_{qa} = W'_{ca}$ where W'_{ca} are the transformed Chebyshev weights corresponding to the failed elements. The remainder of the minimization is reduced in order to a N-r order minimization where the optimal quiescent solution for W_{qb} is identical to (14)-(19) and is given by

$$\overline{W}'_{qb} = \left(I - C'_2(C'^H_2 C'_2)^{-1} C'^H_2\right) W'_{cb} + W'_{qb} \tag{42}$$

The constraint equations can be augmented to reflect the additional Chebyshev constraint as in (20) and (21). In this case the partitioned constraint matrix will be

$$\overline{C}_1' = \begin{bmatrix} 0_{Mxr} \\ - \\ W'_{sa} \end{bmatrix} \tag{43a}$$

$$\overline{C}_2' = \begin{bmatrix} C'_2 \\ - \\ W'_{sb} \end{bmatrix} \tag{43b}$$

where

$$W'_s = \begin{bmatrix} W'_{sa} \\ - \\ W'_{sb} \end{bmatrix} = TW_s \tag{44}$$

The reduction of the active weights will also reduce the required dimensions of the blocking matrix W_b.

Applying condition 2 of the blocking matrix on the transformed system results in

$$[C'_1 \,|\, C'_2] \begin{bmatrix} W'_{ba} \\ - \\ W'_{bb} \end{bmatrix} = C'_1 W'_{ba} + C'_2 W'_{bb} = \overline{0} \tag{45}$$

Since $C'_2 = \overline{0}$, the blocking condition reduces to

$$C'_2 W_{bb} = \overline{0} \tag{46}$$

The reduced blocking matrix can take the form of (23) with reduced dimensions of (N-r) x (N-r-M). When the Chebyshev constraints are included as a desired constraint, the augment constraint matrix $\overline{C}'$ is used in (45). In this case, $\overline{C}'_1 \neq 0$ because of the augmentation, however, since the submatrix W'_{ba} is associated with the failed elements, the coefficients of W'_{ba} can be set to zero reducing the equation to (46).

The structure of the W'_{bb} submatrix ensures that the total blocking matrix W'_b is linearly independent thus satisfying blocking matrix condition 1.

Because of the reduction in the size of the blocking matrix, the adaptive weights W_a are reduced in number from N-M to N-M-r. The minimization of the weight distance in the two paths of the GSC structure determines the adaptive weights. From equation (9) we have

$$\mathrm{Min}_{W_a}\left\{\left(\begin{bmatrix} W'_{qa} \\ W'_{qb} \end{bmatrix} - \begin{bmatrix} W'_{aa} \\ W'_{ab} \end{bmatrix} W_a\right) R'_{xx} \left(\begin{bmatrix} W'_{qa} \\ W'_{qb} \end{bmatrix} - \begin{bmatrix} W'_{aa} \\ W'_{ab} \end{bmatrix} W_a\right)^H\right\} \tag{47}$$

where

$$R'_{xx} = T^H R_{xx} T \tag{48}$$

The upper partition of the minimization can be removed because $W'_{qa} = \bar{0}$. The lower partition determines the optimal weight values W_a. The optimal weights are given by

$$W_a^{opt} = W'_{bb}(W'^H_{bb} R_{xx} W'_{bb})^{-1} W'^H_{bb} R'_{xx} W'_{bb} \tag{49}$$

The original system can be recovered through the inverse transformation

$$W_q = T^{-1}\hat{W}_q \tag{50a}$$

$$W_b = T^{-1}\hat{W}_b \tag{50b}$$

SIMULATION ISSUES

The original GSC algorithm was programmed on a microVAX 3600 in FORTRAN [10]. Appropriate modifications have been made to accommodate the reduced order algorithm. The initial condition inputs include the total number of elements, the number of failed elements and position of the failed elements in the array, signal to noise level, desired pattern nulls, interference signal strength and directions, and desired mainlobe to sidelobe ratio for a Dolph-Chebyshev array pattern. Pattern nulls represent anticipated directions of interference signals. The anticipated nulls are programmed into the initial quiescent array. The actual jamming signal will modify the signal statistics and will be compensated for in the adaptive weights, producing a null in the direction of the actual interference. The noise variance σ_n^2 was set equal to 1.0 for reference.

Element failure effects are modeled by modifying the correlation matrices for the interference and desired signals, and adjusting the input at the array element as a signal is swept between $-90°$ and $90°$. The correlation matrices are modified by setting the jth row and column to 0 (except the diagonal component equals σ_j^2) for a element failure in the jth position. This assumes the noise in the failed element will be uncorrelated with any other element signal.

The input to a failed element is simulated as 0. Although several cases are possible for a failed element (i.e. signal grounded, +5Vdc, etc.) it is assumed that the known failed element can always be forced to ground state. The only signal processed through the failed element will be Gaussian noise with a variance of 1.0.

Figure 3 represents a 32 element array pattern with SNR=0.0, four desired pattern nulls at 12°, 14°, 16°, and 18°, a 35 dB Dolph-Chebyshev pattern, and a single interference signal located at 22° with a 20dB gain relative to the desired signal. Figure 4 shows the same example after element # 3 has failed. The pattern effectiveness has been reduced significantly with a single failure. Figure 5 shows the pattern obtained from the reduced order GSC algorithm.

If multiple element failures occur, the desired pattern is distorted accordingly. Figure 6 and figure 7 display the failure pattern and compensated pattern respectively for the previous example with elements 2,5,15,17, and 30 failed. The pattern corresponding to the failed array is unrecognizable whereas that of the compensated array matches all the desired constraints at the cost of the pattern floor raising.

If the event that array end elements fail, the array can be compensated for with the element failure algorithm, or as an alternative, the array can be reconfigured as a smaller array. Figure 8 demonstrates the above example with elements 1-6 failed in the 32 element array. Figure 9 shows the same desired pattern with an unfailed 26 element array. Both configurations meet the desired pattern null and adaptive null constraints. The reconfigured 26 element array matches the Dolph-Chebyshev pattern better than the failure compensated array. The 3dB main beam width is slightly smaller in the 32 element array even though 6 elements are failed.

Figures 10-16 demonstrate a 64 element array example. A 45 dB Dolph-Chebyshev pattern is desired with an anticipated signal at 8° and 2 actual jamming signals appear at 12° and 19° with 40 dB gain. The SNR is 0. Figures 10-12 represent a full array, failed uncompensated array, and failed array using the reduced GSC respectively. In this example, 15 simultaneous element failures were simulated (elements 29-43). The failed uncompensated array does not match the desired constraints of 0 dB gain at broadside and pattern null at 8°. The compensated array matches these constraints at the expense of slightly higher sidelobes. Figures 13 and 14 show the quiescent weight vector magnitude for the full array and fail compensated array respectively. The reconfigured weights tend to reflect the same Chebyshev pattern as the original weights with a slightly higher gain to compensate for the gain lost due to the failed elements. The failed element gains are set to the original weight vector gains. These values, although meaningless because of the failed elements, are necessary for the other weights to adjust accordingly. Figures 15 and 16 show the quiescent weight phase for the full array and compensated array. These figures reflect the increase in phase variations needed to match the desired constraints.

COMPUTATIONAL REQUIREMENTS FOR THE FAILURE COMPENSATED GSC

Upon determination of a failed element or elements, the GSC routine must be modified to compensate for the failure. The major recalculations include updating the blocking matrix W_b, and the two weight vectors W_q and W_a. These calculations may be done either on-line, pre-calculated off-line and stored in memory, or some combination of these.

Off-line Calculation and Storage

Off-line pre-computation and storage requires that every possible combination of failed elements be determined and weight vectors and blocking matrix calculated for each scenario. The possible number of combinations of failed elements, assuming parameters such as pattern nulls and desired pattern sidelobe strength are constant, become massively large. As an example, if N=32, there are approximately 1.28 billion total combinations of failures of 1 element to 14 elements. If 32 weights are stored, each weight requiring 4 bytes, the total memory required is 164 GBytes. These figures do not include memory required to store the blocking matrix coefficients. From these results, it is obvious that 100% data storage is not possible given the current storage devices available.

On-line Computation

The recalculation of the GSC weights consists of determining the correct transformation matrix, T, recomputing the blocking matrix elements, and calculating the weight vectors W_q and W_a. Recalculation of the blocking matrix coefficients requires that (N-M-r) matrix inverses of size M x M be calculated. If M is large, these matrix inverses may take considerable time, depending on the hardware used for the GSC algorithm. The weight vector W_a requires a single (N-M-r) x (N-M-r) matrix inverse and five (N-M-r) x (N-M-r) matrix multiplications using the direct solution method (i.e. R_{xx} known a priori). The quiescent weight vector calculation requires one (N-r) x (N-r) matrix inverse. The advantage to the reduced order GSC is that the total number of computations required to determine new weights when elements fail decreases with increasing numbers of element failures. Table 1 shows various calculation times on a microVAX 3600 for computing T, W_b, W_q, and W_a under several conditions of element failures for a system with 6 total desired constraints.

Elements	Failures	Time (s)
96	0	137.85
96	48	90.89
64	0	43.4
64	32	26.5
32	0	7.26
32	16	4.98
16	0	2.01

Table 1.

FUTURE RESEARCH DIRECTIONS

Extended research on the modified GSC and element failures can be applied in several directions. Some of the more important questions to be answered include:

1. Using the direct form of calculation for computing $W_{opt} = R_{xx}^{-1} r_{xd}$ involves an N X N matrix inversion. If high sampling rates are required, this is computationally very difficult. Also, assuming stationarity, N(N+3)/2 correlation measurements may be required before R_{xx} and r_{xd} are statistically accurate, whereas, in real practice, the statistical environment is usually changing. An adaptive algorithm must be incorporated to estimate the statistical environment more efficiently with fewer calculations.

2. Element failure can show up in various forms, including partial element failure (degradation of element gain). The algorithm for failure correction may be expanded to include

modification for partial element failure.

3. A method of on-line modification of the GSC for element failure must be found to gracefully modify the predefined constraint matrix and blocking matrix structures so adaptation time is minimized. Methods of augmenting a small off-line database with minimal on-line calculations must be studied to truly optimize the total reconfiguration time.

4. Parallel processing structures must be studied to determine the optimal computational distribution so the GSC and corresponding failure routines can be performed on-line in a real-time environment.

5. Array output sensitivity to the weight vector must be studied for large numbers of element failures. The off-line storage requirements may be reduced considerably if the output is robust with respect to weight values when large numbers of element failures are present.

6. Multi-array modes should be studied as an option for restructuring the array in the event of element failure. For example, in special cases, the array may be divided into equally spaced sub-arrays where the GSC is applied and the total array output is weighted sums of the sub-array outputs.

7. Methods for determining on-line element characterization must be found so that failed elements can be identified. For example, a near-field test could be conducted periodically which correlates the array output with a pre-stored pattern to determine if failed or deteriorated elements exist.

8. The comparison of the original weight vectors with the reconfigured weight vectors after element failures suggest that predictions may be made to the relative importance of array elements in the overall pattern robustness. In certain cases, total weight reconfiguration may not be necessary.

CONCLUSIONS

This paper demonstrates a technique for adapting to random element failures in a linear narrow band array. A reduced order Generalized Sidelobe Canceller algorithm was shown to include constraints associated with the failed elements. Simulation results show an improvement in main beam isolation when using the corrector algorithm of greater than 30dB over that of an uncorrected array with a single element failure. An added feature of the reduced order algorithm is that the computational requirements for recomputing array weights reduces as the number of element failures increase. Multiple element failure examples show the reduced order algorithm matches the linear constraint conditions and the general shape of the desired beam pattern at the expense of a decrease in the overall mainlobe to sidelobe isolation.

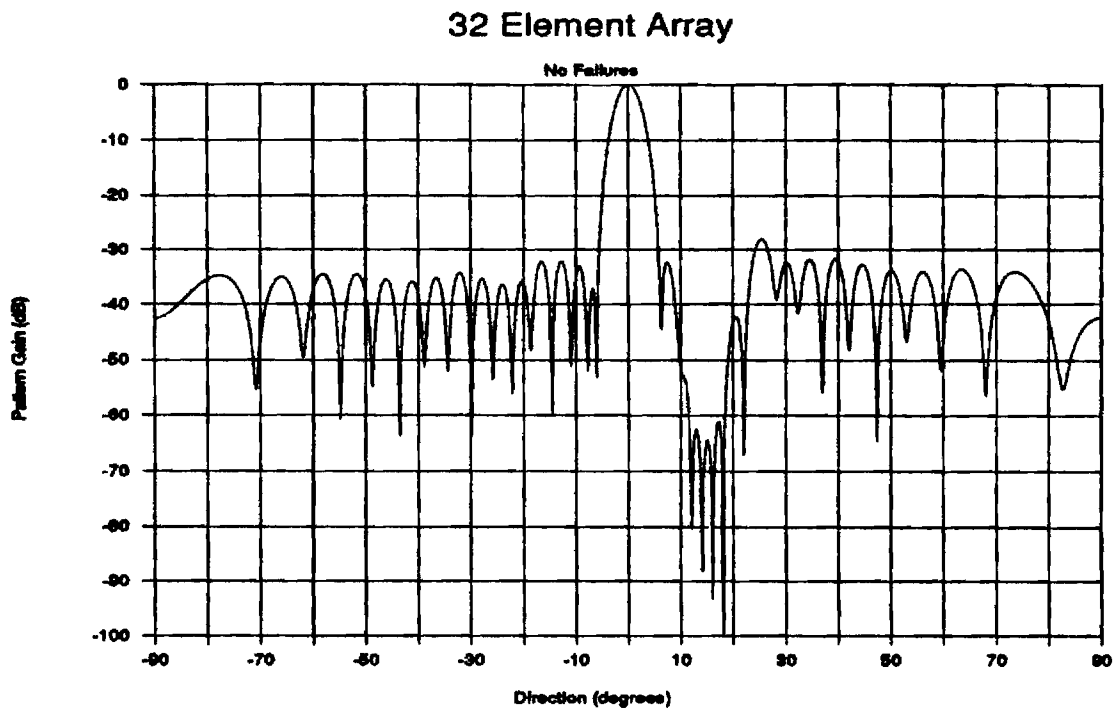

Figure 3.

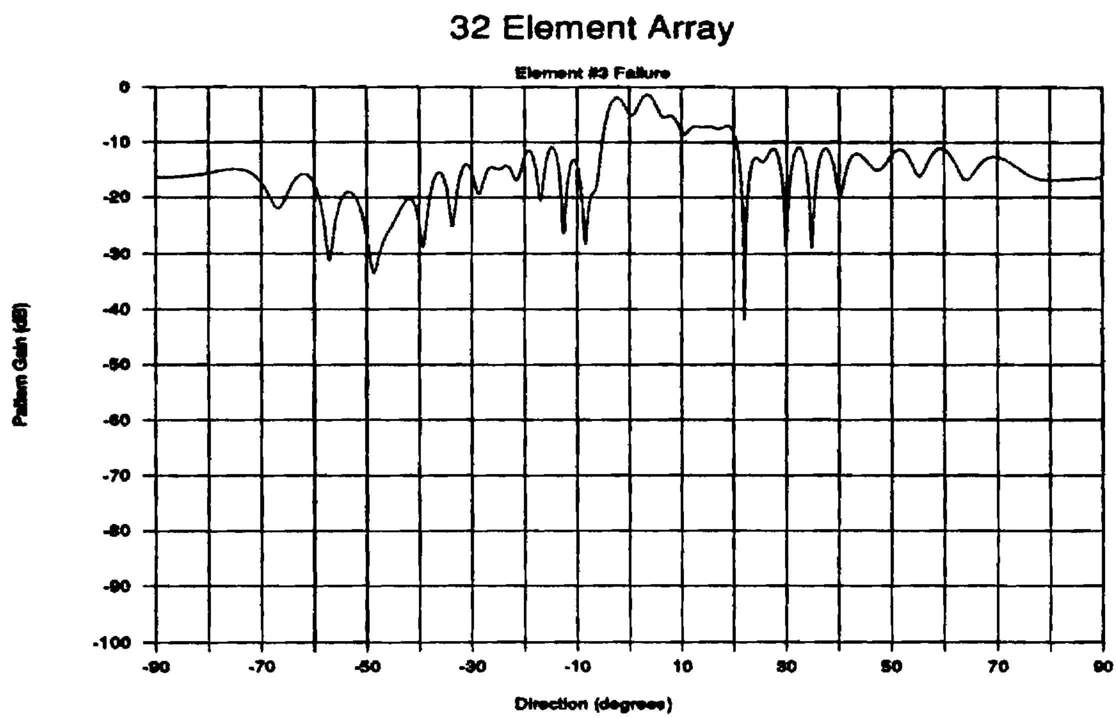

Figure 4.

32 Element Array

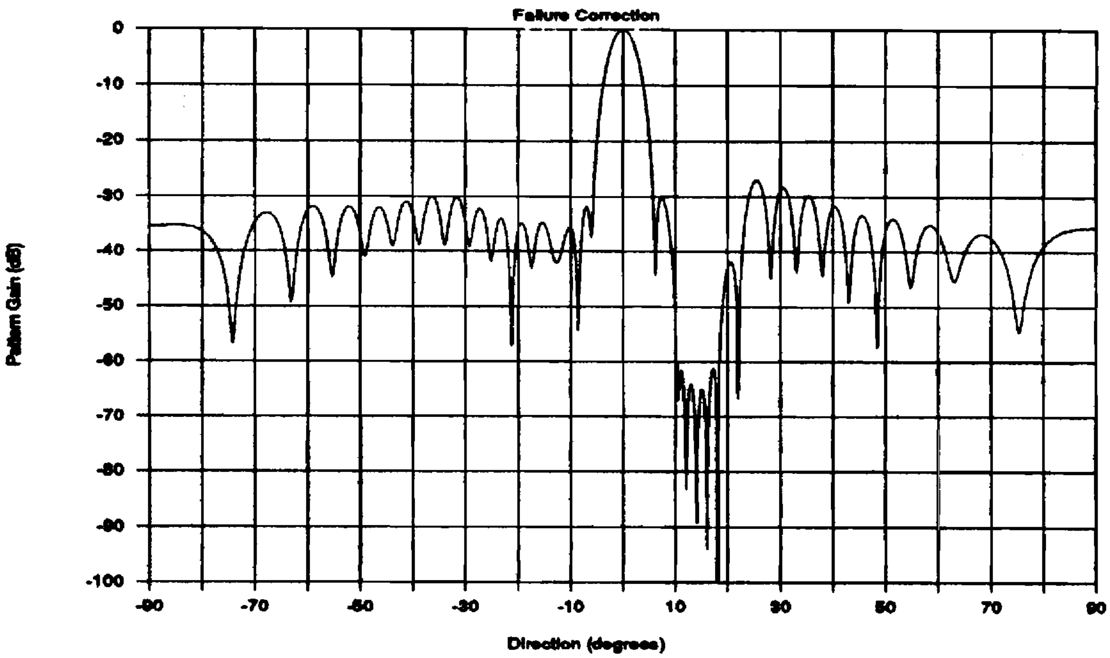

Figure 5.

32 Element Array

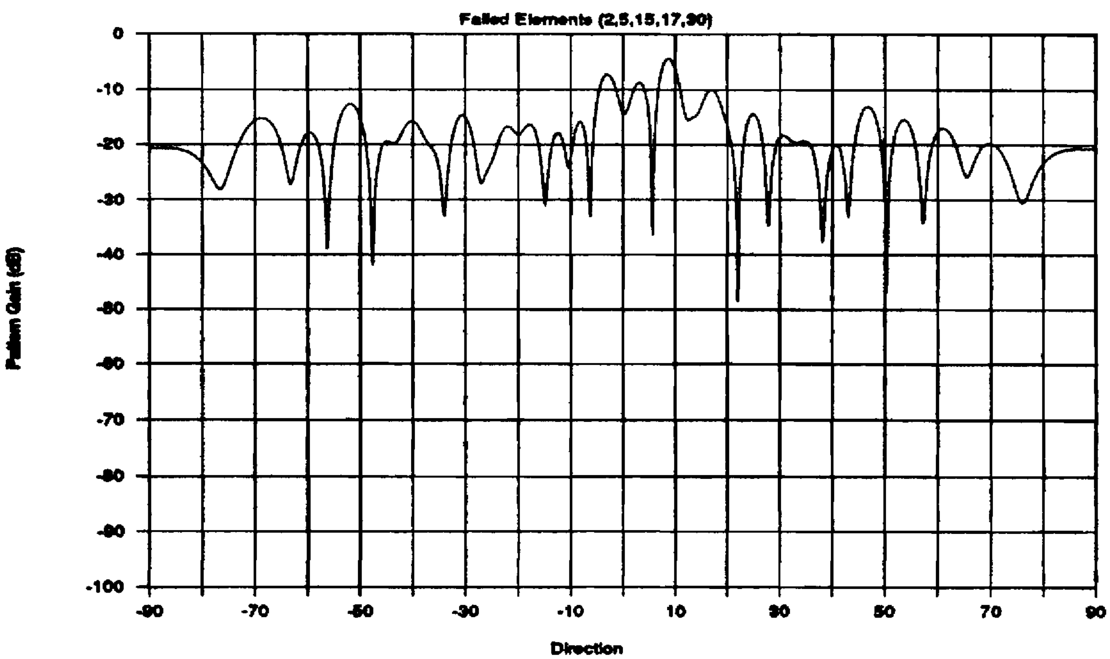

Figure 6.

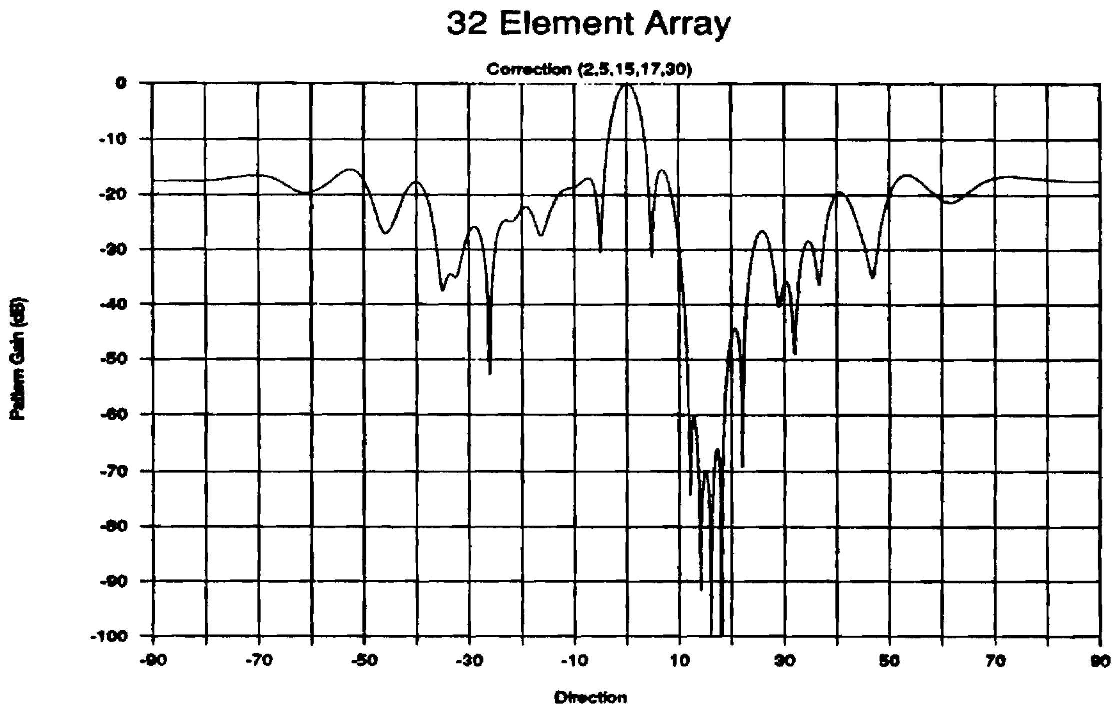

Figure 7.

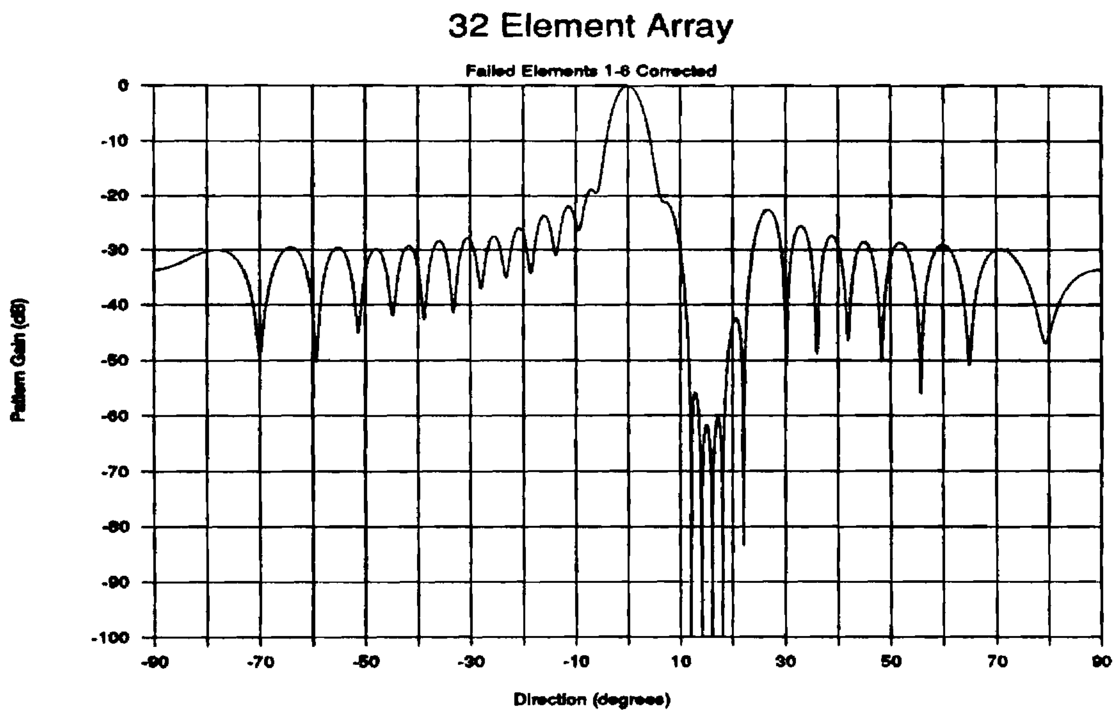

Figure 8.

26 Element Array

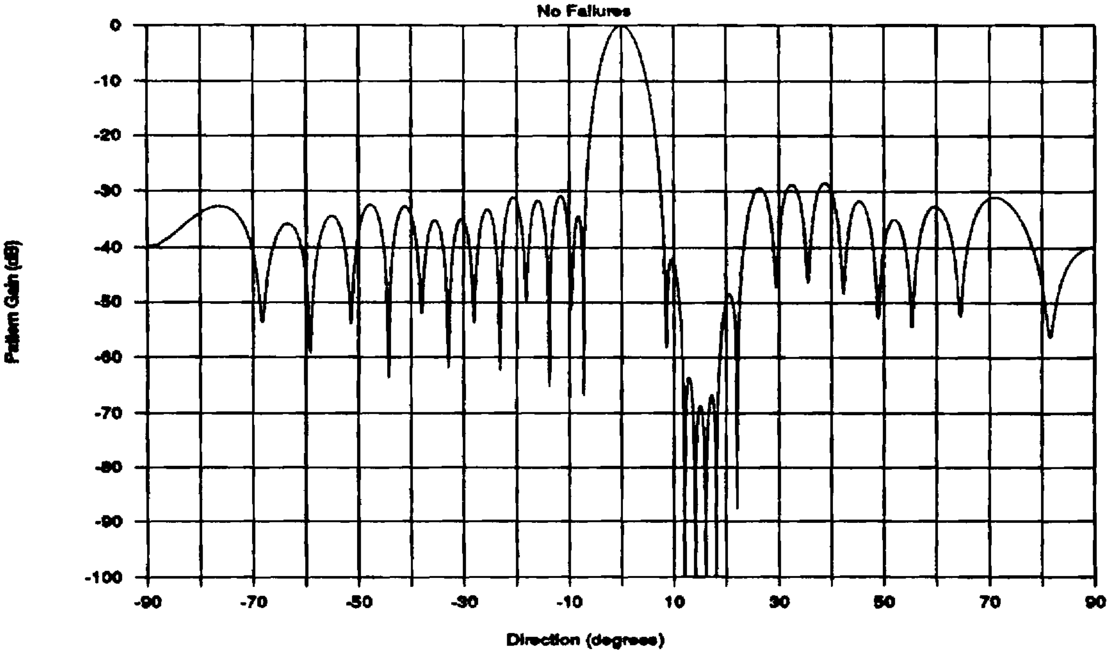

Figure 9.

64 Element Array

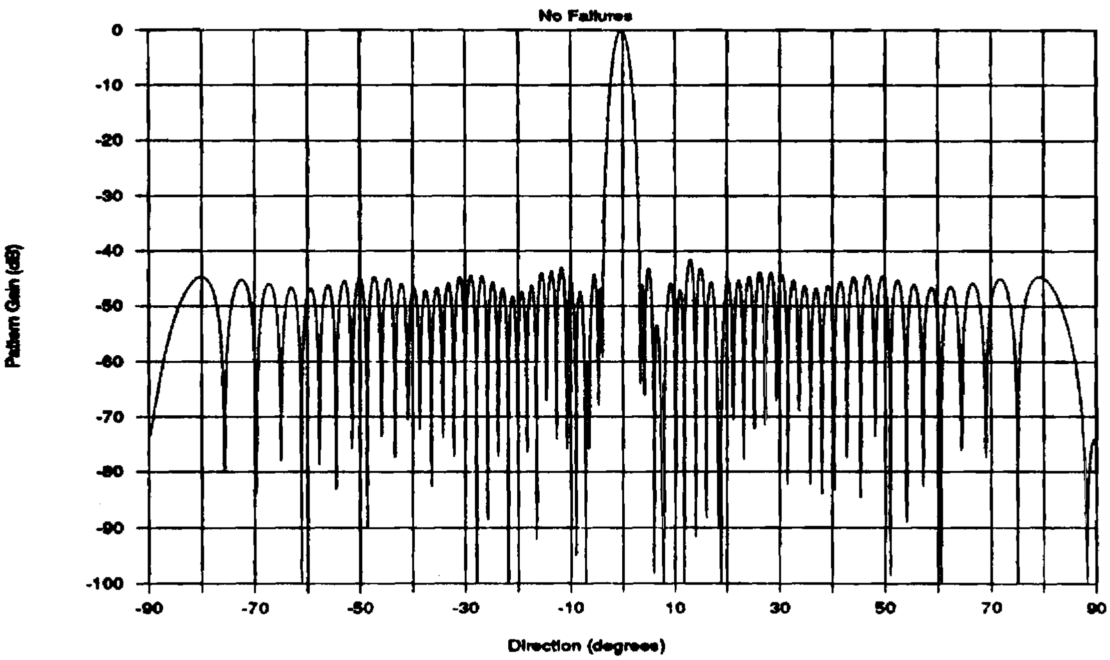

Figure 10.

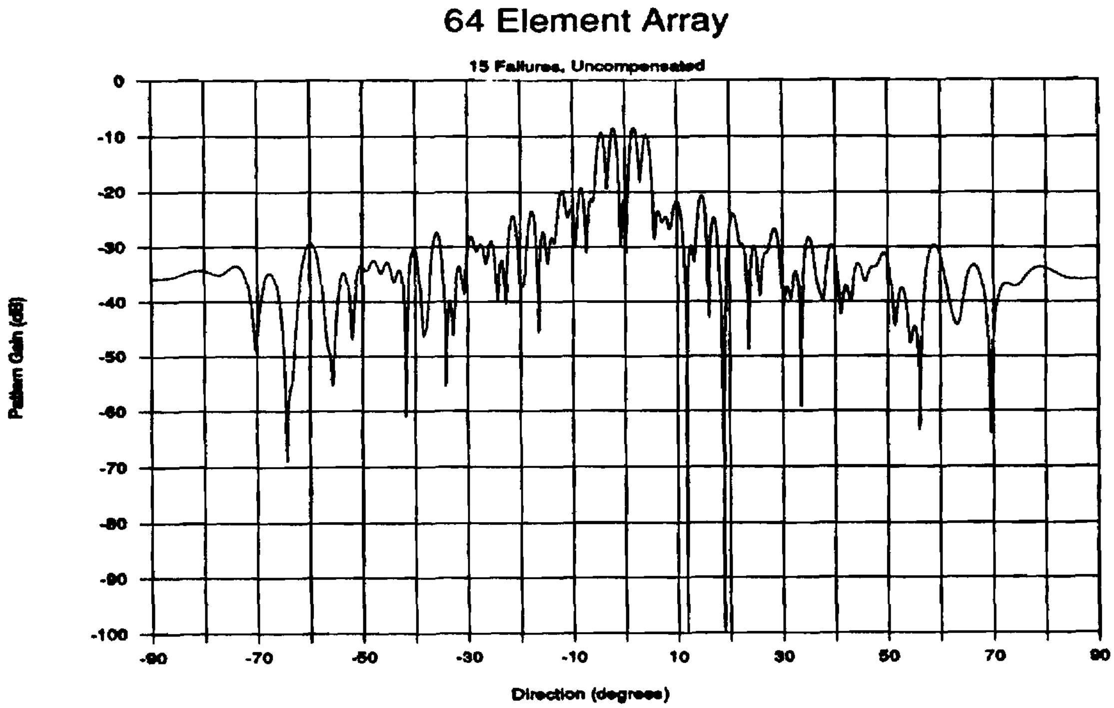

Figure 11.

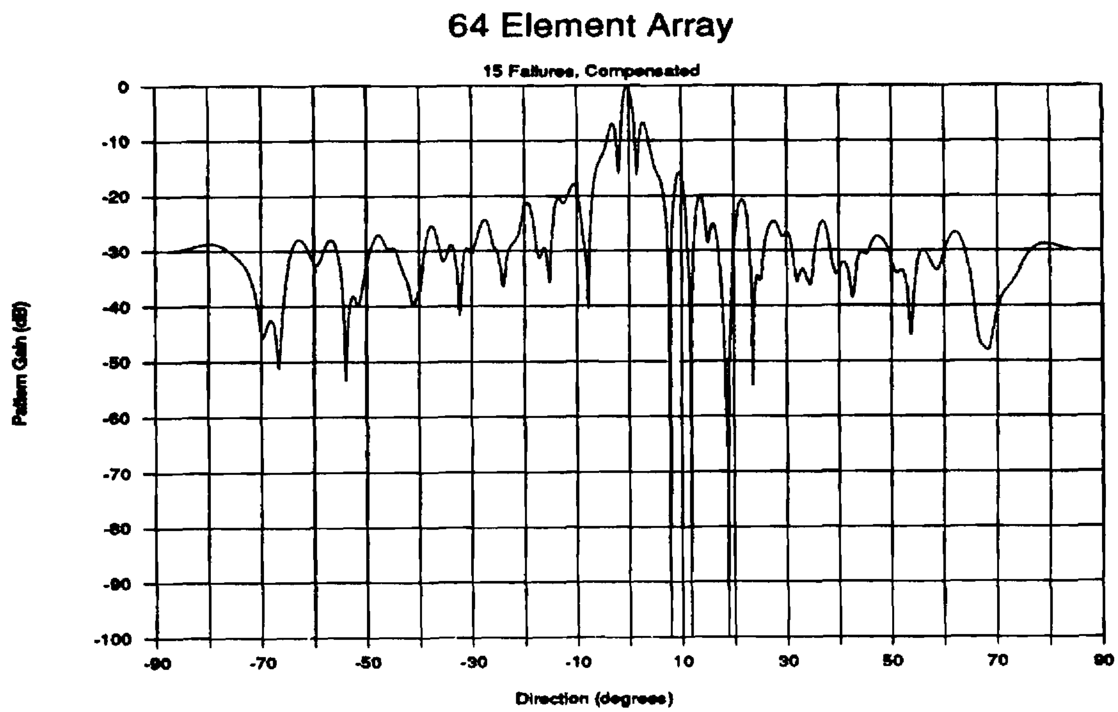

Figure 12.

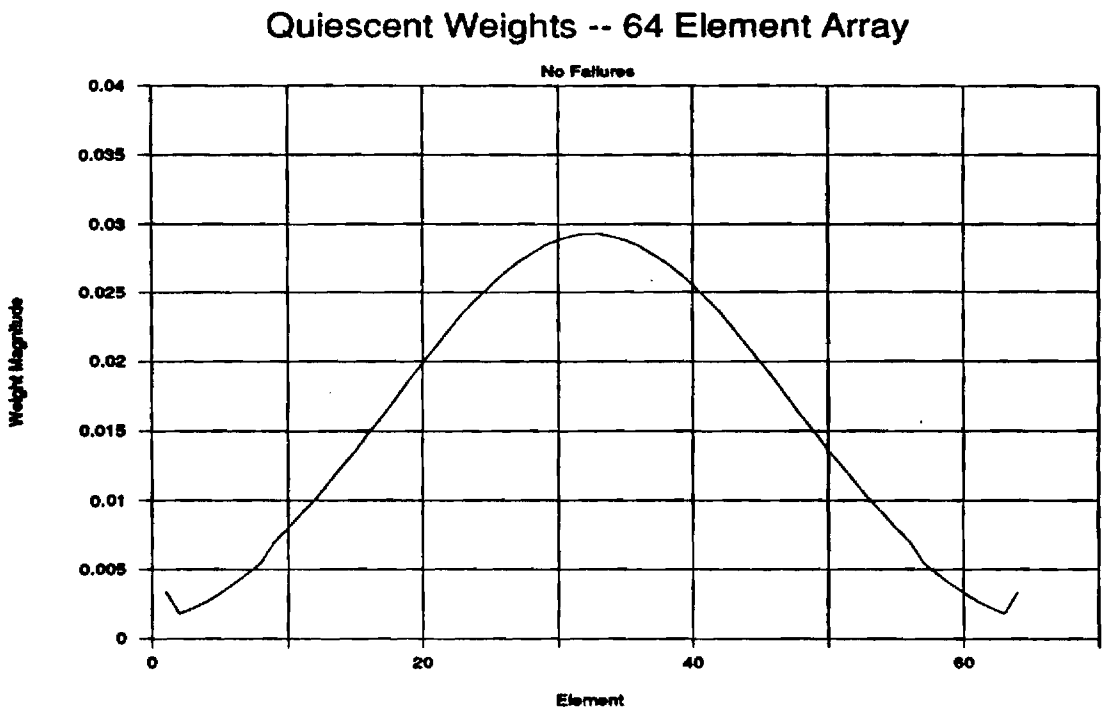

Figure 13.

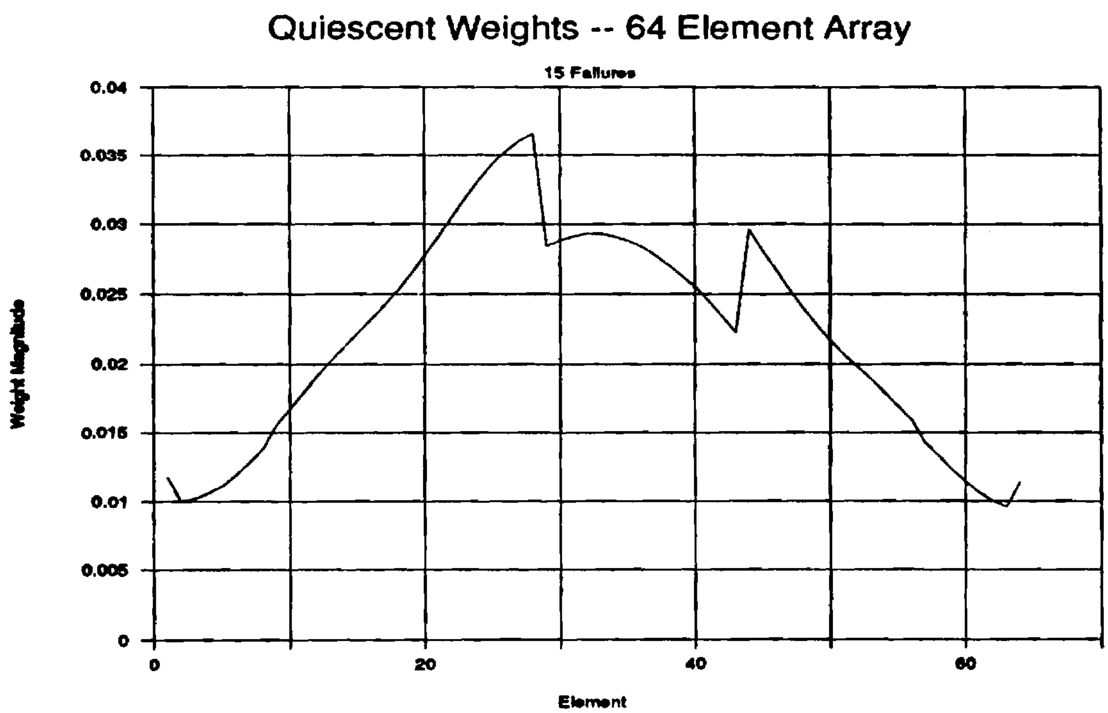

Figure 14.

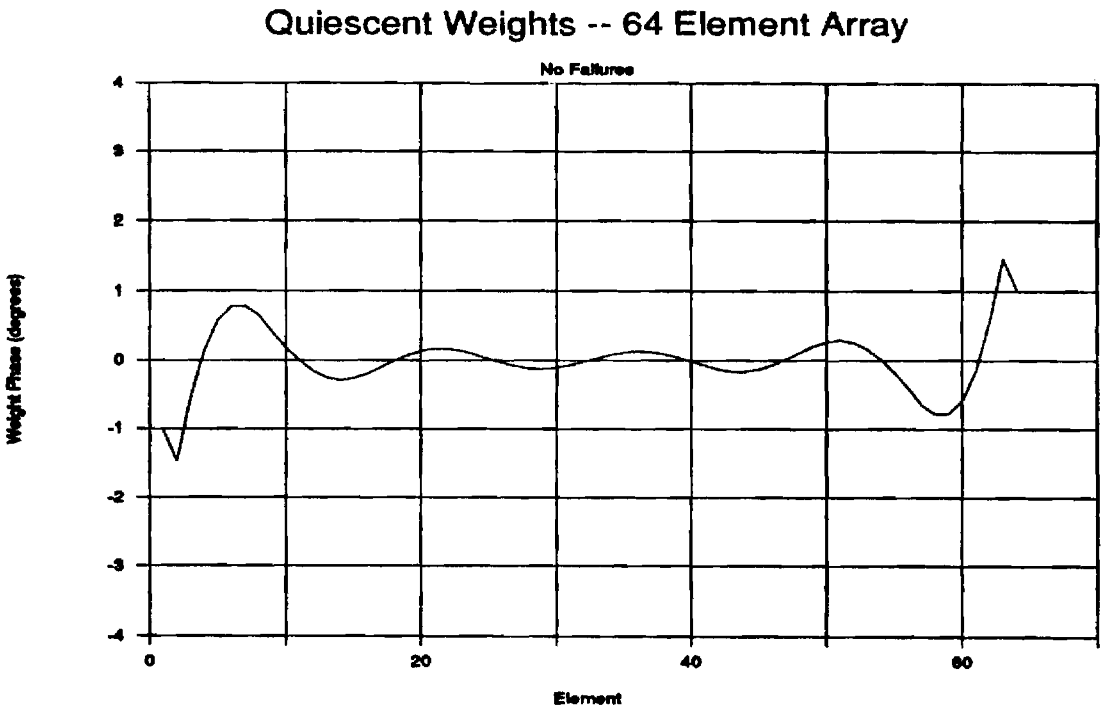

Figure 15.

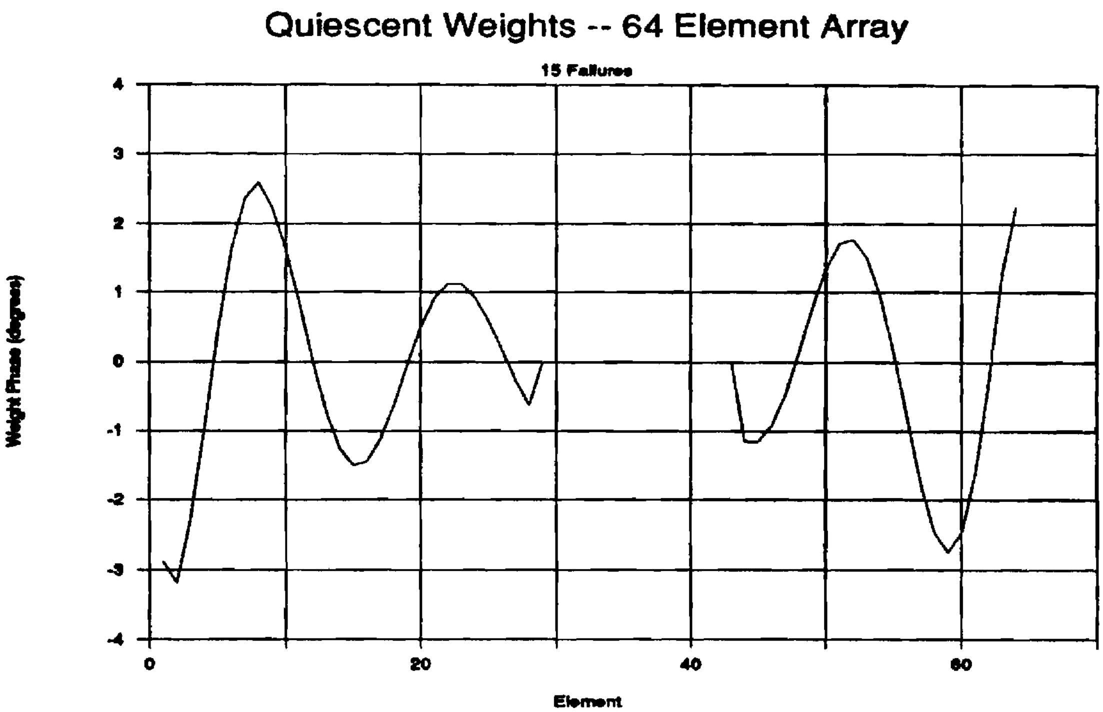

Figure 16.

References

[1] Monzingo, R. N., Miller, T. W. [1980], *Introduction to Adaptive Arrays,* Wiley and Sons, New York, 1980.

[2] Van Veen, B.D., Buckley, K. M., "Beamforming: A Versatile Approach to Spatial Filtering," *IEEE ASSP Magazine*,pp. 4-24, Apr. 1988.

[3] Steinhardt, A. O., Van Veen, B. D., "Adaptive Beamforming," *Inter. J. of Adaptive Control and Signal Proc.,* Vol 3., pp. 253-281, Sep 1989.

[4] Widrow, B., Mantey, P. E., Griffiths, L. J., Goode, B. B., "Adaptive Antenna Systems," *Proc. of the IEEE,* pp. 2143-2159, Dec. 1967.

[5] Griffiths, L. H., Charles W. J., "An Alternative Approach to Linearly Constrained Adaptive Beamforming," *IEEE Trans. on Ant and Prop.,* pp. 27-34, Jan. 1982.

[6] Buckley, K. M., Griffiths, L. J., "An Adaptive Generalized Sidelobe Canceller with Derivative Constraints," *IEEE Trans. on Ant and Prop.,* pp. 311-319, Mar 1986.

[7] Griffiths, L. H., Buckley, K. M., "Quiescent Pattern Control in Linearly Constrained Adaptive Arrays," *IEEE Trans. on ASSP,* pp. 917-926, Jul 1987.

[8] Hanson, R. J., Lawson, C. L., "Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems," *Math. Comp.,* Vol. 23, pp. 917-926, 1969.

[9] Drane, C. J. Jr., "Dolph-Chebyshev Excitation Coefficient Approximation," *IEEE Trans. on Ant and Prop.,* Vol. AP-12, No. 6, Nov 1964.

[10] Trudnowski, D. J., "Adaptive Array Beamforming Using the Generalized Sidelobe Canceller," *Elec Res Lab Report No. 188,* Montana State Univ. Feb. 1987.

APPENDIX

NOTATION

W_q Quiescent Weight Vector

W'_q Transformed quiescent weight vector

$\overline{W}_q$ Quiescent Weight Vector modified with Chebyshev constraint

W_b Blocking Matrix

W'_b Transformed Blocking Matrix

W'_{ba} Failed partition of transformed blocking matrix

W'_{bb} Remaining partition of transformed blocking matrix

W_a Adaptive Weight Vector

W'_a Transformed adaptive weight vector

W'_{qa} Failed partition of quiescent weight vector

W'_{qb} Remaining partition of quiescent weight vector

W_{cheb} Chebyshev weight vector

C Constraint Matrix

$\overline{C}$ Augmented Constraint Matrix (includes Chebyshev constraint)

C'_1 Failed partition of transformed constraint matrix

C'_2 Remaining partition of transformed constraint matrix

C' Transformed constraint matrix

L Number of gain constraints

M Number of total constraints

r Number of failed elements

T Failure Transformation Matrix

R_{xx} Element Correlation Matrix

R_{xx}' Element Correlation Matrix for Active Elements

APPENDIX

Fortran Code Program Listing

```
***********************************************************************
*
*   THIS PROGRAM FORMS AN ADAPTIVE ARRAY PATTERN USING THE GENERALIZED
*   SIDELOBE CANCELLER (GSC).  THE FOLLOWING ARRAY DATA IS INPUT:
*
*   NEL  NUMBER OF ELEMENTS,
*   DIS = ELEMENT SPACING IN TERMS OF WAVELENGTH,
*   RATIO = MAINLOBE-TO-SIDELOBE RATIO (USED TO FORM THE CHEBYSHEV WEIGHTS),
*   NANUL = NUMBER OF ANTICIPATORY NULLS,
*   ANUL(I) = ANTICIPATORY NULL LOCATIONS.
*
*
*   SUBROUTINE QUIESCENT.FOR IS USED TO FORM MATRUX WS AND
*   VECTOR WQ.  THE FOLLOWING ENVIROMENT INFORMATION IS INPUT:
*
*   SNR = SIGNAL-TO-NOIES RATIO,
*   NAJ = NUMBER OF INTERFERENCE SIGNALS,
*   THETA(I) = LOCATION OF INTERFERENCE SIGNALS,
*   JNR(I) = INTERFERENCE SIGNAL STRENGTH.
*
*   THE FOLLOWING IS OUTPUT IN DATA FILE PATTERN.DAT:
*        THE QUIESCENT GAIN VERSUS LOOK ANGLE (DEGREES).
*        THE ADAPTIVE GAIN VERSUS LOOK ANGLE (DEGREES).
*
*   THE FOLLOWING IS OUTPUT TO WEIGHTS:
*        "NDIM";
*        "MDIM";
*        "WQH" - Vector OF DIMENSION 1 X NDIM,
*                WHERE WQH IS THE HERMATIAN OF WQ;
*        "WSH" - Matrix WITH DIMENSION MDIM x NDIM,
*                WHERE WSH IS THE HERMATIAN OF WS; AND
*        "WAH" - Vector OF DIMENSION 1 X NDIM, W
*                HERE WAH IS THE HERMATIAN OF WA.
*
*        Modified to do failure simulations  --- dev  8-7-89
*
***********************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 JNR
        Real T1,delta
        COMPLEX*16 WQH, WSH, WS, R, P, TEM, WQ, WA, DET, RX, CSIMUL, T
        COMPLEX*16 RXT,RXA,RXB,RXC,RXD,TINV,WQT,WST,WQTA,WQTB
        COMPLEX*16 WQTBH,WSTB,WSTBH,WSTA
        INTEGER NEL, NANUL, MCON, fel(100),nfel
        character      fail
        DIMENSION ANUL(20),WSH(100,100),WS(100,100), WQH(100), R(100,100),
     1          P(100), WA(100), RX(100,100), TEM(100,100), WQ(100),
     1          APJ(10), THETA(10), JNR(10), T(100,100),TINV(100,100),
     1          RXT(100,100),RXA(100,100),RXB(100,100),RXC(100,100),
     1          RXD(100,100),WST(100,100),WQT(100),WQTB(100),WQTBH(100),
     1          WSTB(100,100),WSTBH(100,100),WQTA(100),WSTA(100,100),
     1          WQM(100),WQP(100),WAM(100),WAP(100)
        COMMON NEL,RATIO,DIS,NANUL,ANUL,MCON,nfel
        OPEN(17, FILE = 'WEIGHTS.DAT', STATUS = 'NEW')
        open(18, file = 'STATS.DAT',status='new')
        open(19, file = 'WA.DAT', STATUS = 'NEW')
        DATA PI / 3.141592654 /

C--INPUT THE ARRAY DATA, I.E., QUIECSENT INFORMATION:

        print*,'Is this failure simulation?'
        read(*,340) fail
        WRITE(*,210)
        READ*, NEL
        if(fail.eq.'y') then
           print*,'Number of Failed Elements'
```

```
                read*,nfel
            do 15, i=1,nfel
                print*,'Failed Element #'
                read*,fel(i)
 15            continue
        else
                fel(1)=-1
        endif
        WRITE(*,220)
        WRITE(*,230)
        READ*,DIS
        WRITE(*,235)
        WRITE(*,237)
        READ*,RATIO
        WRITE(*,240)
        READ*,NANUL
        IF (NANUL .EQ. 0) GOTO 10
        WRITE(*,250)
        WRITE(*,260)
        READ*, (ANUL(I), I=1,NANUL)
C  Start a Timer to determine caculation time
        T1=secnds(0.0)

C  DETERMINE THE TRANSFORMATION MATRIX T
        DO 5,I=1,nfel
          DO 4,J=1,NEL
             T(I,J)=(0.0,0.0)
             IF(J.EQ.fel(I)) T(I,J)=(1.0,0.0)
4         CONTINUE
5       CONTINUE
        DO 7,I=1,NEL-nfel
           J=I
           DO 6,K=1,nfel
            IF(I+JO.EQ.fel(K)) JO=JO+1
6         CONTINUE
          DO 8,K=1,NEL
                IF(K.EQ.J+JO)  THEN
                        T(I+nfel,K)=(1.0,0.0)
                ELSE
                        T(I+nfel,K)=(0.0,0.0)
                ENDIF
8       CONTINUE
7       CONTINUE

C***********    DETERMINE THE INVERSE TRANSFORM TINV  ***************
        DO 11, I=1,NEL
                DO 11, J=1,NEL
                TINV(I,J)=T(J,I)
11      CONTINUE

CXXXXXXXXXXX    T Matrix print statements XXXXXXXXXXXXXXXX
c       DO 9, I=1,NEL
c         WRITE(*,3) (T(I,J), J=1,NEL)
c9      CONTINUE
CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

C  SEND THE ARRAY INFORMATION TO THE QUIESCENT WEIGHT SUBROUTINE (CONTR.FOR).
C  CONTR.FOR RETURNS WITH THE VECTOR WQT AND THE MATRIX WST, THE TRANSPOSED
C  SYSTEM BLOCKING MATRIX AND QUIESCIENT WEIGHTS.

 10     CALL QUIESCENT(T, WST, WQT)
        DELTA=SECNDS(T1)
        print*,delta

C*************   SPLIT WST AND WQT INTO WST(A&B), AND WQT(A&B) *********
        DO 13, I=1,nfel
```

```
            WQTA(I)=WQT(I)
         DO 14, J=1,NEL-MCON-nfel
            WSTA(I,J)=WST(I,J)
14      CONTINUE
13      CONTINUE

        DO 17, I=nfel+1,NEL
            WQTB(I-nfel)=WQT(I)
        DO 16, J=1,NEL-MCON-nfel
            WSTB(I-nfel,J)=WST(I,J)
16          CONTINUE
17      CONTINUE

C  INPUT THE SIGNAL ENVIROMENT:

        WRITE(*,270)
        READ*,SNR
        WRITE(*,280)
        READ*, NAJ
        IF (NAJ .EQ. 0) GOTO 60
        WRITE(*,290)
        WRITE(*,300)
        READ*, (THETA(I), I=1,NAJ)
        WRITE(*,310)
        WRITE(*,320)
        READ*, (JNR(I), I=1,NAJ)
        VARDS = (10.0**(SNR/20.0))**2.0

C  CONVERT INTERFERENCE SIGNAL INFORMATION TO RADIANS AND RATIO AMPLITUDE:
        print*,'convert to radians and ratio'
        DO 20 I=1,NAJ
           THETA(I) = THETA(I)*PI*2.0/360.0
 20        APJ(I) = 10.0**(JNR(I)/20.0)

C  INPUT THE INTERFERENCE INTO THE COVARIANCE MATRIX RX:
        print*,'Input into covariance matrix rx'
        write(18,*)'Rii'
        DR = 2.0*PI*DIS
        DO 51 I=1,NEL
           DO 50 J=1,NEL
                IF (I .EQ. J) THEN
                    X = 0.0
                    DO 30 K=1,NAJ
 30                     X = X + APJ(K)**2.0
                    RX(I,J) = DCMPLX(X,0.0D0)
                ELSE
                   X = 0.0D0
                   Y = 0.0D0
                   DO 40 K=1,NAJ
                        PHI = DR*SIN(THETA(K))
                        X = X + APJ(K)**2.0*COS((J-I)*PHI)
 40                     Y = Y + APJ(K)**2.0*SIN((J-I)*PHI)
                   RX(I,J) = dCMPLX(X,Y)
                ENDIF
 50     CONTINUE

                write(18,330) (dreal(rx(i,j1)),dimag(rx(i,j1)),j1=1,nel)
51      continue
C  ADD THE DESIRED SIGNAL AND WHITE GAUSSIAN NOISE TO THE COVARIANCE MATRIX RX:

        print*,'add desired signal and noise'
        write(18,*)'Rxx=Rss+Rdd+Rii'
 60     DO 71 I=1,NEL
           DO 70 J=1,NEL
                IF (I .EQ. J) THEN
                   RX(I,J) = RX(I,J) + DCMPLX(1.0D0,0.0D0)
```

```
     1                         + DCMPLX(VARDS,0.0D0)
                ELSE
                   RX(I,J) = RX(I,J) + DCMPLX(VARDS,0.0D0)
                ENDIF

        do 61,if=1,nfel
           if(fail.eq.'y'.and.j.eq.fel(if)) then
                rx(i,j)=(0.,0.)
           endif

           if(fail.eq.'y'.and.i.eq.fel(if)) then
                rx(i,j)=(0.,0.)
           endif

           if(fail.eq.'y'.and.j.eq.i.and.i.eq.fel(if)) then
                rx(i,j)=(1.,0.)
           endif
 61     continue

 70     CONTINUE
c                write(18,330) (dreal(rx(i,j1)),dimag(rx(i,j1)),j1=1,nel)
 71     continue
c               write(18,*)'VARDS',vards

C*************    Calculate the transformed Covariance matrix RXT ********
C*************    RXT= T * RX * T' ****************************************
        CALL MATMUL(T,NEL,NEL,100,100,RX,NEL,100,100,TEM,100,100)
        CALL MATMUL(TEM,NEL,NEL,100,100,TINV,NEL,100,100,RXT,100,100)

C************    PARTION RXT INTO RXA,RXB,RXC,AND RXD MATRICES   **********

        DO 73, I=1,nfel
           DO 73, J=1,nfel
           RXA(I,J)=RXT(I,J)
73      CONTINUE
        DO 75, I=1,nfel
           DO 75, J=nfel+1,NEL
            RXB(I,J-nfel)=RXT(I,J)
75      CONTINUE
        DO 77, I=nfel+1,NEL
           DO 77, J=1,nfel
            RXC(I-nfel,J)=RXT(I,J)
77      CONTINUE
        DO 79, I=nfel+1,NEL
          DO 79, J=nfel+1,NEL
            RXD(I-nfel,J-nfel)=RXT(I,J)
79      CONTINUE


C  CALCULATE THE HERMATIAN OF THE BLOCKING MATRIX (WSTBH) AND THE
C  HERMATION OF THE QUIESCENT WEIGHT VECTOR (WQTBH):

        print*,'calculating the hermatian of ws and wq'
        DO 80 I=1,NEL-MCON-nfel
           DO 80 J=1,NEL-nfel
 80             WSTBH(I,J) = DCONJG(WSTB(J,I))

        DO 90 I=1,NEL-nfel
 90        WQTBH(I) = DCONJG(WQTB(I))

C  CALCULATE THE COVARIANCE MATRIX R AND THE VECTOR P:
        print*,'calculating covariance matrix r and vector p'

        CALL MATMUL(WSTBH,NEL-MCON-nfel,NEL-nfel,100,100,RXD,NEL-nfel,
     1              100,100,TEM,100,100)
        CALL MATMUL(TEM,NEL-MCON-nfel,NEL-nfel,100,100,WSTB,NEL-MCON-nfel,
```

```
     1               100,100,R,100,100)
        CALL MTVTMUL(TEM,NEL-MCON-nfel,NEL-nfel,100,100,WQTB,100,P,100)

C  SOLVE WA = R**-1 * P USING THE SUBROUTINE GAUSS:
        print*,'solve wa=r**-1*p'

        CALL GAUSS(R,P,WA,NEL-MCON-nfel,100)
        delta=secnds(t1)
        print*,delta

C**********  TRANSFORM SYSTEM WEIGHTS BACK INTO ORIGINAL SYSTEM *******
        CALL MTVTMUL(TINV,NEL,NEL,100,100,WQT,100,WQ,100)
        CALL MATMUL(TINV,NEL,NEL,100,100,WST,NEL-MCON-nfel,100,100,
     1          WS,100,100)

C******** CALCULATE THE HERMITIANS OF WS AND WQ  ***********************
        DO 110,I=1,NEL
              WQH(I)=DCONJG(WQ(I))
110     CONTINUE

        DO 115, I=1,NEL
           DO 115, J=1,NEL-MCON-nfel
              WSH(J,I)=DCONJG(WS(I,J))
115     CONTINUE

C  Output NEL, NEL-MCON, WQH, WSH, and WAH to 'WEIGHTS.DAT.'
        DO 117, I=1,NEL
          WQM(I)=DSQRT(DREAL(WQ(I))*DREAL(WQ(I))+DIMAG(WQ(I))*DIMAG(WQ(I)))
          WQP(I)=(180/3.1415)*DATAN2(DIMAG(WQ(I)),DREAL(WQ(I)))
c       print*,i,wqm(i),wqp(i)
117     CONTINUE
        DO 119, I=1,NEL-MCON-NFEL
          WAM(I)=DSQRT(DREAL(WA(I))*DREAL(WA(I))+DIMAG(WA(I))*DIMAG(WA(I)))
          WAP(I)=(180/3.1415)*DATAN2(DIMAG(WA(I)),DREAL(WA(I)))
119     CONTINUE
        J = NEL-MCON-nfel
        WRITE(17,*) 'NEL=',NEL, 'j',J
        WRITE(19,*) 'NEL =',NEL
        write(17,*) ' WQ Vector'
        WRITE(19,*) 'WA VECTOR'
        write(17,*) ' 2'
        WRITE(19,*) ' 2'
        DO 118,I=1,NEL
          WRITE(17,350) I,WQM(I),WQP(I)
118     CONTINUE
        DO 120,I=1,NEL-MCON
          WRITE(19,350) I,WAM(I),WAP(I)
120     CONTINUE

C       write(17,*) 'WS MATRIX'
C       DO 140 I=1,NEL
C 140      WRITE(17,*) (WS(I,J), J=1,NEL-MCON-nfel)
C       write(17,*) 'WA Vector'
        DO 150 I=1,NEL-MCON-nfel
  150      WA(I) = DCONJG(WA(I))
C       WRITE(17,*) (WA(I), I=1,NEL-MCON-nfel)
        CALL PATTERN(NEL,NEL-MCON,DIS,WQH,WSH,WA,fel,nfel)

C  FORMAT STATMENTS:

 210    FORMAT(/,' INPUT THE NUMBER OF ELEMENTS IN THE ARRAY: ',$)
 220    FORMAT(/,' INPUT THE DISTANCE BETWEEN THE ELEMENTS IN THE ARRAY')
 230    FORMAT(' IN TERMS OF THE WAVELENGTH: ',$)
 235    FORMAT(/' INPUT THE DESIRED MAINLOBE-TO-SIDELOBE RATIO FOR THE',$)
 237    FORMAT(' CHEBYSHEV WEIGHTS IN POSITIVE DB:',$)
 240    FORMAT(/,' INPUT THE NUMBER OF ANTICIPATORY NULLS DESIRED: ',$)
```

```
250     FORMAT(/,' INPUT THE LOCATION OF THESE NULLS IN DEGREES,')
260     FORMAT(' (WHERE THE BROADSIDE VIEW IS ZERO DEGREES): ',$)
270     FORMAT(/,' INPUT THE DESIRED SIGNAL TO NOISE RATIO IN DB: ',$)
280     FORMAT(/,' INPUT THE NUMBER OF INTERFERENCE SIGNALS: ',$)
290     FORMAT(/,' INPUT THE LOCATION OF THESE INTEFERENCE SIGNALS')
300     FORMAT(' IN DEGREES: ',$)
310     FORMAT(/,' INPUT THE STRENGTH OF THESE INTERFERENCE SIGNALS')
320     FORMAT(' WITH RESPECT TO THE NOISE, IN DECIBALS: ',$)
330     format(f8.1,f8.1,3x,f8.1,f8.1,3x,f8.1,f8.1,3x,f8.1,f8.1)
340     format(a1)
350     FORMAT(I4,3X,F10.4,3X,F10.4)
3       FORMAT(8(2(F3.0,X)X))

        WRITE(*,*)
        WRITE(*,*)'  THE ARRAY PATTERN IS IN THE DATA FILE PATTERN.DAT.'
        STOP
        END
```

```
*****************************************************************************
*
*  THIS PROGRAM CONTAINS A SUBROUTINE THAT CALCULATES THE QUIESCENT
*  WEIGHTS OF A GENERALIZED SIDELOBE CANCELOR (GSC).
*  THE FOLLOWING ARE INPUT TO THE SUBROUTINE:
*       NEL - NUMBER OF ELEMENTS IN THE ARRAY,
*       NANUL - THE NUMBER OF ANTICIPATORY NULLS,
*       DIS - THE DISTANCE BETWEEN THE ELEMENTS,
*       ANUL - THE LOCATION OF THE ANTICIPATORY NULLS.
*
*  RETURNS
*       WS  - The Blocking Matrix
*       WQ  - THE QUIESCENT WEIGHT VECTOR
*
*****************************************************************************

      SUBROUTINE QUIESCENT(T,WST,WQT)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 CON,CONH,CON2,WS,WQ,A,B,XY,WCHEB,FCON,CSIMUL
      COMPLEX*16 CT,C1,C2,C2X,T,WCHEBT,WQA,WQB,WSA,WSB,WST,C2H,WQT
      COMPLEX*16 WCHEBTB
      DIMENSION ANUL(20),WCHEB(100),CON(100,100),CONH(100,100),CON2(100,100),
     &  WS(100,100),FCON(100),WQ(100),A(100,100),B(100),T(100,100)

      DIMENSION CT(100,100),C2(100,100),C1(100,100),WCHEBT(100),
     &  C2X(100,100),WSA(100,100),WSB(100,100),WST(100,100)

      DIMENSION WQA(100),WQB(100),WCHEBTB(100),
     &  C2H(100,100),WQT(100)
      COMMON NEL,RATIO,DIS,NANUL,ANUL,MCON,nfel
      integer nfel,fel(20)
      DATA PI / 3.141592654 /
      OPEN(5,FILE='MATRIX.DAT',STATUS='NEW',RECL=5000)

C  CALL THE SUBROUTINE CHEBYSHEV.FOR TO OBTAIN THE CHEBYSHEV WEIGHTS.
C  THE MAINLOBE-SIDELOBE RATIO (RATIO), AND THE NUMBER OF ELEMENTS ARE
C  PASSED; THE CHEBYSHEV WEIGHTS (WCHEB)ARE RETURNED.

2     FORMAT(2X,F6.3,'+j',F6.3,2X)
      CALL CHEBYSHEV(NEL,DIS,RATIO,WCHEB)
c     WRITE(5,*) 'WCHEB'
c     DO 20, I=1,NEL
c        WRITE(5,2) WCHEB(I)
c20   CONTINUE


C  FORM THE CONSTRAINT MATRIX (CON) AND THE f VECTOR (FCON) USING
C  TWO STEPS:

      NCON = NANUL + 1
      DR = 2.0D0*PI*DIS

C***** STEP (1)  FORM THE LOOK DIRECTION (0 DB AT 0 DEGREES) CONSTRAINT;

      DO 400 I = 1,NEL
           CON(I,1) = (1.0D0,0.0D0)
 400  CONTINUE
      FCON(1) = (1.0D0,0.0D0)

C***** STEP (2)  FORM THE ANTICIPATORY NULL CONSTRAINTS;

      IF (NANUL .EQ. 0) GOTO 605
      print*,'Anticipatory null constraint matrix'
      DO 600 I = 1,NANUL
           K = I + 1
           THETA = ANUL(I)*2.0D0*PI/360.0D0
```

```
                  DO 500 J = 1,NEL
                          XJ = J
                          X = (1.0D0-XJ)*DR*DSIN(THETA)
                          CON(J,K) = DCMPLX(DCOS(X),DSIN(X))
  500             CONTINUE
                  FCON(K) = (0.0D0,0.0D0)
 600    CONTINUE

CXXXXXXXXX  PRINTOUT FOR C  XXXXXXXXXXXXXXXXXXXXXXXXXXX
c       WRITE(5,*) 'C MATRIX'
c       DO 602,I=1,NEL
c         WRITE(5,1),(CON(I,J), J=1,NCON)
c602    continue
1       FORMAT(16(2X,F6.3,'+',F6.3,'*i',2X))
CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C***** STEP (3)  COMPUTE THE TRANSFORMED MATRICES  CT

        CALL MATMUL(T,NEL,NEL,100,100,CON,NCON,100,100,CT,100,100)

CXXXXXXXXX  PRINTOUT FOR CT  XXXXXXXXXXXXXXXXXXXXXXXXXX
        WRITE(5,*) 'CT'
c       DO 601,I=1,NEL
c         WRITE(*,1) (CT(I,J), J=1,NCON)
c601    continue
CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

C*****  STEP (4)  DETERMINE THE PARTIONED MATRICES C1 AND C2 ****
        WRITE(5,*) 'PARTITIONED C1 MATRIX'
        DO 610, I=1,nfel
                DO 611, J=1,NCON
                   C1(I,J)=CT(I,J)
611             CONTINUE
c       write(5,1)  (C1(I,J),J=1,NCON)
610     CONTINUE

        WRITE(5,*) 'Partioned C2 matrix'
        DO 614, I=nfel+1,NEL
                DO 615, J=1,NCON
                  C2(I-nfel,J)=CT(I,J)
615             CONTINUE
c         write(5,1)  (C2(I-nfel,J),J=1,NCON)
614     CONTINUE

C******  STEP (5)  DETERMINE THE TRANSFORMED CHEBYSHEV-- WCHEBT ***
        CALL MTVTMUL(T,NEL,NEL,100,100,WCHEB,100,WCHEBT,100)

C******  WQA=WCHEBT  FOR THE FAILED ELEMENTS  **********
        DO 620,I=1,nfel
           WQA(I)=WCHEBT(I)
620     CONTINUE

C****  FORM WCHEBTB = WCHEBT FOR THE UNFAILED ELEMENTS ******
c       WRITE(5,*) 'WCHEB PARTION B'
        DO 630,I=nfel+1,NEL
           WCHEBTB(I-nfel)=WCHEBT(I)
c          WRITE(5,1) WCHEBTB(I-nfel)
630     CONTINUE

C*****  WQB=(I-C2*(C2H*C2)**-1*C2H*WCHEBTB + WQ   ***********
C  CALCULATE THE WEIGHTS WQ = B + C2* (C2H*C2)**-1 * FCON.
C  B = (I - C2 * (C2H * C2)**-1 * C2H) * WCHEBTB  (WHERE C2H IS THE
C  HERMATION OF C2):

C***** STEP (1)  CALCULATE C2X = C2 * (C2H * C2)**-1;

 605    DO 640,I = 1,NCON
```

```
         DO 640, J = 1,NEL-nfel
              C2H(I,J) = DCONJG(C2(J,I))
 640     CONTINUE
         WRITE(5,*) 'C2 HERMITIAN  ==> C2H'
c        DO 641, I=1,NCON
c          WRITE(5,1)  (C2H(I,J),J=1,NEL-nfel)
c641     CONTINUE

         CALL MATMUL(C2H,NCON,NEL-nfel,100,100,C2,NCON,100,100,A,100,100)
         WRITE(5,*) 'C2H*C2'
c        DO 643, I=1,NCON
c          WRITE(5,1)  (A(I,J),J=1,NCON)
c643     CONTINUE


         IF (NCON .EQ. 1) THEN
            A(1,1) = 1.0D0/A(1,1)
         ELSE
            XY = CSIMUL(NCON, A, B, 1E-14, -1, 100)
            IF (XY .EQ. (0.0D0,0.0D0)) THEN
                 PRINT*,'        AN ERROR HAS RESULTED WHEN CALCULATING'
                 PRINT*,'        THE (C2H * C2)**-1 MATRIX.'
                 STOP

            ENDIF
         ENDIF

         WRITE(5,*) '(C2H*C2)**-1  ==> A'
c        DO 642, I=1,NCON
c          WRITE(5,1)  (A(I,J),J=1,NCON)
c642     CONTINUE

         CALL MATMUL(C2,NEL-nfel,NCON,100,100,A,NCON,100,100,C2X,100,100)

         WRITE(5,*) 'C2X=C2*C2H*C2)**-1'
c        DO 645, I=1,NEL-nfel
c           WRITE(5,1)  (C2X(I,J), J=1,NCON)
c645     CONTINUE

C***** STEP (2)  CALCULATE WQB = C2 * (C2H * C2)**-1 * FCON, AND
C                WQB = (I - C2 * (C2H*C2)**-1 *C2H) * WCHEBTB + WQB;

         CALL MTVTMUL(C2X,NEL-nfel,NCON,100,100,FCON,100,WQB,100)

         WRITE(5,*) 'WQB=C2*C2H*C2)**-1 * FCON'
c        DO 646, I=1,NEL-nfel
c           WRITE(5,2) WQB(I)
c646     CONTINUE

         CALL MATMUL(C2X,NEL-nfel,NCON,100,100,C2H,NEL-nfel,100,100,A,100,100)
         DO 650 I=1,NEL-nfel
            DO 650 J=1,NEL-nfel
                 IF (I .EQ. J) THEN
                    C2X(I,J) = (1.0D0,0.0D0) - A(I,J)
                 ELSE
                    C2X(I,J) = -A(I,J)
                 ENDIF

 650     CONTINUE

         WRITE(5,*) '(I-C2*C2H*C2)**-1*C2H'
c        DO 647, I=1,NEL-nfel
c           WRITE(5,1)  (C2X(I,J), J=1,NEL-nfel)
c647     CONTINUE
c        DO 648,I=1,NEL-nfel
c                WRITE(5,2) WCHEBTB(I)
```

```
c648    CONTINUE

        PRINT*,'MULTILPY C2X AND WCHEBTB'
        CALL MTVTMUL(C2X,NEL-nfel,NEL-nfel,100,100,WCHEBTB,100,B,100)
        WRITE(5,*) 'B(I)=(I-C2*INV(C2H*C2)*C2H)*WCHEBB'
        DO 660 I=1,NEL-nfel
c          WRITE(5,2) B(I)
660        WQB(I) = B(I) + WQB(I)

C**********  FORMULATE THE COMPLETE WQ VECTOR  ***************
        DO 670,I=1,nfel
                WQT(I)=WQA(I)
670     CONTINUE
        DO 680,I=nfel+1,NEL
                WQT(I)=WQB(I-nfel)
680     CONTINUE
        WRITE(5,*) 'WQ TRANSPOSE'
c       DO 685, I=1,NEL
c               WRITE(5,2) WQT(I)
c685    CONTINUE

cccccccccccccccccccccc   I am Here  ccccccccccccccccccccccccccc


C  FORM WS SO THAT ALL ITS COLUMNS ARE LINEARLY INDEPENDENT AND
C  IT SATISFIES C2H*WSB = 0:

C******STEP (1)  MODIFY CONSTRAINTS TO INCLUDE THE CHEBY PATTERN;

        MCON=NCON+1
        DO 700 I=1,NEL-nfel
700        C2H(MCON,I) = DCONJG(B(I))
        WRITE(5,*) 'C2H APPENDED'
c       DO 701, I=1,MCON
c         WRITE(5,1) (C2H(I,J), J=1,NEL-nfel)
c701    CONTINUE


C***** STEP (2)  SET UP WSB INTO THE DESIRED FORM (TO ASSURE
C                THE COLUMNS ARE LINEARLY INDEPENDENT);

        DO 705 I = 1,NEL-nfel
           DO 705 J = 1,NEL-MCON-nfel
              WSB(I,J) = (0.0,0.0)
 705    CONTINUE
        DO 720 J = 1,NEL-MCON-nfel
           DO 710 I = J+1,J+MCON
                 WSB(I,J) = (3.0D0,0.0D0)
 710       CONTINUE
                 WSB(J,J) = (1.0D0,0.0D0)
 720    CONTINUE
        do 801,i=1,NEL-nfel
         do 801,j=1,NEL-MCON-nfel
c          print*,i,j,WSB(i,j)
801     continue
C***** STEP (3)  DETERMINE THE REMAINING WS TERMS FROM CH*WS = 0;

        DO 900 J = 1,NEL-MCON-nfel
c         PRINT*,'ROW',J
           K = 1
           DO 880 I = 1,NEL-nfel
                IF (WSB(I,J) .EQ. (1.0D0,0.0D0)) THEN
                   DO 850 L = 1,MCON
                        A(L,MCON+1) = -C2H(L,I)
 850               CONTINUE
                ELSEIF (WSB(I,J) .EQ. (3.0D0,0.0D0)) THEN
```

```
                    DO 875 L=1,MCON
                         A(L,K) = C2H(L,I)
 875                CONTINUE
c       do 799, ia=1,MCON
c         print*,ia,k,a(ia,k)
c 799   continue

                    K = K + 1
                 ENDIF
 880       CONTINUE
           XY = CSIMUL(MCON, A, B, 1.E-18, 1, 100)
c       print*,'xy',xy
           IF (XY .EQ. (0.0D0,0.0D0)) THEN
                 PRINT*,'AN ERROR HAS RESULTED IN THE WS MATRIX,'
                 print*,'Column ',j
                 PRINT*,'CANNOT CONTINUE!'
                 STOP
           ENDIF
           L = 1
           DO 890 I = 1,NEL-nfel
                 IF (WSB(I,J) .EQ. (3.0D0,0.0D0)) THEN
                    WSB(I,J) = B(L)
                    L = L+1
                 ENDIF
 890       CONTINUE
 900    CONTINUE

C  SEND WS, AND WQ BACK TO MAIN4.FOR

CCCCC   Now formulate the complete transformed WB matrix CCCCCCCC
        DO 910, I=1,nfel
          DO 910, J=1,NEL-MCON-nfel
            WSA(I,J)=0.0D0
            WST(I,J)=WSA(I,J)
910     CONTINUE
        DO 920, I=1,NEL-nfel
          DO 920, J=1,NEL-MCON-nfel
           WST(I+nfel,J)=WSB(I,J)
920     CONTINUE

        RETURN
        END
```

```
***********************************************************************
*
*  THIS PROGRAM CONTAINS A SUBROUTINE TO COMPUTE THE DOLPH-CHEBYSHEV
*  WEIGHTS FOR AN ARRAY WITH EQUALLY SPACED ELEMENTS.  THE NUMBER OF
*  WEIGHTS MUST BE ODD.  THE SIGNAL IS AT THE BROADSIDE VIEW.  THE
*  ALGORITHM USED IS FROM C.J. DRANE, JR., "DOLPH-CHEBYSHEV EXCITATION
*  COEFFICIENT APPROXIMATION," IEEE TRANS. ON ANT. AND PROP,
*  VOL. AP-12, NUM. 6, NOV. 1964.
*
***********************************************************************

        SUBROUTINE CHEBYSHEV(NEL,DIS,R,W)
        implicit real*8 (a-h,o-z)
        real*8 delta
        COMPLEX*16 W
        DIMENSION W(100)
        DATA PI / 3.14159265 /
        R = 10.0**(R/20.0)

C---DETERMINE IF THE NUMBER OF ELEMENTS IS ODD OR EVEN:
        print*,'subroutine chebyshev'
        EVOD = NEL/2 - FLOAT(NEL)/2.0
        IF (evod .EQ. 0.0) THEN
           N = NEL/2
        ELSE
           N = (NEL - 1)/2
        ENDIF

C---SOLVE FOR THE DOLPH-CHEBYSHEV WEIGHTS:

        DELTA = PI*(1.-2.0d0*DIS)
        Z = DCOSH(DLOG(R + DSQRT(R*R - 1.0D0))/N)
        ALPHA = DACOS((3.0D0 - Z)/(1.0D0 + Z))
        W(1) = ((Z+1.0D0)/(1.0D0+DCOS(DELTA)))**N
        DO 20 I=2,N
           XX = DSQRT(DFLOAT(N*N - (N + 1 - I)**2))
         W(I) = N*ALPHA*BESSELf(XX*ALPHA)/XX
     &           + N*(-1)**(2*N-I+1)*
     &           DELTA*BESSELf(XX*DELTA)/XX
20      continue
        IF (EVOD .NE. 0.0) THEN
C          CALL BESSEL(N*ALPHA,X1)
C          CALL BESSEL(N*DELTA,Y1)
           W(N+1) = ALPHA*BESSELf(N*ALPHA) + DELTA*BESSELf(N*DELTA)
        ENDIF
           DO 30 I=1,N
 30            W(NEL-I+1) = DCONJG(W(I))

C---SCALE THE WEIGHTS SO THEY ADD UP TO ONE:

        SUM = 0.0
        DO 40 I=1,NEL
 40        SUM = SUM + CDABS(W(I))
        DO 50 I=1,NEL
           W(I) = W(I)/SUM
c              print*,w(i)
50      Continue

        print*,'Exit chebyshev'
        RETURN
        END

C---THE FOLLOWING FUNCTION (BESSEL) FINDS THE MODIFIED BESSEL FUNCTION
C   OF THE FIRST KIND AND FIRST ORDER:

       Real*8 FUNCTION BESSELf(X)
```

```
      real*8 x
      T = X/3.75
      IF (X .GT. -3.75 .AND. X .LT. 3.75) THEN
         XI = 0.5 + 0.87890594*T**2. + 0.51498869*T**4.+
&             0.15084934*T**6. + 0.02658733*T**8. +
&             0.00301532*T**10.0 + 0.00032411*T**12.
         XI = XI*X
      ELSE IF (X .GE. 3.75) THEN
         XI = 0.39894228 - 0.03988024*T**(-1.)-
&             0.00362018*T**(-2.) + 0.00163801*T**(-3.)-
&             0.01031555*T**(-4.) + 0.02282967*T**(-5.)-
&             0.02895312*T**(-6.) + 0.01787654*T**(-7.)-
&             0.00420059*T*(-8.)
         XI = XI*X**(-0.5D0)*DEXP(X)
      ELSE
         PRINT*,'      BESSEL ERROR'
         STOP
      ENDIF
      BESSELf = XI
      RETURN
      END
```

```
************************************************************************
*
*  THIS SUBROUTINE USES THE GSC TO FORM THE ARRAY PATTERN.
*  THE OUTPUT DATA FILE (PATTERN.DAT) CONTAINS THE ARRAY PATTERN.
*  Note:  This file was modified from the original pattern.for program
*         and will be used to separate the individula outputs into their
*         own files named que.dat and adapt.dat.   (Without headers so
*         they are readily accesible by pplot.
*              8-7-89  dev
*  Note:  Another file was added 'fail.dat' to store the information from
*         the failed element.
*
************************************************************************
        SUBROUTINE PATTERN(KANT,NDIM,DIS,WQH,WSH,WAH,fel,nfel)
        IMPLICIT REAL*8 (A-H,O-Z)
        COMPLEX*16 WQH, WSH, WAH, E, OUT, U, Y, YQ, YA ,efail(100),yqf
        integer fel(100),nfel
        DIMENSION E(100), WQH(100), WSH(100,100), WAH(100), U(100)
        OPEN(10,FILE = 'QUE.DAT', STATUS = 'NEW')
        open(11,file = 'ADAPT.DAT',Status = 'NEW')
        open(14,file='output.dat',status='new')
        open(12,file= 'real.dat',status='new')
        open(13,file='imag.dat',status='new')
C       WRITE(10,*)'  THE FOLLOWING IS THE PATTERN FOR THE GSC.'
C       WRITE(10,*)'  LOOK ANGLE (DEGREES),   QUIESCENT GAIN,
C     &                  ADAPTIVE GAIN'
C       WRITE(10,*)2
        if(fel(1).gt.0) then
        do 30,i=1,nfel
           print*,'Failed Element!!!  #',fel(i)
30      continue
          else
           print*,'Elements OK'
        endif
        PI = 3.141592654
        E(1) = (1.0D0,0.0D0)
        DO 1000 I=1,1801
           XI = I-1.0D0
           THET=90.0D0-(XI*180.0D0/1800.0D0)
           THETA = THET*2.0D0*PI/360.0D0


C  CALCULATE INCOMING SIGNAL:

           DO 90 J=2,KANT
                XJ = J
                PHI = (1.0D0-XJ)*2.0D0*DIS*PI*DSIN(THETA)
                E(J) = DCMPLX(DCOS(PHI),DSIN(PHI))
 90        CONTINUE

          if(fel(1).gt.0) then
          do 35 if=1,nfel
                efail(if)=e(fel(if))
               e(fel(if))=(0.0D0,0.0D0)
35        continue
          endif


C  CALCULATE OUTPUT (Y = YQ - YA):

           YQ = (0.0D0,0.0D0)
           DO 95 J=1,KANT
                yqf=0.0D0
             do 94,if=1,nfel
                if(j.eq.fel(if)) then
                        yqf=yqf+wqh(j)*efail(if)
```

```
                endif
 94           continue
 95             YQ = YQ + WQH(J)*E(J)

            CALL MTVTMUL(WSH,NDIM,KANT,100,100,E,100,U,100)

            YA = (0.0D0,0.0D0)
            DO 100 J=1,NDIM
 100            YA = YA + WAH(J)*U(J)

            Y = YQ - YA
            yqfout= cdabs(yqf)
            write(12,110)thet,Dreal(yq),Dreal(yqf)
            write(13,110)thet,dimag(yq),dimag(yqf)
c           yqfout= 20.0D0*Dlog10(yqfout)
            YQOUT = CDABS(YQ)
            YQOUT = 20.0D0*DLOG10(YQOUT)
            YAOUT = CDABS(YA)
            YAOUT = 20.0D0*DLOG10(YAOUT)
            AOUT =  CDABS(Y)
            AOUT = 20.0D0*DLOG10(AOUT)

            WRITE(10,110)THET,YQOUT,YAOUT
            write(11,120)thet,YAOUT
            WRITE(14,120) THET,AOUT
 110      FORMAT(2X,F15.8,6X,F15.8,6X,F15.8)
 120      format(2x,f15.8,6x,f15.8)
 130      format(2x,f15.8,2x,f15.8,2x,f15.8,2x,f15.8,2x,f15.8,2x,f15.8)
 1000     CONTINUE
          RETURN
          END
```

```
*************************************************************************
*
*  THIS PROGRAM CONTAINS THE FUNCTION CSIMUL, A GAUSS-ELIMINATION SUBROUTINE,
*  AND THREE MATRIX MULTIPLICATION SUBROUTINES.
*
**************************************************************************
C
*************************************************************************
*
      FUNCTION CSIMUL(N,CMAT,CX,EPS,INDIC,NRC)
C     CSIMUL IS A COMPLEX (DOUBLE PRECISION) MATRIX INVERSION ROUTINE
C
C     USE:
C     WHEN INDIC IS NEGATIVE, CSIMUL COMPUTES THE INVERSE OF THE N BY N
C     COMPLEX MATRIX CMAT IN PLACE. WHEN INDIC IS ZERO,CSIMUL COMPUTES
C     THE N SOLUTIONS CX(1)...CX(N) CORRESPONDING TO THE SET OF LINEAR
C     EQUATIONS WITH AUGMENTED MATRIX OF COEFFICIENTS IN THE N BY N+1
C     ARRAY CMAT AND IN ADDITION COMPUTES THE INVERSE OF THE COEF-
C     FICIENT MATRIX IN PLACE AS ABOVE. IF INDIC IS POSITIVE,
C     THE SET OF LINEAR EQUATIONS IS SOLVED BUT THE INVERSE IS NOT
C     COMPUTED IN PLACE. THE GAUSS-JORDAN COMPLETE ELIMINATION METHOD
C     IS EMPLOYED WITH THE MAXIMUM PIVOT STRATEGY. THE RESULTING VALUE
C     IN LOCATION CSIMUL IS THE COMPLEX DETERMINANT OF THE SYSTEM.
C
      IMPLICIT COMPLEX*16 (A-H,O-Z)
        REAL*8 EPS
      COMPLEX*16 CMAT,CX,Y,DETER,PIVOT,CIJCK,CSIMUL
      DIMENSION IR(100),JC(100),JORD(100),Y(100),CMAT(NRC,NRC+1)
      DIMENSION CX(NRC)
      MAX=N

      IF(INDIC.GE.0)MAX=N+1
      IF(N.GT.NRC-1)THEN
        WRITE(5,200)
        CSIMUL=DCMPLX(0.0d0,0.0d0)
        RETURN
      END IF
      DETER=DCMPLX(1.,0.)
      DO 18 K=1,N
      KM=K-1
      PIVOT=DCMPLX(0.d0,0.d0)
      DO 11 I=1,N
      DO 11 J=1,N
      IF(K.EQ.1)GO TO 9
      DO 8 IS=1,KM
      DO 8 JS=1,KM
      IF(I.EQ.IR(IS))GO TO 11
    8 IF(J.EQ.JC(JS))GO TO 11
    9 IF(CDABS(CMAT(I,J)).LE.CDABS(PIVOT))GO TO 11
      PIVOT=CMAT(I,J)
      IR(K)=I
      JC(K)=J
   11 CONTINUE
      IF(CDABS(PIVOT).GT.EPS)GO TO 13
c       print*,'pivot',pivot
      CSIMUL=DCMPLX(0.d0,0.d0)
      RETURN
   13 DETER=DETER*PIVOT
      DO 14 J=1,MAX
   14 CMAT(IR(K),J)=CMAT(IR(K),J)/PIVOT
      CMAT(IR(K),JC(K))=1.0D0/PIVOT
      DO 18 I=1,N
      CIJCK=CMAT(I,JC(K))
      IF(I.EQ.IR(K))GO TO 18
      CMAT(I,JC(K))=-CIJCK/PIVOT
      DO 17 J=1,MAX
```

```
   17 IF(J.NE.JC(K))CMAT(I,J)=CMAT(I,J)-CIJCK*CMAT(IR(K),J)
   18 CONTINUE
      DO 20 I=1,N
      JORD(IR(I))=JC(I)
   20 IF(INDIC.GE.0)CX(JC(I))=CMAT(IR(I),MAX)
      INTCH=0
      NM=N-1
      DO 22 I=1,NM
      IP=I+1
      DO 22 J=IP,N
      IF(JORD(J).GE.JORD(I))GO TO 22
      JT=JORD(J)
      JORD(J)=JORD(I)
      JORD(I)=JT
      INTCH=INTCH+1
   22 CONTINUE
      IF((INTCH/2)*2.NE.INTCH)DETER=-DETER
   24 IF(INDIC.LE.0)GO TO 26
      CSIMUL=DETER
      RETURN
C IF INDIC IS NEGATIVE OR ZERO, UNSCRAMBLE THE INVERSE
   26 DO 28 J=1,N
      DO 27 I=1,N
   27 Y(JC(I))=CMAT(IR(I),J)
      DO 28 I=1,N
   28 CMAT(I,J)=Y(I)
      DO 30 I=1,N
      DO 29 J=1,N
   29 Y(IR(J))=CMAT(I,JC(J))
      DO 30 J=1,N
   30 CMAT(I,J)=Y(J)
      CSIMUL=DETER
      RETURN
  200 FORMAT(13H N IS TOO BIG)
      END

*************************************************************************
*
*  THIS SUBROUTINE USES GAUSS ELIMINATION WITH NO PIVOTING TO SOLVE
*  THE MATRIX EQUATION Ax = b, WHERE A IS AN N BY N COMPLEX MATRIX,
*  X IS AN N BY 1 UNKNOWN COMPLEX VECTOR, AND b IS AN N BY 1 KNOWN VECTOR.
*  IT IS A WELL KNOWN FACT THAT GAUSS ELIMNATION WITH NO PIVOTING IS
*   EQUIVALENT TO LU-DECOMPOSITION.

        SUBROUTINE GAUSS(A,B,X,N,NRC)
        COMPLEX*16 A,B,X,Q
        DIMENSION A(NRC,NRC), B(NRC), X(NRC)

C  AUGMENTE THE MATRIX A TO INCLUDE B:

        DO 10 I=1,N
 10        A(I,N+1) = B(I)

C  CHECK THE DIAGONAL OF A TO DETERMINE IF THERE IS AN 0,0 ELEMENT:

        DO 20 I=1,N
           IF (A(I,I) .EQ. (0.0D0,0.0D0)) THEN
           WRITE(*,*)'  AN ERROR HAS OCCURED IN THE GAUSS SUBROUTINE.'
           STOP
           ENDIF
 20     CONTINUE

C  ZERO ENTRIES (I+1,I), (I+2,I),...,(N,I) IN THE AUGMENTED MATRIX:

        DO 30 I=1,N-1
           DO 30 K=I+1,N
```

```
                  Q = -A(K,I)/A(I,I)
                  A(K,I) = (0.0D0,0.0D0)
                  DO 30 J=I+1,N+1
                     A(K,J) = Q*A(I,J) + A(K,J)
 30      CONTINUE

C  BACKSOLVE TO OBTAIN A SOLUTION TO AX = B:

         X(N) = A(N,N+1)/A(N,N)
         DO 40 K=1,N-1
            Q=0.0
            DO 35 J=1,K
  35              Q=Q+A(N-K,N+1-J)*X(N+1-J)
            X(N-K) = (A(N-K,N+1)-Q)/A(N-K,N-K)
 40      CONTINUE
         RETURN
         END

************************************************************************
*
C  THE FOLLOWING THREE SUBROUTINES ARE USED TO MULTIPLY:  TWO COMPLEX
C  MATRICES, A VECTOR AND A MATRIX, AND A MATRIX AND VECTOR.
C
C  IN THE FIRST SUBROUTINE, TWO MATRICES ARE MULTIPLIED.  THE FOLLOWING
C  IS INPUT:  MATRICES A AND B, THE DIMENSION OF MATRIX A (K x L),
C  THE DIMENSION OF MATRIX B (M x N), AND THE DIMENSION OF THE ARRAYS.
C  C = A * B IS CALCULATED.

         SUBROUTINE MATMUL(A,K,L,KM,LM,B,N,MM,NM,C,IM,JM)
         COMPLEX*16 A,B,C
         DIMENSION A(KM,LM),B(MM,NM),C(IM,JM)

                  DO 10 I=1,K
                     DO 10 J=1,N
                        C(I,J)=(0.,0.)
                        DO 10 LPP=1,L
                           C(I,J)=C(I,J)+A(I,LPP)*B(LPP,J)
 10               CONTINUE
         RETURN
         END

C  THE FOLLOWING SUBROUTINE IS USED TO MULTIPLY A VECTOR BY A MATRIX
C  C = A * B IS SOLVED, WHERE THE VECTOR A IS 1 x L, THE MATRIX B IS
C  L x N, AND THE VECTOR C IS 1 x N.

         SUBROUTINE VTMTMUL(A,L,LM,B,N,MM,NM,C,JM)
         COMPLEX*16 A,B,C
         DIMENSION A(LM),B(MM,NM),C(JM)

         DO 20 I=1,N
                  C(I) = (0.,0.)
                  DO 10 J=1,L
                     C(I) = C(I)+A(J)*B(J,I)
 10      CONTINUE
 20      CONTINUE
         RETURN
         END

C  THE FOLLOWING SUBROUTINE IS USED TO MULTIPLY A MATRIX BY A VECTOR.
C  C = A*B IS SOLVED, WHERE THE MATRIX A IS L x M, THE VECTOR B IS M x 1,
c  AND THE VECTOR C IS L x 1.

         SUBROUTINE MTVTMUL(A,L,M,LM,MM,B,JM,C,IM)
         COMPLEX*16 A,B,C
         DIMENSION A(LM,MM),B(JM),C(IM)
```

```
        DO 10 I=1,L
                C(I) = (0.,0.)
                DO 10 J=1,M
                  C(I) = C(I)+A(I,J)*B(J)
10      CONTINUE
        RETURN
        END
```